Discrete Optimization

Parametric mixed-integer 0–1 linear programming: The general case for a single parameter

Alexander Mitsos, Paul I. Barton *

Department of Chemical Engineering, Massachusetts Institute of Technology, 66-464, 77 Massachusetts Avenue, Cambridge, MA 02139, United States

Received 3 February 2007; accepted 8 January 2008
Available online 18 January 2008

Abstract

Two algorithms for the general case of parametric mixed-integer linear programs (MILPs) are proposed. Parametric MILPs are considered in which a single parameter can simultaneously influence the objective function, the right-hand side and the matrix. The first algorithm is based on branch-and-bound on the integer variables, solving a parametric linear program (LP) at each node. The second algorithm is based on the optimality range of a qualitatively invariant solution, decomposing the parametric optimization problem into a series of regular MILPs, parametric LPs and regular mixed-integer nonlinear programs (MINLPs). The number of subproblems required for a particular instance is equal to the number of critical regions. For the parametric LPs an improvement of the well-known rational simplex algorithm is presented, that requires less consecutive operations on rational functions. Also, an alternative based on predictor–corrector continuation is proposed. Numerical results for a test set are discussed.

© 2008 Elsevier B.V. All rights reserved.

Keywords: Parametric programming; Post-optimality sensitivity analysis; Matrix case; MILP; MINLP

1. Introduction

Mathematical programs often involve unknown parameters and the task of parametric optimization is, in principle, to solve the mathematical program for each possible value of these unknown parameters. Discretization of the parameter range is not rigorous in general, since there is no guarantee for optimality between the mesh points. Moreover, discretization on a fine mesh is a very expensive procedure, especially for high dimension parameter spaces. Algorithms for parametric optimization typically divide the parameter range into regions of optimality, also called areas [1], or critical regions [2]. For one parameter the boundary between critical regions is called a breakpoint. For each critical region either the problem is infeasible or a qualitatively invariant solution, typically a smooth function of the parameters, is optimal. The notion of qualitatively invariant solution depends on the specific case. In parametric mixed-integer linear programs (MILPs), the topic of this paper, it means an optimal integer realization along with an optimal basis for the linear program (LP) resulting with this integer realization.

Parametric optimization has several applications [2] including waste management [3], fleet planning [4], model-predictive control [5] and process synthesis under uncertainty [6–8]. Recently, Balas and Saxena [9] used good feasible solutions to parametric MILPs to generate cuts for MILPs. Also recently, Eppstein [10] introduced the notion of inverse parametric optimization where the values of parameters that result in a given solution are searched for. Wallace [11] has argued that

* Corresponding author. Tel.: +1 617 253 6526; fax: +1 617 258 5042.

E-mail addresses: amitsos@alum.mit.edu (A. Mitsos), pib@mit.edu (P.I. Barton).
Parametric optimization is valuable for decision making only when the value of the parameters is not known during the optimization phase but known during the decision making phase.

In this paper the general case of parametric MILPs is considered

\[
    f^*(p) = \min_{x,y} (c^e(p))^T x + (c^l(p))^T y \\
    \text{s.t. } A^{l_1}(p)x + A^{l_2}(p)y = b^l(p), \\
    A^{v_1}(p)x + A^{v_2}(p)y \leq b^v(p), \\
    x_L \leq x \leq x_U, \\
    x \in \mathbb{R}^n_x, y \in \{0,1\}^m,
\]

where \( c^e(p) \in \mathbb{R}^m_n, c^l(p) \in \mathbb{R}^m_n, A^{l_1}(p) \in \mathbb{R}^{m_1 \times n_n}, A^{l_2}(p) \in \mathbb{R}^{m_2 \times n_n}, A^{v_1}(p) \in \mathbb{R}^{m_1 \times n_n}, A^{v_2}(p) \in \mathbb{R}^{m_2 \times n_n}, b^l(p) \in \mathbb{R}^{m_1}, b^v(p) \in \mathbb{R}^{m_2}. \)

Note that in deviation from the standard form finite upper and nonzero lower bounds on the variables as well as inequality constraints are allowed, to show how these can be treated efficiently. Note that in some cases the discussion is restricted to finite bounds \( x^c, x^U \in \mathbb{R}^n_x, \) while in other cases infinite bounds are allowed. As is done in most algorithmic contributions, the host set of the parameter \( P \) is assumed to be the interval \([0,1]\). This is essentially equivalent to the assumption of a compact host set, excluding unbounded parameter ranges.

Two interesting special cases of (1), that are often considered in the literature, are the cost-vector case and the right-hand side case. In both of these cases the matrices \( A^{l_1}, A^{v_1}, A^{l_2}, A^{v_2} \) do not depend on the parameter \( p \); in the former case also the right-hand side vectors \( b^l, b^v \) are parameter independent, while in the latter case the cost-vectors \( c^e, c^l \) are parameter independent.

Cost-
vectors have the benign property that the feasibility region does not depend on the parameter. As a consequence the optimality regions of a given basis are (convex) polyhedra and the optimal solution function is piecewise-affine and concave [12]. In the right-hand side case the optimality region of a given basis can be calculated relatively easily [13,14].

When the integer variables in (1) are fixed (to 0 or 1), or relaxed (to \([0,1]\)), a parametric LP is obtained, which is an important problem in its own right, and can also be used as a subproblem for the solution of (1).

Parametric optimization is a mature field. Most of the theoretical properties were established by the 1980’s and in recent years the focus has been on algorithmic contributions, which is also the focus of this paper. There are several textbooks and review articles on parametric optimization; Dinkelbach [15] and Gal [2] consider LP; Bank et al. [16] and Fiacco [17] consider nonlinear parametric programs. Algorithms for the right-hand side case with an affine dependence on one or many parameters exist for MILPs [3,7,13,18,20–22] and also for (mixed-integer) nonlinear programs [23–25]. For the cost-vector case of MILPs with an affine dependence on a single parameter a well-known algorithm is based on intersections of the objective functions of feasible points [3,26,27]. Note also that in principle (1) can be reformulated to a right-hand side problem [23,25] by introducing an auxiliary variable \( z \) and the constraint \( z = p \); however in general this leads to a nonconvex parametric MINLP.

Based on the possible number of optimality regions, Murty [28] shows that the complexity of parametric optimization cannot be bounded above by a polynomial even in the right-hand side case of parametric LPs with a single parameter. Therefore, rather than basing the computational complexity on the size of the instance, it is probably more appropriate to compare the computational requirement of an algorithm with the computational requirement of solving as many regular optimization problems (at fixed parameter values) as there are optimality regions. For instance, in the cost-vector case of MILP with a single parameter the intersection-based algorithm [3,26,27] requires a number of MILP calls that is less than twice the number of optimality regions of the particular instance.

The algorithmic approaches for parametric MILP can be divided into two broad classes. In the first class, algorithms for the solution of a regular MILP are altered to solve the parametric MILP. For instance Ohtake and Nishida [20] solve the right-hand side case of parametric MILP by a branch-and-bound (B&\text{B}) on the integer variables with a parametric LP at each node. Methods based on this principle have the promise of being relatively computationally efficient if the formulated parametric subproblems are only slightly more expensive than their regular counterparts. The other broad class is to use MILP calls for fixed parameter values and process the result post-optimally. This is, for instance, employed in the well-known intersection-based algorithm for the cost-vector case. Methods based on this principle can take advantage of state-of-the-art MILP solvers and are also relatively easy to implement.

To our best knowledge no algorithm exists for the solution of the general case of parametric MILPs, apart from our conference presentations [29,30], and extension of the available algorithms for the right-hand side and cost-vector case is nontrivial because the general case does not have the benign properties of these special cases. The most relevant contributions are on parametric LPs. Post-optimal sensitivity analysis of the matrix coefficients of nonbasic columns is covered in linear programming textbooks, e.g., [31]. Freund [32] proposes to obtain post-optimal sensitivity information through Taylor series expansions and Greenberg [33] considers post-optimal sensitivity analysis from interior solutions via duality. Gal [2] reviews the case that a single column or a single row of the matrix depends on the parameter; in this case an analytical inversion of the parametric matrix is possible based on a formula by Bodewig [34]. Dinkelbach [15] proposes an
algorithm based on an extension of the simplex method from real valued coefficients to rational functions of the parameters. We propose two algorithms for the general case of parametric LP, based on the algorithm by Dinkelbach [15]. For implementation purposes, we allow a violation of the primal constraints and marginal values by a positive tolerance. The proposed algorithms rely on the availability of simple and exact optimality conditions, which is not the case for MILP. To overcome this limitation, we consider and discuss generalizations of a formulation by Pertsinidis et al. [7,13] and propose an algorithm for parametric MILPs based on decomposition into a series of regular MILPs, parametric LPs and regular mixed-integer nonlinear programs (MINLPs). We also propose an algorithm based on branch-and-bound on the integer variables solving a parametric LP at each node.

Section 2 presents the assumptions necessary for the algorithms presented as well as discussing some complicating properties. In Section 3 a variant of the algorithm by Dinkelbach for parametric LPs is proposed. In Section 4 the optimality region formulation by Pertsinidis is extended to general parametric dependence. In Section 5 the algorithms for parametric MILPs are presented. Section 6 discusses a prototype implementation along with numerical results, followed by a discussion of the results and potential future work in Section 7. Finally the appendix contains a description of the small-scale examples.

2. Assumptions and theoretical properties

Throughout the paper unbounded optimization problems are excluded:

**Assumption 1 (Bounded problems).** For any parameter value \( p \in [0, 1] \) the LP-relaxation of (1) is bounded from below, i.e., it is either infeasible or has a finite optimal objective value.

As a direct consequence of **Assumption 1** the parametric MILP (1) is bounded from below for all \( p \in [0, 1] \).

To ensure the applicability of the algorithm by Dinkelbach [15] for parametric LPs, we further limit the functional dependence of the data on the parameter, a limitation not needed for all algorithms presented.

**Assumption 2 (Data are rational functions of the parameter).** The data (matrix, cost vector, right-hand side vector) are continuous rational functions of the parameter \( p \in [0, 1] \), i.e., quotients of polynomial functions with nonzero denominator.

Under **Assumptions 1 and 2** the parameter set \( P \) can be divided into a finite number of intervals and/or segments, such that for each interval/segment the problem is either infeasible or the optimal solution is also a rational function in the parameter \( p \) corresponding to a constant basis. This is a direct consequence of the finite number of integer realizations and bases and that for each parameter value there exists an optimal basis. Some optimality intervals are degenerate (singletons), but there are also others of nonzero length. The marginal costs \( c \) in the LP case are also rational functions of the parameter. No convexity or concavity properties can be shown for the optimal objective value as a function of the parameter.

As **Example 2.1** demonstrates, at changes of optimal integer realization a discontinuity may be observed.

**Example 2.1 (Integer variables can lead to discontinuity).** The parametric LP

\[
\begin{align*}
\min_x & \quad y \\
\text{s.t.} & \quad y \geq p - 1/2, \\
& \quad y \in \{0, 1\}
\end{align*}
\]

is not continuous at \( p = 1/2 \). Indeed, for \( p \leq 1/2 \) the constraint \( y \geq p - 1/2 \) is redundant and we obtain the unique optimal solution \( y = 0 \) and the optimal objective function 0. For \( p > 1/2 \) the constraint \( y \geq p - 1/2 \) implies \( y = 1 \) and the optimal solution is \( y = 1 \) with an objective value of 1.

A more subtle source of discontinuity is linear dependence of equations, as **Example 2.2** shows.

