Sensitivity analysis for linear changes of the constraint matrix of a (mixed-integer) linear program

Guillaume Derval
Montefiore Institute, ULiège, Belgium
gderval@uliege.be

Damien Ernst
Montefiore Institute, ULiège, Belgium
dernst@uliege.be

Quentin Louveaux
Montefiore Institute, ULiège, Belgium
q.louveaux@uliege.be

Bardhyl Miftari
Montefiore Institute, ULiège, Belgium
bmiftari@uliege.be

Abstract

Understanding how the optimal value of an optimisation problem changes when its input data is modified is an old question in mathematical optimisation. This paper investigates the computation of the optimal values of a family of (possibly mixed-integer) linear optimisation problems in which the constraint matrix is subject to linear perturbations controlled by a scalar parameter that varies within a given interval. This is a largely unresolved question with the additional burden that the resulting value function may be largely irregular. We propose several bounding techniques that provide formal guarantees on the behaviour of the objective value across the entire parameter range. The proposed bounds rely on tools from robust optimisation, Lagrangian relaxation, and ad-hoc reformulations. Each method is assessed in terms of accuracy, precision, and computational performance. Experimental results on a large benchmark set show that the proposed bounding techniques effectively address this class of problems, delivering strong guarantees and good precision. In addition, we introduce a spatial branch-and-bound algorithm that incorporates these bounds to compute an anytime approximation of the value function within a given error tolerance, and we analyse its computational performance.

Keywords Linear Programming, Mixed-Integer Programming, Sensitivity, Bounds, Parametric Programming.

1 Introduction

One strength of linear programming models is the ability to analyze the variation of an optimal solution when we account for slight changes in the data. A popular example comes from modifying one cost coefficient in the objective. In this case, sensitivity analysis allows you to determine in which range the cost coefficient can vary while keeping the current optimal solution optimal. Similarly, if you assume the variation of one right-hand-side value, we can compute the interval in which the current optimal basis remains optimal. In this case, though, the optimal value varies linearly and the optimal dual variable provides the slope of such a linear variation. This can be very helpful for practitioners as it helps understand how robust their solution is and identify which parameters have the greatest impact on the outcome. As a result, linear programming not only provides an optimal solution but also valuable insights into how changes in the model's inputs affect that solution.

However the analysis becomes more complicated when we try to extend it to changing several coefficients simultaneously, or when we want to analyze the variation implied by the change of a matrix coefficient, especially when this matrix coefficient appears in the basis.

Let us detail an application in which this may be natural to do though. Consider a linear optimization problem coming from modeling the investment of an electric infrastructure. To model such a problem, we typically want to simulate the production and consumption of electricity of a few devices over a long horizon. We therefore need to consider a time-indexed formulation that includes, over each period, the same constraints that repeat but include time-indexed variables depending on the time we consider them. For such a model, we typically need to make a few assumptions : the efficiency of the devices, the amount of production of the renewable energy,

the amount of consumption, etc. If we look in particular at the assumed efficiency of a device, we can see that the parameter that we have assumed may appear several times in the problem, mostly with the same value, repeated for every time-period. So if this parameter is uncertain, changing it would involve changing by a same factor several coefficients of the matrix of the linear program.

There are countless similar examples. We can consider the uncertain loss of a percentage of some product during transport, the uncertain ramp-up/down rates of some machine or electrical units, the unknown amount of food needed for livestock, the percentage of workers leaving a company each year and so on.

From theses examples, we want to propose a new model for studying the variation of the optimal value on a linear program. Building on the specific efficiency example, we assume that one parameter varies, but this single parameter may appear in several coefficients of the matrix A. More specifically, we consider a nominal linear (or mixed-integer) program

$$\begin{aligned} \min \quad & \mathbf{c}^t\mathbf{x} \\ \text{s.t} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \in \mathbb{R}^{n_c} \times \mathbb{Z}^{n_i}. \end{aligned} \tag{1}$$

We aim at analyzing how the optimal value varies when one specific parameter $\lambda \in \mathbb{R}$ varies. We assume without loss of generality that the nominal value of λ is 0. This parameter has a linear influence on a subset of the coefficients of the matrix A. More formally, we are interested in the following problem:

$$\begin{aligned} f(\lambda) = \min \quad & \mathbf{c}^t\mathbf{x} \\ \text{s.t} \quad & A\mathbf{x} + \lambda D'\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \in \mathbb{R}^{n_c} \times \mathbb{Z}^{n_i}. \end{aligned} \tag{2}$$

where $\lambda \in [\lambda_1, \lambda_2]$ models the uncertainty, more specifically the potential variation of the coefficients within some limits. The nominal optimal value is therefore given by $f(0)$ with $0 \in [\lambda_1, \lambda_2]$. In this paper, we propose a novel approach consisting of bounding methods on $f(\lambda)$ that provide strong guaranties on the function's behavior while remaining computationally efficient. First, we discuss the existing solutions in the literature.

1.1 Related works

Two closely related fields dealing with these types of problems are Sensitivity Analysis (SA) and Parametric Linear Programming (PLP). They differ based on the assumptions they make upon the modification of λ. Sensitivity analysis considers the effect of the small variation of a parameter around the optimum, whereas linear parametric programming assesses the effect of a parameter on the objective when it varies within a certain range. They both assess the impact of uncertainty on the objective. The methods discussed in this paper are agnostic of the amount of variation and can be applied in both fields.

A complementary field is Robust Optimisation (RO). In contrast to both SA and PLP, RO does not assess the impact of the modification on the objective. Indeed, while SA and PLP focus on the evolution of the optimal objective function given the changes in the parameter, RO techniques focus on finding a solution that is robust to the modification, i.e. a sub-optimal solution that remains feasible for every change in parameter. The three methods, SA, PLP and RO, are complementary as they all try to deal with uncertainty. Several methods discussed in this paper use robust optimisation techniques to achieve the sensitivity analysis.

In its most generic form, the linear problem with a single scalar parameter λ, where the objective coefficients, the constraints and the right-hand side can be simultaneously modified, can be written as follows:

$$\begin{aligned} \min \quad & (\mathbf{c} + \lambda\mathbf{c}')^t\mathbf{x} \\ \text{s.t} \quad & (A + \lambda D')\mathbf{x} \leq \mathbf{b} + \lambda\mathbf{b}' \\ & \mathbf{x} \in \mathbb{R}^{n_c} \times \mathbb{Z}^{n_i} \end{aligned} \tag{3}$$

where λ is a parameter that varies in a given range $[\lambda_1, \lambda_2]$; $\mathbf{c}'$, D' and $\mathbf{b}'$ are the parameter's impact on the objective, constraint matrix and right-hand-side term, respectively. The end goal is to evaluate the objective function $f(\lambda)$ over the range $[\lambda_1, \lambda_2]$.

Reoptimising the problem from scratch for a finite subset of values of λ in $[\lambda_1, \lambda_2]$ can be computationally very costly, especially for big problems that take several minutes or hours to solve once (for a single λ) and does not provide any guarantees in between the computed points. Therefore, several approaches have been considered to mitigate this issue.

Most of the literature focuses on one specific modification at a time; in general, either the objective (via $\mathbf{c}'$), or the right-hand side (RHS, via $\mathbf{b}'$), rarely the left-handside (LHS, via $\mathbf{D}'$) of the continuous version of the linear problem (3), i.e. where all the variables are continuous $\mathbf{x} \in \mathbb{R}^n$. The mixed integer case is hardly considered in the literature even though there is some work [7, 2, 18].

Modification of either the objective or the right-hand side.

For the continuous linear case, let us consider $\mathbf{x}^\star$ and $\boldsymbol{\rho}^\star$ the primal and dual optimal solutions of the continuous unmodified problem, i.e. where $\lambda = 0$:

$$\begin{aligned} \mathbf{x}^\star = \text{argmin} \quad & \mathbf{c}^t\mathbf{x} \\ \text{s.t} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned} \qquad\qquad \begin{aligned} \boldsymbol{\rho}^\star = \text{argmax} \quad & \mathbf{b^t}\boldsymbol{\rho} \\ \text{s.t} \quad & A^t\boldsymbol{\rho} \geq \mathbf{c} \\ & \boldsymbol{\rho} \leq \mathbf{0}. \end{aligned}$$

The modifications on the objective and the right-hand side are closely related as by dualising the problem, one can be transformed into the other. Note that when the modification is only performed on the objective, i.e. only $\mathbf{c}' \neq \mathbf{0}$, the previous primal optimal solution $\mathbf{x}^\star$ remains feasible and similarly, the previous optimal dual solution $\boldsymbol{\rho}^\star$ is still a feasible solution. The methods used for these types of modifications can be categorized into three families [17]:

- *ones using optimal partitions.* The set of active (tight) constraints on the primal and on the dual form an optimal partition of the linear problem. From this partition, we can derive validity intervals upon λ. The methods derived from this family use an LP solver as a subroutine to solve the problem and iteratively find these partitions. Adler and Monteiro [1] introduce the first algorithm from thus family. Berkelaar et al [3] introduce another algorithm based on the additional property that either the primal or the dual optimal set remains unchanged when $\mathbf{c}'$ or $\mathbf{b}'$ is non-zero.
- *ones using optimal values.* The optimal values of the primal, dual and objective are used to build two auxiliary LP problems that give the range upon which solution stays optimal.
- *ones using optimal basis.* The simplex algorithm returns the optimal basis related to a problem. These methods use warm starting and properties related to the basis to perform only a few iterations of the simplex to find another basis when the objective function changes. A two-part paper from Gass and Saaty [25, 12] uses a modified simplex method to retrieve the optimal solution for multiple changes in the coefficient of the objective function. Gal and Nedoma [10] present an effective method for finding the regions that keep a basis optimal for multiple changes in the parameters for both types of modification using the simplex tableaux of multiple reoptimisation and graph theory to combine the tableaux.

Various works [14, 4, 17] make a summary of some techniques and conceptual foundation for post-optimality analysis for modifications on $\mathbf{c}$ and $\mathbf{b}$.

Modifications on the constraint matrix.

It should be noted that when considering modifications $\mathbf{c}'$ and $\mathbf{b}'$, there always exists an equivalent problem (of the form (3)) that encompasses these modifications inside its matrix D', and such that its objective and right-hand side do not depend on λ. The converse is not true: the modifications of the constraint matrix are more general. Gal [11] presents a summary of the different existing techniques for modifications on the constraint matrix A, mostly based on two papers [27, 26]. Sherman and Morison [27] offer a methodology for adjusting an inverse matrix with respect to change of one of its entries. Sherman and Morison [26] offer a formula for the objective considering the rank-one modification $D' = \mathbf{u^t}\mathbf{v}$. Both of these methods lack genericity. Woodbury [28] generalizes the Sherman and Morison formula for any decomposition $D' = UCV$ which can hardly be applied in our case due to having a λ-dependent matrix inversion which is what we are trying to avoid.

More recently, Zuidwijk [29] derives explicit local formulas to compute the evolution of the objective function and intervals on which these formulas are valid. They rewrite the basic inverse matrix as a function of λ and recompute it for several values in the range. Their algorithm involves the finding of the optimal basis, computation

of the range of λ for which the basis stays optimal, the evaluation of the derived formulas using the said basis and reoptimisation when switching to another basis. Their formulas for computing the objective for a given basis require the computation of the eigenvalues of the matrices A and D' and leads to a polynomial problem with orders equal to the number of constraints. For big LPs, finding the solution of the polynomial problem can only be done using approximation methods. Khalipour et al. [19] also discuss an algorithm similar to Zuidwijk [29]. However, they claim that their algorithm can be applied when working on larger problems. They use the Flavell and Salkin [8] formula to compute an approximation of the basic inverse matrix.
All of the above-mentioned methods for the modification of the constraint matrix rely either on heavy computations, reoptimisations, approximations or are not sufficiently generic. Most of the methods cannot be applied on mixed-integer problems. Additionally, we underline that even discretizing $[\lambda_1, \lambda_2]$, with a very fine granularity, at the price of heavy computations, may still lead to a false assessment of the behavior of $f(\lambda)$ over the interval. Indeed, as the matrix D' can contain an arbitrary number of constraints, the resulting problem for a given λ can behave in an unpredictable and sometimes erratic fashion in between two closely located points.

1.2 Our contributions

In this paper, we propose methods to compute upper and lower bounds of problems given in (2), for varying $\lambda \in [\lambda_1, \lambda_2]$. The bounding methods provide guarantees between the computed points. They can capture any erratic behaviour that might be missed by sampling approaches. They also give an indication of the evolution of the objective and can help practitioners focus on particular values of λ within the interval that have an interesting or unconventional behaviour. The bounding methods proposed are based on three approaches: robust optimisation, Lagrangian relaxations and specific reformulations. Out of these approaches, we derive several bounding methods: ones that computes constant bounds, ones giving variables bounds depending on λ and ones computing envelopes on the objective function. The envelopes are a combination of multiple variable bounds, proven optimal in the sense that no other variable bound of the same type is tighter, that form an envelope around the objective function. Some methods add new degrees of freedom, and therefore, can lead to multiple variations.

All the bounding methods are able to provide upper and lower bounds for the continuous case. Three of them of the shelf generate lower bounds for MILPs. Via linearization, two additional methods are able to generate lower bounds for MILPs. Two methods are able to generate upper bounds for MILPs.

To test our methods, we build a new dataset of continuous and mixed-integer problems problems. The dataset is divided in random generated continuous equality and inequality problems, random generated mixed-integer equality and inequality problems, real-life mixed-integer facility location problems and mixed-integer unit commitment problems. For each of these categories, we study the bounding methods in terms of availability (whether a bound is available or not), error (with respect to the best possible solution) and timing. We show that some bounding methods are always available for the continuous inequality problems (to generate lower and upper bounds). For the continuous equality problems, some bounding methods are always available to generate lower bounds. However, generating upper bounds for that class of problems is more complicated with an availability of up-to 41%. For the MILP datasets, the availability of all bounding methods is high. Some reaching 100% availability.

In terms of error, the best performing bounds on the inequality continuous dataset have an error of 9% for lower bounds and 0.15% on upper bounds. For the continuous equality dataset, the best performing have an error of 0.69% on lower bounds and 4% on upper bounds. On the MILP problems, as expected the error is bigger. When bounding the inequality MILPs, the smallest error is 39% for the lower bounds and 0% for the upper bounds. For the equality MILPs, the smallest error is 7.77% to generate lower bounds and 0% for upper bounds. For the facility location problems, the smallest lower bound error is 4.96% and the smallest lower bound error for upper bounds is 5.07%. For the unit commitment dataset, the smallest lower and upper bound error is 47.5% and 12% respectively. In addition to the bounding methods, we also introduce a spatial branch-and-bound algorithm to compute these bounds with relatively small gaps and discuss its efficiency.

Paper outline.

