Solving Mixed-Integer Partial Differential Equation Constrained Optimization

Otto von Guericke University Magdeburg
Faculty of Mathematics

Master's Thesis

submitted by

Anna Thünen

born May 28, 1992 in Aschersleben,
in partial fulfillment of the requirements for the degree
Master of Science
as part of the degree program in Mathematics

April 18, 2017

This thesis was supervised at the Instiute of Mathematical Optimization by
Prof. Dr. rer. nat. habil. Sebastian Sager
and at Argonne National Laboratory by
Sven Leyffer, Ph.D.

Contents

Introduction

Optimization is an important tool in today's world. There are optimization problems in almost every discipline of industry and research. For instance, economics and management sciences usually optimizes profit or costs, in engineering, we may optimize performance or the design of facilities. Alternatively as a combination of these scopes, logistics is a frequent application of optimization. The common goal is to make decisions. Therefore we either want to maximize or to minimize a certain quantity which we describe by an objective function, depending on the application it might be e.g. cost, time, energy, etc. Usually, the decision variables are constrained and we call the set of possible decisions feasible set. Thus, the goal of optimization is to find the feasible decisions which give the best objective value.

As the range of applications is very large, the number of mathematical subfields is also significant. The new class of mixed-integer partial differential equation constrained optimization (MIPDECO) has been established recently [15]. It combines two key classes of optimization, namely mixed-integer nonlinear optimization (MINLP) and partial differential equation (PDE) constrained optimization. In MINLP the feasible set and the objective function are quantified by nonlinear functions. In addition to continuous decision variables, we have to deal with integer variables leading to combinatorial difficulties. On the other hand, many applications in optimization involve complex systems which might be modeled by partial differential equations. PDE constrained optimization poses different challenges since discretizations lead to a large number of variables and numerical complexity, so both problem classes are studded with big challenges by themselves. Hence we need to develop new approaches to overcome the issues both of integer variables and the number of variables. This is of high importance since MIPDECO has a broad range of applications, in particular: topology optimization such as in [10] and technology of both renewable and conventional energies, e.g. mentioned by Leyffer et al. [15].

In this thesis we present a preprocessing algorithm and rounding based heuristics for a family of MIPDECOs governed by the heat equation. With the developed techniques we are able to solve discretized formulations which are originally not solvable with conventional MINLP solvers within a day. The developed preprocessing scheme accelerates the solving but it is still very time and memory consuming. However, with the rounding approaches we are able to compute good solutions fast. As we present multiple heuristics, we can apply them in parallel and choose the best solution among them. Furthermore, the preprocessing can be used to accelerate the rounding as well. With these rounded solutions we able to hot start MINLP solvers. Solutions obtained in this manner are computed in reasonable time and optimal or arbitrarily close to optimality.

This thesis is structured as follows: first, we give an overview of the mathematical background of the thesis. Then, we introduce the two problems in Chapter 2. As we follow a "first discretize, then optimize" approach, we then present a discretization of the problem. Due to the large number of variables of these discretized formulations, we reduce the computational time by introducing powerful preprocessing techniques in

Chapter 3. In Chapter 4, we present three rounding approaches which deliver feasible solutions. Furthermore, we give an example how to incorporate spatial information prior to rounding. Finally, we discuss the numerical performance of the presented algorithms in Chapter 5. Thereby, we compare computational effort of the algorithms as well as the quality of rounded solutions.

Chapter 1

Mathematical Background

In this chapter, we briefly summarize the mathematical background which is required for the following chapters of this thesis. As the problem class of mixed-integer partial differential equation constrained optimization (MIPDECO) involves aspects of mixed-integer nonlinear programming (MINLP) as well as partial differential equations (PDE), we revise both fields.

Therefore we begin with a general formulation of a MIPDECO. Then we state a subclass which is governed by the heat equation on which we focus in this thesis. As we follow the "first discretize then optimize" approach throughout the thesis, we explain the methods to discretize occurring function norms, the heat equation, and further conditions and constraints. Since the discretization of a MIPDECO is a MINLP, we define the class and explain a broadly used algorithm to solve this problem class. The algorithm presented for general MINLP and the heuristics, which we introduce in Chapter 4, involve the solving of nonlinear programs (NLP). Therefore, we define this problem class and explain an algorithm to solve it.

At the end of this chapter, we introduce a concept of functional analysis, namely convolution, which we apply in Chapter 3 to speed up the solving of our instances.

1.1 Mixed-Integer PDE Constrained Optimization

First, we formulate a generic MIPDECO as a minimization of an objective functional $\mathcal{J} : \mathcal{U}_{ad} \times \mathcal{V}_{ad} \times \mathcal{W}_{ad} \to \mathbb{R}$, where $\mathcal{U}_{ad}$ is the function space of the admissible states, $\mathcal{V}_{ad}$ is the function space of the admissible continuous controls, and $\mathcal{W}_{ad}$ is the function space of the admissible binary controls. It is constrained by a PDE with suitable conditions and other constraints which are represented as the functional $\mathcal{C} : \Omega \times T \times \mathcal{U}_{ad} \times \mathcal{V}_{ad} \times \mathcal{W}_{ad} \to \mathbb{R}^n$, where Ω denotes the spatial domain and T the time horizon. With that, we can write the MIPDECO as follows:

$$\underset{u,v,w}{\text{minimize}} \quad \mathcal{J}(u, v, w) \tag{1.1a}$$

$$\text{subject to} \quad \mathcal{C}(x, t; u, v, w) = 0 \tag{1.1b}$$

$$u \in \mathcal{U}_{ad}, v \in \mathcal{V}_{ad}, w \in \mathcal{W}_{ad} \tag{1.1c}$$

We assume that the state and the continuous controls are integrable and we define the space of square integrable functions, as in [22, p. 19].

Definition 1 *Let E be an open subset of $\mathbb{R}^n$, then $L^2(E)$ denotes the space of all measurable functions $h : E \to \mathbb{R}$ for which it holds that:*

$$\int_E |h(x)|^2 \, \mathrm{d}x < \infty$$

The space is equipped with the norm:

$$\|h\|_{2,E} = \left(\int_E |h(x)|^2 \, \mathrm{d}x \right)^{\frac{1}{2}}$$

Thus, we require that $\mathcal{V}_{ad} \subseteq L^2(\Omega \times T)$. In order to guarantee existence of the derivatives of the state, we have to assume the stronger condition that u is continuous differentiable. Therefore we define the space of k-times continuous differentiable functions as in [20, p. 147]:

Definition 2 *Let E be an open subset of $\mathbb{R}^n$ and let $h: E \to R^m$ be a function. We say that h is continuously differentiable if the partial derivatives are continuous on E. The space of these functions is denoted by $C^1(E)$.*

We say that h is twice continuously differentiable if it is continuously differentiable, and the partial derivatives are themselves continuously differentiable. The space of these functions is denoted by $C^2(E)$. In this manner, one can also define C^3, C^4, and C^k.

With this notation, we state that the $\mathcal{U}_{ad} \subseteq C^k(\Omega \times T)$, where k is the highest order of the derivatives occurring in the PDE.

Furthermore, we define the space of the admissible binary controls $\mathcal{W}_{ad} = \{h : E \to \{0,1\}^m \big| E \subseteq \mathbb{R}^n\}$. In general the functions in W_{ad} are neither integrable, continuous, nor differentiable.

The generic MIPDECO (1.1) is constrained by the functional $\mathcal{C}$ which can contain a vast variety of constraint types. In order to illustrate this, we introduce a subclass of MIPDECO which is governed by the heat equation.

We assume the time horizon $T = [0, T_f]$ and a two dimensional domain Ω with the boundary $\partial\Omega = \mathcal{D} \cup \mathcal{N} \cup \mathcal{R}$ and coordinates (x, y).

$$\underset{u,v,w}{\text{minimize}} \quad \mathcal{J}(u,v,w) \qquad \text{objective functional} \tag{1.2a}$$

$$\text{subject to} \quad \frac{\partial u}{\partial t}(x,y,t) - \kappa \Delta u(x,y,t) = \mathcal{F}(x,y,t,v,w) \quad \text{in} \quad \Omega \times T \qquad \text{heat equation} \tag{1.2b}$$

$$u(x,y,t) = g_D(t) \quad \text{on } \mathcal{D} \times T \qquad \text{Dirichlet boundary} \tag{1.2c}$$

$$\frac{\partial u}{\partial n}(x,y,t) = g_N(t) \quad \text{on } \mathcal{N} \times T \qquad \text{Neuman boundary} \tag{1.2d}$$

$$\frac{\partial u}{\partial n}(x,y,t) = g_R(u(x,y,t),t) \quad \text{on } \mathcal{R} \times T \qquad \text{Robin boundary} \tag{1.2e}$$

$$u(x,y,0) = g_0(x,y) \quad \forall (x,y) \in \Omega \qquad \text{initial condition} \tag{1.2f}$$

$$\mathcal{C}(x,t;u,v,w) = 0 \qquad \text{further constraints} \tag{1.2g}$$

$$(v,w) \in \mathcal{V}_{ad} \times \mathcal{W}_{ad} \qquad \text{admissible controls} \tag{1.2h}$$

$$u \in \mathcal{U}_{ad} \qquad \text{admissible states} \tag{1.2i}$$

We assume that the boundary and initial conditions are consistent. The variables in this problem are the states $u = u(x,y,t) \in C^2(\Omega \times T)$, a finite set of time dependent continuous controls $v = v_{1,\dots,L}(t) \in C^2(T)$, and a finite set of time dependent binary controls $w = w_{1,\dots,L}(t) \in \{w : T \to \{0,1\}^L\}$. The parameter κ describes the thermal dissipativity of the domain Ω, and can be either constant $\kappa \in \mathbb{R}_+$ or vary in space $\kappa = \kappa(x,y)$. The right-hand-side of the PDE, $\mathcal{F}(x,y,t,v,w)$ depends on space, time, and the controls. The objective functional $\mathcal{J}(u,v,w)$ depends on the states and the controls.

With this generic formulation we can describe a large number of control problems. In this thesis we study two problems which are governed by the heat equation with Dirichlet boundary and initial conditions.

1.2 Discretization

In order to compute approximate solutions of a MIPDECO like (1.2), we introduce standard discretization methods in this section.

The objective functional $\mathcal{J}(u,v,w)$ of (1.2) usually involves the norm of states and controls. The function norm introduced in Definition 1 contains integration of a function. Therefore, we introduce the numerical integration scheme trapezoidal rule like in [8, p. 165]:

Definition 3 *Let $h : [0,X] \subset \mathbb{R} \to \mathbb{R}$ be a smooth and integrable function and $0 = x_0 < \dots < x_N = X$ be a set of equidistant grid points with step size h_x. The approximation*

scheme

$$\int_{x_k}^{x_{k+1}} h(x)\,\mathrm{d}x \approx \frac{h_x}{2}\left(h(x_k) + h(x_{k+1})\right)$$

is called elementary trapezoidal rule. Summing over all subintervals gives the composite trapezoidal rule:

$$\int_0^X h(x)\,\mathrm{d}x \approx h_x\left(\frac{1}{2}h(x_0) + h(x_1) + \cdots + h(x_{N-1}) + \frac{1}{2}h(x_N)\right)$$

This definition considers a one dimensional domain, in case of higher dimensional domains, we apply the trapezoidal rule successively in every dimension.

The PDE with its boundary conditions is discretized using the finite difference method. For details and theory of this method, we refer to [21]. The heat equation is a PDE involving second order derivatives in space and the first order derivative in time. We begin to discretize the PDE in space, resulting a set of time dependent ordinary differential equations (ODEs), which are then discretized in time. This method is known as method of lines, for details we refer to [9]. The Laplace operator Δ is discretized by a two dimensional five-point stencil which we define as in [1, Table 25.3.30]:

Definition 4 *Let E be an open subset of $\mathbb{R}^2$ and $h : E \to \mathbb{R}$ be a twice continuously differentiable function. Then the finite difference approximation of the Laplacian of h with the step sizes h_x and h_y:*

$$\Delta h(x,y) \approx \frac{h(x-h_x,y) + h(x+h_x,y) + h(x,y-h_y) + h(x,y+h_y) - 4h(x,y)}{h_x h_y}$$

is called five-point stencil.

The time derivative $\frac{\partial}{\partial t}$ is discretized by the backward Euler method which we define as in [1, Table 25.3.30]:

Definition 5 *Let $h : [0,T_f] \subset \mathbb{R} \to \mathbb{R}$ be a continuously differentiable function. Then the finite difference approximation of its first derivative with the step size h_t:*

$$\frac{\partial h}{\partial t}(t+h_t) \approx \frac{h(t+h_t) - h(t)}{h_t}$$

is called backward Euler method (or implicit Euler method).

Besides backward Euler method we implemented forward Euler method and Crank-Nicholson method (also known as trapezoidal rule). However, we do not want to discuss the influence of the choice of discretization in this thesis and we refer for details of these methods to [8, ch. 5] or [4].

With this discretization tools, we discretize the PDE by first discretizing the domain and the variables and then we discretize the space and time derivatives.

We assume a rectangle domain, $\Omega = [0, X] \times [0, Y] \subset \mathbb{R}^2$, the time horizon $T = [0, T_f]$, and uniform step-sizes $h_x > 0$, $h_y > 0$, and $h_t > 0$. With this we define an equidistant mesh:

$$\begin{aligned} 0 = x_0 < \cdots < x_N = X \text{ with } h_x \\ 0 = y_0 < \cdots < y_M = Y \text{ with } h_y \\ 0 = t_0 < \cdots < t_{T_n} = T_f \text{ with } h_t \end{aligned}$$

On this mesh, we define approximate values of the states and controls for $i = 0, \dots N$, $j = 0, \dots M$, and $k = 0, \dots T_n$, In these grid points we define the variables as evaluation of the unknown states and controls for $i = 0, \dots N$; $j = 0, \dots M$; $k = 0, \dots T_n$, and $l \in \{1, \dots, L\}$:

$$u_{i,j,k} = u(ih_x, jh_y, kh_t) \qquad v_k = v(kh_t) \qquad w_{k,l} = w_l(kh_t) \qquad v_{k,l} = v_l(kh_t) \tag{1.3}$$

Similarly, we write for the thermal dissipativity $\kappa_{i,j} = \kappa(ih_x, jh_y)$.

With that, we can write the discretized PDE for $i = 1, \dots N-1$, $j = 1, \dots M-1$, $k = 1, \dots T_n$, and $l \in \{1, \dots, L\}$ as follows:

$$\begin{aligned} \frac{1}{h_t}(u_{i,j,k+1} - u_{i,j,k}) - \frac{\kappa_{i,j}}{h_x h_y}(u_{i-1,j,k+1} + u_{i+1,j,k+1} + u_{i,j-1,k+1} + u_{i,j+1,k+1} - 4u_{i,j,k}) \\ = \mathcal{F}(ih_x, jh_x, (k+1)h_t, v_{k+1,l}, w_{k+1,l}) \end{aligned} \tag{1.4}$$

We observe, that the discretized equation is linear in terms of $u_{i,j,k}$.

The Dirichlet boundary is discretized by fixing the state variables $u_{i,j,k} = g_D(kh_t)$ for indices on the boundary of the domain Ω. Boundary conditions of Neumann or Robin type can be discretized similarly to the PDE. However, we do not give details since the problems in this thesis contain Dirichlet boundary only.

1.3 Mixed-Integer Nonlinear Programming

The previous section discussed discretization methods for function norms and PDEs. A MIPDECO, discretized in that manner, is a mixed-integer nonlinear program. In order to discuss an algorithm to solve a problem of this class we introduce a generic definition like in [3, ch. 1]:

Definition 6 *Let $h : \mathbb{R}^n \to \mathbb{R}$ and $c : \mathbb{R}^n \to \mathbb{R}^m$ be twice continuously differentiable functions, $X \subseteq R^n$ is a bounded polyhedral set, and $I \subseteq \{1, \dots, n\}$ is the index set of integer variables. Then we call:*

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & h(x) \\ \text{subject to} \quad & c(x) \leq 0 \\ & x \in X \\ & x_i \in \mathbb{Z}, \forall i \in I \end{aligned}$$

mixed-integer nonlinear program and use MINLP as the abbreviation.

In case of a convex objective h, convex constraints c, and bounded and polyhedral X, we call the MINLP convex. In case of a quadratic objective h and linear constraints c, we call the problem mixed-integer quadratic program and use MIQP as the abbreviation.

We remark that problem class MINLP is NP-hard since its definition includes mixed-integer linear programs, which are NP-hard, proved by Kannan and Monma [14]. Moreover, some nonconvex MINLPs are even undecidable, proved by Jeroslow [13].

Throughout this thesis, we often use the term NLP relaxation of a MINLP. The NLP relaxation is the problem which has the same objective and constraints as the original MINLP or MIQP except for the integer condition.

1.3.1 Branch and Bound

In order to solve the MINLP formulation of a MIPDECO we apply the state-of-the-art solver CPLEX [12]. This complex software includes many different algorithms and heuristics. However its basis is the divide and conquer algorithm branch and bound. This single-tree algorithm was introduced for MINLP by Dakin [5]. We give a brief overview of this method for convex MINLP with suitable constraint qualification like in [3, ch. 3].

The branch and bound algorithm splits the feasible set of a MINLP in a recursive manner and generates a search tree in this way. This splitting is also called branching. We start with solving the NLP relaxation of the MINLP in the root node. On the basis of its solution, we split the feasible set in two (or more) disjunct subsets. In the child nodes, we again solve NLP relaxations in order to divide the feasible set of the subproblems. In order to avoid the complete enumeration of all feasible solutions, we exclude some sub trees by the following pruning rules:

- The subproblem is infeasible.
- The solution of the NLP relaxation of the subproblem is integer feasible. The objective value of current best integer feasible solution is saved as upper bound UB.
- The objective value of the NLP relaxation in a node is worse than the best known integer feasible point, i.e. $> UB$.

In these cases, we do not continue to split from this node, because we cannot find better integer feasible points in this subtree.

The algorithm terminates for a problem when all subproblems are solved and we cannot branch anymore. It returns the integer feasible solution with the best objective value, or it is stated that the problem is infeasible.

In general it is unclear how to branch the problems or in which order we proceed the child nodes in the tree. However there are many powerful heuristics, e.g. depth-first-search or breadth-first search, which can reduce the number of subproblems and therefore increase the speed of the algorithm.

1.4 Nonlinear Programming

In the previous section we introduced the branch and bound algorithm which is broadly used to solve MINLPs. This algorithm requires to solve the NLP relaxation of the problem and of subproblems. Furthermore, these relaxations are crucial for the heuristics presented in this thesis.

The NLP relaxation of a MINLP is a nonlinear program, which can be defined as follows:

Definition 7 *Let $h : \mathbb{R}^n \to \mathbb{R}$ and $c : \mathbb{R}^n \to \mathbb{R}^m$ be twice continuously differentiable functions. Then we call:*

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & h(x) \\ \text{subject to} \quad & c(x) \leq 0 \end{aligned}$$

nonlinear program and use NLP as the abbreviation.

In case of a convex objective h and convex constraints c, we call the NLP convex. In case of a quadratic objective h and linear constraints c, we call the problem quadratic program and use QP as the abbreviation.

1.4.1 Interior Point Method

In order to solve an NLP we apply the interior point solver IPOPT by Wächter and Biegler [24]. We briefly explain the method in this section like in [23], for details and more general methods we recommend [24] and [18].

Geometrically, adding a logarithmic term to the objective prevents iterates to approach the boundary of the feasible set unless the objective is significantly decreased. Therefore it is called interior point method.

We assume that the NLP is constrained by equality constraints only and that the variables are nonnegative:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & h(x) && \text{(1.5a)} \\ \text{subject to} \quad & c(x) = 0 && \text{(1.5b)} \\ & x \geq 0,\ x \in \mathbb{R}^n && \text{(1.5c)} \end{aligned}$$

This assumption does not restrict Definition 7 since we can obtain this formulation by adding slack variables and separate the variables in their nonnegative and negative parts.

The interior point algorithm is a barrier method and we formulate the barrier problem of (1.5) for the barrier parameter $0 < \mu << 1$:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \varphi_\mu(x) = h(x) - \mu \sum_{i=1}^{n} \ln(x_i) && \text{(1.6a)} \\ \text{subject to} \quad & c(x) = 0 && \text{(1.6b)} \\ & x \in \mathbb{R}^n && \text{(1.6c)} \end{aligned}$$

We start with an initial guess and compute the next iterate by solving the barrier problem (1.6). For this we apply a Newton-type method with moderate accuracy. The step length of the obtained Newton steps is adjusted by line search and filter methods in order to achieve a sufficient improvement of the next iterate.

We continue the Newton method until the iterate fulfills the first-order optimality conditions (e.g. [18, p. 321]) of the barrier problem (1.6) up to a moderate tolerance. Then we choose a smaller barrier parameter μ and the barrier problem with the smaller μ is solved. Again, we generate Newton steps for the new barrier problem until optimality conditions are fulfilled to the desired (low) tolerance. Then μ is further decreased.

We solve this sequence of barrier problems with decreasing parameter μ until the iterate fulfills the first-order optimality conditions of the problem (1.5).

Since we do not assume convexity, the obtained solution might not be a minimizer but a local maximizer or a saddle point. To prevent this behavior, IPOPT included globalization strategies, e.g. it regularizes the Hessian.

1.5 Convolution

In Chapter 3, we present a powerful preprocessing tool which is based on a decomposition of the solution of a PDE. Some solution parts are related to each other with respect to time. This relation can be represented as the analytical concept of convolution. We adapt the definitions from [16, p. 435].

