

IMPROVED PENALTY ALGORITHM FOR MIXED INTEGER PDE CONSTRAINED OPTIMIZATION PROBLEMS

DOMINIK GARMATTER*, MARGHERITA PORCELLI†, FRANCESCO RINALDI‡, AND
MARTIN STOLL*

Abstract. Optimal control problems including partial differential equation (PDE) as well as integer constraints merge the combinatorial difficulties of integer programming and the challenges related to large-scale systems resulting from discretized PDEs. So far, the branch-and-bound framework has been the most common solution strategy for such problems. In order to provide an alternative solution approach, especially in a large-scale context, this article investigates penalization techniques. Taking inspiration from a well-known family of existing exact penalty algorithms, a novel *improved penalty algorithm* is derived, whose key ingredients are a basin hopping strategy and an interior point method, both of which are specialized for the problem class. A thorough numerical investigation is carried out for a standard stationary test problem. Extensions to a convection-diffusion as well as a nonlinear test problem finally demonstrate the versatility of the approach.

Key words. mixed integer optimization, optimal control, PDE-constrained optimization, exact penalty methods, interior point methods

AMS subject classifications. 65K05, 90C06, 90C11, 93C20, 90C51

1. Introduction. Optimal control problems that are governed by a partial differential equation (PDE) as well as integer constraints on the control and possible additional constraints are commonly referred to as mixed integer PDE-constrained optimization (MIPDECO) problems. They pose several challenges as they combine two fields that have been surprisingly distinct from each other in the past: integer programming and PDEs. While integer optimization problems have an inherent combinatorial complexity that has to be dealt with, PDE-constrained optimization problems have to deal with possibly large-scale linear systems resulting from the discretization of the PDE, see, e.g., [1].

Albeit these challenges, MIPDECO problems are gaining an increased attention as they naturally arise in many real world applications such as gas networks [2], [3], the placement of tidal and wind turbines [4]–[6] or power networks [7]. From the theoretical point of view, there have been recent advances in the field including a Sum-up-Rounding strategy [8], [9], a derivative-free approach [10], and new sophisticated rounding techniques [11].

A classical solution approach for a MIPDECO problem is to *first-discretize-then-optimize* where the PDE and the control are discretized such that the continuous MIPDECO problem is then approximated by a finite-dimensional (and possibly large-scale) mixed-integer nonlinear programming problem (MINLP). Standard techniques, see, e.g., [12] for an excellent overview, such as branch-and-bound can then be used to solve the MINLP. Unfortunately, depending on the size of the finite dimensional approximation, these techniques may struggle. On the one hand, the discretization of the control might (especially for problems with a time-dependent control) result in a large amount of integer variables and thus an immense combinatorial complexity of the MINLP. On the other hand, the discretization of the PDE results in large-scale

*Department of Mathematics, Chemnitz University of Technology, Germany (dominik.garmatter@math.tu-chemnitz.de, martin.stoll@math.tu-chemnitz.de)

†Department of Mathematics, University of Bologna, Italy (margherita.porcelli@unibo.it)

‡Department of Mathematics "Tullio Levi-Civita", University of Padova, Italy (rinaldi@math.unipd.it)

linear systems occurring whenever an NLP-relaxation of the MINLP has to be solved.

The contribution of this article to the field is to provide an alternative approach for MIPDECO problems via an equivalent penalty formulation of the original problem. While penalty reformulations have been studied in the context of integer programming, see, e.g., [13]–[17], and penalty approaches have been developed, see, e.g., [18]–[20], there have been (to the knowledge of the authors) no contributions that explicitly deal with MIPDECO problems.

The general idea of penalty reformulations is to relax the integer constraints of the problem and add a suitable penalty term to the objective function, thus penalizing controls that violate the previously present integer constraints. A naive solution strategy could then be to iteratively solve the resulting penalty formulation while increasing the amount of penalization in each iteration until one ends up with an integer solution. The upside of such penalization strategies is that the combinatorial complexity of the integer constraints is eliminated from the problem formulation and the penalty term then ensures that the resulting solution satisfies the integer constraints. The downside is that penalty terms are usually concave such that one has to deal with non-convex NLPs with a possibly exponential amount of local minimizers.

To still provide qualitative solutions in this context, the main contribution of this article is the development of a novel algorithm that is closely related to a family of existing exact penalty (EXP) algorithms, which have been analyzed both in the context of general constrained optimization [21], [22] and in the context of integer optimization [19]. Roughly speaking, a general EXP algorithmic framework, which is an iterative procedure, provides an automatic tool for when to increase penalization and when to aim for a better minimizer via a suitable global solver for the penalized subproblems. One can then show convergence towards a global minimizer of the original problem, see, e.g., [19, Corollary 1] for the analysis of the integer case.

A practical implementation of an EXP algorithm is carried out in this paper. Although the algorithm is developed taking into account a model problem, it will become clear that it can handle quite general MIPDECO problems. The idea of the resulting improved penalty algorithm (IPA) is to combine the EXP framework with a suitably developed search approach, closely connected to basin hopping or iterated local search methods, see, e.g., [23], [24]. The search combines a local optimization algorithm with a perturbation strategy (both tailored to the specific application) in order to find either the global or a good local minimum of the penalty reformulation.

Our suitably developed local optimization solver is an interior point method that exploits the structure of the penalty formulation related to a MIPDECO problem in the following ways:

- it explicitly handles the non-convexity introduced by the penalty term;
- it uses a specific preconditioner to efficiently handle the linear algebra.

Via this approach, large-scale problems can be handled and the IPA is numerically compared, both for a standard test problem and a convection-diffusion problem, to a traditional penalty method as well as a branch-and-bound routine from CPLEX [25].

The remainder of this work is organized as follows: the model problem is presented and discretized in Section 2. Section 3 reviews the EXP algorithm, extends its convergence theory to the class of MIPDECO problems considered, and then develops the novel improved penalty algorithm. Section 4 gathers implementation details of the IPA, such as the interior point method, and briefly collects the remaining algorithms for the numerical comparison that is carried out in Section 5. Finally, conclusions are drawn in Section 6 including an outlook on MIPDECO problems with a nonlinear PDE constraint.

2. Problem formulation. We begin with the description of the optimal control model problem in function spaces. Following the first-discretize-then-optimize approach, we then present the discretized model problem as well as its continuous relaxation. Finally, we review existing solution techniques and make some remarks on the model problem.

2.1. Continuous optimal control problem. We begin with the description of the PDE in order to formulate the optimal control problem. Consider a bounded domain $\Omega \subset \mathbb{R}^2$ with Lipschitz boundary, source functions $\phi_1, \dots, \phi_l \in L^2(\Omega)$ and based on these the PDE: for a given control vector $u = (u_1, \dots, u_l)^\top \in \mathbb{R}^l$ find the state $y \in H_0^1(\Omega)$ solving

$$(2.1) \quad -\Delta y(x) = \sum_{i=1}^l u_i \phi_i(x), \quad x \in \Omega,$$

where the PDE is to be understood in the weak sense. Existence and uniqueness of a solution $y \in H_0^1(\Omega)$ of (2.1) follow from the Lax-Milgram theorem. For now, we choose to model the sources ϕ_1, \dots, ϕ_l as Gaussian functions with centers $\tilde{x}_1, \dots, \tilde{x}_l \in \tilde{\Omega}$, and $\tilde{\Omega} \subset \Omega$. Thus, for $x \in \mathbb{R}^2$,

$$(2.2) \quad \phi_i(x) := \kappa e^{-\frac{\|x - \tilde{x}_i\|_2^2}{\omega}}, \quad i = 1, \dots, l,$$

with height $\kappa > 0$ and width $\omega > 0$. The optimal control problem in function spaces then reads: given a desired state $y_d \in L^2(\Omega)$, find a solution pair $(y, u) \in H_0^1(\Omega) \times \{0, 1\}^l$ of

$$(2.3) \quad \begin{aligned} \min_{y \in H_0^1(\Omega), u \in \{0, 1\}^l} & \quad \frac{1}{2} \|y - y_d\|_{L^2(\Omega)}^2, \\ \text{s.t.} & \quad (y, u) \text{ fulfill (2.1), and } \sum_{i=1}^l u_i \leq S \in \mathbb{N}, \end{aligned}$$

where the inequality constraint in (2.3) is commonly referred to as a *knapsack constraint*. This problem can be interpreted as fitting a desired heating pattern y_d by activating up to S many sources that are distributed around the domain Ω . Since the amount of controls $u \in \{0, 1\}^l$ is finite and for each control there is a uniquely determined state y , problem (2.3) is in its essence a combinatorial problem so that existence of at least one global minimizer is guaranteed. We close this section with some remarks on the presented model problem.

- Remark 2.1.* (a) The Gaussian source functions are motivated by porous-media flow applications to determine the number of boreholes, see, e.g., [26], [27], and problem (2.3) with this choice is furthermore a model problem mentioned in [28, Section 19.3]. We will see throughout the development of our algorithm that it does not rely on this particular modelling of the control. Exemplarily, Section 5.2 will deal with a convection-diffusion equation with piece-wise constant sources (and we mention that piece-wise constant sources were also used in [29]), but one could also use a general distributed control as proposed in [11].
- (b) It is well-known that problems with general integer constraints can be reduced to problems with binary constraints, see, e.g., [30]. Furthermore, [15, Section 4] provides an alternative in the context of penalty approaches by directly penalizing general integer constraints. Extending the presented model problem from binary to general integer constraints and developing strategies to efficiently deal with these is an interesting aspect for future research.

2.2. Discretized model problem and continuous relaxation. Introducing a conforming mesh over Ω using N vertices, let $M \in \mathbb{R}^{N \times N}$ and $K \in \mathbb{R}^{N \times N}$ be the mass and stiffness matrices of a suitable finite element discretization of (2.1). Furthermore, let the matrix $\Phi \in \mathbb{R}^{N \times l}$ contain the finite element coefficients of the source functions in its columns, i.e., each column contains the evaluation of the respective source function at the N vertices of the grid. With these matrices at hand, we formulate the *discretized optimal control problem*

$$(2.4) \quad \begin{aligned} \min_{y \in \mathbb{R}^N, u \in \{0,1\}^l} \quad & \frac{1}{2}(y - y_d)^\top M(y - y_d), \\ \text{s.t.} \quad & Ky = M\Phi u, \quad \text{and} \quad \sum_{i=1}^l u_i \leq S \in \mathbb{N}. \end{aligned}$$

In (2.4) and for the remainder of this article, y denotes the vector of the finite element coefficients of the corresponding finite element approximation of (2.1) rather than the actual PDE-solution. The same holds true for the desired state y_d which from now on represents a finite element coefficient vector instead of an actual $L^2(\Omega)$ -function. Relaxing the integer constraints in (2.4) yields the *continuous relaxation*

$$(2.5) \quad \begin{aligned} \min_{y \in \mathbb{R}^N, u \in \mathbb{R}^l} \quad & \frac{1}{2}(y - y_d)^\top M(y - y_d) \\ \text{s.t.} \quad & Ky = M\Phi u, \quad 0 \leq u \leq 1, \quad \text{and} \quad \sum_{i=1}^l u_i \leq S \in \mathbb{N}. \end{aligned}$$

We reformulate both problems (2.4) and (2.5) in a more compact way.

LEMMA 2.2. *Introducing for $x \in \mathbb{R}^{N+l}$*

$$\tilde{J}(x) := \frac{1}{2}x^\top \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix} x - x^\top \begin{bmatrix} My_d \\ 0 \end{bmatrix} + \frac{1}{2}y_d^\top My_d$$

and $f : \mathbb{R}^l \rightarrow \mathbb{R}^N : u \mapsto K^{-1}M\Phi u$, problems (2.4) and (2.5) are equivalent to

$$(P) \quad \min_{x \in W} \tilde{J}(x) \quad W := \left\{ x = (y, u)^\top \in \mathbb{R}^{N+l} \mid u \in \{0, 1\}^l, \sum_{i=1}^l u_i \leq S, y = f(u) \right\}$$

and

$$(P\text{cont}) \quad \min_{x \in X} \tilde{J}(x) \quad X := \left\{ x = (y, u)^\top \in \mathbb{R}^{N+l} \mid u \in [0, 1]^l, \sum_{i=1}^l u_i \leq S, y = f(u) \right\},$$

respectively. $W \subset \mathbb{R}^{N+l}$ is a compact set and $X \subset \mathbb{R}^{N+l}$ is compact and convex such that (Pcont) is a convex problem.

Proof. The equivalence of the problems in question follows from the definition of the sets W and X and the map f . Furthermore, W is obviously compact and X as the image of a compact convex set under a linear map is compact and convex. Thus, the convexity of (Pcont) follows from the convexity of X and the convexity of \tilde{J} where the matrix

$$\begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix},$$

with M being positive definite, is positive semidefinite. \square

The authors acknowledge that (P) might be tackled by existing methods, see, e.g., [18], [20], [29], and thus want to comment on the limitations of these approaches in a large-scale context.

