# Automated decomposition-based optimization algorithm selection and configuration via artificial intelligence and network science

**A THESIS**
**SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL**
**OF THE UNIVERSITY OF MINNESOTA**
**BY**

**Ilias Mitrai**

**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS**
**FOR THE DEGREE OF**
**DOCTOR OF PHILOSOPHY**

**Advised by Prodromos Daoutidis**

**July, 2023**

# Acknowledgement

First, I would like to thank my advisor, Professor Prodromos Daoutidis, for his guidance and constant support in academic or technical issues and for teaching me how to write, present, and communicate ideas. I am grateful for his patience during the beginning of my Ph.D. and for allowing me to explore new research directions and always make connections between different fields.

I would also like to thank Professor Chris Bartel, Professor Ankur Mani, and Professor Qi Zhang for agreeing to serve on this committee and for their feedback and comments on this thesis. Additionally, I would like to thank Julie Prince, Graduate Program Coordinator in the Chemical Engineering and Materials Science department, for always being here to answer all my questions about courses, enrollment, the preliminary exam, and everything else I was asking.

I want to thank my friends here in Minnesota. I consider myself very lucky to have met people with whom I share the same interests and create a supportive environment that helps us go through the long journey of a Ph.D. I want to thank current and past members of the Daoutidis group with whom I shared the same office: Andrew Allman, Pedro Constantino, Harman Dewantoro, Ritoban Ghosh, Victoria Jones, Shaaz Khatib, Baiwen Kong, Jingjun Liu, Oswaldo Andres Martinez, Manjiri Moharir, Matthew Palys, Davood Babaei Pourkargar, Benjamin Riley, Alexander Smith, Wentao Tang, Hanchu Wang. I want to thank Wantao Tang and Matthew Palys for our long discussions about optimization, control, and energy systems. Also, I would like to thank the members of Qi Zhang's research group: Shivi Dixit, Rishabh Gupta, Jnana Sai Jagana, Yeonseo (Eden) Jeong, Yen-An Lu, Tushar Rathi, Prahalad Srinivasan. I would also like to thank Rishabh Gupta and Hanchu Wang for their friendship during the last five years. I also want to thank the Greek community at the University for creating an environment where I could feel like being back in Greece even though we were in the cold winter of Minnesota. Also, I want to thank Konstantinos Polyzos for the endless discussions about research and the long afternoon walks during the covid era.

Finally, I would like to thank my parents, Dimitris Mitrai and Mirela Nasai Mitrai, for their constant support during my undergraduate studies in Greece and my graduate studies here in the US.

*To my parents*

# Abstract

Decision-making problems arise in a wide range of applications in chemical engineering. The efficient solution of complex and large-scale optimization problems is key for addressing problems related to the decarbonization of the chemical industry and the design and operation of sustainable manufacturing systems. Decomposition-based solution algorithms can be used to improve the tractability of such problems by exploiting their underlying structure. However, their application is problem-specific and time-consuming, and a generic framework for their automatic implementation is currently lacking. This thesis is focused on automating the application of decomposition-based solution algorithms and accelerating their performance using tools from artificial intelligence and network science, as discussed in the two parts of this thesis. The main contributions are:

1. We develop a graph classification approach, which can be applied to generic problems and considers the detailed coupling among the variables and constraints to determine whether an optimization problem should be solved using a decomposition-based or a monolithic solution approach.

2. We use Stochastic Blockmodeling (and its variants) and Bayesian inference to learn the underlying structure of the problem based on an appropriate graph representation of an optimization problem. The learned structure is used as the basis for the application of decomposition-based solution algorithms.

3. We propose a machine learning approach to learn how to optimally initialize cutting plane-based decomposition-based methods for the solution of optimization problems that arise in the real-time operation of chemical processes.

4. We explore the relation between problem formulation and the computational performance of decomposition-based solution methods and propose a new formulation and solution approach for the integration of process operations with dynamic optimization.

5. We propose a machine learning-based approach to accelerate Generalized Benders Decomposition for the solution of mixed integer model predictive control problems that arise in the operation of chemical processes.

# Contents

# List of Tables

# List of Figures

# Chapter 1
# Introduction

Mathematical optimization is one of the pillars of process systems engineering (PSE) and has been widely used to solve decision-making problems that arise in a wide range of applications, such as process design and synthesis, supply chain management, production planning and scheduling, and process control. Over the last three decades, a large number of papers have been published focusing on modeling these problems [293, 172, 89, 211, 158, 142, 187, 55, 104, 115] as mathematical programming problems in standardized forms. Simultaneously, significant advances have been made in both optimization theory and algorithms for the solution of a broad class of problems such as Nonlinear (NLP) [40], Mixed Integer Linear (MILP) [71], and Mixed Integer Nonlinear (MINLP) [255, 47, 100] programming problems, as well as the modeling, conversion, and solution of stochastic programming [117], robust optimization [31, 26], and disjunctive programming problems [278].

In recent years *large-scale and complex optimization problems* have become increasingly important, for example, those related to the decarbonization of the chemical industry and energy sector, the design of resilient and circular supply chain networks, and the sustainable operation of the manufacturing system [216]. These problems include control or dynamic optimization of large scale systems [29, 163, 214], design or operation in an uncertain economic environment [301, 164, 112, 296], integrated decision making across multiple time scales [17, 245, 82, 75], integrated design and operations [297, 227, 4] and logistics on an enterprise-wide or nation-wide level [130, 173, 165]. Due to the nonconvex, nonlinear, and mixed-integer nature of these problems, they are inherently nonscalable and tend to be computationally difficult to solve.

Different approaches have been followed in the process systems engineering community to improve the tractability of such problems. These approaches either focus on the problem formulation or on the solution method. In the former case, one may develop alternative formulations [101, 169], use alternative ways of conversion to standard mathematical programs [72, 299, 278], or use reduced-order and hybrid surrogate models [73, 34, 259, 49, 257]. This approach aims at reducing the computational complexity of the model, however, as expected it introduces some approximation error that might affect the solution of the optimization problem.

For cases where a drastic reduction in complexity can not be achieved or the approximation error is significant, *decomposition* is a natural and common approach for solving

optimization problems. The decomposition approach exploits the fact that although the aforementioned problems are complex and large-scale, they are also structured. The structure of an optimization problem, or a system in general, arises due to the coupling or interaction among the elements of the underlying system, and this structure is reflected in the interaction pattern between the variables and constraints of an optimization model. Decomposition, as a principle, has been widely used to model, analyze, and control complex engineering systems [194, 193, 211, 186] and may refer to (i) the decomposition of algebraic operations into maximally parallelized tasks [70, 298, 291, 286] and (ii) the decomposition of the problem into interacting subproblems that need to be solved iteratively [69, 96]. We focus on decomposition in the sense of decomposing an optimization problem into a number of easier-to-solve subproblems, whose iterative solution and coordination can lead to the solution of the original problem. This type of decomposition has been applied to a wide spectrum of problems in process systems engineering, including supply chain design [224, 148], operations under uncertainty [224, 117], integrated decision making [176, 64], planning and scheduling [213, 263, 166], model predictive control [180, 61, 3, 271], and learning from data [12, 51, 267, 45].

Despite the wide success of decomposition-based solution algorithms, their off-the-shelf application is challenging, and their computational efficiency over monolithic approaches is not known a-priory. Decomposition-based solution approaches are usually problem specific, their development is time-consuming and requires domain knowledge about the underlying problem and the decomposition-based solution algorithm that will be used. The literature on decomposition-based solution methods is mainly focused on analyzing the theoretical aspects of these algorithms; the implementation is usually left to the user. Although this can be justified since these algorithms exploit the underlying structure, which can differ significantly for different classes of problems, it also limits the wide application of these methods by optimization practitioners. Therefore, *automated methods* are necessary to determine w*hen to select* a decomposition-based solution algorithm over a monolithic one, and *how to configure* (i.e., implement) the algorithm.

## 1.1 When to use a decomposition-based solution algorithm

The first step during the solution of a large-scale optimization problem is to determine which solution strategy should be followed. This gives rise to the following question:

**Question 1.1.** *Should a given optimization problem be solved with a decomposition or a monolithic-based solution algorithm?*

Identifying the best solution strategy for a computational task is nontrivial; the computational performance of a decomposition-based solution method depends on multiple factors, such as problem formulation, decomposition, coordination, and initialization. Considering, for example, the solution of an MINLP problem with Lagrangean decomposition, the efficiency of the algorithm depends on the decomposition of the problem since it affects the number of complicating constraints, the update scheme for the Lagrangean multipliers, and the initialization of the variables and the multipliers. Similarly, the efficiency of monolithic solution approaches, such as branch and bound, depends on the type and number of nonconvex terms in the problem formulation, branching strategy, etc. Machine learning methods have been used to train a classifier that determined whether a MILP problem should be solved using Dantzig-Wolfe decomposition or branch and cut, based on a vectorial feature representation of the optimization problem [161]. Although this approach can potentially be applied to nonlinear and mixed integer nonlinear problems, representing these problems by a set of features is challenging due to the different types of nonlinear functions that are present in the constraints and objective. To overcome these limitations the following contribution is made:

- In Chapter 2 we propose a graph classification approach to determine when to decompose generic (linear, nonlinear, and mixed integer) optimization problems by considering the detailed interaction pattern between the variables and constraints of an optimization problem. The proposed approach can be potentially applied to different graph representations of an optimization problem and consider multiple features obtained from the problem formulation.

## 1.2 How to decompose an optimization problem

The prerequisite for implementing a decomposition-based solution algorithm is to determine the *decomposition itself*, i.e., the partition of the variables and constraints into subproblems, and the hierarchical relation between the subproblems. Such decompositions are typically adopted based on intuition, assuming a certain structure based on insights about the system or the optimization problem. However, inferring this structure is time-consuming, and there might exist problems that do not have an apparent structure for decomposition.

So far, determining systematically the most suitable decompositions of optimization problems remains a crucial and open problem. Key to the solution of this problem is finding a conceptual and methodological framework for evaluating the underlying structure and sparsity of an optimization problem and determining the most suitable decomposition in concert with the corresponding decomposition-based solution method. Therefore, the question we will consider is the following:

**Question 1.2.** *How to learn the stucture of a mathematical optimization problem and how to exploit it for the automatic application of decomposition-based solution methods?*

Learning or identifying the underlying structure of an optimization problem is not a new idea. Initial attempts relied on graph theory to identify the underlying structure by representing an optimization problem as a graph or hypergraph and applying graph partitioning algorithms to identify a partition of the graph into $N$ independent blocks [195, 287, 91, 11]. The main limitation of these algorithms is that the number of blocks must be determined a-priory and the objective with which the partition is optimal is the size of the set of nodes that separates the blocks. Therefore, the structure identified might not necessarily be the actual structure of the problem.

An alternative approach is to use tools from network science [20, 219] which focuses on the coarse-grained and statistical properties of a complex system. This approach has been applied to optimization problems where the decomposition is done according to the presence of community structures in the network representations of the optimization problem. Communities refer to subnetworks that are strongly interconnected inside but weakly interconnected between, where the statistical significance of such interactions is typically quantified according to a metric called modularity [220, 21, 42]. This approach has been applied for the solution of mixed integer linear programming problems using Dantzig-Wolfe decomposition [30, 153], nonlinear programming problems [271], and mixed integer nonlinear programming problems [5, 6]. Although the application of modularity-based community detection leads to high-quality decompositions, it is limited in the sense that it can only perform well if the problem has such a community structure. However, given the wide range of optimization problems that arise for different applications, one would expect that other types of structure might be present. Finally, this approach can not detect the possible hierarchical structure among the communities of an optimization problem. To overcome these limitations the following contributions were made in the first part of the thesis:

- In Capter 3 we propose the combination of modularity-based community detection and centrality analysis to identify the community structure of an optimization problem and the underlying hierarchical relation between the communities.
- In Capter 4 we propose the application of Stochastic Blockmodeling [240] as a systematic framework to learn the underlying structure of an optimization problem without any a-priory assumptions on the structure of the problem.
- In Capter 5, we consider the structure detection problem for enterprise-wide optimization problems that consider simultaneously decisions across multiple scales. Specifically, we propose the application of nested Stochastic Blockmodeling [236], which assumes that a graph is generated by a nested sequence of Stochastic Blockmodels, for learning the structure of such problems at different scales.

## 1.3 How to initialize a decomposition-based solution algorithm

It is widely known that the initialization of an algorithm can have a significant effect on computational performance. However, for complex algorithms with multiple steps, such as decomposition-based ones, the effect of the initialization on the computational performance is not obvious. In general, prior to the application of a decomposition-based algorithm, the following question arises:

**Question 1.3.** *How to initialize decomposition-based solution algorithms such that the solution time is minimized?*

Identifying the optimal initialization is especially important for cases where an optimization problem is solved online to compensate for updated process information. The initialization can be considered as a hyperparameter of the algorithm, thus identifying the optimal initialization can be considered as an algorithm configuration task [258]. This is a black box optimization problem since the solution time is not known a-priory and evaluating the solution time for a given initialization of a decomposition-based algorithm is time-consuming. Additionally, based on the decomposition-based algorithm used, the initialization itself can be a complex task. For example, for distributed algorithms, the initialization determines the initial guess for the values of the Lagrangean

multipliers, whereas for Benders decomposition, the initialization determines the number of cuts that can be added in the master problem. In this case, selecting the cuts is an open problem. To overcome these issues, the following contributions were made:

- In Chapter 6 we consider the initialization of Generalized Benders Decomposition. Specifically, we propose the application of active learning for learning a surrogate model that predicts the solution time of a given problem and a given initialization. This surrogate model is subsequently used to identify the optimal initialization.

## 1.4 How to accelerate the computational performance of decomposition-based solution algorithms

Despite the wide success of decomposition-based solution methods in reducing the solution time, convergence can be an issue, especially for nonlinear and nonconvex problems, and the computational time can still be above the computational budget for online applications. The computational performance of a decomposition-based solution algorithm depends on the problem formulation and the implementation of the algorithm. The following questions will be considered:

**Question 1.4.** *How to combine knowledge from mathematical optimization and chemical engineering to develop problem formulations whose structure is ideal for the application of decomposition-based solution algorithms, and how to improve the performance of the algorithm itself?*

The problem formulation determines the structure of the problem and therefore the subsequent computational complexity of the subproblems. Usually, the modeling step is guided by the modeler's understanding of the underlying system and general modeling practices [212, 284]. It does not consider the effect of the modeling on the structure of the problem and therefore on the computational efficiency of a decomposition-based solution strategy. However, considering the effect of the modeling on the efficiency of decomposition-based methods, especially for large-scale problems, might lead to problem formulations whose structure is ideal for the application of a specific decomposition-based solution algorithm. The second approach to accelerate decomposition-based methods is to focus on the algorithm itself and accelerate the individual steps of the method. This approach, as expected, is problem specific, yet it can significantly reduce the solution time for optimization problems that must be solved online. We consider these two

questions for the application of Generalized Benders Decomposition to problems that arise in the operation of chemical processes. The relevant contributions in the thesis are:

- In Chapter 7 we consider the integration of planning, scheduling, and dynamic optimization which arise in the operation of chemical processes. We propose a new formulation for the integrated problem where the integrated problem can be decomposed into a master problem that considers the planning and scheduling decision and a number of independent subproblems which consider the dynamic operation of the systems. Given this decomposition, we propose two multicut Generalized Benders Decompositions which can solve the integrated problem in reduced computational time.

- In Chapter 8 we propose a machine learning-based branch and check Benders decomposition algorithm for the solution of mixed integer model predictive control. THe reduction in solution time is achieved by solving the master problem only once and approximating the information necessary to construct the Bender cuts via machine learning models.

## 1.5 Summary

This Ph.D. thesis collects most of my research works on developing an automated framework for automated decomposition-based solution algorithm selection and configuration. The contents of the following chapters are based on the author's written journal and conference papers (Chapter 2 – [205, 203], Chapter 3 – [198], Chapter 4 – [207], Chapter 5 – [199], Chapter 6 – [204, 206], Chapter 7 - [201], Chapter 8 – [202]) with necessary minor revisions. Chapters 2 and 6 correspond to papers under review, the first part of Chapter 8 is also a paper under review and the second part is not published.

# Part I
# Learning when and how to decompose an optimization problem

# Chapter 2
# Learning when to decompose optimization problems via graph classification

## 2.1 Introduction

Decision-making problems arise in a wide range of applications in chemical engineering, ranging from the molecular to the enterprise-wide scale [128, 115, 75, 244]. Mathematical programming is a framework that has been widely used to model such decision-making (optimization) problems. Different solution methods have been developed for the solution of a broad class of optimization problems, such as Mixed Integer Linear (MILP) (cite), Nonlinear (NLP) [285], and Mixed Integer Nonlinear Programming problems [273, 196]. Despite the significant advances, the solution of optimization problems that arise in chemical engineering applications can be challenging due to the nonlinear nature of most physical and chemical processes, the presence of multiple temporal and spatial scales, and the discrete nature of design and operational decisions. The combination of these features leads to complex and large-scale optimization problems whose monolithic solution is challenging.

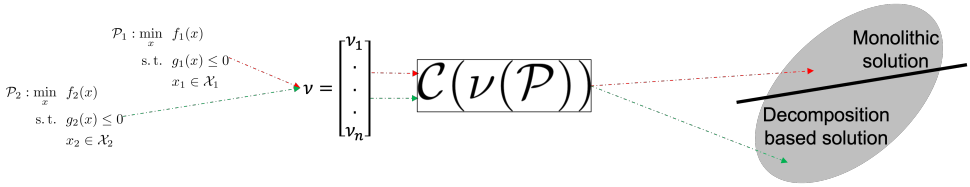


Figure 2.1: Algorithm selection via classification based on handcrafted features $\nu$

Decomposition-based solution algorithms have been widely used to solve such complex and large-scale optimization problems by exploiting the underlying structure of the problem [69, 148, 176, 213, 12, 51, 166, 271]. In general, these algorithms can be

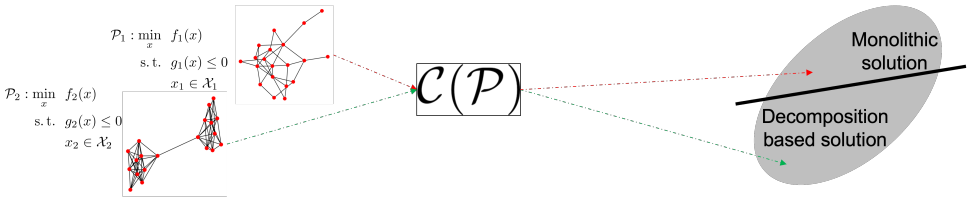


Figure 2.2: Algorithm selection via graph classification

classified as distributed or hierarchical. In both cases, a complex optimization problem is decomposed into a number of easier-to-solve subproblems which can be solved efficiently. The main difference lies in the solution and coordination of the subproblems. In distributed decomposition-based algorithms, the subproblems are solved in parallel and are coordinated via dual variables. Typical examples include Lagrangean relaxation [94, 93] and decomposition [119], and the Alternating Direction methods of Multipliers [48]. In hierarchical algorithms, the subproblems are solved sequentially, based on some underlying hierarchy of the problem, and are coordinated via the addition of cuts. Typical examples are Benders [27], Generalized benders [108], Outer Approximation [86, 95], and Bilevel [69] decomposition algorithms. Finally, there are algorithms that exploit both distributed and hierarchical structures such as cross-decomposition [281].

Despite the wide success of decomposition-based solution methods their efficiency over monolithic solution approaches is not known apriory since it depends on multiple factors. For example, the performance of Benders and Generalized Benders decomposition depends on the computational complexity of the master problem and the subproblem, the number of infeasible subproblems that are solved, and the quality of cuts that are generated. Similarly, for distributed algorithms such as Lagrangian decomposition, the convergence depends on the coordination scheme and the initialization of the dual variables.

In general, state-of-the-art optimization algorithms are black-box systems, and determining the computational performance of an algorithm for a given problem apriory is challenging. Similarly, determining whether to use a decomposition or a monolithic-based solution algorithm for the solution of a given problem is not obvious. Although the number of variables and constraints can guide this decision, for some classes of problems, such as convex MINLP problems, decomposition-based solution methods can be more efficient than monolithic solution approaches even for problems with few variables and constraints. Therefore, automated methods are needed to determine whether a problem should be solved using a monolithic or a decomposition-based solution strategy.

The problem of selecting the best solution method for a given computational task is formally known as the algorithm selection problem [248, 150]. Although this is a well-studied problem in Computer Science it has received less attention in the Process Systems Engineering community. The algorithm selection problem for optimization problems has three components; (1) the space of optimization problems $\mathcal{P}$, i.e., all the problems that can be considered, (2) the set of algorithms $\mathcal{A}$, i.e., all algorithms that

are available for the solution of the problems in $\mathcal{P}$, and (3) the performance space $\mathcal{M}$, i.e., some metric used to compare solution methods for a given problem. Given these sets the algorithm selection problem is defined as follows:

**Problem 2.1.** (Algorithm selection) Given an optimization problem $P$ and a set of algorithms $\mathcal{A} = \{a_1, ..., a_n\}$, which algorithm $\alpha^*$ should be used to solve the problem such that a desired performance function $m : \mathcal{P} \times \mathcal{A} \to \mathcal{M}$ is optimized.

The performance function $m$ can be either the solution time or the best feasible solution for a given computational time budget. This problem can be posed as an optimization problem as follows

$$a^* \in \arg\min_{a \in \mathcal{A}} m(P, a). \tag{2.1}$$

This is known as the per-instance algorithm selection problem, since the algorithm $\alpha^*$ is optimal for a given problem $P$. Given the inherent complexity of optimization algorithms, the computational efficiency of an optimization algorithm for a given problem is not known a priory. Thus the algorithm selection problem is a black box optimization problem since the function $m$ is now known explicitly. Although black-box optimization algorithms, such as Bayesian Optimization [103] can be used to solve this problem, they can be slow for automated online applications.

Automated algorithm selection tools rely on Artificial Intelligence (AI) and Machine Learning (ML) to approximate the solution of the algorithm selection problem (see Fig. 2.1). A typical approach is to approximate the computational performance of an algorithm, measured in terms of solution time for global optimization solvers, for a given problem (or class of problems) using a surrogate model [265, 141, 159, 111, 243], such as Decision Trees [22], Gaussian Processes [137], Neural Networks [266], and Ridge regression [134]. The surrogate model is subsequently used to select the best solution method. Another approach is to approximate the solution of the algorithm selection problem itself using classification techniques, where a classifier is used to approximate the best solution strategy [292]. These approaches have been applied to determine the best solution strategy for satisfiability problems, Mixed Integer Linear (MILP) and Mixed Integer Quadratic (MIQP) Programming problems [292, 44, 161].

The main limitation of these AI/ML approaches is that an optimization problem can not be the input to standard surrogate models mentioned earlier. An optimization problem is a complex data point since it has different types of variables and constraints

and an objective function. Furthermore, two optimization problems can have different number of variables and constraints making their comparison difficult. This necessitates the usage of a handcrafted set of features, such as the number of variables, constraints, range of coefficients in constraints and objective, the condition number of constraint matric, etc. (see [265, 292, 161] for a detailed description of the features). In essence, the usage of features is a dimensionality reduction step which although enables the application of standard ML tools to a broad class of optimization problems, aggregates information about the problem and does not consider the exact structural coupling among the variables and constraints which is the basis of decomposition-based solution algorithms.

To overcome these obstacles, in this paper we propose a graph classification approach to determine if a decomposition-based solution algorithm should be used. Specifically, based on our previous work we represent an optimization problem as a graph [6, 207], where every node is a constraint or a variable (see Fig. 2.3). This representation captures the structural coupling among the variables and constraints of a problem. We extend this representation by adding features in every node (see Fig. 2.4). These features are obtained from the problem formulation and their concatenation forms the feature matrix. Overall, the structural coupling among the variables and the constraint is captured via the adjacency matrix and the functional coupling via the feature matrix. Given this representation, we use geometric deep learning [50] to build a graph classifier to approximate the solution of the algorithm selection problem (Eq. 2.1) as presented in Fig. 2.2. In this approach, the classifier predicts whether an optimization problem $P_i$ should be solved using a decomposition-based solution method by considering the exact structural and functional coupling among the variables and the constraints through the adjacency $A_i$ and feature $F_i$ matrices. The proposed approach can be applied to generic optimization problems with different number and type of variables and constraints. For illustration, we apply the proposed approach to determine if a convex MINLP should be solved using Branch and Bound or the Outer Approximation algorithm. We use benchmark optimization problems for training and testing. The results show that the accuracy of the classifier on the testing dataset is 90%. Finally, we show how the proposed approach can be integrated into existing technology regarding the solution of convex MINLP problems.