**Example 2.2 (Linear dependence for a single parameter value can lead to discontinuity).** The parametric LP

\[
\begin{align*}
\min_x & \quad -x \\
\text{s.t.} & \quad (p - 1/2)x = 0, \\
& \quad x \in [0, 1]
\end{align*}
\]

is not continuous at \( p = 1/2 \). Indeed, for \( p \neq 1/2 \) we obtain the unique, parameter-independent optimal solution \( x(p) = 0 \) and the optimal objective function 0. For \( p = 1/2 \) there are infinitely many feasible points \( x \in [0, 1] \) and the optimal solution is \( x(1/2) = 1 \) with an objective value of -1.
At the discontinuity of Example 2.2 an equation becomes redundant and this causes a removable point discontinuity. The point-to-set mapping from the parameter set to the feasible set is not closed. A discussion of point-to-set mappings is outside the scope of this paper and the reader is referred to Bank et al. [16] for a discussion of the implications for parametric optimization. Note only that for all parameter values for which (1) is feasible, the optimal objective value is lower semi-continuous. As described in the following section, to deal with linearly dependent equations we introduce surplus and slack variables and then make an assumption on the augmented systems.

3. Parametric linear program

As mentioned in Section 1, an important subproblem of parametric MILP is the parametric LP

$$\min_x (c(p))^T x$$

s.t. $$A^1(p)x = b^1(p),$$
$$A^2(p)x \leq b^2(p),$$
$$x \in \mathbb{R}^n, \quad x^L \leq x \leq x^U,$$

where similarly to (1) the data $$c(p) \in \mathbb{R}^n, A^1(p) \in \mathbb{R}^{m_1 \times n}, A^2(p) \in \mathbb{R}^{m_2 \times n}, b^1(p) \in \mathbb{R}^{m_1},$$ and $$b^2(p) \in \mathbb{R}^{m_2}$$ are assumed to be continuous rational functions of the parameter $$p \in [p_l, p_u].$$ We again deviate from the standard form to show how inequality constraints and bounds on the variables can be treated efficiently.

Algorithm 0 gives a high-level outline of our proposal for the solution of (2) and then two alternatives for the algorithmic subproblems are presented in detail. The first is based on operations with rational functions. As discussed later on, there are significant problems with error propagation of rational operations, particularly for increasing problem size. This motivates our alternative proposal based on continuation. Finally, we provide a termination statement.

Algorithm 0 is inspired by the algorithm by Dinkelbach [15], that essentially extends the full tableau implementation of the simplex method to the parametric case. As such, Dinkelbach’s algorithm is cumbersome to implement, cannot take advantage of state-of-the-art LP solvers and requires a large number of operations with rational functions. In Algorithm 0 the LP (2) is instead solved at the breakpoints (for fixed parameter values) with a regular LP solver. Then, assuming a feasible problem, the algorithm moves to the following breakpoint by only considering the basis matrix along with the feasibility and optimality conditions. In the case that the LP is infeasible this implies that there is a parameter region for which the parametric LP is infeasible and this region is identified by the formulation of an auxiliary problem, based on phase I of the simplex method and described in detail in Section 3.4.

In addition to taking advantage of state-of-the-art LP solvers our proposal is less vulnerable to error propagation than the original algorithm by Dinkelbach because the worst-case number of consecutive operations with rational functions is much smaller. Our proposal only requires the factorization of a square matrix of size $$n \times n$$ and therefore at most of the order of $$(m_1 + m_2)^2$$ operations with rational functions. On the other hand, no heuristics are known that guarantee a polynomial number of iterations for the simplex method, and as a consequence for Dinkelbach’s algorithm. For large problems our proposal is therefore expected to be significantly more robust than the algorithm by Dinkelbach, since the latter operates on the entire matrices ($$A^1(p)$$ and $$A^2(p)$$) and error propagates throughout the iterations. Our proposal is particularly suitable for LPs with many more variables than constraints $$n \gg m_1 + m_2.$$

Algorithm 0 (Parametric linear program). Input to the algorithm are the problem data, tolerances for violation of the primal inequalities $$\varepsilon_{\text{inf}}$$ and the marginal cost inequalities $$\varepsilon_{\text{opt}},$$ and a guess for the minimal parameter step $$\delta p > 0.$$ The algorithm uses a set $$R$$ to store the optimal solutions. The elements $$R_i$$ of $$R$$ are quadruplets, composed of a parameter value $$p_i^R,$$ a boolean $$g_i^R,$$ describing whether the problem is feasible for this element ($$g_i^R = \text{true}$$) or not ($$g_i^R = \text{false}$$), a point $$x_i^R(p),$$ and the corresponding objective function $$f_i^R(p).$$

1. Initialize with $$p_i = p_l.$$  
2. REPEAT  
   (a) Solve LP (2) for $$p = p_i + \delta p$$ and obtain an optimal basis.  
   (b) IF feasible THEN  
      • Set $$g = \text{true}.$$  
      ELSE  
      • Set $$g = \text{false}.$$  
      • Set $$f(p) = +\infty.$$  
      • Solve the phase I problem (7) for $$p = \bar{p} = p_i + \delta p$$ and obtain an optimal basis.  
   (c) Set up the parametric system of equations and inequalities.  
   (d) Obtain the parametric dependence of the solution $$x(p),$$ $$f(p)$$ for $$p \in [p_l, p_u].$$
The parametric system of equations.

In Algorithm 0, the LP solver is called for (2) whereas for Subroutine 1 the augmented problem (3) is used for either their lower basic at a nonzero value. If, on the other hand, the equation is redundant, the corresponding surplus variable is basic but at

demand form in which nonbasic variables have zero value. If an equation is infeasible the corresponding surplus variable is infeasible for \( p \in [p_L, p_u] \). We propose an alternative based on the numerical solution of a set of equations by continuation discussion we assume for simplicity that this has been done. Note also that between calls to the LP solver, it typically is advan-

ted, e.g., LU factorization followed by back substitutions, with symbolic operations for the

This inversion can be, at least in principle, performed by elementary row operations on the basis matrix \( B \), e.g., LU factorization followed by back substitutions, with symbolic operations for the matrix elements. If all data are rational functions of the parameter \( p \), so is the candidate optimal solution. Based on this, Dinkelbach [15] proposed an extension of the simplex algorithm to rational functions. This approach does not scale to large problems because of exploding numerical error [35–38]. Note that the use of high or infinite precision arithmetic is prohibitively expensive. We propose an alternative based on the numerical solution of a set of equations by continuation along with detection of violation of primal inequalities and marginal cost inequalities.

A complication with the use of state-of-the-art solvers such as CPLEX [39] is that they typically introduce auxiliary variables, i.e., surplus and slack variables for equalities and inequalities, respectively. One of the reasons for this is that LPs often have linearly dependent equality constraints (redundant or infeasible). At termination, the solvers furnish a set of \( m_1 + m_2 \) basic variables, some of which are original and some are surplus/slack variables. The nonbasic variables are at

either their lower \( (x_i = x_i^L) \) or upper bound \( (x_i = x_i^U) \); note that this is a straightforward extension compared to LPs in standard form in which nonbasic variables have zero value. If an equation is infeasible the corresponding surplus variable is basic at a nonzero value. If, on the other hand, the equation is redundant, the corresponding surplus variable is basic but at zero level. Note first that (2) is equivalent to

\[
\min_{x,s} \quad (c(p))^T x
\]

s.t. \[
A^1(p)x + I^1s = b^1(p),
A^2(p)x + I^2t = b^2(p),
x \in \mathbb{R}^{n}, \quad x^L \leq x \leq x^U,
s \in \mathbb{R}^{m}, \quad s = 0,
t \in \mathbb{R}^{m}, \quad 0 \leq t.
\]

where \( I^1 \) and \( I^2 \) are identity matrices of size \( m_1 \) and \( m_2 \) respectively. Clearly an optimal solution to (2) is optimal in (3) and vice-versa. In Algorithm 0, the LP solver is called for (2) whereas for Subroutine 1 the augmented problem (3) is used for the parametric system of equations.
Suppose that at termination the LP solver returns two integer vectors \( \mathbf{d}^* \in \{0,1,2\}^n \) and \( \mathbf{d}' \in \{0,1\}^{m_1+m_2} \). The \( j \)th component of \( \mathbf{d}' \) indicates if variable \( j \) is at its lower bound \( (d'_j = 0) \), basic \( (d'_j = 1) \), or at its upper bound \( (d'_j = 2) \). The \( j \)th component of \( \mathbf{d}^* \) indicates if auxiliary variable \( j \) is basic \( (d^*_j = 1) \) or nonbasic \( (d^*_j = 0) \). Subroutine 1 describes how to set up the square system of equations storing a parameter dependent matrix \( \mathbf{B}(p) \in \mathbb{R}^{(m_1+m_2) \times (m_1+m_2)} \) and a parameter dependent right-hand side vector \( \mathbf{b}(p) \). Solving the system \( \mathbf{B}(p)\mathbf{x}^B = \mathbf{b}(p) \) as a function of the parameter gives the desired functional dependence. For later use also the bounds of the basic variables are stored in \( \mathbf{x}^{B_L} \) and \( \mathbf{x}^{B_U} \) and the cost coefficients of the basic variables are stored in \( \mathbf{c}^B(p) \).

Subroutine 1 (Setting up parametric system of equations). The subroutine uses counters \( i, j \) and stores the number of basic variables in \( n_b \).

1. Set \( b_i(p) = b^*_i(p) \) for \( i = 1, \ldots, m_1 \).
2. Set \( b_{i+m}(p) = b^*_i(p) \) for \( i = 1, \ldots, m_2 \).
3. Set \( n_b = 0 \).
4. FOR \( i = 1, \ldots, n_c \) DO
   - IF \( d^*_i = 1 \) THEN
     - \( n_b = n_b + 1 \).
     - Set \( B_{j,n_b}(p) = A^*_j(p) \) for \( j = 1, \ldots, m_1 \).
     - Set \( B_{j+m_1:n_b}(p) = A^*_j(p) \) for \( j = 1, \ldots, m_2 \).
     - Set \( c^b_{i,n_b}(p) = c_i(p) \).
     - Set \( x^b_{n_b} = x^*_i \).
   - ELSE IF \( d^*_i = 0 \) THEN
     - Set \( b_j(p) = b^*_j(p) - A^*_j(p)x^*_i \) for \( j = 1, \ldots, m_1 \).
     - Set \( b_{j+m}(p) = b^*_j(p) - A^*_j(p)x^*_i \) for \( j = 1, \ldots, m_2 \).
   - ELSE
     - Set \( b_j(p) = b^*_j(p) - A^*_j(p)x^*_i \) for \( j = 1, \ldots, m_1 \).
     - Set \( b_{j+m}(p) = b^*_j(p) - A^*_j(p)x^*_i \) for \( j = 1, \ldots, m_2 \).
   END
5. FOR \( i = 1, \ldots, m_1 + m_2 \) DO
   - IF \( d^*_i = 1 \) THEN
     - \( n_b = n_b + 1 \).
     - Set \( B_{j,n_b}(p) = 1 \).
     - Set \( B_{j+m}(p) = 0 \) for \( j = 1, \ldots, m_1 + m_2 : j \neq i \).
     - Set \( x^b_{n_b} = 0 \).
     - IF \( i \leq m_1 \) THEN Set \( x^u_{n_b} = 0 \) ELSE \( x^u_{n_b} = +\infty \).
     - Set \( c^b_{i,n_b}(p) = 0 \).
   END

As demonstrated in Example 2.2 there is an issue when \( \mathbf{B}(p) \) becomes singular while remaining feasible at some parameter values, because it is no longer a basis and the primal and marginal constraints do not guarantee optimality. To exclude cases like this we make an additional assumption:

Assumption 3. If an augmented basis \( \mathbf{B}(p) \) is optimal for some \( p \) and singular for some other parameter value \( \tilde{p} \in P \) then there exists \( \delta > 0 \), such that for \( \forall p \in P : |\tilde{p} - p| \leq \delta \) the system \( \mathbf{B}(p)\mathbf{x}^B = \mathbf{b}(p) \) does not have any solutions satisfying the primal constraints \( \mathbf{x}^B \in [\mathbf{x}^{B_L}, \mathbf{x}^{B_U}] \).