1. In [29], a branch-and-cut algorithm is presented, where the computation of a cutting plane requires one linear PDE solution per dimension of the control space. Therefore, this approach can become excessively time-consuming for large l .
2. In [18], an EXP framework that embeds an iterative genetic algorithm is presented, where the amount of objective function evaluations per iteration usually scales quadratically with the problem dimension l . But in the PDE-constrained optimization context of (P) an evaluation of the objective function requires a PDE solution, such that the approach can become costly for large l and/or N .
3. In [20], a penalty-based approach combined with a smoothing method is considered to solve nonlinear and possibly non-convex optimization problems with binary variables. The main drawback in this case is: there is no theoretical guarantee that one converges towards the global minimum. Hence, the smoothing and penalty parameters need to be carefully initialized and handled during the optimization process in order to avoid getting stuck in bad local minima.
4. Finally, a comparison of our method towards freshly discovered solutions strategies such as a Sum-Up Rounding method for PDEs [9] and a sophisticated rounding technique [11] are of interest for future work.

Finally, we want to comment on a time-dependent extension of the model problem as well as what is often referred to as the *reduced formulation*.

Remark 2.3. (a) Introducing the time interval $[0, T]$ with final time $T > 0$, the model problem (2.3) can be expanded to a time-dependent problem by introducing the parabolic PDE: for $\tilde{u} := (\tilde{u}_1(t), \dots, \tilde{u}_l(t))^T \in \mathbb{R}^l \times (0, T)$, find $\tilde{y} \in L^2(0, T, H_0^1(\Omega))$ solving

$$(2.6) \quad \frac{\partial}{\partial t} \tilde{y}(t, x) - \Delta \tilde{y}(t, x) = \sum_{i=1}^l \tilde{u}_i(t) \phi_i(x), \quad (t, x) \in (0, T) \times \Omega,$$

$$\tilde{y}(0, x) = 0, \quad x \in \bar{\Omega}.$$

A possible corresponding optimal control problem could then be: given a desired state $\tilde{y}_d \in L^2((0, T) \times \Omega)$, solve

$$(2.7) \quad \min_{\substack{\tilde{y} \in L^2(0, T, H_0^1(\Omega)) \\ \tilde{u} \in \{0, 1\}^l \times (0, T)}} \frac{1}{2} \|\tilde{y} - \tilde{y}_d\|_{L^2((0, T) \times \Omega)}^2,$$

$$\text{s.t.} \quad (\tilde{y}, \tilde{u}) \text{ fulfill (2.6), and } \sum_{i=1}^l \tilde{u}_i(t) \leq S \in \mathbb{N}, \forall t \in (0, T).$$

It is easy to see that after discretizing (2.7) in space and time, the dimensions of both the PDE constraint and the control will scale with the number of time steps, so that the resulting discretized optimal control problem is of large scale both in l and N . It will become obvious in the upcoming sections that our approach aims at such large-scale problems and we plan to tackle time-dependent problems in future work.

- (b) We acknowledge that (2.4), from a theoretical point of view, can be recast in *reduced form* via the substitution $y = K^{-1}M\Phi u$ so that the resulting *reduced optimal control problem* only depends on the control u . Due to the large scale context present in this work and in possible extensions, we consider the frequent

inversion of the stiffness matrix that is necessary in order to obtain this reduced form to be prohibitive and impractical for numerical investigations.

3. Improved penalty algorithm (IPA). This section contains the main contribution of this article, the development of our novel improved penalty algorithm (IPA). We will first introduce a well-known equivalent penalty reformulation of (P), followed by an exact penalty algorithm from [19]. Afterwards we will develop the IPA, where the idea is to combine the EXP framework with a local search strategy such that the resulting algorithm only relies on a local solver.

3.1. Penalty formulation and exact penalty (EXP) algorithm. Starting from the continuous relaxation (2.5), we add the well-known penalty term

$$(3.1) \quad \frac{1}{\varepsilon} \sum_{i=1}^l u_i(1 - u_i)$$

to the objective function. Obviously, this concave penalty term penalizes a non-binary control, where $\varepsilon > 0$ controls the amount of penalization. This yields the following *penalty formulation*

$$(3.2) \quad \begin{aligned} \min_{y \in \mathbb{R}^N, u \in \mathbb{R}^l} \quad & \frac{1}{2}(y - y_d)^\top M(y - y_d) + \frac{1}{\varepsilon} \sum_{i=1}^l u_i(1 - u_i) \\ \text{s.t.} \quad & Ky = M\Phi u, \quad 0 \leq u \leq 1 \quad \text{and} \quad \sum_{i=1}^l u_i \leq S \in \mathbb{N}. \end{aligned}$$

Following Lemma 2.2, (3.2) can be rewritten as

$$(Ppen) \quad \begin{aligned} \min_{x \in X} J(x; \varepsilon), \quad & \text{with} \\ J(x; \varepsilon) := \quad & \frac{1}{2}x^\top \begin{bmatrix} M & 0 \\ 0 & -\frac{2}{\varepsilon}I_l \end{bmatrix} x - x^\top \begin{bmatrix} My_d \\ -\frac{1}{\varepsilon}\mathbf{1} \end{bmatrix} + \frac{1}{2}y_d^\top My_d, \end{aligned}$$

where $I_l \in \mathbb{R}^{l \times l}$ is the identity-matrix and $\mathbf{1} := (1, \dots, 1)^\top \in \mathbb{R}^l$.

PROPOSITION 3.1. *There exists an $\tilde{\varepsilon} > 0$ such that for all $\varepsilon \in (0, \tilde{\varepsilon}]$ problems (P) and (Ppen) have the same minimum points. Having the same minimum points here means that both problems (P) and (Ppen) have the same global minima (if there exist multiple). In this sense both problems (P) and (Ppen) are equivalent.*

Proof. From Lemma 2.2 it is clear that $J \in C^1(\mathbb{R}^{N+l})$ and that W and X are compact. Together with the results derived in [15, Section 3] all assumptions of [15, Theorem 2.1] are fulfilled such that the desired statement follows. \square

We mention that the equivalence result from Proposition 3.1 also holds for a variety of concave penalty terms, see, e.g., [15, (19)-(23)] or [16, (21)]. We chose the penalty term (3.1) in this article since it is quadratic and thus the combined objective function J remains quadratic.

Before we formulate the exact penalty algorithm, we introduce a rounding strategy that suitably handles the knapsack constraint in X and W and prove that it is the correct tool required for the algorithm design.

DEFINITION 3.2. *Let for $x = [y, u]^\top \in X$ and $S \in \mathbb{N}$, with $S \leq l$, denote $u_S \in \mathbb{R}^S$ the S largest components of u . We then define the smart rounding $[x]_{SR} \in W$ of x as follows:*

- define $[u]_{SR}$ by rounding u_S component-wise to the closest integer and set the remaining components to 0,

- define $[x]_{SR} := (f([u]_{SR}), [u]_{SR})^\top \in W$.

We illustrate the smart rounding by considering a simple example.

EXAMPLE 3.3. Let $S = 2$ and $l = 3$ and let $[\cdot]$ denote the usual rounding to the closest integer. Then, for

$$u_1 = (0.8, 0.7, 0.1)^\top \quad \text{and} \quad u_2 = (0.63, 0.62, 0.61)^\top$$

it is $[u_1]_{SR} = (1, 1, 0)^\top \equiv [u_1]$, but $[u_2]_{SR} = (1, 1, 0)^\top \neq [u_2] = (1, 1, 1)^\top$. Thus, for u_2 , the smart rounding does satisfy the knapsack constraint while the usual rounding does not.

DEFINITION 3.4. The Chebyshev distance between a point $x \in \mathbb{R}^{N+l}$ and a set $C \subset \mathbb{R}^{N+l}$ is defined as

$$\text{dist}_\infty(x, C) = \min_{y \in C} \|x - y\|_\infty.$$

PROPOSITION 3.5. Let f , W and X be the linear map and the sets defined in Lemma 2.2. For $z = (f(z_u), z_u)^\top \in W$, let $B(z)$ be the set

$$(3.3) \quad B(z) := \{x = (y, u)^\top \in \mathbb{R}^{N+l} \mid \|y\|_\infty \leq \beta, \|u - z_u\|_\infty \leq \rho\},$$

where $\rho, \beta > 0$ are chosen such that $z \in B(z)$ and

$$B(z_a) \cap B(z_b) = \emptyset, \quad \text{for all } z_a, z_b \in W \text{ with } z_a \neq z_b.$$

Given a point $\bar{x} = (f(\bar{u}), \bar{u})^\top \in X$, then the point $\bar{z} := [\bar{x}]_{SR} \in W$ minimizes the Chebyshev distance between \bar{x} and the sets $B(z)$ with $z \in W$, that is

$$\bar{z} \in \arg \min_{z \in W} \text{dist}_\infty(\bar{x}, B(z)).$$

Proof. We first note that for example $\rho = 0.4$ and $\beta \geq \max_{z \in W} \|f(z_u)\|_\infty$ are sufficient choices such that the requirements for the sets $B(z)$, $z \in W$ are met.

Now, if there exists a $z \in W$ such that $\bar{x} \in B(z)$, it has to be $z \equiv \bar{z} = [\bar{x}]_{SR}$. In this trivial case, we have $\text{dist}_\infty(\bar{x}, B(\bar{z})) = 0$ and the result follows.

Therefore, we assume in the following that $\bar{x} \notin B(z)$ for all $z \in W$. By contradiction, there then exists a point $\hat{z} = (f(\hat{z}_u), \hat{z}_u)^\top \in W$ satisfying

$$(3.4) \quad \text{dist}_\infty(\bar{x}, B(\hat{z})) < \text{dist}_\infty(\bar{x}, B(\bar{z})).$$

We can hence find two points $\hat{p} = (\hat{p}_y, \hat{p}_u)^\top \in B(\hat{z})$ and $\bar{p} = (\bar{p}_y, \bar{p}_u)^\top \in B(\bar{z})$ satisfying

$$(3.5) \quad \|\hat{p} - \bar{x}\|_\infty = \text{dist}_\infty(\bar{x}, B(\hat{z})) \quad \text{and} \quad \|\bar{p} - \bar{x}\|_\infty = \text{dist}_\infty(\bar{x}, B(\bar{z})).$$

From the definition (3.3) and $\bar{x} = (f(\bar{u}), \bar{u})^\top$, it is easy to see that

$$(3.6) \quad \hat{p}_y = \bar{p}_y = f(\bar{u}) \quad \text{and} \quad \|\hat{p}_u - \hat{z}_u\|_\infty = \|\bar{p}_u - \bar{z}_u\|_\infty = \rho.$$

From (3.6), we obtain

$$(3.7) \quad \|\hat{p} - \bar{x}\|_\infty = \max\{\|\hat{p}_u - \bar{u}\|_\infty, \underbrace{\|\hat{p}_y - f(\bar{u})\|_\infty}_{=0}\} = \|\hat{p}_u - \bar{u}\|_\infty$$

and equivalently $\|\bar{p} - \bar{x}\|_\infty = \|\bar{p}_u - \bar{u}\|_\infty$ such that we have on the one hand

$$(3.8) \quad \|\bar{u} - \hat{z}_u\|_\infty = \|\bar{u} - \hat{p}_u\|_\infty + \underbrace{\|\hat{p}_u - \hat{z}_u\|_\infty}_{=\rho}$$

and

$$(3.9) \quad \|\bar{u} - \bar{z}_u\|_\infty = \|\bar{u} - \bar{p}_u\|_\infty + \underbrace{\|\hat{p}_u - \bar{z}_u\|_\infty}_{=\rho}.$$

On the other hand, we obtain from (3.4) and (3.5) that $\|\bar{u} - \bar{p}_u\|_\infty > \|\bar{u} - \hat{p}_u\|_\infty$ such that we conclude from equations (3.4)-(3.9) that

$$(3.10) \quad \|\bar{u} - \bar{z}_u\|_\infty - \|\bar{u} - \hat{z}_u\|_\infty = \|\bar{u} - \bar{p}_u\|_\infty + \rho - \|\bar{u} - \hat{p}_u\|_\infty - \rho > 0.$$

Remembering that $\bar{z} = [\bar{x}]_{SR}$, such that $\bar{z}_u \equiv [\bar{u}]_{SR}$, and that $\bar{z} \neq \hat{z} \Rightarrow \bar{z}_u \neq \hat{z}_u$ (follows from the definition of W), we know that for at least one component $i \in I = \{1, \dots, l\}$ it holds $\bar{z}_{u,i} \neq \hat{z}_{u,i}$. Let us now define the set

$$I_L := \{i \in I \mid \bar{u}_i \geq 0.5\}.$$

If $|I_L| < S$, we have $\bar{z}_u = [\bar{u}]_{SR} = [\bar{u}]$, where $[\cdot]$ denotes the usual rounding, and it is easy to see that

$$\|\bar{u} - \bar{z}_u\|_\infty \leq \|\bar{u} - \hat{z}_u\|_\infty,$$

thus contradicting (3.10). Therefore, we assume that $|I_L| \geq S$ in the following and define the set I_S , with $|I_S| = S$, so that $\bar{u}_i > \bar{u}_j$ for all $i \in I_S$ and $j \in I_L \setminus I_S$, i.e., the index set of the S largest components of \bar{u} . By the definition of the smart rounding, it is then obvious that $\bar{z}_{u,i} = 1$ for $i \in I_S$ and $\bar{z}_{u,i} = 0$ for $i \in I \setminus I_S$.