The rest of the paper is organized as follows: in section 2.2 we present the graph representation of an optimization problem, in section 2.3 we pose the question of whether

to decompose as an algorithm selection problem, in Section 2.4 we present the graph classification-based algorithm selection solution approach and Section 6.5 we show how the proposed approach can be used to determine when to decompose convex MINLPs.

## 2.2 Graph representation of an optimization problem and node features

Consider the following general optimization problem

$$\begin{aligned} P := \underset{x,y}{\text{minimize}} \quad & f(x,y) \\ \text{subject to} \quad & g_i(x,y) \leq 0 \quad \forall i = 1,...,m_{in} \\ & h_j(x,y) = 0 \quad \forall j = 1,...,m_{eq} \\ & x \in \mathbb{R}^{n_x^c}, y \in \mathbb{Z}^{n_y^d}, \end{aligned} \tag{2.2}$$

where $n_x^c + n_y^d = n$, $m_{in} + m_{eq} = m$. The computational complexity of this problem depends on the number of variables and constraints, the interaction pattern among the variables, and the form of the constraints and objective function, i.e., whether they are convex or nonconvex.

In previous work, we have proposed a graph representation of optimization problems [6]. Three types of graphs were proposed (see Fig. 2.3). The first is a bipartite variable-constraint graph, $\mathcal{G}_b(V_n, V_m, E)$ ($|V_n| = n, |V_m| = m$), which has two sets of nodes, one representing the constraints ($V_m$) and the other the variables ($V_n$). The edges $E$ capture the presence of a variable in a constraint. The second type of graph is the variable graph $\mathcal{G}_v(V_n, E_m)$ ($|V_n| = n$) where the nodes are the variables of the problem and the edges represent the constraints that couple the variables. The third type is the constraint graph $\mathcal{G}_c(V_m, E_m)$($|V_m| = m$) where the nodes are the constraints and the edges represent the variables that couple two constraints. This representation captures the structural coupling between the variables and the constraints, i.e. the presence or not of a variable in a constraint, as reflected in the adjacency matrix. Such graph representations have enabled the application of tools from network science and graph theory in order to "learn" the structure of an optimization problem which can be subsequently used as the basis for the application of decomposition-based solution algorithms [6, 207, 199, 201, 198]. Whereas a bipartite graph variable-constraint graph is the more general representation, the unipartite variable or constraint graphs may be better suited

to specific decomposition-based solution methods depending on whether complicating variables or constraints are involved [207, 199]. Yet, all these representations do not capture information about the type and domain of the variables or their functional form in the different constraints (linear, power law, etc.) in the case of nonlinear problems. We will refer to such information as functional coupling.

In this paper, we extend the aforementioned representation by incorporating a set of features for every node. These features capture the different characteristics of the node, such as the domain, and the upper and lower bound. Specifically, for a given graph $G(V, E)$ with $|V| = n_V$ (this can be any graph presented in Fig. 2.3), every node $n_i$ has a set of features $\phi_i \in \mathbb{R}^{n_\phi}$. Concatenation of these features forms the feature matrix $F \in \mathbb{R}^{n_V \times n_\phi}$. Under this representation, every graph is represented by three sets; the nodes $V$, the edges $E$, and the node features $F$, i.e., $G(V, E, F)$. This representation simultaneously captures the structural and functional coupling among the variables and the constraints of an optimization problem as presented in Fig. 2.4. We note that similar representations have been proposed in the literature [83, 122, 123, 175, 179, 215, 231], however, they consider the case of Mixed Integer Linear Programming problems and have been applied for tuning optimization solvers.

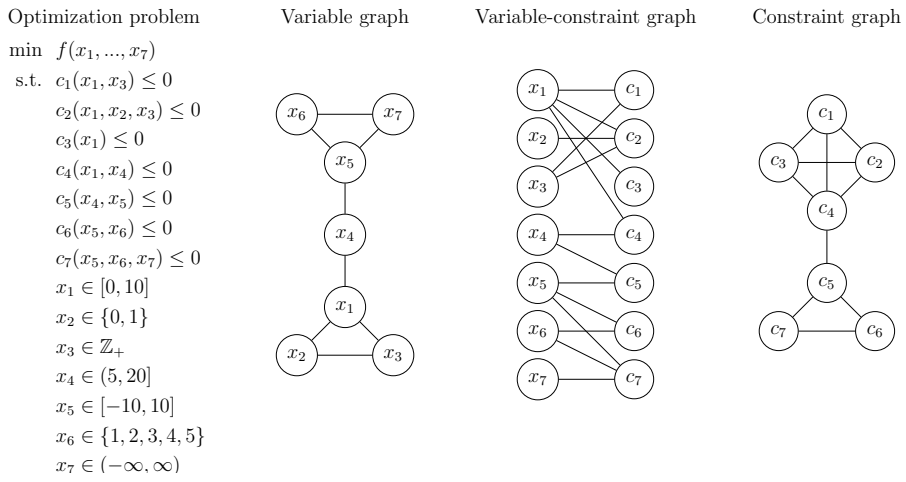


Figure 2.3: Graph representation of an optimization problem

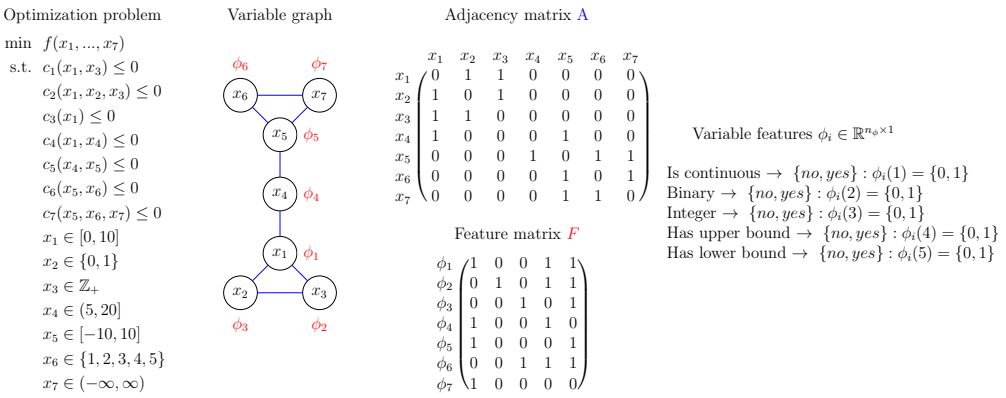


Figure 2.4: Graph and feature representation of an optimization problem

## 2.3 Learning when to decompose as an algorithm selection process

In this section, we pose the question of whether to decompose as an algorithm selection problem. Specifically, the set $\mathcal{P}$ represents the class of optimization problems which are considered, i.e., MILP, convex MINLP, etc. The set $\mathcal{A}$ has two algorithms, monolithic based $\alpha_M$ and decomposition-based $\alpha_D$, $\mathcal{A} = \{\alpha_M, \alpha_D\}$. Finally, the performance space is the solution time, i.e., $\mathcal{M} = \mathbb{R}_+$. Given these sets, we can determine whether to solve a given problem using a decomposition or monolithic-based solution algorithm by solving the following problem:

$$a^* \in \arg \min_{a \in \{\alpha_M, \alpha_D\}} m(P, a). \tag{2.3}$$

**Remark 2.1.** In this section we assumed that one monolithic and one decomposition-based solution method is available. However, in general, multiple monolithic and decomposition-based solution algorithms might be available. This scenario can be easily accommodated in the proposed framework by increasing the size of the possible algorithms $\mathcal{A}$.

**Remark 2.2.** In this section the performance function is the solution time. However, one can select another metric, such as the duality gap or the quality of the best feasible solution found after a fixed computational budget.

**Remark 2.3.** The question of whether to use a decomposition-based solution algorithm over a monolithic one can be posed for every class of problems. However, different

performance functions must be used for different classes of problems and algorithms, based on the available convergence guarantees. For example, one can consider whether a nonconvex MINLP should be solved using branch and bound or Generalized Benders Decomposition (GBD). Since the convergence of GBD is not guaranteed for this class of problems, the performance function can not be simply the solution time, since GBD can converge faster than branch and bound but the solution can be highly suboptimal.

## 2.4 Learning when to decompose via graph classification

In this section, we present the graph classification framework which will approximate the solution of the algorithm selection problem presented in Eq. 2.3. For illustration, we use the variable graph of an optimization problem. However, the constraint and bipartite variable constraint graphs can also be used.

### 2.4.1 Graph classification approach and architecture

Given an optimization problem P and the variable graph $\mathcal{G}_n(A, F, E)$ the goal is to develop a classifier $\mathcal{C} : F(\mathcal{P}) \times A(\mathcal{P}) \mapsto p \in \mathbb{R}^{N_a}$ to determine if problem P should be solved using a monolithic ($\alpha_M$) or decomposition-based ($\alpha_D$) solution approach ($\mathcal{A} = \{\alpha_M, \alpha_D\}, |\mathcal{A}| = N_a = 2$). The inputs in the classifier are the adjacency $A(P)$ and feature $F(P)$ matrices (which depend on the problem $P$), and the output is a vector $p \in \mathbb{R}^{N_a \times 1}$, where $p_i$ is equal to the probability that algorithm $\alpha_i$ solves the problem in the minimum computational time. Under this setting, the algorithm selection problem is transformed into a graph classification problem, where for a given problem $P$ the best solution strategy $\alpha^*$ is equal to

$$\alpha^* = \arg\min \{p_i\}_{i=1}^{N_a}, \tag{2.4}$$

where with $p = \mathcal{C}(F(P), A(P))$. Comparing Eq. 2.3 with Eq. 2.4 we observe that the graph classifier $\mathcal{C}(F(P), A(P))$ approximates $\min_{a \in \{\alpha_M, \alpha_D\}} m(P, a)$. The overall framework is presented in Fig. 2.6.

The prediction of the best solution strategy is performed by considering the exact structural and functional coupling among the variables and the constraints of the problem. This prediction has three steps; (1) message passing, (2) pooling, (3) final classification step.

The first step updates the features of a node (e.g., variable) by considering the features of the adjacent nodes, propagating information about the features of a node across the graph. The pooling layer, creates a graph-level feature which is used to characterize the whole graph based on the features of the individual nodes. The last step, performs the classification step based on the graph level features. In the rest of this section we present each step in detail using the variable graph as an example. However, both the constraint and the variable-constraint bipartite graphs can be used.

### 2.4.2 Message passing

Given the graph representation of the problem and the adjacency $A$ and feature matrices $F$, message passing is performed to update the features of the nodes using information from the neighbors. Specifically, given an optimization problem $P_i$ (see Eq. 2) with $N_v$ variables and $N_m$ constraints, we can generate the variable graph $G_v(V_v, E_v)(|V_v| = N_v)$ and obtain the adjacency $A \in \mathbb{R}^{N_v \times N_v}$ and feature $F \in \mathbb{R}^{N_v \times N_\phi}$ matrices where $N_\phi$ is the number of features per node. We define as $\phi_i$ the features of node i ($\phi_i$ is the $i^th$ row of matrix F). The features of node $i$ are updated using information about the features of the neighbor nodes $\mathcal{N}(i)$ of node $i$ as presented below [50]

$$h_i = \sigma\left(h_i, \bigoplus_{j \in \mathcal{N}(i)} \psi(h_i, h_j; W)\right), \tag{2.5}$$

where $\bigoplus$ denotes an aggregation function that is independent of the order of the neighbors (sum, average, max, etc.), $\psi$ is a message function that takes as input the features of node $i$ and the neighbor $j \in \mathcal{N}(i)$, $W$ as learnable weights, and $\sigma$ is an update function that returns the new features of node $i$. Based on the type of function $\psi$ and the aggregation function $\bigoplus$ that are used different graph neural network models have been proposed, such as Graph Convolutional Neural Networks (GCN) [157], Graph Attention Networks (GAT) [283] and GraphSage [127] (see [50] for a review).

Stacking $L$ such layers sequentially leads to a deep learning architecture where the features of a node in layer $l$ are given by:

$$\begin{aligned} h_i^1 &= \phi_i \\ h_i^{l+1} &= \sigma\left(h_i^l, \bigoplus_{j \in \mathcal{N}(i)} \psi(h_i, h_j; W^l)\right) \ \forall l = 1, .., L-1, \end{aligned} \tag{2.6}$$

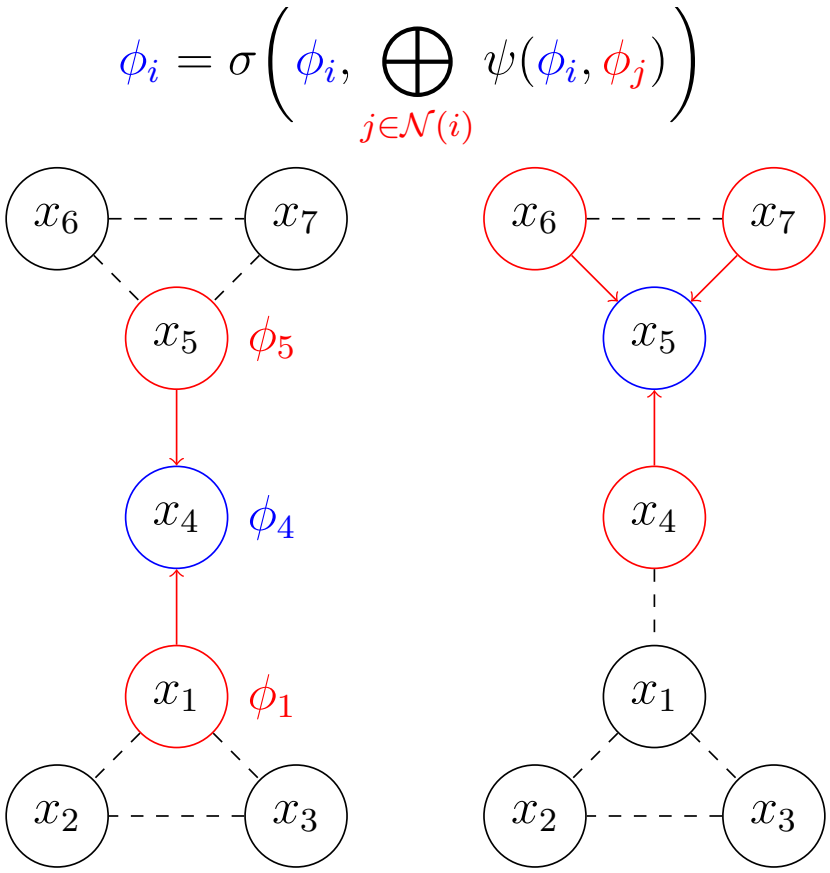


Figure 2.5: Message passing on the graph representation of the optimization problem

where $h_i^l \in \mathbb{R}^{N_h \times 1}$ $\forall l \geq 1$ are the features of node $i$ in level $l$, $W^l$ are the learnable weights at layer $l$, and $N_h$ is the dimension of the hidden features. An example of the message passing on the variable graph of the optimization problem is presented in Fig 2.5.

### 2.4.3 Pooling

The output of the last message passing layer is the original graph but the features of the nodes have been updated to $H$. At this stage, the feature matrix $H$ can not be used for classification, since the dimensions of $H$ depend on the number of the nodes (variables and constraints) in the graph of the optimization problem. To overcome this obstacle, a pooling layer is used to create a graph-level feature $r \in \mathbb{R}^{N_\phi}$. Different pooling functions

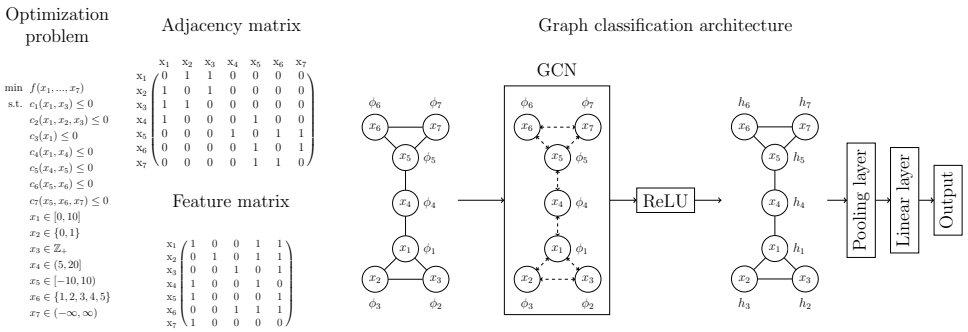


Figure 2.6: Learning when to decompose framework

can be used [50, 113], however, in this paper $r$ is equal to

$$r_i = \frac{1}{N_v} \sum_{j=1}^{N_v} H_{ij}^L \quad \forall i = 1, ..., N_v. \tag{2.7}$$

This pooling function returns the average value of every feature across all the nodes in the graph.

### 2.4.4 Final classification step

Finally, the graph level feature is an input to a linear transformation layer, where the output is $y \in \mathbb{R}^{N_a \times 1}$ ($y_i$ is the probability that algorithm $i$ solves the problem in the minimum computational time) and is equal to

$$y = \Theta r + b, \tag{2.8}$$

where $\Theta \in \mathbb{R}^{N_A \times N_h}$, $b \in \mathbb{R}^{N_A \times 1}$ are parameters.

### 2.4.5 Training

We assume that a set of optimization problems $\{P_i\}_{i=1}^{N_{data}}$ is available and for these problems we can obtain the variable graphs $\{G_i\}_{i=1}^{N_{data}}$, the adjacency and feature matrix $\{A_i, F_i\}_{i=1}^{N_{data}}$ and labels $\{z_i\}_{i=1}^{N_{data}}$ where $z_i$ is the algorithm that solves problem $P_i$ in the minimum computational time. Given these data, the task is to learn the classifier's parameters that will maximize its accuracy by optimizing some loss function, such as

cross-entropy. The learning problem is

$$\underset{\{W_i\}_{i=1}^{L-1},\theta,b}{\text{minimize}} \quad \mathcal{L}\left(z^{data}, z^{pred}; \{W_i\}_{i=1}^{L-1}, \theta, b\right) \tag{2.9}$$

**Remark 2.4.** This classifier can classify problems with different number of variables and constraints since the pooling layer after the last convolution layer creates the vector $r$ which has $N_h \times 1$ dimension. Therefore, for the final classification step, every problem is represented in the hidden features dimension space $N_h$.

**Remark 2.5.** In the case where multiple monolithic and decomposition -based solution methods are available the same architecture can be used, however, the resulting problem is a multiclass classification one.

## 2.5 Application to convex MINLP problems

In this section we implement the proposed approach to determine whether a convex MINLP problem should be solved using Branch and Bound (B&B) [121], a monolithic-based solution approach, or the Outer Approximation (OA) algorithm [86], a decomposition-based solution approach, as implemented in BONMIN [43].

### 2.5.1 Branch and bound

Branch and Bound can be used to solve a convex MINLP problem by branching on the binary variables. Given a convex MINLP as presented in Eq. 6.2 branch and bound starts by solving the continuous relaxation of the problem which is a convex NLP. We will denote the solution of this problem as $x^{NLP}, y^{NLP}$. Given this solution, branching on the binary variables is performed using some branching rules to select a binary variable $y_i$ and create two nodes in the branch and bound tree, one solved for $y_i$ fixed to $\lfloor y_i^{NLP} \rfloor$ and the other for $y_i$ equal to $\lceil y_i^{NLP} \rceil$. This procedure continues until the global optimal solution of the problem is found.

### 2.5.2 Outer Approximation algorithm

The Outer Approximation algorithm [86] is a decomposition-based solution algorithm that alternates between the solution of a MIP master problem and a nonlinear convex

optimization problem. Given a value of the binary variables, we obtain the subproblem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x, \bar{y}) \\ \text{subject to} \quad & g(x, \bar{y}) \leq 0 \\ & x \in \mathbb{R}^{n_x^c}. \end{aligned} \tag{2.10}$$

Given the optimal solution of the subproblem $\bar{x}$ for a fixed value of $y = \bar{y}$, the constraints and objective of the subproblem are approximated via hyperplanes as follows

$$\begin{aligned} & f(x, y) \geq f(\bar{x}, \bar{y}) + \nabla f(\bar{x}, \bar{y})^\top \begin{bmatrix} x - \bar{x} \\ y - \bar{y} \end{bmatrix} \quad \forall x \in \mathbb{R}^n, y \in \mathbb{Z}^{n_y} \\ & g(\bar{x}, \bar{y}) + \nabla g(\bar{x}, \bar{y})^\top \begin{bmatrix} x - \bar{x} \\ y - \bar{y} \end{bmatrix} \leq 0 \quad \forall x \in \mathbb{R}^{n_x^c}, y \in \mathbb{Z}^{n_y^d}. \end{aligned} \tag{2.11}$$

These approximations are generated iteratively as dictated by the master problem which is equal to

$$\begin{aligned} \underset{x,y}{\text{minimize}} \quad & \eta \\ \text{subject to} \quad & \eta \geq f(\bar{x}^q, \bar{y}^q) + \nabla f(\bar{x}^q, \bar{y}^q)^\top \begin{bmatrix} x - \bar{x}^q \\ y - \bar{y}^q \end{bmatrix} \quad \forall q \in \mathcal{Q} \\ & g(\bar{x}^q, \bar{y}^q) + \nabla g(\bar{x}^q, \bar{y}^q)^\top \begin{bmatrix} x - \bar{x}^q \\ y - \bar{y}^q \end{bmatrix} \leq 0 \ \forall l \in \mathcal{Q} \\ & x \in \mathbb{R}^n, y \in \mathbb{Z}^{n_y}, \end{aligned} \tag{2.12}$$

where $\mathcal{Q} = \{1, .., N_{\mathcal{Q}}\}$ denotes iteration number. The OA algorithm alternated between the solution of the master problem, which provides a lower bound and the subproblem which provides an upper bound. This is the standard application of the algorithm assuming the subprolbem is always feasible (see [86] for more details)

Table 2.1: Performance metrics for the graph classifier on the testing data set

| Accuracy | 0.9 | | | Confusion matrix | | |
|---|---|---|---|---|---|---|
| | | | | | Predicted label | |
| Label | Precision | Recall | F1 score | True label | OA | B&B |
| OA | 0.83 | 1.00 | 0.91 | OA | 15 | 0 |
| B&B | 1.00 | 0.80 | 0.89 | B&B | 3 | 12 |

### 2.5.3 Feature representation of the problem

We will consider the variable graph of an optimization and will consider five features per node. The features are

- Variable domain: continuous, binary, integer
- Upper bound
- Lower bound

We use one hot encoding to represent these features as follows:

- $\phi_1 \in \{0, 1\}$: 1 if the variable is continuous and 0 otherwise
- $\phi_2 \in \{0, 1\}$: 1 if the variable is binary and 0 otherwise
- $\phi_3 \in \{0, 1\}$: 1 if the variable is integer and 0 otherwise
- $\phi_4 \in \{0, 1\}$: 1 if the variable has an upper bound and 0 otherwise
- $\phi_5 \in \{0, 1\}$: 1 if the variable has a lower bound and 0 otherwise

Using these features the dimensions of the feature matrix are $N_v \times 5$.

### 2.5.4 Data gathering for classification

We use benchmark convex MINLP problems to train and test the classifier. For every problem $P_i$, we obtain the adjacency matrix $A_i$ using DecODe [207] and the feature matrix $F_i$ using the procedure presented in Algorithm 2.2. Every problem is solved with both algorithms (branch and bound and outer approximation) using BONMIN 1.8.8 [43] with a maximum computational time of 3000 seconds. All the other parameters of both parameters are set equal to their default values. From the 295 problems, 227 are solved with at least one solver (151 problems are solved faster with OA and 57 with B&B). Given the solution times, we use Algorithm 2.1 to obtain the label $z_i$ of every problem. Overall, we obtain the dataset $\mathcal{D} = \{(A_i, F_i), z_i\}_{i=1}^{N_{data}}$, where 66% of the data points have label OA and 34% B&B.

### 2.5.5 Graph classification architecture and implementation

We split the dataset $\mathcal{D}$ at random into a training set and a testing set. The training set has 197 data points and the testing set has 30 data points picked at random (15 random data points have label OA and 15 have label B&B). We perform hyperparameter optimization regarding the number of hidden features ($N_\phi = \{12,16,24,32,64,128\}$), the batch size ($\{10, 20, 40, 50\}$), and the learning rate ($\{0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05\}$).