Note at this point that Assumption 3 is only required for one of the two alternatives we present for the subproblems of Algorithm 0, namely continuation. If the assumption is violated the only consequence is that for the parameter values with singularity, a suboptimal solution will be furnished by our algorithm. Note also that for variables with finite bounds Assumption 3 can be derived from the weaker statement “if an augmented basis \( \mathbf{B}(p) \) is optimal for some \( \tilde{p} \) and singular for some other parameter value \( \tilde{p} \in P \) then the system \( \mathbf{B}(\tilde{p})\mathbf{x}^B = \mathbf{b}(\tilde{p}) \) has no solution”.

3.1.1. Solution with rational operations

The algorithm by Dinkelbach [15] solves the parametric LP directly using rational operations. As a consequence at each iteration \( (\mathbf{B}(p))^{-1}\mathbf{b}(p) \) is available. We propose instead to take the matrix and perform an LU factorization with rational operations. We assume the existence of an LU-factorization algorithm of the rational matrix \( \mathbf{B}(p) \) obtaining a permutation
matrix $\mathbf{M}$, with elements independent of $p$, a lower-triangular matrix $\mathbf{L}(p)$, with all diagonal entries equal to 1 and an upper-triangular matrix $\mathbf{U}(p)$ satisfying

$$\mathbf{M} \mathbf{b}(p) = \mathbf{L}(p) \mathbf{U}(p).$$

This method allows, at least in principle, to check directly if the matrix becomes singular for some parameter point, by first calculating the magnitude of the determinant

$$|\text{det}(\mathbf{B}(p))| = \Pi_{i=1}^{m_1+m_2}|\mathbf{U}_{i,i}(p)|$$

and then calculating the first (if any exists) parameter value for which $|\text{det}(\mathbf{B}(p))| = 0$. So in principle Assumption 3 is not needed if rational operations are performed. Note that for the original algorithm by Dinkelbach it is required.

Once the range for which the basis is invertible has been identified, the system $\mathbf{B}(p)\mathbf{x}^B = \mathbf{b}(p)$ can be solved in two steps. First forward elimination

$$\mathbf{L}(p)\mathbf{v} = \mathbf{M}\mathbf{b}(p)$$

is performed giving the temporary vector $\mathbf{v}(p)$ as a function of the parameter. Then by back substitution

$$\mathbf{U}(p)\mathbf{x}_B = \mathbf{v}(p),$$

the dependence $\mathbf{x}_B(p)$ is obtained.

**Remark 3.1 (Ordering of matrix before LU-factorization).** Since the surplus and slack variables have only one nonzero entry, it is more efficient to reorder the variables and rows, putting the artificial variables in the upper left corner, prior to performing the factorization. For simplicity we did not describe this in Subroutine 1.

### 3.1.2. Solution with continuation

A problem with the matrix inversion with rational operations, is that either multiprecision arithmetic has to be used, which is prohibitively expensive, or the numerical error explodes with increasing system size. We therefore propose an alternative approach based on continuation. This method avoids operations with rational functions and therefore scales to large systems. Note that even the calculation of the right-hand side in Subroutine 1 can be done within the continuation method. An additional advantage is that Assumption 2 is not required. On the other hand, a disadvantage over the solution with rational operations is that Assumption 3 is required. Recall also that Nazareth [40] considers a variation of the simplex method based on continuation for parametric linear programs, where the parameter affects only the cost vector and right-hand side.

We here give a brief introduction to continuation. The reader is referred to [41,42] for a thorough discussion. The general idea of continuation is to follow an implicitly defined curve

$$f(z, \lambda) = 0,$$

where $f: \mathbb{R}^{n+1} \rightarrow \mathbb{R}^n$ is typically assumed smooth. Predictor–corrector methods trace the curve by generating a sequence of points $(z', x')$ that are on the curve within a given tolerance ($\|f(z, \lambda)\| \leq \varepsilon$). At a given iteration a step is taken from $z'^{-1}$ to $z'$ and a predictor $\tilde{z}'$ is calculated, e.g., by a polynomial spline approximation. The correction is typically performed by a Newton solver for the solution of $f(z', \tilde{z}') = 0$ with $\tilde{z}'$ as the initial guess. The step size $\lambda' - \lambda'^{-1}$ depends on the quality of approximation of the previous step(s).

Our proposal should not be confused with homotopy continuation, where only the final point (for $\lambda = 1$) of the curve is needed. We are interested in actually obtaining an estimate to the solution of

$$\mathbf{B}(p)\mathbf{x}_B = \mathbf{b}(p)$$

for a range of parameter values and we use the predictor polynomials as an approximation to the solution. Therefore estimates are needed for the predictor quality at all generated points as well as for the points in-between. While it is possible to have rigorous guarantees for $\|f(z, \lambda)\|$, see, e.g., Neubert [43], most methods rely on error estimates [44,45]. Note that similar error estimates are successfully applied to the solution of differential-algebraic equation systems [46], which suggests that our proposal will be successful in practice.

In the application considered here, for a fixed parameter value a linear system of equations has to be solved. Therefore the corrector step can be performed by any linear solver (direct or iterative). Assuming that the predictor is a good approximation, it may be beneficial to use an iterative method, especially for large systems.

Because no direct inversion of the basis $\mathbf{B}(p)$ is performed, the algorithm does not necessarily detect parameter values for which the matrix is singular. Formulations that detect singularity are possible, but they are overly expensive. Another possibility would be to detect the singularity within the continuation code which is also possible, but for reasons of efficiency
the LU factors are typically not available. This motivates Assumption 3, which guarantees that a primal constraint would be violated before the matrix becomes singular.

3.2. Feasibility range

The feasibility range of a given basis (or \( x^B(p) \)) is implicitly defined by
\[
\begin{align*}
B(p)x^B &= b(p), \\
x^{B,L} &\leq x^B \leq x^{B,U}. 
\end{align*}
\]

In general this feasibility range is a disjoint set. In Algorithm 0 feasibility for a parameter point \( p \) has been established and the smallest parameter value \( p_t \) for which (5) is violated by a prespecified tolerance \( \varepsilon_{inf} \) is required. In the cost-vector case this step is not needed, since feasibility does not depend on the parameter. In the right-hand side case with affine dependence on the parameter, \( x^B \) is an affine function of the parameter and only a set of linear inequalities needs to be checked.

3.2.1. Solution with rational operations

Recall that with the use of forward elimination and back substitution we have obtained the functional dependence of \( x^B \) on the parameter. Subroutine 2 determines the smallest parameter value \( p_t \) for which (5) is violated by a prespecified tolerance \( \varepsilon_{inf} \).

Subroutine 2 (Obtain feasibility range).

1. Set \( p_t = p_n \).
2. FOR \( i = 1, \ldots, m_1 + m_2 \) DO
   - IF \( x^{B,L}_i > -\infty \) THEN set \( p_t \) equal to the smallest root of \( x^B_i(p) = x^{B,L}_i - \varepsilon_{inf} \) for \( p \in [p_l, p_l] \).
   - IF \( x^{U,B}_i < +\infty \) THEN set \( p_t \) equal to the smallest root of \( x^B_i(p) = x^{B,U}_i + \varepsilon_{inf} \) for \( p \in [p_l, p_l] \).
END

Note that if a rational does not have a root in \( [p_l, p_l] \), \( p_t \) is left unchanged.

3.2.2. Solution with continuation

In addition to tracking the solution of (4) we propose to identify \( \varepsilon_{inf} \)-violation of the primal constraints \( x^{B,L} \leq x^B \leq x^{B,U} \) by an event detection code, such as the one proposed by Park and Barton [47] for hybrid differential-algebraic equation systems. This algorithm first identifies an event by solving the interpolating polynomials within a continuation step and then accurately locates the event.

3.3. Optimality range

In LPs explicit optimality conditions are available [31] and can be used to identify the range of optimality. Recall the meaning of the basis matrix \( B(p) \), cost vector of the basic solutions \( c^B(p) \) and the integer vector \( d^e \) from Subroutine 1. The following system implicitly defines the range of optimality of a basis
\[
\begin{align*}
(B(p))^Tz &= c^B(p), \\
\sum_{i=1}^{m_1} z_i A^{i,j}_i(p) - \sum_{i=1}^{m_2} z_i A^e_{i+m_1,j}(p) &\geq 0 \quad \forall j : d^e_j = 0, \\
\sum_{i=1}^{m_1} z_i A^{i,j}_i(p) - \sum_{i=1}^{m_1} z_i A^e_{i+m_1,j}(p) &\leq 0 \quad \forall j : d^e_j = 2, \\
z_j &\leq 0 \quad \forall j \leq m_1 : d^e_j = 0,
\end{align*}
\]

where \( z \in \mathbb{R}^{m_1+m_2} \) is a vector of auxiliary variables. The first and second inequalities calculate the marginal cost of the nonbasic original variables. The inequality \( z_j \leq 0 \) calculates the marginal cost of slack variables for nonbasic inequalities, since the cost of slack variables is equal to zero and there is only one entry for slack variables in the matrix. Note that the marginal cost of surplus variables (for nonbasic equations) need not be calculated, since surplus variables only enter the basis to overcome infeasibility.

Remark 3.2 (Elimination of auxiliary variables). Since the cost of surplus variables is set equal to zero and the columns corresponding to the surplus variables are equal to the identity vector we can eliminate the \( z \)-variables corresponding to the basic surplus variables from (6). These steps are omitted for the sake of simplicity.
Similarly to the feasibility range, in general (6) defines a disjoint set for the parameter. In Algorithm 0 we have established the optimality conditions for a parameter value $p_s$ and we want to find the smallest parameter value $p$, for which (6) is violated by a prespecified tolerance $\varepsilon_{opt}$. In the right-hand side case this step is not needed, since the marginal costs do not depend on the parameter. In the cost-vector case with affine dependence on the parameter, a set of linear inequalities needs to be checked.

### 3.3.1. Solution with rational operations

Recall that we have performed an LU factorization of the matrix $B(p)$. We can use this factorization to first calculate the auxiliary variables $z$ and then the marginal costs, as shown in Subroutine 3.

**Subroutine 3 (Obtain optimality range).**

1. Solve $U^T(p)u = c_u(p)$ for $u$ by forward elimination.
2. Solve $L^T(p)v = u(p)$ for $v$ by back-substitution.
3. Calculate $z(p) = M^T u(p)$.
4. FOR $j = 1, \ldots, m_1$
   - IF $d_j = 0$ THEN Set $p_j$ equal to the first root of $z_j(p) = -\varepsilon_{opt}$ in $p \in [p_s, p_t]$. END
5. FOR $j = 1, \ldots, n$, $d_j \neq 1$
   - Set: $t(p) = c_i(p) - \sum_{i=1}^{m_1} z_i(p) A^2_{i,j}(p) - \sum_{i=1}^{m_2} z_i(p) A^2_{i,j}(p)$.
   - IF $d_j = 0$ THEN Set $p_j$ equal to the first root of $t(p) = -\varepsilon_{opt}$ in $p \in [p_s, p_t]$ ELSE Set $p_j$ equal to the first root of $t(p) = \varepsilon_{opt}$ in $p \in [p_s, p_t]$. END

Note that if a rational does not have a root in $[p_s, p_t]$, $p_t$ is left unchanged.