In Section 2, we formalise and analyse the problem at hand. In Section 3, we introduce all the bounding methods in the form of theorems and provide the associated proofs. A summary of the bounding methods is given in Table 1. Some methods work only on specific subsets of problems, like continous problems or non-negative variables. In Section 5, we introduce a new dataset of problems to serve as a benchmark and perform two experiments. First, we benchmark the different bounds on the data problems in terms of availability, precision and timing for providing upper and lower bounds. In the second experiment, we introduce a spatial branch-and-bound algorithm that uses lower and upper bounds to compute the objective function and refines the uncertainty interval to minimize the gap between the bounds.

			LP		MILP		
Approach	Type	Section	LB	UB	LB	UB	Cond.
Robust	Constant	3.1	●	✓		✓	
	Linear in λ	3.2	●	✓			
	Envelope	3.3	●	✓			
Coeff.	Constant	3.4	✓	✓	✓	✓	$\mathbf{x} \geq \mathbf{0}$
Lagrangian	Bi-segment	3.5	✓	●	✓		
	Bi-segment + coeff.	3.5	✓	●	✓		$\mathbf{x} \geq \mathbf{0}$

Table 1 Summary of all the bounding methods presented in this paper. ✓: the bounding method is available for the category. ●: available via dualisation.

Notations

We denote a scalar with a standard-font symbol a, a vector with a bold lower case symbol $\mathbf{a}$, a matrix with an uppercase letter A, a vector space with a calligraphic uppercase letter $\mathcal{A}$. The i^{th} element of a vector $\mathbf{a}$ is noted a^i. The i^{th} row and j^{th} column of a matrix A are respectively given by $\mathbf{a}^{i,\cdot}$ and $\mathbf{a}^{\cdot,j}$. The element i, j of a matrix A is given by $a^{i,j}$.

2 Problem formalisation

In this section, we provide a full formalisation of the considered problem. Let us denote by $\mathcal{P}(\lambda)$ the following optimisation problem:

$$\begin{aligned} \mathcal{P}(\lambda) \equiv \min \quad & \mathbf{c}^t\mathbf{x} \\ \text{s.t} \quad & A\mathbf{x} + \lambda D'\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \in \mathbb{R}^{n_c} \times \mathbb{Z}^{n_i}. \end{aligned}$$

Thus, with this notation, the function $f(\lambda)$ defined in (3) is the optimal objective function of $\mathcal{P}(\lambda)$ for varying λ.

We can modify this problem without losing any generality: we decompose the constraint matrix $A \in \mathbb{R}^{m \times n}$ in two matrices A_1 and A_2 of respective size $m_1 \times n$ and $m_2 \times n$, with $m_1 + m_2 = m$. The matrix A_1 encompasses all the constraints upon which there are no modifications depending on the external parameter λ, while A_2 contains the constraints affected by the modification $\lambda D'$. Similarly, we divide $\mathbf{b}$ in $\mathbf{b_1}$ and $\mathbf{b_2}$ of respective size $m_1 \times 1$ and $m_2 \times 1$. As we can have $m_1 = 0$, we indeed do not lose any generality. This results in the following equivalent problem:

$$\begin{aligned} \mathcal{P}(\lambda) \equiv \min \quad & \mathbf{c}^t\mathbf{x} \\ \text{s.t} \quad & A_1\mathbf{x} \leq \mathbf{b_1} \\ & A_2\mathbf{x} + \lambda D\mathbf{x} \leq \mathbf{b_2} \\ & \mathbf{x} \in \mathbb{R}^{n_c} \times \mathbb{Z}^{n_i}. \end{aligned} \tag{4}$$

For a fixed value $\bar{\lambda}$, an optimal solution of $\mathcal{P}(\bar{\lambda})$ is denoted $\mathbf{x}^*_{\mathcal{P}(\bar{\lambda})}$ and in the case of a purely continuous problem (i.e. $n_i = 0$) an optimal dual solution $\boldsymbol{\rho}^*_{\mathcal{P}(\bar{\lambda})}$. For the sake of readability, when explicit, we may omit the subscript of $\mathbf{x}^*_{\mathcal{P}(\bar{\lambda})}$ and $\boldsymbol{\rho}^*_{\mathcal{P}(\bar{\lambda})}$ and write $\mathbf{x}^*$ and $\boldsymbol{\rho}^*$.

The goal of this paper is to find $ub_{\lambda_1,\lambda_2}(\lambda)$ and $lb_{\lambda_1,\lambda_2}(\lambda)$, respectively an upper bound and lower bound of the optimal value on the problem written in (4), i.e. $lb_{\lambda_1,\lambda_2}(\lambda) \leq f(\lambda) \leq ub_{\lambda_1,\lambda_2}(\lambda), \forall \lambda \in [\lambda_1, \lambda_2]$. In general, $f(\lambda)$ has no desirable properties. Depending on $A_1, A_2, D, \mathbf{b_1}$ and $\mathbf{b_2}$, it is generally non-convex, non-concave and non-differentiable. As the problem may be unbounded or infeasible for some λ, it can even be non-continuous. In the following, we illustrate these behaviours with two examples.

▷ Example 2.1. Let us consider an LP problem where $A_1 = 0$, $\mathbf{b_1} = 0$, $D = -A_2$, $\mathbf{c} \neq 0$ and $\lambda \in [0, 2]$, the problem $\mathcal{P}(\lambda)$ becomes

$$\begin{aligned} \min \quad & \mathbf{c}^t\mathbf{x} \\ \text{s.t } & A_2\mathbf{x} - \lambda A_2\mathbf{x} \leq \mathbf{b_2}. \end{aligned}$$

For $\lambda = 1$, the problem is infeasible if $\mathbf{b_2} < \mathbf{0}$ or unbounded if $\mathbf{b_2} \geq \mathbf{0}$.

▷ Example 2.2. In this example, we showcase the erratic behaviour of $f(\lambda)$. Let us consider the following problem,

$$\begin{aligned} \mathcal{P}_{toy}(\lambda) \equiv \min \quad & \begin{pmatrix} 2 & -2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \\ \text{s.t} \quad & \begin{pmatrix} -2 & 2 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \leq \begin{pmatrix} 4 \\ 1 \end{pmatrix} \\ & \begin{pmatrix} 2 & 1 \\ -2 & -3 \\ 2 & 2 \\ -1 & -4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \lambda \begin{pmatrix} -1 & -4 \\ 0 & 4 \\ -4 & -3 \\ 2 & 4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \leq \begin{pmatrix} 4 \\ 2 \\ 0 \\ 2 \end{pmatrix} \\ & \forall \lambda \in [-10, 9]. \end{aligned} \tag{5}$$

We denote the associated optimal objective function by $f_{\text{toy}}(\lambda)$. Note that for this particular example we do not impose $x, y \geq 0$.

To highlight the difficulty of predicting the behaviour of the objective function of $f_{\text{toy}}(\lambda)$ through sampling and choosing the set size for sampling, we compute $f_{\text{toy}}(\lambda)$ for the same range of λ values whilst using two different levels of granularity. On one hand, we use a coarser discretisation step of 0.01 for all the optimal objectives for $\lambda \in [-10, 9]$ shown as by the blue curve in Figure 1. As we can see, the value of $f(\lambda)$ varies strongly in a neighbourhood located after $\lambda = 0.5$ and is neither concave nor convex. On the other, we use a discretisation step of 1 shown as by the orange curve on the same figure. As we can see, with this discretisation step, we would have missed the erratic behaviour of the function after 0, illustrated by the blue peak. While the difference between the two steps is rather large in the context of this toy example, this problem can also occur in much larger problems at much finer scales.

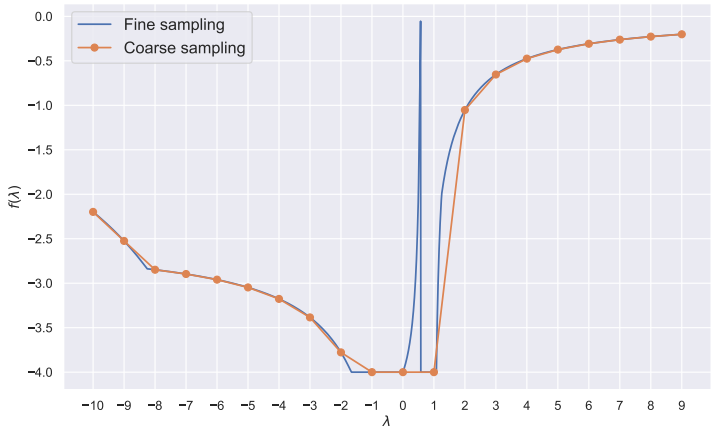


Figure 1 Plot of $f_{\text{toy}}(\lambda)$ between $[-10, 9]$. The orange line with dots is a coarse sampling of the function, made for each $\lambda \in \{-10, 9\}$, and linearly interpolated between these points.

In Figure 2, we show the evolution of the feasible space for three values of λ in the interval $[0, 1]$ where $f_{\text{toy}}(\lambda)$ shows an erratic behaviour. We see that the feasible space changes constantly due to some constraints switching from tight to non-tight and vice-versa, with small changes in λ.

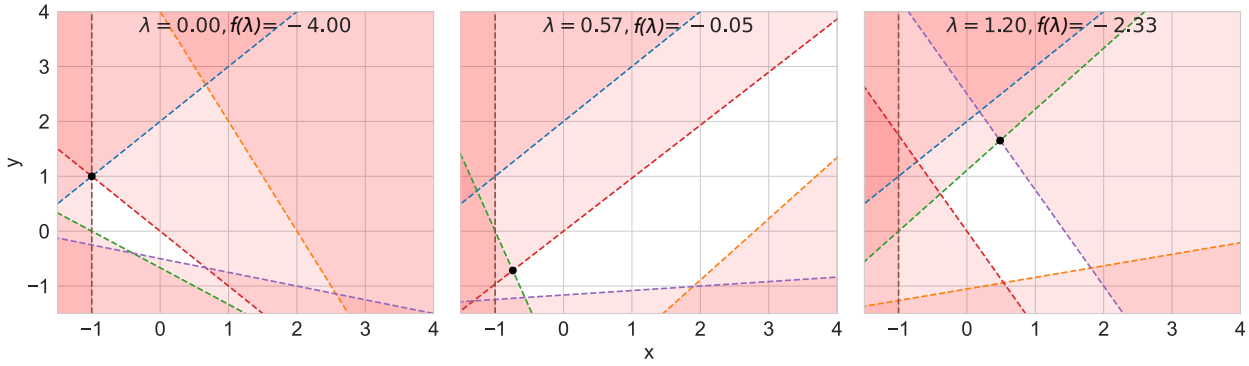


Figure 2 The feasible space of the problem given in Equation (5) for different values of λ. Each inequality partitions the space into two parts: a feasible space, coloured in white, and a non-feasible space, coloured in red. The optimum is shown as a black dot.

3 Bounding methods formulations

In this section, we show how to compute bounds on the optimal objective function $f(\lambda)$. We present our three approaches:

- Robust optimisation: we reformulate the inner problem using several robust optimisation bounding methods separated in three types:
  - one consisting of a constant solution for a given range,
  - a second solution linearly dependent on λ,
  - a third one takes a combination of affinely dependent solution, to form an envelope around the objective function.

  These techniques provide upper bounds. On LP problems, they can also be applied on the dual to provide lower bounds. The constant bound can be applied to MILP to provide upper bounds.
- Coefficient-wise relaxation and tightening: we reformulate the λD part of the modification constraints into a single matrix that do not depend on λ. This is done by considering each coefficient of the D matrix independently, hence the *coefficient-wise* name. These bounding methods provide upper and lower bounds, even if used only on the primal. They however require the variables to be nonnegative. These methods can be applied to MILP.
- Lagrangian relaxation: we dualise the constraints linked to the modification, $A_2\mathbf{x} + \lambda D\mathbf{x} \leq \mathbf{b_2}$. Depending on the choice of the Lagrangian multiplier, we get three types of this bounding method: constant, polynomially-dependent on λ, and an envelope formed from a set of λ-dependant solutions. These techniques provide lower bounds. On LP problems, they can also be applied on the dual problem to provide upper bounds. These methods can be applied to MILP by sometimes linearization to get lower bounds

These techniques are discussed in the next section on the primal. Their application to the dual problem leads to the opposite type of bound. In other words, if the technique applied on the primal gives a lower bound $lb(\lambda)$, resp. upper bound $ub(\lambda)$, its application on the dual yields an upper bound $ub(\lambda)$, resp. lower bound $lb(\lambda)$ on $f(\lambda)$.

3.1 Constant robust solution

The first approach taken consists of sacrificing the guaranteed optimality of the solution for a feasible solution on the whole interval $[\lambda_1, \lambda_2]$. Therefore, we turn to robust optimisation in order to search for a feasible solution that provides the tightest bound on this interval.

An upper bound can be obtained by finding a solution that satisfies $(A_2 + \lambda D)\mathbf{x} \leq \mathbf{b}_2 \ \forall \lambda \in [\lambda_1, \lambda_2]$. The corresponding robust optimisation problem can be written as follows and gives an upper bound on $f(\lambda)$ in the range $[\lambda_1, \lambda_2]$:

$$\begin{aligned}\mathcal{P}^{\mathrm{CR}}_{\lambda_1,\lambda_2} \equiv \underset{x}{\text{minimize}} \quad & \mathbf{c}^t\mathbf{x} \\ \text{subject to} \quad & A_1\mathbf{x} \leq \mathbf{b_1} \\ & (A_2+\lambda D)\mathbf{x} \leq \mathbf{b_2} \qquad \forall\lambda\in[\lambda_1,\lambda_2] \\ & \mathbf{x}\in\mathbb{R}^{n_c}\times\mathbb{Z}^{n_i}.\end{aligned} \tag{6}$$

This new linear reformulation of the initial problem is unusual in the sense that it possesses an infinite number of constraints, one for each $\lambda \in [\lambda_1, \lambda_2]$. It is possible to reformulate the problem in a short finite optimisation problem.

▷ Theorem 1. The following linear problem is equivalent to problem (6) and provides an upper bound for $f(\lambda)$ $\forall\lambda \in [\lambda_1, \lambda_2]$:

$$\begin{aligned}\mathcal{P}^{\mathrm{CR}}_{\lambda_1,\lambda_2} \equiv \underset{\mathbf{x}}{\text{minimize}} \quad & \mathbf{c}^t\mathbf{x} \\ \text{subject to} \quad & A_1\mathbf{x} \leq \mathbf{b_1} \\ & A_2\mathbf{x}+\lambda_1 D\mathbf{x} \leq \mathbf{b_2} \\ & A_2\mathbf{x}+\lambda_2 D\mathbf{x} \leq \mathbf{b_2} \\ & \mathbf{x}\in\mathbb{R}^{n_c}\times\mathbb{Z}^{n_i}.\end{aligned} \tag{7}$$

Proof. Any feasible solution $\mathbf{x}$ to $\mathcal{P}^{\mathrm{CR}}_{\lambda_1,\lambda_2}$ must respect the following condition:

$$(A_2+\lambda D)\mathbf{x} \leq \mathbf{b_2}\ \forall\lambda\in[\lambda_1,\lambda_2]\ . \tag{8}$$

This can be reformulated using as its robust counterpart

$$\max_{\lambda\in[\lambda_1,\lambda_2]}(A_2+\lambda D)\mathbf{x} \leq \mathbf{b_2} \tag{9}$$

which is equivalent to

$$A_2\mathbf{x} + \max_{\lambda\in[\lambda_1,\lambda_2]}\lambda D\mathbf{x} \leq \mathbf{b_2}, \tag{10}$$

where the max operator is here applied component-wise.