Based on the hyperparameter optimization results, the graph classifier has four convolution layers ($L = 4$), a global mean pooling layer, a linear layer (with dropout probability equal to 0.5), the number of hidden features is 12 ($N_h = 12$), and `tahnh` is used as the activation function. For the training, we use the Adam algorithm [156] for 50 iterations with random initialization, the learning rate equal to 0.005, and batch size equal to 10. The loss function is the cross entropy and different weights are assigned to the two classes to account for the imbalance in the training dataset. The weights are computed as follows:

$$\omega_i = \frac{N_{data}}{N_\alpha} \sum_{j=1}^{N_{data}} \mathbb{1}_i(z_j))^{-1}, \tag{2.13}$$

where $\mathbb{1}_i$ is the indicator function, i.e., $\mathbb{1}_i(z_i) = 1$ if $z_j = i$ and 0 otherwise. The weight for class B&B is 1.6148 and for class OA is 0.7243. The GCN is implemented in PyTorch Geometric [92] and the training is done using PyTorch [226].

### 2.5.6 Graph classification results

The accuracy of the classifier on the testing dataset is presented in Table 2.1. From the results we observe that the accuracy of the classifier is 90% and all the problems with label $OA$, i.e., solved faster with the outer approximation algorithm, are classifier correctly and only 3 problems solved faster with branch and bound as misclassified. These results can be attributed to the small dataset since only 197 data points are used for leanring and the imbalance in the training set.

### 2.5.7 Automated algorithm selection for convex MINLP problems

The classification results show that the proposed approach can be used to determine the best solution strategy for a convex MINLP problem. In this section we present

```
from pyomo.environ import *
import torch, torch_geometric
# import optimization problem
from optimization_problem import model
# get features of the optimization problem
A,F = get_features(model)
# determine the best solver
best_solver = graph_classifier(A,F)
# solve the problem with the best solver
best_solver.solve(model)
```

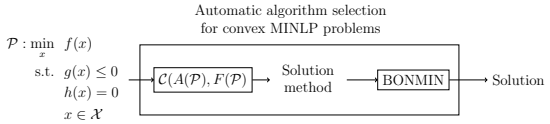


Figure 2.7: Automated algorithm selection for convex MINLP problems

how the graph classifier can be readily incorporated in current mixed integer optimization technology. Specifically, we consider the integration of the graph classifier with BONMIN and Pyomo [129] as presented in Fig. 2.7. Regarding the implementation in Pyomo, the graph classifier will predict the best solution strategy $\alpha^*$ which is either `B-BB` (branch and bound) or `B-OA` (outer approximation). The best solution strategy passed to BONMIN as follows `solver.options['algorithm']=`$\alpha^*$.

## 2.6 Conclusions and discussion

Decomposition-based solution algorithms have been widely used to solve complex and large optimization problems. However, their efficiency over monolithic methods is not known a priory. In this paper, we proposed a graph classification approach to determine whether an optimization problem should be solved using a monolithic or a decomposition-based solution method. In the proposed approach the prediction of the best solution strategy is performed by considering the detailed structural coupling among the variables and constraints of an optimization problem, therefore alleviating the need for a set of handcrafted features. Application of the proposed approach to convex MINLP problems shows that the proposed approach can predict whether such problems must be solved with a monolithic or a decomposition-based solution algorithm. Finally, we showed how the graph classifier can be easily integrated with optimization solvers, therefore leading to an automated algorithm selection for decomposition-based optimization algorithms.

The proposed approach requires data regarding the performance of the solution time or solution quality of optimization problems with different solvers. Such data can be obtained using the numerous available benchmark optimization problem [109, 2] and continuously enriched with data obtained from new problems that will be formulated and solved. The availability of such data will enable the application of this approach to other

**Data:** Optimization problem $\mathcal{P}$, Outer approximation algorithm ($OA$), Branch and Bound algorithm ($BB$), solution time limit $t_{max}$
**Result:** Label of product $\mathcal{P}$

**for** $i \in \{OA, BB\}$ **do**
  Solve problem $\mathcal{P}$ using algorithm $i$ with time limit $t_{max}$
  Store solution time $t_i$
**end**
sFind solver with minimum CPU time $i^* = \arg\min_{i \in \{OA,BB\}} \ t_i$
**if** $t_{i^*} \leq t_{max}$ **then**
  **if** $i^* = BB$ **then**
    $y = 1$
  **end**
  **if** $i^* = OA$ **then**
    $y = 0$
  **end**
**else**
  Problem not solved, data point not considered
**end**

**Algorithm 2.1:** Procedure to obtain the label of an optimization problem

classes of problems such as LP, MILP, convex MIQPs as future work. Finally, potential users such as process industries can build their own libraries of classes of problems of their interest. Process industries solve a wide range of optimization problems daily, such as production planning, production scheduling for different types of plants (continuous, batch, etc.), process control, real-time optimization, and supply chain management (planning and operation). Given the on-going digital transformation of the chemical industry, this kind of data can be easily stored and used to inform the tools that will be developed in the proposed research.

**Data:** Variable $v$
**Result:** Features $\phi$
Initialize the feature vector $\phi = [0\ 0\ 0\ 0\ 0]$;
**if** *$v$ is continuous* **then**
  Set $\phi(1) = 1$;
  Set $\phi(2) = 0, \phi(3) = 0$;
**else**
  Set $\phi(1) = 0$;
**end**
**if** *$v$ is binary* **then**
  Set $\phi(2) = 1$;
  Set $\phi(1) = 0, \phi(3) = 0, \phi(4) = 1, \phi(5) = 1$;
**else**
  Set $\phi(2) = 0$;
**end**
**if** *$v$ is integer* **then**
  Set $\phi(3) = 1$;
  Set $\phi(1) = 0, \phi(2) = 0, \phi(4) = 1, \phi(5) = 1$;
**else**
  Set $\phi(2) = 0$;
**end**
**if** *$v$ has upper bound* **then**
  Set $\phi(4) = 1$;
**else**
  Set $\phi(4) = 0$;
**end**
**if** *$v$ has lower bound* **then**
  Set $\phi(5) = 1$;
**else**
  Set $\phi(5) = 0$;
**end**
Get features $\phi = [\phi(1)\ \phi(2)\ \phi(3)\ \phi(4)\ \phi(5)]$;

**Algorithm 2.2:** Procedure to obtain the features of a variable

# Chapter 3
# Decomposition of integrated scheduling and dynamic optimization problems using community detection

## 3.1 Introduction

Process scheduling and control constitute the basis of the decision making pyramid. The goal of scheduling is to determine the optimal production decisions in order to satisfy the demand and maximize the profit. The task of process control is to maintain or track desired operating conditions and reject disturbances. Although the objectives of these tasks might seem different, their interaction can have a significant economic impact on the operation of a plant [75].

The first attempt to integrate scheduling and control was to incorporate the dynamic behavior of the system into the scheduling calculations as constraints [97]. In this approach a time slot formulation of the scheduling problem was used and the integrated problem was a mixed integer dynamic optimization problem (MIDO) which was transformed into a mixed integer nonlinear programming (MINLP) problem using orthogonal collocation on finite elements. This formulation was further extended to account for multiproduct parallel lines [98] and find optimal scheduling and control policies of polymerization reactors [274]. In these cases, the scheduling and control decisions are made simultaneously. This approach is called "top-down"[17] since open-loop optimal control is applied to find the optimal transition policies. The solution of this integrated problem is challenging due to the nonlinear behavior of chemical processes, and the fact that the scheduling time horizon is longer than the control horizon and thus the discretization of the dynamic model results in a high dimensional MINLP problem.

Two approaches have been proposed to reduce the computational time in such integrated problems. In the first approach the dynamic model is replaced by a surrogate model. Scale-bridging and Hammerstein-Weiner models have been used to describe the dynamic behavior of the system. These models are used as constraints in the scheduling calculations instead of the detailed dynamic model [85, 16, 228]. The dynamic behavior of the system has also been approximated by piece-wise affine models and based

on this representation multiparametric and fast MPC was used to close the control loop [306, 307]. A multiparametric approach has also been proposed where surrogate modeling is used to approximate the dynamic model [52]. Finally, manifold learning has been used to build low order models from high dimensional data using artificial neural networks [279]. In these approaches the goal is to reduce the complexity of the dynamic model that will be used in the scheduling calculations in order to reduce the computational time.

The second approach is to use high-fidelity dynamic models and exploit the structure of the integrated problem through decomposition-based solution algorithms [69]. The decomposition-based algorithms can be classified as distributed or hierarchical based on the interactions of the subproblems [76]. Lagrangean decomposition has been used to obtain solutions of integrated problems with highly nonlinear dynamic behavior where solving the entire problem monolithically is intractable [275]. This is a distributed optimization approach where all the subproblems are equivalent and the coordination of the subproblems is based on the difference of the shared variables among the subproblems. Examples of hierarchical optimization involve the application of Generalized Benders Decomposition (GBD) [64, 221] and bilevel decomposition [62] for the solution of the integrated problem. In this approach the system is comprised of a master (leader) and primal (follower) problem. The coordination in these cases is based on a hierarchy where the solution of the master problem provides the values of the shared variables in the primal problem. Then, based on the new information the master problem is solved again. Although decomposition-based algorithms can reduce the computational time for the solution of the integrated problem, a decomposition of the optimization problem itself is required and its choice can have a strong effect on the resulting solution. The decomposition of the optimization problem in the above cases was obtained by using the binary variables as shared.

In this work, the goal is to develop a systematic way to decompose the integrated scheduling and dynamic optimization problem. We will employ methods from network theory to analyze the structure of this problem. In recent work in our group we have proposed the use of network decomposition concepts and algorithms for distributed control and optimization [77]. Specifically for optimization problems we have proposed graph representations of the interconnections between variables and constraints [271, 6]. The application of community detection to such graphs can provide high-quality decompositions whereby groups of constraints or variables are identified with weak interactions (in

a statistical sense) among them. These groups can subsequently be used as the basis for decomposition-based solution algorithms. This process has been automated in the software package DeCODe [6]. In this paper we will use community detection to examine the community structure of the graph representation of integrated cyclic scheduling and dynamic optimization problems. We will show that such a framework allows identifying systematically a hybrid hierarchical/community structure in such problems, along with the set of corresponding shared variables, which can then be used effectively in a hierarchical optimization solution approach. The rest of the paper is organized as follows: In Section 2 the integrated scheduling and dynamic optimization problem is presented. In Section 3 community detection is applied to the unipartite constraint graph of the integrated optimization problem of an isothermal CSTR and the hierarchical structure of the communities is further examined. Based on the decomposition obtained from the community detection GBD is applied to solve the problem. Finally, in Section 4 the integrated optimization problem of more general systems is examined.

## 3.2 Problem formulation

The integrated cyclic scheduling and dynamic optimization problem is defined as follows: Given $N_p$ products, costs, demand rates, operating conditions and constraints find the optimal production sequence, amount of products to be manufactured, optimal transition profiles and minimum total operating time. The mathematical formulation presented below is based on [97, 275] and is composed of a scheduling and a dynamic optimization problem. Initially, each problem will be presented separately and then the integrated problem will be stated.

### 3.2.1 Scheduling problem

The main task of the scheduling problem is to decide on the sequence and amount of each product that will be manufactured. These decisions affect the profit, production and inventory cost. Hence, the objective of the scheduling problem is to maximize the profit and is given by the following equation [275]:

$$\sum_{i=1}^{N_p} \frac{C_i^p W_i}{T_c} - \sum_{i=1}^{N_p} \frac{C_i^s (G_i - W_i/T_c)\Theta_i}{2} \tag{3.1}$$

The first term defines the profit from the sales and the second term the inventory

cost. $W_i$ $(kg)$ is the amount of product $i$ manufactured, $C_i^p(\$/kg)$ is the selling price of product $i$, $C_i^s$ $(\$/(kg\ h))$ is the inventory cost of product $i$, $0.5(G_i - W_i/T_c)\Theta_i$ is the product accumulation over time [305], $G_i$ $(kg/hr)$ is the production rate of product $i$, $\Theta_i$ $(hr)$ is the production time of $i$ and $T_c$ is the total operating time. The objective function units are $(\$/hr)$. The scheduling time horizon is divided into $k$ slots using a time slot based formulation. We assume that each product is manufactured only once and that in each slot only one product can be manufactured. These constraints are imposed by the following equations:

$$\sum_{k=1}^{N_s} y_{ik} = 1 \quad \forall i \tag{3.2}$$

$$\sum_{i=1}^{N_p} y_{ik} = 1 \quad \forall k \tag{3.3}$$

where $y_{ik}$ is a binary variable which is equal to one if product $i$ is manufactured in slot $k$ and zero otherwise, $N_s$ is the number of slots and $N_p$ is the number of products. Since each product is manufactured only once, at the start of a slot the operating conditions correspond to the steady state values of the product that was manufactured in the previous slot. Thus, the slot is divided into a transition and a production regime. An illustration of the slot formulation and the two operating regimes is presented in Figure 3.1. Since the transition and production time depend on the dynamic characteristics of

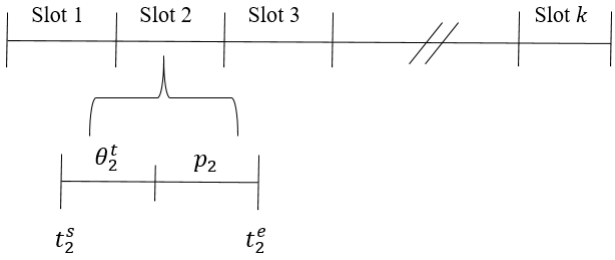


Figure 3.1: Discretization of time horizon into slots

the system, the duration of each slot is a variable captured by the starting $t_k^s$, ending $t_k^e$, production $p_k$ and transition time $\theta_k^t$. Equations 4-7 are then the timing relationships for each slot:

$$t_k^e = t_k^s + p_k + \theta_k^t \quad \forall k \tag{3.4}$$

$$t_k^s = t_{k-1}^e \quad \forall k \neq 1 \tag{3.5}$$

$$t_1^s = 0 \tag{3.6}$$

$$t_k^e \leq T_c \quad \forall k \tag{3.7}$$

Equation 7 is used since cyclic scheduling is assumed and the demand rate ($D_i$) of each product is constant. The production time of product $i$ in slot $k$ ($\theta_{ik}$) is bounded by $\theta^{max}$ and the total manufacturing time of product $i$ ($\Theta_i$) is given by Equation 9. The processing time $p_k$ in each slot is given by Equation 10 and the production rate of each product depends on the production process and can either be constant or time dependent (see Equation 11). Finally, the total production of product $i$ must satisfy the demand. These constraints are given below:

$$\theta_{ik} \leq \theta^{max} y_{ik} \quad \forall i, k \tag{3.8}$$

$$\Theta_i = \sum_{k=1}^{N_s} \theta_{ik} \quad \forall i \tag{3.9}$$

$$p_k = \sum_{i=1}^{N_p} \Theta_i \quad \forall k \tag{3.10}$$

$$W_i = G_i \Theta_i \quad \forall i \tag{3.11}$$

$$W_i \geq D_i T_c \quad \forall i \tag{3.12}$$

Combining the aforementioned constraints and objective function the scheduling problem is:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{N_p} \frac{C_i^p W_i}{T_c} - \sum_{i=1}^{N_p} \frac{C_i^s (G_i - W_i/T_c)\Theta_i}{2} \\ \text{subject to} \quad & \text{Equations 3.2-3.12} \end{aligned} \tag{3.13}$$

This formulation of the scheduling problem results in a mixed integer nonlinear programming problem due to the binary assignment variables ($y_{ik}$) and the nonlinear objective function.

### 3.2.2 Dynamic optimization problem

A continuous production system can be modeled by the following general system of differential equations:

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{u})$$

where $\mathbf{x} = [x_1, ..., x_n]$, $\mathbf{u} = [u_1, ..., u_m]$ are the states and the manipulated variables of the system. We assume that multiple products can be manufactured, thus the operating conditions of the plant can change over time. This frequent transitional operating policy requires the usage of a control system that can track the time varying set points and minimize the transition cost between products. A dynamic optimization problem for this system can be stated as:

$$\begin{aligned} \underset{\mathbf{u}}{\text{minimize}} \quad & J(u) \\ \text{subject to} \quad & \dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{u}) \\ & \mathbf{x}(t_0) = \mathbf{x}_0 \\ & \mathbf{x}(t_f) = \mathbf{x}^{\mathbf{sp}} \\ & \mathbf{u}(t_0) = \mathbf{u}_0 \\ & \mathbf{u}(t_f) = \mathbf{u}^{\mathbf{sp}} \\ & \mathbf{x}_{lb} \leq \mathbf{x}(t) \leq \mathbf{x}_{ub} \\ & \mathbf{u}_{lb} \leq \mathbf{u}(t) \leq \mathbf{u}_{ub} \end{aligned} \tag{3.14}$$

where $J(u)$ is the transition cost and $\mathbf{x}^{sp}, \mathbf{u}^{sp}$ are the set point values for the states and the manipulated variables. Concerning the integration with scheduling, the control actions should minimize the transition cost.

### 3.2.3 Integration of scheduling and dynamic optimization

The integrated problem combines the scheduling and dynamic optimization problem resulting in a mixed integer dynamic optimization problem. It is usually solved using a simultaneous solution approach and the method of orthogonal collocation on finite elements [36]. Since a transition occurs in each slot, the transition time is discretized in $N_f$ finite elements and each element has $N_c$ collocation points. The discretized

differential equations are:

$$x_{nfck} = x0_{nfk} + h_k \sum_{l=1}^{N_c} \Omega_{lc} \dot{x}_{nflk} \quad \forall n, f, c, k \tag{3.15}$$

$$x0_{nfk} = x0_{nf-1,k} + h_k \sum_{l=1}^{N_c} \Omega_{l,N_{cp}} \dot{x}_{nf-1,l,k} \quad \forall n, f \geq 2, k \tag{3.16}$$

$$\dot{x}_{nfck} = F_n(x_{1fck}, ...x_{nfck}, u_{1fck}, ..., u_{mfck}) \quad \forall n, f, c, k \tag{3.17}$$

$$t_{fck} = (f - 1 + \gamma_c)h_k \quad \forall f, c, k \tag{3.18}$$

$$h_k = \theta_k^t / N_f \quad \forall k \tag{3.19}$$

$$x_n^{lb} \leq x_{nfck} \leq x_n^{ub} \quad \forall n, f, c, k \tag{3.20}$$

$$u_m^{lb} \leq u_{mfck} \leq u_m^{ub} \quad \forall m, f, c, k \tag{3.21}$$

where $x_{nfck}$, is the value of the $n^{th}$ state at finite element $f$, collocation point $c$ at slot $k$, $u_{mfck}$ is the value of the $m^{\text{th}}$ manipulated variable at finite element $f$, collocation point $c$ at slot $k$, $x0_{nfk}$ is the value of the $n$ state at the start of the finite element $f$ in slot $k$, $\Omega$ is the collocation matrix [97] and $\gamma_c$ is the $c^{th}$ Radau root [37]. In the integrated problem the set point values of the states and manipulated variables for each slot are decision variables and are calculated as follows:

$$x_{nk}^{in} = \sum_{i=1}^{N_p} x_{ni}^{ss} y_{i,k-1} \quad \forall n, k \neq 1 \tag{3.22}$$

$$x_{n1}^{in} = \sum_{i=1}^{N_p} x_{ni}^{ss} y_{i,N_{slot}} \quad \forall n, k = 1 \tag{3.23}$$

$$u_{mk}^{in} = \sum_{i=1}^{N_p} u_{mi}^{ss} y_{i,k-1} \quad \forall m, k \neq 1 \tag{3.24}$$

$$u_{m1}^{in} = \sum_{i=1}^{N_p} u_{mi}^{ss} y_{i,N_{slot}} \quad \forall m, k = 1 \tag{3.25}$$

$$x_{nk}^{end} = \sum_{i=1}^{N_p} x_{ni}^{ss} y_{i,k} \quad \forall n, k \tag{3.26}$$

$$u_{mk}^{end} = \sum_{i=1}^{N_p} u_{mi}^{ss} y_{i,k} \quad \forall m, k \tag{3.27}$$

where $x_{ni}^{ss}$ is the steady state value of state $n$ when product $i$ is produced and $u_{mi}^{ss}$ is the steady state value of the manipulated variable $m$ when product $i$ is produced. Equations 2.23 and 2.25 are used because cyclic scheduling is assumed, thus the starting operating conditions at the first slot are equal to the operating conditions at the end of the last slot. The values of the states and manipulated variables at the start (finite element 1, collocation point 1) and end (finite element $N_f$, collocation point $N_c$) of each slot are equal to their set point:

$$x0_{n1k} = x_{nk}^{in} \quad \forall n, k \tag{3.28}$$

$$u_{m11k} = u_{mk}^{in} \quad \forall m, k \tag{3.29}$$

$$x_{nN_fN_ck} = x_{nk}^{end} \quad \forall n, k \tag{3.30}$$

$$u_{mN_fN_ck} = u_{mk}^{end} \quad \forall m, k \tag{3.31}$$

The objective of the control problem is to minimize the transition cost. Thus, the objective function is a weighted average of the manipulated variables for each finite element and slot and is given by the following equation [275]:

$$\sum_{m=1}^{M} \frac{C^m}{T_c} \sum_{k=1}^{Ns} \sum_{f=1}^{N_f} h_{fk} \theta_k^t \sum_{c=1}^{N_c} u_{mfck} \gamma_c \tag{3.32}$$

where $C^m$ is the cost of the $m^{th}$ manipulated variable. Finally, bounds on the value of the state and manipulated variables are imposed:

$$x_n^{lb} \leq x_{nfck} \leq x_n^{ub} \quad \forall n, f, c, k \tag{3.33}$$

$$u_m^{lb} \leq u_{mfck} \leq u_m^{ub} \quad \forall m, f, c, k \tag{3.34}$$

Overall, the integrated optimization problem is:

$$\begin{aligned} \text{maximize} \quad & \left( \sum_{i=1}^{N_p} \frac{C_i^p W_i}{T_c} - \sum_{i=1}^{N_p} \frac{C_i^s (G_i - W_i/T_c)\Theta_i}{2} - \sum_{m=1}^{M} \frac{C^m}{T_c} \sum_{k=1}^{Ns} \sum_{f=1}^{N_f} h_{fk} \theta_k^t \sum_{c=1}^{N_c} u_{mfck} \gamma_c \right) \\ \text{subject to} \quad & \text{Equations} \quad 3.2 - 3.12 \\ & \text{Equations} \quad 3.15 - 3.31, 3.33, 3.34 \end{aligned} \tag{3.35}$$

This is a mixed integer nonlinear problem due to the binary assignment variables, the nonlinear objective function and the nonlinear discretized differential equations.

## 3.3 Decomposition of the integrated optimization problem and decomposition-based solution

With the formulation presented above, finding a solution for this problem, even for a simple case, is a challenging task. The computational complexity of the problem depends on the number of products that can be manufactured and the nonlinearities of the dynamic model. The number of products affects the number of the binary variables and highly nonlinear systems require finer discretization of the dynamic model. Decomposition-based solutions [64, 275] may reduce the computational time, however a decomposition of the optimization problem is required. In this section, the integrated optimization problem will be decomposed using community detection and solved using Generalized Benders Decomposition. The community detection is applied to the constraint graph of the integrated optimization problem [6]. The constraint graph is composed of nodes which are the constraints of the optimization problem. The edges are the variables that link two constraints. Each edge has a weight which is equal to the number of shared variables between two constraints. The goal of the community detection algorithm is to divide a graph into communities such that the intraconnections are higher than the interconnections between the communities. Since the weight of the edges represent the number of variables between two constraints, the minimization of the interconnections in the constraint graph results in weakly connected subproblems. From the community detection results, the group membership of each constraint is obtained and the shared variables are detected, i.e. variables that belong to constraints with different group membership. The structure of the optimization problem can also be examined from the variable graph of the problem. In this case the nodes are the variables of the optimization problem and the edges are the constraints that connect two variables. The community detection is performed using the Louvain algorithm [42] in NetworkX [126], a package for network operations in Python. Initially, the cyclic scheduling and dynamic optimization problem of an isothermal CSTR will be used as a case study to illustrate the community-based decomposition and solve the problem. In Section 4 the decomposition of more general systems will be discussed.