### 3.3.2. Solution with continuation

We propose to identify violation of the marginal cost inequalities in a similar way as the violation of primal inequalities, i.e., solve

$$(B(p))^T z = c^d(p)$$

as a function of $p$, using a continuation method until one of the following inequalities is violated

$$c_j(p) - \sum_{i=1}^{m_1} z_i A^2_{i,j}(p) - \sum_{i=1}^{m_2} z_i A^2_{i,j}(p) \geq 0 \quad \forall j : d_j = 0,$$

$$c_j(p) - \sum_{i=1}^{m_1} z_i A^2_{i,j}(p) - \sum_{i=1}^{m_2} z_i A^2_{i,j}(p) \leq 0 \quad \forall j : d_j = 2,$$

$$z_j \leq 0 \quad \forall j \leq m_1 : d_j = 0$$

by a value equal to $\varepsilon_{opt}$.

### 3.4. Infeasibility range

Suppose that for some parameter value $\bar{p}$ the parametric LP (2) is infeasible. To determine the range of infeasibility we consider an auxiliary problem, similar to the phase I of the simplex method [31], where surplus variables are introduced for both equality and inequality constraints.

$$\min_{x, s} \sum_{i=1}^{m_1} s_i$$

s.t. $$\sum_{j=1}^{n} A^1_{i,j}(p)x_j + \text{sign} \left( b^1_i(\bar{p}) - \sum_{j=1}^{n} A^1_{i,j}(\bar{p}) x_j \right) s_i = b^1_i(\bar{p})$$

$$- \sum_{j=1}^{n} A^2_{i,j}(p)x_j, \quad i = 1, \ldots, m_1,$$

$$\sum_{j=1}^{n} A^2_{i,j}(p)x_j - s_{i+m_1} \leq b^2_i(\bar{p}) - \sum_{j=1}^{n} A^2_{i,j}(\bar{p}) x_j, \quad i = 1, \ldots, m_2,$$

$$x \in \mathbb{R}^n, \quad 0 \leq x \leq x^U - x^L,$$

$$s \in \mathbb{R}^{m_1+m_2}, \quad 0 \leq s.$$
Note that in the equality constraints, the sign of the coefficient of the auxiliary variables is calculated at the fixed parameter value \( \hat{p} \) to ensure by construction that (7) is feasible for this parameter value. Before explaining how this auxiliary problem is used, we prove some properties required:

**Proposition 1** (Parametric phase I problem).

1. The auxiliary parametric LP (7) is feasible for \( p = \hat{p} \).
2. For any \( \hat{p} \) the parametric LP (2) is feasible if and only if the optimal solution value of the auxiliary parametric LP (7) is equal to zero.
3. For any \( \hat{p} \) the parametric LP (2) is infeasible if and only if the optimal solution value of the auxiliary parametric LP (7) is greater than zero.

**Proof.** We prove the three sub-statements in the order presented.

1. It suffices to show the feasibility of a pair \( (\bar{x}, \bar{s}) \). Pick \( \bar{x} = 0 \), \( \bar{s}_i = |b_i(\hat{p}) - \sum_{j=1}^{n_x} A_{ij}^1(\hat{p}) \bar{x}_j| \) for \( i \leq m_1 \) and \( \bar{s}_i = \max\{0, -b_i^2(\hat{p}) + \sum_{j=1}^{n_x} A_{ij}^2(\hat{p}) \bar{x}_j\} \) for \( i > m_1 \). Note first that this choice satisfies the variable bounds. The equality constraint \( i \) is equivalent to

\[
\text{sign}(b_i(\hat{p}) - \sum_{j=1}^{n_x} A_{ij}^1(\hat{p})) \left| b_i(\hat{p}) - \sum_{j=1}^{n_x} A_{ij}^1(\hat{p}) \bar{x}_j \right| = b_i(\hat{p}) - \sum_{j=1}^{n_x} A_{ij}^1(\hat{p}) \bar{x}_j,
\]

which is satisfied by the definition of sign and absolute value. The inequality constraint \( i \) is equivalent to

\[
-\max \left\{ 0, -b_i^2(\hat{p}) + \sum_{j=1}^{n_x} A_{ij}^2(\hat{p}) \bar{x}_j \right\} \leq b_i(\hat{p}) - \sum_{j=1}^{n_x} A_{ij}^1(\hat{p}) \bar{x}_j,
\]

which is satisfied by the definition of max.

2. Suppose first that for \( \hat{p} \) the optimal solution value of the auxiliary problem (7) is zero. We can therefore find a pair \( (\bar{x}, \bar{s}) \) which is feasible in (7) and satisfies \( \bar{s} = 0 \). By feasibility with respect to the bound variables we obtain \( 0 \leq \bar{x} \leq \bar{x}^U - \bar{x}^L \) or equivalently \( \bar{x}^L \leq \bar{x} \leq \bar{x}^U \), where \( \bar{x} = \bar{x}^L + \bar{x} \). The feasibility with respect to the equality constraints gives

\[
\sum_{j=1}^{n_x} A_{ij}^1(\hat{p}) \bar{x}_j + \text{sign} \left( b_i(\hat{p}) - \sum_{j=1}^{n_x} A_{ij}^1(\hat{p}) \bar{x}_j \right) 0 = b_i(\hat{p}) - \sum_{j=1}^{n_x} A_{ij}^1(\hat{p}) \bar{x}_j, \quad i = 1, \ldots, m_1,
\]

or

\[
\sum_{j=1}^{n_x} A_{ij}^1(\hat{p}) (\bar{x}_j + \bar{x}_j) = b_i(\hat{p}), \quad i = 1, \ldots, m_1.
\]

Similarly, the feasibility with respect to the inequality constraints gives

\[
\sum_{j=1}^{n_x} A_{ij}^2(\hat{p}) (\bar{x}_j + \bar{x}_j) \leq b_i(\hat{p}), \quad i = 1, \ldots, m_2.
\]

By the above relations it follows that \( \bar{x} \) is feasible in (2) for \( \hat{p} \).

3. Suppose now that for \( \hat{p} \) the original parametric LP (2) is feasible. We can therefore find \( \bar{x} \) which is feasible for \( \hat{p} \). Take the pair \( (\bar{x}, \bar{s}) \), where \( \bar{x} = \bar{x} - \bar{x}^L \) and \( \bar{s} = 0 \). Similarly to above feasibility of \( \bar{x} \) in (2) implies feasibility of \( (\bar{x}, \bar{s}) \) in (7) for \( \hat{p} \). Since by construction the optimal solution value of (7) is nonnegative, \( (\bar{x}, \bar{s}) \) is optimal for \( \hat{p} \), or (7) has an optimal solution value of zero for \( \hat{p} \).

3. Since by construction the optimal solution value of the auxiliary parametric LP is nonnegative the desired result follows directly.

Based on the properties described in Proposition 1 when (2) is found infeasible for some parameter value \( \hat{p} \) the auxiliary problem (7) is solved at \( \hat{p} \) and then the feasibility and optimality range are determined using this problem, in analogy to the previous sections. For parameter values for which the basis obtained for \( \hat{p} \) remains optimal for (7), the original problem (2) is infeasible. Note that if for some parameter value the current basis becomes infeasible or suboptimal, this does not directly imply feasibility of (2); rather, as indicated in Algorithm 0, when the current basis of the auxiliary problem becomes infeasible or suboptimal, the original problem needs to be resolved.
3.5. Algorithm termination

**Theorem 1.** Under Assumptions 1–3 Algorithm 0 terminates finitely. At termination R contains an approximate solution to the parametric LP. For \( g^R = \text{true} \), a basis of (2) and a corresponding point \( x^R(p) \) are given, that satisfy the primal and marginal cost constraints within \( e_{\text{inf}} \) and \( e_{\text{opt}} \)-tolerance respectively for \( p \in [p^R_1, p^R_1] \). For \( g^R = \text{false} \), infeasibility has been (approximately) established for \( p \in [p^R_1, p^R_1] \) by a basis of (7) that satisfies its primal and marginal cost constraints within \( e_{\text{inf}} \) and \( e_{\text{opt}} \)-tolerance respectively.

**Proof.** Assumption 1 ensures the existence of optimal solutions. Assumption 2 allows the use of LU factorization based on rational operations. Finally, Assumption 3 ensures that the bases remain invertible in the regions of interest and therefore the primal and marginal cost inequalities suffice for optimality. The primal and marginal cost constraints of (2) or (7) are satisfied by construction of the subproblems. It therefore suffices to demonstrate finite termination. We demonstrate this by showing the existence of a positive \( \delta p \) so that no backtracking is needed and that Algorithm 0 takes a positive step in the parameter interval \([p_1, p_u]\) at each iteration. For simplicity we consider (2) and (7) without distinction.

Suppose that (2) or (7) has been solved for \( p = p_i \) and an optimal basis has been obtained. By definition of a basis, for \( p = p_i \) the basis determinant is nonzero and the primal and marginal cost constraints are satisfied. Since the matrix elements are continuous, the determinant of the basis is also a continuous function of \( p \). We can thus determine a parameter range \( P^1 = [p_{11}, p_{1u}] \) with \( p_{11} < p_i < p_{1u} \) for which the determinant is nonzero, and therefore the basis is nonsingular and invertible. The basic variables and marginal costs are also rational functions of \( p \) and continuous within \( P^1 \). As a consequence we can find a parameter range \( P^2 = [p_{21}, p_{2u}] \) with

\[
P_{11} \leq p_{21} < p_i < p_{2u} \leq p_{1u}
\]

for which the basis does not violate the primal and marginal cost inequalities by more than the specified tolerances. Thus for the given basis we can find \( \delta p_{\text{min}} > 0 \) such that if the basis is optimal for \( p_0 \) it is also \( \epsilon \)-optimal in \([p_0 - \delta p_{\text{min}}, p_0 + \delta p_{\text{min}}]\). The host set of \( p \) is compact and therefore continuity implies uniform continuity and therefore \( \delta p_{\text{min}} > 0 \) can be found that does not depend on \( p_0 \). Moreover, since there are a finite number of bases, \( \delta p_{\text{min}} > 0 \) can be determined irrespective of the basis considered.

After a finite number of iterations \( \delta p \) is therefore sufficiently small and no backtracking is necessary. Moreover, at each iteration a nonzero step forward is taken. Since \([p_{11}, p_{1u}]\) is bounded, and at each iteration a nonzero step forward is taken, after a finite number of iterations \([p_{11}, p_{1u}]\) is covered. \( \square \)

**Remark 3.3.** If one does not allow the \( \epsilon \)-violation of the primal and dual constraints, linear dependence of constraints can result in one qualitatively invariant solution being optimal for a single parameter value. In that case the algorithms need to be slightly modified and a more elaborate argument for finite termination is needed. The use of finite precision arithmetic justifies the use of the tolerances. Moreover, LP solvers typically give an approximately optimal solution, that satisfies similar tolerances. Obviously, the tolerances used in our algorithms need to be larger than the ones used by the LP solvers.