If we have $\mathbf{d}^{i,\cdot}\mathbf{x} \geq 0$ (resp. ≤ 0), then $\lambda = \lambda_2$ (resp. $\lambda = \lambda_1$) is an optimal solution to $\max_{\lambda\in[\lambda_1,\lambda_2]}\lambda\mathbf{d}^{i,\cdot}\mathbf{x}$. Satisfying these two cases is thus equivalent to the following set of constraints:

$$\begin{cases} A_2\mathbf{x}+\lambda_1 D\mathbf{x} \leq \mathbf{b_2} \\ A_2\mathbf{x}+\lambda_2 D\mathbf{x} \leq \mathbf{b_2} \end{cases} \tag{11}$$

For each component i of the vector $\mathbf{b_2}$, one of these constraints is redundant. Hence, Problem (6) is equivalent to (7). ◀

This reformulation has n variables and $m_1 + 2m_2$ constraints and works for MILPs. It should be noted that for two given values of λ_1 and λ_2, the feasible space respecting both $A_2\mathbf{x}+\lambda_1 D\mathbf{x} \leq \mathbf{b_2}$ and $A_2\mathbf{x}+\lambda_2 D\mathbf{x} \leq \mathbf{b_2}$ can be empty if at least two constraints are conflicting.

▷ Example 3.1. The following problem is a very simple case where constraints can be conflicting:

$$\begin{aligned}&\min_{x,y}\begin{pmatrix}-2 & -2\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix} \\ &\text{s.t.}\ \begin{pmatrix}3 & 1\\0 & -1\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix} \leq \begin{pmatrix}3\\3\end{pmatrix} \\ &\quad \begin{pmatrix}-5 & -2\\1 & 4\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix} + \lambda\begin{pmatrix}-3 & -2\\-3 & 0\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix} \leq \begin{pmatrix}0\\-3\end{pmatrix} \\ &\quad \forall\lambda\in[-2,2]\ .\end{aligned} \tag{12}$$

Its objective depending on λ is shown in Figure 3.

By applying the Theorem 1, an upper bound is given by,

$$\begin{aligned}
&\min_{x,y} \begin{pmatrix} -2 & -2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \\
&\text{s.t.} \begin{pmatrix} 3 & 1 \\ 0 & -1 \\ 1 & 2 \\ 7 & 4 \\ -11 & -6 \\ -5 & 4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \leq \begin{pmatrix} 3 \\ 3 \\ 0 \\ 0 \\ -3 \\ -3 \end{pmatrix}
\end{aligned} \tag{13}$$

We show by applying Farkas' lemma that the problem (13) is infeasible. Farkas' lemma states that for a given problem $\min c^t\mathbf{x}$ s.t. $A\mathbf{x} \leq \mathbf{b}$ if we find a vector $\mathbf{u} \geq \mathbf{0}$ such that $\mathbf{u}^t A = \mathbf{0}$ and $\mathbf{u}^t\mathbf{b} < \mathbf{0}$ then this problem is infeasible. If we consider $\mathbf{u} = [2, 0, 0, 7, 5, 0]$, we indeed have $\mathbf{u}^t A = \mathbf{0}$ and $\mathbf{u}^t\mathbf{b} = -9$.

3.2 Variable robust solution, affine on λ

A second possibility is to find solutions that vary linearly in λ. We perform a reformulation of all the variables $\mathbf{x}$ by replacing them with $\mathbf{y} + \lambda\mathbf{z}$, a linear function of λ. This transformation allows us to find a solution that adapts to a change in the value of λ.

We aim to find an upper bound $\text{ub}^{\text{rl}}_{\lambda_1,\lambda_2}(\lambda)$ on the function $f(\lambda)$ of the form $\mathbf{c}^t(\mathbf{y} + \lambda\mathbf{z})$. The *rl* superscript stands for **r**obust **l**inear. To ensure that $\text{ub}^{\text{rl}}_{\lambda_1,\lambda_2}(\lambda)$ is indeed an upper bound, the newly introduced variables $\mathbf{y}$ and $\mathbf{z}$ must satisfy the following conditions:

$$A_1(\mathbf{y} + \lambda\mathbf{z}) \leq \mathbf{b_1} \qquad \forall\lambda \in [\lambda_1, \lambda_2] \tag{14}$$

$$(A_2 + \lambda D)(\mathbf{y} + \lambda\mathbf{z}) \leq \mathbf{b_2} \qquad \forall\lambda \in [\lambda_1, \lambda_2] \tag{15}$$

$$\mathbf{y} + \lambda\mathbf{z} \in \mathbb{R}^{n_c} \times \mathbb{Z}^{n_i} \qquad \forall\lambda \in [\lambda_1, \lambda_2] \tag{16}$$

The conditions (16) are complex to satisfy if the problem is mixed-integer. In the remaining of this section and the following one, we thus focus on linear problems, i.e. with $n_i = 0$, and the last set of constraints simplifies to $\mathbf{y}, \mathbf{z} \in \mathbb{R}^{n_c}$.

As in the previous case, we can rewrite the remaining infinite number of constraints to a finite albeit more constrained form and obtain the following Theorem.

▷ Theorem 2. Any $\mathbf{y}$ and $\mathbf{z}$ satisfying the constraints

$$\begin{cases}
A_1\mathbf{y} + \lambda_1 A_1\mathbf{z} \leq \mathbf{b_1} \\
A_1\mathbf{y} + \lambda_2 A_1\mathbf{z} \leq \mathbf{b_1} \\
(A_2 + \lambda_1 D)(\mathbf{y} + \lambda_1\mathbf{z}) \leq \mathbf{b_2} \\
(A_2 + \lambda_2 D)(\mathbf{y} + \lambda_2\mathbf{z}) \leq \mathbf{b_2} \\
A_2\mathbf{y} + D\mathbf{z}\lambda_1\lambda_2 + (D\mathbf{y} + A_2\mathbf{z})\frac{\lambda_1+\lambda_2}{2} \leq \mathbf{b_2}
\end{cases} \tag{17}$$

also satisfy both equations (14) and (15).

If $\mathcal{P}(\lambda)$ is a linear problem ($n_i = 0$), any $\mathbf{y}$ and $\mathbf{z}$ satisfying these conditions provide an upper bound in the form $\text{ub}^{\text{rl}}_{\lambda_1,\lambda_2}(\lambda) = \mathbf{c}^t(\mathbf{y} + \lambda\mathbf{z}) \geq f(\lambda)\ \forall\lambda \in [\lambda_1, \lambda_2]$.

Proof. We write the robust counterpart in order to obtain a finite number of constraints. Let us first focus on constraints (14):

$$A_1(\mathbf{y} + \lambda\mathbf{z}) \leq \mathbf{b_1}\ \forall\lambda \in [\lambda_1, \lambda_2] \tag{18}$$

which can be rewritten as

$$\max_{\lambda\in[\lambda_1,\lambda_2]} A_1(\mathbf{y} + \lambda\mathbf{z}) \leq \mathbf{b_1} \tag{19}$$

and

$$A_1\mathbf{y} + \max_{\lambda\in[\lambda_1,\lambda_2]} \lambda A_1\mathbf{z} \leq \mathbf{b_1} \tag{20}$$

As the max operator is applied component-wise, for every constraint in $\max_{\lambda\in[\lambda_1,\lambda_2]} \lambda A_1\mathbf{z}$, the maximum is either achieved for $\lambda = \lambda_1$ or $\lambda = \lambda_2$.

$$A_1\mathbf{y} + \max_{\lambda\in[\lambda_1,\lambda_2]} \lambda A_1\mathbf{z} \leq \mathbf{b_1} \equiv \begin{cases} A_1\mathbf{y} + \lambda_1 A_1\mathbf{z} \leq \mathbf{b_1} \\ A_1\mathbf{y} + \lambda_2 A_1\mathbf{z} \leq \mathbf{b_1} \end{cases}. \tag{21}$$

Hence, any $\mathbf{x}$ satisfying (21) also satisfies (14) and conversely. The case of (15) is more complex as its left-hand side is a quadratic function of λ.

$$(A_2 + \lambda D)(\mathbf{y} + \lambda\mathbf{z}) = A_2\mathbf{y} + \lambda(D\mathbf{y} + A_2\mathbf{z}) + \lambda^2 D\mathbf{z} \leq \mathbf{b_2}\ \forall\lambda \in [\lambda_1, \lambda_2] \tag{22}$$

Again, we rewrite it using the robust counterpart as a maximization

$$\max_{\lambda\in[\lambda_1,\lambda_2]} A_2\mathbf{y} + \lambda(D\mathbf{y} + A_2\mathbf{z}) + \lambda^2 D\mathbf{z} \leq \mathbf{b_2} \tag{23}$$

$$:= \max_{\lambda\in[\lambda_1,\lambda_2]} g(\lambda) \leq \mathbf{b_2}. \tag{24}$$

The function $g(\lambda)$ is a quadratic function of λ. The maximum of the quadratic equation can be found analytically but is itself non-linear in z. Instead of directly using $g(\lambda)$, we propose using a piecewise-linear upper bound of this function in the above condition; this allows us to relax (15) into a linear program in $\mathbf{y}$ and $\mathbf{z}$.

For this, we can use the following lemma:

▷ Lemma 1. Given a quadratic function $q(x) = ax^2 + bx + c$. The maximum of $q(x)$ over $x_1 \leq x \leq x_2$ is upper bounded by

$$\max \begin{cases} ax_1^2 + bx_1 + c \\ ax_2^2 + bx_2 + c \\ ax_1x_2 + b\frac{x_1+x_2}{2} + c. \end{cases}$$

This lemma is proved in Annex A.1. We apply the lemma on $g(\lambda)$ and we obtain that

$$\max_{\lambda\in[\lambda_1,\lambda_2]} g(\lambda) \leq \max \begin{cases} (A_2 + \lambda_1 D)(\mathbf{y} + \lambda_1\mathbf{z}) \\ (A_2 + \lambda_2 D)(\mathbf{y} + \lambda_2\mathbf{z}) \\ A_2\mathbf{y} + D\mathbf{z}\lambda_1\lambda_2 + (D\mathbf{y} + A_2\mathbf{z})\frac{\lambda_1+\lambda_2}{2} \end{cases} \tag{25}$$

Imposing the following is thus sufficient to obtain $\max_{\lambda\in[\lambda_1,\lambda_2]} g(\lambda) \leq \mathbf{b_2}$:

$$\begin{cases} (A_2 + \lambda_1 D)(\mathbf{y} + \lambda_1\mathbf{z}) \leq \mathbf{b_2} \\ (A_2 + \lambda_2 D)(\mathbf{y} + \lambda_2\mathbf{z}) \leq \mathbf{b_2} \\ A_2\mathbf{y} + D\mathbf{z}\lambda_1\lambda_2 + (D\mathbf{y} + A_2\mathbf{z})\frac{\lambda_1+\lambda_2}{2} \leq \mathbf{b_2} \end{cases}. \tag{26}$$

The conditions (21) and (26) are thus sufficient for $\mathbf{y}, \mathbf{z}$ to respect constraints (14) and (15). ◀

Note that the converse is not true: the set of constraints (17) are only a sufficient condition for y and z to be an affine robust solutions. We can derive an upper bound for $f(\lambda)$ by using Theorem 2.

▷ Corollary 1. Consider the set

$$\begin{aligned} S^{AR}_{\lambda_1,\lambda_2} = \{ \quad & \mathbf{y}, \mathbf{z} \in \mathbb{R}^n \mid A_1\mathbf{y} + \lambda_1 A_1\mathbf{z} \leq \mathbf{b_1} \\ & A_1\mathbf{y} + \lambda_2 A_1\mathbf{z} \leq \mathbf{b_1} \\ & (A_2 + \lambda_1 D)(\mathbf{y} + \lambda_1\mathbf{z}) \leq \mathbf{b_2} \\ & (A_2 + \lambda_2 D)(\mathbf{y} + \lambda_2\mathbf{z}) \leq \mathbf{b_2} \\ & A_2\mathbf{y} + D\mathbf{z}\lambda_1\lambda_2 + (D\mathbf{y} + A_2\mathbf{z})\frac{\lambda_1+\lambda_2}{2} \leq \mathbf{b_2} \quad \}. \end{aligned} \tag{27}$$

For any $(\mathbf{y}, \mathbf{z}) \in S^{AR}$, the function $h^{AR}(\lambda) = c^T(\mathbf{y} + \lambda\mathbf{z}) \geq f(\lambda)$.

When it is clear from the context, we may refer to $S^{AR}_{\lambda_1,\lambda_2}$ as simply S^{AR}. There can be a large number of $\mathbf{y}$ and $\mathbf{z}$ in the set S^{AR} from Corollary 1. The question is to select them in order to have the tightest possible bound. We may fix λ to a specific value $\mu \in [\lambda_1, \lambda_2]$ and obtain the tightest possible bound by optimizing the problem

$$\mathcal{P}^{AR}_{\lambda_1,\lambda_2}(\mu) \equiv \min c^t(\mathbf{y} + \mu z) \text{ subject to } (\mathbf{y}, \mathbf{z}) \in S^{AR}_{\lambda_1,\lambda_2}. \tag{28}$$

An optimal solution $(\mathbf{y}, \mathbf{z})$ to $\mathcal{P}^{AR}_{\lambda_1,\lambda_2}(\mu)$ provides the upper bound $h^{AR}_{\mathbf{y},\mathbf{z}}(\lambda)$ over the entire interval $[\lambda_1, \lambda_2]$ but there is no guarantee that, by doing so, we obtain a tight bound for the rest of the interval outside of the specific value μ. In the next paragraph, we describe strategies to derive a tight bound.

Ways to select y and z

There are several degrees of freedom in selecting $\mathbf{y}$ and $\mathbf{z}$ from S^{AR} because it doubles the size of the space compared to the initial variables $\mathbf{x}$. For instance, any constant robust solution $\mathbf{x}$ can be mapped to a solution in S^{AR}.

▷ Theorem 3. Any constant robust solution $\mathbf{x}$ provides a solution $\mathbf{y} = \mathbf{x}, \mathbf{z} = \mathbf{0}$ for eq. (17).

Proof. Enforcing $\mathbf{z} = \mathbf{0}$ in eq. (17) provides the same equations as eq. (1) defining the space of constant robust solution (after renaming $\mathbf{x}$ to $\mathbf{y}$), with the following additional constraint:

$$A_2\mathbf{y} + \frac{\lambda_1 + \lambda_2}{2} D\mathbf{y} \leq \mathbf{b}_2. \tag{29}$$

This constraint is redundant with the other constraints (it is a linear combination of the two constraints related to A_2), hence the two solution spaces are the same. ◀

Of course, many more solutions may exist with $\mathbf{z} \neq 0$ and with different values of $\mathbf{y}$. Most of these solutions will provide upper bounds that are actually not very tight. It is thus important to select *suitable* $\mathbf{y}$ and $\mathbf{z}$. We propose four ways to select $\mathbf{y}$ and $\mathbf{z}$. Two of these ways are based on the sequential optimisation of two objectives. The third and fourth introduce an additional constraint.