### 3.3.1 Isothermal CSTR

**Community detection on the unipartite constraint graph**

In this case study a multiproduct CSTR is considered with an irreversible reaction 3A→B. The production cost and operating data are taken from [97]. We assume that five products can be manufactured and the system is modeled by the following differential equation:

$$\frac{dc(t)}{dt} = \frac{Q(t)}{V}(c_{feed} - c(t)) - kc(t)^3 \tag{3.36}$$

where $c$ (mol/L) is the concentration of the reactant, $Q$ (L/h) is the inlet flowrate and the volume $V$, the rate constant $k$ and inlet concentration of the reactant $c_{feed}$ are constant. In this system the manipulated variable is the inlet flowrate and the system has one state, the concentration of the reactant. The production rate of each product is constant and is calculated based on the steady state conversion of the reactant. The dynamic model is discretized using 20 finite elements with 3 collocation points. For five products, one state and one manipulated variable, the number of variables and constraints are 1411 and 1115, respectively. The result of the community detection on the unipartite constraint graph of the system is shown in Figure 3.2.

In the constraint unipartite graph six communities are identified. One community corresponds to the scheduling subproblem ($S$) and the other five are the dynamic optimization subproblems in each slot ($D_k$). Each dynamic optimization subproblem is directly connected only with the scheduling subproblem. The shared variables between the scheduling subproblem and the dynamic optimization subproblem in slot $k$ are:

$$c_k^{in},\ c_k^{end},\ Q[1,1,k],\ Q[20,3,k],\ \theta_k^t$$

The shared variables obtained from the community detection on the unipartite constraint graph are the set points for the concentration at the start $c_k^{in}$ and end $c_k^{end}$ of each slot, the inlet flowrate at the start $Q[1,1,k]$ (finite element 1, collocation point 1 in slot $k$) and the end $Q[20,3,k]$ (finite element 20, collocation point 3, slot $k$) of each slot and the transition time $\theta_k^t$ at slot $k$. All the shared variables in this case are continuous. This set of shared variables differs from the one used in literature [275, 64] where the set of shared variables includes the binary assignment variables. The conclusion reached from the community detection on the constraint unipartite graph can be seen

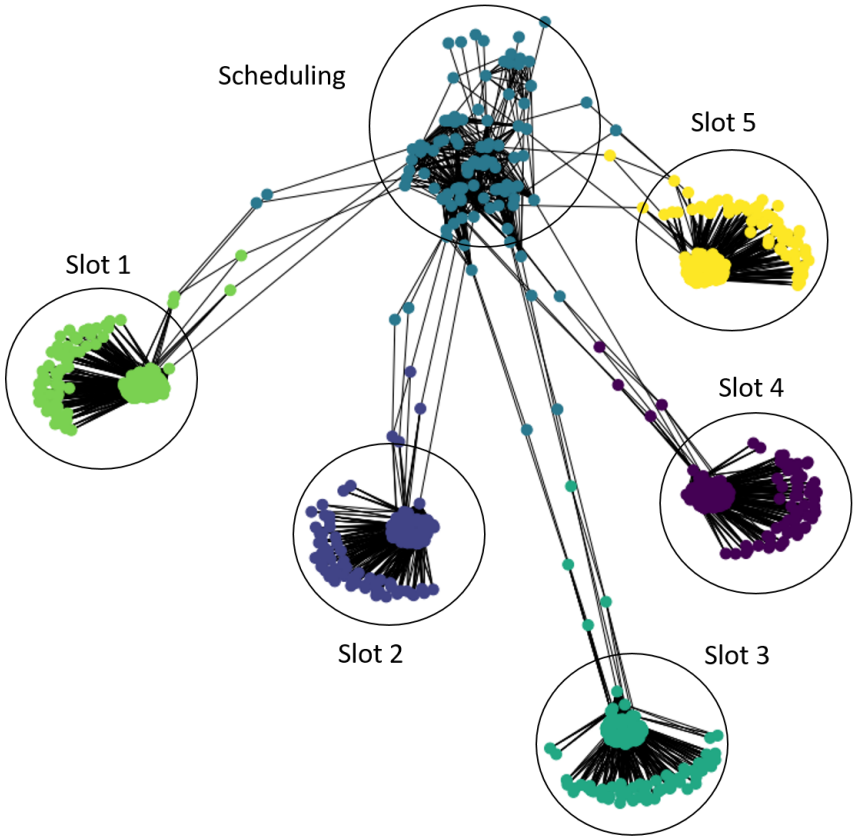


Figure 3.2: Unipartite constraint graph community detection results

also by considering the variable graph. Figure 3.3 shows specifically the interaction of the variables between the scheduling and dynamic optimization subproblem in slot $k$. The two subproblems are coupled through the red nodes, which are the shared variables obtained from the community detection on the constraint unipartite graph of the integrated optimization problem.

### Hierarchical structure of the communities

The results of the community detection on the constraint unipartite graph reveal the community structure of the integrated optimization problem. In this part two metrics from graph theory, closeness and betweenness centrality [219] will be evaluated in order to further analyze the structure of the constraint graph. Closeness centrality is

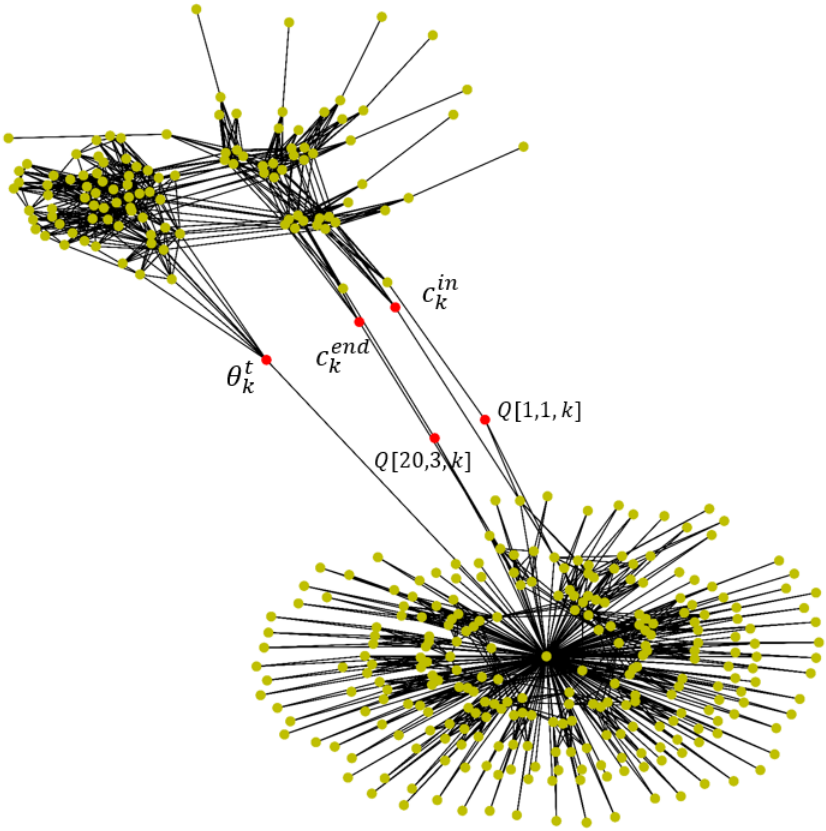


Figure 3.3: Variable unipartite graph presenting the interaction of the scheduling and dynamic optimization variables in slot $k$

proportional to the distance of a node from the rest of the graph and betweenness centrality measures the extent to which a node lies in the shortest paths between other nodes. These metrics quantify the importance of a node in a graph since nodes with high centrality connect the different parts of the graph. Details of the calculation of the centralities can be found in Appendix. For each node its centrality is calculated and based on the group membership of the nodes, the average centrality of the communities is computed and presented in Table 1. We note that the centrality of the scheduling subproblem is higher than that of the dynamic optimization subproblems. Also, the centralities of the dynamic optimization subproblems have the same value. These results indicate that the graph has a hierarchical structure. From a graph theory perspective, this structure emerges since the shortest path from any node that belongs

Table 3.1: Average closeness and betweenness centrality for the constraint unipartite graph

| Subproblem | Closeness centrality | Betweenness centrality |
|---|---|---|
| Scheduling | 0.24 | 0.021 |
| Dynamic opt. slot 1 | 0.18 | 0.002 |
| Dynamic opt. slot 2 | 0.18 | 0.002 |
| Dynamic opt. slot 3 | 0.18 | 0.002 |
| Dynamic opt. slot 4 | 0.18 | 0.002 |
| Dynamic opt. slot 5 | 0.18 | 0.002 |

in Slot 1 (see Figure 2) to any node in Slot 2 must visit at least one node that belongs in the scheduling subproblem. Thus, we can relate this hierarchical structure of the graph with a hierarchical structure in the optimization problem, where the scheduling subproblem is in the first level and the dynamic optimization subproblems in the second level.

**Decomposition-based solution of the integrated problem**

In this section the decomposed problem obtained from the community detection will be solved. The data for the problem are presented in Tables 2 and 3 and the cost of the inlet feed is 15 \$/L.

Table 3.2: Steady state and cost data for the multiproduct CSTR [97]

| Product | $Q^{ss}(L/h)$ | $c^{ss}(mol/L)$ | Demand $(kg/h)$ | Product price $(\$/kg)$ | Inventory cost (\$) |
|---|---|---|---|---|---|
| A | 10 | 0.0967 | 3 | 200 | 1 |
| B | 100 | 0.2 | 8 | 150 | 1.5 |
| C | 400 | 0.3032 | 10 | 130 | 1.8 |
| D | 1000 | 0.393 | 10 | 125 | 2 |
| E | 2500 | 0.5 | 10 | 120 | 1.7 |

Table 3.3: Data of the dynamic problem

| | |
|---|---|
| States: | $c(t)$ |
| Manipulated variable: | $Q(t)$ |
| Number of finite elements: | 20 |
| Number of collocation points: | 3 |

Based on the hierarchy identified above, GBD [108] is used in order to solve the

integrated problem. In the GBD algorithm the master problem (scheduling) is the first level of the hierarchy and provides target values for the states, manipulated variables and operating constraints like the transition time in each slot. The second level of the hierarchy is composed of the dynamic optimization problem in each slot. Thus the master problem provides the set point values for the states and the manipulated variables and the available transition time. Given these constraints a dynamic optimization problem is solved for each slot and based on the transition cost a Benders cut is added to the master problem. From the results of the community detection, the proposed set of shared variables does not provide a separable objective function as the total operating time that affects the transition cost belongs to the scheduling subproblem. Thus, we will use the total operating time as a shared variable in order to decompose the objective function. Also, due to equations 29 and 31 the value of the inlet flowrate $Q$ at the start (finite element 1, collocation point 1) and end (finite element 20, collocation point 3) are equal to the inlet flowrate set points. Thus the set points $q_k^{in}$ and $q_k^{end}$ are used as shared variables. Overall, the shared variables between the scheduling problem ($S$) and the dynamic optimization problem in slot $k$ ($D_k$) are:

$$(S, D_k) \ : \ c_k^{in},\ c_k^{end},\ q_k^{in}, q_k^{end},\ \theta_k^t,\ T_c$$

The total number of shared variables between the scheduling subproblem and the dynamic optimization subproblem in slot $k$ are six. The integrated problem ($I$) is described by the following general MINLP problem:

$$\begin{aligned} \text{maximize} \quad & f(\mathbf{w}, \mathbf{y}, \mathbf{z}) \\ \text{subject to} \quad & h_{sc}(\mathbf{z}, \mathbf{y}) = 0 \\ & g_{sc}(\mathbf{z}, \mathbf{y}) = 0 \\ & h_{dyn}(\mathbf{z}, \mathbf{w}) = 0 \\ & g_{dyn}(\mathbf{z}, \mathbf{w}) = 0 \\ & \mathbf{w} \in W, \mathbf{y} \in Y, \mathbf{z} \in Z \end{aligned} \tag{3.37}$$

where $\mathbf{z}$ are the shared variables, $\mathbf{y}$ are variables that belong only to the scheduling problem and $\mathbf{w}$ are variables that belong only to the dynamic optimization problem. Also, $h_{sc}$ and $g_{sc}$ are constraints that have scheduling and shared variables and $h_{dyn}$ and $g_{dyn}$ are constraints that contain shared and dynamic optimization variables. The

master problem $(M)$ is:

$$\begin{aligned} \underset{\eta,\mathbf{x},\mathbf{y}}{\text{maximize}} \quad & \eta \\ \text{subject to} \quad & \eta \leq L(\mathbf{w}^k, \mathbf{y}, \mathbf{z}, \boldsymbol{\lambda}^k, \boldsymbol{\mu}^k) \\ & h_{sc}(\mathbf{z}, \mathbf{y}) = 0 \\ & g_{sc}(\mathbf{z}, \mathbf{y}) \leq 0 \\ & \mathbf{z} \in Z, \quad \mathbf{y} \in Y \end{aligned} \tag{3.38}$$

where $L(\mathbf{w}^k, \mathbf{y}, \mathbf{z}, \boldsymbol{\lambda}^k, \boldsymbol{\mu}^k) = f(\mathbf{w}^k, \mathbf{y}, \mathbf{z}) + (\boldsymbol{\lambda}^k)^T h_{dyn}(\mathbf{w}^k, \mathbf{z}) + (\boldsymbol{\mu}^k)^T g_{dyn}(\mathbf{w}^k, \mathbf{z})$. The superscript $k$ in this equation is the iteration number in the GBD algorithm and $\boldsymbol{\lambda}$ , $\boldsymbol{\mu}$ are the Lagrange multipliers for the equality and inequality constraints in the primal problem. The primal problem $(P)$ is:

$$\begin{aligned} \underset{\mathbf{w}\in W}{\text{maximize}} \quad & f(\mathbf{w}, \mathbf{y}^k, \mathbf{z}^k) \\ \text{subject to} \quad & h_{dyn}(\mathbf{w}, \mathbf{z}^k) = 0 \\ & g_{dyn}(\mathbf{w}, \mathbf{z}^k) \leq 0 \end{aligned} \tag{3.39}$$

We use the second variant of the Generalized Benders decomposition [99] and we assume that the solution of the primal problem is the same as the solution of the following problem:

$$\min_{\mathbf{w}} L(\mathbf{w}, \mathbf{y}, \mathbf{z}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\mathbf{w}^k, \mathbf{y}, \mathbf{z}) + (\boldsymbol{\lambda}^k)^T h_{dyn}(\mathbf{w}^k, \mathbf{z})(\boldsymbol{\mu}^k)^T g_{dyn}(\mathbf{w}^k, \mathbf{z})$$

The steps of the Benders decomposition algorithm are:

1. Initialize the master problem solving the original problem without any cuts;
2. Given $y$ solve the subproblems and obtain a lower bound (LB);
3. Add Benders cut to the master problem;
4. Solve the master problem and obtain an upper bound (UB);
5. Check if $|UB - LB| \leq \epsilon$ or $UB < LB$ [99], if true, stop otherwise, go to step 2

Before applying Benders decomposition we change the master problem as follows: The objective of the scheduling problem is to maximize the hourly profit. Since the total

operating time is in the denominator of the objective function, minimizing the operating time contributes to maximizing the profit. From Equations 4,5,7, a lower bound on the total operating time can be obtained:

$$t_k^e = t_k^s + p_k + \theta_k^t \rightarrow t_{N_s}^e = \sum_{k=1}^{N_s} p_k + \theta_k^t$$

$$t_{N_s}^e \leq Tc \rightarrow Tc \geq \sum_{k=1}^{N_s} p_k + \theta_k^t$$

In the scheduling subproblem of the decomposed problem the transition time in each slot is a variable. Since the total operating time ($T_c$) must be minimized, the optimal solution is to set all the transition times equal to zero. However, the dynamic optimization subproblems will be infeasible for this choice of the shared variables $\theta_k^t$. In this case, a relaxed primal subproblem must be solved and a feasibility cut must be added to the master problem. In order to decrease the number of iterations, a lower bound on the transitions time is added. This bound is obtained by solving the following dynamic optimization problem [64]:

$$\begin{aligned} \theta_{ip}^{min} = \arg\min_{\theta^t} \quad & \theta^t \\ \text{subject to} \quad & \dot{c} = \frac{Q(t)}{V}(c_{feed} - c(t)) - kc(t)^3 \\ & c(0) = c_{in} \\ & c(\theta^t) = c_{end} \\ & Q(0) = q_{in} \\ & Q(\theta^t) = q_{end} \\ & c_{lb} \leq c \leq c_{ub} \\ & Q_{lb} \leq Q \leq Q_{ub} \end{aligned} \tag{3.40}$$

where the objective of this problem is to find the minimum time $\theta_{ip}^{min}$ for a transition from product $i$ to $p$, $c_{in}$, $c_{end}$, $q_{in}$, $q_{end}$ are the values of the states and manipulated variables at the start and end of the time horizon. The lower bound is added as a

constraint in the master (scheduling) problem using Equation 41 [63]:

$$\theta_k^t \geq \sum_{i=1}^{N_p} \sum_{p=1}^{N_p} z_{ipk} \theta_{ip}^{min} \quad \forall k \tag{3.41}$$

where $z_{ipk}$ are calculated as [97]:

$$z_{ipk} \geq y'_{ik} + y_{pk} - 1 \; \forall \; i, p = i, k \tag{3.42}$$

$$y'_{ik} = y_{i,k-1} \; \forall \; k \; \neq 1 \tag{3.43}$$

$$y'_{i,1} = y_{i,5} \tag{3.44}$$

Equations 42-44 are used in order to define a transition from product $i$ to product $j$ in slot $k$. Also, we add operating constraints in the primal subproblems:

$$u_{f,c,k} - u_{f,c-1,k} \leq u_{cont}^{c} \forall f, k, c \neq 1 \tag{3.45}$$

$$u_{f,c,k} - u_{f,c-1,k} \geq -u_{cont}^{c} \forall f, k, c \neq 1 \tag{3.46}$$

$$u_{f,1,k} - u_{f-1,N_{cp},k} \leq u_{cont}^{f} \forall k, f \neq 1 \tag{3.47}$$

$$u_{f,1,k} - u_{f-1,N_{cp},k} \geq -u_{cont}^{f} \forall k, f \neq 1 \tag{3.48}$$

Equations 45,46 bound the change of the manipulated variable between the collocation points, while Equations 47,48 between finite elements. The maximum rate of change between collocation points is 200 $L/h$ ($u_{cont}^{c}$) and between finite elements is 500 $L/h$ ($u_{cont}^{f}$). The detailed formulation of the master and primal subproblems are presented on Appendix A. The master problem is an MINLP problem and is solved using BARON [254]. The dynamic optimization subproblems are solved with IPOPT [285]. The algorithm was implemented in Python using Pyomo [129]. In this case study the master problem is a nonconvex MINLP and the subproblems are nonconvex NLPs thus global optimality of the solution can not be guaranteed [13, 256].

**Benders Decomposition results**

The initial guess for the sequence was B→ E→ D→ A→ C and the initial guess for the values of the state variables in the dynamic optimization problem is obtained by solving a feasibility problem for the entire integrated problem. The algorithm converges

after 3 iterations and the optimal sequence found is A→ B→ D→ E→ C. The objective function is 17260 $\$/h$ and the total operating time is 109.37 hours. The solution was obtained after 50 CPU seconds. The absolute tolerance between the upper and lower bound was $10^{-3}$. The production results of the problem can be found in Table 4 and

Table 3.4: Results of the integrated problem

| Slot | Product | $W_i(kg)$ | Starting time $(h)$ | Trans. time $(h)$ | Prod. time $(h)$ | End. time $(h)$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 328 | 0 | 24.96 | 36.32 | 61.28 |
| 2 | 2 | 875 | 61.28 | 0.17 | 10.93 | 72.38 |
| 3 | 4 | 1093 | 72.38 | 0.42 | 1.80 | 74.6 |
| 4 | 5 | 35649 | 74.6 | 0.37 | 28.52 | 103.49 |
| 5 | 3 | 1093 | 103.49 | 1.92 | 3.93 | 109.37 |

the evolution of the lower and upper bound with the number of iterations is presented in Figure 4 while the concentration and inlet flowrated are presented in Figure 5. From Table 4, all the demands are satisfied and the amount of product 5 manufactured is higher than the demand since we did not impose an upper bound on the amount of each product that is manufactured.

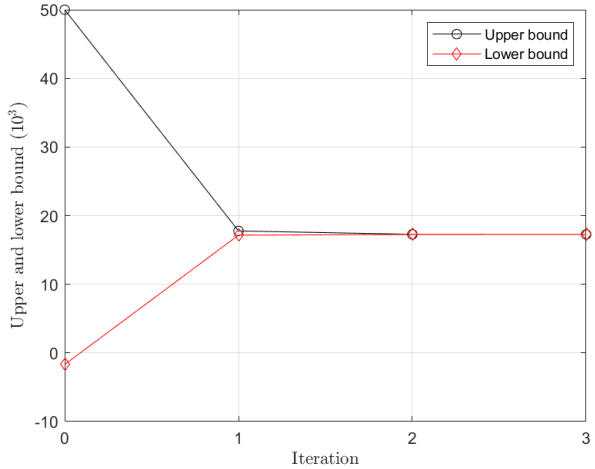


Figure 3.4: Evolution of upper and lower bound

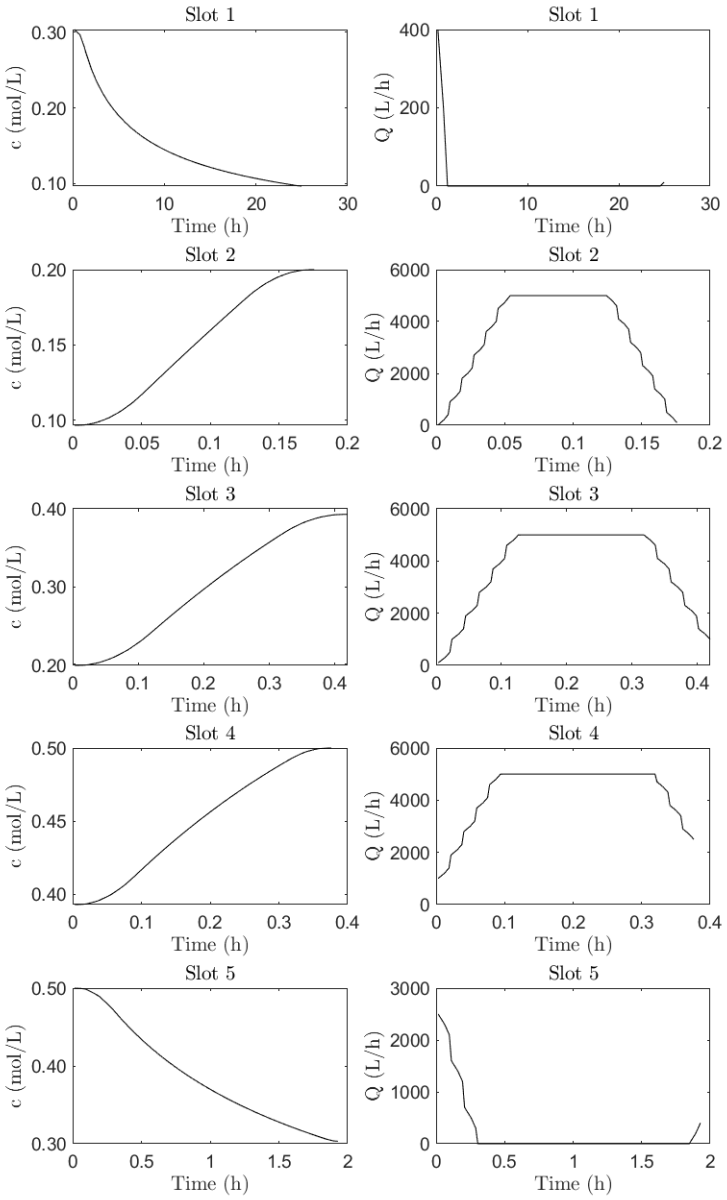


Figure 3.5: Concentration ($c$) and inlet flowrate ($Q$) profile in each slot during the transition regime

From the transition profiles in Figure 3.5 we can draw the following conclusions. The transition in slot 1, from low to high conversion is slow since the transition time is 24.96 hours, while transition from high to low conversion is fast (slot 2). In order to achieve these fast transitions the value of the manipulated variable is increased from the initial value to its upper bound for a short amount of time and then is reduced to its steady state value. The changes of the manipulated variables are not instantaneous due to the operating constraints that were imposed on each primal subproblem. The value of the manipulated variable reaches its maximum value and then is reduced; this excess usage of feed is due to the difference in the cost of the feed (15\$/L) and the lowest price of the product (120\$/$kg$ for product E). Also, this action reduced the total operating time. Concerning the computational time, solving the primal problem (MINLP) with BARON required 50 CPU seconds. The computational time of solving the subproblems using IPOPT was less than five CPU seconds. Finally, trying to solve the entire problem with BARON, after 9000 seconds the gap between the lower and upper bound is 67.5%.