4. MILP optimality range

In the general case of parametric MILP (1) the notion of a qualitatively invariant solution is analogous to the parametric LP (2), by fixing the integer variables to an arbitrary value and considering the resulting parametric LP. Unlike LPs, no explicit optimality conditions are available for MILPs and therefore calculating the optimality range of a qualitatively invariant solution is challenging in general. Pertsinidis et al. [13,7] considered the right-hand side case with an affine dependence of the right-hand side vector on the parameter and under a uniqueness assumption formulated a new optimization problem in which the parameters are added to the variable list. Pertsinidis et al. refer to this as “sensitivity analysis”, but to avoid confusion with the parametric dependence of the optimal solution we refer to it as finding the optimality range. We consider and discuss generalizations of a variation of this formulation, compare also [12] for alternatives that only establish the optimality of a qualitatively invariant solution over a given parameter set.

4.1. Optimality of a pair

Suppose that feasibility of \((\tilde{x}(p), \tilde{y})\) has been established for all \( p \in P' \subset P \) and optimality for \( p = \tilde{p} \) with \( \tilde{p} \in P' \). Starting from (1) the parameter is added to the list of variables and the following optimization problem is formulated
\[
\begin{align*}
\min_{x,y,p} \quad & p \\
\text{s.t.} \quad & (c'(p))^T x + (c'(p))^T y - f(p) \leq -\varepsilon, \\
& A^{1x}(p)x + A^{1y}(p)y = b^1(p), \\
& A^{2x}(p)x + A^{2y}(p)y \leq b^2(p), \\
& x^L \leq x \leq x^U, \\
& x \in \mathbb{R}^n, y \in \{0,1\}^n, \\
& p \in P',
\end{align*}
\]

where \( f(p) \equiv (c'(p))^T x + (c'(p))^T y \) and \( \varepsilon \) is a prespecified optimality tolerance. If \( (8) \) is infeasible, \( (x(p), y) \) is an \( \varepsilon \)-optimal solution for the MILP \( (1) \) for all \( p \in P' \). Otherwise \( (x(p), y) \) is not optimal in \( (1) \) for \( p = p' \) (the parameter value furnished by \( (8) \)). The objective function ensures that if \( p' > p \) then \( (x(p), y) \) is \( \varepsilon \)-optimal in \( (1) \) for \( p \in [p, p') \) and this is used in Algorithm 1. Note that the tolerance \( \varepsilon \) introduces an overestimation of the optimality range. Note also that the solution point \( (x^*, y^*) \) furnished by \( (8) \) need not be optimal in \( (1) \) for \( p' \) and this motivates solving \( (1) \) at \( p' \) in Algorithm 1.

4.2. Range of infeasibility

A similar question to the optimality range considered in the previous subsection is the “range of infeasibility”. Recall that in LPs, the introduction of surplus variables allows the infeasibility to be treated similarly to a feasible basis. This is not possible for MILPs, because parametric variation may render some other integer realization feasible. Supposing that infeasibility has been shown for \( p = p' \) one possibility to obtain the infeasibility range is to consider a variant of \( (8) \) where the bound on the optimal solution value is dropped

\[
\begin{align*}
\min_{x,y,p} \quad & p \\
\text{s.t.} \quad & A^{1x}(p)x + A^{1y}(p)y = b^1(p), \\
& A^{2x}(p)x + A^{2y}(p)y \leq b^2(p), \\
& x^L \leq x \leq x^U, \\
& x \in \mathbb{R}^n, y \in \{0,1\}^n, \\
& p \in P'.
\end{align*}
\]

If \( (9) \) is feasible for some parameter value \( p' \) then \( (1) \) is also feasible for \( p = p' \). Otherwise, \( (1) \) is infeasible in \( P' \).

4.3. Classification of optimality region formulations

In general, both formulations \( (8) \) and \( (9) \) contain nonconvex functions and are characterized as separable nonconvex MINLPs. The global solution of nonconvex MINLPs is very computationally expensive but there exist algorithms [48–50] and at least one commercial program [51] that can do this with optimality guarantees. Moreover, the specific structure of the MINLP considered could be exploited by specialized algorithms. The nonlinearity originates from \( f(p) \) as well as from the products between the parameter dependent data and the parameters. In that sense, the parameter \( p \) can be considered as a complicating variable. Note that even in the multiparametric case typically there is a relatively small number of parameters.

There are some cases where the optimization formulations are MILPs or can be reformulated exactly to MILPs. The simplest case is the right-hand side case with an affine dependence of the right-hand side vector \( b^1 : P \rightarrow \mathbb{R}^n \) on the parameter. For more general cases see [12].

Typically MINLPs are solved finitely only to a tolerance \( \epsilon_{\text{MINLP}} > 0 \), i.e., finitely terminating solvers furnish a feasible point which gives an upper bound UBD to the optimal objective value \( f^* \) (\( f^* \leq \text{UBD} \)) along with lower bound LBD to \( f^* \) (certificate of optimality \( \text{LBD} \leq f^* \)), such that \( \text{LBD} + \epsilon_{\text{MINLP}} > \text{UBD} \). For our purposes, \( \epsilon_{\text{MINLP}} \) needs to be set tightly and the final lower bound provided by the MINLP be used. Similarly to the considerations presented for LPs, feasibility of MINLPs is typically only guaranteed within \( \epsilon_{\text{feas}} \) tolerance. This does not introduce any complications relative to the ones already discussed.

5. Parametric mixed-integer linear program

5.1. Optimality-region algorithm for parametric MILP

Based on the optimality region formulation we propose an algorithm for parametric MILPs based on decomposition into a series of regular MILPs, parametric LPs and regular mixed-integer nonlinear programs (MINLPs). We therefore
assume the existence of a MILP and a MINLP solver that either establish infeasibility or give an integer optimal solution point. The general structure of the proposed algorithm is the same as for Algorithm 0 described for the parametric LP. Note that state-of-the-art MINLP solvers require bounded variables [51,52], and therefore Algorithm 1 is only applicable to problems with finite \( x^l, x^u \).

**Algorithm 1 (Parametric mixed-integer linear program via optimality region).** Input to the algorithm are the tolerances for violation of the primal inequalities \( \varepsilon_{\text{inf}} \) and the marginal cost inequalities \( \varepsilon_{\text{opt}} \), the tolerance \( \varepsilon \) for the optimality region formulation and a guess for the minimal parameter step \( \delta p > 0 \). The algorithm uses a set \( R \) to store the optimal solutions. The elements \( R_i \) of \( R \) are quadruplets, composed of parameter values \( p^R \), a boolean \( g^R \), describing whether the problem is feasible for this element (\( g^R = \text{true} \)) or not (\( g^R = \text{false} \)), a point \( (x^R(p), y^R) \), and the corresponding objective function \( f^R(p) \).

1. Initialize with \( p_s = 0 \).
2. REPEAT
   a. Solve MILP (1) for \( p = p_s + \delta p, \text{IF feasible THEN} \)
   i. Fix integer variables to the optimal solution value \( y \).
   ii. Solve LP (2) for \( p = p_s + \delta p \) and obtain an optimal basis.
   iii. Set up the parametric system of equations and inequalities.
   iv. Obtain the parametric dependence of the solution \( x(p) \) for \( p \in [p_s, 1] \).
   v. Get the feasibility range of \( x(p) \) for \( [p_s, 1] \), set \( p_t \) equal to the lowest parameter value for which a primal constraint is violated. IF \( p_t < p_s + \delta p \) THEN Set \( \delta p = \delta p/2 \). GOTO Step 2a(ii).
   vi. Optional LP Optimality Step Get the optimality range of \( x(p) \) for \( [p_s, p_t] \), set \( p_t \) equal to the lowest parameter value for which a marginal value constraint is violated. IF \( p_t \leq p_s + \delta p \) THEN Set \( \delta p = \delta p/2 \). GOTO Step 2(a)ii.
   vii. Set \( \varepsilon_{\text{MINLP}} > \delta p/2 \) THEN set \( \varepsilon_{\text{MINLP}} = \delta p/2 \). Solve MINLP (8) for \( p \in [p_s, p_t] \), to \( \varepsilon_{\text{MINLP}} \) optimality and obtain the final lower bound \( p^* \) to the optimal objective value. IF feasible THEN \( p_t = p^* \).
   viii. IF \( p_t \leq p_s + \delta p \) THEN Set \( \delta p = \delta p/2 \). GOTO Step 2a.
   ix. Store \( p_t, g = \text{true}, (x(p), y), (f(p)) \) in \( R \).
   ELSE
   i. Solve MINLP (9) for \( p \in [p_s, p_t] \) and obtain optimal objective value \( p^*, \text{IF feasible THEN} p_t = p^* \text{ ELSE} p_t = 1 \).
   ii. IF \( p_t \leq p_s + \delta p \) THEN Set \( \delta p = \delta p/2 \). GOTO Step 2a.
   iii. Store \( p_t, g = \text{false}, f(p) = +\infty \) in \( R \).
   END
   b. Set \( p_s = \min\{p_t, 1\} \).
UNTIL \( p_s \geq 1 \)

The benefit of Step 2(a)vi is that the range of \( p \) is smaller in the MINLP, which typically accelerates convergence.

**Theorem 2.** If Assumptions 1–3 hold for each fixed integer realization, Algorithm 1 terminates finitely. At termination \( R \) contains an approximate solution to the parametric MILP. For \( g^R = \text{true} \), a point \( x^R(p) \) is given associated with a basis of (1) for fixed \( y = y^R \), that satisfies its primal and marginal cost constraints within \( \varepsilon_{\text{inf}} \) and \( \varepsilon_{\text{opt}}\)-tolerance respectively for \( p \in [p^R_1, p^R_{n+1}] \). Moreover, the pair \( (x^R(p), y^R) \) is \( \varepsilon \)-optimal in (1) for \( p \in [p^R_1, p^R_{n+1}] \). For \( g^R = \text{false} \), infeasibility has been established for \( p \in [p^R_1, p^R_{n+1}] \) by (9).

Theorem 2 can be proved similarly to Theorem 1. Finite termination of the algorithm is guaranteed due to the use of the tolerances \( \varepsilon_{\text{inf}}, \varepsilon_{\text{opt}} \) and \( \varepsilon \), in the primal conditions, dual conditions and optimality region formulation respectively, which essentially overestimate the true bounds of the optimality range. As a consequence at each iteration there is a finite increase in the parameter value. Another consequence is that due to the overestimation the next MILP call results in a new solution.

**5.2. Branch-and-bound algorithm for parametric MILP**

B&B is a well-established solution method for regular MILP, aiming to avoid enumeration of all integer realizations. The optimal solution value is bracketed between a lower bound and an upper bound. The lower bound is typically obtained by relaxing the integer variables \( y \in \{0,1\}^n \) to \( y \in \{0,1\}^n \) and solving the resulting LP. The upper bound is typically obtained from a LP relaxation that is integer feasible. Convergence of the lower and upper bounds is obtained by branching, i.e., picking an integer variable \( y_i \), and creating two child nodes: one with \( y_i = 0 \) and one with \( y_i = 1 \). In the worse case \( 2^{n+1} - 1 \) LPs have to be solved, but typically B&B outperforms explicit enumeration. For a thorough discussion of B&B the reader is referred to [53].
B&B algorithms for parametric mixed-integer problems parallel the B&B algorithm for a regular MILP, by branching on the integer variables and solving a parametric LP at each node in the B&B tree. For instance, Ohtake and Nishida [20] have applied this idea to the right-hand side case.