Definition 8 *Let $h_1, h_2 : \mathbb{R}^n \to \mathbb{R}$ be $L^2(\mathbb{R})$ functions. Then their convolution is given by:*

$$(h_1 * h_2)(x) = \int_{\mathbb{R}^n} h_1(\tau) h_2(x - \tau) \, \mathrm{d}\tau$$

If the functions are defined on a subset of $\mathbb{R}^n$ the functions will be extended by zero.

Continuous convolution can be transferred to discrete domains, therefore it is applicable for the discretized functions emerging in the MINLP formulation of a MIPDECO.

Definition 9 *Let $h_1, h_2 : \mathbb{Z} \to \mathbb{R}$ be sequences, then their convolution is defined by:*

$$(h_1 * h_2)[n] = \sum_{m=-\infty}^{\infty} h_1[m] h_2[n - m]$$

If the sequences are defined on a subset of $\mathbb{Z}$ the sequences will be extended with zero.

Chapter 2

Control of Heat Equation Problems

We introduce two MIPDECO problems which are governed by the heat equation with suitable initial and boundary conditions. The problems are adapted from [11]. Their goal is to place a small and fixed number of actuators, e.g. one or two, over time in a given domain. So, we need to choose for any time a fixed number of locations to place the actuators. The possible locations are given as a finite set of coordinates in space, hence they do not depend on a discretization, an example of a possible actuator distribution is given in Figure 2.1.

This chapter is structured as follows: first, we give the mathematical formulation of a problem with only binary controls and discretize it. Second, we introduce a problem which includes, besides actuator placement, an intensity control of the actuators. We present the mathematical formulation which is then discretized.

2.1 Heat Equation with Actuator Placement

First, we introduce a simplified formulation of the original problem where we do not include intensity control.

2.1.1 Mathematical Formulation

We consider a rectangle $\Omega = [0, 1] \times [0, 2]$ with the boundary $\partial\Omega$ and the time horizon $T = [0, T_f]$ as the domains. The objective (2.1a) is quadratic, its first term is of tracking type and captures the desired final state u_f and the second term regularizes the state over time. The constraints are a source budget (2.1e), which limits the quantity of placed actuators, and the two-dimensional heat equation (2.1b) with some source term. Additionally, we assume Dirichlet boundary (2.1c) and initial conditions (2.1d). This can be written as

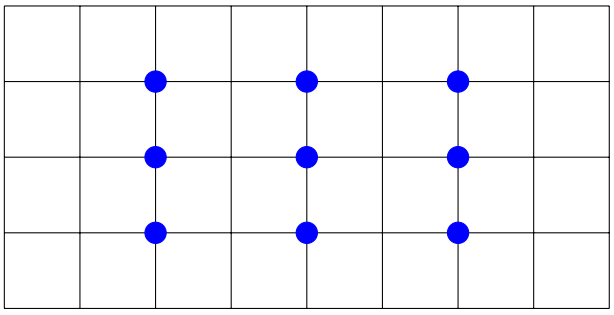

Figure 2.1: Domain Ω with actuator locations

follows:

$$\underset{u,w}{\text{minimize}} \quad J(u) = \|u(\cdot,\cdot,T_f) - u_f(\cdot,\cdot)\|^2_{2,\Omega} + 2\,\|u(\cdot,\cdot,\cdot)\|^2_{2,\Omega\times T} \tag{2.1a}$$

$$\text{subject to} \quad \frac{\partial u}{\partial t}(x,y,t) - \kappa\Delta u(x,y,t) = \sum_{l=1}^{L} w_l(t)\bar{v}f_l(x,y) \qquad \text{in } \Omega\times T \tag{2.1b}$$

$$u(x,y,t) = 0 \qquad \text{on } \partial\Omega\times T \tag{2.1c}$$

$$u(x,y,0) = u_0(x,y) \qquad \text{in } \Omega \tag{2.1d}$$

$$\sum_{l=1}^{L} w_l(t) = \bar{w} \qquad \text{in } T \tag{2.1e}$$

$$w_l(t) \in \{0,1\} \text{ for all } l \in \{1,\dots,L\} \qquad \text{in } T \tag{2.1f}$$

The variables are the state $u \in C^2(T\times\Omega)$ and the binary controls w_l. The nonnegative integer $\bar{w}$ denotes the quantity of actuators and L the quantity of their possible locations. The thermal diffusivity κ can be either constant $\kappa \in \mathbb{R}_+$ or vary in space $\kappa = \kappa(x,y) \in \mathbb{R}_+$ representing a certain material or a distribution of various materials. We define the source term for all locations $l \in \{1,\dots,L\}$ and a fixed parameter $\varepsilon \in \mathbb{R}_+$:

$$f_l(x,y) = \frac{1}{\sqrt{2\pi\varepsilon}} e^{\frac{-((x_l-x)^2+(y_l-y)^2)}{2\varepsilon}} \tag{2.2}$$

where (x_l, y_l) is the coordinate of the mesh point of the lth possible location. The intensity $\bar{v} \in \mathbb{R}$ is fixed and scales the source term.

2.1.2 Discretization

In order to get a MINLP formulation of the problem (2.1) we discretize the components with the techniques introduced in section 1.2.

The function norms occuring in the objective (2.1a) are discretized by the trapezoidal rule, this yields the discrete objective J_1, c.f. (A.1). The heat equation is discretized by using a five-point stencil in space and the implicit Euler in time, like in (1.4). Moreover, we discretize the source term by simply evaluating it at the grid points. The initial and boundary conditions (2.1c-2.1d) are discretized by fixing the corresponding variable values. Because of the discretization of the time horizon, we get for every time step a source budget constraint which can be interpreted for $\bar{w} = 1$ as a special ordered set constraint [2].

With the index sets $\mathcal{I}_1, \ldots, \mathcal{I}_6$, this discretizations yield:

$$
\begin{aligned}
\underset{u,w}{\text{minimize}} \quad & J_1(u) \\
\text{subject to} \quad & \frac{1}{h_t}(u_{i,j,k+1} - u_{i,j,k}) \\
& - \frac{\kappa_{i,j}}{h_x h_y}(u_{i-1,j,k+1} + u_{i+1,j,k+1} + u_{i,j-1,k+1} + u_{i,j+1,k+1} - 4u_{i,j,k+1}) \\
& = \sum_{l=1}^{L}(\bar{v} w_{k+1,l} f_l(ih_x, jh_y)) \quad \forall (i,j,k) \in \mathcal{I}_1 \\
& u_{i,j,k} = 0 \quad \forall (i,j,k) \in \mathcal{I}_2 \\
& u_{i,j,0} = u_0(ih_x, jh_y) \quad \forall (i,j) \in \mathcal{I}_3 \\
& \sum_{l=1}^{L} w_{k,l} = \bar{w} \quad \forall k \in \mathcal{I}_4 \\
& w_{k,l} \in \{0,1\} \quad \forall (k,l) \in \mathcal{I}_5 \\
& u_{i,j,k} \in \mathbb{R} \quad \forall (i,j,k) \in \mathcal{I}_6
\end{aligned}
$$

In order to keep the MINLP readable, we define the index sets separately:

$$
\begin{aligned}
\mathcal{I}_1 &= \{(i,j,k) \big| i = 1, \ldots, N-1; j = 1, \ldots, M-1; k = 0, \ldots, T_n - 1\} \\
\mathcal{I}_2 &= \{(i,j,k) \big| i = 0, N; j = 0, M; k = 0, \ldots, T_n\} \\
\mathcal{I}_3 &= \{(i,j) \big| i = 1, \ldots, N-1; j = 1, \ldots, M-1\} \\
\mathcal{I}_4 &= \{k \big| k = 0, \ldots, T_n\} \\
\mathcal{I}_5 &= \{(i,j,k) \big| k = 0, \ldots, T_n - 1 : l = 1, \ldots, L\} \\
\mathcal{I}_6 &= \{(i,j,k) \big| i = 0, \ldots, N; j = 0, \ldots, M; k = 0, \ldots, T_n\}
\end{aligned} \tag{2.3}
$$

2.2 Heat Equation with Actuator Placement and Operation

In addition to the actuator placement, the original problem in [11] includes the control of the actuator intensity. We introduce the extended problem in this section.

2.2.1 Mathematical Formulation

The rectangle domain $\Omega = [0,1] \times [0,2]$ and the time horizon $T = [0, T_f]$ are defined like in the previous problem. The objective (2.4a) is extended by a third term regularizing the continuous controls. We keep the source budget (2.4f) to limit the actuator quantity, the two-dimensional heat equation (2.4b), Dirichlet boundary (2.4c), and initial conditions

(2.4d) as constraints. This can be written as follows:

$$\underset{u,v,w}{\text{minimize}} \quad J(u,v) = \|u(\cdot,\cdot,T_f) - u_f(\cdot,\cdot)\|^2_{2,\Omega} + 2\,\|u(\cdot,\cdot,\cdot)\|^2_{2,\Omega\times T} + \frac{1}{500}\sum_{l=1}^{L}\|v_l(\cdot)\|^2_{2,T} \tag{2.4a}$$

$$\text{subject to} \quad \frac{\partial u}{\partial t}(x,y,t) - \kappa\Delta u(x,y,t) = \sum_{l=1}^{L} v_l(t) f_l(x,y) \qquad \text{in } \Omega\times T \tag{2.4b}$$

$$u(x,y,t) = 0 \qquad \text{on } \partial\Omega\times T \tag{2.4c}$$

$$u(x,y,0) = u_0(x,y) \qquad \text{in } \Omega \tag{2.4d}$$

$$-M_{\text{big}} w_l(t) \le v_l(t) \le M_{\text{big}} w_l(t) \text{ for all } l\in\{1,\dots,L\} \qquad \text{in } T \tag{2.4e}$$

$$\sum_{l=1}^{L} w_l(t) = \bar{w} \qquad \text{in } T \tag{2.4f}$$

$$w_l(t)\in\{0,1\} \text{ for all } l\in\{1,\dots,L\} \qquad \text{in } T. \tag{2.4g}$$

In addition to state variables $u \in C^2(T\times\Omega)$ and binary controls w_l, we add continuous controls $v_l \in L^2(T)$ which represent the intensity control. As in the first problem, $\bar{w}$ denotes the quantity of actuators and L the number of their possible locations. The same way, the thermal diffusivity $\kappa \in \mathbb{R}_+$ can be constant or vary in space. Furthermore, we keep the definition of the source term $f_l(x,y)$ like in (2.2).

Originally, the problem formulation was non-convex. We overcome this issue by substitution of $v(t)w_l(t)$ by $v_l(t)$ on the right-hand-side of the heat equation (2.4b) and adding the Big M formulation (2.4e).

We remark that if the equal sign in the source budget constraint (2.4f) is replaced by a less or equal it yields an equivalent formulation of the problem.

2.2.2 Discretization

Similarly to the previous problem, we get the MINLP formulation of the problem by discretizating its components. By discretizing (2.4a), we get the discrete objective J_2, c.f. (A.2). The PDE with its initial and boundary conditions (2.4b-2.4d) is discretized using the presented finite difference method. In addition, we have to discretize the big M constraint (2.4e) for this problem. Similar to the source budget constraint, we obtain a big M constraint for every time step and every location.

This yields the following MINLP with the index sets (2.3):

$$
\begin{aligned}
\underset{u,v,w}{\text{minimize}} \quad & J_2(u,v) \\
\text{subject to} \quad & \frac{1}{h_t}(u_{i,j,k+1} - u_{i,j,k}) \\
& - \frac{\kappa_{i,j}}{h_x h_y}(u_{i-1,j,k+1} + u_{i+1,j,k+1} + u_{i,j-1,k+1} + u_{i,j+1,k+1} - 4u_{i,j,k+1}) \\
& = \sum_{l=1}^{L}(v_{k+1,l} f_l(ih_x, jh_y)) \quad \forall (i,j,k) \in \mathcal{I}_1 \\
& u_{i,j,k} = 0 \quad \forall (i,j,k) \in \mathcal{I}_2 \\
& u_{i,j,0} = u_0(ih_x, jh_y) \quad \forall (i,j) \in \mathcal{I}_3 \\
& -M_{\text{big}} w_{k,l} \leq v_{k,l} \leq M_{\text{big}} w_{k,l} \quad \forall (k,l) \in \mathcal{I}_5 \\
& \sum_{l=1}^{L} w_{k,l} = \bar{w} \quad \forall k \in \mathcal{I}_4 \\
& w_{k,l} \in \{0,1\} \quad \forall (k,l) \in \mathcal{I}_5 \\
& v_{k,l} \in \mathbb{R} \quad \forall (k,l) \in \mathcal{I}_5 \\
& u_{i,j,k} \in \mathbb{R} \quad \forall (i,j,k) \in \mathcal{I}_6
\end{aligned}
$$

2.3 Solution

Since the objectives of the MINLP formulation of the problems is quadratic and their constraints are linear, the problems are MIQPs. They can be solved by a suitable solver, e.g. CPLEX, but due to the high number of variables occurring due to the PDE discretization, the solving is very expensive in terms of memory consummation and CPU time. In the following chapters we present accelerating approaches which reduce both issues.

Chapter 3

Preprocessing

The discretized problems are MINLP with only linear constraints and a convex objectives. Thus, we can solve them to optimality by applying standard techniques, such as branch and bound, cutting plane, or hybrid algorithms.

In every node of the huge search tree we have to solve a big QP. To shorten the computational time, we may reduce the number of state variables which, in consequence, causes a faster computation of the relaxed problem in every node of the search tree. Concretely, the number of state variables depends cubicly on the discretization size, the controls only linearly. Hence, if we are able to eliminate state variables we will get a problem formulation with a significantly lower number of variables. Such variable elimination is possible for our problems particularly because they are governed by a linear PDE.

This chapter is structured as follows: first, we introduce the simple PDE Elimination scheme for the Heat Equation with Actuator Placement problem. Second, we extend the approach for the problem with additional actuator operation. Third, we further exploit the structure of the presented problems in order to reduce the computational effort and the memory consumption. This leads to the PDE Elimination with Convolution approach.

3.1 PDE Elimination Simple

As the discretized PDE is a system of linear equations we can actually eliminate the PDE by solving a linear system for every binary variable $w_{k,l}$. This idea leads to a MIQP formulation of the problem with reduced size, which can be solved within less computational time then the original MIQP formulation.

3.1.1 PDE Elimination Simple for Binary Controls

This approach can be written in matrix notation of the discretized PDE:

$$AU = \bar{v}BW + d \tag{3.1}$$

where $A \in \mathbb{R}^{T_n NM \times T_n NM}$ and $B \in \mathbb{R}^{T_n NM \times T_n L}$ contain coefficients, $d \in \mathbb{R}^{T_n NM}$ contains initial and boundary conditions, and $U \in \mathbb{R}^{T_n NM}$ and $W \in \{0,1\}^{T_n L}$ are the unknown states and controls. The scalar $\bar{v} \in \mathbb{R}$ represents the fixed intensity of the actuator. The dimension of the system is determined by the discretization size, so N is the number of grid points in x, M the number of grid points in y, and T_n the number of time steps.

The matrix A can be written as the sum of two Kronecker products:

$$A = C \otimes I_{NM} + I_{T_n} \otimes \kappa K \tag{3.2}$$

Thereby I_{NM} and I_{T_n} denote identity matrices of dimension NM, and T_n respectively. The matrix $C \in \mathbb{R}^{T_n \times T_n}$ is an implicit Euler matrix:

$$C = \frac{1}{h_t} \begin{bmatrix} 1 & & & & \\ -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{bmatrix} \tag{3.3}$$

with $h_t = \frac{T_f}{T_n}$ being the time step size, and $K \in \mathbb{R}^{NM \times NM}$ is the coefficient matrix of the five-point stencil discretization of the Laplace operator:

$$K = \frac{1}{h_x h_y} \begin{bmatrix} D & -I_N & & & & & \\ -I_N & D & -I_N & & & & \\ & -I_N & D & -I_N & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -I_N & D & -I_N & \\ & & & & -I_N & D & -I_N \\ & & & & & -I_N & D \end{bmatrix} \tag{3.4}$$

where $h_x = \frac{1}{N}$ and $h_y = \frac{2}{M}$ denote the space step sizes. The I_N denote identity matrices of dimension N and $D \in \mathbb{R}^{N \times N}$ is a tridiagonal matrix:

$$D = \begin{bmatrix} 4 & -1 & & & & & \\ -1 & 4 & -1 & & & & \\ & -1 & 4 & -1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -1 & 4 & -1 & \\ & & & & -1 & 4 & -1 \\ & & & & & -1 & 4 \end{bmatrix} \tag{3.5}$$

The right hand side of the linear system (3.1) consists of the right hand side of the heat equation (2.1b) written as:

$$B = I_{T_n} \otimes F \tag{3.6}$$

with I_{T_n} the identity matrix of dimension T_n, the source term (2.2) $F = [\mathrm{vec}(f_1), \ldots, \mathrm{vec}(f_L)] \in \mathbb{R}^{NM \times L}$, and the initial (2.1d) and boundary conditions (2.1c): $d = [\mathrm{vec}(u_{0,\cdot,\cdot})^T, 0, \ldots, 0]^T \in \mathbb{R}^{NMT_n}$. To express the unknowns we again use the vector operator, such that the state variables and the binary control variables can be written as vectors:

$$U = \begin{bmatrix} \mathrm{vec}(u_{1,\cdot,\cdot}) \\ \vdots \\ \mathrm{vec}(u_{T_n,\cdot,\cdot}) \end{bmatrix} \quad W = \begin{bmatrix} \mathrm{vec}(w_{1,\cdot}) \\ \vdots \\ \mathrm{vec}(w_{T_n,\cdot}) \end{bmatrix} \tag{3.7}$$

To solve the linear system (3.1) we use a key property of W. As W is a vector in $\{0,1\}^{T_n L}$ we can write it as a linear combination of unit vectors of $\mathbb{R}^{T_n L}$:

$$W = \sum_{i=1}^{T_n L} W_i e_i \tag{3.8}$$

where $\{e_i\}_{i=1,\dots,T_nL}$ denotes the standard basis of $\mathbb{R}^{T_nL}$ and $W_i \in \{0,1\}$ the ith entry of the vector W, the factor W_i corresponds to a $w_{t,l}$, cf. (3.7). Thus, we can reformulate (3.1) with the representation of W in (3.8) :

$$AU = \bar{v}B\sum_{i=1}^{LT_n} W_i e_i + d$$

As A is a regular matrix, we can solve the system by inverting, and we exploit linearity:

$$U = \sum_{i=1}^{LT_n} W_i \underbrace{\bar{v}A^{-1}Be_i}_{\text{inhomogeneous part}} + \underbrace{A^{-1}d}_{\text{homogeneous part}}$$

Applying this approach we can now define presolved solutions. First, we introduce the homogeneous solution $\bar{u}^h$ which is the solution part only representing thermal diffusion without any control application. Thus, it is the solution of the discretization of the homogeneous heat equation with the given initial and boundary conditions (2.1b-2.1d), i.e. all controls are fixed to zero. Second, we define the presolved solution representing the inhomogeneous part of the solution. Let $\bar{u}^{k,l}$ be the set of variables which represent the solution of the discretized heat equation (1.4) with the initial state $u_0 \equiv 0$, the control $w_{k,l} = 1$ and all other $w_{\cdot,\cdot} = 0$. We transfer the notation introduced for the original state in (1.3): $\bar{u}^h_{k,i,j} = \bar{u}^h(kh_t, ih_x, jh_j)$ and $\bar{u}^{t,l}_{k,i,j} = \bar{u}^{t,l}(kh_t, ih_x, jh_j)$ respectively.

Using these presolved solutions we can formulate the reduced MIQP. We plug the presolved solutions in the objective (2.1a) and, as in the regular formulation, we discretize by applying the trapezoidal rule. The reduced objective J_3 of this problem, c.f. (A.3), in combination with the source budget (2.1e) form the reduced MIQP formulation of the problem:

$$\begin{aligned}
\underset{w}{\text{minimize}} \quad & J_3(w) && && \text{(3.9a)}\\
\text{subject to} \quad & \sum_{l=1}^{L} w_{k,l} = \bar{w} && \text{for } k = 1,\dots,T_n && \text{(3.9b)}\\
& w_{k,l} \in \{0,1\} \text{ for all } l \in \{1,\dots,L\} \text{ and } k = 1,\dots,T_n. && && \text{(3.9c)}
\end{aligned}$$

This can be solved with standard techniques by using a MIQP solver. We remark, that this problem is a purely combinatorial one since there are binary variables only. We summarize the approach in Algorithm 1.

The preprocessing happens in steps 1 and 2 of the algorithm, where we solve $L \cdot T_n + 1$ linear systems. The obtained precalculated solutions can be plugged into the reduced model (3.9) which does not depend on state variables anymore and is solved in step 3. Finally, we build the solution by sticking together the optimal controls and the corresponding presolved solutions, cf. line 4.

We remark that step 1 and 2 of the algorithm can be easily parallelized.