- The first method (called the *left* linear robust solution or *robust line left* in the experiments) requires two optimisations. It first optimises $\mathcal{P}^{AR}_{\lambda_1,\lambda_2}(\lambda_1)$, with optimal value v. It then solves $\mathcal{P}^{AR}_{\lambda_1,\lambda_2}(\lambda_2)$ with the additional constraint $\mathbf{c}^t(\mathbf{y} + \lambda_1\mathbf{z}) = v$ whose optimal solution is $\mathbf{y}^l, \mathbf{z}^l$. The bound $h^{AR}_{\mathbf{y}^l,\mathbf{z}^l}(\lambda)$ is the tightest around λ_1 and among those, the one that has the lowest possible slope.
- Similarly, the second method works in the same manner but in reverse order, and is called the *right* linear robust solution or *robust line right* in the experiments. It first optimises $\mathcal{P}^{AR}_{\lambda_1,\lambda_2}(\lambda_2)$ with optimal value w, then solves $\mathcal{P}^{AR}_{\lambda_1,\lambda_2}(\lambda_1)$ with the additional constraint $\mathbf{c}^t(\mathbf{y} + \lambda_2\mathbf{z}) = w$.
- The third method adds the constraint $\mathbf{c}^t\mathbf{z} = \delta$ to the problem $\mathcal{P}^{AR}_{\lambda_1,\lambda_2}(\lambda_1)$ to force the variable slope to be δ and optimises the objective $\mathbf{c}^t\mathbf{y}$ (as $\mathbf{c}^t\mathbf{z}$ is now constant). In the experiments, we fix the slopes to two values. The first one is $\delta = \frac{f(\lambda_2)-f(\lambda_1)}{\lambda_2-\lambda_1}$, called robust fixed slope pairwise. The second is $\delta = 0$, called robust yzflat. However, any δ can be chosen. Note that in the case $\delta = 0$, the solution $\mathbf{y} + \lambda\mathbf{z}$ is not necessarily "flat" in the solution space (i.e. $\mathbf{z}$ may not be $\mathbf{0}$) as illustrated in the Example 3.2.

▷ Example 3.2. The problem presented in eq. (12) has no constant robust solution over the range $[-2, 2]$ (see Example 3.1). It has, however, variable robust solutions (even a flat one), as shown in Figure 3.

Figure 3 Bounds obtained with the different robust variable solution for the problem (12).

Relation with the constant robust solution

Example 3.2 above shows that even when the bound is selected to be flat ($\mathbf{c}^t\mathbf{z} = 0$), it can exists even when the constant robust solution (as presented in Section 3.1) does not. Moreover, any constant robust solution maps to a variable one as shown in Theorem 3.

As a corollary, this implies that the bounds provided by the variable solutions are tighter (and exist more often) than the constant ones:

▷ Corollary 2. The (flat) variable robust solution $\mathbf{y} + \lambda\mathbf{z}$ respecting eq. (17) with $\mathbf{c}^t\mathbf{z} = 0$ and minimizing $\mathbf{c}^t(\mathbf{y} + \lambda\mathbf{z}) = \mathbf{c}^t\mathbf{y}$ provides a better lower bound than the constant robust solution $\mathbf{x}$ minimizing $\mathbf{c}^t\mathbf{x}$ if both exist:

$$\mathbf{c}^t(\mathbf{y} + \lambda\mathbf{z}) = \mathbf{c}^t\mathbf{y} \leq \mathbf{c}^t\mathbf{x}. \tag{30}$$

3.3 Envelope of robust linear solutions

In the previous section, we proved that, on linear problems, we can find a robust variable solution with several possible slopes. In this section, we introduce a bound tightness criterion to find the interval $[\nu_1, \nu_2]$ upon which a given optimal (for a certain value of $\lambda = \nu$) robust variable solution $\mathbf{y} + \lambda\mathbf{z}$ remains optimal over the interval, i.e. on that particular interval, we cannot find another robust variable solution whose objective function improves. From that criterion, coupled with warm starting, we describe a corresponding bound and discuss its strengths and weaknesses.

Consider eq. (28) with $\mu = \lambda$, $\mathcal{P}^{AR}_{\lambda_1,\lambda_2}(\lambda)$. An interesting observation is that λ is now only present in the objective function; we have reduced a generic problem where modifications are applied to the constraints' coefficients to a simpler problem where the modifications are only located on the objective function coefficients. In principle, we would have to solve this optimization problem for every value of λ. In the following, we show how to do it without reoptimising for an infinite number of values of λ.

Assume that we find for a specific value $\nu \in [\lambda_1, \lambda_2]$, an optimal solution $(\mathbf{y}_\nu, \mathbf{z}_\nu)$ leading to a bound $h^{AR}_{\mathbf{y}_\nu,\mathbf{z}_\nu}(\lambda)$. Techniques based on reduced costs can be used to find the interval $[\nu_1, \nu_2]$ where $(\mathbf{y}_\nu, \mathbf{z}_\nu)$ remains optimal for all problems $\mathcal{P}^{AR}_{\lambda_1,\lambda_2}(\lambda)$ with $\lambda \in [\nu_1, \nu_2]$.

In order to do this, and to simplify part of the proofs below, we rewrite the problem $\mathcal{P}^{AR}_{\lambda_1,\lambda_2}(\lambda)$ by creating explicit slack variables and putting all the constraints into a single matrix M and the right-hand side in a vector $\mathbf{E}$:

$$\begin{aligned} \mathcal{P}^{AR}_{\lambda_1,\lambda_2}(\lambda) \equiv \underset{\mathbf{y},\mathbf{z}}{\text{minimize}} \quad & \begin{pmatrix}\mathbf{c}^t & \lambda\mathbf{c}^t & \mathbf{0}\end{pmatrix}\begin{pmatrix}\mathbf{y}\\ \mathbf{z}\\ \mathbf{s}\end{pmatrix} \\ \text{subject to} \quad & M\begin{pmatrix}\mathbf{y}\\ \mathbf{z}\\ \mathbf{s}\end{pmatrix} = \mathbf{E}, \\ & \mathbf{s} \geq \mathbf{0}. \end{aligned} \tag{31}$$

We use this formulation to derive the associated theorem.

▷ Theorem 4 (Robust bound tightness criterion). Given $\nu \in [\lambda_1, \lambda_2]$, let $\langle \mathbf{y}^*, \mathbf{z}^*, \mathbf{s}^* \rangle$ be an optimal basic solution of the problem $\mathcal{P}^{AR}_{\lambda_1,\lambda_2}(\nu)$ and let B and N be the basic and non-basic sets of variables associated to the solution. Additionally, let $\mathbf{r}$ be the vector of reduced costs. For any λ such that

$$(\lambda - \nu)\left(\begin{pmatrix}\mathbf{0} & \mathbf{c}^T_{N_z} & \mathbf{0}\end{pmatrix} - \begin{pmatrix}\mathbf{0} & \mathbf{c}^T_{B_z} & \mathbf{0}\end{pmatrix} M_B^{-1} M_N\right) \geq -\mathbf{r}, \tag{32}$$

$\langle \mathbf{y}^*, \mathbf{z}^*, \mathbf{s}^* \rangle$ is an optimal solution to $\mathcal{P}^{AR}_{\lambda_1,\lambda_2}(\lambda)$, where $\mathbf{c}^T_{B_z}$ (resp. $\mathbf{c}^T_{N_z}$) is the vector $\mathbf{c}^T$ restricted to the variables in B_z (resp. N_z), itself the subset of $\mathbf{z}$ variables in B (resp. N).

Proof. The reduced costs for the basis related to $\langle \mathbf{y}^*, \mathbf{z}^*, \mathbf{s}^* \rangle$ on the problem $\mathcal{P}^{AR}_{\lambda_1,\lambda_2}(\nu)$ are

$$\mathbf{r} := \begin{pmatrix}\mathbf{c}^T_{N_y} & \nu\mathbf{c}^T_{N_z} & \mathbf{0}\end{pmatrix} - \begin{pmatrix}\mathbf{c}^T_{B_y} & \nu\mathbf{c}^T_{B_z} & \mathbf{0}\end{pmatrix} M_B^{-1} M_N \tag{33}$$

with $\mathbf{r} \geq \mathbf{0}$, as the solution is optimal. If we now consider the same basis on the sightly modified problem $\mathcal{P}^{AR}_{\lambda_1,\lambda_2}(\nu+\delta)$, we obtain that the new reduced costs are:

$$\begin{pmatrix}\mathbf{c}^T_{Ny} & (\nu+\delta)\mathbf{c}^T_{Nz} & \mathbf{0}\end{pmatrix} - \begin{pmatrix}\mathbf{c}^T_{By} & (\nu+\delta)\mathbf{c}^T_{Bz} & \mathbf{0}\end{pmatrix} M_B^{-1} M_N \tag{34}$$

$$= \mathbf{r} + \begin{pmatrix}\mathbf{0} & \delta\mathbf{c}^T_{Nz} & \mathbf{0}\end{pmatrix} - \begin{pmatrix}\mathbf{0} & \delta\mathbf{c}^T_{Bz} & \mathbf{0}\end{pmatrix} M_B^{-1} M_N \tag{35}$$

For the basis to stay optimal for the given δ, these reduced costs must be nonnegative. Therefore, we have that

$$\begin{pmatrix}\mathbf{0} & \delta\mathbf{c}^T_{Nz} & \mathbf{0}\end{pmatrix} - \begin{pmatrix}\mathbf{0} & \delta\mathbf{c}^T_{Bz} & \mathbf{0}\end{pmatrix} M_B^{-1} M_N \geq -\mathbf{r}$$

If we pose $\delta = \lambda - \nu$, we obtain the conditions stated initially. ◀

Derived bounding method

Theorem 4 allows us to find the interval $[\nu_1, \nu_2]$ for which a given robust linear (basic) solution $(\mathbf{y}, \mathbf{z})$ remains optimal for $\mathcal{P}^{AR}_{\lambda_1,\lambda_2}(\lambda)$ for all $\lambda \in [\nu_1, \nu_2]$. Once the basic solution is known, computing the range of $[\nu_1, \nu_2]$ for which it stays optimal can be done in $\mathcal{O}(m^2 + mn)$ where n is the number of variables in the (full) problem and m the number of constraints. Most of the operations are indeed vector-matrix multiplications and rely on standard warm starting techniques.

This procedure paves the way to compute a so-called "envelope" of robust linear solutions for linear problems, i.e. an envelope containing the best combination of lower and upper linear robust bounds, in a fast way using simplex warm starting. For minimization problems, the upper bounds give together a concave envelope, while the lower bounds give a convex one. Their combination form a convex space.

Given an interval $[\lambda_1, \lambda_2]$, the steps to compute this envelope are given as follows:

- First, we solve $\mathcal{P}^{AR}_{\lambda_1,\lambda_2}(\lambda_1)$ and retrieve an optimal basic solution $\langle \mathbf{y}_{\lambda_1}, \mathbf{z}_{\lambda_1}, \mathbf{s}_{\lambda_1} \rangle$with its associated basis.
- Second, we compute the largest value of δ for which the current solution is still optimal, i.e. by using Theorem 4 we take the largest δ such that the reduced costs of $\mathcal{P}^{AR}_{\lambda_1,\lambda_2}(\lambda_1 + \delta)$ are nonnegative.
- Third, we use warm starting methods to reoptimise the problem $\mathcal{P}^{AR}_{\lambda_1,\lambda_2}(\lambda_1 + \delta + \epsilon)$. We get a new optimal basis and new optimal solution.
- Fourth, we repeat the three previous steps until λ_2 is reached.
- The algorithm finishes and produces a finite number of solutions (as the polytope of the problem has a finite number of vertices), which all taken together form a concave upper bound (for minimization problems) or a convex lower bound (for maximization ones) for the range.

In Figure 4, we illustrate the algorithm on both upper and lower bounds. For the lower bound, the algorithm first finds the linear bound in purple that minimizes the problem for $\lambda = -4$ and its equivalent basis. Then iteratively, it updates the next basis, found by using the optimality criterion, and reoptimises for the several values of λ obtaining the bound: pink, grey, yellow and finally light blue in that order. For each reoptimisation, the previous basis is used, upon it, a few simplex iterations are performed to find the new optimal objective and basis.

The main drawback of the method resides in the need for an optimal basis whereas the other methods only need an optimal solution. Similarly to the robust variable solution, this method reformulates the problem using $2n$ variables and $2m_1 + 3m_2$ constraints. Finding an optimal basis would require the use of either the simplex algorithm or interior points methods with crossover enabled. This operation may take a lot of time. However, if an optimal basis is known, the method is fast and accurate. In the experiments this bound is named the robust envelope.

3.4 Coefficient-wise bounding methods

In the previous methods, and especially in the case of the robust constant solution, we look for a robust feasible solution that is valid for the problem with infinitely many constraints. The drawback of that method (and also of the affine robust solution) is computational. Indeed we double the number of constraints that are affected by D. The good point is that we keep a linear program, but that linear program is itself computationally more demanding than the nominal linear program. In this section, we propose two bounding methods : one relaxation (providing a lower bound) and one restriction (providing an upper bound) that keep the size of the nominal

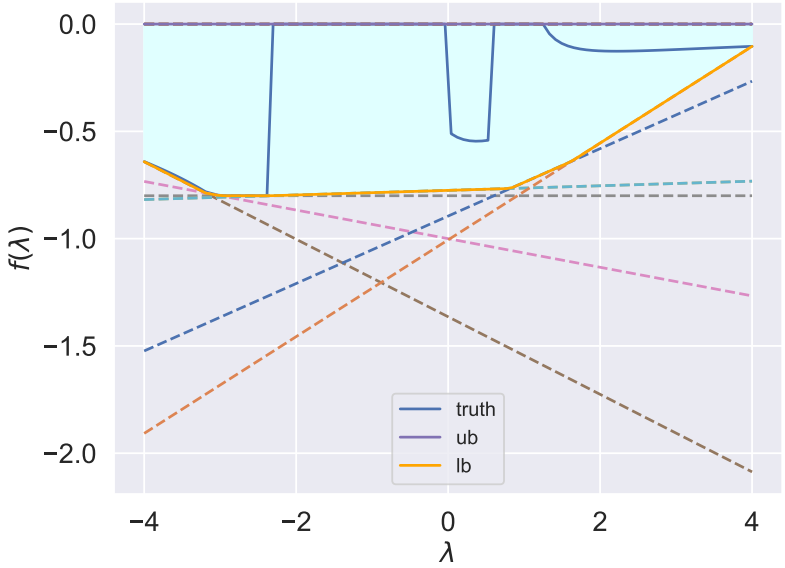


Figure 4 Illustration of the robust envelope algorithm on problem $\mathcal{P}_{\text{toy 1}}$ (see Annex A.2)

linear program. The essential idea is to relax or restrict every coefficient that is potentially modified by the parameter λ. Throughout this section we need that additional assumption that the variables are all nonnegative. Observe that we can transform free variables into the difference of two nonnegative variables if we want to apply the techniques presented in this section to problems with free variables.