Within Algorithm 2 nodes are created, each corresponding to a parametric MILP with some binary variables fixed:

$$\min_{x,y} (c_i(p))^T x + (c_j(p))^T y$$

s.t.

$$A^{li}(p)x + A^{lj}(p)y = b^l(p),$$

$$A^{zi}(p)x + A^{zj}(p)y \leq b^z(p),$$

$$x \in \mathbb{R}^n, x^L \leq x \leq x^U,$$

$$y_j = 0 \quad \forall j \in Z^i,$$

$$y_j = 1 \quad \forall j \in O^i,$$

$$y_j \in \{0,1\} \quad \forall j \in \{1,\ldots,n_i\}, \quad j \notin Z^i \cup O^i,$$

$$p \in [0,1]$$

for the finite index sets $Z^i, O^i$. Note that the fixed binary variables are only dummy variables. A lower bound can be obtained from the parametric solution of the LP-relaxation of (10), i.e., a problem of the form (2) in which $y_j$ have applied this idea to the right-hand side case. Out of the solutions obtained over the parameter range an upper bound to the solution of (1) is provided by those that are binary feasible.

Algorithm 2 (Branch-and-bound algorithm for a general parametric MILP). The upper bound is stored in the set $R^0$. The elements $R^0_i$ of $R^0$ are quadruplets, composed of parameter values $p^{R^0_i}$, a boolean $g^{R^0_i}$, describing whether an upper bound exists for this element ($g^{R^0_i} = true$) or not ($g^{R^0_i} = false$), a point $(x^{R^0_i}(p), y^{R^0_i})$, and the corresponding objective function $f^{R^0_i}(p)$. The elements are ordered according to $p^{R^0_i}$, and by convention $p^{R^0_{i+1}} = 1$ for $l = |R^0|$. For $g^{R^0_i} = true$, the point $(x^{R^0_i}(p), y^{R^0_i})$ is $\epsilon$-optimal for the parametric MILP for $p \in [p^{R^0_i}, p^{R^0_{i+1}}]$ and $f^{R^0_i}(p)$ is an upper bound to the optimal objective function in that interval.

Let the index set $A$ contain the indices $i$ of the currently active nodes, each corresponding to a parametric MILP (10) and each associated with index sets $Z^i, O^i$. Also associated with each node $i$ is a set $R^i$, with the same type of elements as $R^0$. The boolean $g^{R^i}$ describes if the element is active ($g^{R^i} = true$), i.e., branching needs to be performed, or inactive ($g^{R^i} = false$), i.e., no further branching is needed. An element can become inactive by an infeasible lower bounding problem or by value dominance. Note that the definition of the boolean conforms with the definition of the booleans in Algorithm 0.

1. (Initialization) Set $A = \{1\}$. Set $Z^1 = O^1 = \emptyset$. Set $p^{R^1} = 0$, $g^{R^1} = true$, $f^{R^1} = -\infty$, $R^1 = \{R^1\}$. Set $p^{R^0} = 0$, $g^{R^0} = false$, $f^{R^0} = +\infty$, $R^0 = \{R^0\}$. Set $k = 1$.

2. (Termination test) If $A = \emptyset$ then terminate.

3. (Node selection) Select and delete a node $i$ from $A$.

4. (Comparison of lower and upper bound) CALL Check Lower Bound (Subroutine 4).

5. (Relaxation) FOR $l = 1, \ldots, |R^i|$ DO

   IF $g^{R^i} = true$ THEN solve the LP relaxation of node $i$ and replace $R^i$ with the solution of the parametric LP.

END

6. (Update of upper bound and fathoming) CALL Update Upper Bound (Subroutine 5).

7. (Branching) IF $g^{R^i} = true$ for some $l \in \{1, \ldots, |R^i|\}$ THEN

   Select a free binary variable $j \notin Z^i \cup O^i$.

   Create subproblems with $Z^{k+1} = Z^i \cup j$, $O^{k+1} = O^i$ and $Z^{k+2} = Z^i$, $O^{k+2} = O^i \cup j$.

   SET $R^{k+1} = R^i$ and $R^{k+2} = R^i$.

   Add nodes $k + 1$ and $k + 2$ to $A$.

   Set $k = k + 2$.

8. GOTO 2.

On termination $R^0$ contains the solution of the parametric mixed-integer program. For all elements $R^0_i$ such that $g^{R^0_i} = false$ the program is infeasible for $p \in [p^{R^0_i}, p^{R^0_{i+1}}]$ while for $g^{R^0_i} = true$, the point $(x^{R^0_i}(p), y^{R^0_i})$ is an optimal solution (within the tolerance satisfied by the solution of the LP relaxations) for $p \in [p^{R^0_i}, p^{R^0_{i+1}}]$ with the objective function $f^{R^0_i}(p)$.

Note that the definition of the boolean conforms with the definition of the booleans in Algorithm 0.
In the following we describe the subroutines called by Algorithm 2. To simplify notation we use two actions called *merge* and *split*. As their name suggests, the former takes two adjacent elements of \( R^i \) (\( i \geq 0 \)) and merges them to one, while the latter takes one element and generates two adjacent elements. After either action the numbering of the elements is reset according to the element order.

The purpose of Subroutine 4 is to fathom some elements of \( R^i \) based on value dominance prior to solving the parametric LP. First elements of \( R^i \) and \( R^0 \) are split in such a way that \(| R^i | = | R^0 | \) and \( p^{R_i} = p^{R_0} \) for all \( i = 1, \ldots, | R^i | \). Then parameter intervals are identified for which the lower bound at node \( i \) is higher than the incumbent, and therefore the parametric LP does not need to be solved. Finally, active elements of \( R^i \) are merged in order to minimize the number of LP calls required for the solution of the parametric LP. This is an algorithmic heuristic, and if information concerning the parent node is used for the solution of the child node, merging the intervals may actually be detrimental for the computational speed.

**Subroutine 4** (*Check lower bound (integer \( i \))*).

1. **FOR** \( l = 1, \ldots, | R^i | \) **DO**
   - **IF** \( p^{R_{i+1}} < p^{R^i_{i+1}} \) **THEN** split \( R^i_l \) at \( p^{R^i_{i+1}} \)
   - **ELSE IF** \( p^{R_{i+1}} > p^{R^i_{i+1}} \) **THEN** split \( R^i_l \) at \( p^{R^i_{i+1}} \).
2. **END**
3. **FOR** \( l \in \{1, \ldots, | R^i | : g^{R_i} = g^{R^i} = \text{true} \} \) **DO**
   - Set \( p_l \) equal to first root of \( f^{R_i}(p) = f^{R^i}(p) \) in \([p^{R_i}, p^{R^i + 1}]\).
   - **IF** \( p_l < p^{R^i + 1} \) **THEN**
     - Split \( R^i_l \) at \( p_l \).
     - Split \( R^i_{l+1} \) at \( p_l \).
   - **IF** \( f^{R_i}(p) \geq f^{R^i}(p) \) **THEN** set \( g^{R_i} = \text{false} \).
4. **END**
5. **FOR** \( l = 1, \ldots, | R^i | \) **DO**
   - **IF** \( g^{R_i} = g^{R^i} = \text{true} \) **THEN** merge \( R^i_l \) and \( R^i_{l+1} \).
6. **END**

In Subroutine 5 the incumbent is updated based on the parametric solution of node \( i \). First, elements of \( R^i \) and \( R^0 \) are split in such a way that \(| R^i | = | R^0 | \) and \( p^{R_i} = p^{R_0} \) for all \( i = 1, \ldots, | R^i | \). Then, parameter intervals are identified for which the lower bound at node \( i \) is higher than the incumbent, and therefore the parametric LP does not need to be solved at child nodes. Similarly, intervals with integer feasible solutions need not be solved at child nodes. The upper bound is updated for parameter intervals for which the lower bound at node \( i \) is lower than the incumbent and the solution integer feasible. Finally, elements of \( R^0 \) without an upper bound as well as elements with the same solution are merged.

**Subroutine 5** (*Update upper bound (integer \( i \))*).

1. **FOR** \( l = 1, \ldots, | R^i | \) **DO**
   - **IF** \( p^{R_{i+1}} < p^{R^i_{i+1}} \) **THEN** split \( R^i_l \) at \( p^{R^i_{i+1}} \)** ELSE IF** \( p^{R_{i+1}} > p^{R^i_{i+1}} \) **THEN** split \( R^i_l \) at \( p^{R^i_{i+1}} \).
     **END**
2. **FOR** \( l \in \{1, \ldots, | R^i | : g^{R_i} = \text{true} \} \) **DO**
   - Set \( p_l \) equal to first root of \( f^{R_i}(p) = f^{R^i}(p) \) in \([p^{R_i}, p^{R^i + 1}]\).
   - **IF** \( p_l < p^{R^i + 1} \) **THEN**
     - Split \( R^i_l \) at \( p_l \).
     - Split \( R^i_{l+1} \) at \( p_l \).
   - **IF** \( (g^{R_i} = \text{true} \text{ AND } f^{R_i}(p) \geq f^{R^i}(p)) \) **THEN** set \( g^{R_i} = \text{false} \).
   - **IF** \( (y^{R_i} \in [0, 1]^n \text { AND } (g^{R_i} = \text{false} \text{ OR } f^{R_i}(p) < f^{R^i}(p))) \) **THEN** set \( f^{R_i}(p) = f^{R^i}(p), x^{R_i}(p) = x^{R^i}(p), y^{R_i} = y^{R^i} \) and \( g^{R_i} = \text{true} \).
   - **IF** \( y^{R_i} \in [0, 1]^n \) **THEN** \( g^{R_i} = \text{false} \).
     **END**
3. **FOR** \( l = 1, \ldots, | R^i | \) **DO**
   - **IF** \( g^{R_i} = g^{R^i} = \text{false} \) **THEN** merge \( R^i_l \) and \( R^i_{l+1} \).
     **END**
4. **FOR** \( l = 1, \ldots, | R^0 | \) **DO**
• If \((g^0_p = g^{C_p+1}_p = \text{true} \text{ AND } x^0_p(p) = x^{C_p+1}_p(p) \text{ AND } y^0_p = y^{C_p+1}_p) \text{ OR } (g^0_p = g^{C_p+1}_p = \text{false})\) then merge \(R^0_p\) and \(R^{C_p+1}_p\).

End

Theorem 3. Algorithm 2 terminates finitely with the optimal solution of the parametric MILP (within the tolerances satisfied by the solution to the parametric LP).

Proof. Finite termination is guaranteed since branching is only performed on the integer variables and therefore at most \(2^{n_p+1} - 1\) nodes are visited. At each node the solution of a parametric LP is compared to the incumbent. Since a parametric LP has a finite number of optimality regions, this comparison is a finite procedure. The validity of the solution to the parametric MILP (within the tolerances satisfied by the solution to the parametric LP) is guaranteed since for each optimality region, the rules of regular B&B are observed, i.e., fathoming by infeasibility and value dominance and updating of the incumbent by integer feasible solutions to the parametric LP.

6. Implementation

To test the algorithms presented we implemented a prototype in c and in this section we give implementation details and briefly discuss some interesting points.

6.1. Polynomials and rationals

Each polynomial \(q(p)\) is stored as an array of coefficients \(q_i\). The required array size is equal to the order of the polynomial, but for simplicity fixed-size arrays are used, based on a predefined maximal array size. The order of the polynomial is stored to avoid unnecessary operations. The array coefficients are allowed to be either a native c real number of double precision, or one of three types defined in the GNU Multiprecision Library [54], namely (i) long precision float, (ii) integer, or (iii) rational. When the absolute value of the leading coefficient is less than a prespecified tolerance the order of the polynomial is reduced to avoid explosion of the order of the polynomials. Note that the operations are sensitive to this tolerance, and careful tuning is needed to solve systems of more than a few variables successfully.