Now, any $\tilde{z} \in W$ can be obtained from \bar{z} by considering any combination of the following operations:

1. $\bar{z}_{u,i} = 1 \rightarrow \tilde{z}_{u,i} = 0$ for one $i \in I_S$;
2. $\bar{z}_{u,i} = 1 \rightarrow \tilde{z}_{u,i} = 0$ for one $i \in I_S$ and $\tilde{z}_{u,j} = 1$ for one $j \in I \setminus I_L$;
3. $\bar{z}_{u,i} = 1 \rightarrow \tilde{z}_{u,i} = 0$ for one $i \in I_S$ and $\tilde{z}_{u,j} = 1$ for one $j \in I_L \setminus I_S$.

Since $\bar{u}_i \geq 0.5$ for all $i \in I_S$, the first part of any of these operations results in

$$|\bar{u}_i - \bar{z}_{u,i}| \leq |\bar{u}_i - \tilde{z}_{u,i}|.$$

In the second operation $j \in I \setminus I_L$ implies that $\bar{u}_j < 0.5$ and $\bar{z}_{u,j} = 0$ and we obtain

$$|\bar{u}_j - \bar{z}_{u,j}| \leq |\bar{u}_j - \tilde{z}_{u,j}|.$$

In the third operation $j \in I_L \setminus I_S$ implies that $\bar{u}_j \geq 0.5$ but $\bar{z}_{u,j} = 0$ such that

$$|\bar{u}_j - \bar{z}_{u,j}| \geq |\bar{u}_j - \tilde{z}_{u,j}|.$$

Taking the whole third operation into account and remembering that $i \in I_S$ as well as the definition of the smart rounding, we can see that

$$\max\{|\bar{u}_j - \bar{z}_{u,j}|, |\bar{u}_i - \bar{z}_{u,i}|\} \leq \max\{|\bar{u}_j - \tilde{z}_{u,j}|, |\bar{u}_i - \tilde{z}_{u,i}|\}.$$

Forming any $\tilde{z} \in W$ from \bar{z} via these operations thus implies that

$$\|\bar{u} - \bar{z}_u\|_\infty \leq \|\bar{u} - \tilde{z}_u\|_\infty$$

and as especially $\hat{z}_u \in W$ can be obtained from \bar{z}_u , we have $\|\bar{u} - \bar{z}_u\|_\infty \leq \|\bar{u} - \hat{z}_u\|_\infty$ which is a contradiction to (3.10). Hence, we get that

$$\text{dist}_\infty(\bar{x}, B(\hat{z})) \geq \text{dist}_\infty(\bar{x}, B(\bar{z})), \text{ for all } \hat{z} \in W,$$

which concludes the proof. \square

With this result at hand, we state in Algorithm 3.1 the adaptation of the EXP algorithm from [19, Section 4] to our model problem (Ppen).

Algorithm 3.1 EXP($\varepsilon^0 > 0, \delta^0 > 0, \sigma \in (0, 1)$)

- 1: $n = 0, \varepsilon^n = \varepsilon^0, \delta^n = \delta^0$
 - 2: **Step 1.** Compute $x^n \in X$ such that $J(x^n; \varepsilon^n) \leq J(x; \varepsilon^n) + \delta^n$ for all $x \in X$.
 - 3: **Step 2.**
 - 4: **if** $x^n \notin W$ **and** $J(x^n; \varepsilon^n) - J([x^n]_{SR}; \varepsilon^n) \leq \varepsilon^n \|x^n - [x^n]_{SR}\|_2$ **then**
 - 5: $\varepsilon^{n+1} = \sigma\varepsilon^n, \delta^{n+1} = \delta^n$
 - 6: **else**
 - 7: $\varepsilon^{n+1} = \varepsilon^n, \delta^{n+1} = \sigma\delta^n$
 - 8: **end if**
 - 9: **Step 3.** Set $n = n + 1$ and go to Step 1.
-

Algorithm 3.1 assumes that in Step 1 a so-called δ -global optimizer, i.e., an iterate fulfilling the condition in Step 1, can be found, for example via a global optimization method, see, e.g., [31] for an overview of existing methods. Step 2 of the algorithm then provides a tool to decide when to increase penalization and when to seek for a better global minimizer. The main convergence property of Algorithm 3.1 is reported in the upcoming Proposition 3.6 and shows that Algorithm 3.1 extends global optimization methods for continuous problems to integer problems.

PROPOSITION 3.6. *Every accumulation point x^* of a sequence of iterates $\{x^n\}_{n \in \mathbb{N}}$ of Algorithm 3.1 is a global minimizer of (P).*

Proof. Using Proposition 3.5 the statement follows from [19, Corollary 1]. \square

Before we develop our novel algorithm in the upcoming section, we want to comment on the second condition in line 4 of Algorithm 3.1: this condition is based on [19, (3)], a Hoelder-condition for the unpenalized objective function. Since our objective function J is quadratic, it is Hoelder-continuous with Hoelder-exponent equal to 1. Furthermore, the Hoelder-constant that appears in the original formulation of the algorithm in [19, Section 4], can for simplicity be set to 1 since it only influences the convergence speed of the algorithm. Thus, it does not appear in our formulation.

3.2. Development of the improved penalty algorithm (IPA). Based on the EXP algorithm from the previous section, we now want to develop our novel algorithm. In our practical implementation, we do not require a δ -global optimizer in Step 1 of Algorithm 3.1 for a certain level of δ , but instead want to compute an iterate $x^n \in X$ that simply reduces the objective function such that $J(x^n; \varepsilon^n) < J(x^{n-1}; \varepsilon^n)$. Do note that this corresponds to Step 1 of Algorithm 3.1 but with an unknown rule of reducing δ . Having a look at Algorithm 3.1, one could terminate it as soon as δ is smaller than some tolerance. Since our new approach does not make use of any explicit δ , our algorithm will terminate as soon as we are unable to find a new iterate that reduces the objective function.

Both these adjustments are collected in the Sub-Algorithm 3.2.a, where we aim at improving the current iterate by perturbing it and utilizing this perturbation as initial guess for a local optimization solver. This strategy is closely connected with classic basin hopping or iterated local search strategies, see, e.g., [23], [24], for global optimization problems. The sub-algorithm is then terminated after a certain amount of these perturbation cycles and this bears the information that no better iterate could be found. Gathering these thoughts results in the *improved penalty algorithm (IPA)*, i.e., Algorithm 3.2 reported below.

Algorithm 3.2 Improved penalty algorithm($x^0 \in X, \varepsilon^0 > 0, \sigma \in (0, 1)$)

```

1:  $n = 0, x^n = x^0, \varepsilon^n = \varepsilon^0$ 
2: Step 1. Call Algorithm 3.2.a( $x^n, \varepsilon^n$ ) to generate a new iterate  $x^{n+1}$ .
3: Step 2.
4: if  $x^{n+1} \notin W$  and  $J(x^{n+1}; \varepsilon^n) - J([x^{n+1}]_{SR}; \varepsilon^n) \leq \varepsilon^n \|x^{n+1} - [x^{n+1}]_{SR}\|_2$  then
5:    $\varepsilon^{n+1} = \sigma \varepsilon^n$ 
6: else
7:    $\varepsilon^{n+1} = \varepsilon^n$ 
8: end if
9: Step 3.
10: if  $x^n \equiv x^{n+1}$  then
11:   return  $[x^{n+1}]_{SR}$ 
12: else
13:   Set  $n = n + 1$  and go to Step 1.
14: end if

```

Algorithm 3.2.a Reduction via perturbation($x \in X, \varepsilon > 0$)

```

1:  $x^{init} = x$ 
2: for  $j = 1, \dots, p_{\max}$  do
3:   Use a local optimization solver to determine a solution  $x^{loc}$  of (Ppen) for  $\varepsilon$ 
   using  $x^{init}$  as initial guess.
4:   if  $J(x^{loc}; \varepsilon) < J(x; \varepsilon)$  then
5:     return  $x^{loc}$ 
6:   else
7:     Generate a point  $x^{pert} = Perturbation(x^{loc})$  and set  $x^{init} = x^{pert}$ .
8:   end if
9: end for
10: return  $x$ 

```

Do note that if the for-loop in Algorithm 3.2.a does finish (and thus no better iterate was found after p_{\max} perturbations), the algorithm terminates with x which was the input iterate. In that case it is $x^{n+1} \equiv x^n$ and the overall Algorithm 3.2 then terminates. Therefore, the perturbation strategy in Algorithm 3.2.a together with the choice of p_{\max} give the information at what point no further reduction in the objective function can be found. Algorithm 3.2.a does not specify a perturbation strategy in line 7 and one can develop a strategy that does fit his own problem in question. We will present our strategy in the upcoming Section 4.1.

While ε is decreased during Algorithm 3.2 (and thus the amount of penalization is increased), the concave penalty term (3.1) introduces local minima to $J(x; \varepsilon)$ near

the integer points of X which, as ε is decreased, move towards the integer points. Due to this behavior, the condition $J(x^{loc}; \varepsilon^n) < J(x; \varepsilon^n)$ in line 4 of Algorithm 3.2.a is always fulfilled as long as $\varepsilon^n < \varepsilon^{n-1}$ holds in Algorithm 3.2. By this, we expect Algorithm 3.2 to have a two-phase behavior: in the first phase, the penalization is increased until a feasible iterate $x^{n+1} \in W$ is found. In the second phase, Algorithm 3.2.a then tries to improve the current iterate by perturbing it and restarting the local solver with this perturbed iterate. This way, one wants to escape bad basins of attraction of J and then move towards better local minima and eventually the global one.

Depending on the perturbation strategy and on the choice of p_{\max} , the search strategy described in Algorithm 3.2.a would again find a δ -global optimizer (for an unknown δ) and Algorithm 3.2 would reproduce Algorithm 3.1. The main difference now is that Algorithm 3.2.a only requires a local solver. Therefore, Algorithm 3.2 will be applicable to large-scale problems: the combinatorial complexity induced by a large control dimension l is taken care of via the penalty term as well as the framework of the EXP algorithm, and a large state dimension N induced by, e.g., a parabolic or a 3d PDE constraint can be handled via a sophisticated local solver.

Finally, we want to mention that a new iterate $x^{n+1} = [y^{n+1}, u^{n+1}]^\top$ found by Algorithm 3.2.a is always feasible so that $x^{n+1} \in X$. Thus, the criterion $x^{n+1} \notin W$ in line 4 of Algorithm 3.2 can, in an actual implementation, be replaced by

$$\|u^{n+1} - [u^{n+1}]_{SR}\|_\infty > \varepsilon_{feas}$$

with some feasibility tolerance ε_{feas} . Thus, it is reasonable to return $[x^{n+1}]_{SR}$ such that the control of our output iterate is always integer and respects the knapsack constraint.

4. Implementation details, local solver, and numerical setup. We begin with a discussion on various details of our implementation of the IPA, i.e., Algorithm 3.2/3.2.a, including the perturbation strategy and the local solver. Afterwards, we shortly introduce two other solution strategies for problem (P) and discuss the setup and parameter choices for the numerical investigation that will be carried out in Section 5.

4.1. Implementation details of the IPA. We start with the presentation of our perturbation strategy used in Algorithm 3.2.a. The details are described in Algorithm 3.2.b.

As mentioned in Section 3.2, this perturbation strategy should only be called upon in the later stages of Algorithm 3.2 where the amount of penalization is significant enough such that the set I_S in Algorithm 3.2.b is not empty. When Algorithm 3.2.b is called by Algorithm 3.2.a inside Algorithm 3.2, x is equal to the current iterate $x^n = [y^n, u^n]^\top$. The algorithm then essentially performs $\theta \in \mathbb{N}$ flips to the current control u^n , where a flip is one iteration of the for-loop of Algorithm 3.2.b, i.e., a large value of u^n is set to a small value and an entry of u^n corresponding to a source that is *adjacent* to the source corresponding to the large value is set to a large value. By this strategy the resulting perturbation x^{pert} possibly lies outside the current basin of attraction and therefore might be an initial guess for the local solver in Algorithm 3.2.a resulting in a point with a potentially better function value. It remains to explain what we mean by *adjacent* in the above context.