3.1.2 PDE Elimination Simple for Binary and Continuous Controls

We extend the approach of the last paragraph for the problem which includes additional continuous controls. Therefore, we modify the matrix notation of the discretized PDE

Algorithm 1: PDE Elimination Simple for Binary Controls

Input: Control of Heat Equation with Actuator Placement instance
Output: Optimal solution (u, w)

1 calculate the homogeneous solution $\bar{u}^h$ with

$$\text{vec}(\bar{u}^h) = A^{-1}d$$

2 calculate the presolved solutions $\bar{u}^{t,l}$ for all (t, l) tupel by solving:

$$\text{vec}(\bar{u}^{t,l}) = \bar{v}A^{-1}Be_i$$

(where (t, l) corresponds to i)

3 solve the reduced MIQP (3.9) $\Rightarrow$ optimal w

4 construct the solution for $k = 1, \ldots, T_n$, $i = 1, \ldots, N$, and $j = 1, \ldots, M$:

$$u_{k,i,j} = \bar{u}^h_{k,i,j} + \sum_{t=1}^{T_n} \sum_{l=1}^{L} w_{t,l}\ \bar{u}^{t,l}_{k,i,j}$$

5 **return** (u, w)

(3.1) by replacing the vector of the binaries W by the vector of the continuous controls V. As the continuous controls represent the intensity of placed actuators, we remove $\bar{v}$ from (3.1) and get:

$$AU = BV + d$$

where the matrices A and B and the vector d are identical to the ones introduced earlier in (3.2-3.6). As the vector V has the same linearity property as W, c.f. (3.8), we can apply the same reformulations. By inverting A, it yields the following representation of the state variables:

$$U = \sum_{i=1}^{LT_n} V_i \quad \underbrace{A^{-1}Be_i}_{\text{inhomogeneous part}} \quad + \quad \underbrace{A^{-1}d}_{\text{homogeneous part}}$$

With this, we get presolved state solutions and for which we use the same notation as earlier: $\bar{u}^h$ denotes the homogeneous part of the solution and $\bar{u}^{k,l}$ denotes the solution part for which the corresponding control is set to one, i.e. $v_{k,l} = 1$. We plug these presolved solutions into the original objective (2.4a) and get the reduced objective J_4 (A.4) for this problem. With the Big M (2.4e) and the source budget (2.4f) constraints, we can formulate the reduced MIQP as follows:

$$\underset{v,w}{\text{minimize}} \quad J_4(w) \tag{3.10a}$$

$$\text{subject to} \quad -Mw_{k,l} \leq v_{k,l} \leq Mw_{k,l} \qquad \text{for } k = 1, \ldots, T_n \tag{3.10b}$$

$$\sum_{l=1}^{L} w_{k,l} = \bar{w} \qquad \text{for } k = 1, \ldots, T_n \tag{3.10c}$$

$$w_{k,l} \in \{0, 1\} \text{ for all } l \in \{1, \ldots, L\} \text{ and } k = 1, \ldots, T_n. \tag{3.10d}$$

We summarize the approach in Algorithm 2. First, we compute the presolved states by solving $T_n \cdot L + 1$ linear systems, c.f. steps 1 and 2, which can be computed in parallel. Second, we solve the reduced MIQP (3.10) with a suitable solver in step 3. Finally, we can construct the state solution with the optimal control values.

Algorithm 2: PDE Elimination Simple for Binary and Continuous Controls

Input: Control of Heat Equation with Actuator Placement and Operation instance

Output: Optimal solution (u, v, w)

1 calculate the homogeneous solution $\bar{u}^h$ with

$$\text{vec}(\bar{u}^h) = A^{-1}d$$

2 calculate the presolved solutions $\bar{u}^{t,l}$ for all (t, l) tupel by solving:

$$\text{vec}(\bar{u}^{t,l}) = A^{-1}Be_i$$

(where (t, l) corresponds to i)

3 solve the reduced MIQP (3.10) $\Rightarrow$ optimal (v, w)

4 construct the solution for $k = 1, \ldots, T_n$, $i = 1, \ldots, N$, and $j = 1, \ldots, M$:

$$u_{k,i,j} = \bar{u}^h_{k,i,j} + \sum_{t=1}^{T_n} \sum_{l=1}^{L} v_{t,l}\ \bar{u}^{t,l}_{k,i,j}$$

5 **return** (u, v, w)

3.2 PDE Elimination with Convolution

Further improvements of this preprocessing scheme can be obtained by looking more in detail at the presolved solutions $\bar{u}^{k,l}$. For every possible actuator location we have to solve a linear system in every time step to get these presolved state solutions. That effort can be reduced since the effect of the control depends in particular on the location since the source term is time independent, i.e. the right-hand side of the PDE depends on time implicitly only through the controls on time. The time dependency can be expressed as

the difference to the time when the actuator was placed. So instead of solving for every time step and every location a linear system, it suffices to solve for all locations once.

3.2.1 PDE Elimination with Convolution for Binary Controls

Therefore we define new presolved solutions $\bar{u}^l$ as the solution of the heat equation with Dirichlet boundary if $w_{1,l} = 1$.

With the concept of discrete convolution, introduced in Chapter 1, the relation between the state solution of the original MIQP and the solution of its reduction is for $k = 1, \dots, T_n$, $i = 1, \dots, N$, and $j = 1, \dots, M$:

$$u_{k,i,j} = \bar{u}^h_{k,i,j} + \sum_{l=1}^{L} (w_{\cdot,l} * \bar{u}^l_{\cdot,i,j})[k] = \bar{u}^h_{k,i,j} + \sum_{l=1}^{L} \sum_{m=1}^{k} w_{m,l} \bar{u}^l_{k-m,i,j} \tag{3.11}$$

Similar to the reduced objective of the simple PDE Elimination (A.3), we apply this relation to the objective of the MIPDECO (2.1) and obtain the reduced objective J_5 of this problem formulation (A.5): With this objective we formulate the reduced MIQP of the problem as follows:

$$\begin{aligned}
\underset{w}{\text{minimize}} \quad & J_5(w) && && \text{(3.12a)}\\
\text{subject to} \quad & \sum_{l=1}^{L} w_{k,l} = \bar{w} && \text{for } k = 1, \dots, T_n && \text{(3.12b)}\\
& w_{k,l} \in \{0, 1\} \text{ for all } l \in \{1, \dots, L\} \text{ and } k = 1, \dots, T_n. && && \text{(3.12c)}
\end{aligned}$$

We summarize this approach in Algorithm 3. First, we solve $L+1$ linear systems in order to obtain the presolved solutions $\bar{u}$ in step 1 and 2 of the algorithm. This linear systems solving can be done in parallel. The presolved solutions serve as data for the reduced MIQP, which is then solved in step 3 by a suitable solver. Finally, we can construct the state variables with the optimal $w_{k,l}$, cf. line 4.

3.2.2 PDE Elimination with Convolution for Binary and Continuous Controls

As for the PDE Elimination Simple, we want to extend the approach with convolution to the problem Heat Equation with Actuator Placement and Operation.

Therefore we keep the notation of the presolved states: $\bar{u}^h$ denotes the homogeneous state solution and $\bar{u}^l$ denotes the solution of the PDE if $v_{1,l} = 1$ and all other controls are inactive.

Similarly to (3.11), we can express the relation between presolved solutions and the continuous controls by the help of discrete convolution:

$$u_{k,i,j} = \bar{u}^h_{k,i,j} + \sum_{l=1}^{L} (v_{\cdot,l} * \bar{u}^l_{\cdot,i,j})[k] = \bar{u}^h_{k,i,j} + \sum_{l=1}^{L} \sum_{m=1}^{k} v_{m,l} \bar{u}^l_{k-m,i,j} \tag{3.13}$$

With the aid of this relation, we reformulate the objective (2.4a) in order to get the objective of the reduced problem J_6, c.f. (A.6). The reduced MIQP of the problem is composed by the reduced objective and by the constraints Big M (3.14c) and source budget (3.14d).

Algorithm 3: PDE Elimination with Convolution for Binary Controls

Input: Control of Heat Equation with Actuator Placement instance
Output: optimal solution (u, w)

1 calculate the homogeneous solution $\bar{u}^h$ with

$$\mathrm{vec}(\bar{u}^h) = A^{-1}d$$

2 calculate the presolved solutions $\bar{u}^l$ for all $l = 1, \ldots, L$ by solving:

$$\mathrm{vec}(\bar{u}^l) = \bar{v}A^{-1}Be_l$$

3 solve the reduced MIQP (3.12) $\Rightarrow$ optimal w
4 construct the solution for $k = 1, \ldots, T_n$, $i = 1, \ldots, N$, and $j = 1, \ldots, M$:

$$u_{k,i,j} = \bar{u}^h_{k,i,j} + \sum_{l=1}^{L}\sum_{m=1}^{k} v_{l,m}\bar{u}^l_{k-m,i,j}$$

5 **return** (u, w)

$$\begin{aligned}
\underset{v,w}{\text{minimize}} \quad & J_6(w) && && \text{(3.14a)}\\
\text{subject to} \quad & -Mw_{k,l} \leq v_{k,l} \leq Mw_{k,l} && \text{for } k = 1, \ldots, T_n && \text{(3.14b)}\\
& \sum_{l=1}^{L} w_{k,l} = \bar{w} && \text{for } k = 1, \ldots, T_n && \text{(3.14c)}\\
& w_{k,l} \in \{0, 1\} \text{ for all } l \in \{1, \ldots, L\} \text{ and } k = 1, \ldots, T_n. && && \text{(3.14d)}
\end{aligned}$$

We summarize the approach in Algorithm 4. First, we solve $L+1$ linear systems in order to obtain the presolved states in step 1 and 2. This presolving can be done in a parallel manner. With the presolved states, we formulate the reduced MIQP which is solved in step 3 by a suitable solver. Finally, we construct the state solution on the basis of the precalculated states and the optimal controls in step 4.

Algorithm 4: PDE Elimination with Convolution for Binary and Continuous Controls

Input: Control of Heat Equation with Actuator Placement and Operation instance

Output: optimal solution (u, v, w)

1 calculate the homogeneous solution $\bar{u}^h$ with

$$\mathrm{vec}(\bar{u}^h) = A^{-1}d$$

2 calculate the presolved solutions $\bar{u}^l$ for all $l = 1, \ldots, L$ by solving:

$$\mathrm{vec}(\bar{u}^l) = A^{-1}Be_l$$

3 solve the reduced MIQP (3.14) $\Rightarrow$ optimal (v, w)

4 construct the solution for $k = 1, \ldots, T_n$, $i = 1, \ldots, N$, and $j = 1, \ldots, M$:

$$u_{k,i,j} = \bar{u}^h_{k,i,j} + \sum_{l=1}^{L} \sum_{m=1}^{k} v_{l,m} \bar{u}^l_{k-m,i,j}$$

5 **return** (u, v, w)

Chapter 4

Heuristics

Due to the preprocessing techniques of state elimination in the previous chapter, we can reduce the computational effort for our problems significantly. Nevertheless, the effort for finer meshes is doubtless still large. To further increase the computational speed, we present rounding based heuristics which also produce meaningful solutions. Rounding is of great importance since it can be applied even for other problems which do not necessary allow state elimination, e.g. problems which are governed by nonlinear PDEs. First, we want to introduce a general rounding scheme:

Algorithm 5: General Rounding for MIPDECO

Input: MINLP formulation of MIPDECO
Output: integer feasible (u, v, w)

1 **while** *unfixed integer variables* w *left* **do**
2 | solve the NLP relaxation of the (sub-)problem $\Rightarrow (u, v, w)$
3 | round some integer variables w
4 **end**
5 solve remaining NLP with the fixed $w \Rightarrow (u, v)$
6 do MINLP-improvement heuristic
7 resolve PDE on a fine mesh $\Rightarrow u$
8 **return** (u, v, w)

In the first step, line 2, we ignore integer conditions on w, called NLP relaxation, and solve the problem. Next, we choose a certain subset of the integer variables w and fix them in a certain way to integers, cf. line 3. The details of this rounding can be problem specific. After rounding and fixing some integers we resolve the problem relaxation which is now of smaller size. We round and resolve until we fixed all integer variables. This way we may get a feasible solution. To further increase the solution quality, we can apply MINLP-improvement heuristics, e.g. local branching [6, 17], cf. line 6. This includes usually solving a smaller MINLP, compared to the original discretized problem. Afterward, we can improve the solution quality by resolving the PDE without optimization on a finer mesh, cf. line 7. In general the obtained solution is not necessarily feasible nor optimal, but usually "good". Furthermore, rounded solutions which are feasible can be used as an upper bound in a branch and bound algorithm in order to improve the processing time. In this chapter, we present four rounding heuristics for the problems introduced in Chapter 2. Therefore we first introduce the heuristic for the Heat Equation with Actuator Placement problem and then extend it for the Heat Equation with Actuator Placement and Operation problem.

4.1 Maximum-Rounding

The Maximum-Rounding is inspired by the fact that we allow only a certain number of actuators locations to be active at one time, c.f. source budget constraint (2.1e). As we round on the basis of NLP relaxed solution, we have fractional values of the binaries and then round the largest of the time step to one.

4.1.1 Maximum-Rounding for Control of Heat Equation with Actuator Placement

The binary variable $w_{k,l}$ represents the activity of the actuator in position l at the time k, taking the value 1 for activity and 0 for no activity. First (line 1 in Alg. 6), we solve the NLP relaxation of the problem. Then, the Maximum-Rounding chooses the location l for every time step k such that $w_{k,l}$ has the largest value, cf. line 3, and in line 4 this variable is rounded to 1. If we allow more than one actuator, we round the $\bar{w}$ largest values to 1. Due to the source budget constraint (2.1e) we can fix the other variables of the time step to 0, cf. line 5, which represent the other possible, but in this time step inactive, actuator locations. Then, in line 1, we can solve the remaining relaxation and repeat this procedure for the next time step.

If we do not resolve in every time step, we have to solve the remaining QP after we completed the rounding, c.f. step 8. As the PDE states u are determined in a unique manner by the controls w, this last solving after fixing all controls is a simulation only. So a linear system is solved and the objective is evaluated.

Algorithm 6: Maximum-Rounding for Binary Controls

Input: Control of Heat Equation with Actuator Placement instance
Output: integer feasible (u, w)

1 solve relaxation of MIQP $\Rightarrow (u, w)$
2 **for** $k = 1$ ***to*** T_n **do**
3 $\quad \mathcal{L} = \{l \in \{1, \ldots, L\} \big|\ w_{k,l}$ is one of the $\bar{w}$ largest for $k\}$
4 $\quad$ fix $w_{k,l} = 1$ for all $l \in \mathcal{L}$
5 $\quad$ fix $w_{k,l} = 0$ for all $l \notin \mathcal{L}$
6 $\quad$ solve the NLP relaxation of the remaining sub-MIQP $\Rightarrow$ new (u, w) (optional resolve)
7 **end**
8 solve the remaining sub-QP $\Rightarrow u$
9 **return** (u, w)

We remark that the resolving in step 6 is optional and expensive because the number of QP solvings depends on the number of time steps but we expect better solutions due to this additional effort.

4.1.2 Maximum-Rounding for Control of Heat Equation with Actuator Placement and Operation

As in the actuator placement problem, the binary variable $w_{k,l}$ represents the activity of the actuator in position l at the time k. But we have additional continuous variables $v_{k,l}$ representing the intensity of this activity.

The Maximum-Rounding for Heat Equation with Actuator Placement (Alg. 6) can also be applied on the problem with additional actuator operation. However, the experiments indicated that in NLP relaxed solution all binary variables have approximately the same value:

$$w_{k,l} \approx \frac{\bar{w}}{L} \quad \text{for } l = 1, \dots, L \text{ and } k = 1, \dots, T_n \tag{4.1}$$

But instead of rounding on the basis of the binary controls, we can round upon the values of the continuous controls $v_{k,l}$. Thus, we choose the location l for every time step k such that $v_{k,l}$ has the largest absolute value, cf. line 3 in Alg. 7, and round the corresponding $w_{k,l}$ to 1.

Therefore we replace the line 3 in Algorithm 6 and get Algorithm 7.

Unlike in Algorithm 6, the last step (line 8) of this algorithm is still an optimization since the continuous controls v are unfixed.

Algorithm 7: Maximum-Rounding for Binary and Continuous controls

Input: Control of Heat Equation with Actuator Placement and Operation instance

Output: integer feasible (u, v, w)

1 solve relaxation of MIQP $\Rightarrow (u, v, w)$
2 **for** $k = 1$ ***to*** T_n **do**
3 $\quad \mathcal{L} = \{l \in \{1, \dots, L\} \mid |v_{k,l}| \text{ is one of the } \bar{w} \text{ largest for } k\}$
4 $\quad$ fix $w_{k,l} = 1$ for all $l \in \mathcal{L}$
5 $\quad$ fix $w_{k,l} = 0$ for all $l \notin \mathcal{L}$
6 $\quad$ solve the NLP relaxation of the remaining sub-MIQP $\Rightarrow$ new (u, v, w) (optional resolve)
7 **end**
8 solve the remaining sub-QP $\Rightarrow (u, v)$
9 **return** (u, v, w)

Alternatively, it can be considered to round upon the largest absolute product value $|w_{k,l} v_{k,l}|$. In this setting, we did not study this alternative because of the behavior of the binary controls (4.1) but it may be crucial for other problems.

4.2 Sum-Up-Rounding for MIPDECO

The main idea is based on the fact that we first discretize in space. Spatially discretized PDEs can be interpreted as a set of ordinary differential equations (ODEs), which is an approach known as the method of lines, c.f. Section 1.2. Therefore we can apply a well known technique for mixed-integer optimal control: Sum-Up-Rounding (SUR), which was introduced by Sager [19]. It is guaranteed that for sufficient fine meshes that the SUR solution is arbitrarily close to the optimal solution. The SUR approach rounds a vector of time ordered binary variables on the basis of an NLP relaxed solution. We proceed forward in time by not only taking the fractional value of a binary into account but the rounding residual of the previous times as well.

4.2.1 Sum-Up-Rounding for Control of Heat Equation with Actuator Placement

First, we solve the NLP relaxation of the MIQP, cf. line 2. Next, in lines 4 and 5, we choose one location l and apply SUR to fix the control variables $w_{\cdot,l}$ at this location to binary values. Then we resolve the NLP relaxation with the fixed variables, cf. line 7. Then we continue with the next location until we have handled all possible locations.

To satisfy the source budget constraint (2.1e) we need to track the time steps where we fixed a binary variable to 1. We initialized a as memory for proceeded time steps, if we reach the limit of $\bar{w}$ actuators for a time step, we fix the open variables in this time step to 0, cf. line 7. However, it is possible that there are still time steps which do not satisfy the source budget constraint. To overcome this issue, we detect these time steps and $(**)$ randomly choose locations which are not yet active in this time step.

If we do not resolve in every time step, we have to solve the remaining QP after we completed the rounding, c.f. step 16, which is like in Algorithm 6 a simulation only.

Algorithm 8: Sum-Up-Rounding for MIPDECO

Input: Control of Heat Equation with Actuator Placement instance
Output: integer feasible (u, w)

1 Initialize set of open locations $Q = \{1, \dots, L\}$ and proceeded time steps $a = (a_1, \dots, a_{T_n}) = 0$
2 solve relaxation of MIQP $\Rightarrow (u, v, w)$
3 **while** $Q \neq \varnothing$ **do**
4 | choose* and remove a location $l \in Q$
5 | **SUR**$(w, l, a) \Rightarrow$ modified w and a
6 | for all k with $a_k = \bar{w}$ fix $w_{k,l} = 0$ for all $l \in Q$
7 | solve the NLP relaxation of the remaining sub-MIQP $\Rightarrow (u, w)$ (optional resolve)
8 **end**
9 **for** $k = 1$ ***to*** T_n **do**
10 | **while** $a_k < \bar{w}$ **do**
11 | | choose** $l \in \{1, \dots, L \big| w_{k,l} = 0\}$
12 | | fix $w_{k,l} = 1$
13 | | let $a_k = a_k + 1$
14 | **end**
15 **end**
16 solve the remaining sub-QP $\Rightarrow u$
17 **return** (u, w)

Sum-Up-Rounding for MIPDECO calls a subroutine **SUR** (Alg. 9) in line 5 which applies the Sum-Up-Rounding strategy to a time ordered set of binary variables.

For this we initialize two variables: the sum of the relaxed values q and the sum of the rounded values p. Forward in time, we update the sum of the relaxed values q by adding the current $w_{k,l}$, cf. line 3. Then, in line 4, we compare the sum variables: If the difference, also called rounding residual, is less or equal 0.5, we fix the current $w_{k,l}$ to 0, cf. line 9. Else we fix it to 1 and update the rounded sum, cf. lines 5 and 6. Further, in line 7, the tracking variable a_k is updated to ensure the actuator budget (2.1e) of the

current time step. Afterward we repeat this procedure for the next time step.

Algorithm 9: $(w, a) = \mathbf{SUR}(w, l, a)$

```
1  Initialize sum of relaxed values q = 0 and sum of rounded values p = 0
2  for k = 1 to T_n do
3  |   q = q + w_{k,l}
4  |   if q − p > 0.5 then
5  |   |   fix w_{k,l} = 1
6  |   |   p = p + 1
7  |   |   a_k = a_k + 1
8  |   else
9  |   |   fix w_{k,l} = 0
10 |   end
11 end
12 return (w, a)
```

*We observe that it is not obvious in which order we should proceed through the locations. In our implementation we used a greedy approach: To determine the proceeding order of the locations, we choose always the location which corresponds to the w_l with the largest L_1-norm. We approximate the L_1-norm by the sum of the fractional values of the binary controls:

$$||w_l(t)||_{1,T} = \int_0^{t_f} |w_l(t)| \, \mathrm{d}t \approx h_t \sum_{k=1}^{T_n} w_{k,l}$$

As for the Maximum-Rounding algorithms, the resolve in step 7 is optional but we expect better solutions due to this additional effort. However, an advantage of Sum-Up-Rounding for MIPDECO is that the number of QP resolvings depends on the number of actuator locations, so it does not depend on the discretization.