Implementation of addition and subtraction of polynomials is straightforward, by respectively adding or subtracting the coefficients. The order of the resulting polynomial is equal to the maximal order of the two polynomials. Multiplication of polynomials \(q^1(p) = q^1(p) \times q^2(p)\) is a little more elaborate; the order of the resulting polynomial \(q^3(p)\) is equal to the sum of the orders of the two multiplicands \(q^1(p), q^2(p)\) and the \(i\)th coefficient is calculated as \(q^3_i = \sum_{j=0}^{i} q^1_j q^2_{i-j}\).

Rationals are stored as the quotient of two polynomials \(r(p) = \frac{q^1(p)}{q^2(p)}\) with an additional integer that describes whether a rational is identical to zero. This integer allows unnecessary operations to be avoided, by the use of the elementary rules \(0 \pm r(p) = r(p)\) and \(0 \times r(p) = 0\). This (partially) takes advantage of the sparsity that typical MILPs have. The rational operations needed are defined based on elementary operations on the numerator and denominator polynomials. Consider two rationals \(r^1(p) = \frac{q^1(p)}{q^2(p)}\) and \(r^2(p) = \frac{q^3(p)}{q^4(p)}\) and two scalars \(x^1, x^2\). Multiplication and division of rationals is straightforward

\[
\begin{align*}
r^1(p) \times r^2(p) &= \frac{q^1_1(p)q^2_1(p)}{q^3_1(p)q^4_1(p)}; \\
r^1(p) / r^2(p) &= \frac{q^1_1(p)q^4_1(p)}{q^2_1(p)q^3_1(p)}.
\end{align*}
\]

The linear combination of rationals (for the LU factorization) is given as

\[
x^1 r^1(p) + x^2 r^2(p) = \frac{x^1 q^1_1(p)q^2_2(p) + x^2 q^3_1(p)q^4_2(p)}{q^3_1(p)q^4_1(p)}.
\]

Simplification of the rationals is needed if the numerator and denominator have a common denominator. This simplification is implemented with the simple Euclidean algorithm [35,36], which finds the greatest common denominator without resorting to root finding. In the following numerical experiments coefficients with magnitude less than \(10^{-10}\) are considered as zero. Finally, the calculation of roots of polynomials is performed via the Harwell Subroutine pa17bd [55].

6.2. Optimization subproblems

For LP and MILP at fixed parameter values the CPLEX 9.1 callable library is used [39]. For the optimality region formulation as a MINLP the general-purpose global solver BARON version 7.4 [56] available in GAMS version 22.0 [57] is used. All CPLEX optimization tolerances are set to the default value of \(10^{-6}\). The tolerances for violation of primal
feasibility and of the marginal costs within our algorithms is set to $3 \times 10^{-6}$. Default constraint violation tolerances of $10^{-6}$ are used for BARON. For both CPLEX and BARON the termination tolerances are set to $10^{-6}$. A relative optimality tolerance of $1\%$ is used in (8).

6.3. Continuation

Instead of a continuation code with event detection, we use DSL48SE [58] available through DAEPACK [59], which is a numerical integrator for hybrid differential-algebraic equation systems with state-event detection and as such a solver for a more general problem than the one considered here. The default tolerances of $10^{-6}$ are used. A general system of residuals is formulated in FORTRAN and the specific system is defined through parameters. This exploits the system sparsity without the need of recompilation. For simplicity the primal system (4) and the system of marginal costs (6) are solved simultaneously, which has a small overhead, since DSL48SE efficiently exploits sparsity and problem structure. Consistent initialization is performed through a call to MA48 from the Harwell Subroutines [55].

6.4. Branch-and-bound tree

The B&B tree of Algorithm 2 was implemented using the depth-first node selection heuristic. For simplicity the binary variables were selected in numerical order ($j = 1, 2, \ldots, n_y$) and always the node with $y_j = 0$ was solved before the node with $y_j = 1$. Some of the problems contain constraints which only depend on integer variables and are parameter independent. At each node these constraints were checked for integer feasibility. For all the cases considered, the formulated MILP is of insignificant size and CPU requirement.

6.5. Numerical results

We consider 10 small-size parametric LPs and MILPs with interesting theoretical properties, two mid-size MILP case studies from man-portable power generation [12,60], five randomly generated mid-size MILP case studies and two parametric variants from problems from the GAMS model library [57]. In Appendix A the small-scale examples are given along with a summary of the properties of all examples. All examples are available at http://yoric.mit.edu/parametricMILP in GAMS format. Some of the problems have special structure, e.g., the parameter affects only one row, so that specialized algorithms could be used for these problems. The size of the problems was limited to 40 binary variables and a few hundred continuous variables, because on the one hand our implementation is a prototype and not an efficient program, and on the other hand the only comparison available is via discretization of the parameter space on a fine mesh.

In the following the case studies are solved with four alternatives. The first alternative, labeled CPLEX, is a discretization of the parameter range with 1000 equidistant points. This alternative is not rigorous and is done in order to check the results, and as a benchmark for the CPU requirements. The second alternative, labeled B&B+Rational, is the B&B algorithm (Algorithm 2) with the parametric LP solved by rational operations. The third alternative, labeled B&B+DSL48, is the B&B algorithm (Algorithm 2) with the parametric LP solved by a continuation approach. The fourth alternative, labeled DSL48+BARON, is the optimality region algorithm (Algorithm 1). This alternative is not applied on the examples that do not contain binary variables (Examples A.1, A.2, A.3, A.4, A.5, A.6). Note that the MINLP optimization problems are solved separately from the continuation problems after a polynomial is fitted to the solution.

Table 1 compares the computational requirements for the four alternatives on a 64-bit Xeon processor 3.2 GHz running Linux 2.6.13. The first four columns indicate the size of the problem (number of binary variables, number of continuous variables, number of equality constraints, number of inequality constraints) and the following four columns the CPU seconds for each alternative. As expected, the alternative employing rational operations is not robust, particularly for increasing problem size. For several examples, numerical errors in the rational operations caused failure of the algorithm solving the parametric LPs at some node of the B&B tree. This was considered a failure and is indicated by ‘‘–” in Table 1. The two alternatives based on continuation are very robust. Depending on the problem instance, either the branch-and-bound algorithm (Algorithm 2) or the algorithm based on optimality region (Algorithm 1) is more efficient. In all cases at least one of the two was more efficient than the discretization of the parameter space, although the problems did not pose a significant challenge to CPLEX.

Table 2 shows some CPU-independent metrics for the larger MILPs considered. The first column (labeled CPLEX) gives the number of nodes and iterations as reported by CPLEX for $p = 0$ to indicate the relative difficulty of the MILPs considered. Because CPLEX has a powerful preprocessor and is using sophisticated cuts, the number of nodes and iterations can be seen as a lower bound for a simple B&B algorithm. The second column (labeled B&B+DSL48) gives the performance of the B&B tree of Algorithm 2: (i) the number of nodes visited, which gives the number of parametric LPs solved, (ii) the total number of critical regions visited, and (iii) the number of calls to the LP solver. The number of critical regions visited is typically larger than the number of nodes, but can be somewhat smaller if nodes are eliminated based on integer infeasibility. The numbers of calls to the LP solver is greater or equal to the number of critical regions, because for
infeasible LPs for which also the phase I problem has to be solved. The third column (labeled DSL48+BARON) gives the performance of the sensitivity-based algorithm (Algorithm 1). The major iterations correspond to the number of optimality regions identified and the minor iteration to the total number of iterations taken by the MINLP solver. Note that the number of iterations required by the MINLP solver are not directly comparable to the nodes in the B&B tree of Algorithm 2 because BARON performs branching on both binary and continuous variables. Finally, in the last column the maximal number of integer realizations (leaf nodes) \(2^n\) is also given for comparison purposes.

### 7. Discussion and future work

We consider the general case of parametric MILPs, where a parameter affects the cost-vector, right-hand side vectors and matrix. Based on the work by Pertsinidis et al. [13,7] we formulate a single level optimization problem that identifies the optimality region, which in general is a MINLP with nonconvex functions. We first consider parametric LPs and formulate an alternative to the parametric simplex algorithm by Dinkelbach [15]. The algorithm by Dinkelbach works with operations on rational functions, and due to error propagation is not practical for large size problems. Our alternative requires less consecutive operations and as such is somewhat more robust. We also propose an alternative based on the numerical solution of parametric systems of equations via continuation. For the MILPs we propose two algorithms. One algorithm is based on branch-and-bound on the integer variables and the other is based on the identification of optimality regions via MINLPs. We present a number of test problems and numerical results from implementation of our algorithms. As expected, algorithms based on operations with rational functions are only applicable to problems of small size, approximately 10 variables. On the other hand, continuation methods scale favorably to bigger problems. Both algorithms are tractable for medium-scale MILPs, with up to a few tens of binary and few hundreds of continuous variables. To test the algorithms on large-scale problems a more efficient implementation is required.
Using DSL48SE as a continuation code is not the optimal choice. The first reason is related with the step size selection. The integrator used calculates the step size by controlling the truncation and the approximation error associated with the interpolating polynomials [61]. For the continuation problems formulated there is no truncation error so that the step size calculation in DSL48 is not optimal. The second reason is that the state events correspond to the state variables plus a constant and therefore introducing separate discontinuity functions is not the most efficient solution. Furthermore, the corrector used in DSL48 is based on the Newton method, whereas the solution of a linear system suffices. These considerations suggest that a specialized continuation algorithm would show a significant computational improvement. Note also, that the structure of the linear system does not change, so that analysis of the sparsity pattern only needs to be done once. In [12] a simple procedure is sketched.

In the optimality-region-based algorithm for parametric MILP (Algorithm 1) the solution of the MINLP is by far the most time consuming step, especially for increasing problem size. For large-scale parametric MILPs the solution of these MINLPs with state-of-the-art general-purpose solvers is likely intractable. General-purpose solvers can only partially exploit the problem structure and therefore specialized algorithms are expected to outperform these. A simple possibility to improve the computational performance is to perform sampling of the parameter range before calling the MINLP, i.e., choose some parameter points, solve the original MILP and check if the optimal objective value is better than the assumed optimal solution; if it is the parameter range for the MINLP can be reduced. Moreover, if the LP-optimality range for a fixed integer realization has been established for \( p \in P \), then an integer cut \( y \neq y \) can be added to (8), likely accelerating the convergence. Such a cut can be formulated as in Balas and Jeroslow [62]

\[
\sum_{i \in O} y_i - \sum_{i \in Z} y_i \leq |O| - 1,
\]

where \( O, Z \) are index sets for the elements of \( y \) with values of one and zero, respectively.

Different variable selection and branching heuristics can be explored in the branch-and-bound algorithm (Algorithm 2). Moreover, so far upper bounds are only obtained when the LP relaxations give an integer feasible solution. Another possibility to obtain upper bounds is to solve the parametric MILP at a fixed parameter value, then fix the integer variables and finally solve the resulting parametric LP. This is essentially a combination of the branch-and-bound algorithm (Algorithm 2) with the algorithm based on optimality region (Algorithm 1), and is likely to show improved performance over both algorithms. In Algorithm 2 information from a parent node is only used in a very limited way. In analogy to the solution of a regular MILP by B&B computational gains are possible when the solution to the parent node is used to solve the child node, e.g., by providing an initial basis for the LP-solver. A drawback is an increase in the storage requirements. Another limitation of the branch-and-bound algorithm is that it does not take advantage of advanced features of MILP solvers, such as the introduction of cuts. The development of cuts that are valid in the parametric case are expected to significantly improve the convergence of branch-and-bound.