For now, we assume that the centers $\tilde{x}_1, \dots, \tilde{x}_l$ of our source functions are arranged in a uniform $m \times m$ grid over Ω (and make more specifications in Section 4.4). Using

Algorithm 3.2.b Perturbation($x \in X$)

-
- 1: Split $x = (y, u)^\top$ into the state $y \in \mathbb{R}^N$ and control $u \in \mathbb{R}^l$. Define $u^{pert} := u$.
 - 2: Find I_S , the set containing the indices of the entries of u that are larger than $\frac{1}{2}$.
 - 3: **for** $j = 1, \dots, \min\{|I_S|, \theta\}$ **do**
 - 4: Randomly select $\hat{i} \in I_S$.
 - 5: Define I_{adj} the set of indices corresponding to sources *adjacent* to $\tilde{x}_{\hat{i}}$.
 - 6: Randomly select $\hat{i}_{adj} \in I_{adj}$.
 - 7: Set $(u^{pert})_{\hat{i}}$ to a randomly chosen value smaller than $\frac{1}{2}$.
 - 8: Set $(u^{pert})_{\hat{i}_{adj}}$ to a randomly chosen value larger than $\frac{1}{2}$.
 - 9: Remove \hat{i} from I_S .
 - 10: **end for**
 - 11: Compute the state y^{pert} corresponding to u^{pert} , i.e., $y^{pert} = f(u^{pert})$.
 - 12: **return** $x^{pert} := [y^{pert}, u^{pert}]^\top$
-

this, we define I_{adj} the index set of sources adjacent to $\tilde{x}_{\hat{i}}$ in the case when $\tilde{x}_{\hat{i}}$ is an interior source as

$$(4.1) \quad \{\hat{i} - 1, \hat{i} + 1, \hat{i} - m - 1, \hat{i} - m, \hat{i} - m + 1, \hat{i} + m - 1, \hat{i} + m, \hat{i} + m + 1\}.$$

Since $\tilde{x}_{\hat{i}}$ is assumed to correspond to an interior source this set corresponds to the indices of all sources that are direct neighbors of $\tilde{x}_{\hat{i}}$ in the $m \times m$ grid of sources. Obviously this set has to be adjusted if $\tilde{x}_{\hat{i}}$ is on the boundary of the source grid.

Although the perturbation strategy presented so far depends on the uniform grid of source centers in order to determine the index set I_{adj} , the underlying concept of this *flipping* does not depend on the chosen modelling. The large component $(u^{pert})_{\hat{i}}$ of the control can often be associated to a spatial counterpart denoted, for the purpose of clarity, as $x_{\hat{i}}$ here. In our case this is $x_{\hat{i}} = \tilde{x}_{\hat{i}}$, the center of the Gaussian source function. If the control would for example be modeled via piecewise constant functions $\{\chi_i(x)\}_{i=1}^l$ (as in [29] or Section 5.2), $x_{\hat{i}}$ could be the center of the patch of the subdomain that corresponds to $\chi_{\hat{i}}(x)$. If the control would be distributed, $x_{\hat{i}}$ would be the vertex of the grid that corresponds to $u_{\hat{i}}$. Finding the set of *adjacent indices* in line 5 of Algorithm 3.2.b then translates to selecting all indices that correspond to spatial counterparts \bar{x} with $\|x_{\hat{i}} - \bar{x}\|_2 \leq r$, where the radius $r > 0$ controls the degree of *adjacency*. The set of indices in (4.1) could thus be generated with the radius $r = \frac{\sqrt{2}}{m+1}$.

With this interpretation, as long as the control can be associated to spatial counterparts of the domain Ω , the presented perturbation strategy can easily be applied to different kinds of controls, models, and domains.

Finally, we found it effective in our experiments to set $(u^{pert})_{\hat{i}}$ to a random value in $[0.1, 0.2]$ during Algorithm 3.2.b. Afterwards, we calculate $d_{\hat{i}} := |(u)_{\hat{i}} - (u^{pert})_{\hat{i}}|$ and set $(u^{pert})_{\hat{i}_{adj}}$ to a random value in $[d_{\hat{i}} - 0.1, d_{\hat{i}}]$. This strategy ensures that the perturbed control u^{pert} is still feasible (especially fulfilling the knapsack constraint). Furthermore, this prohibits the perturbed control of having values that are too close to 0 or 1. By this, x^{pert} is then an initial guess for the local solver in Algorithm 3.2.a that (possibly) lies outside the current basin of attraction and is at the same time not too close to other local minimizers (at this stage of the IPA there are local minimizers nearby all integer points).

In the remainder of this section, we want to discuss the termination of Algorithm

3.2.a and thus Algorithm 3.2. The criterion $J(x^{loc}; \varepsilon^n) < J(x^n; \varepsilon^n)$ in Algorithm 3.2.a (as it is called inside Algorithm 3.2 with $x = x^n$ and $\varepsilon = \varepsilon^n$) can be numerically challenging in an actual implementation. Although the criterion should be fulfilled when it was $\varepsilon^n < \varepsilon^{n-1}$ in Algorithm 3.2 as mentioned in Section 3.2, this might not be the case numerically since any local solver used in Algorithm 3.2.a only computes $x^{loc} = [y^{loc}, u^{loc}]^\top$ up to some internal tolerance. Furthermore, if x^n is close to an integer already, we do not want to accidentally fulfill $J(x^{loc}; \varepsilon^n) < J(x^n; \varepsilon^n)$ due to numerical effects although $[u^{loc}] \equiv [u^n]$ such that no progress towards a better integer solution would be made. To cover both of these cases in our implementation, we first calculate the two distances

$$d^{loc} := \|u^{loc} - u^n\|_\infty \quad \text{and} \quad d_{SR}^{loc} := \|[u^{loc}]_{SR} - [u^n]_{SR}\|_\infty$$

and replace $J(x^{loc}; \varepsilon^n) < J(x^n; \varepsilon^n)$ by the following two criteria and thus return x^{loc} in Algorithm 3.2.a if one of these is fulfilled.

1. If $\varepsilon^n < \varepsilon^{n-1}$ and either $J(x^{loc}; \varepsilon^n) < J(x^n; \varepsilon^n)$, $d^{loc} < 0.2$, or $d_{SR}^{loc} = 0$ are fulfilled.
2. If $\varepsilon^n = \varepsilon^{n-1}$ and $d_{SR}^{loc} \neq 0$, as well as $J(x^{loc}; \varepsilon^n) < J(x^n; \varepsilon^n)$, and additionally $J([x^{loc}]_{SR}; \varepsilon^n) < J([x^n]_{SR}; \varepsilon^n)$ are fulfilled.

The first criterion targets the case when $J(x^{loc}; \varepsilon^n) \not< J(x^n; \varepsilon^n)$ numerically (although it was $\varepsilon^n < \varepsilon^{n-1}$ in Algorithm 3.2) and thus also accepts iterates that are either close to, or presumably in the same basin of attraction as, the previous iterate. We mention that this usually happens during the first phase of the IPA where the amount of penalization is increased (due to $\varepsilon^{n+1} = \sigma\varepsilon^n$) and is not yet large enough for the local solver to produce near integer solutions fulfilling $\|u^{loc} - [u^{loc}]_{SR}\|_\infty \leq \varepsilon_{feas}$. As a result it is not necessary to search for better solutions via the perturbation strategy such that this criterion tries to prevent non-productive iterations in Algorithm 3.2.a. If a feasible iterate was found and the amount of penalization was not increased, the second criterion only accepts better iterates that lie outside the current basin of attraction and thus enforces progress towards a better integer solution and should prevent the algorithm from getting stuck in an unsatisfactory local minimum.

4.2. Implementation of the local solver: an interior point framework for the large scale setting. We now discuss our implementation of line 3 in Algorithm 3.2.a, that is the choice of the local solver for finding a solution x^{loc} of (Ppen) for a given ε . Due to the structure of (Ppen), which was equivalent to (3.2), we opt for an interior point method (IPM) that is particularly suitable for solving quadratic programming problems and it also allows the use of an efficient preconditioner in the linear algebra phase, see, e.g., [32], [33]. Following [33], we present the derivation of a standard interior point method for the following reformulation of problem (Ppen), that is

$$\begin{aligned} \min_{y \in \mathbb{R}^N, u \in \mathbb{R}^l, z \in \mathbb{R}} \quad & J(y, u; \varepsilon) = \frac{1}{2}(y - y_d)^\top M(y - y_d) + \frac{1}{\varepsilon}(\mathbf{1}^\top u - u^\top u), \\ \text{s.t.} \quad & Ky = M\Phi u \quad \text{and} \quad \mathbf{1}^\top u + z - S = 0, \\ & 0 \leq u \leq 1 \quad \text{and} \quad z \geq 0, \end{aligned}$$

where $z \geq 0$ is a scalar slack variable and the notation has been adapted to distinguish the control u and the state y . For the sake of generality we include the case when the stiffness matrix K is non-symmetric. The main idea of an IPM is the elimination of

the inequality constraints on u and z via the introduction of corresponding logarithmic barrier functions. The resulting Lagrangian associated with the barrier subproblem reads

$$L_{\mu,\varepsilon}(y, u, z; p, q) = J(y, u; \varepsilon) + p^\top (Ky - M\Phi u) + q(\mathbf{1}^\top u + z - S) - \mu \sum_{i=1}^l \log(u_i) - \mu \sum_{i=1}^l \log(1 - u_i) - \mu \log(z),$$

where $p \in \mathbb{R}^N$ is the Lagrange multiplier (or adjoint variable) associated with the state equation, $q \in \mathbb{R}$ is the Lagrange multiplier associated with the scalar equation $\mathbf{1}^\top u + z - S = 0$, and $\mu > 0$ is the barrier parameter that controls the relation between the barrier term and the original objective $J(y, u; \varepsilon)$. As the method progresses, μ is decreased towards zero.

First-order optimality conditions are derived by applying duality theory resulting in a nonlinear system parametrized by μ as detailed below. Thus, differentiating $L_{\mu,\varepsilon}$ with respect to the variables y, u, z, p, q gives the nonlinear system

$$(4.2a) \quad My - My_d + K^\top p = 0,$$

$$(4.2b) \quad \frac{1}{\varepsilon}(\mathbf{1} - 2u) - \Phi^\top Mp + q\mathbf{1} - \lambda_{u,0} + \lambda_{u,1} = 0,$$

$$(4.2c) \quad q - \lambda_{z,0} = 0, \quad Ky - M\Phi u = 0, \quad \mathbf{1}^\top u + z - S = 0,$$

where the Lagrange multipliers $\lambda_{u,0}, \lambda_{u,1} \in \mathbb{R}^l$ and $\lambda_{z,0} \in \mathbb{R}$ are defined as

$$(\lambda_{u,0})_i := \frac{\mu}{u_i}, \quad (\lambda_{u,1})_i := \frac{\mu}{1 - u_i} \quad \text{for } i = 1, \dots, l, \quad \text{and} \quad \lambda_{z,0} := \frac{\mu}{z}.$$

Furthermore, the bound constraints $\lambda_{u,0} \geq 0$, $\lambda_{u,1} \geq 0$ and $\lambda_{z,0} \geq 0$ then enforce the constraints on u and z .

The crucial step of deriving the IPM, is the application of Newton's method to the above nonlinear system. Letting $y, u, z, p, q, \lambda_{u,0}, \lambda_{u,1}$ and $\lambda_{z,0}$ denote the most recent Newton iterates, these are then updated in each iteration by computing the corresponding Newton steps $\Delta y, \Delta u, \Delta z, \Delta p, \Delta q, \Delta \lambda_{u,0}, \Delta \lambda_{u,1}$ and $\Delta \lambda_{z,0}$ through the solution of the following Newton system

$$(4.3) \quad \begin{bmatrix} M & 0 & 0 & K^\top & 0 \\ 0 & -\frac{2}{\varepsilon}I_l + \Theta_u & 0 & -\Phi^\top M & \mathbf{1} \\ 0 & 0 & \theta_z & 0 & 1 \\ K & -M\Phi & 0 & 0 & 0 \\ 0 & \mathbf{1}^\top & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta u \\ \Delta z \\ \Delta p \\ \Delta q \end{bmatrix} = - \begin{bmatrix} My - My_d + K^\top p \\ \frac{1}{\varepsilon}(\mathbf{1} - 2u) - \Phi^\top Mp + q\mathbf{1} - \lambda_{u,0} + \lambda_{u,1} \\ q - \lambda_{z,0} \\ Ky - M\Phi u \\ \mathbf{1}^\top u + z - S \end{bmatrix}.$$

Here, $\Theta_u := U^{-1}\Lambda_{u,0} + (I_l - U)^{-1}\Lambda_{u,1}$, $\theta_z := \lambda_{z,0}/z$, and $U, \Lambda_{u,0}$, and $\Lambda_{u,1}$ are diagonal matrices with the most recent iterates of $u, \lambda_{u,0}$, and $\lambda_{u,1}$ appearing on their diagonal entries. The matrices Θ_u and $\theta_z > 0$, while being positive definite, are typically very ill-conditioned. Also, due to the term $-\frac{2}{\varepsilon}I_l$, the block $-\frac{2}{\varepsilon}I_l + \Theta_u$ may

be indefinite, especially for small values of ε . Following suggestions in [32, Chapter 19.3] to handle nonconvexities in the objective function by promoting the computation of descent directions, we heuristically keep the diagonal matrix $-\frac{2}{\varepsilon}I_l + \Theta_u$ positive definite by setting any negative values to some value $\gamma > 0$.