To make things clear, we consider the problem

$$\begin{aligned} \mathcal{P}^{\geq 0}(\lambda) \equiv \min \quad & \mathbf{c}^t\mathbf{x} \\ \text{s.t} \quad & A_1\mathbf{x} \leq \mathbf{b_1} \\ & A_2\mathbf{x} + \lambda D\mathbf{x} \leq \mathbf{b_2} \\ & \mathbf{x} \in \mathbb{R}_+^{n_c} \times \mathbb{Z}_+^{n_i}. \end{aligned} \tag{36}$$

First, a tool to define the corresponding relaxation and restriction is defined.

▷ Definition 1. Consider a linear function $a\lambda$ of the variable λ and the fixed interval $[\lambda_1, \lambda_2]$. Let us define $(a\lambda)^\uparrow$ and $(a\lambda)^\downarrow$ as

$$(a\lambda)^\uparrow = \max_{\lambda\in[\lambda_1,\lambda_2]}(a\lambda) = \begin{cases} a\lambda_1 & \text{if } a \leq 0 \\ a\lambda_2 & \text{otherwise} \end{cases} \qquad (a\lambda)^\downarrow = \min_{\lambda\in[\lambda_1,\lambda_2]}(a\lambda) = \begin{cases} a\lambda_1 & \text{if } a \geq 0 \\ a\lambda_2 & \text{otherwise} \end{cases}. \tag{37}$$

▷ Theorem 5. The problem

$$\begin{aligned} \mathcal{P}^{\sqsubset}(\lambda) \equiv \min \quad & \mathbf{c}^t\mathbf{x} \\ \text{s.t} \quad & A_1\mathbf{x} \leq \mathbf{b_1} \\ & \sum_{j=1}^{n} a_{ij}^{(2)} x_j + \sum_{j=1}^{n} (\lambda d_{ij})^\uparrow x_j \leq \mathbf{b_2} \qquad \forall i \\ & \mathbf{x} \in \mathbb{R}_+^{n_c} \times \mathbb{Z}_+^{n_i} \end{aligned} \tag{38}$$

is a restriction of $\mathcal{P}^{\geq 0}(\lambda)$ for all $\lambda \in [\lambda_1, \lambda_2]$. Hence, its optimal value is an upper bound of the optimal value of $\mathcal{P}^{\geq 0}(\lambda)$ for all $\lambda \in [\lambda_1, \lambda_2]$. Similarly, the problem

$$\begin{aligned} \mathcal{P}^{\sqsupset}(\lambda) \equiv \min \quad & \mathbf{c}^t\mathbf{x} \\ \text{s.t} \quad & A_1\mathbf{x} \leq \mathbf{b_1} \\ & \sum_{j=1}^{n} a_{ij}^{(2)} x_j + \sum_{j=1}^{n} (\lambda d_{ij})^\downarrow x_j \leq \mathbf{b_2} \qquad \forall i \\ & \mathbf{x} \in \mathbb{R}_+^{n_c} \times \mathbb{Z}_+^{n_i} \end{aligned} \tag{39}$$

is a relaxation of $\mathcal{P}^{\geq 0}(\lambda)$ for all $\lambda \in [\lambda_1, \lambda_2]$. Hence, its optimal value is a lower bound of the optimal value of $\mathcal{P}^{\geq 0}(\lambda)$ for all $\lambda \in [\lambda_1, \lambda_2]$.

Proof. (sketch) The proof relies on the fact that, for constraints involving only nonnegative variables, increasing all coefficients provides a restriction, whereas decreasing them provides a relaxation. ◀

▷ Theorem 6. In terms of bound tightness, the restricted version of this bound is always more restrictive (less tight) than the robust flat bound.

Proof. The robust flat bound for nonnegative problems can be rewritten as

$$\begin{aligned}
\min \quad & \mathbf{c}^t\mathbf{x} \\
\text{s.t} \quad & A_1\mathbf{x} \leq \mathbf{b_1} \\
& \sum_{j=1}^{n} a_{ij}^{(2)} x_j + \max_{\lambda\in[\lambda_1,\lambda_2]} (\sum_{j=1}^{n} \lambda d_{ij} x_j) \leq \mathbf{b_2} \qquad \forall i \\
& \mathbf{x} \in \mathbb{R}_+^{n_c} \times \mathbb{Z}_+^{n_i}
\end{aligned}$$

which is reformulated as

$$\begin{aligned}
\min \quad & \mathbf{c}^t\mathbf{x} \\
\text{s.t} \quad & A_1\mathbf{x} \leq \mathbf{b_1} \\
& \sum_{j=1}^{n} a_{ij}^{(2)} x_j + \sum_{j=1}^{n} \lambda_1 d_{ij} x_j \leq \mathbf{b_2} \qquad \forall i \\
& \sum_{j=1}^{n} a_{ij}^{(2)} x_j + \sum_{j=1}^{n} \lambda_2 d_{ij} x_j \leq \mathbf{b_2} \qquad \forall i \\
& \mathbf{x} \in \mathbb{R}_+^{n_c} \times \mathbb{Z}_+^{n_i}.
\end{aligned}$$

As we have that $\max_{\lambda\in[\lambda_1,\lambda_2]}(\sum_{j=1}^{n} \lambda d_{ij} x_j) \leq \sum_{j=1}^{n} \max_{\lambda\in[\lambda_1,\lambda_2]} \lambda d_{ij} x_j, \forall i$, the terms

$$\begin{aligned}
& \sum_{j=1}^{n} a_{ij}^{(2)} x_j + \sum_{j=1}^{n} (\lambda d_{ij})^{\uparrow} x_j \geq \sum_{j=1}^{n} a_{ij}^{(2)} x_j + \sum_{j=1}^{n} \lambda_1 d_{ij} x_j \\
\text{and } & \sum_{j=1}^{n} a_{ij}^{(2)} x_j + \sum_{j=1}^{n} (\lambda d_{ij})^{\uparrow} x_j \geq \sum_{j=1}^{n} a_{ij}^{(2)} x_j + \sum_{j=1}^{n} \lambda_2 d_{ij} x_j, \forall i
\end{aligned}$$

making it harder for the constraint to be respected therefore being more restrictive. ◀

However, in terms of computation, this bound keeps the same number of constraints and variables as the nominal problem, making it faster to compute than the robust flat bound.

3.5 Lower bounds using Lagrangian relaxations

In this section, we propose two lower bounds based on Lagrangian relaxations of (4). Let us fix $\boldsymbol{\rho} \in \mathbb{R}_-^{m_2}$ and define the problem $\mathcal{P}^L(\boldsymbol{\rho}, \lambda)$ as the Lagrangian relaxation of $\mathcal{P}(\lambda)$, obtained by dualizing the second set of constraints, with dual multipliers $\boldsymbol{\rho} \leq 0$:

$$\begin{aligned}
\mathcal{P}^L(\boldsymbol{\rho}, \lambda) \equiv \underset{x}{\text{minimize}} \quad & \mathbf{c}^t\mathbf{x} - \boldsymbol{\rho}^t((A_2 + \lambda D)\mathbf{x} - \mathbf{b_2}) \\
\text{subject to} \quad & A_1\mathbf{x} \leq \mathbf{b_1} \\
& \mathbf{x} \in \mathbb{R}^{n_c} \times \mathbb{Z}^{n_i}.
\end{aligned} \tag{40}$$

The optimal objective of $\mathcal{P}^L(\boldsymbol{\rho}, \lambda)$ is denoted by $h^L(\boldsymbol{\rho}, \lambda)$ and provides a lower bound to $f(\lambda)$ for every λ. In order to define the bounding methods, let us first prove some properties of $h^L(\boldsymbol{\rho}, \lambda)$. It is well-known that, for a fixed λ, the function $h^L(\boldsymbol{\rho}, \lambda)$ is concave. In the following lemma, we show that, for a fixed $\boldsymbol{\rho}$, the function is also concave.

▷ Lemma 2. Assume that $\boldsymbol{\rho} \leq 0$ is fixed. The function $h(\boldsymbol{\rho}, \lambda)$ is defined over a convex set $C \subseteq [\lambda_1, \lambda_2]$ and concave in λ over that set.

Proof. Consider a fixed $\boldsymbol{\rho} \leq 0$. Observe that the problem (40) is now back to a linear program with an affinely varying cost vector in λ. That is also the case if the variables are mixed integer as we can then replace the constraints $A\mathbf{x} \leq \mathbf{b}_1$ by $\text{conv}(\{\mathbf{x} \in \mathbb{R}^{n_c} \times \mathbb{Z}^{n_i} \mid A\mathbf{x} \leq \mathbf{b}_1\})\}$. The resulting value function is concave (see for example [4], Theorem 5.3 p217). This follows from the fact that, the feasible region of (40) is the same polyhedron, whatever the value of λ, and therefore the optimal value is obtained as the minimum over all extreme points of the polyhedron, resulting in the minimum of a finite number of affine functions, which is concave. The set on which the value function is not $-\infty$ is itself convex. ◀

In practice, it is complicated to determine the right Lagrange multipliers $\boldsymbol{\rho}$ in order to obtain the tightest possible lower bounds. Our first suggestion is to use, as Lagrange multipliers the values that are obtained for the dual problem for some specific values of λ. For example, if the initial problem does not include integer variables, we can solve the linear problem $\mathcal{P}(\lambda_1)$ which typically comes with optimal dual variables from which we retrieve the dual variables associated with the constraints dualized in the Lagrangian relaxation, we denote them by $\boldsymbol{\rho}^*_{\lambda_1}$. Similarly, we may assume that we compute $\boldsymbol{\rho}^*_{\lambda_2}$. To ease notation, we need the following definition.

▷ Definition 2. Consider two pairs of real numbers (x_1, y_1) and (x_2, y_2), we denote the linear interpolation between theses two pairs of points as

$$\ell[(x_1, x_2), (y_1, y_2)](t) = y_1 + (t - x_1)\frac{y_2 - y_1}{x_2 - x_1}.$$

In order to derive the first bounding method using the Lagrangian relaxation (40), we can use a specific value of $\boldsymbol{\rho}$ through the following result.

▷ Lemma 3. Consider $\boldsymbol{\rho} \leq 0$ fixed. For every $\lambda \in [\lambda_1, \lambda_2]$, a lower bound to (40) is given by

$$\ell[(\lambda_1, \lambda_2), (h^L(\boldsymbol{\rho}, \lambda_1), h^L(\boldsymbol{\rho}, \lambda_2))](\lambda) \leq h^L(\boldsymbol{\rho}, \lambda) \leq f(\lambda). \tag{41}$$

Proof. This follows directly from the concavity of $h^L(\boldsymbol{\rho}, \lambda)$ stated in Lemma 2 for fixed $\boldsymbol{\rho}$, and the fact that any line segment joining two points on the graph of a concave function is a lower bound to that function. ◀

We can now use Lemma 3, with specific values of $\boldsymbol{\rho}$, in the case with no integer variables.

▷ Theorem 7. Assume that the initial problem (4) has no integer variables, i.e. $n_i = 0$. Let $\boldsymbol{\rho}^*_{\lambda_1}$ (resp. $\boldsymbol{\rho}^*_{\lambda_2}$) be the optimal dual variables related to constraints $A_2\boldsymbol{x} + \lambda_1 D\boldsymbol{x} \leq \boldsymbol{b}_2$ (resp. $A_2\boldsymbol{x} + \lambda_2 D\boldsymbol{x} \leq \boldsymbol{b}_2$) in (4) when the initial problem (4) is solved optimally for $\lambda = \lambda_1$ (resp. $\lambda = \lambda_2$). We have

$$f(\lambda) \geq \ell[(\lambda_1, \lambda_2), (f(\lambda_1), h^L(\boldsymbol{\rho}^*_{\lambda_1}, \lambda_2))](\lambda) \tag{42}$$

$$f(\lambda) \geq \ell[(\lambda_1, \lambda_2), (f(\lambda_2), h^L(\boldsymbol{\rho}^*_{\lambda_2}, \lambda_1))](\lambda) \tag{43}$$

Proof. This follows directly from Lemma 3 by using $\boldsymbol{\rho}^*_{\lambda_1}$ (resp. $\boldsymbol{\rho}^*_{\lambda_2}$) and the fact that $h^L(\boldsymbol{\rho}^*_{\lambda_1}, \lambda_1) = f(\lambda_1)$ (resp. $h^L(\boldsymbol{\rho}^*_{\lambda_2}, \lambda_2) = f(\lambda_2)$ by strong duality of linear programming. ◀

Both lines on the right-hand-sides of (42) and (43) provide a lower bound on $f(\lambda)$. Combining the two by taking their maximum provides in general an even tighter bound. This is what we call the bisegment Lagrangian bound.

▷ Theorem 8. Assume that the initial problem (4) has no integer variables, i.e. $n_i = 0$. We have

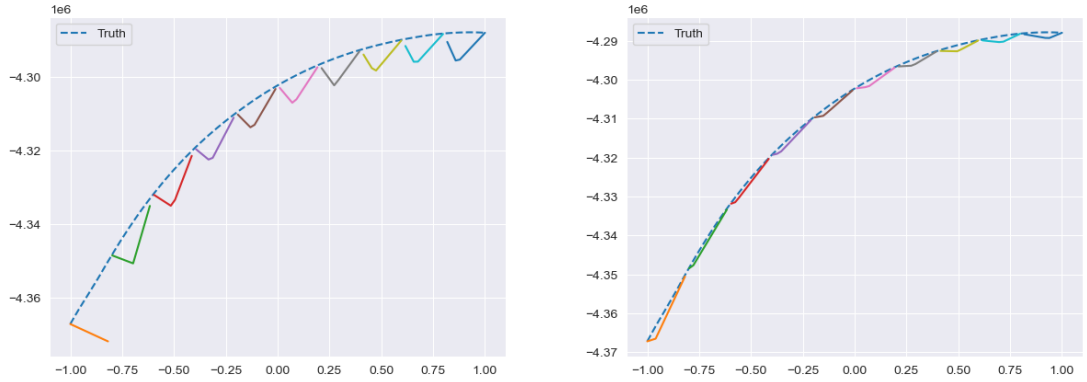
$$f(\lambda) \geq \max\{\ell[(\lambda_1, \lambda_2), (f(\lambda_1), h^L(\boldsymbol{\rho}^*_{\lambda_1}, \lambda_2))](\lambda), \ell[(\lambda_1, \lambda_2), (f(\lambda_2), h^L(\boldsymbol{\rho}^*_{\lambda_2}, \lambda_1))](\lambda)\}.$$

If we do have integer variables, Lemma 3 can also be used, using the optimal dual variables of the linear relaxation, for example. In that case, we have no guarantee of the optimality of the Lagrangian relaxation for one side of the interval though.