The extension of our algorithms to the multiparametric case is not straightforward. A major complication is that the optimality regions are arbitrary sets [12]. Another complication is that some of the approaches presented, such as predictor–corrector continuation, do not readily extend to the multiparametric case. On the other hand, some of the ideas presented can be extended to the general nonlinear case [12].

Acknowledgements

This work was supported by the DoD Multidisciplinary University Research Initiative (MURI) program administered by the Army Research Office under Grant DAAD19-01-1-0566.

We would like to thank George John Gesslein II [63] for his helpful comments regarding the implementation of rational operations.

Appendix A. Parametric optimization test set

This section contains a summary of the examples used. All examples are available at http://yoric.mit.edu/parametric-MILP in GAMS format [57].

Example A.1. The following parametric LP is based on Murty [28]

\[
\begin{align*}
\min_x & \quad 16x_4 + 4x_5 + x_6 \\
\text{s.t.} & \quad x_1 + x_4 = 8 - 20(1 + 2p), \\
& \quad x_2 + x_4 + x_5 = 4 - 20(1 + 2p), \\
& \quad x_3 + 2x_4 + 2x_5 + x_6 = 2 - 20(1 + 2p), \\
& \quad x \in \mathbb{R}, \quad x \geq 0.
\end{align*}
\]

There are eight optimal solutions with the breakpoints \( p = 0.55, p = 0.6, p = 0.65, p = 0.7, p = 0.75, p = 0.8, \) and \( p = 0.85. \)
Example A.2. The cost-vector parametric LP

\[
\begin{align*}
\min_x & \quad (-1 + 2p)x \\
\text{s.t.} & \quad x \leq 1, \\
& \quad x \in [0, 10]
\end{align*}
\]

is feasible for \( p \in [0, 1] \). For \( p \in [0, 0.5) \) the unique optimal solution is \( x = 1 \) (active inequality constraint) and an objective function of \(-1 + 2p\). For \( p = 0.5 \) any \( x \in [0, 1] \) is optimal with an objective value of 0. For \( p \in (0.5, 1] \), the unique optimal solution is \( x = 0 \) (inactive inequality constraint) with an objective function of 0.

Example A.3. The parametric LP

\[
\begin{align*}
\min_x & \quad x_1 + 10x_2 \\
\text{s.t.} & \quad -0.6x_1 - x_2 \leq -1, \\
& \quad (0.1 + p)x_1 = 1, \\
& \quad x \in [0, 10]^2
\end{align*}
\]

is feasible for \( p \in [0, 1] \). For \( p \in [0, 0.5] \) the optimal solution is \( x_1 = 1/(p + 0.1), x_2 = 0 \) and the optimal objective function \( 1/(p + 0.1) \); here the inequality constraint is inactive. For \( p \in [0.5, 1] \) the optimal solution is \( x_1 = 1/(p + 0.1), x_2 = 1 - 0.6/(0.1 + p) \) and the optimal objective function \( 1 + 5/(0.1 + p) \); here the inequality constraint is active.

Example A.4. The parametric LP

\[
\begin{align*}
\min_x & \quad x_2 \\
\text{s.t.} & \quad x_2 - 100px_1 = 0, \\
& \quad x_1 = p, \\
& \quad x \in [0, 10]^2
\end{align*}
\]

is feasible for \( p \in [0, 0.1\sqrt{10}] \approx [0, 0.3162] \) with an optimal solution \( x_1 = p, x_2 = 100p^2 \) and an optimal objective function \( 100p^2 \).

Example A.5. The right-hand side parametric LP

\[
\begin{align*}
\min_x & \quad x_3 \\
\text{s.t.} & \quad x_1 + x_2 = 0, \\
& \quad x_3 = -0.5 + p, \\
& \quad x \in [0, 10]^3
\end{align*}
\]

if feasible for \( p \in [0.5, 1] \) with an optimal solution \( x_1 = x_2 = 0, x_3 = p - 0.5 \) with an optimal objective function of \( p - 0.5 \).

Example A.6. The parametric LP

\[
\begin{align*}
\min_x & \quad x_2 \\
\text{s.t.} & \quad (1 - p)x_2 = 0.1, \\
& \quad x_1 = 0, \\
& \quad x \in [0, 1]^2
\end{align*}
\]

is feasible for \( p \in [0, 0.9] \) with an optimal solution \( x_1 = 0, x_2 = 0.1/(1 - p) \) and an objective function \( 0.1/(1 - p) \).

Example A.7. The parametric MILP

\[
\begin{align*}
\min_{x,y} & \quad y + x \\
\text{s.t.} & \quad (-1 + 2p)x = 1, \\
& \quad x \in [0, 10], \\
& \quad y \in \{0, 1\}
\end{align*}
\]
is feasible for \( p \in [0.55, 1.0] \) with an optimal solution \( y = 0 \) and \( x = 1 / (-1 + 2p) \) and an optimal objective function \( 1 / (-1 + 2p) \). For \( p \in [0, 0.5] \) the solution of the equality constraint gives a negative \( x \) and the problem is infeasible. For \( p = 0.5 \) the matrix is singular and no solution exists for the equality constraint. For \( p \in (0.5, 0.55) \) the solution to the equality constraint gives \( x > 10 \) which is infeasible.

**Example A.8.** The right-hand side parametric MILP

\[
\begin{align*}
\min_{y, x} & \quad y + x \\
\text{s.t.} & \quad x = -1 + 2p, \\
& \quad x \in \mathbb{R}, \quad x \geq 0, \\
& \quad y \in \{0, 1\}
\end{align*}
\]

is infeasible for \( p \in [0, 0.5] \), because \( x \geq 0 \) is violated. For \( p \in [0.5, 1.0] \) the optimal solution is \( y = 0, x = -1 + 2p \) with an optimal objective function of \(-1 + 2p\).

**Example A.9.** The parametric MILP

\[
\begin{align*}
\min_{x, y} & \quad -x + y_1 \\
\text{s.t.} & \quad y_2 + (0.99p + 0.01)x = 1, \\
& \quad y_1 + y_2 = 1, \\
& \quad x \in [0, 20], \\
& \quad y \in \{0, 1\}^2
\end{align*}
\]

has two candidate integer realizations. The first, \( y_1 = 1, y_2 = 0 \), gives \( x = 1 / (0.99p + 0.01) \) and an objective value of \( 1 - 1 / (0.99p + 0.01) \). This solution is feasible and optimal for \( p \in [0.0404, 1] \). The second integer realization, \( y_1 = 0, y_2 = 1 \), gives \( x = 0 \) and an objective value of 0. This solution is feasible for \( p \in [0, 1] \) and optimal for \( p \in [0, 0.0404] \cup \{1\} \).

**Example A.10.** The parametric MILP

\[
\begin{align*}
\min_{x, y} & \quad 10y_1 + 12y_2 + x_1 + x_2 + x_3 + x_4 + x_5 \\
\text{s.t.} & \quad (-3 - 10p)x_3 + (-3 - 12p)x_4 + (9 - 10p)x_5 + x_9 = 0, \\
& \quad (-1 - 1p)x_1 - x_2 + x_5 = 0, \\
& \quad -x_1 + (-2 - p)x_2 + x_7 = 0, \\
& \quad -20px_3 + x_8 = 0, \\
& \quad x_6 + x_8 = 8, \\
& \quad x_7 + x_9 = 8, \\
& \quad -10y_1 + x_1 \leq 0, \\
& \quad -10y_1 + x_2 \leq 0, \\
& \quad -10y_2 + x_3 \leq 0, \\
& \quad -10y_2 + x_4 \leq 0, \\
& \quad -10y_2 + x_5 \leq 0, \\
& \quad x_1 + x_3 \leq 10, \\
& \quad x_2 + x_4 \leq 10, \\
& \quad y \in \{0, 1\}^2, \\
& \quad x \in [0, 10]^9
\end{align*}
\]

has three optimality regions. For \( p \in [0, 0.093] \) the optimal solution is \( y_1 = 1, y_2 = 0 \) with \( x_1, x_4, x_5, x_6, x_9 \) fixed to zero. For \( p \in [0.093, 0.299] \) the optimal solution is \( y_1 = 0, y_2 = 1 \) with \( x_1, x_2, x_4, x_6, x_7 \) fixed to zero. For \( p \in [0.299, 1] \) the optimal solution is \( y_1 = 0, y_2 = 1 \) with \( x_1, x_2, x_3, x_6, x_7 \) fixed to zero.

**Example A.11 (Small superstructure).** This example considers the flowsheet design of a process superstructure for man-portable power generation devices subject to an unknown parameter [12]. It corresponds to a simple variant of the
superstructure described in [64] with the solid oxide fuel cell efficiency as the unknown parameter $p$. The MILP used does not correspond to a typical model, because we tried to introduce the least number of few variables and constraints possible; it contains 14 inequality constraints, two equality constraints, seven continuous variables and four binary variables. The matrix contains 55 nonzero elements and only one element depends on the parameter. Note that since only one column is affected by the parameter variation, an analytical solution of the matrix inversion in the LPs is possible [2] but not used here. We considered only $p \in [0, 0.97]$, because after 0.97 the current basis becomes infeasible. There are two optimal solutions with optimality intervals [0, 0.35] and [0.35, 0.97].

**Example A.12 (Big superstructure).** This example considers the flowsheet design of a process superstructure [12]. It corresponds to the superstructure described in [60] with the solid oxide fuel cell efficiency as the unknown parameter $p$. The model is more typical than (A.11) and contains many intermediate variables and equality constraints. The resulting system has a total of $m_1 = 216$ inequality constraints, $m_2 = 226$ equality constraints, $n_c = 329$ continuous variables, $n_b = 20$ binary variables, while the matrix (before the augmentation) contains 1253 nonzero entries. Only one row depends on the parameter. Note that since only one column is affected by the parameter variation, an analytical solution of the matrix inversion in the LPs is possible [2] but not used here.

**Example A.13.** This randomly generated example contains 20 binary and 20 continuous variables and 20 inequalities. The matrix contains 54 nonzero elements and 13 data entries are parameter dependent.

**Example A.14.** This randomly generated example contains 20 binary and 20 continuous variables and 20 inequalities. The matrix contains 102 nonzero elements and 13 data entries are parameter dependent.

**Example A.15.** This example is a larger variant of a bid example (bid.gms) found in the GAMS library [57]. Here each of the five vendors offers four alternatives. The minimal quantity for alternative 3 of vendor 3 is used as a parameter. The problem contains 20 binary and 61 continuous variables with 42 equality and five inequality constraints. The matrix contains 195 nonzero elements and four data entries are parameter dependent.

**Example A.16.** This randomly generated example contains 20 binary and 200 continuous variables and 100 inequalities. The matrix contains 193 nonzero elements and 70 data entries are parameter dependent.

**Example A.17.** This randomly generated example contains 30 binary and 200 continuous variables and 200 inequalities. The matrix contains 622 nonzero elements and 167 data entries are parameter dependent.

**Example A.18.** This randomly generated example contains 30 binary and 200 continuous variables and 200 inequalities. The matrix contains 622 nonzero elements and 167 data entries are parameter dependent.

**Example A.19.** This randomly generated example contains 30 binary and 200 continuous variables and 200 inequalities. The matrix contains 589 nonzero elements and eight data entries are parameter dependent.

References