Once the above system is solved, one can compute the steps for the Lagrange multipliers via

$$\begin{aligned}\Delta\lambda_{u,0} &= -U^{-1}\Lambda_{u,0}\Delta u - \lambda_{u,0} + \mu U^{-1}\mathbf{1}, \\ \Delta\lambda_{u,1} &= (I_l - U)^{-1}\Lambda_{u,1}\Delta u - \lambda_{u,1} + \mu(I_l - U)^{-1}\mathbf{1}, \\ \Delta\lambda_{z,0} &= -(\lambda_{z,0}/z)\Delta z - \lambda_{z,0} + \mu/z.\end{aligned}$$

A general IPM implementation only involves one Newton step per iteration. Thus, after choosing suitable step-lengths so that the updated iterates remain feasible, the new iterates can be calculated and the barrier parameter μ is reduced, thus concluding one iteration of the IPM. Finally, we report the primal and dual feasibilities

$$\xi_p := \begin{bmatrix} Ky - M\Phi u \\ \mathbf{1}^\top u + z - S \end{bmatrix} \quad \text{and} \quad \xi_d := \begin{bmatrix} My - My_d + K^\top p \\ \frac{1}{\varepsilon}(\mathbf{1} - 2u) - \Phi^\top Mp + q\mathbf{1} - \lambda_{u,0} + \lambda_{u,1} \\ q - \lambda_{z,0} \end{bmatrix},$$

as well as the complementarity gap

$$\xi_c := \begin{bmatrix} U\lambda_{u,0} - \mu\mathbf{1} \\ (I_l - U)\lambda_{u,1} - \mu\mathbf{1} \\ z\lambda_{z,0} - \mu \end{bmatrix},$$

where measuring the change in the norms of ξ_p , ξ_d and ξ_c allows us to monitor the convergence of the entire process.

Clearly, the computational burden of this IPM lies in the solution of the Newton system (4.3) and our strategy regarding this issue is twofold: on the one hand we employ an inexact Newton-Krylov strategy for the solution of the nonlinear system (4.2) and on the other hand we design a suitable preconditioner to speed up the convergence of our Krylov method of choice for the Newton system (4.3). Regarding the inexactness strategy, the idea is to increase the accuracy in the solution of the Newton equation as μ decreases. This minimizes the occurrence of so-called *oversolving* in the first interior point steps. Global convergence results to a solution of the first-order optimality conditions for the resulting inexact IPM can be found in [34].

We will now present our linear algebra strategy for the solution of the Newton system, i.e., we choose our Krylov method and design a suitable preconditioner.

Preconditioning for the interior point method. Investigating the system matrix of the Newton system (4.3), we observe that with the choice

$$A = \begin{bmatrix} M & 0 & 0 \\ 0 & -\frac{2}{\varepsilon}I_l + \Theta_u & 0 \\ 0 & 0 & \theta_z \end{bmatrix}, \quad B = \begin{bmatrix} K & -M\Phi & 0 \\ 0 & \mathbf{1}^\top & 1 \end{bmatrix}$$

we have to solve a saddle point system $\begin{bmatrix} A & B^\top \\ B & 0 \end{bmatrix}$. As discussed already, the block $-\frac{2}{\varepsilon}I_l + \Theta_u$ is kept positive definite throughout the interior point method, so that we can assume that A is positive definite.

Such systems are a cornerstone of applied mathematics and appear in many application scenarios, see, e.g., [35], [36]. While the system is symmetric and we could apply MINRES [37], we here use a nonsymmetric method, namely GMRES [38], because we found that a block-triangular preconditioner

$$P = \begin{bmatrix} \hat{A} & 0 \\ B & -\hat{S} \end{bmatrix}$$

performs better in our experiments. It would also be possible to use symmetric solvers, which are based on nonstandard inner products, see, e.g., [39], [40].

We here focus on the design of the approximations for the $(1, 1)$ -block $\hat{A} \approx A$ and for the Schur-complement

$$\hat{S} \approx S = \begin{bmatrix} KM^{-1}K^\top + M\Phi(-\frac{2}{\varepsilon}I_l + \Theta_u)^{-1}\Phi^\top M & 0 \\ 0 & \mathbf{1}^\top(-\frac{2}{\varepsilon}I_l + \Theta_u)^{-1}\mathbf{1} + \theta_z^{-1} \end{bmatrix}.$$

In our preconditioning approach, we neglect the term $\mathbf{1}^\top(-\frac{2}{\varepsilon}I_l + \Theta_u)^{-1}\mathbf{1} + \theta_z^{-1}$ and set the preconditioner to 1 as this typically does not result in many additional iterations and we avoid dealing with the ill-conditioning of both $(-\frac{2}{\varepsilon}I_l + \Theta_u)$ and θ_z . In our setup here we thus end up with the following approximation

$$\hat{S} = \begin{bmatrix} KM^{-1}K^\top & 0 \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad A = \hat{A}.$$

We close this section with two short remarks.

Remark 4.1. The purpose of this basic preconditioner is to speed up the solution process of our IPM, but for future research we need to enhance this based on recent progresses in preconditioning for interior point methods, see, e.g., [41]–[43].

Remark 4.2. Although this IPM together with the preconditioner are formulated for the penalty formulation (Ppen) that refers to the model problem (2.1), it is clear that the IPM generalizes to general linear PDE constraints (in fact, Section 5.2 will contain experiments for a convection-diffusion problem resulting in a nonsymmetric stiffness matrix K). Furthermore, the IPM and the preconditioner can be formally adapted to a nonlinear PDE constraint $F(y, u) = 0$, where $F : \mathbb{R}^{N+l} \rightarrow \mathbb{R}^N$ is a smooth nonlinear function. One simply has to introduce $F'(y, u) \in \mathbb{R}^{N \times (N+l)}$, the Jacobian of F as well as $F'_y \in \mathbb{R}^{N \times N}$ and $F'_u \in \mathbb{R}^{N \times l}$, the submatrices of the Jacobian such that $F'(y, u) = [F'_y, F'_u]$. We then obtain the IPM for this nonlinear problem by replacing in the Newton system (4.3) K^\top by $(F'_y)^\top$, $-M\Phi$ by F'_u (and thus $-\Phi^\top M$ by $(F'_u)^\top$), and $Ky - M\Phi u$ by $F(y, u)$. In the nonlinear case, convergence of the IPM is ensured when embedded in suitable globalization strategies [32].

4.3. Simple penalty and branch-and-bound method. We shortly discuss a simple penalty approach and a branch-and-bound strategy for the solution of (P) to which we want to compare our IPA in the numerical Section 5.

Starting with the penalty formulation (Ppen), we use the following naive iterative approach: given a feasible initial guess $x^0 \in X$, an initial penalty parameter $\varepsilon^0 > 0$ and $n = 0$

1. use a local solver to determine a solution $x^{n+1} \in X$ of (Ppen) for ε^n using x^n as initial guess,
2. stop if $x^{n+1} \in W$, else set $\varepsilon^{n+1} = \sigma\varepsilon^n$ ($\sigma \in (0, 1)$), $n = n + 1$ and go to the first step.

As mentioned at the end of Section 3.2, instead of checking for $x^{n+1} \in W$ one can use the criterion $\|u^{n+1} - [u^{n+1}]_{SR}\|_\infty < \varepsilon_{feas}$. Putting these thoughts into an algorithm, we end up with the *penalty algorithm*, i.e., Algorithm 4.1.

Algorithm 4.1 Penalty($x^0, \varepsilon^0, \sigma \in (0, 1)$)

- 1: $n = 0, x^n = x^0, \varepsilon^n = \varepsilon^0$
 - 2: **repeat**
 - 3: Use a local solver to determine a solution x^{n+1} of (Ppen) for ε^n using x^n as initial guess.
 - 4: $\varepsilon^{n+1} = \sigma \varepsilon^n$
 - 5: $n = n + 1$
 - 6: **until** $\|u^n - [u^n]_{SR}\|_\infty < \varepsilon_{feas}$
 - 7: **return** $[x^n]_{SR}$
-

Algorithm 4.1 is a simplification of Algorithm 3.2 in several ways. The penalty parameter is reduced in every iteration, a new iterate x^{n+1} generated by the local solver is always accepted as such, and the algorithm terminates as soon as an iterate $x^{n+1} \in W$ is found. By this, there is no theoretical framework that might support the algorithm anymore and one has to hope that the iterates approach the global minimizer of (P), or at least a good local minimizer.

We can also interpret Algorithm 4.1 as a more sophisticated rounding strategy: starting from the initial guess x^0 , by slowly decreasing ε (and thus increasing penalization of a non-binary control), the control is then component-wise slowly driven towards either 0 or 1.

Regarding the branch-and-bound routine, we neither describe the branch-and-bound framework in detail here nor present an actual branch-and-bound routine. We rather refer the reader to [12] for an elaborate overview of the topic. For the numerical comparison that will be carried out in Section 5, we will utilize `cplexmiqp`, the branch-and-bound routine of CPLEX [25] for quadratic mixed integer problems, to solve (P). Do note that this algorithm incorporates many algorithmic features lately developed to improve branch-and-bound performance.

4.4. Numerical setting and parameter choices. We present the general setting in which the numerical experiments will be conducted as well as default parameter choices for the algorithms. If different choices are utilized, it is mentioned in the respective experiment.

We choose $\Omega := [0, 1]^2$ as our computational domain for the numerical experiments. Regarding the Gaussian sources defined in (2.2), we choose $l = 100$ sources with centers $\tilde{x}_1, \dots, \tilde{x}_l$ being arranged in a uniform 10×10 grid over $\tilde{\Omega} := [0.1, 0.9]^2 \subset \Omega$ (thus, $m = 10$ w.r.t. (4.1)). The height of the sources is $\kappa = 100$ and the width ω is chosen such that every source takes 5% of its center-value at a neighboring center. We mention that this choice of height and width is motivated by [6, Section 4.2]. The PDE (2.1) is discretized using uniform piece-wise linear finite elements with a step size of 2^{-7} (unless specified otherwise) resulting in $N = 16641$ vertices.

Whenever a local solver is required, i.e., in Algorithms 3.2.a and 4.1, we use the interior point method derived in Section 4.2. The outer interior point iteration is stopped as soon as either $\max\{\|\xi_p\|_2, \|\xi_d\|_2, \|\xi_c\|_2\} \leq 10^{-6}$ or the safeguard $\mu \leq 10^{-15}$ is triggered. Furthermore, starting from an initial $\mu = 1$ we decrease μ by factor 0.1 in each outer interior point iteration. The inexactness is implemented by stopping GMRES when the norm of the unpreconditioned relative residual is below

$\eta = \max\{\min\{10^{-1}, \mu\}, 10^{-10}\}$. Finally, the block $-\frac{2}{\varepsilon}I_l + \Theta_u$ in the Newton system (4.3) is kept positive definite by setting any negative values to $\gamma = 10^{-6}$ and the preconditioner proposed at the end of Section 4.2 is applied by performing the Cholesky decomposition of both M and K once at the beginning of the IPA process.

As initial guess for Algorithms 3.2 and 4.1 the solution of (Pcont) obtained by our IPM is used. Do note that this is not necessary since (Ppen) for large enough ε^0 is usually still a convex problem in the first iteration of these algorithms so that any initial guess would be sufficient.

Further default parameters are $\varepsilon^0 = 10^5$ for both algorithms as well as $\sigma = 0.9$ for Algorithm 4.1 and $\sigma = 0.7$ for Algorithm 3.2. The more conservative value of σ for Algorithm 4.1 is necessary here, since with σ being closer to 0 one would risk increasing the amount of penalization too fast and thus possibly 'skipping' a good local minimum and settling for an unsatisfactory local minimum. Finally, we used the feasibility tolerance $\varepsilon_{feas} = 0.1$ in Algorithm 4.1 and, as discussed at the end of Section 3.2, in line 4 of Algorithm 3.2.

Regarding `cplexmiqp`, we use default options except that we set a time limit of 1 hour (unless specified otherwise) and a memory limit of 16000 megabytes for the search tree.

All experiments were conducted on a PC with 32 GB RAM and a QUAD-Core-Processor INTEL-Core-I7-4770 (4x 3400MHz, 8 MB Cache) utilizing Matlab 2019a via which CPLEX 12.9.0 was accessed.

5. Numerical Experiments. We begin with four different experiments for our Poisson model problem (P) and then shortly discuss a convection-diffusion problem as well as the behaviour of our local solver.