The potential weakness of Lagrangian relaxation is that, by relaxing the full set of constraints affected by λ, we may end up with an optimization problem that is too weakly constrained, often ending up with weak relaxations. Observe that a second bounding method can be obtained by combining the technique with Lagrangian relaxation with coefficient-wise relaxation in order to overcome this issue. For any $\boldsymbol{\rho} \leq 0$, we can therefore consider

$$\begin{aligned} \mathcal{P}^{L,\sqsupset}(\boldsymbol{\rho},\lambda) \equiv \underset{x}{\text{minimize}} \quad & \mathbf{c}^t\mathbf{x} - \boldsymbol{\rho}^t((A_2+\lambda D)\mathbf{x} - \mathbf{b_2}) \\ \text{subject to} \quad & A_1\mathbf{x} \leq \mathbf{b_1} \\ & \sum_{j=1}^{n} x_j + \sum_{j=1}^{n} (\lambda d_j)^{\downarrow} x_j \leq \mathbf{b}_2 \\ & \mathbf{x} \in \mathbb{R}^{n_c} \times \mathbb{Z}^{n_i}. \end{aligned} \tag{44}$$

We denote by $h^{L,\sqsupset}(\boldsymbol{\rho},\lambda)$ the optimal value of (44). It is straightforward to see that $h^{L,\sqsupset}(\boldsymbol{\rho},\lambda)$ is a lower bound on $f(\lambda)$ for every λ. All the results previously derived for the Lagrangian relaxation extend to the relaxation enforced by the coefficient-wise relaxation. The two bounds are illustrated in Figure 5.



(a) Lagrangian bisegment **(b)** Lagrangian bisegment coefficient-wise

Figure 5 Illustration of the Lagrangian bounds on the Netlib problem Greenbeb where the uncertainties impact inequalities.

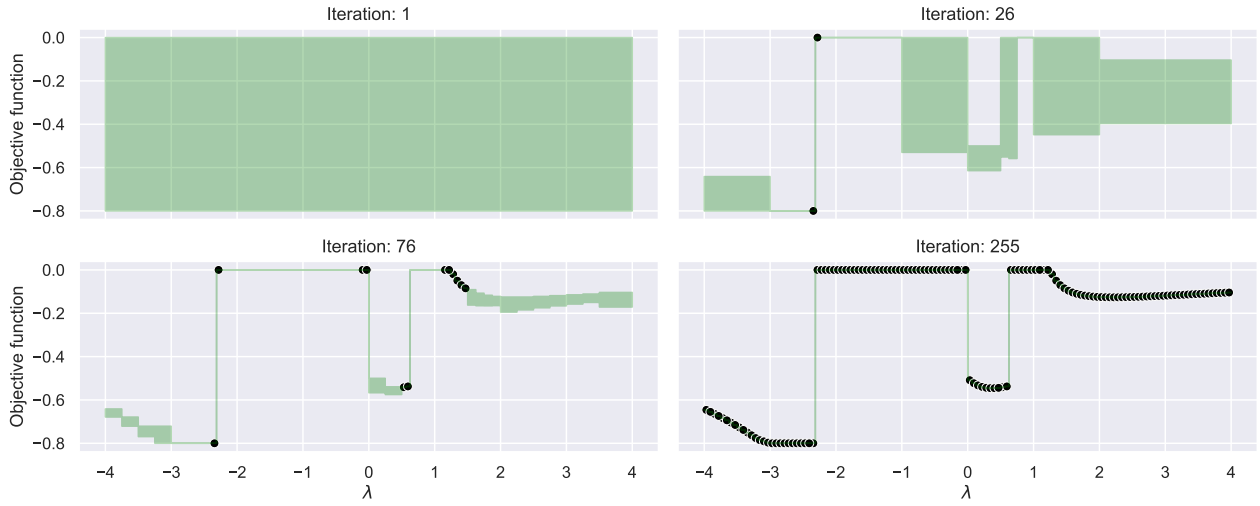


Figure 6 Four steps of the spatial branch and bound algorithm using robust yzflat for both upper and lower bounds on Problem (58) (Annex A.2). At any iteration, the algorithms ensures that $f(\lambda)$ lies in the green zone or is equal to the black dots.

4 Spatial branch and bound

The idea behind the spatial branch and bound algorithm is simple: we have two sets of methods, one that generates an upper bound and one a lower bound. The resulting bounds can be refined by reducing the range

$[\lambda_1, \lambda_2]$, i.e. dividing the range and reapplying the methods to find new upper and lower bounds for this new smaller range.

Algorithm 1 works using this principle. The node strategy chosen maintains a priority queue of ranges to explore, starting with $[\lambda_1, \lambda_2]$. It iteratively selects the range with the largest error (defined by the maximum area between the bounds computed in each range) and refines it, by dividing it into two equal parts and computing the bounds on these two parts, then reinserting them into the priority queue. Degenerate cases exist where the upper/lower bounds cannot be found for a given range (for example when the Lagrangian relaxation is unbounded); in such cases, we write that the lower/upper bounds are respectively $-\infty$ and ∞ and that the distance between the bounds is infinite. This can lead the algorithm to exclusively focus on these ranges. In order to avoid this behaviour, we add a stopping condition on the algorithm: when the range length being refined is smaller than a user-defined ϵ_λ, the algorithm computes the ground truth $f(\lambda)$ at the middle of the range and stops refining this particular range (i.e. it considers that the error in this range is 0). This is an anytime algorithm: at any point in time, the algorithm maintains the sets of bounds that have been computed. Figure 6 shows four different steps of the spatial branch and bound algorithm using robust yzflat for both upper and lower bounds, nominally the 1^{st}, 26^{th}, 76^{th} and last (255^{th}) iteration for Problem (58) (Toy 1 from Annex A.2). The black dots are the true $f(\lambda)$ points computed.

Algorithm 1 Spatial branch and bound algorithm

function IT_ALG($\mathcal{P}$, $\underline{f}$, $\overline{f}$, $[\lambda_1, \lambda_2], \epsilon_\lambda, T$)
Input: $\mathcal{P}$ the problem
$\underline{f} : \mathbb{R}^2 \mapsto (\mathbb{R} \mapsto \mathbb{R} \cup \{-\infty\})$ which provides lower bounds
$\overline{f} : \mathbb{R}^2 \mapsto (\mathbb{R} \mapsto \mathbb{R} \cup \{\infty\})$ which provides upper bounds
$[\lambda_1, \lambda_2]$ the range to consider
ϵ_λ the size of smallest interval to consider before computing a point
T the time limit.
Output: $\mathcal{L}, \mathcal{U}, \mathcal{D}$ the sets of lower bounds, upper bounds and points (resp.) that have been computed.

$\mathcal{T} \leftarrow$ empty max-Priority Queue ▷ Todo-list of ranges
$\mathcal{L} \leftarrow \emptyset$ ▷ Set of computed lower bounds
$\mathcal{U} \leftarrow \emptyset$ ▷ Set of computed upper bounds
$\mathcal{D} \leftarrow \emptyset$ ▷ Set of computed points

$lb_i, ub_i \leftarrow \underline{f}(\lambda_1, \lambda_2), \overline{f}(\lambda_1, \lambda_2)$
Insert $[\lambda_1, \lambda_2]$ with priority $\max_{\lambda \in [\lambda_1, \lambda_2]} ub_i(\lambda) - lb_i(\lambda)$ in $\mathcal{T}$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{lb_i\}$
$\mathcal{U} \leftarrow \mathcal{U} \cup \{ub_i\}$

while $\mathcal{T} \neq \emptyset$ and time limit is not reached **do**
$[\underline{\lambda_i}, \overline{\lambda_i}] \leftarrow$ pop range from $\mathcal{T}$ with highest priority
mid $\leftarrow \frac{\overline{\lambda_i} + \underline{\lambda_i}}{2}$
if $\overline{\lambda_i} - \underline{\lambda_i} > \epsilon_\lambda$ **then**
$lb_1, ub_1 \leftarrow \underline{f}(\underline{\lambda_i}, \text{mid}), \overline{f}(\underline{\lambda_i}, \text{mid})$
$lb_2, ub_2 \leftarrow \underline{f}(\text{mid}, \overline{\lambda_i}), \overline{f}(\text{mid}, \overline{\lambda_i})$
Insert $[\underline{\lambda_i}, \text{mid}]$ with priority $\max_{\lambda \in [\underline{\lambda_i}, \text{mid}]} ub_1(\lambda) - lb_1(\lambda)$ in $\mathcal{T}$
Insert $[\text{mid}, \lambda_2]$ with priority $\max_{\lambda \in [\text{mid}, \lambda_2]} ub_2(\lambda) - lb_2(\lambda)$ in $\mathcal{T}$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{lb_1, lb_2\}$
$\mathcal{U} \leftarrow \mathcal{U} \cup \{ub_1, ub_2\}$
else ▷ stop condition
$\mathcal{D} \leftarrow \mathcal{D} \cup \{(\text{mid}, f_{\mathcal{P}}(\text{mid}))\}$
end if
end while
return $\mathcal{L}, \mathcal{U}, \mathcal{D}$
end function

5 Experiments

In this section, we discuss the different bounding methods proposed and provide several empirical studies related to them. A summary of the ten different bounds method used in the experiments is given in Table 2. The section is divided in three parts:

- We discuss the dataset.
- First, we define our three metrics: the availability, error and timing. We then compare the performance of each bounding method in terms of these metrics.
- We discuss the performance of the proposed spatial branch and bound algorithm for a combination of lower and upper bounds.

Approach	Type	Section	Name	Comment
Robust	Constant	3.1	Robust flat	
Optimization	Linear in λ	3.2	Robust line left	First optimize for λ_1 then for λ_2.
			Robust line right	First optimize for λ_2 then for λ_1.
			Robust yzflat	Add constraint $\mathbf{c^t z} = 0$.
			Robust fixed slope pairwise	$\mathbf{c^t z} = \delta$ where δ is the slope between the optimum for λ_1 and λ_2.
	Envelope	3.3	Robust envelope	
Coefficient-wise	Constant	3.4	Coefficient-wise	Requires $\mathbf{x} \geq 0$.
Lagrangian Relaxations	Piecewise linear	3.5	Lagrangian bisegment	
	Piecewise linear	3.5	Lagrangian bisegment coeff	Add coefficient-wise constraints, requires $\mathbf{x} \geq 0$.

Table 2 Summary of all the bounding methods used in the experiments

5.1 Dataset

To the best of our knowledge, there does not exist a dataset for our category of problems, i.e. where there is a modification on the constraint matrix coefficients, hence, the need to build a dataset. Our proposed dataset is divided into two main categories: LPs and MILPs. The LP instances are further classified into equality and inequality constrained problems, i.e. where the uncertainty matrix D impacts respectively only equality or inequality constraints. The MILP instances include equality and inequality constrained problems from the MIPLIB as well as facility location and unit commitment problems. The dataset is made-of

- The LP dataset which consists of 115 instances derived from 81 original problems from the Netlib library[13]. For each Netlib problem, we randomly select 100 constraints. Within each selected constraint, three coefficients are randomly chosen and copied into the matrix D. Each coefficient in D is then multiplied by a scalar drawn uniformly from $[0.1, 0.9]$, and its sign is flipped with uniform probability. The parameter is defined as $\lambda \in [-1, 1]$. We retain only instances that remain feasible for $\lambda = -1, 0, 1$, excluding degenerate cases where $f(-1) = f(0) = f(1)$. All LP instances are further classified into two categories: problems with only parametric inequality constraints (i.e., $Ax + \lambda Dx \leq b$) and problems with only parametric equality constraints (i.e., $Ax + \lambda Dx = b$).
- The MILP dataset[15] which is composed of 31 instances generated from MIPLIB, 72 facility location problems, and 54 unit commitment problems. The MIPLIB-based instances are generated following the same procedure as for the Netlib problems and are likewise divided into equality and inequality categories. The facility location and unit commitment instances are taken from [6, 16, 9]. For the facility location problems, we assume that locations with a capacity exceeding 100 require a specific *special* machine whose efficiency is uncertain and varies uniformly between 70% and 110% of the nominal production capacity. For the unit commitment problems, uncertainty is introduced in the load-balance constraint by multiplying the electricity production variable by a random scalar drawn uniformly from $[-0.3, 0.3]$.

Because each problem spans a different range of values, we rescale them so that the minimum over the interval $[\lambda_1, \lambda_2]$ is set to 1 and the maximum over the interval $[\lambda_1, \lambda_2]$ to 2, making comparison easier. All problems in the dataset are minimization problems and their size in terms of variables and constraints is shown in Figure 7. The dataset is available on Zenodo [22] and the readers are encouraged to use it and extend the dataset with more problems.

Original dataset	Name	Type	#problems	Comment
Netlib[13]	Netlib ineq	Linear	60	Only inequality constraints in D
	Netlib eq	Linear	55	Only equality constraints in D
MIPLIB[15]	MIPLIB ineq	MILP	19	Only inequality constraints in D
	MIPLIB eq	MILP	12	Only equality constraints in D
Facility location[16]	Facility location	MILP	72	Only inequality constraints in D
Unit commitment[9]	Unit commitment	MILP	54	Only inequality constraints in D

Table 3 Summary of all the datasets

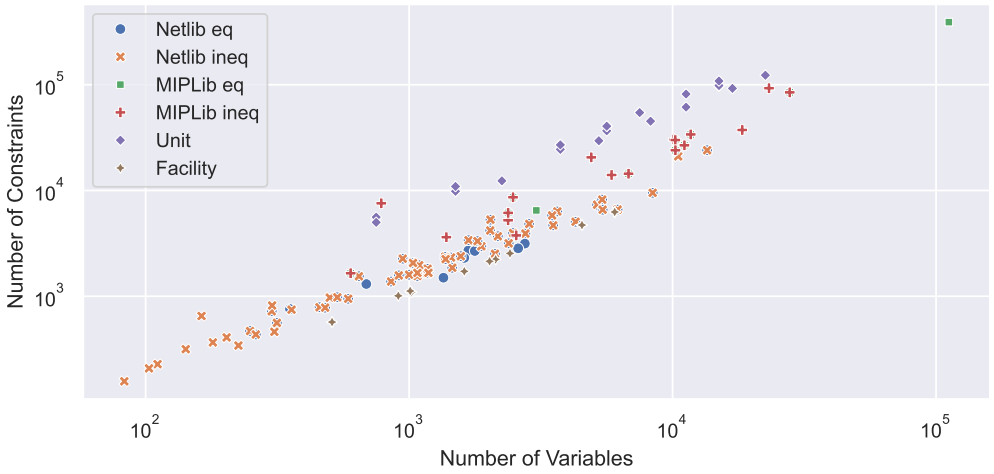


Figure 7 Scatterplot of the number of constraints with respect of the number of variables for all the problems in the dataset by category.

5.2 Bound assessment

In this experiment, we apply each bounding method to each problem presented in Section 5.1. We compare the generated bounds in terms of availability (percentage of points for which a bounding method returns a finite value), error and time taken to compute each bound in order to generate upper and lower bounds. We benchmark the performance of each bound based on these criteria.