4.2.2 Sum-Up-Rounding for Control of Heat Equation with Actuator Placement and Operation

The Sum-Up-Rounding for actuator placement can be used for instances with actuator placement and operation. However, the proceeding order of the locations is different in our implementations. Instead of choosing upon the norm of binary controls, an approximate L_1-norm of the continuous controls is computed:

$$||v_l(t)||_{1,T} = \int_0^{t_f} |v_l(t)| \, \mathrm{d}t \approx h_t \sum_{k=1}^{T_n} |v_{k,l}|$$

So to determine the proceeding order, the locations are ordered in a descending manner with respect to their L_1-norm approximation.

4.3 Maximum-Sum-Up-Rounding

The Maximum-Rounding chooses the largest controls on every time step to fix them to 1 but it does not track the decisions made in previous time steps. In contrast, the Sum-

Up-Rounding keeps track of the decisions made prior to the current time step by the computation of a rounding residual. But its performance is limited due to the source budget constraint and to the fact that we can only use coarser time meshes than usually applied in an ODE setting.

With the Maximum-Sum-Up-Rounding we develop a rounding scheme which combines the advantages of the previous seen rounding approaches. Therefore we proceed forward in time and choose the control location with the largest rounding residual at the time step to round to 1.

This procedure can be interpreted as a global or a simultaneous Sum-Up-Rounding.

4.3.1 Maximum-Sum-Up-Rounding for Control of Heat Equation with Actuator Placement

As in the other rounding heuristics we start in line 1 of Alg. 10 by computing a solution of the NLP relaxed problem. The fractional values of the binary controls w are saved as $\mathcal{W}$ in line 3. Then we compute all rounding residuals at the current time step by adding the fractional w to the rounding residual made in the previous time steps.

We choose the locations with the $\bar{w}$ largest residuals and round the corresponding controls to one, c.f. line 4 and 5. As we need to satisfy the source budget constraint, we can round the remaining controls of this time step to zero (line 6). Then we can solve the NLP relaxation of the remaining sub-MIQP and proceed to the next time step until we fixed all binary variables. The resolve in every time step is optional and expensive but we expect to improve the solution quality with the additional effort.

If we decide not to resolve in every time step, we need to solve once more after completion of the rounding. As for the other roundings for the Placement problem, this is a simulation only.

Algorithm 10: Maximum-Sum-Up-Rounding for Binary Controls

Input: Control of Heat Equation with Actuator Placement instance
Output: integer feasible (u, w)

1 solve relaxation of MIQP $\Rightarrow (u, w)$
2 **for** $k = 1$ ***to*** T_n **do**
3 let $\mathcal{W}_{k,l} = w_{k,l}$ for $l = 1, \dots, L$
4 calculate rounding residual $r_{k,l} = w_{k,l} + \sum_{\tau=0}^{k-1} (\mathcal{W}_{\tau,l} - w_{\tau,l})$ for $l = 1, \dots, L$
5 $\mathcal{L} = \{l \in \{1, \dots, L\} \,|\, r_{k,l}$ is one of the $\bar{w}$ largest for $k\}$
6 let $w_{k,l}$=1 for $l \in \mathcal{L}$
7 let $w_{k,l}$=0 for $l \notin \mathcal{L}$
8 solve the QP relaxation of the remaining sub-MIQP $\Rightarrow (u, w)$ (optional resolve)
9 **end**
10 solve the remaining QP $\Rightarrow u$
11 **return** (u, w)

4.3.2 Maximum-Sum-Up-Rounding for Control of Heat Equation with Actuator Placement and Operation

The problem with Actuator Placement and Operations includes, besides the binary controls w, continuous controls v representing the intensity of a placed actuator. In order to take into account both control types, we modify Algorithm 10 slightly.

After solving the NLP relaxation of the problem (line 1, we compute a measure of the total control amount in the current time step, c.f. line 3. In line 4, we then save for all locations the fractional value $\mathcal{W}$ representing the locations distribution to the total control amount s_k of the time step. We determine the rounding residuals on the basis of $\mathcal{W}$. Then, we proceed like in Algorithm 10. After choosing the locations with the largest rounding residual, we fix the corresponding binary variables to one and the others to zero, c.f. lines 6 to 8. After this rounding, we can chose to resolve before moving to the next time step. If we decide not to resolve in every for loop, we have to solve the remaining QP, line 11, to determine the optimal continuous controls v and the corresponding state variables u.

Algorithm 11: Maximum-Sum-Up-Rounding for Binary and Continuous Controls

Input: Control of Heat Equation with Actuator Placement and Operation instance

Output: integer feasible (u, v, w)

1 solve relaxation of MIQP $\Rightarrow (u, v, w)$
2 **for** $k = 1$ ***to*** T_n **do**
3 let $s_k = \sum_{l=1}^{L} |w_{k,l} v_{k,l}|$
4 let $\mathcal{W}_{k,l} = |w_{k,l} v_{k,l}| / s_k$ for $l = 1, \ldots, L$
5 calculate rounding residual $r_{k,l} = \mathcal{W}_{k,l} + \sum_{\tau=0}^{k-1} (\mathcal{W}_{\tau,l} - w_{\tau,l})$ for $l = 1, \ldots, L$
6 $\mathcal{L} = \{l \in \{1, \ldots, L\} \big|\ r_{k,l}$ is one of the $\bar{w}$ largest for $k\}$
7 let $w_{k,l}$=1 for $l \in \mathcal{L}$
8 let $w_{k,l}$=0 for $l \notin \mathcal{L}$
9 solve the QP relaxation of the remaining sub-MIQP $\Rightarrow (u, v, w)$ (optional resolve)
10 **end**
11 solve the remaining sub-QP $\Rightarrow (u, v, w)$
12 **return** (u, v, w)

4.4 Spatial-Rounding

Unlike the rounding based heuristics we have seen so far in this chapter, Spatial-Rounding does not generate a feasible solution. However, it is meant to be combined with any of the other three rounding schemes.

We start with computing the NLP relaxed solution of the problem. On this basis we manipulate the control values by using spatial information of the possible actuator locations. Then, we apply one of the other rounding schemes (without resolving). So instead of rounding the control values of the relaxed solution, it is rounded on basis of the modified controls which are obtained by a weighted average of the control and its neighboring

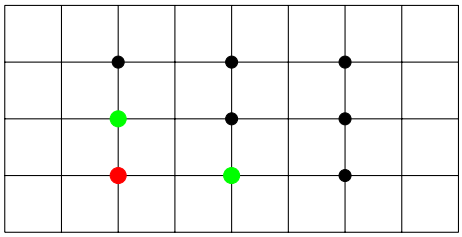

Figure 4.1: Order one neighbors

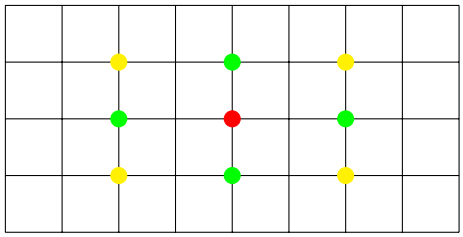

Figure 4.2: Order one and two neighbors

controls.

Therefore let $\mathcal{N}^1(l)$ be the set of neighboring locations of order one of the location l. These are the locations which are next to l on the grid. In Figure 4.1, the location l is colored red and its order-one neighbors are green. We abbreviate the set of neighboring locations of order two of the location l with $\mathcal{N}^2(l)$. This set is illustrated in Figure 4.2 by the color yellow.

Furthermore, we introduce the weights $\sigma_1, \sigma_2 \in (0,1]$, usually $\geq \frac{1}{2}$. With this we are able to define a weighted average for the binary controls:

$$\hat{w}_{k,l} = \sigma_1 w_{k,l} + (1-\sigma_1)\left(\frac{\sigma_2}{|\mathcal{N}^1(l)|}\sum_{\bar{l}\in\mathcal{N}^1(l)} w_{k,\bar{l}} + \frac{1-\sigma_2}{|\mathcal{N}^2(l)|}\sum_{\bar{l}\in\mathcal{N}^2(l)} w_{k,\bar{l}}\right)$$

for $k = 1,\dots,T_n$ and $l = 1,\dots,L$ and in case of additional continuous controls, respectively:

$$\hat{v}_{k,l} = \sigma_1 v_{k,l} + (1-\sigma_1)\left(\frac{\sigma_2}{|\mathcal{N}^1(l)|}\sum_{\bar{l}\in\mathcal{N}^1(l)} v_{k,\bar{l}} + \frac{1-\sigma_2}{|\mathcal{N}^2(l)|}\sum_{\bar{l}\in\mathcal{N}^2(l)} v_{k,\bar{l}}\right)$$

In general the new controls $\hat{w}$ do not satisfy the source budget constraint, c.f. appendix B for a counter example. In order to avoid constraint violation, we scale them for $k = 1,\dots,T_n$ and $l = 1,\dots,L$:

$$w^s_{k,l} = \frac{1}{\sum_{\bar{l}=1}^{L}\hat{w}_{k,\bar{l}}}\,\hat{w}_{k,l}$$

Similarly, we scale $\hat{v}_{k,l}$ to preserve the total amount of (continuous) control application in one time step for $k = 1,\dots,T_n$ and $l = 1,\dots,L$:

$$v^s_{k,l} = \frac{\sum_{\bar{l}=1}^{L} v_{k,\bar{l}}}{\sum_{\bar{l}=1}^{L}\hat{v}_{k,\bar{l}}}\,\hat{v}_{k,l}$$

After completing Spatial-Rounding, Maximum-Rounding, Sum-Up-Rounding, or Maximum-Sum-Up-Rounding is applied. But we leave out the first step of the algorithms, where we would solve the NLP relaxation of the MIQP since we computed it prior to Spatial-Rounding and since we want to round on the basis of w^s and v^s.

4.5 Feasibility Proof

In this section, we argue why the rounding schemes (except for Spatial Rounding) terminate with feasible solutions of the MIQP formulation of the problem.

4.5.1 Control of Heat Equation with Actuator Placement

In Chapter 3 Preprocessing, we developed reduced problem formulations. These formulations are equivalent to the original MIQPs since the presolved state solutions are uniquely determined by a regular system of linear equations. The reduced MIQPs (3.9) and (3.12) contain only binary control variables and have the source budget constraints only:

$$\sum_{l=1}^{L} w_{k,l} = \bar{w} \text{ for } k = 1, \dots, T_n$$

On the basis of the reduced MIQPs, we analyze the rounding schemes with respect to constraint violation.

The Maximum-Rounding (Algorithm 6) chooses the $\bar{w}$ largest control values for every time step which are rounded to 1, all other controls of this time step are rounded to 0. Therefore the binary condition on the control variables $w_{k,l}$ is satisfied. Furthermore, as we round exactly $\bar{w}$ to 1, the source budget constraint is satisfied, too.

The Sum-Up-Rounding for MIPDECO applies the Sum-Up-Rounding scheme for ODEs on every location, c.f. Algorithm 8 line 5. The variables are either rounded to 0 or to 1, c.f. Algorithm 9, so the binary condition is satisfied. As we keep track how many variables are already rounded to 1 in any time step, we never round more than $\bar{w}$ to 1 in a time step, c.f. Algorithm 9 line 7 and Algorithm 8 line 6. In order to fulfill the constraint with equality we randomly choose after Sum-Up-Rounding variables to set to 1 until the source budget constraint is satisfied, c.f. Algorithm 8 lines 9 to 13.

The Maximum-Sum-Up-Rounding (Algorithm 10) chooses the $\bar{w}$ largest rounding residuals in every time step and rounds the corresponding control variables to 1 and the others of the time step to 0. Therefore, the binary condition and the source budget constraints are satisfied.

4.5.2 Control of Heat Equation with Actuator Placement and Operation

The reduced MIQP of the Heat Equation with Actuator Placement and Operation problem contains the source budget constraints and the binary conditions on the binary control variables, too. In addition, it includes continuous controls $v_{k,l}$, their feasible set is restricted by the Big M constraints:

$$-M_{\text{big}} w_{k,l} \leq v_{k,l} \leq M_{\text{big}} w_{k,l} \text{ for } k = 1, \dots, T_n$$

The Maximum-Rounding, the Sum-Up-Rounding for MIPDECO, and the Maximum-Sum-Up-Rounding are applied in a very similar way as for the version with only binary variables, so source budget and binary conditions are fulfilled. However, the Big M constraints are violated after rounding. But as we require a subsequent optimization run after fixing the binary variables to 0/1, c.f. last steps of Algorithm 7, 8, and 11, the QP solver returns a feasible solution with respect to the Big M constraint.

Chapter 5

Numerical Results

In Chapter 2 we introduced two MIPDECOs governed by the heat equation with Dirichlet boundary conditions:

- Control of Heat Equation with Actuator Placement (Pl)
- Control of Heat Equation with Actuator Placement and Operation (PlOp)

Furthermore, we presented their MIQP formulation. In Chapter 3 and 4, we developed algorithms and heuristics to reduce the computational effort for solving the problems. The models and algorithms are implemented in AMPL (A Mathematical Programming Language). The details of the developed framework can be found in Appendix D.

In this chapter we analyze and discuss the approaches. First, we explain the decoupling of states and control meshes and give the parameter set used for the experiments. Then, we do a CPU time comparison of the straight forward application of CPLEX and the application of CPLEX with prior preprocessing. Moreover, we look at benefit of preprocessing in terms of time consumption for the computation of the NLP relaxations and rounding heuristics.

Then, we compare the rounded solutions without resolving between rounding steps with the optimal solutions obtained by CPLEX. Afterward, we analyze the behavior of Sum-up rounding approaches for the placement only problem on finer time grids.

Furthermore, we compare the rounding heuristics with and without prior spatial rounding. Finally, we discuss the benefit and cost of resolving between rounding steps.

5.1 Preliminaries

5.1.1 Decoupling

The state and the control variables are discretized on the same time mesh (1.3). In the AMPL implementation, we decoupled states and controls such that coarser meshes for control variables are feasible. For example we experiment mostly with the space discretization $N = 32$ and $M = 64$ and the time discretization $T_n = 32$. With this fine state discretization we compare different coarser control grids: $T_c = 4, 8, 16, 32$. In Figure 5.1, we visualize the objective values for different control grid choices for both problems (Pl/PlOp). As expected, we observe that finer control grids lead to smaller objective values for both, the NLP relaxation and the MIQP solutions.

With this decoupling we can compare different control grids more reasonably, since we do not have effects which arise from different mesh sizes of PDE discretization. However, we are aware that fine PDE simulation is usually very costly.

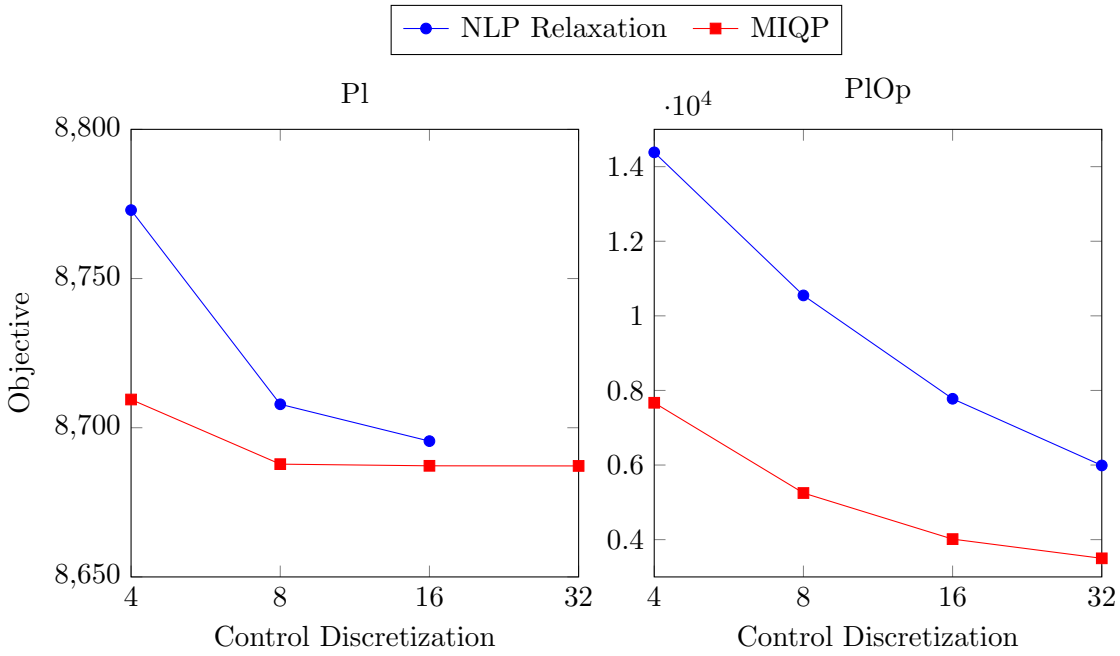

Figure 5.1: Objective Values MIQP and NLP relaxation

5.1.2 Studied Instances

The implementation of the problems is designed in a flexible manner in order to allow easy comparison of different parameter sets. In early experiments, we tried out many parameter sets to find interesting instances. In this chapter, we discuss results obtained with the parameters given in Table 5.1.

Parameter Name	Value
Desired final state	$u_f(x,y) = 0$
Thermal diffusivity	$\kappa = 0.01$
Actuator intensity	$\bar{v} = 5$
Source term variance	$\varepsilon = 0.01$
Actuator quantity	$\bar{w} = 1$
Actuator locations	$L = 9$
Big M	$M_{\text{big}} = 2500$
Mesh points in x	$N = 32$
Mesh points in y	$M = 64$
Initial values (Pl)	$u_0(x,y) = (a_1x^4 + a_2x^3 + a_3x^2) \cdot (b_1y^4 + b_2y^3 + b_3y^2)$
Initial values (PlOp)	$u_0(x,y) = 100\sin(\pi x)\sin(\pi y)$
Spatial rounding	$\sigma_1, \sigma_2 = 0.25$

Table 5.1: Parameter Values

The coefficients a_1, a_2, a_3 and b_1, b_2, b_3 are such that the function u_f has a stationary point in $(x,y) = (0.7, 0.7)$ and its maximum is $\max_{(x,y)\in\Omega} |u_f(x,y)| = 100$.

5.2 Process Time Discussion of MIQP Solvings

In this section, we present the numerical performance of the preprocessing schemes, where we eliminate the state variables prior to optimization. We illustrate CPU times in Figure 5.2, more detailed information can be found in Table C.1. The CPU time of the MIQP solver CPLEX is limited to 24 hours for our experiments. Within this time limit, we are not able to obtain a solution of any of the problems with our parameter set and the used compute server. However, it is possible to compute solutions in less than one day with the preprocessing schemes developed in Chapter 3:

- PDE Elimination Simple
- PDE Elimination with Convolution

We expect the latter to perform better but regard both since the simple version can be applied for a larger class of problems. Furthermore, we extended the approach with convolution by doing first PDE Elimination and second rounding (Maximum-Sum-Up-Rounding) in order to reduce the computational time. On the basis of the (feasible) rounded solution we call CPLEX, we abbreviate the approach with "Hot Start".

The total CPU time and the time consumed by CPLEX only is visualized in Figure 5.2, left for the Pl problem, right for the PlOp problem. Particularly noticeable is that CPLEX terminates without a solution for $T_c = 32$ for the Pl problem. The error code is associated to out-of-memory errors which indicates that the search-tree grows too large.

Equally noticeable is that the CPU time for both problems does not increase significantly with growing fineness of the control discretization. This effect is due to preponderance of preprocessing in terms of process time, the order of magnitude is two orders higher compared to the time consumed by CPLEX. Furthermore, the effort of preprocessing is independent of the choice of the control grid. However, the increase of time consumption of CPLEX for fine control meshes is more significant, c.f. second row of Figure 5.2.

Comparing the different approaches, it is obvious that the total time consumption varies in particular between the simple PDE Elimination approach and the approaches including convolution. The effect arises since the number of linear equation systems solved prior to optimization is about 32 times higher for the simple PDE Elimination.

Concerning the approaches with convolution, rounding prior optimization reduced the CPU time further. Nevertheless, this benefit is less (or not) visible in the second row where CPLEX CPU times are compared. Especially for the Pl problem, all approaches perform roughly equally good regarding the time consumed by CPLEX. Indeed for this problem, the search-tree for the simple PDE Elimination is smaller than for the approaches including convolution, but the computational costs of a node varies between the approaches. We assume that this is due to the larger amount of data in the objective in the simple PDE Elimination case which makes objective evaluation more expensive.

This observation also explains why the optimization for PDE Elimination Simple for the PlOp problem takes longer than for the approaches with convolution, even though that the search-tree are roughly of the same size.

Besides the major differences of the approaches with respect to the process time, we want to point to differences occurring in the objective values for the PlOp problem. De-

pending on the chosen approach, the optimal objective values differ about the order of magnitude of 10^{-2}. This arises from the symmetry of the initial conditions of the PlOp problem in combination with the (abs-)migap tolerance in CPLEX. In case the objective of a current integer feasible solution differs less than this threshold to the objective values of the remaining nodes, CPLEX stops with the current solution. Therefore, we obtain with the different approaches varying solutions which are optimal with respect to a threshold.

Finally, we mention that rounding prior to optimization without PDE Elimination is not discussed in this thesis. However, it can be a promising strategy for problems with nonlinear PDEs where the presented PDE Elimination approaches are not possible to apply.

5.3 Process Time and Solution Quality of Heuristics

In this section, we present the numerical results of experiments with the rounding based heuristics introduced in Chapter 4. First, we compare the simple versions of the rounding approaches for instances of which we know an optimal solution. Second, the results of Spatial-Rounding is discussed. Third, we consider potential benefits of resolving between rounding steps. Finally, we analyze the behavior of Sum-Up-Rounding and Maximum-Sum-Up-Rounding for fine time discretizations.