## 3.4 Decomposition of the integrated problem for more general production systems

In this section the integrated problem will be decomposed using community detection for different production systems. The scheduling problem remains the same in all the cases, specifically, we will assume that five products can be manufactured. In general, the relative values of the cost data and steady state values of the states for each product do not affect the result of the decomposition, since it is based on the structural interactions between the constraints and the variables, thus relative cost data are not presented. Finally, in the cases below, the integrated dynamic optimization problem will again be discretized using orthogonal collocation on finite elements using 20 finite elements with 3 collocation points.

### 3.4.1 Non isothermal CSTR

We consider a non isothermal CSTR with two states (concentration and temperature). The system is modeled using the following system of ordinary differential equations

[275]:

$$\begin{aligned}\frac{dc(t)}{dt} &= \frac{1-c(t)}{\tau} - k_{10}e^{-ER/T(t)}c(t) \\ \frac{dT(t)}{dt} &= \frac{T_f - T(t)}{\tau} + k_{10}e^{-ER/T(t)}c(t) - \alpha Q(t)(T(t)-T_c)\end{aligned} \tag{3.49}$$

where $T$ and $c$ are the scaled temperature and concentration, $\tau$ is the residence time, $ER$ is the scaled activation energy, $\alpha$ is the dimensionless heat transfer area and $Q$ is the coolant flowrate. For this system two different sets of manipulated variables will be analyzed. In the first case, the manipulated variable is the coolant flowrate, while in the second case, the manipulated variables are the coolant flowrate $Q$ and the residence time $\tau$.

**One manipulated variable - Cooling flowrate**

In this case [97] the system has two states, one manipulated variable and the number of variables and constraints are 2121 and 1830 respectively. Applying community detection on the unipartite constraint graph six communities are identified; one corresponds to the scheduling problem and the other five are the dynamic optimization subproblems in each slot. The shared variables are the initial and final set point values of the states ($T_k^{in}, T_k^{end}, c_k^{in}, c_k^{end}$), the initial and final value of the inlet flowrate ($Q[1,1,k], q_k^{end}$) and the transition time $\theta_k^t$. This set of shared variables is similar as the one obtained from the community detection in the case of an isothermal CSTR reactor with one state (Section 3.3.1). The final set of shared variables between the scheduling and dynamic optimization subproblem in slot $k$ are:

$$T_k^{end}, T_k^{in}, c_k^{end}, c_k^{in}, q_k^{in}, q_k^{end}, T_c, \theta_k^t$$

where $T_c$ is used for the separability of the objective function. The centrality of the different communities (subproblems) is presented in Table 5. Again, the scheduling subproblem has higher average centrality than the dynamic optimization subproblems. Thus, a hierarchical structure can be identified where the scheduling problem is in the first level of the hierarchy and the dynamic optimization subproblems in the second level.

Table 3.5: Average closeness and betweenness centrality for the constraint unipartite graph of the integrated problem for a non-isothermal CSTR with two states and one manipulated variable

| Subproblem | Closeness centrality | Betweenness centrality |
|---|---|---|
| Scheduling | 0.21 | 0.022 |
| Dynamic opt. slot 1 | 0.16 | 0.0015 |
| Dynamic opt. slot 2 | 0.16 | 0.0013 |
| Dynamic opt. slot 3 | 0.16 | 0.0013 |
| Dynamic opt. slot 4 | 0.16 | 0.0013 |
| Dynamic opt. slot 5 | 0.16 | 0.0015 |

**Two manipulated variables - Cooling flowrate and inlet flowrate**

In this case [16] the system has two states and two manipulated variables, the inlet flowrate $Q_{in}$ ($\tau = V/Q_{in}$) and the coolant flowrate $Q$. The number of variables and constraints are 2431 and 1845 respectively. From the community detection on the unipartite constraint graph, the problem is decomposed into six communities and the modularity of the graph is 0.8. The shared variables between the scheduling subproblem ($S$) and the dynamic optimization subproblem in slot $k$ ($D_k$) are the initial ($T_k^{in}, c_k^{in}$) and final ($T_k^{end}, c_k^{end}$) set point values of the states, the value of the inlet and coolant flowrate at the start ($Q[1,1,k],\ Q_{in}[1,1,k]$) and end ($Q[20,3,k],\ Q_{in}[20,3,k]$) of each slot and the transition time ($\theta_k$) in each slot. This set of shared variables, is similar to the one obtained for the case of an isothermal CSTR and the final set of shared variables is:

$$T_k^{end},\ T_k^{in},\ c_k^{end},\ c_k^{in},\ q_k^{in},\ q_k^{end},\ Q_{in,k}^{in},\ Q_{in,k}^{end},\ \theta_k^t,\ T_c$$

where $T_c$ is used as a shared variable in order to decompose the objective function. Based on the average centrality values of the communities (Table 6) the scheduling subproblem has the highest centrality and the dynamic optimization subproblems centralities are equal. Hence, this system with two manipulated variables also has a hierarchical structure.

Table 3.6: Average closeness and betweenness centrality for the constraint unipartite graph of the integrated problem for a non-isothermal CSTR with two states and two manipulated variable

| Subproblem | Closeness centrality | Betweenness centrality |
|---|---|---|
| Scheduling | 0.21 | 0.021 |
| Dynamic opt. slot 1 | 0.16 | 0.0014 |
| Dynamic opt. slot 2 | 0.16 | 0.0013 |
| Dynamic opt. slot 3 | 0.16 | 0.0013 |
| Dynamic opt. slot 4 | 0.16 | 0.0013 |
| Dynamic opt. slot 5 | 0.16 | 0.0014 |

### 3.4.2 Cascade of CSTRs

The last example is a cascade of isothermal CSTRs (Figure 3.6) with an irrevesible reaction 3A $\rightarrow$ B. The dynamic equations for five reactors are:

$$\begin{aligned} \frac{dc_i}{dt} &= \frac{Q}{V}(c_{in} - c_i(t)) - kc_i(t)^3, \quad i = 1 \\ \frac{dc_i}{dt} &= \frac{Q}{V}(c_{i-1}(t) - c_i(t)) - kc_i(t)^3, \quad i = 2, 3, 4, 5 \end{aligned} \tag{3.50}$$

where $c_i(t)$ is the concentration in reactor $i$. The volumes of the reactors, the rate constant and the inlet concentration into the first reactor are constant. This system has five states and one manipulated variable, the inlet flowrate $Q$ in the first reactor, and the output is the concentration in the last reactor. The transition time is equal to the time required that the concentration in the last reactor changes from an initial value to the new set point. Applying community detection on the unipartite constraint

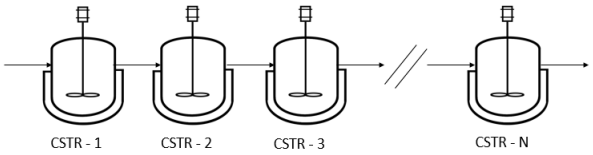


Figure 3.6: Cascade of N CSTRs

graph we again identify six communities. One community is the scheduling subproblem and the other five are the dynamic optimization subproblems in each slot. The shared

variables between the scheduling and dynamic optimization subproblem in slot $k$ are:

$$c_{1k}^{in},\ c_{2k}^{in},\ c_{3k}^{in},\ c_{4k}^{in},\ c_{5k}^{in},\ c_{1k}^{end},\ c_{2k}^{end},\ c_{3k}^{end},\ c_{4k}^{end},\ c_{5k}^{end},\ q_k^{in},\ q_k^{end}, \theta_k^t$$

where $c_{ik}^{in}, c_{ik}^{end}$ are the set point values of the concentration in reactor $i$ at the start and end of slot $k$ ,$q_k^{in}, q_k^{end}$ are the values of the manipulated variable and $\theta_k^t$ is the transition time in slot $k$. The total operating time is used as a shared variable in order to decompose the objective function, thus the final set of shared variables is ($n$ is the reactor number from 1 to 5):

$$c_{nk}^{in}, c_{nk}^{out}, q_k^{in}, q_k^{end}, \theta_k^t, T_c$$

Finally, from Table 7, the average centrality of the scheduling problem is higher than

Table 3.7: Average closeness and betweenness centrality for the constraint unipartite graph of the integrated problem for a cascade of five isothermal CSTRs

| Subproblem | Closeness centrality | Betweenness centrality |
|---|---|---|
| Scheduling | 0.21 | 0.019 |
| Dynamic opt. slot 1 | 0.16 | 0.0007 |
| Dynamic opt. slot 2 | 0.16 | 0.0006 |
| Dynamic opt. slot 3 | 0.16 | 0.0006 |
| Dynamic opt. slot 4 | 0.16 | 0.0006 |
| Dynamic opt. slot 5 | 0.16 | 0.0007 |

the centrality of the dynamic optimization subproblems. Thus, the system has a two-level hierarchical structure like the previous systems considered. Overall, examining the shared variables in the above cases (and in Section 3.3.1), the following remarks can be made.

**Remark 1.** For the integrated cyclic scheduling and dynamic optimization problem of a broad class of continuous processes with time slot formulation, a natural two-level hierarchy can be identified, where one layer is the scheduling problem and the second layer are the dynamic optimization problems in each slot. The two layers are connected by the shared variables that naturally arise from the community detection on the unipartite constraint graph of the integrated optimization problem. For general systems the shared variables between the scheduling ($S$) and the dynamic optimization

subproblem $(D_k)$ in slot $k$ are:

$$(S, D_k) \ : \ x_{nk}^{in}, x_{nk}^{end}, u_{mk}^{in}, u_{mk}^{end}, \theta_k^t, T_c \ \ \forall \ n \ , m, \ k$$

with $x_{nk}^{in}, x_{nk}^{end}$ the initial and final set-points of the $n^{th}$ state in slot $k$ , $u_{mk}^{in}, u_{mk}^{end}$ the initial and final set-point value of the $m^{th}$ manipulated variable in slot $k$, $\theta_k^t$ the transition time in slot $k$ and $T_c$ the total operating time.

**Remark 2.** In all the cases examined in this work, the integrated mixed integer dynamic optimization problem was discretized using orthogonal collocation on finite elements. It can be shown that the same set of shared variables and hierarchical structure are obtained also in the case that Explicit Euler, Implicit Euler or Runge-Kutta methods are used.

## 3.5 Conclusions

The integration of scheduling and dynamic optimization is a challenging task due to the combinatorial nature of the problem, the nonlinear behavior of chemical processes and the time scale separation of the scheduling and dynamic optimization problem. The solution of the monolithic problem is challenging, even impossible for large-scale systems, thus decomposition-based solutions have been proposed in order to reduce the computational load. In this paper, we obtain the decomposition of the integrated optimization problem using community detection on a unipartite constraint graph of the discretized problem. In the proposed community-based decomposition, the shared variables are continuous and the discretization method that is used to convert the integrated MIDO problem to an MINLP problem does not affect the results. A hierarchical structure in the communities is also identified using two centrality measures. Based on this structure, Generalized Benders Decomposition is used as a decomposition-based solution algorithm and is shown to significantly reduce the computational time over a monolithic solution approach.

# Chapter 4
# Stochastic Blockmodeling for Learning the Structure of Optimization Problems

## 4.1 Introduction

Optimization algorithms and solvers are key to solving a wide range of problems in process systems engineering (PSE) [99, 38, 116]. Solving these problems in an efficient and scalable way can be challenging due to the nonlinearity of most physical and chemical systems, multi-scale behavior [39], heat and mass integration among process units [15], demand and supply relations within the enterprise [114], integration of multiple layers of decision-making [17, 81], etc.

When the direct solution of an optimization problem is not possible, techniques such as surrogate modeling and derivative-free optimization, driven by *data*, can be used to build a model with lower computational complexity than a detailed first-principles model and handle cases where a detailed equation-based model is unavailable [73, 46, 34, 155]. Alternatively, one could attempt to exploit the inherent *structure* of the problem and follow a decomposition-based solution [76], i.e., construct multiple subproblems which can be solved efficiently and upon coordination, their iterative solution can lead to the solution of the original problem.

There is an abundance of such decomposition-based solution approaches in the optimization literature [69]. These algorithms are well established in their theoretical aspects and have been applied to a variety of problems [275, 177, 64, 63]. They can be broadly classified as distributed and hierarchical ones [76].

- In distributed decomposition-based solution algorithms, the subproblems are coordinated by a central agent. For example, in Lagrangean decomposition [119] the subproblems interact through a number of constraints called complicating constraints, and the coordination is achieved through the update of the dual variables based on the value of the complicating constraints. For such an approach the subproblems should be weakly coupled and hence the problem should have a block diagonal structure.

- In hierarchical decomposition-based solution algorithms, the coordination is based on a hierarchical relation between the subproblems. For example, in Benders decomposition [27, 108], a small fraction of variables called complicating variables are assigned into the master problem and the rest are assigned into the subproblem which depends on the values of the complicating variables. This partition of the variables implies an L-shaped interaction pattern, since the complicating variables are present in both the master and subproblem. The coordination is achieved through Benders and feasibility cuts which inform the master problem about the effect of the shared variables on the subproblem.

It is commonly known that the decomposition of the problem itself can have an important effect on the performance of decomposition-based solution algorithms [253, 276]. Yet decompositions have been typically chosen based on intuition or relying on a-priori knowledge of certain structure of the problem.

In recent research of our group, we have proposed the application of network science tools for the decomposition of optimization and control problems [77, 76]. The idea is to represent the structural coupling among the variables and constraints of an optimization problem or between the inputs, states and outputs of a control problem, as a network (graph). Then, through community detection algorithms [102], the network can be partitioned according to its community structure, namely blocks of nodes that are weakly coupled between them but with dense interconnections inside. This approach has been applied to nonlinear optimization problems arising in distributed model predictive control [271], mixed integer nonlinear problems arising in integration of scheduling and control [198] and an optimal biorefinery/microgrid design problem [6], documenting the ability of community detection to provide high-quality decompositions. The above approach is restricted to problems with community structure.

In general, the structure of a network is determined by the connectivity of the nodes which can be analyzed at different scales; the micro, macro and meso scale. The first provides information about a specific node and the second about statistical properties of the entire graph. In this work we will focus on the meso scale structure which considers the interaction pattern among blocks of nodes. A block refers to a group of nodes that have a similar interaction pattern. Based on these interaction patterns, different block structures exist. For example, core periphery is a block structure where the nodes are partitioned into two blocks; a dense core and a sparse periphery which is mainly connected with the core. This heterogeneity in the number of connections between the

two blocks reveals a hierarchy which can be exploited and used as the basis for the application of hierarchical decomposition based solution algorithms. Hybrid structures such as multicore-periphery and core-community are also common. The detection of the underlying block structure in an optimization problem can guide the adoption of the appropriate decomposition based solution algorithm.

In this paper we aim at a method for automatically learning general latent block structures of optimization problems and utilizing them as the basis for distributed and hierarchical decomposition-based solution approaches. Specifically, we propose stochastic Blockmodeling (SBM) [110] as a systematic framework to learn the underlying structure of optimization problems without any a-priori assumptions on the structure of the problem. SBM is a stochastic generative model of networks, which, can generate graphs with any specific block structure based on the input parameters that specify the expected interconnection densities among the blocks. This feature allows SBM, which originated in the data analysis of social networks [133, 7] to capture the generative mechanisms and model the structure of complex networks, including community structure as assortative (preferring internal linkage) blocks. For networks whose block structures are unknown a-priori, the density parameters together with the latent block affiliations of the nodes can be estimated through statistical inference procedures based on the data, i.e., the network itself [131, 149, 78]. Hence, for an optimization problem represented as a network, we assume that it is generated by a SBM and estimate the parameters involved in the SBM to reveal its underlying block structure. We then use the decompositions obtained from SBM as the basis for the application of decomposition-based solution algorithms, and show with several benchmark problems that such decompositions can reduce the necessary computational time to obtain an optimal solution or an estimate of the upper and lower bound.

The rest of the paper is organized as follows. In Section 2, we introduce the fundamentals of SBM, the inference problem and the different inference approaches. In Section 3, we apply SBM and inference to the constraint and variable graphs of benchmark optimization problems, exposing their inherent block structures and demonstrating the numerical advantage of exploiting them in subsequent decomposition-based algorithms. In Section 4, we illustrate how such a systematic structure learning and decomposition-based solution framework is implemented in our DecODe software package. Conclusions are given in Section 5.

## 4.2 Stochastic Blockmodeling

### 4.2.1 Inference of latent block structure

We will consider an undirected graph $G = (V, E)$ ($V$ and $E$ are the set of nodes and edges, respectively). Let the number of nodes and number of edges be $n = |V|$ and $m = |E|$, respectively, and the adjacency matrix be $A \in \mathbb{R}^{n\times n}$, where $A_{ij} = A_{ji}$ is equal to the number of edges between node $i$ and node $j$. One self loop on node $i$ corresponds to $A_{ii} = 2$. We will define the partition vector $b \in \mathbb{R}^n$, where $b_i \in \{1, 2, ..., B\}$ denotes the group membership of node $i$ and $B$ is the number of blocks (groups). We will assume that the number of edges between a node belonging to group $r$ and a node in group $s$ is a Poisson distributed random variable with expected value $\omega_{rs}$. Since the graph is undirected, the matrix $\omega \in \mathbb{R}^{B\times B}$ is symmetric. Then the probability to observe a network with adjacency matrix $A$ is:

$$P(A|b,\omega) = \prod_{1\le i<j\le n} \frac{\omega_{b_i b_j}^{A_{ij}}}{A_{ij}!} e^{-\omega_{b_i b_j}} \times \prod_{i=1}^{n} \frac{(\frac{1}{2}\omega_{b_i b_i})^{A_{ii}}}{(A_{ii}/2)!} e^{-\frac{1}{2}\omega_{b_i b_i}}, \tag{4.1}$$

where a term in the first product is the probability of an edge between two distinct nodes and a term in the second product is the probability of a self edge.

Since the adjacency matrix $A$ and the block interconnection matrix $\omega$ are both symmetric, the logarithm of the likelihood can be written as (by dropping the constant terms):

$$\mathcal{L}(A|b,\omega) = \log P(A|\omega, b) = \sum_{r,s=1}^{B} \Big( m_{rs} \log \omega_{rs} - n_r n_s \omega_{rs} \Big). \tag{4.2}$$

Here $m_{rs}$ is the number of edges between blocks $r$ and $s$, and $n_r$ is the number of nodes in block $r$, defined as

$$m_{rs} = \sum_{i,j=1}^{n} A_{ij}\delta_{b_i,r}\delta_{b_j,s}, \quad n_r = \sum_{i=1}^{n} \delta_{b_i,r}, \tag{4.3}$$

where $\delta$ is the Kronecker delta. This is the basic SBM model where the inputs are the number of the nodes $n$, number of blocks $B$, partition of the nodes $b$, and the matrix $\omega$ characterizing the interconnection densities among the $B$ blocks, and the output is a graph with a structure depending on the elements of the $\omega$ matrix.

The main limitation of the basic SBM model is that it does not account for the degree distribution of the nodes. This can be resolved by defining the degree-corrected SBM (DC-SBM) [149] which has an additional group of parameters – $\theta_i$, the expected degree of node $i$, $i = 1, \ldots, n$ (the degree of node $i$ is equal to the number of edges incident to node $i$). If we assume the same probability distribution as before, then the probability to observe the network is:

$$P(A|\theta, \omega, b) = \prod_{1 \le i < j \le n} \frac{(\theta_i \theta_j \omega_{b_i b_j})^{A_{ij}}}{A_{ij}!} e^{-\theta_i \theta_j \omega_{b_i b_j}} \times \prod_{i=1}^{n} \frac{(\frac{1}{2}\theta_i^2 \omega_{b_i b_i})^{A_{ii}}}{(A_{ii}/2)!} e^{-\frac{1}{2}\theta_i^2 \omega_{b_i b_i}}. \tag{4.4}$$

Following the same steps as before, the logarithm of the probability can be written as

$$\mathcal{L}(A|\theta, \omega, b) = \log P(A|\theta, \omega, b) = 2 \sum_{i=1}^{n} k_i \log \frac{k_i}{k_{g_i}} + \sum_{r,s=1}^{B} m_{rs} \log m_{rs} - 2m, \tag{4.5}$$

where $k_i$ is the degree of node $i$, $k_r$ is equal to the sum of the degrees of all the nodes in block $r$:

$$k_i = \sum_{j=1}^{n} A_{ij}, \quad k_r = \sum_{s=1}^{B} m_{rs} = \sum_{i=1}^{n} k_i \delta_{b_i, r} \tag{4.6}$$

and $m$ is the total number of edges.

Given the observed network $A$, in order to learn the latent block structures, one then needs to carry out a parametric statistical inference procedure to estimate the block affiliations of the nodes $b$ and the interconnection densities $\omega$. In the sequel of this section, we will present two major types of inference methods, namely maximum likelihood estimation and Bayesian estimation, for the case of the basic SBM model in order to keep the notation simple. For the degree-corrected case, the results can be found in the cited references.

**Remark 4.1.** Different forms of probability distributions other than Poisson distribution can be assumed for the number of edges between the nodes, e.g., Bernoulli or normal distributions. However, it is known that for sparse networks, the choice of the probability distribution does not significantly affect the inference results [241] and that the Poisson distribution makes the calculations easier. Also based on the definition of matrix $A$ we can consider multi-graphs (graphs where there may exist multiple edges between node pairs). This assumption is valid in the limit of sparse networks since the probability of multiple edges is low.

**Remark 4.2.** Based on the structure of the input parameters different types of graphs can be generated. For example, a non-symmetric matrix $\omega$ leads to a directed graph. Also, SBM can be extended to generate bipartite graphs where the entries of the $\omega$ matrix that connect nodes that belong to the same type are set equal to zero [167].

### 4.2.2 Maximum Likelihood Estimation

Given a graph $G = (V, E)$, which is assumed to be generated by a SBM, the task of Maximum Likelihood Estimation (MLE) is to find the values of the parameters $(b, \omega)$ that maximize the logarithm of the probability $P(A|b, \omega)$:

$$\underset{b,\omega}{\text{maximize}} \log P(A|b, \omega). \tag{4.7}$$

The estimates of $\omega$ can be obtained by simple differentiation:

$$\partial \log P(A|\omega, b)/\partial \omega_{rs} = 0 \quad \Rightarrow \quad \hat{\omega}_{rs} = m_{rs}/n_r n_s. \tag{4.8}$$

Then we obtain the normalized log-likelihood:

$$\mathcal{L}(A|b) = \sum_{r,s} m_{rs} \log(m_{rs}/n_r n_s) - 2m. \tag{4.9}$$

The maximization with respect to the assignment vector $b$ is a combinatorial problem. One approach to solve this problem is to recursively move nodes across blocks with maximum increase or minimum decrease in $\mathcal{L}$ [149]. This is a local optimization approach. Therefore, the calculations should be performed under different initializations of the partition $b$.