5.1. Poisson model problem. In the first experiment we want to investigate the performance of the improved penalty algorithm (IPA), i.e., Algorithm 3.2, with respect to $p_{\max} \in \mathbb{N}$ and $\theta \in \mathbb{N}$, i.e., how these choices in Algorithms 3.2.a and 3.2.b affect the solution quality and solution time of the overall algorithm. In the second experiment we want to see that the IPA can indeed handle large scale problems and convince ourselves that `cplexmiqp`, the branch-and-bound method of CPLEX introduced in Section 4.3, can not handle large scale problems. In the third experiment we then carry out a detailed comparison of the IPA with the solution strategies presented in Section 4.3. In the final experiment we investigate the robustness of the stochastic component of the IPA, i.e., how the random choices in Algorithm 3.2.b affect the solution time and quality of the overall algorithm. Do note that due to the different implementation languages included in these experiments, the reported computational times only give a qualitative information on the performance of the solvers.

First experiment. For the first experiment we generate a test set of desired states, which contains 20 different desired states for each value of $S \in \{3, 6, 10, 15, 20\}$. Each desired state y_d in this test set is a solution of (the discretized version of) (2.1) with S active sources in the right-hand side and the centers of these sources are randomly distributed over $\tilde{\Omega} = [0.1, 0.9]^2$. The height and width of these sources coincide with the values that were used for the source-grid in Section 4.4. Clearly, the combinatorial complexity of the optimization problem corresponding to such a desired state increases drastically for larger values of S and this will be a good challenge for `cplexmiqp` as well as Algorithm 3.2 throughout the various experiments. To further illustrate the optimization problem here, Figure 5.1 exemplarily shows two desired states, one for $S = 3$ and one for $S = 20$, where the white stars depict $\tilde{x}_1, \dots, \tilde{x}_l$, the centers of the source grid introduced in Section 4.4, and the red stars depict the centers of the S

active sources in y_d .

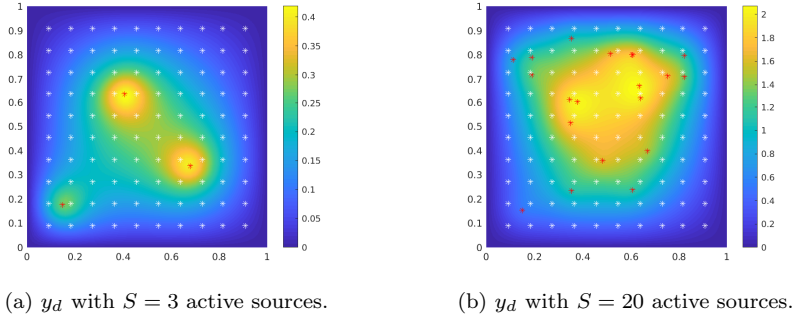


Fig. 5.1: Exemplary desired states including the centers of the source grid (white stars) and the centers of the active sources of the respective desired state (red stars).

In order to investigate the IPA with respect to $p_{\max} \in \mathbb{N}$, the amount of perturbation cycles in Algorithm 3.2.a, and $\theta \in \mathbb{N}$, the amount of flips per perturbation in Algorithm 3.2.b, we solve the created test set with the IPA using the following nine different variations.

- variations 1 – 3: it is always $\theta = 1$ and p_{\max} varies as $p_{\max} \in \{100, 300, 500\}$.
- variations 4 – 6: it is always $\theta = 3$ and p_{\max} varies as $p_{\max} \in \{100, 300, 500\}$.
- variations 7 – 9: for $S \in \{3, 6, 10, 15, 20\}$ we always choose $\theta \in \{1, 2, 3, 4, 5\}$ and p_{\max} varies as $p_{\max} \in \{100, 300, 500\}$.

After solving the test set with these nine variations of the IPA, we compare the results with respect to solution time and quality. For the solution time, we report 't_av' the average solution time in seconds and for the solution quality, we choose the following two criteria.

- 'min_count': in each of the 20 runs for a value of S , it is noted which algorithm (here the nine variations of the IPA) achieved the smallest objective function value and this algorithm is then awarded a 'min_count'-score. Surely, multiple algorithms can be awarded a score in the same run (when multiple algorithms find the same 'best' minimum).
- 'rel_err_av': the average relative error. In each of the 20 runs for a value of S , we store for each algorithm the relative error between the objective function value achieved by that algorithm and the smallest objective function value in that run (the one that was awarded a 'min_count'-score). Thus, only runs that produced a non-zero relative error are taken into account when computing the average relative error for an algorithm.

We chose to measure the quality of the algorithms via the described two quantities since, as the centers of the desired states in the test set are randomly distributed over Ω , the global minimum of the optimization problem is not known analytically. Therefore, the 'min_count'-value simply tells us how often an algorithm performed best compared to the other algorithms. The average relative error is an additional measure of quality.

The results of this first experiment can be seen in Figure 5.2, where the data, i.e., 'min_count', 'rel_err_av', and 't_av', is displayed for the 100 instances of the whole test set (resulting in a total average error and total average solution time). The data

is displayed in more detailed fashion (splitting up the test set for each value of S) in the Appendix.

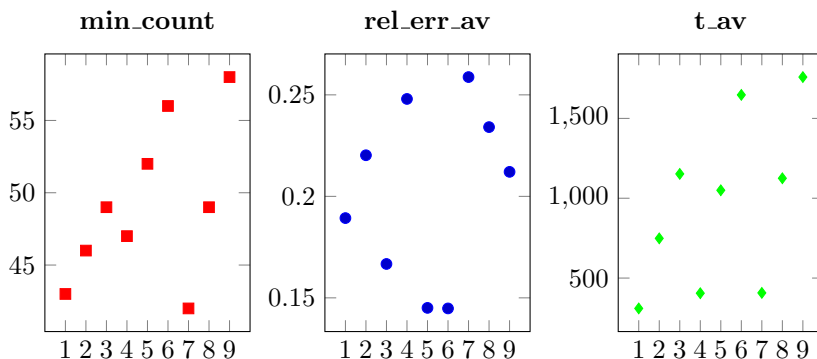


Fig. 5.2: Results of the first experiment. The respective data is plotted over the nine different variations of the IPA.

As the x-axis represents the nine different variations of the IPA, the left-most data point in Figure 5.2 exemplarily means: from the 100 different desired states in the test set, the IPA with variation 1 described above, i.e., $\theta = 1$ and $p_{\max} = 100$, got a 'min_count'-score of 43. Remembering the definition of the 'min_count', this means that compared to the other eight variations of the IPA, variation 1 found the smallest objective function value available amongst these nine variations in 43 out of the 100 runs. Similarly, the remaining data points in Figure 5.2 have to be understood.

Now, having a look at the average solution time, we can observe that increasing p_{\max} has much more influence on the solution time than increasing θ . This is expected as an increase in p_{\max} directly results in more calls of the local solver in Algorithm 3.2.a as the IPA can only terminate when it was confirmed in Algorithm 3.2.a that no better iterate was found after p_{\max} iterations.

As the main observation of this experiment, we can see that variations 5, 6, and 9 seem to be the most competitive ones. While variation 9 has the overall highest 'min_count'-score, it has a rather large average relative error (compared to variations 5 and 6) and it is also the overall most time consuming one. Variation 6 has again a very good 'min_count'-score and also convinces with the lowest average relative error, although it is the second most time consuming variation. Variation 5, being the computationally cheapest variation (out of these three), does still have a good 'min_count'-score and relative errors very much comparable to variation 6.

As a result from this first experiment, we choose variation 5 and fix $p_{\max} = 300$ as well as $\theta = 3$ in the IPA for the remaining experiments. This choice should yield a reasonable balance between solution time and solution quality and prevent an 'overfitting' of these parameters towards the test set used in this experiment.

Second experiment. We now want to see that the IPA can handle large-scale problems and `cplexmiqp`, the branch-and-bound routine of CPLEX, can not. Therefore, we create a problem instance per value of $S \in \{3, 10, 20\}$ and per step-size $h \in \{2^{-7}, 2^{-8}\}$ of the FEM grid and solve each instance with the IPA, `cplexmiqp` with a 1 hour time limit, `cplexmiqp` with a 10 hour time limit, and (for comparison reasons) with the simple penalty approach from Algorithm 4.1. Regarding the solution quality, the algorithm with the lowest objective function value is indicated with

a 'min' in Table 5.1 (or a 'min*' if it was the global minimum) and for each other algorithm the relative error towards this minimum objective function value is then displayed. Furthermore, Table 5.1 contains the run times in seconds for each algorithm in each instance, where in case of `cplexmiqp` 'TL' indicates that the respective time limit was reached.

h	S		Penalty	IPA	<code>cplexmiqp</code> 1h	<code>cplexmiqp</code> 10h
2^{-7}	3	rel_err	min*	min*	min*	-
		time (s)	89	925	1527	
	10	rel_err	20%	13%	13%	min
time (s)		163	1035	TL	TL	
20	rel_err	57%	min	19%	1%	
	time (s)	188	1143	TL	TL	
2^{-8}	3	rel_err	min	min	6805%	6805%
		time (s)	541	6219	TL	TL
	10	rel_err	8%	min	45535%	45535%
time (s)		1101	7550	TL	TL	
20	rel_err	12%	min	93718%	93718%	
	time (s)	1400	9190	TL	TL	

Table 5.1: Results of the second experiment. For each problem instance the algorithm with the lowest objective function value is indicated. The respective relative error of other algorithms as well as the solution times are furthermore reported.

We observe that for $h = 2^{-7}$ and $S = 3$ all algorithms find the global minimum, although we stress that this is only a single problem instance which does not allow for a conclusive comparison with respect to solution quality. A more detailed comparison will be carried out in the next experiment. With an increase in S (and thus an increase in the combinatorial complexity of the problem), `cplexmiqp`, while hitting the prescribed time limit, is still able to provide good solutions, although our IPA is able to at least keep up with `cplexmiqp`. Refining the FEM-mesh once and thus moving towards $h = 2^{-8}$ (resulting in $N = 66049$ instead of $N = 16641$) results in `cplexmiqp` not being able to handle the problem at all. The time limit is always reached and the algorithm (even given 10 hours time) terminates with a tremendous relative error w.r.t. the qualitative solutions found by our IPA. The solution found by the IPA should then by construction always be better than the solution found by the simple penalty algorithm. One might be tempted to believe that the simple penalty algorithm could also be a viable alternative due to its inherent fast solution time but the next experiment will reveal that the algorithm cannot produce qualitative points in a reliable way.

Third experiment. In this third experiment we carry out a detailed comparison between the IPA, the penalty algorithm in Algorithm 4.1 and `cplexmiqp`. In order to do so, we construct another test set as described in the first experiment and as the desired states in these test sets are randomly generated, this new test set differs from the one used in the first experiment (upon which θ and p_{\max} were chosen). We then solve this test set with the algorithms under analysis and compare solution time and quality using the three criteria introduced in the first experiment ('min_count', 'rel_err_av' and 't_av'). The results of this experiment can be found in Table 5.2. Starting the discussion with the $S = 3$ -block of the data (the first 3 rows of Table 5.2),

S		t_av (s)	min_count	rel_err_av
3	Penalty	88	12	33%
	IPA	918	20	0
	cplexmiqp	1885	20	0
6	Penalty	125	5	41%
	IPA	1112	13	6%
	cplexmiqp	3486	18	13%
10	Penalty	152	2	52%
	IPA	1149	14	37%
	cplexmiqp	TL	9	5035%
15	Penalty	184	1	33%
	IPA	1343	16	11%
	cplexmiqp	TL	5	4311%
20	Penalty	202	1	56%
	IPA	1239	15	12%
	cplexmiqp	TL	5	7582%

Table 5.2: Results of the third experiment. Comparison of the Penalty algorithm, the IPA and `cplexmiqp` for different values of S .

we see that `cplexmiqp` as well as the IPA always find the global minimum, where the IPA is about 50% faster. The simple penalty algorithm in Algorithm 4.1, while being very fast, only finds the global minimum in 12 cases with an average relative error of 33% in the remaining 8 cases. Increasing S (and thus the combinatorial complexity of the problem), we observe that `cplexmiqp` (especially for $S \geq 10$) fails to find the global minimum in the given time. The IPA on the other hand then starts to be the most competitive algorithm in the 'min_count'-sense, i.e., producing the smallest objective function values compared to the other algorithms (do also note the small relative average error of the IPA). We can also observe, that the computational time of the IPA only slightly increases with S while `cplexmiqp` at some point always hits the prescribed time limit. Furthermore, we report that for $S = 10, 15$, and 20 there was always one problem instance where `cplexmiqp` only returned the zero solution (which is feasible but does not make sense from the application point of view). As a result the average relative error is significantly large. Combining the results of this experiment with the results from the second experiment, we can conclude that the IPA can solve large scale problems, and can, at the same time, compete with `cplexmiqp` in smaller problem instances. The simple penalty approach is very fast but, as we can see in this experiment, fails to produce solutions of high quality in a reliable fashion.