5.3.1 Rounding simple

Similar to the previous section, we visualize the processing time of the rounding approaches in the first row of Figure 5.3, for detailed information we refer to Table C.2. Therefore, we average the processing time of the three main types of rounding with and without prior PDE Elimination:

- Maximum-Rounding
- Sum-Up-Rounding for MIPDECO
- Maximum-Sum-Up-Rounding

In the first row of Figure 5.3, these averages are compared with the processing time of CPLEX with prior PDE Elimination with Convolution, right for the problem Pl, left for PlOp, respectively. We describe "Rounding accumulated“ at the end of this paragraph.

Especially noticeable is that the generic application of rounding performs worse than with prior PDE Elimination. The main effort of the simple roundings is to solve two NLPs: one prior and one after rounding. Since these NLPs are high dimensional, the computational cost of PDE Elimination with Convolution is worth to be applied.

However we observe that the application of CPLEX, which certifies optimality, is cheaper than rounding in most cases. But a feasible solution can be computed for the instance with $T_c = 32$ of Pl, for which CPLEX runs out of memory.

In this context, we observed that the consumed time by PDE Elimination varies for identical instances which makes it difficult to compare processing times within the same order of magnitude. We trace this effect back to possible diverging utilization rates of the

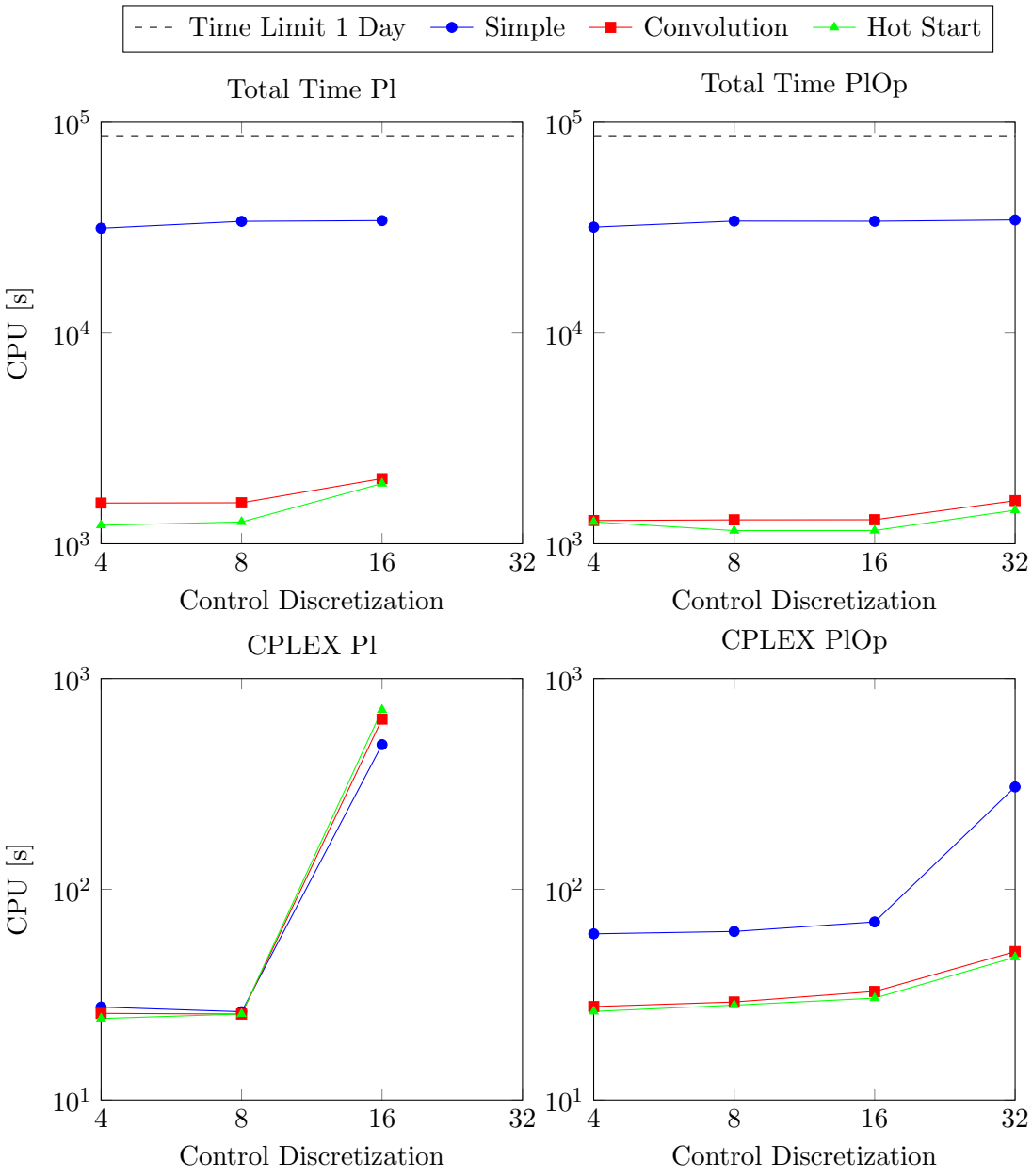

Figure 5.2: CPU time in seconds for PDE Elimination Simple, PDE Elimination with Convolution, and Hot Start CPLEX

used server during the computations.

Since the solution quality of different heuristics may vary widely and is often not apriori estimable, it is reasonable to apply multiple heuristics. This can be done in parallel on suitable machines.

For the presented problems, it is reasonable to apply first PDE Elimination once and then to do the heuristics on basis of the reduced model. Since PDE Elimination usually dominates the total processing time, a significant amount of time can be saved.

Therefore we plotted the minimum CPU time observed for PDE Elimination of an instance plus the sum of the consumed time for each of the three rounding approaches. In Figure 5.3, it can be observed that this accumulated approach performs the best in terms of time consumption.

Besides processing time, solution quality is crucial for heuristics. Therefore we computed a relative optimality gap for the solutions obtained by the three rounding schmemes. We do so by computing the difference to the optimal solution value and norming it by dividing by the optimal solution value.

In the second row of Figure 5.3, the relative gaps are plotted, left for Pl and right for PlOp. Particularly noticeable is that Maximum-Rounding and Sum-Up-Rounding for MIPDECO perform poorly compared to Maximum-SUR. For the Pl problem, we observe an optimality gap for the SUR approach of about 100%. This indicates that the manner how the ideas of SUR for ODEs are transfered to the MIPDECO case is questionable. In particular, we trace the effect back to the coarse meshes used and the manner how the source budget constraint is implemented. The performance of the Maximum-Rounding may be caused by the fact of not considering any information of other time steps while rounding. However, since we merged advantages of both rounding schemes when developed Maximum-SUR, we obtain high quality solutions with a gap less than 1% for both problems.

Furthermore, we mention that uncovered that solutions of a rounding approach vary dependent if PDE Elimination is applied prior to rounding. The difference of the objective values vary up to the order of magnitude of 10^1. This effect is similar to what we observed for the CPLEX solutions in the previous section. As the effect particularly appears for the PlOp problem, we assume that the symmetry of its initial values in combination with properties of the rounding approaches cause the behavior. The Maximum-Rounding choses the location(s) of the control taking the largest absolute value, the processing orders of the location depends on the L_1 norm of the corresponding controls for SUR for MIPDECO, and the Maximum-SUR choses the location with the largest rounding residual with respect to the fractional control values. To sum up, all three rounding heuristics make decisions upon the values of the controls. So if, due to the symmetry, controls are (almost) equal but vary due to different methods of calculations, different decisions are made which then result differences regarding the objective values.

We conclude that for the regarded instances CPLEX and the rounding based heuristics perform equally good in terms of CPU time if PDE Elimination is applied prior to solving. Nevertheless, we remark that for other instances, e.g. including nonlinearities of the PDE or for finer meshes, rounding approaches can be powerful tools to obtain meaningful

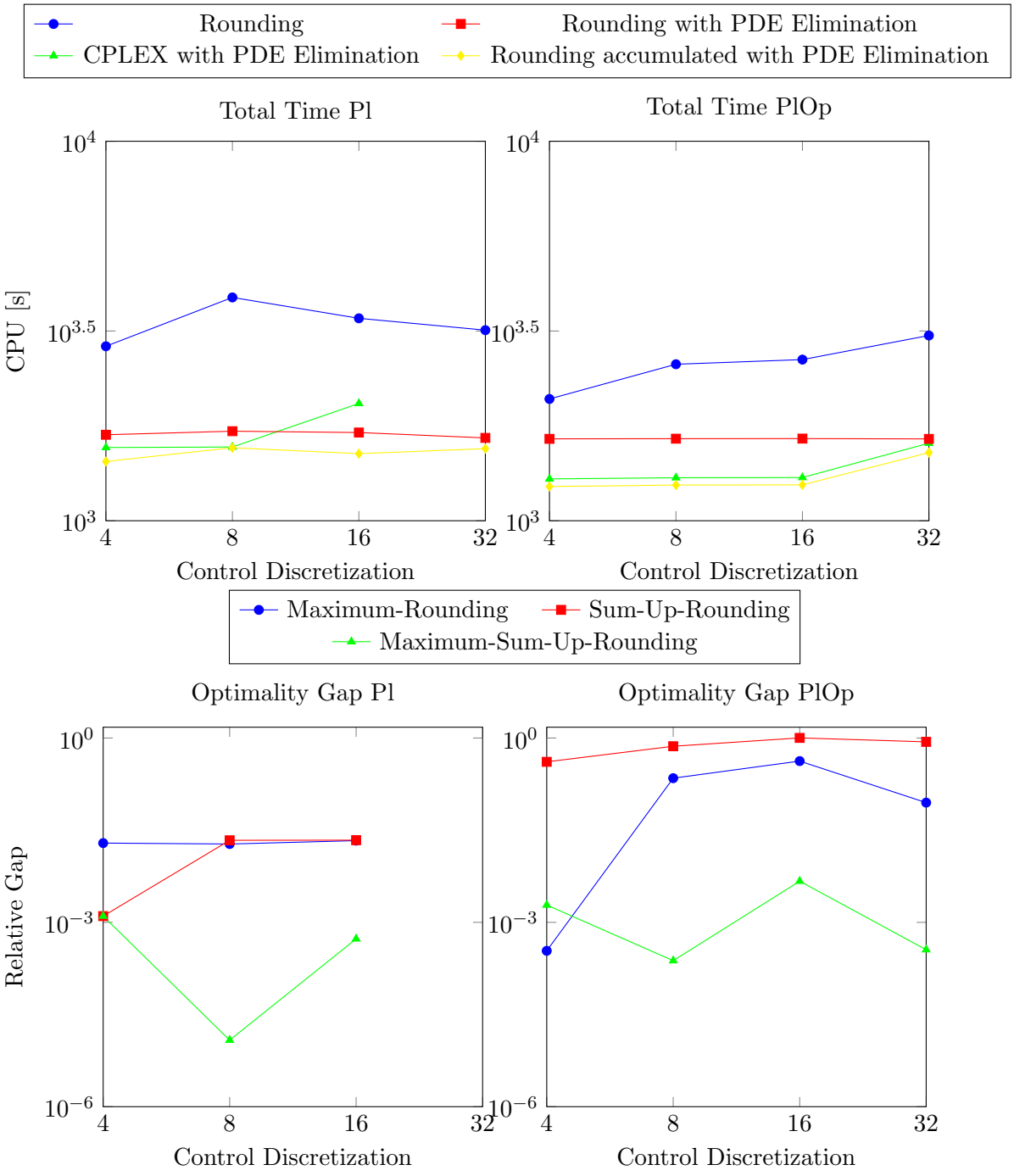

Figure 5.3: Rounding

solutions.

5.3.2 Spatial-Rounding

Spatial-Rounding manipulates the control values of the NLP relaxation of the problem prior to the actual application of the rounding schemes. Its goal is to further improve the heuristics by considering spatial information.

Spatial-Rounding keeps or reduces the optimality gap in 50% of the instances compared to the gap obtained with the same rounding approach without prior Spatial-Rounding. This performance lags behind our expectations since for the other half of instances we increased the optimality gap. Especially for Maximum-Sum-Up-Rounding, we observe deteriorations. However, Maximum-Sum-Up-Rounding generally performs best regarding the solution quality, so further improvements are potentially more challenging to achieve. Detailed information of experiments with Spatial Rounding can be found in Table C.3.

Nevertheless, the ideas of Spacial-Rounding may be more beneficial in different settings within MIPDECO and therefore should be considered for other instances.

5.3.3 Resolving

The term resolving refers to the optional solving fo an NLP relaxation of the problem after every rounding step. This means instead of solving two NLPs in total for the simple versions of the rounding approaches, we solve during the rounding additional NLPs. The structure of the problem Pl does not allow this approach since after fixing a large amount of controls, the NLP solver reaches the chosen maximum CPU time limit without solution. We visualize the computational effort of resolving for PlOp in Figure 5.4, for more detailed information, we refer to Table C.4.

The graph referring to Maximum-Sum-Up-Rounding and Maximum-Sum-Up-Rounding are parallel and increase with finer control discretizations. The parallelism can be explained with the effect, we discussed in previous sections, that the processing time of the PDE Elimination varies, here by about $4 \cdot 10^2$. This increase is due to the fact that we proceed forward in time and resolve after every time step for both approaches.

Thus more NLPs are solved for more time steps. In contrast the process time of Sum-Up-Rounding stays rather constant because we resolve after every proceeded actuator location which yields a constant amount of NLP solvings.

However, the resolve versions of the three rounding schemes are slower than the application of CPLEX with prior PDE Elimination with Convolution but they are signifivcaltly faster than CPLEX with PDE Elimination Simple (not plotted in this Figure).

In terms of solution quality, Maximum-Rounding and Maximum-Sum-Up-Rounding perform excellently since this rounding return an optimal solution of the problem PlOp. Also, the optimality gap of Sum-Up-Rounding for MIPDECO is halved for our instances. However, we remark that there are no theoretical guarantees for this improvement. Nevertheless, for settings where PDE Elimination cannot be applied, we have a high-potential tool which does not require to save a large search-tree during computation.

5.3.4 Rounding for Fine Time Discretizations

For this experiment, we solve the NLP relaxation for the (coarse) time discretization $T_c = T_n = 32$. Then the fractional solution is transfered to a finer mesh ($T_c = T_n = 64, 128, 256$)

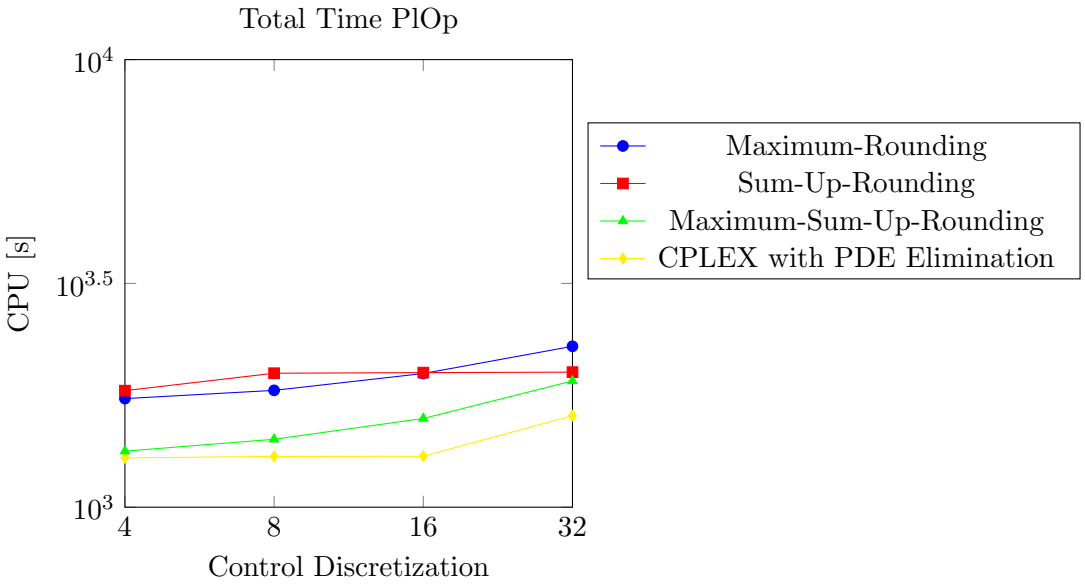

Figure 5.4: Resolve

where one of the rounding schemes is applied. In order to maintain comparability, we evaluate the objective of these rounded solutions on the finest mesh ($T_c = T_n = 256$). Since objective evaluation after rounding for the PlOp problem includes the solving of an NLP, this experiment is done for the Pl problem only. If Maximum-Rounding is applied, the experiment does not affect the objective value. Therefore, this heuristic is not regarded here. We illustrated the objective values for Sum-Up-Rounding and Maximum-Sum-Up-Rounding in Figure 5.5, detailed information is summarized in Table C.5.

Prior to rounding, we solve a (coarse) NLP relaxation and after solving a large linear system of equations which comes from a finer discretization. For both roundings, we observe an asymptotic decrease of the objective value if we round on finer meshes. This effect has a great potential since this manner of applying the heuristics is relatively cheap. Especially because the NLP relaxation of the finer discretization is not computable in our setting.

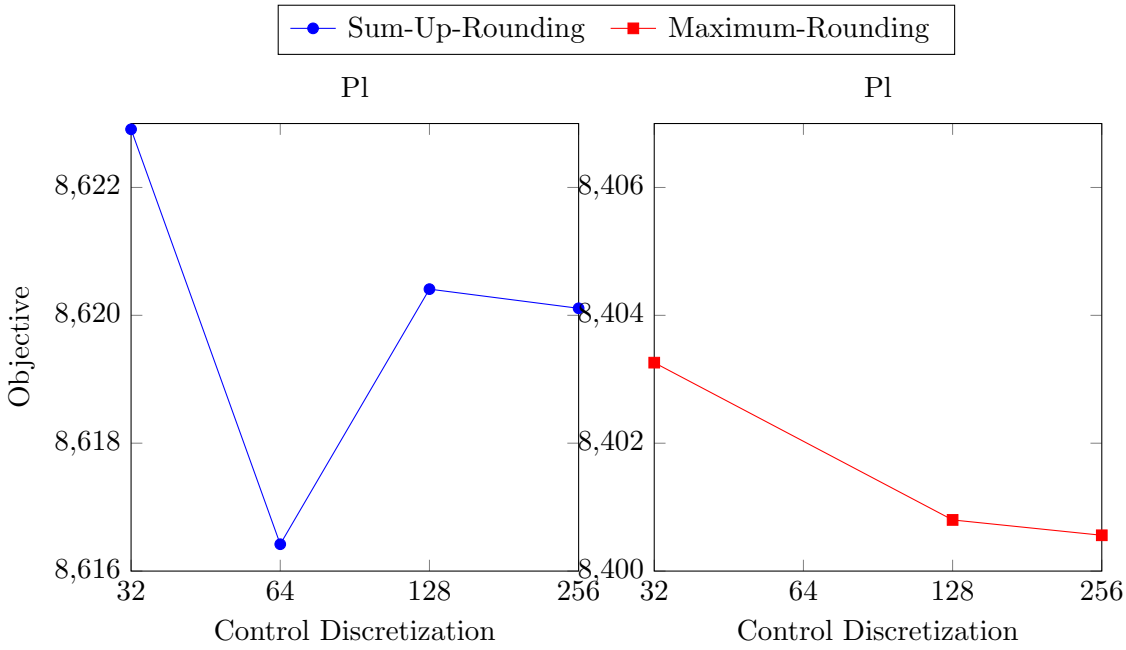

Figure 5.5: Objective Values Fine Mesh

Conclusion

In Chapter 1, we defined the class of mixed-integer PDE constrained optimization and introduced a subclass which is governed by the heat equation. Then we revised methods which allow to formulate a MIPDECO as a mixed-integer nonlinear program (MINLP). Furthermore, we summarized briefly algorithms which are applicable for MINLPs. In Chapter 2, two MIPDECO test problems are presented and formulated as MINLPs for which we developed solving approaches in this thesis. In Chapter 3, we developed a preprocessing algorithm which allows to significantly reduce the high number of variables of the MINLP formulation arising from the PDE discretization. In Chapter 4, we introduced a general rounding scheme for MIPDECO, three rounding based heuristics and an extension for these heuristics which considers spatial information. Furthermore, we proofed that the presented heuristics always return feasible solutions for the studied problems. In Chapter 5, we discussed the developed approaches regarding process time and solution quality.

Already early experiments demonstrated the high numerical complexity of even the rather simple problems discussed in this thesis. We are limited to very coarse PDE discretizations and even these instances could not be solved straight forward by an suitable solver within reasonable time on a server. However, we were able to demonstrate that the MINLP formulations of our test problems have a particular structure. Exploiting this structure, the major amount of variables can be eliminated. MINLP solvers are accelerated with this preprocessing scheme and the developed rounding schemes. We observed that the preprocessing makes MINLP solvings about as fast as two NLP solvings of an instance. However, meaningful solutions are obtained for instances where CPLEX failed with the developed rounding approaches. An optimality gap of less than 1% is reached with the simple versions. We even achieve optimality for the expensive resolve versions-without the necessity of saving a large search-trees. Indeed, for the instances where we could get a certified optimal solution, the rounding approaches performed only about equally fast, but the accelerating preprocessing scheme PDE Elimination is not applicable for the general class of MIPDECO. Therefore, we are proud to demonstrate that rounding can be a promising tool to overcome the challenges of MIPDECO.