An alternative approach to solve the inference problem is to define an additional parameter $\gamma_r$ for each block $r$, which is considered as the probability that any node belongs to group $r$ [302]. If we assume a Bernoulli distribution for simplicity, the logarithm of the probability to observe a graph is:

$$\begin{aligned} P(A|\omega, \gamma) &= \sum_b P(A|\omega, \gamma, b) P(b|\gamma) \\ &= \sum_b \prod_{i<j} \omega_{b_i b_j}^{A_{ij}} (1 - \omega_{b_i b_j})^{1 - A_{ij}} \prod_i \gamma_{b_i}, \end{aligned} \tag{4.10}$$

where the sum is taken over all possible partitions $b$. The MLE problem is to find

the values of $(\omega, \gamma, b)$ that maximize the logarithm of the likelihood. This is done using Jensen's inequality for the logarithm function, which is concave. Let $q(\cdot)$ be any probability distribution over all possible block assignments $\{1, 2, \ldots, B\}^n$. Then we have

$$\begin{aligned} \log P(A|\omega,\gamma) &\geq \sum_b q(b) \\ &\times \log\left[\frac{1}{q(b)}\prod_{i<j}\omega_{b_i b_j}^{A_{ij}}(1-\omega_{b_i b_j})^{1-A_{ij}}\prod_i \gamma_{b_i}\right]. \end{aligned} \tag{4.11}$$

This leads to

$$\begin{aligned} \log P(A|\omega,\gamma) \geq \frac{1}{2}\sum_{i,j}\sum_{r,s}\Bigg[&\sum_{i,r} q_r^i \log\gamma_r - \sum_b q(b)\log q(b) \\ &+ A_{ij}q_{rs}^{ij}\log\omega_{r,s} + (1-A_{ij})q_{rs}^{ij}\log(1-\omega_{rs})\Bigg], \end{aligned} \tag{4.12}$$

where $q_r^i$ is the marginal distribution within the chosen distribution $q(b)$ that node $i$ belongs to group $r$ and $q_{rs}^{ij}$ is the joint marginal probability that node $i$ belongs to group $r$ and node $j$ to group $s$:

$$q_r^i = \sum_b q(b)\delta_{b_i,r}, \quad q_{rs}^{ij} = \sum_b q(b)\delta_{b_i,r}\delta_{b_j,s}. \tag{4.13}$$

The above inequality turns into an equality if $q(b)$ is chosen according to the following formula:

$$q(b) = \frac{\prod_{i<j}\omega_{b_i b_j}^{A_{ij}}(1-\omega_{b_i b_j})^{1-A_{ij}}\prod_i \gamma_{b_i}}{\sum_b \prod_{i<j}\omega_{b_i b_j}^{A_{ij}}(1-\omega_{b_i b_j})^{1-A_{ij}}\prod_i \gamma_{b_i}}. \tag{4.14}$$

If we substitute this expression in Eq. 4.12, the optimal values of the parameters $\omega_{rs}$ and $\gamma_r$ can be obtained by maximizing the logarithm under the constraint that $\sum_r \gamma_r = 1$:

$$\hat{\omega}_{rs} = \frac{\sum_{ij} A_{ij}q_{rs}^{ij}}{\sum_{ij} q_{rs}^{ij}}, \; \hat{\gamma}_r = \frac{1}{n}\sum_i q_r^i. \tag{4.15}$$

The *expectation-maximization* (EM) algorithm uses iterations of Eqs. 4.14 and 4.15 for the maximization of log-likelihood. That is, given an initial guess for $(\omega, \gamma)$, the probability distribution $q(b)$ is computed, and then using the new value of $q(b)$, the values of $\omega, \gamma$ are updated. The EM approach provides a local solution to the maximization

problem; however, the update of $q(b)$ requires the summation over all possible node partitions. To avoid this, the *belief propagation* (BP) algorithm is used to estimate the probability distribution $q(b)$ and marginal probabilities $q_r^i, q_{rs}^{ij}$. This is a message-passing approach to compute the marginal probabilities on graph models. We refer the reader to [302] for a systematic explanation of the BP algorithm.

**Remark 4.3.** In the MLE approach, the number of blocks $B$ is fixed as a hyperparameter for the SBM. From a network science point of view, one often does not know a priori the actual number of latent blocks, and hence $B$ should also be inferred. Discussions can be found in [217] on extending the SBM framework to estimate $B$. For our purpose of decomposing optimization problems, however, it may be useful to keep $B$ as a tunable hyperparameter so that the most appropriate decomposition can be selected.

**Remark 4.4.** Community structure [102] refers to an assortative block structure, i.e., blocks such that the interconnections inside blocks are much stronger than those between blocks. In the parlance of SBM, community structure corresponds to a matrix $\omega$ that is diagonally dominant. Traditionally, community detection is based on maximizing a quantitative index called Newman-Girvan modularity [220], which was proved equivalent to the MLE for SBM [218]. Therefore, the framework of using SBM for learning and exploiting block structure of optimization problems incorporates our previous efforts that focused on community detection for decomposition [271, 6].

### 4.2.3 Bayesian inference

Different from MLE which is a direct point estimation scheme, i.e., which determines a unique value of parameters, Bayesian inference introduces a prior belief of the probability distribution of the parameters, and through the observation, infers an a posteriori distribution. The main advantage of using Bayesian inference for SBM, proposed by Peixoto [241, 237], is the role of the prior distribution of the block partitions $P(b)$ as a regulating factor in the inference.

Specifically, using Bayes' rule, we obtain

$$P(b|A) = \frac{P(A|b)P(b)}{P(A)} \tag{4.16}$$

where

$$P(A|b) = P(A|\omega, b)P(\omega|b), \quad P(A) = \sum_b P(A|b)P(b). \tag{4.17}$$

The task of Bayesian inference is to find the probability distribution $P(b|A)$. Once $P(b|A)$ is known, the partition $b$ that maximizes $P(b|A)$ can be found as a point estimation, which is called the maximum a posteriori (MAP) estimate. Rewriting the numerator of Eq. S.2 as

$$P(A|b)P(b) = P(A|\omega, b)P(\omega|b)P(b) = P(A|\omega, b)P(\omega, b), \tag{4.18}$$

we aim at solving the following optimization problem:

$$\underset{b,\omega}{\text{maximize}} \log P(A|\omega, b) + \log P(\omega, b). \tag{4.19}$$

Comparing to the MLE problem (Eq. 4.7), the MAP problem has a regulating term $\log P(\omega, b)$, which helps to prevent overfitting. For example, for a given network $A$, if the number of blocks is increased, with a larger number of parameters, the fitting based on the log-likelihood $\log P(A|\omega, b)$ alone will always be improved, even if not revealing the actual latent blocks. Nevertheless, $\log P(\omega, b)$ can be made to decrease if the prior distribution has a low probability on large numbers of blocks. Thus, the Bayesian approach provides the user with an additional tuning handle to fit the SBM.

In this paper, the prior distribution is specified according to the microcanonical ensemble approach [237]. Let $\nu = (\nu_1, \ldots, \nu_B)$ be the vector of numbers of nodes allocated to the $B$ blocks, with $\sum_r \nu_r = n$. Supposing $n$ is given, the prior probability that the block affiliations $b$ are the current ones is the reciprocal of the total numbers of permutations that keep $\nu$ unchanged, i.e.,

$$P(b|\nu) = \frac{\prod_r \nu_r!}{n!}. \tag{4.20}$$

Then, the number of ways to separate $n$ nodes into $B$ blocks is $\binom{n-1}{B-1}$. Then the probability that the sizes of blocks are consistent with $\nu$ is

$$P(\nu|B) = \binom{n-1}{B-1}^{-1}. \tag{4.21}$$

To avoid overfitting, one can further assume that the probability of having $B$ blocks, $P(B)$, equals $1/N$ where $N$ is a maximum allowed number of blocks. Hence the prior

of block assignments $P(b)$ is

$$P(b) = P(b|\nu)P(\nu|B)P(B) = \frac{\prod_r \nu_r!}{n!}\binom{n-1}{B-1}^{-1}\frac{1}{N}. \tag{4.22}$$

The prior of $\omega$ can be substituted by a prior of the number of edges $e_{rs}$ between each pair of blocks $r < s$. This prior is assumed to be uniform among all ways of allocating $m$ edges among $B(B+1)/2$ block pairs:

$$P(e) = \binom{B(B+1)/2 + m - 1}{m}^{-1}. \tag{4.23}$$

For the computation, we follow a Markov Chain Monte Carlo (MCMC) approach where $P(b|A)$ will be sampled starting from a random node assignment $b_0$. For each node a move proposal is made $b_i \to b_i'$ and is accepted with probability $\alpha$ following the Metropolis-Hastings criterion. This guarantees that the Markov chain will converge to the equilibrium distribution $P(b|A)$. The selection of the move proposal for each node is done by exploiting the group membership of the neighbor nodes and selecting one of them. The complexity of the algorithm is quasilinear $O(n \ln^2 n)$, where $n$ is the number of nodes in the network, and is independent of the number of the blocks [234]. Hence, the inference problem can be solved efficiently for large graphs. We refer the reader to Peixoto [241, 237] for a detailed explanation of the algorithm. This approach allows the estimation of $P(b|A)$, the number of blocks $B$, and partition $b$. We must notice that since this algorithm is stochastic, the results might differ among runs.

**Remark 4.5.** Let

$$P(A|\omega, b)P(\omega, b) = 2^{-\sigma}, \tag{4.24}$$

where $\sigma$ is called the *description length* of the data:

$$\sigma = -\log_2 P(A|\omega, b) - \log_2 P(\omega, b). \tag{4.25}$$

Description length is a information-theoretical concept that characterizes the amount of information that is necessary to encode the data $A$ and the parameters of the model $(b, \omega)$. Thus, the goal of finding the network partition that maximizes the posterior distribution (Eq. S.2) is equivalent to finding the partition with the minimum description length.

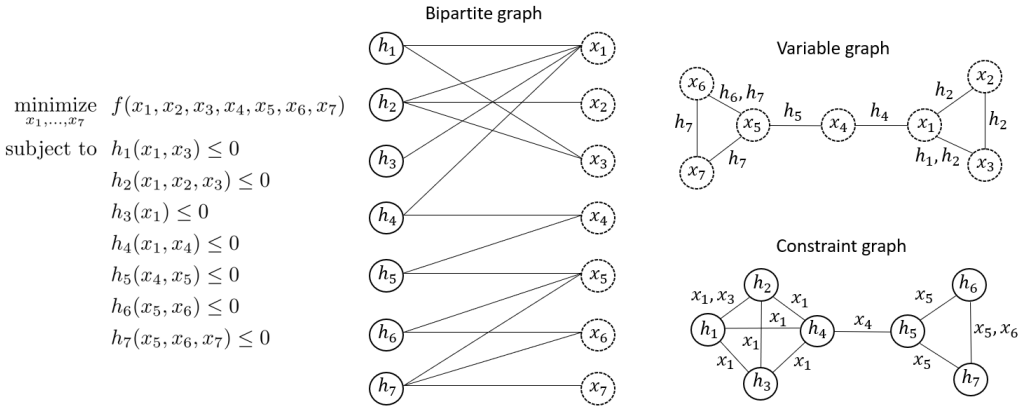


Figure 4.1: Example of the network representation of optimization problems

## 4.3 Application to optimization problems

In this section we will address the learning of the structure and interaction pattern between the variables and constraints of optimization problems using the degree-corrected SBM. First we describe a conceptual, formal relation between different block structures and decomposition-based solution algorithms. Then, through examples we show the ability of the proposed approach to learn the block structure of optimization problems which can subsequently be used in decomposition-based solution algorithms.

### 4.3.1 Relation between block structure in the optimization problem and decomposition-based solution algorithms

#### Network perspective of optimization problems

We will consider a general optimization problem:

$$\begin{aligned} &\underset{x}{\text{minimize}} \quad f(x) \\ &\text{subject to} \quad g(x) \leq 0, \end{aligned} \tag{4.26}$$

where the variables $x$ can be both continuous and binary. Given this problem we can create the bipartite variable-constraint and the unipartite variable and constraint graphs [6] which capture the structural interaction among the variables and constraints of the problem. The bipartite graph has two sets of nodes representing variables and

constraints, with an edge signifying that a variable appears in a constraint. In the constraint graph, the nodes are the constraints of the problem and the edges capture the variables that couple the constraints. Similarly, in the variable graph the nodes are the variables of the problem and the edges correspond to the constraints that couple the variables. The graphs for an example problem are shown in Figure 4.1.

Although all these graphs correspond to the same problem, each one captures different structural interactions, i.e. the constraint graph captures the constraint coupling whereas the variable graph captures the variable coupling. Assuming that the graph of an optimization problem is generated by a SBM, the estimated structure of the $\omega$ matrix will allow us to learn the interaction pattern among the variables or constraints and decide on an appropriate solution algorithm. This selection can be based on the structure of either the constraint or the variable graph. Therefore we recommend that one should analyze the structure of both graphs before selecting the most appropriate decomposition based algorithm. In Figure 4.2 different possible block structures in a graph and their conceptual relation to decomposition-based solution algorithms are presented.

The first example (a) corresponds to a community structure ($\omega_{ii} > \omega_{ij}$) where the two blocks are weakly coupled. This structure can be used for the application of distributed optimization algorithms, where a central agent coordinates the subproblems (blocks). The second example (b) corresponds to a core-periphery structure where the graph is composed of a densely connected core and a sparse periphery which is mainly connected with the core ($\omega_{11} > \omega_{12} > \omega_{22}$). This is a hierarchical (L-shaped) structure and can be used as the basis for the application of hierarchical solution algorithms. In such an approach, the part of the problem that corresponds to the core or the periphery is assigned to the high-level agent, and the other part is assigned to the lower-level agent. The relation between the high/lower-level agent and core/periphery is not strict (see sections 3.1.2 and 3.1.3). The last two cases (c,d) correspond to hybrid core-community structures. In this case multiple lower-level agents exist and might cooperate to solve the problem in a nested decomposition approach.

### Constraint graph

The constraint graph captures the structural coupling of the constraints, hence a partition of the nodes reveals the variables that couple constraints that belong to different blocks (overlapping variables). Let's consider a general optimization problem (Eq. 4.26)

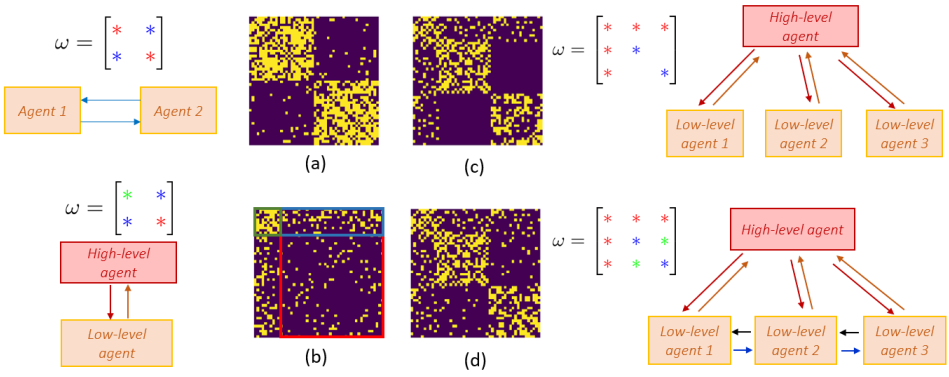


Figure 4.2: Conceptual relation between the $\omega$ matrix and decomposition-based solution algorithms.

and let's assume, for simplicity, that a partition of the constraint graph into two blocks is obtained. Hence, the constraints of the problem are partitioned into two sets $g_1, g_2$ and the variables are partitioned into three sets: variables that belong only in the first block $y$, variables that belong only in the second block $z$, and variables $u$ that couple constraints that belong in different blocks. Given this partition, the original problem can be written as:

$$\begin{aligned} \underset{y,z,u}{\text{minimize}} \quad & f(y,z,u) \\ \text{subject to} \quad & g_1(y,u) \leq 0 \\ & g_2(z,u) \leq 0. \end{aligned} \tag{4.27}$$

Based on the type of variables (continuous or binary) and the hierarchy among the blocks in this problem, we can apply either Lagrangean or Benders decomposition.

If the problem has binary variables which exist in all sets $y, z, u$ we can apply Lagrangean decomposition. Creating a copy of the coupling variables, $\bar{u}$, the problem can be written as (assuming that the objective function is separable):

$$\begin{aligned} \underset{y,z,u,\bar{u}}{\text{minimize}} \quad & f_1(y) + f_2(z) + f_3(u) \\ \text{subject to} \quad & g_1(y,u) \leq 0 \\ & g_2(z,\bar{u}) \leq 0 \\ & u = \bar{u} \end{aligned}$$

Dualizing the last constraint we obtain:

$$\begin{aligned} \underset{y,z,u,\bar{u}}{\text{minimize}} \quad & f_1(y) + f_2(z) + f_3(u) + \lambda(u - \bar{u}) \\ \text{subject to} \quad & g_1(y, u) \leq 0 \\ & g_2(z, \bar{u}) \leq 0, \end{aligned} \tag{4.28}$$

where $\lambda$ are the Lagrange multipliers. This problem can be solved independently for $y, u$ and $z, \bar{u}$. In this case, the block structure of the constraint graph lends itself naturally to a Lagrangean decomposition approach.

If the binary variables belong in some but not all blocks and by fixing these variables the problem can be solved efficiently, we can apply hierarchical solution algorithms. For the case of Benders decomposition, we can formulate the master problem (M), subproblem (S), and infeasible subproblem (IS) based on the block partition. For example, assuming that the binary variables belong in $y, u$ we obtain:

$$\begin{aligned} S := \underset{z}{\text{minimize}} \quad & f(\bar{y}, z, \bar{u}) \\ \text{subject to} \quad & g_2(z, \bar{u}) \leq 0 \end{aligned} \qquad\qquad \begin{aligned} IS := \underset{z,\alpha}{\text{minimize}} \quad & \alpha \\ \text{subject to} \quad & g_2(z, \bar{u}) \leq \alpha. \end{aligned}$$

The master problem is:

$$\begin{aligned} \underset{y,u,\eta}{\text{minimize}} \quad & \eta \\ \text{subject to} \quad & g_1(y, u) \leq 0 \\ & \eta \geq f(y, \bar{z}^p, u) + \bar{\lambda}^p g_2(\bar{z}^p, u) \;\forall\; p \in \mathcal{P} \\ & 0 \geq \bar{\lambda}^k g_2(\bar{z}^k, u) \;\forall\; k \in \mathcal{K}, \end{aligned}$$

where $\lambda, \bar{\lambda}$ are the Lagrange multipliers and $p, k$ are the iteration numbers that the subproblem is feasible and infeasible, respectively. In this case the application of Benders decomposition is based on the idea that some blocks contain only continuous variables and are connected with the blocks that contain the binary variables. Finally, we must note that the complicating variables $u$ can be both binary and continuous and different formulations of the subproblems (S,IS) can be used [99, 69].

### Variable graph

The variable graph captures the interactions among the variables through the constraints. Given a partition of the variables into two blocks $y, z$, the constraints are decomposed into three sets: constraints that contain only $y$ variables $g_1$, constraints that contain only $z$ variables $g_2$, and constraints that contain both $y, z$ variables, $g_c$. The original problem can then be written as:

$$\begin{aligned} \underset{y,z}{\text{minimize}} \quad & f(y,z) \\ \text{subject to} \quad & g_1(y) \leq 0 \\ & g_2(z) \leq 0 \\ & g_c(y,z) \leq 0. \end{aligned} \tag{4.29}$$

Assuming that the binary variables belong in both sets $y, z$ and the objective function is separable, we can dualize the last constraint ($g_c$) to obtain:

$$\begin{aligned} \underset{y,z}{\text{minimize}} \quad & f_1(y) + f_2(z) + \lambda g_c(y,z) \\ \text{subject to} \quad & g_1(y) \leq 0 \\ & g_2(z) \leq 0. \end{aligned} \tag{4.30}$$

In this case the constraints are decoupled and if the constraints $g_c$ are linear or decomposable we can apply Lagrangean relaxation.

If all the binary variables belong in one block, e.g. $y$ ($y$ can contain both continuous and binary variables), and the variable graph has a hierarchical structure such as core-periphery, hierarchical algorithms can be applied. Again, for the case of Benders decomposition, we can assign variables $y$ and $g_1$ into the master problem and variables $x$ and $g_2, g_c$ into the subproblem. Given this partition the subproblems are:

$$\begin{aligned} S := \underset{z}{\text{minimize}} \quad & f(\bar{y}, z) \\ \text{subject to} \quad & g_2(z) \leq 0 \\ & g_c(z, \bar{y}) \leq 0 \end{aligned} \qquad\qquad \begin{aligned} IS := \underset{z,\alpha}{\text{minimize}} \quad & \alpha \\ \text{subject to} \quad & g_2(z) \leq \alpha \\ & g_c(z, \bar{y}) \leq \alpha \end{aligned}$$

and the master problem is:

$$\begin{aligned} \underset{y,\eta}{\text{minimize}} \quad & \eta \\ \text{subject to} \quad & g_1(y) \leq 0 \\ & \eta \geq f(y, \bar{z}^p) + \bar{\lambda}^p g_c(\bar{z}^p, y) \ \forall \ p \in \mathcal{P} \\ & 0 \geq \bar{\lambda}^l g_c(\bar{z}^k, y) \ \forall \ k \in \mathcal{K}, \end{aligned}$$

**Remark 4.6.** In the above formulations we assumed that the constraint or variable graph is decomposed into two blocks. In general, the graph can be decomposed into any number of blocks. The requirements for the application of Lagrangean decomposition is the separability of the objective function and the coupling constraints ($g_c$) in the case that the block structure of the variable graph is used. For the application of Benders decomposition the requirement is that the binary variables belong in some blocks and by fixing the variables in these blocks the problem can be solved efficiently.

**Remark 4.7.** For the application of hierarchical solution algorithms, we considered only Benders decomposition. In general the hierarchical structure of both the constraint and variable graphs can be used as the basis for the application of bilevel decomposition. In this case the master and subproblem are the same and integer/design cuts are added in each iteration.

### 4.3.2 Lagrangean decomposition based on the block structure of the constraint graph

In this section we apply the degree corrected SBM to benchmark optimization problems. We show that the learnt block structure can be used effectively in decomposition-based solution algorithms with either a solution or an estimate of the upper and lower bound obtained in reduced computational time.

#### Case study 1: General_Model_Case1

We will consider problem `General_Model_Case1`[1] from MORG Library. This is a non-convex Mixed Integer Nonlinear Problem (MINLP) with 113 variables (12 binary) and 121 constraints. The constraint graph is presented in Figure 4.3. Based on this figure we

[1]The problem is downloaded in Pyomo format from `https://www.minlp.com/nlp-and-minlp-test-problems`.

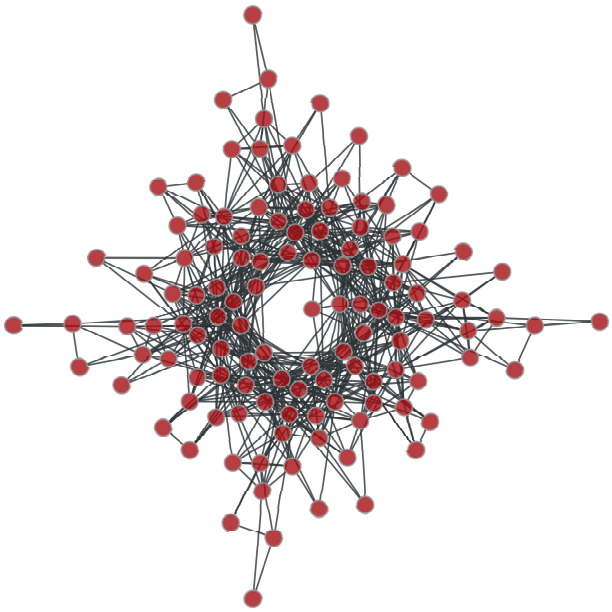

Figure 4.3: Constraint graph of the General_Model_Case1 problem

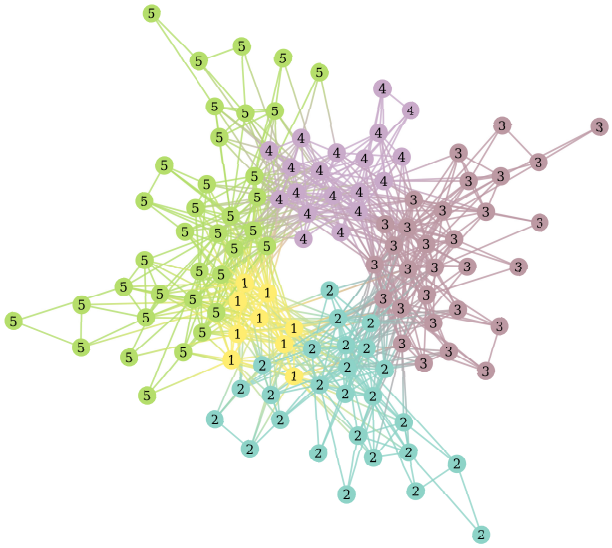

Figure 4.4: SBM results on the constraint graph of problem General_Model_Case1

see that the graph does not have an apparent structure. We applied Bayesian inference in `graph-tool` [235] and the partition that maximizes $P(b|A)$ is presented in Figure 4.4.

Table 4.1: Statistics of the subproblems based on the SBM partition of constraints for problem General Model Case 1.

| | Subproblem | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| Constraints | 26 | 30 | 36 | 19 | 9 |
| Variables (cont./binary) | 26/3 | 29/3 | 38/4 | 24/2 | 18/2 |

From the inference results, the graph is decomposed into five blocks and the $\omega$ matrix is:

$$\omega = \begin{bmatrix} 48 & 88 & 36 & 40 & 99 \\ 88 & 57 & 42 & 0 & 0 \\ 36 & 42 & 96 & 57 & 0 \\ 40 & 0 & 57 & 50 & 55 \\ 99 & 0 & 0 & 55 & 64 \end{bmatrix}.$$

Based on matrix $\omega$ the problem has a complex, hybrid core-community structure. Block 1 acts as a coordinator (core) since it connects all the other blocks which form an ordered community structure. If we consider only the interactions among blocks 2-5, we see that blocks 2,3,5 are assortative communities ($\omega_{ii} > \omega_{ij}$) whereas block 4 is a dissasortative community ($\omega_{ij} > \omega_{ii}$). Given this partition of the constraints, the binary variables are present in all the blocks and the problem is decomposed into five MINLPs with the statistics of the subproblems presented in Table 4.1.