Fourth experiment. In this last experiment, we want to investigate the robustness of the IPA with respect to its stochastic component, i.e., the random choices made in Algorithm 3.2.b, in the chosen setting. To this end, we generate a random problem instance for each value of $S \in \{3, 6, 10, 15, 20\}$ and solve that instance 20 times with the IPA. We then report for each instance a box-plot of the objective function values and a box-plot of the solution times obtained in the 20 runs of the instance with the IPA. The results can be seen in Figure 5.3, where the first row contains the box-plots of the function values and the second row contains the box-plots of the solution times. A box-plot (as depicted in Figure 5.3) consists of several parts: the lower and upper ends of the box represent the 25th and the 75th percentile of the data vector, the

red line inside the box depicts the median of the data and the black dashed lines extending the box (especially seen in the second row of Figure 5.3) are the so called *whiskers* which represent the remaining data points that are not considered outliers (which are depicted as red crosses).

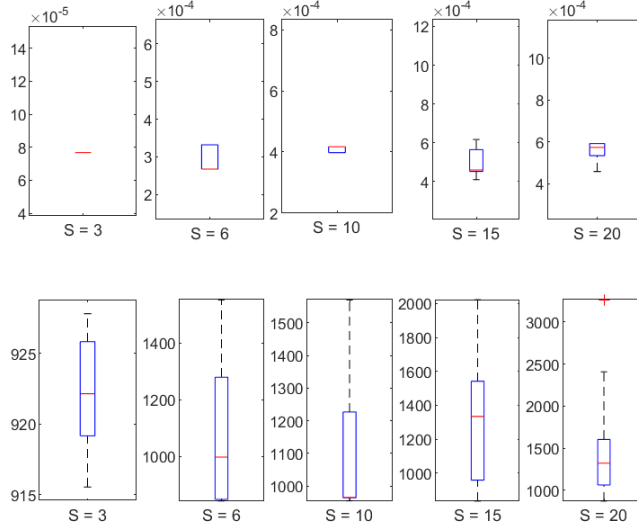


Fig. 5.3: Results of the fourth experiment. Box-plots for the objective function values in the first row and box-plots for the solution times in seconds in the second row.

We begin with the discussion related to the box-plots in the first row of Figure 5.3, i.e., the results with respect to solution quality. In the first plot, we can see that there is no variation in the function values obtained by the IPA. Thus, the median contains all data points such that the IPA is robust in this case. For the remaining box-plots, we see actual boxes, sometimes whiskers and also some outliers. Nevertheless, by having a look at the scale in the y-axis of these plots we can safely say that, while different minimizers are found by the IPA for the same problem instance, all of them still have very high quality and should be satisfactory solutions for the respective problem instance. Investigating the second row, i.e., the box-plots for the solution times, we can observe very little variance in the solution time for $S = 3$. Again, as S and thus the complexity of the problem increases, we can observe more variance in the solution times but it still looks very much acceptable.

5.2. Convection-Diffusion model problem. We now consider the original optimal control problem, but governed by the convection-diffusion PDE

$$(5.1) \quad -\Delta y(x) + w(x) \cdot \nabla y(x) = \sum_{i=1}^l u_i \chi_i(x), \quad x \in \Omega,$$

with the wind vector $w(x) = (2x_2(1 - x_1^2), -2x_1(1 - x_2^2))^T$ and piece-wise constant source functions $\chi_1, \dots, \chi_l \in L^2(\Omega)$, that are constant on the subdomains $\Omega_1, \dots, \Omega_l \subset \Omega$ forming a uniform decomposition of $\Omega = [0, 1]^2$ into l many squares. Here, we use Q1

finite elements, while also employing the Steamline Upwind Petrov-Galerkin (SUPG) [44] upwinding scheme as implemented in the IFISS software package [45] to discretize (5.1) and build the relevant finite element matrices.

For the resulting discretized optimal control problem, we repeat the third experiment from the previous section, where all settings and parameters are chosen as before. We chose not to include the other experiments to keep the length of this presentation healthy but can report that results similar to the Poisson problem are obtained. The result of the third experiment for this convection-diffusion problem can be seen in Table 5.3.

S		t_av (s)	min_count	rel_err_av
3	Penalty	103	16	39%
	IPA	943	19	16%
	cplexmiqp	1937	20	0
6	Penalty	148	7	27%
	IPA	1008	16	19%
	cplexmiqp	TL	16	10%
10	Penalty	198	4	44%
	IPA	1083	15	12%
	cplexmiqp	TL	7	24%
15	Penalty	231	0	50%
	IPA	1223	15	16%
	cplexmiqp	TL	5	52%
20	Penalty	247	0	38%
	IPA	1337	14	12%
	cplexmiqp	TL	6	24%

Table 5.3: Results for the convection-diffusion problem: comparison of the penalty algorithm, the IPA and cplexmiqp for different values of S .

Investigating Table 5.3, we observe that cplexmiqp shows basically the same behaviour as in the Poisson problem: it is always able to solve the problem in the given time for $S = 3$, but then requires much more time and starts to produce unsatisfactory solutions for larger values of S . The IPA again succeeds in finding either the global minimum or a reasonable solution in around 15 – 20 minutes. The simple penalty approach is again very fast, but also quite unreliable in terms of solution quality.

5.3. Analysis of the local solver. As already mentioned, one of the main benefits of our IPA is the possibility to exploit the problem features through the use of a cleverly implemented local solver in line 3 of Algorithm 3.2.a. We now briefly report on the numerical behaviour of our implementation of the IPM described in Section 4.2. Thus, we create an exemplary problem instance (both for the Poisson and convection-diffusion problem) for $S = 10$, and vary the step size of the FEM grid as $h \in \{2^{-5}, 2^{-6}\}$. The instance is then solved for each step size with the IPA, where the settings for the IPM and the IPA are as before. Figure 5.4 shows the number of nonlinear (outer) iterations (NLI) required by the IPM and the average number of preconditioned GMRES iterations (aGMRES) for each value of ε visited during the IPA. Clearly, multiple values reported for a single value of ε correspond to active perturbation cycles of Algorithm 3.2.a.

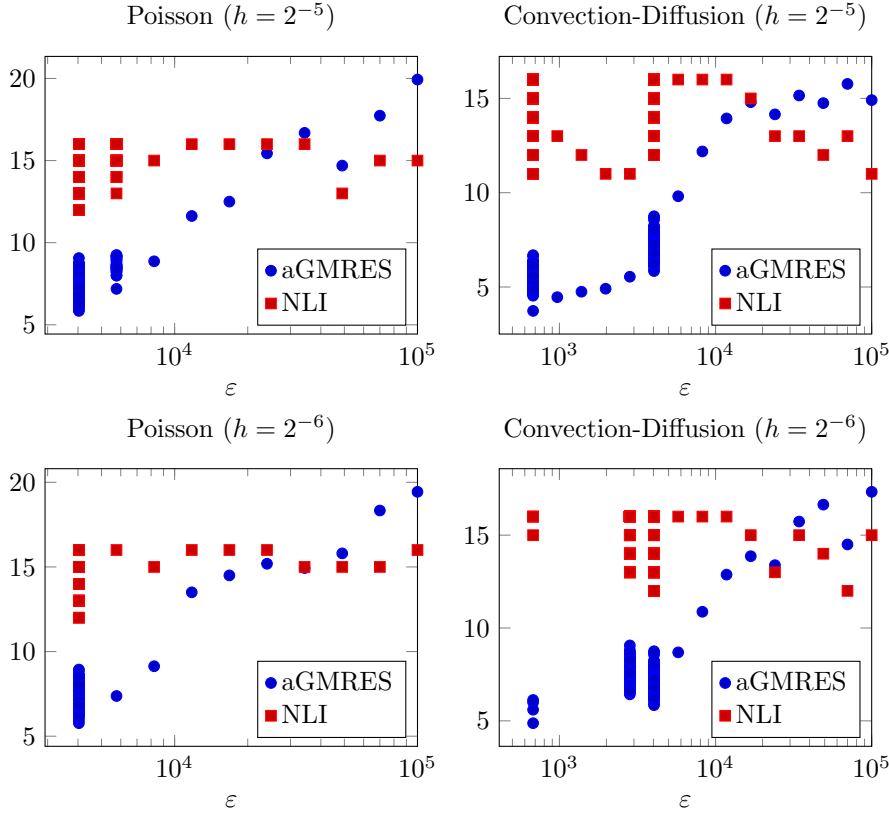


Fig. 5.4: Number of IPM iterations and average GMRES iterations during the IPA steps for Poisson and convection-diffusion problems for varying grid sizes over the penalty parameter ε .

Firstly, we observe that both values of NLI and aGMRES are higher at the beginning of the IPA process, that is for larger values of ε . On the other hand, when ε gets smaller and more perturbation cycles are expected, the number of IPM iterations may get lower and, mostly, the average number of GMRES iterations is reduced. This shows that the IPA together with the IPM efficiently drives the solution of problem (Ppen) to the mixed-integer solution of the original problem. This behaviour is observed in Figure 5.4 for both problems and the varying mesh sizes.

Secondly, the reported number of average number of GMRES iterations is pretty low and does not depend on the mesh size. This reveals the effectiveness of the proposed preconditioner also in combination with the inexact approach. Remarkably, values of aGMRES are extremely low in the last IPA iterations when ε is small.

6. Conclusion & Outlook. A standard MIPDECO problem with a linear PDE constraint and a modelled control was presented and discretized. A novel improved penalty algorithm (IPA) was developed, that combines well-known exact penalty approaches with a basin hopping strategy and an updating tool for the penalty parameter. As a result, only a local optimization solver is required and an interior point method (IPM) that is suited for the problem in question was presented. The linear

algebra phase of the IPM was handled by a Krylov space method together with an efficient preconditioner. Via this, the IPA was shown to work very well in numerical applications for a Poisson as well as a convection-diffusion problem when compared to a simple penalty approach and `cplexmiqp`, the branch-and-bound routine of CPLEX. As an outlook, the authors want to mention that the IPA has already been successfully applied to the presented optimal control problem, but governed by the nonlinear PDE

$$-\Delta y(x) + y(x)^2 = \sum_{i=1}^l u_i \phi_i(x), \quad x \in \Omega,$$

where again the Gaussian source functions defined in (2.2) are used and the IPM has been adapted as described in Remark 4.2. As first results, Figure 6.1 shows the desired state of a random problem instance for $S = 10$, as well as the optimal state found by the IPA. Furthermore, Figure 6.2 shows the result of the experiment from the previous Section 5.3 conducted for this problem instance.

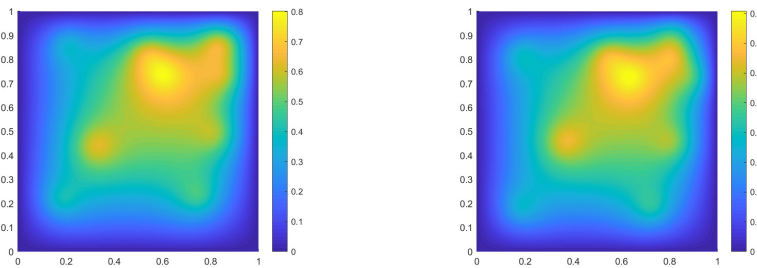


Fig. 6.1: Desired state (left) and optimal state found by the IPA (right) for a problem instance for $S = 10$ of the nonlinear problem.

Overall, these results are already very encouraging and in future work, a comparison of the IPA with state of the art solvers for such nonlinear problems should be carried out (do note that CPLEX cannot deal with nonlinear PDE constraints). Furthermore, future work shall contain the application to MIPDECO problems that are governed by time-dependent PDEs as outlined in Remark 2.3 as these result in a truly large-scale context.

Acknowledgement. D. Garmatter and M. Stoll acknowledge the financial support by the Federal Ministry of Education and Research of Germany (support code 05M18OCB). D. Garmatter, M. Porcelli, and M. Stoll were partially supported by the DAAD-MIUR Joint Mobility Program 2018-2020 (Grant 57396654). The work of M. Porcelli was also partially supported by the National Group of Computing Science (GNCS-INDAM).

References.

- [1] F. Tröltzsch, *Optimal control of partial differential equations: Theory, methods, and applications*. American Mathematical Soc., 2010, vol. 112.
- [2] M. Hahn, S. Leyffer, and V. M. Zavala, *Mixed-Integer PDE-Constrained Optimal Control of Gas Networks*, Argonne National Laboratory, MCS Division Preprint ANL/MCS-P9040-0218, Feb. 2017.

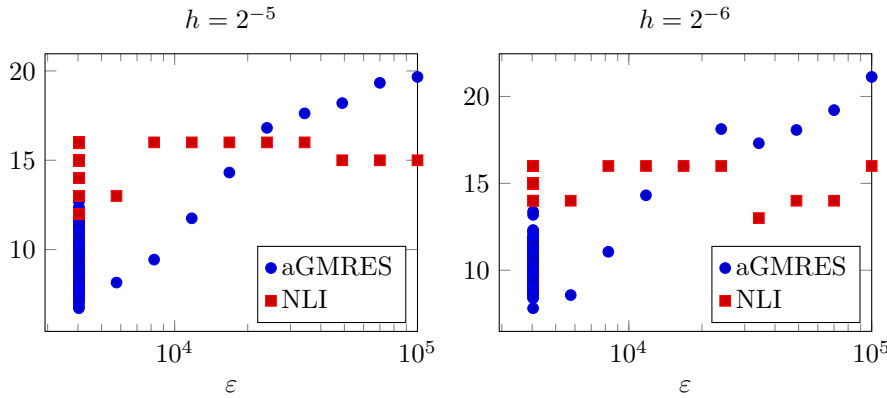


Fig. 6.2: Number of IPM iterations and average GMRES iterations during the IPA steps for the nonlinear problem instance for varying grid sizes over the penalty parameter ε .