However, there are a lot of open research questions. It is crucial to study theoretical background of this new problem class whereby theoretical knowledge of MINLP and PDE constrained optimization might be transferred to MIPDECO. This is of great importance since the variety of applications of MIPDECO is vast since physical models with discrete decision can be well represented. Therefore, important environmental problems could be modeled as MIPDECO, for example remediation of contaminated sites or the design of high-efficient solar cells [15]. With these applications, MIPDECO has potential to support the transition to the renewable energy age.

Appendix A

Objectives

A.1 MINLP Formulation

Heat Equation with Actuator Placement

$$J_1(u) = h_x h_y \sum_{i=1}^{N-1}\sum_{j=1}^{M-1}\left(u_{Tn,i,j} - u_f(ih_x, jh_y)\right)^2 + 2\left(\frac{1}{2}h_x h_y h_t \sum_{i=1}^{N-1}\sum_{j=1}^{M-1} u_{0,i,j}^2 + h_x h_y h_t \sum_{k=1}^{T_n-1}\sum_{i=1}^{N-1}\sum_{j=1}^{M-1} u_{k,i,j}^2 + \frac{1}{2}h_x h_y h_t \sum_{i=1}^{N-1}\sum_{j=1}^{M-1} u_{Tn,i,j}^2\right) \tag{A.1}$$

Heat Equation with Actuator Placement and Operation

$$J_1(u,v) = h_x h_y \sum_{i=1}^{N-1}\sum_{j=1}^{M-1}\left(u_{Tn,i,j} - u_f(ih_x, jh_y)\right)^2 + 2\left(\frac{1}{2}h_x h_y h_t \sum_{i=1}^{N-1}\sum_{j=1}^{M-1} u_{0,i,j}^2 + h_x h_y h_t \sum_{k=1}^{T_n-1}\sum_{i=1}^{N-1}\sum_{j=1}^{M-1} u_{k,i,j}^2 + \frac{1}{2}h_x h_y h_t \sum_{i=1}^{N-1}\sum_{j=1}^{M-1} u_{Tn,i,j}^2\right) + \frac{h_t}{500}\sum_{l=1}^{L}\left(\frac{1}{2}v_{0,l}^2 + \sum_{k=1}^{T_n-1} v_{k,l}^2 + \frac{1}{2}v_{0,l}^2\right) \tag{A.2}$$

A.2 PDE Elimination Simple

Heat Equation with Actuator Placement

$$
\begin{aligned}
J_3(w) = h_x h_y \sum_{i=1}^{N-1}\sum_{j=1}^{M-1}\left(\bar{u}^h_{Tn,i,j} + \sum_{t=1}^{T_n}\sum_{l=1}^{L} w_{t,l}\ \bar{u}^{t,l}_{T_n,i,j} - u_f(ih_x, jh_y)\right)^2 \\
+ 2\left(\frac{1}{2}h_x h_y h_t \sum_{i=1}^{N-1}\sum_{j=1}^{M-1}\left(\bar{u}^h_{0,i,j}\right)^2\right. \\
+ h_x h_y h_t \sum_{k=1}^{T_n-1}\sum_{i=1}^{N-1}\sum_{j=1}^{M-1}\left(\bar{u}^h_{k,i,j} + \sum_{t=1}^{T_n}\sum_{l=1}^{L} w_{t,l}\ \bar{u}^{t,l}_{k,i,j}\right)^2 \\
\left.+\frac{1}{2}h_x h_y h_t \sum_{i=1}^{N-1}\sum_{j=1}^{M-1}\left(\bar{u}^h_{Tn,i,j} + \sum_{t=1}^{T_n}\sum_{l=1}^{L} w_{t,l}\ \bar{u}^{t,l}_{T_n,i,j}\right)^2\right)
\end{aligned}
\tag{A.3}
$$

Heat Equation with Actuator Placement and Operation

$$
\begin{aligned}
J_4(v) =& h_x h_y \sum_{i=1}^{N-1}\sum_{j=1}^{M-1}\left(\bar{u}^h_{Tn,i,j} + \sum_{t=1}^{T_n}\sum_{l=1}^{L} v_{t,l}\ \bar{u}^{t,l}_{Tn,i,j}\right)^2 \\
&+2\left(\frac{1}{2}h_x h_y h_t \sum_{i=1}^{N-1}\sum_{j=1}^{M-1}\left(\bar{u}^h_{0,i,j}\right)^2\right. \\
&\quad + h_x h_y h_t \sum_{k=1}^{T_n-1}\sum_{i=1}^{N-1}\sum_{j=1}^{M-1}\left(\bar{u}^h_{k,i,j} + \sum_{t=1}^{T_n}\sum_{l=1}^{L} v_{t,l}\ \bar{u}^{t,l}_{k,i,j}\right)^2 \\
&\quad \left.+\frac{1}{2}h_x h_y h_t \sum_{i=1}^{N-1}\sum_{j=1}^{M-1}\left(\bar{u}^h_{Tn,i,j} + \sum_{t=1}^{T_n}\sum_{l=1}^{L} v_{t,l}\ \bar{u}^{t,l}_{Tn,i,j}\right)^2\right) \\
&+\sum_{l=1}^{L}\left(\frac{1}{2}\left(v_{0,l}\right)^2 + \sum_{t=1}^{T_n}\left(v_{t,l}\right)^2 + \frac{1}{2}\left(v_{Tn,l}\right)^2\right)
\end{aligned}
\tag{A.4}
$$

A.3 PDE Elimination with Convolution

Heat Equation with Actuator Placement

$$
\begin{aligned}
J_5(w) =& h_x h_y \sum_{i=1}^{N-1}\sum_{j=1}^{M-1}\left(\bar{u}^h_{Tn,i,j} + \sum_{t=1}^{T_n}\sum_{l=1}^{L} w_{t,l}\ \bar{u}^l_{Tn-t,i,j} - u_f(ih_x, jh_y)\right)^2 \\
&+ 2\left(\frac{1}{2}h_x h_y h_t \sum_{i=1}^{N-1}\sum_{j=1}^{M-1}\left(\bar{u}^h_{0,i,j}\right)^2\right. \\
&+ h_x h_y h_t \sum_{k=1}^{T_n-1}\sum_{i=1}^{N-1}\sum_{j=1}^{M-1}\left(\bar{u}^h_{k,i,j} + \sum_{t=1}^{k}\sum_{l=1}^{L} w_{t,l}\ \bar{u}^l_{k-t,i,j}\right)^2 \\
&\left.+\frac{1}{2}h_x h_y h_t \sum_{i=1}^{N-1}\sum_{j=1}^{M-1}\left(\bar{u}^h_{Tn,i,j} + \sum_{t=1}^{T_n}\sum_{l=1}^{L} w_{t,l}\ \bar{u}^l_{T_n-t,i,j}\right)^2\right)
\end{aligned}
\tag{A.5}
$$

A.3.1 Heat Equation with Actuator Placement and Operation

$$
\begin{aligned}
J_6(v) =& h_x h_y \sum_{i=1}^{N-1}\sum_{j=1}^{M-1}\left(\bar{u}^h_{Tn,i,j} + \sum_{t=1}^{T_n}\sum_{l=1}^{L} v_{t,l}\ \bar{u}^l_{Tn-t,i,j} - u_f(ih_x, jh_y)\right)^2 \\
&+ 2\left(\frac{1}{2}h_x h_y h_t \sum_{i=1}^{N-1}\sum_{j=1}^{M-1}\left(\bar{u}^h_{0,i,j}\right)^2\right. \\
&+ h_x h_y h_t \sum_{k=1}^{T_n-1}\sum_{i=1}^{N-1}\sum_{j=1}^{M-1}\left(\bar{u}^h_{k,i,j} + \sum_{t=1}^{k}\sum_{l=1}^{L} v_{t,l}\ \bar{u}^l_{k-t,i,j}\right)^2 \\
&\left.+\frac{1}{2}h_x h_y h_t \sum_{i=1}^{N-1}\sum_{j=1}^{M-1}\left(\bar{u}^h_{Tn,i,j} + \sum_{t=1}^{T_n}\sum_{l=1}^{L} v_{t,l}\ \bar{u}^l_{T_n-t,i,j}\right)^2\right) \\
&+ \sum_{l=1}^{L}\left(\frac{1}{2}\left(v_{0,l}\right)^2 + \sum_{t=1}^{T_n}\left(v_{t,l}\right)^2 + \frac{1}{2}\left(v_{Tn,l}\right)^2\right)
\end{aligned}
\tag{A.6}
$$

Appendix B

Counter Example for Spatial Rounding

In Chapter 4, we introduced Spatial-Rounding. Thereby, we scaled the variables after the modification to prevent the violation of the source budget constraint. Here, we want to present a counter example to demonstrate its necessity.

Let $W = 1$, $L = 9$, and $\sigma_2 = 1$, the we compute new controls by the following scheme:

$$\hat{w}_{k,l} = \sigma_1 w_{k,l} + (1 - \sigma_1) \frac{1}{|\mathcal{N}^1(l)|} \sum_{\bar{l} \mathcal{N}^1(l)} w_{k,\bar{l}}$$

Then we assume that a solution of the NLP relaxation for a fix time step is:

$$\begin{aligned} &w_{k,1} = w_{k,3} = w_{k,4} = w_{k,5} = w_{k,6} = w_{k,7} = w_{k,8} = w_{k,9} = 0.1 \\ &w_{k,2} = 0.2 \end{aligned}$$

Thus, the source budget constraint (2.1e) holds:

$$\sum_{l=1}^{9} = w_{k,l} = 1$$

On this control values, we apply Spatial-Rounding:

$$\begin{aligned} \hat{w}_{k,1} &= 0.1\sigma_1 + (1 - \sigma_1)(0.1 + 0.2)/2 = -0.05\sigma_1 + 0.15 \\ &= \hat{w}_{k,3} \\ \hat{w}_{k,2} &= 0.2\sigma_1 + (1 - \sigma_1)(0.1 + 0.1 + 0.1)/3 = 0.1\sigma_1 + 0.1 \\ \hat{w}_{k,4} &= 0.1\sigma_1 + (1 - \sigma_1)(0.1 + 0.1 + 0.1)/3 = 0.1 \\ &= \hat{w}_{k,6} = \hat{w}_{k,8} \\ \hat{w}_{k,5} &= 0.1\sigma_1 + (1 - \sigma_1)(0.1 + 0.1 + 0.2 + 0.1)/4 = -0.025\sigma_1 + 0.125 \\ \hat{w}_{k,7} &= 0.1\sigma_1 + (1 - \sigma_1)(0.1 + 0.1)/2 = 0.1 \end{aligned}$$

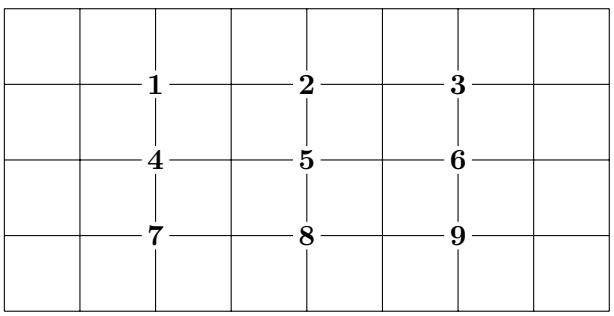

Figure B.1: Domain Ω with actuator locations

The sum of the $\hat{w}$ is then not equal 1, unless $\sigma_1 = 1$ which would mean that the control values would not be modified.

$$\begin{aligned}\sum_{l=1}^{9} \hat{w}_{k,l} =& 5 \cdot 0.1 + 2 \cdot (-0.05\sigma_1 + 0.15) + 0.1\sigma_1 + 0.1 - 0.025\sigma_1 + 0.125 \\ =& 1.025 - 0.025\sigma_1\end{aligned}$$

With this counter example, we illustrated the need of norming the control values after applying Spatial-Rounding.

Appendix C

Tables

Instance	Objective	BnB Nodes	CPU Presolve	CPU Rounding	CPU CPLEX	CPU Total
cplex-D-IE-1-10-32-32-04-u-0-2-0-presolved-Pl	8773	0.00e+00	1.53e+03		2.58e+01	1.56e+03
cplex-D-IE-1-10-32-32-08-u-0-2-0-presolved-Pl	8708	0.00e+00	1.54e+03		2.56e+01	1.56e+03
cplex-D-IE-1-10-32-32-16-u-0-2-0-presolved-Pl	8696	3.84e+05	1.40e+03		6.42e+02	2.04e+03
cplex-D-IE-1-10-32-32-32-u-0-2-0-presolved-Pl	code 9	6.51e+05				
cplex-D-IE-1-10-32-32-04-u-0-2-0-presolved1-Pl	8773	0.00e+00	3.14e+04		2.76e+01	3.15e+04
cplex-D-IE-1-10-32-32-08-u-0-2-0-presolved1-Pl	8708	0.00e+00	3.39e+04		2.62e+01	3.39e+04
cplex-D-IE-1-10-32-32-16-u-0-2-0-presolved1-Pl	8696	2.66e+05	3.37e+04		4.86e+02	3.42e+04
cplex-D-IE-1-10-32-32-32-u-0-2-0-presolved1-Pl	code 9	2.21e+05	3.39e+04		1.21e+02	
cplex-Rounding3aPl-D-IE-1-10-32-32-04-u-0-2-0-presolved-Pl	8773	0.00e+00	1.09e+03	1.09e+02	2.44e+01	1.22e+03
cplex-Rounding3aPl-D-IE-1-10-32-32-08-u-0-2-0-presolved-Pl	8708	0.00e+00	1.10e+03	1.46e+02	2.56e+01	1.27e+03
cplex-Rounding3aPl-D-IE-1-10-32-32-16-u-0-2-0-presolved-Pl	8696	3.91e+05	1.09e+03	1.31e+02	7.11e+02	1.93e+03
cplex-D-IE-1-10-32-32-04-u-0-1-0-presolved-PlOp	14384	5.90e+01	1.26e+03		2.78e+01	1.29e+03
cplex-D-IE-1-10-32-32-08-u-0-1-0-presolved-PlOp	10548	1.75e+03	1.27e+03		2.92e+01	1.30e+03
cplex-D-IE-1-10-32-32-16-u-0-1-0-presolved-PlOp	7776	4.96e+03	1.27e+03		3.27e+01	1.30e+03
cplex-D-IE-1-10-32-32-32-u-0-1-0-presolved-PlOp	5989	3.53e+04	1.55e+03		5.07e+01	1.60e+03
cplex-D-IE-1-10-32-32-04-u-0-1-0-presolved1-PlOp	14384	5.80e+01	3.18e+04		6.15e+01	3.18e+04
cplex-D-IE-1-10-32-32-08-u-0-1-0-presolved1-PlOp	10548	1.88e+03	3.39e+04		6.31e+01	3.40e+04
cplex-D-IE-1-10-32-32-16-u-0-1-0-presolved1-PlOp	7776	6.86e+03	3.39e+04		7.00e+01	3.39e+04
cplex-D-IE-1-10-32-32-32-u-0-1-0-presolved1-PlOp	5989	2.71e+04	3.41e+04		3.06e+02	3.44e+04
cplex-Rounding3aPlOp-D-IE-1-10-32-32-04-u-0-1-0-presolved-PlOp	14384	6.30e+01	1.05e+03	1.89e+02	2.63e+01	1.27e+03
cplex-Rounding3aPlOp-D-IE-1-10-32-32-08-u-0-1-0-presolved-PlOp	10548	1.49e+03	1.07e+03	5.99e+01	2.82e+01	1.15e+03
cplex-Rounding3aPlOp-D-IE-1-10-32-32-16-u-0-1-0-presolved-PlOp	7777	3.43e+03	1.07e+03	5.71e+01	3.04e+01	1.16e+03
cplex-Rounding3aPlOp-D-IE-1-10-32-32-32-u-0-1-0-presolved-PlOp	5989	2.37e+04	1.34e+03	5.75e+01	4.76e+01	1.44e+03

Table C.1: Process Time Discussion of MIQP Solvings

Instance	Objective	Optimality Gap	CPU Solve	CPU Rounding	CpU Presolve	CPU Total
Rounding1aPl-D-IE-1-10-32-32-04-u-0-2-0-presolved-Pl	8944	1.95e-02		1.14e+02	1.57e+03	1.69e+03
Rounding1aPl-D-IE-1-10-32-32-08-u-0-2-0-presolved-Pl	8872	1.88e-02		1.54e+02	1.57e+03	1.72e+03
Rounding1aPl-D-IE-1-10-32-32-16-u-0-2-0-presolved-Pl	8881	2.14e-02		1.38e+02	1.57e+03	1.71e+03
Rounding1aPl-D-IE-1-10-32-32-32-u-0-2-0-presolved-Pl	8848	-		1.54e+02	1.36e+03	1.51e+03
Rounding2aPl-D-IE-1-10-32-32-04-u-0-2-0-presolved-Pl	8784	1.26e-03		1.14e+02	1.56e+03	1.68e+03
Rounding2aPl-D-IE-1-10-32-32-08-u-0-2-0-presolved-Pl	8896	2.16e-02		1.53e+02	1.57e+03	1.72e+03
Rounding2aPl-D-IE-1-10-32-32-16-u-0-2-0-presolved-Pl	8885	2.17e-02		1.38e+02	1.57e+03	1.71e+03
Rounding2aPl-D-IE-1-10-32-32-32-u-0-2-0-presolved-Pl	8916	-		1.53e+02	1.57e+03	1.72e+03
Rounding3aPl-D-IE-1-10-32-32-04-u-0-2-0-presolved-Pl	8784	1.26e-03		1.14e+02	1.57e+03	1.69e+03
Rounding3aPl-D-IE-1-10-32-32-08-u-0-2-0-presolved-Pl	8708	1.21e-05		1.53e+02	1.57e+03	1.72e+03
Rounding3aPl-D-IE-1-10-32-32-16-u-0-2-0-presolved-Pl	8700	5.38e-04		1.37e+02	1.57e+03	1.71e+03
Rounding3aPl-D-IE-1-10-32-32-32-u-0-2-0-presolved-Pl	8691			1.52e+02	1.57e+03	1.72e+03
Rounding1aPl-D-IE-1-10-32-32-04-u-0-2-0-Pl	8944	1.95e-02	1.65e+02	2.67e+03		2.83e+03
Rounding1aPl-D-IE-1-10-32-32-08-u-0-2-0-Pl	8872	1.88e-02	2.03e+02	3.67e+03		3.87e+03
Rounding1aPl-D-IE-1-10-32-32-16-u-0-2-0-Pl	8881	2.14e-02	2.03e+02	3.21e+03		3.41e+03
Rounding1aPl-D-IE-1-10-32-32-32-u-0-2-0-Pl	8848	-	1.64e+02	2.94e+03		3.11e+03
Rounding2aPl-D-IE-1-10-32-32-04-u-0-2-0-Pl	8784	1.26e-03	2.02e+02	2.70e+03		2.90e+03
Rounding2aPl-D-IE-1-10-32-32-08-u-0-2-0-Pl	8896	2.16e-02	2.01e+02	3.67e+03		3.87e+03
Rounding2aPl-D-IE-1-10-32-32-16-u-0-2-0-Pl	8885	2.17e-02	2.01e+02	3.21e+03		3.41e+03
Rounding2aPl-D-IE-1-10-32-32-32-u-0-2-0-Pl	8916	-	2.01e+02	3.01e+03		3.21e+03
Rounding3aPl-D-IE-1-10-32-32-04-u-0-2-0-Pl	8784	1.26e-03	2.01e+02	2.71e+03		2.91e+03
Rounding3aPl-D-IE-1-10-32-32-08-u-0-2-0-Pl	8708	1.21e-05	2.01e+02	3.68e+03		3.88e+03
Rounding3aPl-D-IE-1-10-32-32-16-u-0-2-0-Pl	8700	5.38e-04	2.01e+02	3.22e+03		3.43e+03
Rounding3aPl-D-IE-1-10-32-32-32-u-0-2-0-Pl	8691		2.02e+02	3.02e+03		3.22e+03
Rounding1aPlOp-D-IE-1-10-32-32-04-u-0-1-0-presolved-PlOp	14389	3.44e-04	1.14e+00	5.87e+01	1.58e+03	1.64e+03
Rounding1aPlOp-D-IE-1-10-32-32-08-u-0-1-0-presolved-PlOp	12889	2.22e-01	3.77e+00	5.85e+01	1.58e+03	1.65e+03
Rounding1aPlOp-D-IE-1-10-32-32-16-u-0-1-0-presolved-PlOp	11052	4.21e-01	3.47e+00	5.88e+01	1.58e+03	1.65e+03
Rounding1aPlOp-D-IE-1-10-32-32-32-u-0-1-0-presolved-PlOp	6521	8.88e-02	1.28e+00	5.86e+01	1.58e+03	1.64e+03
Rounding2aPlOp-D-IE-1-10-32-32-04-u-0-1-0-presolved-PlOp	20276	4.10e-01	3.48e+00	5.87e+01	1.58e+03	1.64e+03
Rounding2aPlOp-D-IE-1-10-32-32-08-u-0-1-0-presolved-PlOp	18293	7.34e-01	1.15e+00	5.86e+01	1.58e+03	1.64e+03
Rounding2aPlOp-D-IE-1-10-32-32-16-u-0-1-0-presolved-PlOp	15583	1.00e+00	3.32e+00	5.85e+01	1.58e+03	1.65e+03
Rounding2aPlOp-D-IE-1-10-32-32-32-u-0-1-0-presolved-PlOp	11176	8.66e-01	1.24e+00	5.86e+01	1.58e+03	1.64e+03
Rounding3aPlOp-D-IE-1-10-32-32-04-u-0-1-0-presolved-PlOp	14411	1.91e-03	3.47e+00	5.88e+01	1.59e+03	1.65e+03
Rounding3aPlOp-D-IE-1-10-32-32-08-u-0-1-0-presolved-PlOp	10550	2.38e-04	3.95e+00	5.87e+01	1.59e+03	1.65e+03
Rounding3aPlOp-D-IE-1-10-32-32-16-u-0-1-0-presolved-PlOp	7813	4.67e-03	1.20e+00	5.84e+01	1.59e+03	1.65e+03
Rounding3aPlOp-D-IE-1-10-32-32-32-u-0-1-0-presolved-PlOp	5991	3.58e-04	1.48e+00	5.87e+01	1.59e+03	1.65e+03
Rounding1aPlOp-D-IE-1-10-32-32-04-u-0-1-0-PlOp	14404	1.44e-03	6.91e+02	1.40e+03		2.09e+03
Rounding1aPlOp-D-IE-1-10-32-32-08-u-0-1-0-PlOp	15916	5.09e-01	7.54e+02	1.84e+03		2.59e+03
Rounding1aPlOp-D-IE-1-10-32-32-16-u-0-1-0-PlOp	9682	2.45e-01	1.85e+03	1.34e+03		3.19e+03
Rounding1aPlOp-D-IE-1-10-32-32-32-u-0-1-0-PlOp	9332	5.58e-01	2.36e+03	1.05e+03		3.41e+03
Rounding2aPlOp-D-IE-1-10-32-32-04-u-0-1-0-PlOp	20276	4.10e-01	6.93e+02	1.40e+03		2.09e+03
Rounding2aPlOp-D-IE-1-10-32-32-08-u-0-1-0-PlOp	18820	7.84e-01	6.94e+02	1.84e+03		2.53e+03
Rounding2aPlOp-D-IE-1-10-32-32-16-u-0-1-0-PlOp	14931	9.20e-01	1.11e+03	1.39e+03		2.50e+03
Rounding2aPlOp-D-IE-1-10-32-32-32-u-0-1-0-PlOp	11595	9.36e-01	1.23e+03	1.09e+03		2.32e+03
Rounding3aPlOp-D-IE-1-10-32-32-04-u-0-1-0-PlOp	14411	1.91e-03	6.98e+02	1.40e+03		2.10e+03
Rounding3aPlOp-D-IE-1-10-32-32-08-u-0-1-0-PlOp	10550	2.37e-04	7.91e+02	1.84e+03		2.63e+03
Rounding3aPlOp-D-IE-1-10-32-32-16-u-0-1-0-PlOp	7823	5.92e-03	8.93e+02	1.39e+03		2.29e+03
Rounding3aPlOp-D-IE-1-10-32-32-32-u-0-1-0-PlOp	6016	4.48e-03	2.42e+03	1.09e+03		3.51e+03