- **Availability:** To assess the bounds, we first compute the exact solution $f(\lambda)$ for 100 points uniformly selected in the $[\lambda_1, \lambda_2]$. This set of λ is defined as

$$\mathcal{S} = \{\lambda_1 + \frac{i}{99}(\lambda_2 - \lambda_1) \mid i \in \{0, ..., 99\}\}.$$

To define the availability of a bound, we defined the subset of points for which a bound returns a finite value:

$$\mathcal{A}_b = \{\lambda \in \mathcal{S} \mid \text{bound } b \text{ returns a finite value at } \lambda\} \ . \tag{45}$$

The *availability* is defined as the percentage of points in $\mathcal{S}$ for which the bound is *available*, i.e. returns a finite value:

$$\text{avail}_b = \frac{|\mathcal{A}_b|}{|\mathcal{S}|} \cdot 100 \ . \tag{46}$$

- **Error:** The root mean square error (RMSE) is defined as

$$\text{RMSE}_b = \sqrt{\frac{\sum_{\lambda \in \mathcal{A}_b} (b(\lambda) - f(\lambda))^2}{|\mathcal{A}_b|}} \ . \tag{47}$$

For each problem in the dataset, we compute 100 points in order to have the exact value of $f(\lambda)$ at these given points. We choose to only compute the error for points where the bound exists. If the bound does not exist, the error is considered to be infinite.

- **Timing:** As the problems exhibit a wide variability in computation times, ranging from a few seconds to several minutes, let us consider the median time t_m required to compute $f(\lambda)$ for $\lambda \in \mathcal{S}$ and the time to compute a bound t_b. The reported relative time is given by $t = \frac{t_b}{t_m}$.

In the experiments, we divide the interval $[\lambda_1, \lambda_2]$ on the availability and the RMS error. We split the interval $[\lambda_1, \lambda_2]$ uniformly in N subintervals of same size,

$$[\lambda_1 + \frac{i}{N}(\lambda_2 - \lambda_1), \lambda_1 + \frac{i+1}{N}(\lambda_2 - \lambda_1)] \qquad \forall i \in \{0, ..., N-1\} \ . \tag{48}$$

We illustrate the impact of cutting the interval for $N = 1$, $N = 5$ and $N = 10$ in Figure 8 for the robust envelope bound on one particular problem. We compare the bounding methods with $N = 10$ subintervals in terms of error, timing and availability on the dataset and look for trends in terms of problem and modification type.

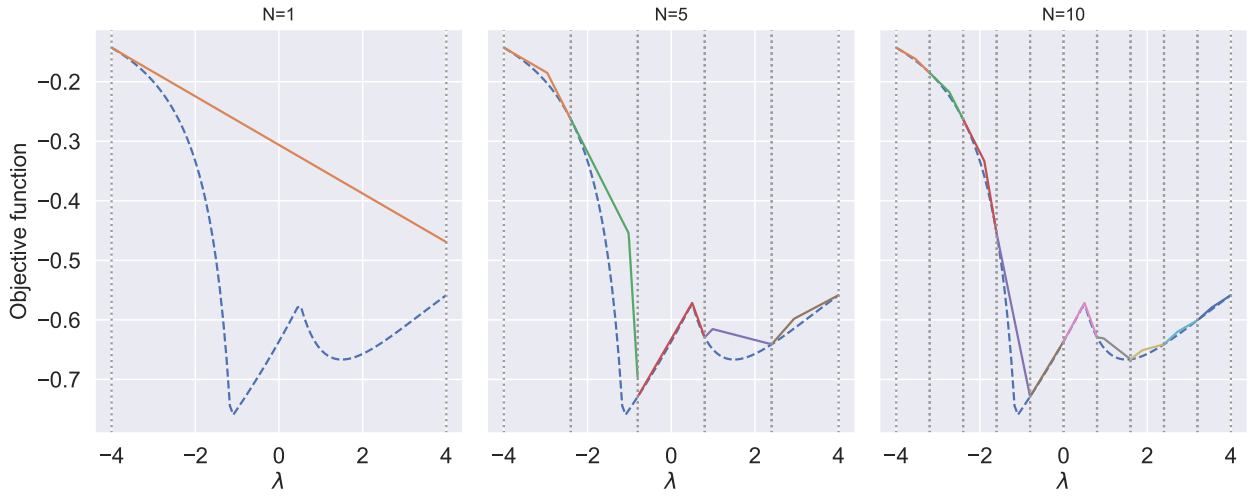


Figure 8 Illustration of robust envelope to provide an upper bound for the problem "Toy 2" given in (59) in Annex A.2.

Bounding methods comparison

The bounding methods are compared in terms of availability, error and timing in Table 4, Table 5 and Table 6 respectively. Let us recall that coefficient-wise methods and Lagrangian bisegment coef need all the variables of the problem to be non-negative.

First, let us compare the methods in terms of **availability**. For the Netlib inequality problems, the robust bounding methods are generally more available when computing upper bounds than lower bounds. In contrast, this trend is reversed for the Netlib equality problems, where lower bounds are more available than upper bounds when using the robust methods. A similar trend can be seen when using Lagrangian methods. The coefficent-wise method is always more available for providing lower bounds than upper bounds on the LP problems. For the MILP problems, in terms of availability, the coefficient-wise method has a higher availability when providing upper bounds than lower bounds for the MILP ineq dataset and a higher availability when providing lower bounds for the MILP eq dataset. For the facility location dataset and unit commitment dataset, the coefficient-wise method has a similar availability when providing upper and lower bounds. The robust flat method is only available for upper bounds and the Lagrangian methods only for lower bounds.

The best bounding methods in terms of availability is the coefficient-wise method for the Netlib ineq lower bound (100% available), Netlib eq lower bound (100% available) and MILP problems. For the Netlib ineq upper

	Netlib				MIPLIB				Facility		Unit	
	ineq		eq		ineq		eq		location		commitment	
	lb	ub	lb	ub	lb	ub	lb	ub	lb	ub	lb	ub
Robust flat	40.0	99.5	70.9	18.2		100		75.0		99.4		100
Robust yzflat	90.2	99.7	95.6	41.8								
Robust line left	75.8	97.3	83.3	34.5								
Robust line right	80.0	97.5	84.2	37.1								
Robust fixed slope pairwise	91.3	100	96.0	41.8								
Robust envelope	90.0	100	98.2	41.8								
Coefficient-wise	100	98.8	100	18.2	78.9	100	100	75.0	100	99.4	100	100
Lagrangian bisegment	37.3	47.7	32.7	9.1	78.9		91.7		99.4		100	
Lagrangian bisegment coef	82.8	89.3	100	9.1	63.2		83.3		99.4		100	

Table 4 Average availability of bounds in percentage, with N=10 bounds computed. Empty values indicate that the bounding method is not usable for this category of problem/bound type.

	Netlib				MIPLIB				Facility		Unit	
	ineq		eq		ineq		eq		location		commitment	
	lb	ub	lb	ub	lb	ub	lb	ub	lb	ub	lb	ub
Robust flat	24.55	0.08	4.49	2.68		0.00		0.00		0.05		0.13
Robust yzflat	0.53	0.06	0.12	0.13								
Robust line left	2.64	0.00	0.17	0.17								
Robust line right	2.61	0.00	0.19	0.14								
Robust fixed slope pairwise	0.50	0.00	0.10	0.06								
Robust envelope	0.38	0.00	0.07	0.04								
Coefficient-wise	0.09	0.09	0.17	2.68	0.39	0.00	0.08	0.00	0.05	0.05	0.48	0.51
Lagrangian bisegment	0.26	0.01	0.12	0.04	2.21		61.87		2.19		0.93	
Lagrangian bisegment coef	0.10	0.00	0.01	0.04	1.69		0.78		1.99		0.93	

Table 5 Median of RMSE computed on a sampling of 100 points, N=10.

	Netlib				MIPLIB				Facility		Unit	
	ineq		eq		ineq		eq		location		commitment	
	lb	ub	lb	ub	lb	ub	lb	ub	lb	ub	lb	ub
Robust flat	2.4	1.3	2.3	1.0		1.1		0.9		1.1		1.3
Robust yzflat	7.5	8.0	8.7	10.4								
Robust line left	20.7	14.9	15.8	20.8								
Robust line right	19.5	14.1	16.7	22.8								
Robust fixed slope pairwise	11.6	9.8	12.7	11.5								
Robust envelope	284.9	113.2	261.9	555.4								
Coefficient-wise	1.1	1.2	1.6	1.2	3.6	2.4	1.2	1.1	1.0	1.1	1.0	0.9
Lagrangian bisegment	4.6	9.2	3.9	8.5	8.3		5.9		1.3		2.0	
Lagrangian bisegment coef	6.0	16.5	7.6	22.7	10.4		7.0		2.1		3.9	

Table 6 Time taken to compute a given bound ($N = 1$). Median on all problems from the category. 1=time to compute a single point of $f(\lambda)$

bound, the most available methods are the robust fixed slope pairwise and robust envelope with an availability of 100%. For the Netlib eq upper bound, the later two methods and the robust yzflat methods have the highest availability of 41.8%. For some datasets in the MILP problems, the robust flat and Lagrangian methods can

match the availability of the coefficient-wise method.

Second, we compare the methods in terms of **error**. For the LP problems, the robust methods and Lagrangian bisegment method have a smaller error when providing upper bounds than when providing lower bounds. On the LP problems, the error is smaller when providing lower bounds than upper bounds for the coefficient-wise and Lagrangian bisegment coef methods. For the MILP problems, the coefficient-wise method has a smaller error on lower bounds than upper bounds. The best bounding methods in terms of error is the coefficient-wise method for the Netlib ineq lb (9%) and MILP problems, exequo with robust flat except for unit commitment ub. For the Netlib ineq ub, Netlib eq datasets, the smallest error is achieved by the Lagrangian bisegment coef method (0% - 1% and 4% respectively), joined by the robust line left/right/fixed sloper pairwise and envelope for Netlib ineq ub dataset. For the MIPLIB ub, the robust flat method has the smallest error.

Third, we compare the methods in terms of **timing**. Over the whole dataset, the coefficient-wise and robust flat bounds are the fastest bound to compute with a median timing equivalent to computing a bit above 1 point to get the bound in the linear dataset and up to 3.6 points in the MILP dataset. The robust methods are slower for computing lower bounds than upper bounds on Netlib ineq but faster on Netlib eq. The Lagrangian methods are faster when computing lower bounds than upper bounds on both Netlib eq and Netlib ineq. For the coefficient-wise bound, the difference in timing between computing upper and lower bounds in the linear dataset is not significant.

5.3 Spatial branch and bound algorithm

For the spatial branch and bound algorithm, we consider the coefficient-wise bounding methods for both upper and lower bounds on the Netlib equality and inequality, MIPLIB inequality, unit commitment, and facility location datasets. We do not consider the MIPLIB equality dataset as there are not enough instances solved in a reasonable time.

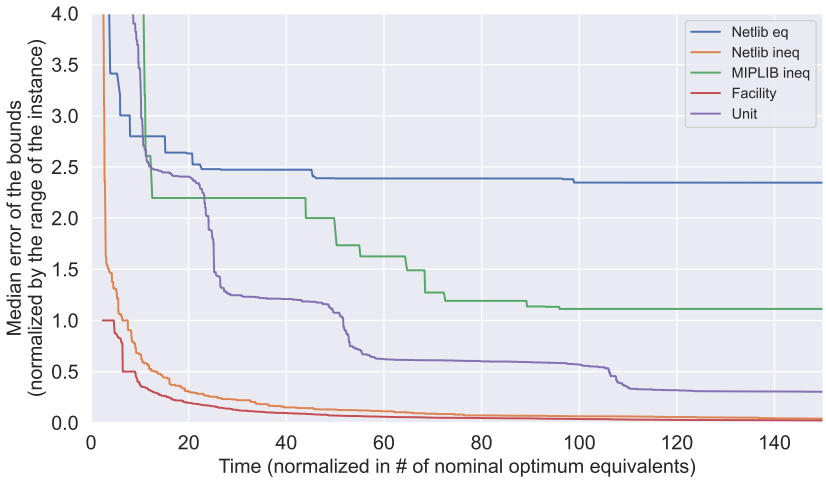


Figure 9 Evolution of the median error (per dataset) with respect to the time (in number of nominal optimum equivalents) for the spatial branch and bound algorithm with coefficient-wise as upper and lower bound.

We show how the error evolves in terms of number of iterations and equivalent time to compute one nominal optimum point. For each problem instance, we compute the error by taking the median difference between the best upper and lower bounds found, over 10 000 sampled points. The overall reported error corresponds to the median of these per-problem errors. To compute an instance-independent time, we compute the average time t_m required to compute $f(\lambda)$ for 100 values of λ and report the timing t as the current timing t_c divided by t_m. The results are shown in Figure 9. We see for all the datasets a steep fall in error with less than 10 equivalent points computed and a stabilization of the error after an equivalent of 120 points computed.

For the Netlib inequality dataset and facility location datasets, we see the steepest decrease in median error. For a time equivalent of 20 points, the branch and bound algorithm reaches an median error smaller than 0.2. For the Netlib equality dataset, we see a significant decrease in median error after 20 iterations, but a stabilization

at an error of 2.5 after 20 iterations. For the MIPLIB inequalities and unit commitment dataset, we see a step by step decrease in error and a stabilization after a time equivalent to computing 110 points.

6 Conclusion

In this paper, we propose a novel approach for assessing the behaviour of the optimal objective function of a problem whose constraint coefficients can linearly vary within a given interval $[\lambda_1, \lambda_2]$. This approach consists of building bounds around the optimal objective function. Building bounds provide strong guarantees on the objective function and helps identify problematic behaviours in the objective that can otherwise be missed.

We present bounding methods based on robust optimization, coefficient-wise reformulation and Lagrangian relaxations. The robust methods are derived in three groups of methods, one that computes constant bounds, the other variable bounds depending on λ and finally, envelope bounds. We benchmark all methods and highlight the efficiency of the bounding approach in terms of precision, availability and timing. For most linear and MILP problems, the bounding methods provide a high precision for little computational time. In particular, the coefficient-wise bounding method often offers an interesting tradeoff between precision and timing. However, we also showed that when the uncertainty term λ impacts equality constraints in Linear Programming, the bounding methods do not perform well leaving room for improvement and further research for that particular dataset. A summary of when to use each bounding method is provided in Figure 10.