- [3] M. E. Pfetsch, A. Fügenschuh, B. Geißler, N. Geißler, R. Gollmer, B. Hiller, J. Humpola, T. Koch, T. Lehmann, A. Martin, *et al.*, “Validation of nominations in gas network optimization: Models, methods, and solutions,” *Optimization Methods and Software*, vol. 30, no. 1, pp. 15–53, 2015.
- [4] S. Funke, P. Farrell, and M. Piggott, “Tidal turbine array optimisation using the adjoint approach,” *Renewable Energy*, vol. 63, pp. 658–673, 2014.
- [5] P. Y. Zhang, D. A. Romero, J. C. Beck, and C. H. Amon, “Solving wind farm layout optimization with mixed integer programs and constraint programs,” *EURO Journal on Computational Optimization*, vol. 2, no. 3, pp. 195–219, Aug. 2014.
- [6] C. Wesselhoeft, “Mixed-Integer PDE-Constrained Optimization,” PhD thesis, Imperial College London, 2017.
- [7] S. Göttlich, A. Potschka, and C. Teuber, “A partial outer convexification approach to control transmission lines,” *Computational Optimization and Applications*, vol. 72, no. 2, pp. 431–456, Mar. 2019, ISSN: 1573-2894.
- [8] P. Manns and C. Kirches, “Multi-dimensional Sum-Up Rounding for Elliptic Control Systems,” *DFG Preprint SPP1962-080*, 2018.
- [9] S. Leyffer, P. Manns, and M. Winckler, *Convergence of Sum-Up Rounding Schemes for the Electromagnetic Cloak Problem*, Argonne National Laboratory, MCS Division Preprint ANL/MCS-P9268-1219, Dec. 2019.
- [10] J. Larson, S. Leyffer, P. Palkar, and S. M. Wild, “A method for convex black-box integer global optimization,” *ArXiv preprint arXiv:1903.11366*, 2019.
- [11] M. Sharma, M. Hahn, S. Leyffer, L. Ruthotto, and B. van Bloemen Waanders, *Inversion of Convection-Diffusion Equation with Discrete Sources*, Argonne National Laboratory, MCS Division Preprint ANL/MCS-P9270-1219, Dec. 2019.
- [12] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan, “Mixed-integer nonlinear optimization,” *Acta Numerica*, vol. 22, pp. 1–131, 2013.
- [13] M. De Santis and F. Rinaldi, “Continuous reformulations for zero–one programming problems,” *Journal of Optimization Theory and Applications*, vol. 153, no. 1, pp. 75–84, 2012.

- [14] F. Giannessi and F. Niccolucci, “Connections between nonlinear and integer programming problems,” in *Symposia Mathematica*, Academic Press New York, vol. 19, 1976, pp. 161–176.
- [15] S. Lucidi and F. Rinaldi, “Exact penalty functions for nonlinear integer programming problems,” *Journal of optimization theory and applications*, vol. 145, no. 3, pp. 479–488, 2010.
- [16] F. Rinaldi, “New results on the equivalence between zero-one programming and continuous concave programming,” *Optimization Letters*, vol. 3, no. 3, pp. 377–386, 2009.
- [17] W. X. Zhu, “Penalty Parameter for Linearly Constrained 0–1 Quadratic Programming,” *Journal of Optimization Theory and Applications*, vol. 116, no. 1, pp. 229–239, 2003.
- [18] M. F. P. Costa, A. M. A. C. Rocha, R. B. Francisco, and E. M. G. P. Fernandes, “Firefly penalty-based algorithm for bound constrained mixed-integer nonlinear programming,” *Optimization*, vol. 65, no. 5, pp. 1085–1104, 2016.
- [19] S. Lucidi and F. Rinaldi, “An exact penalty global optimization approach for mixed-integer programming problems,” *Optimization Letters*, vol. 7, no. 2, pp. 297–307, 2013.
- [20] W. Murray and K.-M. Ng, “An algorithm for nonlinear optimization problems with binary variables,” *Computational Optimization and Applications*, vol. 47, no. 2, pp. 257–288, 2008.
- [21] G. Di Pillo, S. Lucidi, and F. Rinaldi, “An approach to constrained global optimization based on exact penalty functions,” *Journal of Global Optimization*, vol. 54, no. 2, pp. 251–260, 2012.
- [22] G. D. Pillo, S. Lucidi, and F. Rinaldi, “A Derivative-Free Algorithm for Constrained Global Optimization Based on Exact Penalty Functions,” *Journal of Optimization Theory and Applications*, vol. 164, no. 3, pp. 862–882, 2013.
- [23] A. Grosso, M. Locatelli, and F. Schoen, “A population-based approach for hard global optimization problems based on dissimilarity measures,” *Math. Program.*, vol. 110, no. 2, pp. 373–404, 2007.
- [24] R. H. Leary, “Global optimization on funneling landscapes,” *J. Global Optim.*, vol. 18, no. 4, pp. 367–383, 2000.
- [25] *IBM ILOG CPLEX*, <https://www.ibm.com/analytics/cplex-optimizer>.
- [26] S. R. Fipke and A. O. Celli, “The Use of Multilateral Well Designs for Improved Recovery in Heavy-Oil Reservoirs,” in *IADC/SPE Drilling Conference*, Society of Petroleum Engineers, 2008.
- [27] U. Ozdogan and R. N. Horne, “Optimization of well placement under time-dependent uncertainty,” *SPE Reservoir Evaluation & Engineering*, vol. 9, no. 02, pp. 135–145, 2006.
- [28] S. Leyffer, *Optimization: Applications, Algorithms and Computations. 24 lectures on Nonlinear optimization and Beyond*, 2016.
- [29] C. Buchheim, R. Kuhlmann, and C. Meyer, “Combinatorial optimal control of semilinear elliptic PDEs,” *Computational Optimization and Applications*, vol. 70, no. 3, pp. 641–675, 2018.
- [30] F. Giannessi and F. Tardella, “Connections between nonlinear programming and discrete optimization,” in *Handbook of Combinatorial Optimization*, Springer US, 1998, pp. 149–188.
- [31] M. Locatelli and F. Schoen, *Global optimization: Theory, algorithms, and applications*. Siam, 2013, vol. 15.

- [32] J. Nocedal and S. J. Wright, Eds., *Numerical Optimization*. Springer-Verlag, 1999.
- [33] J. Gondzio, “Interior point methods 25 years later,” *European Journal of Operational Research*, vol. 218, no. 3, pp. 587–601, 2012.
- [34] S. Bellavia, “Inexact interior-point method,” *Journal of Optimization Theory and Applications*, vol. 96, no. 1, pp. 109–121, 1998.
- [35] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite elements and fast iterative solvers: With applications in incompressible fluid dynamics*, ser. Numerical Mathematics and Scientific Computation. New York: Oxford University Press, 2005.
- [36] M. Benzi, G. H. Golub, and J. Liesen, “Numerical solution of saddle point problems,” *Acta Numerica*, vol. 14, pp. 1–137, 2005.
- [37] C. C. Paige and M. A. Saunders, “Solutions of sparse indefinite systems of linear equations,” *SIAM J. Numer. Anal.*, vol. 12, no. 4, pp. 617–629, 1975.
- [38] Y. Saad and M. H. Schultz, “GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems,” *SIAM J. Sci. Statist. Comput.*, vol. 7, no. 3, pp. 856–869, 1986.
- [39] M. Stoll and A. Wathen, “Combination Preconditioning and the Bramble–Pasciak⁺ Preconditioner,” *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 2, pp. 582–608, 2008.
- [40] H. S. Dollar, N. I. M. Gould, M. Stoll, and A. Wathen, “Preconditioning saddle point problems with applications in optimization,” *SIAM J. Sci. Computing*, vol. 32, pp. 249–270, 2010.
- [41] J. W. Pearson, M. Porcelli, and M. Stoll, “Interior-point methods and preconditioning for PDE-constrained optimization problems involving sparsity terms,” *Numerical Linear Algebra with Applications*, vol. 27, no. 2, 2020.
- [42] J. W. Pearson and J. Gondzio, “On Block Triangular Preconditioners for the Interior Point Solution of PDE-Constrained Optimization,” *Domain Decomposition Methods in Science and Engineering XXIV*, vol. 125, p. 503, 2019.
- [43] L. Bergamaschi, J. Gondzio, and G. Zilli, “Preconditioning indefinite systems in interior point methods for optimization,” *Computational Optimization and Applications*, vol. 28, no. 2, pp. 149–171, 2004.
- [44] A. N. Brooks and T. J. Hughes, “Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations,” *Computer methods in applied mechanics and engineering*, vol. 32, no. 1-3, pp. 199–259, 1982.
- [45] H. C. Elman, A. Ramage, and D. J. Silvester, “Algorithm 866: IFISS, a Matlab toolbox for modelling incompressible flow,” *ACM Transactions on Mathematical Software (TOMS)*, vol. 33, no. 2, 14–es, 2007.

Appendix. Figure 6.3 contains the detailed representation of the data from the first experiment conducted in Section 5, where the color of the plot indicates the results for the different values of S , i.e., $S = 3$ (blue circles), $S = 6$ (green crosses), $S = 10$ (red crosses), $S = 15$ (turquoise stars) and $S = 20$ (pink boxes), and the x-axis encodes the different variations of the IPA. Furthermore, Figure 6.4 contains a zoomed in version of the data from the same experiment for $S = 6$.

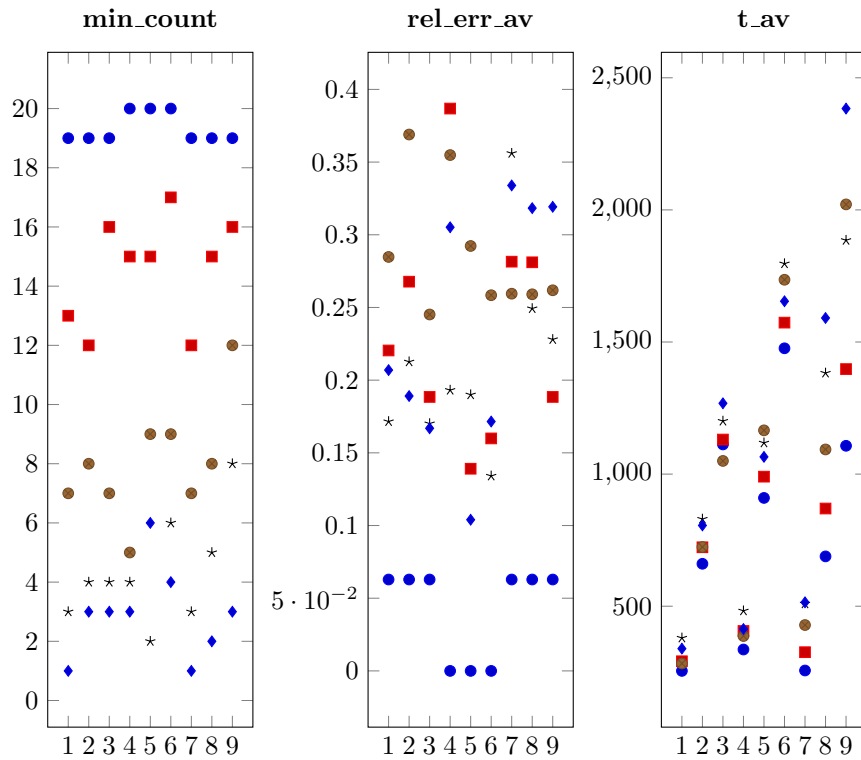


Fig. 6.3: Results of the first experiment. Data for $S = 3$ (blue dots), $S = 6$ (red boxes), $S = 10$ (brown dots), $S = 15$ (black stars) and $S = 20$ (blue diamonds) is plotted over the nine different variations.

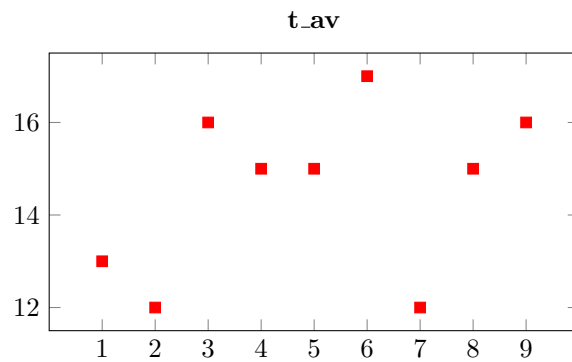


Fig. 6.4: Results of the first experiment: only the 'min_count' data for $S = 6$ is plotted over the nine different variations.