We apply Lagrangean decomposition, the subproblems are solved with BARON [273] and the dual variables are updated using the Fischer formula [93]. The evolution of upper and lower bounds are presented in Figure 4.5. From the results after 400 CPU seconds the gap between the upper and lower bound is 131.8% whereas solving the entire problem monolithically the gap is 526%.

### Case study 2: 4stufen

In the second case study we consider problem `4stufen` from MINLP Library[2]. Before creating the constraint graph one equality constraint is removed since it contains only one variable and the variable appears only in this constraint and the objective function.

[2]The problem is downloaded in Pyomo format from `https://www.minlp.com/nlp-and-minlp-test-problems` [53].

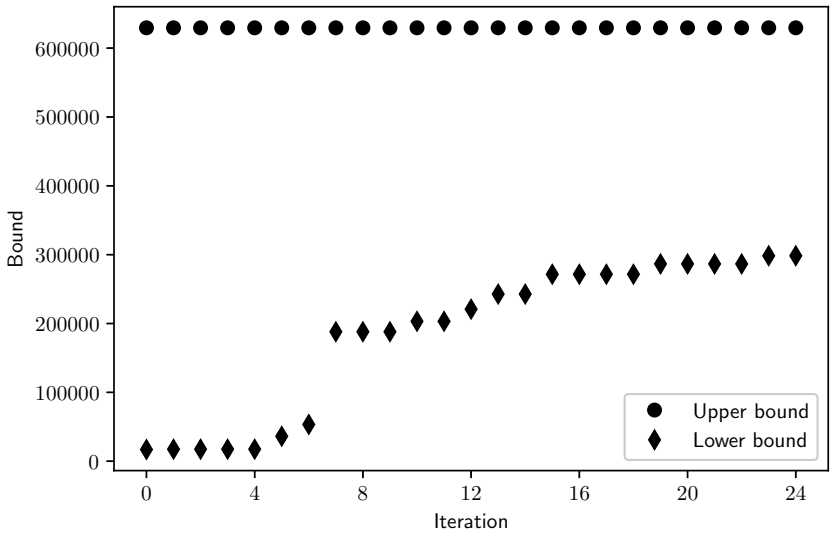


Figure 4.5: Evolution of the upper and lower bound for problem General_Model_Case1 using Lagrangean decomposition based on the results of Bayesian SBM.

The variable is fixed to the value of the constraint and is considered as a parameter in the problem. This is a nonconvex MINLP with 148 variables (48 binary) and 97 constraints. Application of Bayesian SBM on the constraint graph identifies two blocks and the partition with maximum $P(b|A)$ is presented in Figure 4.6.

Based on the entries of the $\omega$ matrix the blocks are weakly connected ($\omega_{ii} > \omega_{ij}$) and have different densities.

$$\omega_{stufen} = \begin{bmatrix} 71 & 16 \\ 16 & 154 \end{bmatrix}$$

The problem that corresponds to block 1 (blue nodes) is a MINLP with 43 constraints and 95 variables and the problem that corresponds to block 2 (yellow nodes) has 54 constraints and 53 variables and is a nonlinear problem (NLP). Based on this decomposition we apply Lagrangean decomposition, the subproblems are solved with BARON [273], the update of the dual variables is based on the Fischer formula [93], and the initial stepsize is one and is reduced by a factor of two if the lower bound is not updated after three iterations. The evolution of the upper and lower bound is presented in Figure 4.7. After 3000 CPU seconds the gap is 18.7% whereas solving the problem monolithically with BARON the gap is 928%.

Figure 4.6: SBM results on the constraint graph of problem 4stufen

### 4.3.3 Generalized Benders decomposition based on the hybrid core community structure of the variable graph

In this section we analyze the variable graph of the optimization problem. We consider problem `feedtray` from MINLP Library[3]. This is a nonconvex MINLP with 98 variables (7 binary) and 92 constraints. The partition of the variables with the maximum value of $P(b|A)$ is presented in Figure 4.8. From the results, the variable graph has a complex block structure and is decomposed into nine blocks. Block 1 contains all the binary variables and the continuous variables are assigned into the other blocks. The matrix $\omega$ is:

$$\begin{bmatrix} 21 & 0 & 1 & 2 & 1 & 0 & 2 & 1 & 0 \\ 0 & 48 & 53 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 53 & 30 & 55 & 1 & 0 & 2 & 1 & 0 \\ 2 & 0 & 55 & 45 & 55 & 0 & 4 & 2 & 0 \\ 1 & 0 & 1 & 55 & 33 & 53 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 53 & 22 & 53 & 0 & 0 \\ 2 & 0 & 2 & 4 & 2 & 53 & 45 & 55 & 0 \\ 1 & 0 & 1 & 2 & 1 & 0 & 55 & 30 & 53 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 53 & 53 \end{bmatrix}.$$

[3]The problem is downloaded in Pyomo format from `https://www.minlp.com/nlp-and-minlp-test-problems` [53].

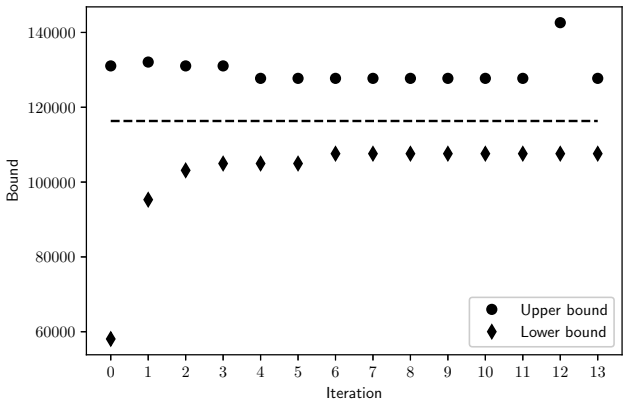


Figure 4.7: Evolution of the upper and lower bound for problem 4stufen. The dashed line corresponds to the solution reported in MINLP Library.

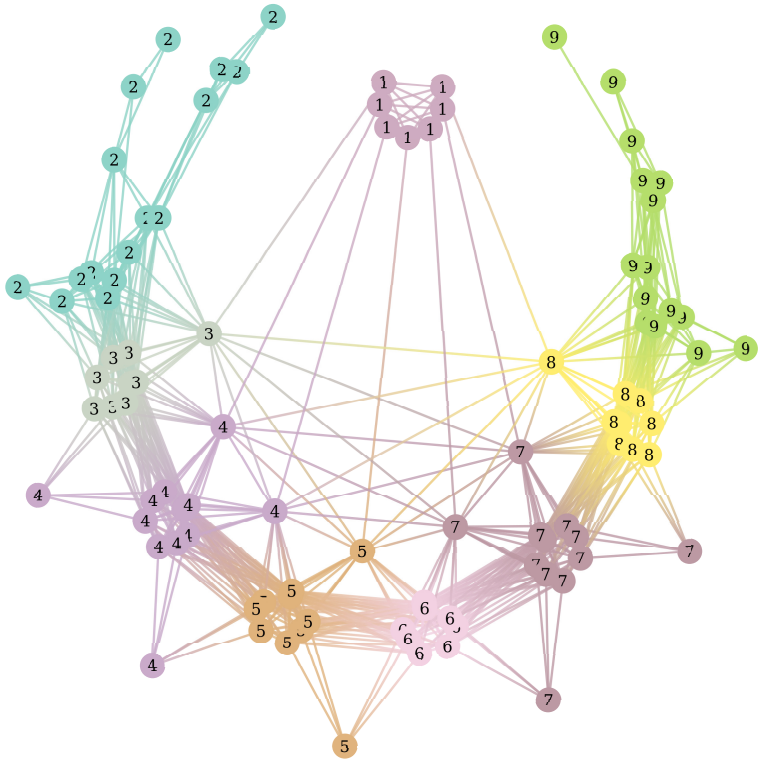

Figure 4.8: Bayesian SBM results of the variable graph of feedtray problem

From the structure of the $\omega$ matrix the graph has a hybrid or two level core-community structure. We can argue that block 1 is in the first level (core) since all the nodes in this block are connected with each other, and this block is connected with the other blocks which contain continuous variables. Blocks 2 to 9 are in the second level, where from the structure of the $\omega$ matrix, we can determine that a dissasortitive community structure exists since $\omega_{ij} > \omega_{ii}$ for the highlighted entries. Given this structure, we solve the problem using Generalized Benders Decomposition (GBD) [99, 108]. The variables in the first block and the constraints that contain only these variables are assigned in the master problem and the other variables and constraints are assigned in the subproblem. Although the variables in the subproblem have a dissasortative community structure, we do not decompose it further since the problem can be solved efficiently. Given this decomposition, the master problem is a mixed integer linear problem which is solved with Gurobi [124] and the subproblem is a nonlinear problem which is solved with IPOPT [285]. GBD converges after four iterations, 2.34 CPU seconds, and the objective function is -13.42 (the global solution from MINLP Library is -13.42). Solving the problem monolithically with BARON [273] the gap after 100 CPU seconds is 412%. We must note that although in this case the global solution of the problem is found, this can not be guaranteed for the general case where the subproblem is nonconvex [256], and the solution depends on the initialization of the GBD algorithm.

These case studies show that application of SBM to optimization problems can reveal the underlying, often complex, block structure of the problem and guide effectively the adoption of the proper decomposition-based solution algorithm. However we must note that the convergence of decomposition-based solution algorithms can not be guaranteed [275, 108, 256, 13], since it depends on the problem formulation (convex/nonconvex objective and constraints), problem decomposition and the algorithm itself (values of the algorithm parameters). Therefore, for a general problem we cannot guarantee that a decomposition-based solution approach based on the underlying block structure is superior to a monolithic approach.

**Remark 4.8.** We note that in the proposed approach the decompositions obtained are optimal from a network structure perspective. Although the examples document improvements over monolithic solution approaches, these decompositions are not guaranteed to be optimal with respect to the computational time or convergence rate. Also, the graph representation of the optimization problem does not differentiate between

convex/nonconvex constraints and continuous/binary variables whose allocation to specific subsystems may lead to further computational improvements. Finally, the computational times for the subproblems generated in the decomposition might not be balanced, hence the decomposition might not fully exploit the benefits of parallel computing. These issues will be addressed in future work.

**Remark 4.9.** In this paper SBM is used to learn the structure and interaction pattern of the variables and constraints at the optimization level (problem formulation) in order to facilitate the solution of these problems. Structure detection can also be used at the linear algebra level [146] to accelerate the solution of the large-scale systems of linear algebraic equations that are solved in the optimization iterations.

## 4.4 Automated structure learning and decomposition based solution of optimization problems

In previous work we have developed DeCODe (**De**tection of **C**ommunities for **O**ptimization **De**composition) [6], a Python package for community-based automatic decomposition of optimization problems. Lagrangean decomposition is then applied based on the community structure of the constraint graph. In this section we describe an extension of this software by considering more general block structures in both the variable and constraint graph of optimization problems, and the application of distributed and hierarchical solution algorithms as discussed in the previous section. The new package, written in Python, is called DecODe (**Dec**omposition of **O**ptimization problems by **De**tection) and is presented in Figure 4.9.

The solution of the optimization problem in DecODe is obtained in three automated steps: learning of the underlying structure, subproblem generation, and decomposition-based solution. Given an optimization problem in Pyomo [129] (Eq. 4.26), first the structure of the problem is learnt. DecODe allows the implementation of community detection (through modularity maximization using the Louvain algorithm [**?**] in the `community` package), core-periphery detection using SBM with two blocks, and SBM without any assumption on the number of the blocks (using Maximum Likelihood Estimation or Bayesian inference in `graph-tool` [235]). These algorithms can be applied to the constraint and variable graphs (created using `Networkx` [126]) of the optimization problem. The pseudo-code for this task is presented in Algorithm 4.1.

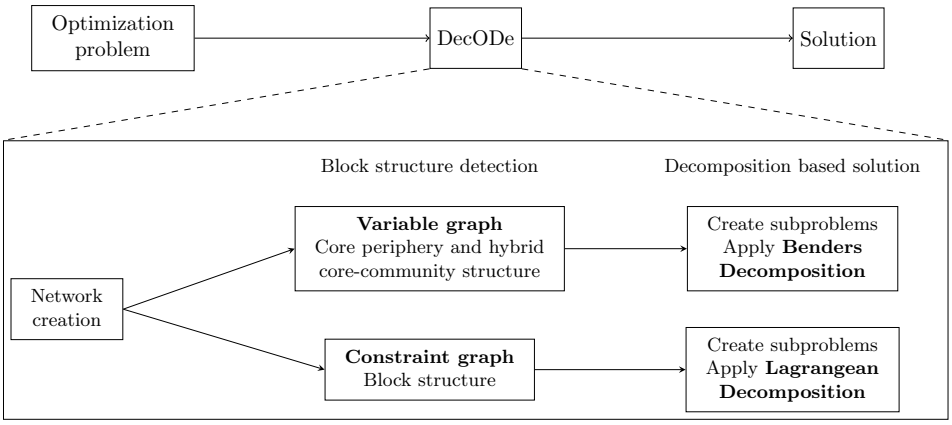


Figure 4.9: General overview of the DecODe Python package

Based on the learnt structure of the constraint and variable graph, different solution algorithms can be applied. A block structure in the constraint graph is used as the basis for the application of Lagrangean decomposition. First the original problem is decomposed into the subproblems based on the partition of the constraints (Algorithm 4.2). Given the partition of the constraints, the variables ($x_c$) that couple constraints that belong to different blocks are identified, a copy is created ($\bar{x}_c$) and equality constraints are added $x_c = \bar{x}_c$. The dualization of these constraints decouples the constraints and the subproblems can be solved independently.

Given the subproblems and the associated dual variables, Lagrangean decomposition is applied (Algorithm 4.3). The update of the dual variables is done using subgradient ascent and the Fischer formula [93]. The initial step size and its reduction schedule are parameters that can be selected by the user. The update of the upper bound is done based on the heuristic of fixing the binary variables.

A core-periphery structure in the variable graph is used as the basis for the application of Benders decomposition [99, 27, 108]. First the affiliation of the binary variables is analyzed. The algorithm requires that the binary variables should be assigned in the master problem. Thus given the learnt partition, if the above requirement is met, the variables are assigned in the master and subproblems. We must note that in this approach the master problem can have both binary and continuous variables. This is

**Data:** Optimization problem in Pyomo
**Result:** Constraint and variable graphs and the learnt structure: $B_n, B_c, b_n, b_c, \omega_n, \omega_c, b_c^{comm}, b_n^{cp}$

1 import NetworkX, graph-tool and community libraries;
2 Create unipartite variable ($B_n$) and constraint ($B_c$) graphs;
3 Apply Stochastic Blockmodeling to $B_c, B_n$ and obtain network partitions $b_n, b_c, \omega_n, \omega_c$;
4 Apply community detection on the constraint graph to obtain a partition of the constraints $b_c^{comm}$;
5 Apply core-periphery detection on the variable graph to obtain a partition of the variables $b_n^{cp}$;

**Algorithm 4.1:** Learn the structure of the optimization problem

**Data:** Constraint graph $B_c$ and partition $b_c$ or $b_c^{com}$
**Result:** Subproblems, dual variables

1 Identify the coupling variables, $x_c$;
2 Create a copy of each coupling variable, $\bar{x}_c$;
3 Create equality constraints $x_c = \bar{x}_c$;
4 Dualize the constraint from previous step, create dual parameter $\lambda$ and assign variables-constraints to their respective subproblems;
5 Distribute the terms of the objective function based on the group membership of each variable and equally distribute the fixed terms among the blocks;

**Algorithm 4.2:** Decompose the problem into subproblems based on the partition of the constraint graph

different compared to the traditional application of the algorithms, where only the binary variables are assigned in the master problem. Let's define as $y$ the variables in the master problem and $x$ the variables in the subproblem. The constraints that contain only $y$ variables are assigned in the master problem and constraints that contain both $x, y$ variables are assigned in the subproblem. Given this partition the infeasible subproblem is created by relaxing the inequality constraints of the subproblem [99]. Finally the Lagrangean functions and objective functions are created (Algorithm 4.4) .

Given the decomposition of the original problem, Benders decomposition is applied. The algorithm is initialized by solving a feasibility problem and the user must define the tolerance, maximum number of iterations, CPU time and the output is the solution of the master problem, subproblems, and the upper and lower bounds (Algorithm 4.5).

**Data:** Subproblems, dual variables, tolerance=tol, maximum number of iterations=max_iter
**Result:** Values of the variables, upper and lower bound
Define solvers;
Initialize the algorithm, fix the value of the dual variables to zero;
Define upper bound, $UB = \infty$ and lower bound $LB = -\infty$;
**while** $|UB - LB|_2 \geq tol$ *and iteration* $\leq max\_iter$ **do**
    Solve the subproblems and obtain a lower bound, $LB$;
    Update upper bound, $UB$, fixing the binary variables (from the solution of the subproblems in the previous step) in the original problem and solving the problem;
    Update the dual variables;
**end**

**Algorithm 4.3:** Apply Lagrangean decomposition based on the block structure of the constraint graph

## 4.5 Conclusions

The solution of large-scale optimization problems is pivotal for the optimal design and operation of process systems. In this paper we introduced Stochastic Blockmodeling and statistical inference as a framework to learn the underlying structure of optimization problems without any assumption on the structure of the problem. We applied this approach to benchmark optimization problems and we showed that it can learn the underlying complex block structure which can subsequently be used effectively in decomposition-based solution methods. Finally, we presented DecODe a python package for automated structure detection and decomposition-based solution of optimization problems.

**Data:** Variable graph $B_n$ and partition $b_n$ or $b_n^{cp}$
**Result:** Master, subproblem and Infeasible subproblem, Langrangean functions
Identify the set with the binary variables and assign them in the master problem, let's assume the set is $y$;
Assign the other set, $x$, in the subproblem;
Based on the decomposition of the variables, the constraints that depend only on $y$ are assigned into the master problem and constraints that depend on $y, x$ are assigned into the subproblem;
Create $\eta$ variable in the master problem and change the objective function;
Create the infeasible subproblem by relaxing the inequality constraints;
Create the Lagrangean functions $L, \bar{L}$;

**Algorithm 4.4:** Decompose the problem into subproblems based on the partition of the variable graph

**Data:** Master problem ($M$), Subproblem ($S$) and Infeasible subproblem ($IS$) problems, Langrangean functions, tolerance=tol, maximum number of iterations=max_iter
**Result:** UB,LB and the values of the variables
Set upper bound $UB = \infty$ and lower bound $LB = -\infty$;
**while** $|UB - LB| \geq \epsilon$ *and iteration* $\leq$ *max_iter* **do**
    Solve problem $M$ and obtain $y = \bar{y}$, $\eta = LB$;
    Solve problem $S$ for $y = \bar{y}$;
    **if** *problem $S$ is feasible* **then**
        Obtain $x$, dual variables $\lambda, \mu$ and $UB = f(\bar{x}, \bar{y})$;
        Add Benders cut: $\eta \geq L(\bar{x}, y, \bar{\lambda}, \bar{\mu})$;
    **else**
        Solve problem $IS$ for $y = \bar{y}$;
        **if** *problem $IS$ is feasible* **then**
            Obtain $\bar{x}$ and dual variables $\bar{\lambda}, \bar{\mu}$;
            Add feasibility cut: $0 \geq \bar{L}(\bar{x}^k, y, \bar{\lambda}^k, \bar{\mu}^k)$;
        **else**
            break: Problem $IS$ is infeasible
        **end**
    **end**
**end**

**Algorithm 4.5:** Apply Benders decomposition

# Chapter 5
# Efficient Solution of Enterprise-wide Optimization Problems Using Nested Stochastic Blockmodeling

## 5.1 Introduction

The integration of process operations is considered as a promising avenue to improve the economic performance of process systems [114, 75]. This approach considers simultaneously different decision making problems, resulting in large scale optimization problems whose solution is a challenging task. The difficulty arises from the inherently nonlinear behavior of most process systems and the different time scales that are involved. For example, supply chain/planning decisions span a time horizon of months, scheduling decisions are made weekly, and control decisions are made in the time scale of seconds or minutes. The integration of these decision layers leads to multi-scale optimization problems which are generally nonscalable and whose monolithic solution can be intractable.

Different approaches have been proposed to reduce the complexity of such problems and improve the computational time. In one approach the problem is decomposed into distinct steps, where the different problems are solved hierarchically and the solution of the upper level problem is an input to the lower level problems [188]. In this approach, although the computational time is reduced the solution can be suboptimal. A second approach to improve the tractability of a problem is to approximate the computationally complex part with a simpler surrogate model [34]. Typically, this approach is used to handle long time horizons and approximate the behavior of nonlinear systems [188, 230, 306, 52, 65, 57, 4].

A different approach is to exploit the structure of the full problem and apply decomposition based solution methods. Typical examples are the application of Lagrangean and augmented Lagrangean relaxation/decomposition [119, 280, 120, 177], Benders and Generalized Benders decomposition [27, 108, 64, 174], and bilevel decomposition

[144, 264]. Although this approach can reduce the computational time, a decomposition of the problem itself is necessary. The decomposition chosen is typically problem specific or is based on intuition. In general we can argue that decomposition based solution algorithms exploit some block structure in the problem [69, 76]. For example, in Benders decomposition the problem is decomposed into a master problem and a subproblem, where the latter provides information about the effect of the master problem on the solution of the subproblem. This can be considered as a hierarchical structure since the solution of the subproblem depends on the solution of the master problem. On the contrary, in Lagrangean relaxation/decomposition, the problem is decomposed into subproblems which are coupled through a number of constraints, called complicating constraints. This partition implies a weakly coupled block structure in the problem, where the value of the coupling constraints affects the solution of the subproblems. Despite this direct relation between block structure and decomposition based solution algorithms, the block structure of an optimization problem is not always apparent. Therefore, the detection of the underlying structure of a problem is an important first step towards the selection of the most appropriate decomposition based solution algorithm.

In recent research, we have proposed the application of network science concepts and tools in order to systematically decompose optimization problems [6, 198]. In this approach the structural coupling among the variables and constraints of an optimization problem is captured through the variable and constraint graphs. In the variable graph, the nodes are the variables of an optimization problem, the edges are the constraints that couple two variables and a weight, which is equal to the number of such constraints, can be assigned to each edge. Similarly, in the constraint graph, the nodes are the constraints of the problem and the edges are the variables that are present in the constraints. This graph representation of an optimization problem allows the systematic analysis of its structure using tools from network science. The first tool that was employed was community detection [102]. This approach can provide high quality decompositions whereby groups (blocks) of variables or constraints are identified with weak interactions (in a statistical sense) [6, 271]. Centrality analysis can be used in conjunction to reveal the hierarchical relation among the communities [198]. Although this approach can provide decompositions that can reduce the computational time, the main assumption is that the graph and hence the optimization problem has a community structure. In [207] we have proposed the application of Stochastic Blockmodeling

(SBM) and statistical inference as a framework to learn the underlying structure of optimization problems without any a-priori assumptions on the structure of the problem. Stochastic Blockmodels are random graph generative models that allow the generation of random graphs with arbitrary block structure [110]. Statistical inference allows learning the SBM that generated a given problem and therefore detecting its underlying block structure. The estimated structure can then be used as the basis for the application of decomposition based solution algorithms, which can reduce the necessary time to obtain a solution or an estimate of its lower and upper bound. However, neither of these approaches (community detection or SBM) can detect the possible hierarchical or multi-scale structure of a problem.

In this paper, we propose the application of *nested* Stochastic Blockmodeling (nSBM) for learning the hierarchical block structure of optimization problems. nSBM describes a nested hierarchy of Stochastic Blockmodels where the connections among the nodes depend solely on their block affiliation [236, 240, 133]. The parameters of the model, which are estimated through statistical inference [236, 237], reveal the block structure of the problem at different hierarchical levels and the hierarchy itself. We apply this approach to optimization problems that arise in the integration of process operations, and we show that it can indeed learn the hierarchical multi-scale structure of these problems and guide the application of decomposition based solution algorithms.