Table C.2: Rounding simple

Instance	Objective	Optimality Gap	CPU Solve	CPU Rounding	CPU Presolve	CPU Total
Rounding1aAPl-D-IE-1-10-32-32-04-u-0-2-0-presolved-Pl	8.94E+03	0.0194986656		113.962	1465.91	1579.872
Rounding1aAPl-D-IE-1-10-32-32-08-u-0-2-0-presolved-Pl	8.87E+03	0.0188292228		153.125	1568.4	1721.525
Rounding1aAPl-D-IE-1-10-32-32-16-u-0-2-0-presolved-Pl	8.87E+03	0.0204144769		137.011	1570.55	1707.561
Rounding1aAPl-D-IE-1-10-32-32-32-u-0-2-0-presolved-Pl	8.86E+03			152.76	1568.96	1721.72
Rounding2aAPl-D-IE-1-10-32-32-04-u-0-2-0-presolved-Pl	9.03E+03	0.0297964656		113.841	1570.93	1684.771
Rounding2aAPl-D-IE-1-10-32-32-08-u-0-2-0-presolved-Pl	9.19E+03	0.0551170019		164.351	1464.45	1628.801
Rounding2aAPl-D-IE-1-10-32-32-16-u-0-2-0-presolved-Pl	9.17E+03	0.0547076208		137.449	1239.17	1376.619
Rounding2aAPl-D-IE-1-10-32-32-32-u-0-2-0-presolved-Pl	9.18E+03			152.796	1567.73	1720.526
Rounding3aAPl-D-IE-1-10-32-32-04-u-0-2-0-presolved-Pl	8.82E+03	0.0048301655		113.967	1569.07	1683.037
Rounding3aAPl-D-IE-1-10-32-32-08-u-0-2-0-presolved-Pl	8.80E+03	0.0104963356		153.466	1492.15	1645.616
Rounding3aAPl-D-IE-1-10-32-32-16-u-0-2-0-presolved-Pl	8.80E+03	0.0121662831		137.769	1465.01	1602.779
Rounding3aAPl-D-IE-1-10-32-32-32-u-0-2-0-presolved-Pl	8.80E+03			152.954	1572.6	1725.554
Rounding1aAPl-D-IE-1-10-32-32-04-u-0-2-0-Pl	8.94E+03	0.0194986656	201.76	2701.7		2903.46
Rounding1aAPl-D-IE-1-10-32-32-08-u-0-2-0-Pl	8.87E+03	0.0188292227	202.391	3675.75		3878.141
Rounding1aAPl-D-IE-1-10-32-32-16-u-0-2-0-Pl	8.87E+03	0.0204144769	203.504	3217.57		3421.074
Rounding1aAPl-D-IE-1-10-32-32-32-u-0-2-0-Pl	8.86E+03		203.871	3015.41		3219.281
Rounding2aAPl-D-IE-1-10-32-32-04-u-0-2-0-Pl	9.03E+03	0.0297964656	203.858	2709.08		2912.938
Rounding2aAPl-D-IE-1-10-32-32-08-u-0-2-0-Pl	9.19E+03	0.055117002	203.319	3681.91		3885.229
Rounding2aAPl-D-IE-1-10-32-32-16-u-0-2-0-Pl	9.17E+03	0.0547076207	203.738	3221.2		3424.938
Rounding2aAPl-D-IE-1-10-32-32-32-u-0-2-0-Pl	9.18E+03		203.459	3016.43		3219.889
Rounding3aAPl-D-IE-1-10-32-32-04-u-0-2-0-Pl	8.82E+03	0.0048301655	204.697	2711.84		2916.537
Rounding3aAPl-D-IE-1-10-32-32-08-u-0-2-0-Pl	8.80E+03	0.0104963356	204.543	3685.58		3890.123
Rounding3aAPl-D-IE-1-10-32-32-16-u-0-2-0-Pl	8.80E+03	0.0121662831	204.342	3224.22		3428.562
Rounding3aAPl-D-IE-1-10-32-32-32-u-0-2-0-Pl	8.80E+03		204.649	3019.17		3223.819
Rounding1aAPlOp-D-IE-1-10-32-32-04-u-0-1-0-presolved-PlOp	1.65E+04	0.1463684597	3.35633	58.6119	1582.12	1644.08823
Rounding1aAPlOp-D-IE-1-10-32-32-08-u-0-1-0-presolved-PlOp	1.29E+04	0.2219628978	1.29768	58.5898	1581.12	1641.00748
Rounding1aAPlOp-D-IE-1-10-32-32-16-u-0-1-0-presolved-PlOp	1.11E+04	0.4212274441	3.67877	61.9094	1581.52	1647.10817
Rounding1aAPlOp-D-IE-1-10-32-32-32-u-0-1-0-presolved-PlOp	6.52E+03	0.0888075716	1.29766	62.1771	1244.11	1307.58476
Rounding2aAPlOp-D-IE-1-10-32-32-04-u-0-1-0-presolved-PlOp	2.03E+04	0.4096386851	3.3587	58.6372	1583.79	1645.7859
Rounding2aAPlOp-D-IE-1-10-32-32-08-u-0-1-0-presolved-PlOp	1.83E+04	0.7343443699	1.40865	58.6937	1583.98	1644.08235
Rounding2aAPlOp-D-IE-1-10-32-32-16-u-0-1-0-presolved-PlOp	1.56E+04	1.0038012571	3.32787	58.6042	1585.49	1647.42207
Rounding2aAPlOp-D-IE-1-10-32-32-32-u-0-1-0-presolved-PlOp	1.12E+04	0.8660999572	1.22792	58.3864	1584.49	1644.10432
Rounding3aAPlOp-D-IE-1-10-32-32-04-u-0-1-0-presolved-PlOp	1.44E+04	0.0017945273	3.41253	58.7206	1587.29	1649.42313
Rounding3aAPlOp-D-IE-1-10-32-32-08-u-0-1-0-presolved-PlOp	1.06E+04	0.000457695	1.352	58.5297	1584.99	1644.8717
Rounding3aAPlOp-D-IE-1-10-32-32-16-u-0-1-0-presolved-PlOp	7.80E+03	0.0032064308	1.19487	58.6522	1585.88	1645.72707
Rounding3aAPlOp-D-IE-1-10-32-32-32-u-0-1-0-presolved-PlOp	5.99E+03	0.000359708	1.20551	58.4669	1584.13	1643.80241
Rounding1aAPlOp-D-IE-1-10-32-32-04-u-0-1-0-PlOp	1.44E+04	0.00143756	691.497	1398.13		2089.627
Rounding1aAPlOp-D-IE-1-10-32-32-08-u-0-1-0-PlOp	1.59E+04	0.5089323728	804.993	1837.65		2642.643
Rounding1aAPlOp-D-IE-1-10-32-32-16-u-0-1-0-PlOp	9.68E+03	0.245289498	1968.93	1392.43		3361.36
Rounding1aAPlOp-D-IE-1-10-32-32-32-u-0-1-0-PlOp	9.33E+03	0.5580888285	2465.04	1046.98		3512.02
Rounding2aAPlOp-D-IE-1-10-32-32-04-u-0-1-0-PlOp	2.03E+04	0.4096386851	693.901	1400.73		2094.631
Rounding2aAPlOp-D-IE-1-10-32-32-08-u-0-1-0-PlOp	1.88E+04	0.7842437584	694.335	1838.83		2533.165
Rounding2aAPlOp-D-IE-1-10-32-32-16-u-0-1-0-PlOp	1.49E+04	0.9199646017	999.865	1392.32		2392.185
Rounding2aAPlOp-D-IE-1-10-32-32-32-u-0-1-0-PlOp	1.16E+04	0.9342532407	1224.9	1090.4		2315.3
Rounding3aAPlOp-D-IE-1-10-32-32-04-u-0-1-0-PlOp	1.44E+04	0.0019139003	698.306	1402.84		2101.146
Rounding3aAPlOp-D-IE-1-10-32-32-08-u-0-1-0-PlOp	1.06E+04	0.0010592601	792.878	1838.62		2631.498
Rounding3aAPlOp-D-IE-1-10-32-32-16-u-0-1-0-PlOp	7.82E+03	0.0059219024	1142.47	1338.61		2481.08
Rounding3aAPlOp-D-IE-1-10-32-32-32-u-0-1-0-PlOp	6.05E+03	0.0094063736	2657.43	1047.07		3704.5

Table C.3: Spatial Rounding

Instance	Objective	Optimality Gap	CPU Presolve	CPU Solve	CPU Rounding	CPU Total
Rounding1bPlOp-D-IE-1-10-32-32-04-u-0-1-0-PlOp	14388	2.69e-04		6.70e+02	5.85e+03	6.52e+03
Rounding1bPlOp-D-IE-1-10-32-32-08-u-0-1-0-PlOp	10552	3.85e-04		6.84e+02	1.29e+04	1.36e+04
Rounding1bPlOp-D-IE-1-10-32-32-16-u-0-1-0-PlOp	7777	3.60e-05		7.60e+02	2.91e+04	2.99e+04
Rounding1bPlOp-D-IE-1-10-32-32-32-u-0-1-0-PlOp	5989	-6.64e-06		2.25e+03	5.51e+04	5.74e+04
Rounding2bPlOp-D-IE-1-10-32-32-04-u-0-1-0-PlOp						
Rounding2bPlOp-D-IE-1-10-32-32-08-u-0-1-0-PlOp	19446	8.44e-01		6.64e+02	1.32e+04	1.39e+04
Rounding2bPlOp-D-IE-1-10-32-32-16-u-0-1-0-PlOp	14419	8.54e-01		1.16e+03	1.17e+04	1.28e+04
Rounding2bPlOp-D-IE-1-10-32-32-32-u-0-1-0-PlOp	9988	6.68e-01		2.18e+03	1.07e+04	1.29e+04
Rounding3bPlOp-D-IE-1-10-32-32-04-u-0-1-0-PlOp	14412	1.98e-03		6.61e+02	5.52e+03	6.18e+03
Rounding3bPlOp-D-IE-1-10-32-32-08-u-0-1-0-PlOp	10549	1.58e-04		6.57e+02	1.38e+04	1.44e+04
Rounding3bPlOp-D-IE-1-10-32-32-16-u-0-1-0-PlOp	7778	1.52e-04		7.74e+02	2.79e+04	2.86e+04
Rounding3bPlOp-D-IE-1-10-32-32-32-u-0-1-0-PlOp	5991	3.48e-04		1.87e+03	6.61e+04	6.80e+04
Rounding1bPlOp-D-IE-1-10-32-32-04-u-0-1-0-presolved-PlOp	14388	2.69e-04	1.64e+03	1.25e+00	1.08e+02	1.75e+03
Rounding1bPlOp-D-IE-1-10-32-32-08-u-0-1-0-presolved-PlOp	10552	3.85e-04	1.64e+03	1.26e+00	1.84e+02	1.82e+03
Rounding1bPlOp-D-IE-1-10-32-32-16-u-0-1-0-presolved-PlOp	7777	3.60e-05	1.64e+03	1.26e+00	3.47e+02	1.99e+03
Rounding1bPlOp-D-IE-1-10-32-32-32-u-0-1-0-presolved-PlOp	5989	-6.64e-06	1.64e+03	1.40e+00	6.47e+02	2.29e+03
Rounding2bPlOp-D-IE-1-10-32-32-04-u-0-1-0-presolved-PlOp	23329	6.22e-01	1.64e+03	1.82e+02		1.82e+03
Rounding2bPlOp-D-IE-1-10-32-32-08-u-0-1-0-presolved-PlOp	19516	8.50e-01	1.64e+03	3.54e+00	3.48e+02	1.99e+03
Rounding2bPlOp-D-IE-1-10-32-32-16-u-0-1-0-presolved-PlOp	14419	8.54e-01	1.64e+03	3.25e+00	3.56e+02	2.00e+03
Rounding2bPlOp-D-IE-1-10-32-32-32-u-0-1-0-presolved-PlOp	9988	6.68e-01	1.64e+03	4.06e+00	3.59e+02	2.00e+03
Rounding3bPlOp-D-IE-1-10-32-32-04-u-0-1-0-presolved-PlOp	14412	1.98e-03	1.22e+03	1.24e+00	1.13e+02	1.33e+03
Rounding3bPlOp-D-IE-1-10-32-32-08-u-0-1-0-presolved-PlOp	10549	1.58e-04	1.22e+03	3.53e+00	1.96e+02	1.42e+03
Rounding3bPlOp-D-IE-1-10-32-32-16-u-0-1-0-presolved-PlOp	7778	1.52e-04	1.22e+03	1.15e+00	3.59e+02	1.58e+03
Rounding3bPlOp-D-IE-1-10-32-32-32-u-0-1-0-presolved-PlOp	5991	3.48e-04	1.24e+03	1.12e+00	6.72e+02	1.91e+03

Table C.4: Resolving

Instance	Objective
D-IE-1-10-256-32-256-u-0-2-0- Rounding1aPl-on-D-IE-1-10-32-32-32-u-0-2-0 -of-ipopt-D-IE-1-10-32-32-32-u-0-2-0-Pl-EvalObj-Pl	8564
D-IE-1-10-256-32-256-u-0-2-0- Rounding1aPl-on-D-IE-1-10-64-32-64-u-0-2-0 -of-ipopt-D-IE-1-10-32-32-32-u-0-2-0-Pl-EvalObj-Pl	8564
D-IE-1-10-256-32-256-u-0-2-0- Rounding1aPl-on-D-IE-1-10-128-32-128-u-0-2-0 -of-ipopt-D-IE-1-10-32-32-32-u-0-2-0-Pl-EvalObj-Pl	8564
D-IE-1-10-256-32-256-u-0-2-0- Rounding1aPl-on-D-IE-1-10-256-32-256-u-0-2-0 -of-ipopt-D-IE-1-10-32-32-32-u-0-2-0-Pl-EvalObj-Pl	8564
D-IE-1-10-256-32-256-u-0-2-0- Rounding2aPl-on-D-IE-1-10-32-32-32-u-0-2-0 -of-ipopt-D-IE-1-10-32-32-32-u-0-2-0-Pl-EvalObj-Pl	8623
D-IE-1-10-256-32-256-u-0-2-0- Rounding2aPl-on-D-IE-1-10-64-32-64-u-0-2-0 -of-ipopt-D-IE-1-10-32-32-32-u-0-2-0-Pl-EvalObj-Pl	8616
D-IE-1-10-256-32-256-u-0-2-0- Rounding2aPl-on-D-IE-1-10-128-32-128-u-0-2-0 -of-ipopt-D-IE-1-10-32-32-32-u-0-2-0-Pl-EvalObj-Pl	8620
D-IE-1-10-256-32-256-u-0-2-0- Rounding2aPl-on-D-IE-1-10-256-32-256-u-0-2-0 -of-ipopt-D-IE-1-10-32-32-32-u-0-2-0-Pl-EvalObj-Pl	8620
D-IE-1-10-256-32-256-u-0-2-0- Rounding3aPl-on-D-IE-1-10-32-32-32-u-0-2-0 -of-ipopt-D-IE-1-10-32-32-32-u-0-2-0-Pl-EvalObj-Pl	8403
D-IE-1-10-256-32-256-u-0-2-0- Rounding3aPl-on-D-IE-1-10-64-32-64-u-0-2-0 -of-ipopt-D-IE-1-10-32-32-32-u-0-2-0-Pl-EvalObj-Pl	
D-IE-1-10-256-32-256-u-0-2-0- Rounding3aPl-on-D-IE-1-10-128-32-128-u-0-2-0 -of-ipopt-D-IE-1-10-32-32-32-u-0-2-0-Pl-EvalObj-Pl	8401
D-IE-1-10-256-32-256-u-0-2-0- Rounding3aPl-on-D-IE-1-10-256-32-256-u-0-2-0 -of-ipopt-D-IE-1-10-32-32-32-u-0-2-0-Pl-EvalObj-Pl	8401
D-IE-1-10-256-32-256-u-0-2-0- Rounding1aAPl-on-D-IE-1-10-32-32-32-u-0-2-0 -of-ipopt-D-IE-1-10-32-32-32-u-0-2-0-Pl-EvalObj-Pl	8579
D-IE-1-10-256-32-256-u-0-2-0- Rounding1aAPl-on-D-IE-1-10-64-32-64-u-0-2-0 -of-ipopt-D-IE-1-10-32-32-32-u-0-2-0-Pl-EvalObj-Pl	8579
D-IE-1-10-256-32-256-u-0-2-0- Rounding1aAPl-on-D-IE-1-10-128-32-128-u-0-2-0 -of-ipopt-D-IE-1-10-32-32-32-u-0-2-0-Pl-EvalObj-Pl	8579
D-IE-1-10-256-32-256-u-0-2-0- Rounding1aAPl-on-D-IE-1-10-256-32-256-u-0-2-0 -of-ipopt-D-IE-1-10-32-32-32-u-0-2-0-Pl-EvalObj-Pl	8579
D-IE-1-10-256-32-256-u-0-2-0- Rounding2aAPl-on-D-IE-1-10-32-32-32-u-0-2-0 -of-ipopt-D-IE-1-10-32-32-32-u-0-2-0-Pl-EvalObj-Pl	8883
D-IE-1-10-256-32-256-u-0-2-0- Rounding2aAPl-on-D-IE-1-10-64-32-64-u-0-2-0 -of-ipopt-D-IE-1-10-32-32-32-u-0-2-0-Pl-EvalObj-Pl	8908
D-IE-1-10-256-32-256-u-0-2-0- Rounding2aAPl-on-D-IE-1-10-128-32-128-u-0-2-0 -of-ipopt-D-IE-1-10-32-32-32-u-0-2-0-Pl-EvalObj-Pl	8875
D-IE-1-10-256-32-256-u-0-2-0- Rounding2aAPl-on-D-IE-1-10-256-32-256-u-0-2-0 -of-ipopt-D-IE-1-10-32-32-32-u-0-2-0-Pl-EvalObj-Pl	8883
D-IE-1-10-256-32-256-u-0-2-0- Rounding3aAPl-on-D-IE-1-10-32-32-32-u-0-2-0 -of-ipopt-D-IE-1-10-32-32-32-u-0-2-0-Pl-EvalObj-Pl	8512
D-IE-1-10-256-32-256-u-0-2-0- Rounding3aAPl-on-D-IE-1-10-64-32-64-u-0-2-0 -of-ipopt-D-IE-1-10-32-32-32-u-0-2-0-Pl-EvalObj-Pl	8536
D-IE-1-10-256-32-256-u-0-2-0- Rounding3aAPl-on-D-IE-1-10-128-32-128-u-0-2-0 -of-ipopt-D-IE-1-10-32-32-32-u-0-2-0-Pl-EvalObj-Pl	8531
D-IE-1-10-256-32-256-u-0-2-0- Rounding3aAPl-on-D-IE-1-10-256-32-256-u-0-2-0 -of-ipopt-D-IE-1-10-32-32-32-u-0-2-0-Pl-EvalObj-Pl	8533

Table C.5: Rounding for Fine Time Discretizations

Appendix D

Framework

The following chapter is structured as follows: First, we give a very brief introduction to AMPL. Second, we name the model files and explain the structure of the data files. Third, we describe how to execute the implemented algorithms.