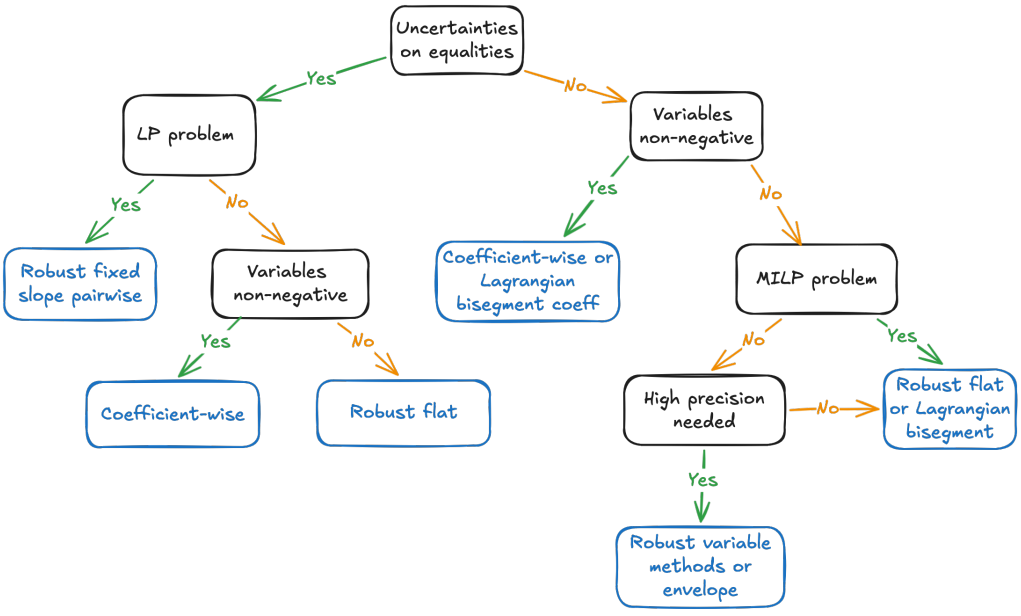


Figure 10 Overview of the decision tree outlining when to use each method

We propose a spatial branch and bound algorithm. It uses a lower and an upper bounding method and iteratively refines the gap between the two by dividing the interval and reapplying the methods. We benchmark using the coefficient-wise method to compute upper and lower bounds and showed that with 70 points equivalent computation we can achieve very small errors.

As future work, we want to further assess our methods on real-life problems. First, we want to study if tuning the methods for particular problems, instead of going with a fully generic approach, can improve the performances of these methods. For instance, we expect "problem-specific" Lagrangian bounds to outperform their generic counterpart presented in this paper. Second, we want to increase the number of instances in our database for the generic study. To achieve that, we plan on integrating and automating the deployment of these bounding methods with other sensitivity analysis approaches in modelling tools such as JuMP[20], Pyomo[5] or The Graph-Based Optimization Modelling Language[21]. From a practitioners point of view, such integration would enable users to perform these analysis in a seamless manner. Third, an aspect that is not exploited in this paper is that, beyond merely having upper and lower bounds on $f(\lambda)$, we often possess a feasible solution to the problem for every $\lambda \in [\lambda_1, \lambda_2]$. Such solutions could serve as effective warm-start points, allowing the problem to be reoptimized efficiently when an exact solution is required.

Disclosure statement

The authors report there are no competing interests to declare.

Funding

The authors gratefully acknowledge the support of the Public Service of Wallonia through the funding of the NKL project in the framework of the Recovery and Resilience Plan (PNRR), initiated and financed by the European Union, and of the SCK-CEN via the SCK-CEN SMR Chair at the University of Liège.

Availability of data and materials

The dataset and codes to run the experiments are available on Zenodo [22]. The code archive contains a Snakemake [24] build file allowing to reproduce all experiments. The raw outputs are also available [23].

References

1 Ilan Adler and Renato D. C. Monteiro. A geometric view of parametric linear programming. *Algorithmica*, 8:161–176, 1992.

2 Kim Allan Andersen, Trine Krogh Boomsma, and Lars Relund Nielsen. Milp sensitivity analysis for the objective function coefficients. *INFORMS Journal on Optimization*, 5(1):92–109, 2023.

3 Arjan B. Berkelaar, Kees Roos, and Tamás Terlaky. *The Optimal Set and Optimal Partition Approach to Linear and Quadratic Programming*, pages 159–202. Springer US, Boston, MA, 1997.

4 Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to linear optimization*. Athena Scientific, 1997.

5 Michael L. Bynum, Gabriel A. Hackebeil, William E. Hart, Carl D. Laird, Bethany L. Nicholson, John D. Siirola, Jean-Paul Watson, and David L. Woodruff. *Pyomo–optimization modeling in python*, volume 67. Springer Science & Business Media, third edition, 2021.

6 CommaLAB. Datasets, 2025. Accessed: 2025-12-22.

7 M. W. Dawande and John N. Hooker. Inference-Based Sensitivity Analysis for Mixed Integer/Linear Programming. 8 2000.

8 Richard Flavell and Gerald R. Salkin. An approach to sensitivity analysis. *Journal of the Operational Research Society*, 26:857–866, 1975.

9 A. Frangioni and C. Gentile. Perspective cuts for a class of convex 0–1 mixed integer programs. *Mathematical Programming*, 106(2):225–236, 2006.

10 Thomas Gal and Josef Nedoma. Multiparametric linear programming. *Management Science*, 7(18):406–422, 1972.

11 Tomas Gal. *Postoptimal Analyses, Parametric Programming, and Related Topics, Degeneracy, Multicriteria Decision Making, Redundancy*. De Gruyter, New York, 1994.

12 Saul I. Gass and Thomas L. Saaty. Parametric objective function (part 2)- generalization. *Journal of the Operations Research Society of America*, 3(4):395–401, 1955.

13 David M. Gay. Electronic mail distribution of linear programming test problems. *Mathematical Programming Society COAL Newsletter*, 13:10–12, 1985.

14 Arthur M. Geoffrion and Ruth Nauss. Parametric and postoptimality analysis in integer linear programming. *Management Science*, 23(5):453–466, 1977.

15 Ambros Gleixner, Gregor Hendel, Gerald Gamrath, Tobias Achterberg, Michael Bastubbe, Timo Berthold, Philipp M. Christophel, Kati Jarck, Thorsten Koch, Jeff Linderoth, Marco Lübbecke, Hans D. Mittelmann, Derya Ozyurt, Ted K. Ralphs, Domenico Salvagnin, and Yuji Shinano. MIPLIB 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library. *Mathematical Programming Computation*, 2021.

16 Kaj Holmberg, Mikael Rönnqvist, and Di Yuan. An exact algorithm for the capacitated facility location problems with single sourcing. *European Journal of Operational Research*, 113(3):544–559, 1999.

17 Benjamin Jansen, Johan J. de Jong, Cees Roos, and Tamas Terlaky. Sensitivity analysis in linear programming: just be careful! *European Journal of Operational Research*, 101(1):15–28, 1997.

18 V. Joseph Bowman Jr. Sensitivity analysis in linear integer programming. *AIIE Transactions*, 4(4):284–289, 1972.

19 Rajab Khalilpour and Iftekar A. Karimin. Parametric optimization with uncertainty on the left hand side of linear programs. *Computers & Chemical Engineering*, 60:31–40, 2014.

20 Miles Lubin and Iain Dunning. Computing in operations research using Julia. *INFORMS Journal on Computing*, 27(2):238–248, 2015.

21 Bardhyl Miftari, Mathias Berger, Guillaume Derval, Quentin Louveaux, and Damien Ernst. Gboml: a structure-exploiting optimization modelling language in python. *Optimization Methods and Software*, 2023.

22 Bardhyl Miftari and Guillaume Derval. MiftariB/lhs-bounding-sensitivity: Paper February version, February 2026.

23 Bardhyl Miftari and Guillaume Derval. Paper: “Sensitivity analysis for linear changes of the constraint matrix of a linear program" output, February 2026.

24 Félix Möder, Kim Philipp Jablonski, Brice Letcher, Michael B. Hall, Christopher H. Tomkins-Tinch, Vanessa Sochat, Jan Forster, Soohyun Lee, Sven O. Twardziok, Alexander Kanitz, Andreas Wilm, Manuel Holtgrewe, Sven Rahmann, Sven Nahnsen, and Johannes Köter. Sustainable data analysis with snakemake. *F1000Research*, 10(33), 2021.

25 Thomas Saaty and Saul Gass. Parametric objective function (part 1). *Journal of the Operations Research Society of America*, 2(3):316–319, 1954.

26 Jack Sherman and Winifred J. Morrison. Adjustment of an inverse matrix corresponding to changes in the elements of a given column or a given row of the original matrix. *The Annals of Mathematical Statistics*, 20:620–624, 1949.

27 Jack Sherman and Winifred J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21(1):124–127, 1950.

28 Max A. Woodbury. *Inverting Modified Matrices*. Memorandum Report / Statistical Research Group, Princeton. Department of Statistics, Princeton University, 1950.

29 Rob A. Zuidwijk. Linear parametric sensitivity analysis of the constraint coefficient matrix in linear programs. *ERIM report series research in management*, ERS-2005-055-LIS, 2005.

A APPENDIX

A.1 Lemma quadratic function

This Lemma is used in Section 3.2.

▷ Lemma 4. Given a quadratic function $q(x) = ax^2 + bx + c$. The maximum of $q(x)$ over $x_1 \leq x \leq x_2$ is upper bounded by

$$\max \begin{cases} ax_1^2 + bx_1 + c \\ ax_2^2 + bx_2 + c \\ ax_1x_2 + b\frac{x_1+x_2}{2} + c \end{cases} \tag{49}$$

Proof. We analyse separately the concave and the convex case:

- If the function $q(x)$ is convex, that is if $a \geq 0$, then the maximum of the function in the polyhedron $[x_1, x_2]$. The equation (4) is known to be attained at one of its vertices, either x_1 or x_2.
- For the concave case, let us consider the two tangents at x_1 and x_2:

$$t_{x_1}(x) = ax_1^2 + bx_1 + c + (x - x_1)(2ax_1 + b) = c + 2ax_1x - ax_1^2 + bx \tag{50}$$
$$t_{x_2}(x) = ax_2^2 + bx_2 + c + (x - x_2)(2ax_2 + b) = c + 2ax_2x - ax_2^2 + bx. \tag{51}$$

If $q(x)$ is concave ($a \leq 0$) then any tangent is an upper bound for the function for any x. Taking two tangent at different points and taking the minimum of the two functions is also an upper bound. That is, for any $x_1 < x_2$:

$$\min(t_{x_1}(x), t_{x_2}(x)) \geq q(x) \quad \forall x. \tag{52}$$

$\min(t_{x_1}(x), t_{x_2}(x))$ forms a piecewise linear function. The tangents t_{x_1} and t_{x_2} intersect at the middle point of $[x_1, x_2]$:

$$t_{x_1}(x) = t_{x_2}(x) \tag{53}$$
$$2ax_1x - ax_1^2 = 2ax_2x - ax_2^2 \tag{54}$$
$$2x_1x - x_1^2 = 2x_2x - x_2^2 \tag{55}$$
$$x = \frac{x_2^2 - x_1^2}{2(x_2 - x_1)} = \frac{x_1 + x_2}{2} \tag{56}$$

with the value $t_1(\frac{x_1+x_2}{2}) = ax_1x_2 + b\frac{x_1+x_2}{2} + c$.
Now, the maximum of the function $\min(t_{x_1}(x), t_{x_2}(x))$ over $[x_1, x_2]$ is achieved either on $t_{x_1}(x_1)$ or $t_{x_2}(x_2)$ or in the middle of the range, at $t_1(\frac{x_1+x_2}{2})$:

$$\max_{x \in [x_1, x_2]} \min(t_{x_1}(x), t_{x_2}(x)) = \max \begin{cases} ax_1^2 + bx_1 + c \\ ax_2^2 + bx_2 + c \\ ax_1x_2 + b\frac{x_1+x_2}{2} + c \end{cases} \tag{57}$$

◀

A.2 Dataset

In this Appendix, we provide a full overview the illustrative problems.

A.2.1 Illustrative problems

This category contains three small two-variable problems that are used for the sake of illustration rather than being of particular interest.

1. **Toy 1:**

$$\begin{aligned}
\mathcal{P}_{\text{toy }1}(\lambda) \equiv \min \quad & \begin{pmatrix} -1 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \\
\text{s.t} \quad & \begin{pmatrix} -2 & -1 \\ 1 & 2 \\ -1 & 1 \\ 1 & -1 \\ 1 & -3 \\ 2 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \leq \begin{pmatrix} 3 \\ 0 \\ 2 \\ 1 \\ 1 \\ 3 \end{pmatrix} \\
& \begin{pmatrix} 3 & -1 \\ -3 & 1 \\ -2 & 1 \\ 2 & -1 \\ -1 & -2 \\ -1 & -3 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \lambda \begin{pmatrix} 3 & 3 \\ 3 & -2 \\ 2 & 3 \\ 2 & -1 \\ -2 & 1 \\ 0 & -2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \\ 0 \\ 3 \\ 3 \\ 1 \end{pmatrix} \\
& \forall \lambda \in [-4, 4]
\end{aligned} \tag{58}$$

2. **Toy 2:**

$$\begin{aligned}
\mathcal{P}_{\text{toy }2}(\lambda) \equiv \min \quad & \begin{pmatrix} 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \\
\text{s.t} \quad & \begin{pmatrix} -2 & 0 \\ 2 & 2 \\ -1 & 0 \\ -1 & -1 \\ 0 & 1 \\ -3 & -2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \leq \begin{pmatrix} 1 \\ 2 \\ 2 \\ 3 \\ 3 \\ 1 \end{pmatrix} \\
& \begin{pmatrix} 2 & 3 \\ 0 & 1 \\ 1 & -3 \\ 0 & -3 \\ 3 & 0 \\ -2 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \lambda \begin{pmatrix} 3 & 3 \\ 0 & 2 \\ -2 & -1 \\ 2 & 1 \\ 2 & 0 \\ 2 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \leq \begin{pmatrix} 3 \\ 1 \\ 2 \\ 3 \\ 3 \\ 1 \end{pmatrix} \\
& \forall \lambda \in [-4, 4]
\end{aligned} \tag{59}$$

3. **Toy 3:** This problem is the one written in (5). We rewrite it here for the sake of completeness.

$$\begin{aligned}
\mathcal{P}_{\text{toy }3}(\lambda) \equiv \min \quad & \begin{pmatrix} 2 & -2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \\
\text{s.t} \quad & \begin{pmatrix} -2 & 2 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \leq \begin{pmatrix} 4 \\ 1 \end{pmatrix} \\
& \begin{pmatrix} 2 & 1 \\ -2 & -3 \\ 2 & 2 \\ -1 & -4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \lambda \begin{pmatrix} -1 & -4 \\ 0 & 4 \\ -4 & -3 \\ 2 & 4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \leq \begin{pmatrix} 4 \\ 2 \\ 0 \\ 2 \end{pmatrix} \\
& \forall \lambda \in [-10, 9]
\end{aligned}$$

4. **Toy 4:** This problem is the one written in (12). We rewrite it for the sake of completeness.

$$\begin{aligned}
\mathcal{P}_{\text{toy }4}(\lambda) \equiv \min_{x,y} \quad & \begin{pmatrix} -2 & -2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \\
\text{s.t} \quad & \begin{pmatrix} 3 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \leq \begin{pmatrix} 3 \end{pmatrix} \\
& \begin{pmatrix} -5 & -2 \\ 1 & 4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \lambda \begin{pmatrix} -3 & -2 \\ -3 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \leq \begin{pmatrix} 0 \\ -3 \end{pmatrix} \\
& \forall \lambda \in [-2, 2]
\end{aligned} \tag{60}$$