AMPL (A Mathematical Programming Language) is a modeling software designed for optimization and was developed by [7]. In a typical AMPL session, a model file with extension `.mod` is loaded. Variables and parameters are declared in this file. Furthermore objective(s) and constraints are defined. On this basis, parameter and data sets are loaded. The corresponding file extension is `.dat`. Then a suitable solver and its options are chosen. If we chose an NLP solver, e.g. IPOPT, to solve a MIQP, the solver will not be able to handle the entire problem instance but in this case will solve an NLP relaxation of the instance. The last step is then to start the solver.

Besides this general framework, AMPL offers the possibility to work with scripts, usually ending with `.ampl`. In this scripts, the user is allowed e.g. to change the model, to manipulate variable values, or to sequentially call a solver.

In this framework, we built an AMPL model and prepared interesting data and parameter sets. Furthermore we implemented the presented preprocessing scheme (chapter 3) as well as the rounding schemes of chapter 4.

The directory `HeatCtrl` contains the AMPL implementation of the two problems introduced in chapter 2. For Heat Equation with Actuator Placement we use the name abbreviation `Pl` and for Heat Equation with Actuator Placement and Operation we use `PlOp`.

D.1 The Model

`*.mod` Files

These files contain the declaration of variables and parameters. Also they define the objectives: `RegularObjective` for normal solves, `PresolvedObjective` for solvings after PDE Elimination and `DummyObjective` for solvings during PDE Elimination. Furthermore the PDE time discretization can be chosen with: `Trapezoidal` for trapezoidal rule, `ExplicitE` for explicit Euler rule, and `ImplicitE` for implicit Euler rule. The boundary conditions are ensured by `Dirichlet{1,2,3,4}` and the source budget constraint with `SOS`. The big-M constraint (`PlOp` only) is formulated as upper bound `UBV` and lower bound `LBV{u,b}` on `v`, it is possible to choose between unbounded (`u`) and bounded from `0` (`b`).

- `HeatCtrl{Pl,PlOp}.mod` is the AMPL model
- `HeatCtrl{Pl,PlOp}1.mod` is the AMPL model if you use PDE Elimination Simple

`Instances/1*-2*-3*-4*-5*-6*-7*-8*-9*-10*-11*.dat` Files

The subdirectory `Instances` contains prepared instance files. Not all possible combinations already exist but they can be easily produced with the aid of an example file. The file name codes its information in a unique way and it is used for further naming of the output files. One file creates a problem instance by successive execution of the corresponding `*.dat` files:

- 1* `D`: Dirichlet boundary
- 2* `TR` or `IE`: trapezoidal rule or implicit Euler in time
- 3* actuator quantity
- 4* time horizon
- 5* time mesh
- 6* space mesh
- 7* control mesh
- 8* `u` or `b`: continuous control unbounded or bounded from 0
- 9* type of kappa
- 10* type of initial values
- 11* type of desired final state

`*.dat` Files

Files with the extension `.dat` are called by the instance files to specify parameters or to delete not needed constraints.

- `SpaceMesh{08,16,32,64}.dat` specifies the mesh size in space
- `TimeMesh{08,16,32,64,128,256}.dat` specifies the mesh size in time
- `ControlMesh{08,16,32,64,128,256}.dat` specifies the mesh size of the controls, must be less or equal the time mesh
- `TimeHorizon{10,15}.dat` specifies the the final time
- `Kappa{0,1,2,3}.dat` specifies the thermal diffusivity, it may vary in space
- `ActuatorQuantity{0,1,2,3}.dat` specifies the number of actuators
- `ActuatorLocations.dat` specifies the possible location of the actuators
- `InitialValues{1,2}.dat` specifies the initial condition for the PDE
- `DesiredFinalState{0,1}.dat` specifies the desired final state
- `Vunbounded.dat` continuous control `v` without bounds
- `Vbounded.dat` continuous control `v` $\geq$ `0`

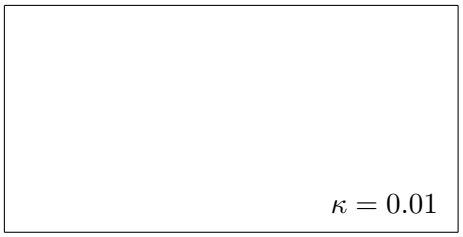

Figure D.1: `Kappa0.dat`

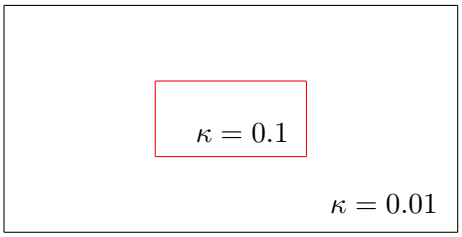

Figure D.3: `Kappa1.dat`

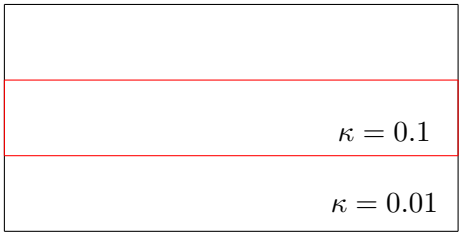

Figure D.2: `Kappa2.dat`

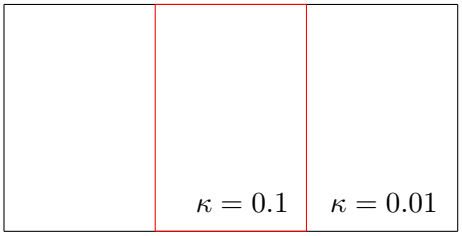

Figure D.4: `Kappa3.dat`

File	Value
`InitialValues1.dat`	$u_0(x,y) = 100\sin(\pi x)\sin(\pi y)$
`InitialValues2.dat`	$u_0(x,y) = (a_1x^4 + a_2x^3 + a_3x^2) \cdot (b_1y^4 + b_2y^3 + b_3y^2)$
`DesiredFinalState0.dat`	$u_f(x,y) = 0$
`DesiredFinalState1.dat`	$u_f(x,y) = 100\sin(\pi x)\sin(\pi y)$

Table D.1: Initial Values, Desired Final State

- `ImplicitEuler.dat`, `Trapezoidal.dat`, `ExplicitEuler.dat` chose one of the implemented discretizations

The geometry of the different thermal diffusivity options is illustrated in Figures (D.1-D.4). The information about initial states and desired final states can be found in Table D.1.

D.2 Algorithms

In chapter 3 and 4, we presented two versions of a preprocessing algorithm "Presolve" and we developed rounding based heuristics.

`*.ampl` Files

The simple PDE elimination is implemented in `Presolve1{Pl,PlOp}.ampl` and the version with convolution in `Presolve{Pl,PlOp}.ampl`. The name of the rounding scheme is composed of the general type of rounding (`1,2,3`), (`a`) for simple version, (`b`) for resolve version, and (`A`) indicates spacial rounding (for simple versions only). Due to the slight

differences in the AMPL models, we implemented the algorithms in a tailored manner for the two problems and indicated the versions with the abbreviations `Pl` and `PlOp`.

- `{ipopt,cplex,gurobi}.ampl` specifies solvers with options
- `{astros,baron,bnb,bonminBB,bonminHyb,bonminOA}.ampl` alternative solvers
- `Rounding{1a,1b,2a,2b,3a,3b}{·,A}{Pl,PlOp}.ampl` implementation of Rounding approaches, c.f. Table D.2
- `Rounding{1V,1W,3V,3W}.ampl` auxiliary code for rounding
- `SUR-W.ampl, EnsureSOS.ampl` auxiliary code for rounding type 2
- `SpacialRoundingA{V,W}` implementation of Spatial-Rounding
- `Presolve{Pl,PlOp}.ampl` implementation of PDE Elimination with Convolution
- `Presolve1{Pl,PlOp}.ampl` implementation of PDE Elimination Simple
- `BuildSol{·,1}{Pl,PlOp}.ampl` calculates state variables from presolved solutions
- `Plot{Pl,PlOp}.ampl` prints solution in `Matlab` syntax
- `solve.ampl` starts either IPOPT, CPLEX, or GUROBI
- `RegularObjective.ampl` deletes Dummy and Presolve Objective
- `DummyObjective.ampl` deletes Regular and Presolve Objective
- `GetRegularObjective{Pl,PlOp}.ampl` returns to Regular Objective (if deleted before)
- `DisplayTime.ampl` displays CPU time after solving
- `SaveControls{Pl,PlOp}.ampl` saves current binary controls in the subdirectory `controls/`
- `LoadControls{Pl,PlOp}.ampl` loads previously saved controls
- `unfixW.ampl` unfixes control variables for subsequent optimization run

D.3 Run Models and Algorithms

In order to simplify the usage of the software we created shell scripts which run solvers or the implemented algorithms on one or multiples instances.

Short Name	Rounding Type	Resolve	Spatial Rounding
Rounding1a{Pl,PlOp}	Maximum		
Rounding1aA{Pl,PlOp}	Maximum		✓
Rounding1b{Pl,PlOp}	Maximum	✓	
Rounding2a{Pl,PlOp}	Sum-Up naive		
Rounding2aA{Pl,PlOp}	Sum-Up naive		✓
Rounding2b{Pl,PlOp}	Sum-Up naive	✓	
Rounding3a{Pl,PlOp}	Maximum SUR		
Rounding3aA{Pl,PlOp}	Maximum SUR		✓
Rounding3b{Pl,PlOp}	Maximum SUR	✓	

Table D.2: Short Names of Rounding Schemes

*.sh files

RunModel{Pl,PlOp}.sh

The simplest possibility to run one or more instances is implemented in RunModel{Pl, PlOp}.sh. This shell script requires two inputs: the solver or rounding scheme and a text file containing a list of instances. Usual solver choices are MIQP solver (e.g. CPLEX, GUROBI) or QP/NLP solver (e.g. IPOPT). If a QP or NLP solver is chosen, integer constraints on variables are ignored and a solution is of the NLP relaxed problem. In case of a rounding scheme, IPOPT is the default solver for solving the NLP relaxation of the instance. The instance list is usually named RunModel{Pl,PlOp}.txt and is formatted as follows:

```
D-IE-1-10-32-32-04-u-0-2-0
D-IE-1-10-32-32-08-u-0-2-0
D-IE-1-10-32-32-16-u-0-2-0
D-IE-1-10-32-32-32-u-0-2-0
```

In order to save the log messages we print the output in a file with the extension .out and the errors messages in a file with extension .err both in the directory output. A command may look like line 1 in Table D.3.

Further, this script produces for every given instance the file:

<solver/Rounding scheme>-<Instance name>-{Pl,PlOp}.out

with solver log messages of the solver and variable values of the solution in Matlab syntax which is saved in the subdirectory output. For later reference, the control values are saved in the subdirectory controls and are named:

<solver/rounding scheme>-<instance name>-{Pl,PlOp}-W.out
<solver/rounding scheme>-<instance name>-PlOp-V.out

We remark that for the problem instances Pl only the control w is saved, because of the absence of a continuous control in this model.

RunModelPresolve{Pl,PlOp}.sh

This script is designed to execute the PDE Elimination with Convolution algorithm prior to optimization. Like RunModel {Pl,PlOp}.sh it is possible to choose either a solver (e.g. Ipopt, CPLEX, GUROBI) or a rounding scheme. The behavior of the solver and the roundings are the same as in the version without Presolve. The second input is the instance list which is usually named RunModelPresolve{Pl,PlOp}.txt and formatted as

`RunModel{Pl,PlOp}.txt`. To start a computation, the command in line 2 of TableD.3 can be used. Log messages and results are saved in a file located in the `output` directory with the naming convention:

`<solver/rounding scheme>-<instance name>-presolved-{Pl,PlOp}.out`

As for `RunModel{Pl,PlOp}.sh` the control variables of the solutions are saved with the same naming scheme in the subdirectory `controls`.

`RunModelPresolve1{Pl,PlOp}.sh`

This script is designed to execute the PDE Elimination Simple algorithm. Due to the inefficiency of this algorithm, we keep the script for the purpose of run time comparison only. Therefore we do not save the controls separately in the `controls` directory. The behavior and input formatting is identical to the other `RunModel*.sh` scripts. The command in line 3 of Table D.3 can serve as a template.

Log messages and results are saved in a file located in the `output` directory with the naming convention:

`<solver/rounding scheme>-<instance name>-presolved1-{Pl,PlOp}.out`

`RunModelRoundingMIQP{Pl,PlOp}.sh`

In this script we combine the preprocessing Presolve, Rounding, and the solver CPLEX. This means, that we first apply Presolve. Then, on the basis of the NLP relaxation, we do the rounding. Finally, we hot start CPLEX with the rounded solution. The input is a rounding scheme and a file containing a list of Instances. This file is formatted like the earlier mentioned `RunModel{Pl,PlOp}.txt`. The terminal command in line 4 of Table D.3 can serve as a template to execute the approach. Log messages and results are saved in a file located in the `output` directory with the naming convention:

`cplex-<rounding scheme>-<instance name>-presolved-{Pl,PlOp}.out`

`FineRoundingPl.sh`

Due to the fact that even to compute an NLP relaxed solution is expensive on fine meshes, we would like to use NLP relaxed solution of instances with coarser meshes but round on a fine mesh. As the problem "Heat Equation with Actuator Placement and Operation" includes besides integer variables continuous controls as well, the `FineRoundingPl.sh` script is available for the problem without continuous controls only. It requires as input a rounding scheme (without resolve) and a file containing the list of instances. This file, usually named `FineRoundingPl.txt`, is formatted as follows:

```
D-IE-1-10-32-16-32-u-0-2-0 ipopt-D-IE-1-10-16-16-16-u-0-2-0-Pl 16
D-IE-1-10-64-16-64-u-0-2-0 ipopt-D-IE-1-10-16-16-16-u-0-2-0-Pl 16
```

Where line corresponds to one instance: The first column represents the data of the desired fine mesh, the second column is the coarse mesh solution, and the third column indicates the time mesh size of the coarse mesh. The data sets of the first and the second column must be identical except for the time and control meshes. The command in line 5 of Table D.3 can serve as a template.

Then the coarse mesh solution is loaded, extended to a finer mesh, and rounded according to the chosen scheme. This rounded control is then saved in the subdirectory `controls` with the file name in line 6 in Table D.3. An example is give in in line 7 of the table.

`EvaluateObjective{Pl,PlOp}.sh`

The script `EvaluateObjective.sh` simulates the PDE and evaluates the objective of given control values. This is useful to compare objective values of solutions which are computed on different mesh sizes or generated by `FineRoundingPl.sh`. A file with a instance list is required as input. For the problem with only binary controls, it is usually named `EvaluateObjectivePl.txt` and is formatted as in line 8 of Table D.3.

The instance file `EvaluateObjectivePlOp.txt`, for the problem with both binary and continuous controls, can be formatted as follows:

```
D-IE-1-10-16-16-16-u-0-1-0 ipopt-D-IE-1-10-08-08-08-u-0-1-0-PlOp 8
D-IE-1-10-16-16-16-u-0-1-0 Rounding1aPlOp-D-IE-1-10-08-08-08-u-0-1-0-PlOp 8
```

For both instance files hold that every line corresponds to one instance. The first column is the data of the desired fine mesh simulation, the second column indicates the origin of the control values, and the third column the time mesh size of the origin instance. The default solver for the simulation is IPOPT. To start a simulation, the following command in line 9 of Table D.3 can serve as a template.

Log messages and results are saved in a file located in the `output` directory with the naming convention given in line 10 of Table D.3.

1	nohup ./RunModelPl.sh ipopt RunModelPl.txt > output/ipopt.out 2> output/ipopt.err < /dev/null &
2	nohup ./RunModelPresolvePl.sh cplex RunModelPresolvePl.txt > output/cplex.out 2> output/cplex.err < /dev/null &
2	nohup ./RunModelPresolve1Pl.sh Rounding1aPl RunModelPresolve1Pl.txt > output/Rounding1aPl.out 2> output/Rounding1aPl.err < /dev/null &
3	nohup ./RunModelRoundingMIQPPl.sh Rounding3aPl RunModelPl.txt > output/RcplexPl.out 2> output/RcplexPl.err < /dev/null &
4	nohup ./FineRoundingPl.sh Rounding1aAPl FineRoundingPl.txt > output/fineRoun1aAPl.out 2> output/fineRoun1aAPl.err < /dev/null &
5	<rounding scheme>-on-<coarse mesh instance (first col.)>-of-<coarse mesh instance (second col.)>-W.out
6	Rounding1aAPl-on-D-IE-1-10-32-16-32-u-0-2-0-of-ipopt-D-IE-1-10-16-16-16-u-0-2-0-Pl-W.out
7	D-IE-1-10-128-16-128-u-0-2-0 Rounding1aPl-on-D-IE-1-10-16-16-16-u-0-2-0-of-ipopt-D-IE-1-10-16-16-16-u-0-2-0-Pl 16 D-IE-1-10-128-16-128-u-0-2-0 Rounding1aPl-on-D-IE-1-10-32-16-32-u-0-2-0-of-ipopt-D-IE-1-10-16-16-16-u-0-2-0-Pl 32 D-IE-1-10-128-16-128-u-0-2-0 Rounding1aPl-on-D-IE-1-10-64-16-64-u-0-2-0-of-ipopt-D-IE-1-10-16-16-16-u-0-2-0-Pl 64
8	nohup ./EvaluateObjectivePlOp.sh EvaluateObjectivePlOp.txt > output/EvaluateObjectivePlOp.out 2> output/EvaluateObjectivePlOp.err < /dev/null &
9	<simulation instance (first col.)>-<origin instance (second col.)>-EvalObj-{Pl,PlOp}.out

Table D.3: Template Code

Bibliography

[1] M. Abramowitz. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover Publications, 1972.

[2] E. Beale and J. Tomlin. Special facilities in a general mathematical programming system for non- convex problems using ordered sets of variables. In J. Lawrence, editor, *Proceedings of the 5th International Conference on Operations Research*, pages 447–454, Venice, Italy, 1970.

[3] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan. Mixed-integer nonlinear optimization. *Acta Numerica*, 22:1–131, 5 2013.

[4] J. Butcher. *Numerical Methods for Ordinary Differential Equations*. Wiley, 2003.

[5] R. J. Dakin. A tree search algorithm for mixed programming problems. *Computer Journal*, 8:250–255, 1965.

[6] M. Fischetti and A. Lodi. Local branching. *Mathematical Programming*, 98:23–47, 2002.

[7] R. Fourer, D. Gay, and B. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Duxbury Press, 2002.

[8] W. Gautschi. *Numerical Analysis*. Birkhäuser Basel, 2011.

[9] S. Hamdi, W. E. Schiesser, and G. W. Griffiths. Method of lines. *Scholarpedia*, 2009.

[10] J. Haslinger and R. A. Mäkinen. On a topology optimization problem governed by two-dimensional Helmholtz equation. *Computational Optimization and Applications*, 62(2):517–544, Nov. 2015.

[11] O. V. Iftime and M. A. Demetriou. Optimal control of switched distributed parameter systems with spatially scheduled actuators. *Automatica*, 45(2):312 – 323, 2009.

[12] I. ILOG. *IBM ILOG CPLEX V12.1, User's Manual for CPLEX*. IBM Corp., New York, USA, 2009.

[13] R. G. Jeroslow. There cannot be any algorithm for integer programming with quadratic constraints. *Operations Research*, 21(1):221–224, 1973.

[14] R. Kannan and C. Monma. On the computational complexity of integer programming problems. In R. Henn, B. Korte, and W. Oettli, editors, *Optimization and Operations Research*, volume 157 of *Lecture Notes in Economics and Mathematical Systems*, pages 161–172. Springer, 1978.

[15] S. Leyffer, T. Munson, S. Wild, B. van Bloemen Waanders, and D. Ridzal. Mixed-integer PDE-Constrained Optimization. Position Paper 15 submitted in response to the ExaMath13 Call for Position Papers, 2013.

[16] N. Morrison. *Introduction to Fourier Analysis*. Wiley, 1994.

[17] G. Nannicini, P. Belotti, and L. Liberti. A local branching heuristic for MINLPs. arXiv:0812.2188v1 [math.CO], 2008.

[18] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer-Verlag, New York, 1999.

[19] S. Sager. *Numerical methods for mixed–integer optimal control problems*. Der andere Verlag, Tönning, Lübeck, Marburg, 2005.

[20] T. Tao. *Analysis II*, volume 38 of *Texts and Readings in Mathematics*. Springer Singapore, 2016.

[21] J. W. Thomas. *Numerical Partial Differential Equations - Finite Difference Methods*. Springer New York, 1995.

[22] F. Tröltzsch. *Optimale Steuerung partieller Differentialgleichungen: Theorie, Verfahren und Anwendungen*. Vieweg+Teubner Verlag, Wiesbaden, second edition, 2009.

[23] A. Wächter. Short tutorial: Getting started with ipopt in 90 minutes. In U. Naumann, O. Schenk, H. D. Simon, and S. Toledo, editors, *Combinatorial Scientific Computing*, number 09061 in Dagstuhl Seminar Proceedings. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009.

[24] A. Wächter and L. Biegler. On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming. *Mathematical Programming*, 106(1):25–57, 2006.

Eidesstattliche Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, wobei ich alle wörtlichen und sinngemäßen Zitate als solche gekennzeichnet habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Magdeburg, den 18. April 2017