In the rest of the manuscript, we begin by presenting the nSBM model, the inference problem, and solution approaches. In the next two sections we apply this approach to the integration of scheduling and dynamic optimization and the integration of planning, scheduling and dynamic optimization. For both cases, we analyze the structure of the problem and compare the decompositions that are obtained at different hierarchical levels.

## 5.2 Nested Stochastic Blockmodeling and Bayesian Inference

In this section the stochastic model and inference approaches will be presented for the simple nested Stochastic Blockmodel in order to keep the notation simple. In the end of the section we will discuss extensions and variants of this basic model.

### 5.2.1 Stochastic Blockmodel

Consider an undirected graph $G(V, E)$ with $N$ nodes, $M$ edges ($|V| = N, |E| = M$) and adjacency matrix $A \in \mathbb{R}^{N \times N}$, where $A_{ij} = A_{ji}$ is equal to the number of edges between node $i$ and $j$. We will assume that the nodes are assigned into $B$ blocks and will define a partition vector $b \in \mathbb{R}^N$, where $b_i \in \{1, ..., B\}$ denotes the group membership of node $i$. We also define the matrix $\omega \in \mathbb{R}^{B \times B}$, where $\omega_{rs}$ is equal to the number of edges between the nodes that belong in block $r$ and the nodes that belong in block $s$, and $\omega_{rr}$ is equal to twice the number of edges between the nodes in block $r$. The entries of the $\omega$ matrix are equal to:

$$\omega_{rs} = \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} \delta_{b_i r} \delta_{b_j s} \ \forall r = 1, ..., B, s = 1, ...B, \tag{5.1}$$

where $\delta$ is the Kronecker delta. Finally, we define $n \in \mathbb{R}^B$, where $n_r$ is equal to the number of nodes in block $r$ and is equal to:

$$n_r = \sum_{i=1}^{N} \delta_{b_i r} \ \forall r = 1, .., B. \tag{5.2}$$

We note that for a given graph with adjacency matrix $A$, both $\omega$ and $n$ depend on the partition of the nodes $b$.

For a simple graph ($A_{ij} \in \{0, 1\}$), given a number of nodes $N$ and a partition $b$ into $B$ blocks, different graphs with the same $\omega$ matrix can be generated, and all the graphs are equally probable. This leads to an ensemble of graphs where the total number of different graphs is equal to [233]:

$$\Omega(\omega, b) = \prod_{s=1, r=1, r \geq s}^{B} \Omega_{rs}, \tag{5.3}$$

where

$$\Omega_{rs} = \binom{n_r n_s}{\omega_{rs}} \ , \ \Omega_{rr} = \binom{\binom{n_r}{2}}{\frac{\omega_{rr}}{2}}. \tag{5.4}$$

The first expression above is equal to all the possible ways to select $\omega_{rs}$ edges from $n_r n_s$ ($n_r n_s$ is equal to all the possible edges between all the nodes in block $r$ and all the nodes in block $s$). Similarly, the second expression is equal to the number of ways to assign

$\omega_{rr}/2$ edges between the nodes in block $r$. Since all these graphs are equally probable, the probability to observe a graph $G$ given a partition $b$ is equal to

$$P(G|b) = \frac{1}{\Omega(\omega, b)} \tag{5.5}$$

and the entropy of the ensemble is equal to

$$S_g(\omega, b) = \ln \Omega(\omega, b). \tag{5.6}$$

For a multigraph, defined as a graph with multiple edges between two nodes ($A_{ij} \in \mathbb{Z}_+$), the different ways to create a graph with $N$ nodes and a partition $b$ into $B$ blocks are equal to [233]

$$\Omega_m(\omega, b) = \prod_{r=1,s=1,r\geq s}^{B} \Omega_{rs}^m, \tag{5.7}$$

where

$$\Omega_{rs}^m = \binom{n_r n_s + \omega_{rs} - 1}{\omega_{rs}}, \; \Omega_{rr}^m = \binom{\frac{n_r n_r}{2} + \frac{\omega_{rr}}{2} - 1}{\frac{\omega_{rr}}{2}}. \tag{5.8}$$

The first expression is equal to the number of $\omega_{rs}$ combinations with repetition from a set with size $n_r n_s$. Similarly as in the case of a simple graph, all multigraphs are equally probable, the probability to observe a multigraph is $1/\Omega_m(\omega, b)$, and the entropy is

$$S_m(\omega, b) = \ln \Omega_m(\omega, b). \tag{5.9}$$

### 5.2.2 Nested Stochastic Blockmodel

The nSBM is based on the idea that for a given graph $G(V, E)$ (simple or multigraph), a partition $b$ into $B$ blocks leads to an $\omega$ matrix whose entries are positive integer values ($\omega_{rs} \in \mathbb{Z}_+$). Therefore this $\omega$ matrix can be considered as the adjacency matrix of a new multigraph $G'$ with $B$ nodes and $E$ edges, i.e. $\omega = A'$, where $A'$ is the adjacency matrix of mutigraph $G'$. Similarly, the nodes in this new multigraph can be partitioned into $B'$ blocks leading in turn to a new $\omega'$ matrix, which can be considered as the adjacency matrix of a new multigraph $G''$ with $B'$ nodes. This procedure can be continued until one block is left, leading to a nested sequence of Stochastic Blockmodels, where the first level simple graph is the observed network and the multigraphs in the other levels form the nested model [236]. An example of a nested stochastic blockmodel with three levels is presented in Figure 5.1. The colors used to denote the blocks in a specific level do

not have any association with the colors in the other levels.

Let's assume that $l = L$ levels exist and $l = 0$ corresponds to the observed network. The number of nodes in level $l$ multigraph is $B_l$ ($B_l \leq B_{l-1}$) and the number of edges is $E$. Based on the definition of the nSBM, the last level $l = L$ has one block ($B_L = 1$). We define $\omega^l \in \mathbb{R}^{B_l \times B_l}$, where $\omega^l_{rs}$ is equal to the number of edges between the nodes in block $r$ and $s$ in level $l$. We also define $n^l \in \mathbb{R}^{B_l}$, where $n^l_r$ is the number of nodes in block $r$ in level $l$, and $b^l \in \mathbb{R}^{B_l}$ is the partition of the nodes in level $l$ into $B_l$ blocks. Given these definitions the following equations hold

$$B_{l-1} = \sum_{r=1}^{B_l} n^l_r \, , \; E = \sum_{r=1}^{B_l} \sum_{s=1}^{B_l} \omega^l_{rs}/2 \tag{5.10}$$

$$\omega^l_{rs} = \sum_{i=1}^{B_{l-1}} \sum_{j=1}^{B_{l-1}} A^l_{ij} \delta_{b^l_i r} \delta_{b^l_j s} \; \forall r = 1, ..., B_{l-1}, s = 1, ...B_{l-1}. \tag{5.11}$$

Based on these equations, the total number of edges is the same in all the levels and the number of nodes decreases. The entropy of the nested SBM is equal to:

$$S_n = S_g(\omega^0, b^0) + \sum_{l=1}^{L} S_m(\omega^l, b^l), \tag{5.12}$$

where $S_g, S_m$ are given by Eq. 5.6, 5.9. The first term is the entropy of the observed simple graph, and the second term accounts for the entropies of the multigraphs in each level $l$. Since all graphs at all levels are equally probable, the probability to observe a nested SBM is equal to

$$P = \frac{1}{\Omega(\omega^0, b^0) \prod_{l=1}^{L} \Omega^l_m(\omega^l, b^l)} \tag{5.13}$$

### 5.2.3 Inference approach

Given a graph $G(V, E)$ the goal is to infer the block partition and $\omega$ matrix for all the levels that best fit the data (the observed graph). Two approaches can be followed to solve the problem, Maximum Likelihood Estimation (MLE) and Bayesian Inference. In this work, we will focus on the second approach, which involves the estimation of the posterior probability of the observed network. If we assume that one level exists

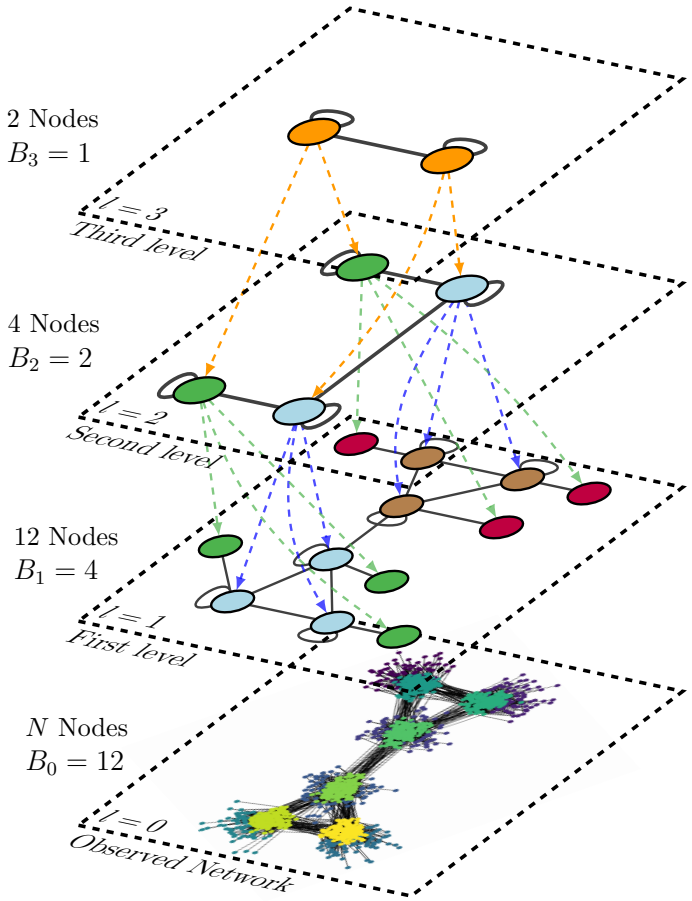


Figure 5.1: Example of a nested SBM (reproduced/adapted with permission from [240]. Copyright 2020 Wiley). The observed graph has $N = 1273$ nodes, $E = 8309$ edges and the nodes of the observed network are partitioned into $B_0 = 12$ blocks. The hollow circles denote self-edges.

($L = 1$), then from Bayes' rule:

$$P(b|A) = \frac{P(A|b)P(b)}{P(A)}, \tag{5.14}$$

where

$$P(A|b) = P(A|\omega, b)P(\omega|b)\,,\; P(A) = \sum_b P(A|b)P(b). \tag{5.15}$$

In Bayesian inference the goal is to find the probability distribution $P(b|A)$. Based on this distribution different partitions can be sampled, and the partition that maximizes the probability can be found. The numerator in the above expression can be written as:

$$P(A|b)P(b) = P(A|\omega, b)P(\omega|b)P(b) = P(A|\omega, b)P(\omega, b). \tag{5.16}$$

Therefore the original task of maximizing $P(A|b)$ is equivalent to maximizing $P(b|A)$ or minimizing $-P(b|A)$ which is equal to:

$$\underset{b,\omega}{\text{minimize}} - \log_2 P(A|\omega, b) - \log_2 P(\omega, b). \tag{5.17}$$

This objective function has an information theoretical interpretation. The first term is the amount of bits necessary to encode the observed data, and the second term is the amount of bits necessary to encode the parameters of the model. Hence, the Bayesian inference approach lends itself naturally to the usage of the description length $\Sigma = -\log_2 P(A|\omega, b) - \log_2 P(\omega, b)$ as the objective function and avoids overfitting. An increase in the number of blocks leads to a reduction in the first term, but the model is more complex leading to an increase in the second term.

For the nested SBM case, the goal is to estimate or maximize $P(\{b^l\}_{l=1}^L|A)$ which is equal to [237]

$$P(\{b^l\}_{l=1}^L|A) = \frac{P(A|\{b^l\}_{l=1}^L)P(\{b^l\}_{l=1}^L)}{P(A)}. \tag{5.18}$$

Similarly to the case of a single level, the numerator can be written as

$$P(A|\{\omega^l\}_{l=1}^L, \{b^l\}_{l=1}^L)\; P(\{\omega^l\}_{l=1}^L, \{b^l\}_{l=1}^L), \tag{5.19}$$

and the inference problem is:

$$\underset{\{b^l\}_{l=1}^L,\{\omega^l\}_{l=1}^L}{\text{minimize}} \left( -\log_2 P(A|\{\omega^l\}_{l=1}^L,\{b^l\}_{l=1}^L) - \log_2 P(\{\omega^l\}_{l=1}^L,\{b^l\}_{l=1}^L)\right). \tag{5.20}$$

In this paper, we will use the microcanonical ensemble approach to define the prior distribution [237]. The prior for the partition in all levels, $P(\{b^l\}_{l=1}^{B^l})$, is equal to

$$P(\{b^l\}_{l=1}^L) = \prod_{l=1}^{L} P(b^l). \tag{5.21}$$

We can assume that the size of the different blocks depends on the number of blocks. Therefore, given the sizes of the different blocks, $n^l$, the probability to observe $b^l$ is equal to

$$P(b^l|n^l) = \frac{\prod_{r=1}^{B_l} n_r^l!}{B_{l-1}!}. \tag{5.22}$$

Additionally, the different ways to assign the nodes in level $l$ into $B_l$ nonempty blocks is $\binom{B_{l-1}-1}{B_l-1}$. Hence, the probability to obtain $n^l$ given $B^l$ is

$$P(n^l|B_l) = \binom{B_{l-1}-1}{B_l-1}^{-1}. \tag{5.23}$$

Finally, we can assume that the probability of having $B^l$ blocks is $P(B^l) = 1/B_{l-1}$, where $B_{l-1}$ is the maximum number of blocks in level $l$, i.e. every node is assigned into one block. Overall, the prior probability of the block assignments in level $l$ is

$$P(b^l) = P(b^l|n^l)P(n^l|B_l)P(B_l) = \frac{\prod_{r=1}^{B_l} n_r^l!}{B_{l-1}!}\binom{B_{l-1}-1}{B_l-1}^{-1} B_{l-1}^{-1}. \tag{5.24}$$

The prior for $P(\{\omega^l\}_{l=1}^L|\{b^l\}_{l=1}^L)$ is equal to

$$P(\{\omega^l\}_{l=1}^L|\{b^l\}_{l=1}^L) = \prod_{l=1}^{L} P(\omega^l|\omega^{l+1},b^l) = \prod_{l=1}^{L} P(A^{l+1}|\omega^{l+1},b^l) \tag{5.25}$$

where

$$P(\omega^l|\omega^{l+1}, b^l) = \prod_{r<s} \binom{n_r^l n_s^l + \omega_{rs}^{l+1} - 1}{\omega_{rs}^{l+1}} \times \prod_r \binom{n_r^l(n_r^l+1)/2 + \omega_{rs}^{l+1}/2 - 1}{\omega_{rs}^{l+1}/2}^{-1}. \tag{5.26}$$

Based on the above equation, the probability $P(\omega^l|\omega^{l+1}, b^l)$ to observe a multigraph in level $l+1$ with adjacency matrix $A^{l+1} = \omega^l$, depends on $b^l, \omega^{l+1}$. $b^l$ dictates the number of nodes in level $l+1$ and $\omega^{l+1}$ dictates the connection pattern among the nodes. Note that the expression for $P(\omega^l|\omega^{l+1}, b^l)$ is equal to $1/\Omega_m^l$. Given the above prior distributions, the posterior is estimated using a Markov Chain Monte Carlo approach as described in [240, 237]. In this method, at each level, node move proposals are made based on the block affiliation of the neighbor nodes using a Metropolis-Hastings criterion which guarantees ergodicity, i.e. all possible partitions are possible.

The above formulation and inference approach for the basic nSBM model can be extended to account for more general models. Specifically, we first mention the degree-corrected nSBM [237, 240]. In the simple nSBM, nodes with high degree tend to be assigned to the same block. This issue is resolved using the degree corrected version (DC-nSBM), where the prior of the degree distribution depends on some hyper-parameters which are estimated from the data, i.e. observed network.

The second variant of the nSBM, called weighted nSBM [238], accounts for edge weights, which denote the strength of coupling among the nodes. We define the observed weights as $x$, where $x_{ij}$ is the weight of an edge between node $i$ and $j$, and the probability distribution of the weights depends solely on the group membership of nodes $i$ and $j$. If we assume for simplicity that only one level exists, then given a number of nodes $n$, a partition $b$ and $\gamma$, a parameter that governs the sampling of the weights, a graph with adjacency matrix $A$ and weights $x$ can be generated, and the probability to observe it is equal to

$$P(A, x|\theta, \gamma, b) = P(x|A, \gamma, b)P(A|\theta, b), \tag{5.27}$$

where $\theta$ are the parameters of the model. The partition that maximizes the probability to observe a partition $b$, given a graph $A$ and weights $x$ is

$$P(b|A, x) = \frac{P(A, x|b)P(b)}{P(A, x)}. \tag{5.28}$$

The posterior distribution can be estimated following a Bayesian inference approach. Based on the data, different models can be used for the weights. We refer the reader to [238] for a detailed explanation of the weighted nSBM model and the inference approach. We must note that although the number of blocks in the observed network $B_0$ can be inferred from the data it can also be used as tunable hyperparameter.

In the subsequent sections we illustrate the application of this modeling and inference framework to two different classes of enterprise-wide optimization problems.

## 5.3 Integration of Scheduling and Dynamic Optimization

### 5.3.1 Optimization model

First we will consider the integration of scheduling and dynamic optimization for continuous parallel lines. The problem formulation is based on [98]. We assume that $N_p$ ($\mathcal{I}_p = \{1, ..., N_p\}$) products must be produced, $N_l$ ($\mathcal{I}_l = \{1, ..., N_l\}$) production lines are available and the time horizon in each line, $H_l$, is divided into $N_s$ ($\mathcal{I}_s = \{1, ..., N_s\}$) slots.

#### Scheduling model

First we define variable $Y_{ikl} \in \{0, 1\}$ which is one if product $i$ is produced at slot $k$ in line $l$ and zero otherwise. In each slot and line only one product can be produced and each product is produced at least once. These constraints are modeled through the following equations:

$$\begin{aligned} &\sum_{i=1}^{N_p} Y_{ikl} = 1 \ \forall \ k \in \mathcal{I}_s, l \in \mathcal{I}_l \\ &\sum_{l=1}^{N_l} \sum_{k=1}^{N_s} Y_{ikl} \geq 1 \ \forall \ i \in \mathcal{I}_p. \end{aligned} \tag{5.29}$$

Additionally, variable $z_{ijkl} \in [0, 1]$ is introduced to denote a transition from product $i$ to product $j$ at slot $k$ and line $l$, which depends on the value of the $Y_{ikl}$ variables as

follows:

$$\begin{aligned}\sum_{i=1}^{N_p} z_{ijkl} &= Y_{jkl} \quad \forall\ j \in \mathcal{I}_p, k \in \mathcal{I}_s, l \in \mathcal{I}_l \\ \sum_{j=1}^{N_p} z_{ijkl} &= Y_{ik-1l} \quad \forall\ i \in \mathcal{I}_p, k \in \mathcal{I}_s, k \neq 1, l \in \mathcal{I}_l.\end{aligned} \tag{5.30}$$

Each line is composed from $N_s$ slots and the total production time of product $i$ in slot $k$ and line $l$ is $t_{ikl}^{prod}$. This time is the sum of the production time and the transition time $\theta_{kl}^{t}$. The timing constraints are the following:

$$\begin{aligned}t_{ikl}^{prod} &\leq t_{max} Y_{ikl} \ \forall\ i \in \mathcal{I}_p, k \in \mathcal{I}_s, l \in \mathcal{I}_l \\ H_l &= \sum_{i=1}^{N_p} \sum_{k=1}^{N_s} t_{ikl}^{prod} \ \forall l \in \mathcal{I}_l\end{aligned} \tag{5.31}$$

The amount of product $i$ produced in slot $k$ at line $l$ is given by $W_{ikl}$ and the production rate is $r_{il}$. The amount of product $i$ that is produced must satisfy the demand ($d_i$). The constraints are the following:

$$\begin{aligned}W_{ikl} &= r_{il}(t_{ikl}^{prod} - \theta_{kl}^{t} Y_{ikl}) \ \forall\ i \in \mathcal{I}_p, k \in \mathcal{I}_s, l \in \mathcal{I}_l \\ \sum_{k=1}^{N_s} \sum_{l=1}^{N_l} \frac{W_{ikl}}{H_l} &= d_i \ \forall\ i \in \mathcal{I}_p\end{aligned} \tag{5.32}$$

The objective of the scheduling model is to minimize the cost which consists of the production, transition and storage cost and is given by the following equation:

$$\sum_{i=1}^{N_p} \sum_{k=1}^{N_s} \sum_{l=1}^{N_l} \left( C_{il}^{p} \frac{t_{ikl}^{prod}}{H_l} + c_{il}^{stor} W_{ikl} \right) + \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} \sum_{k=1}^{N_s} \sum_{l=1}^{N_l} c_{ijl}^{trans} \theta_{kl}^{t} \frac{z_{ijkl}}{H_l}, \tag{5.33}$$

where $c_{il}^{p}$ is the production cost of product $i$ at line $l$, $c_{ij}^{trans}$ is the transition cost between product $i$ and $j$, and $c_{il}^{stor}$ is the inventory cost of product $i$ in line $l$. These parameters are calculated using the following equations

$$c_{il}^{p} = c_{i}^{prod} r_{il}\ ,\ c_{ijl}^{trans} = 0.9 c_{i}^{prod} r_{il}\ ,\ c_{il}^{stor} = 0.5(1 - d_i/r_{il}) c_{i}^{inv}, \tag{5.34}$$

where $c_{i}^{prod}$ is the production cost of product $i$ and $c_{i}^{inv}$ is the inventory holding cost.

**Dynamic model**

We will assume that the dynamic behavior of the system for each line $l$ can be modeled by a general system of differential equations:

$$\dot{x}_l^n = F_l(x_l^n, u_l^n) \ \forall l \in \mathcal{I}_l \tag{5.35}$$

where $x_l^n$ is the value of state $n$ at line $l$. The dynamic model is discretized using collocation on finite elements and the discretized equations are ($N_{fe}, N_{cp}$ is the number of finite elements and collocation points respectively, and $\mathcal{I}_f = \{1, ..., N_{fe}\}, \mathcal{I}_c = \{1, ..., N_{cp}\}$):

$$\begin{aligned}
x_{fckl}^n &= x0_{fkl}^n + h_{kl}^{fe} \sum_{m=1}^{N_{cp}} \Omega_{mc} \dot{x}_{fmkl}^n \ \ \forall n, f \in \mathcal{I}_f, c \in \mathcal{I}_c, k \in \mathcal{I}_s, l \in \mathcal{I}_l \\
x0_{fkl}^n &= x0_{f-1kl}^n + h_{kl}^{fe} \sum_{m=1}^{N_{cp}} \Omega_{mc} \dot{x}_{f-1,mkl}^n \ \ \forall n, f \in \mathcal{I}_f, f \geq 2, \\
&\qquad\qquad\qquad\qquad\qquad\qquad c \in \mathcal{I}_c, k \in \mathcal{I}_s, l \in \mathcal{I}_l \\
\dot{x}_{fckl}^n &= F^n(x_{fckl}^n, u_{fckl}^m) \ \ \forall n, f \in \mathcal{I}_f, c \in \mathcal{I}_c, k \in \mathcal{I}_s, l \in \mathcal{I}_l
\end{aligned} \tag{5.36}$$

$$h_{kl}^{fe} = \frac{\theta_{kl}^t}{N_{fe}} \ \ \forall k \in \mathcal{I}_s, l \in \mathcal{I}_l \tag{5.37}$$

Finally, we define $t_{fckl}^d$ which is equal to the time at finite element $f$, collocation point $c$ at slot $k$ in line $l$, and the following equation holds:

$$t_{fckl}^d = h_{kl}^{fe}(f - 1 + \gamma_c) \ \ \forall f \in \mathcal{I}_f, c \in \mathcal{I}_c, k \in \mathcal{I}_s, l \in \mathcal{I}_l, \tag{5.38}$$

where $\gamma_c$ is the root of the Lagrange orthogonal polynomial at collocation point $c$. The dynamic model is used to find the optimal transition profiles for the states and manipulated variables of the problem. The objective function for a transition from $(x_0, u_0)$ to $(x_f, u_f)$ is given by:

$$\int_0^{t_f} (u - u_f)^2 dt \approx \frac{1}{N_{fe}} \sum_{f=1}^{N_{fe}} \sum_{c=1}^{N_{cp}} t_{fckl}^d \Lambda_{c,N_{cp}} (u_{fc} - u_f)^2 \tag{5.39}$$

where $\gamma_c$ is the Radau root at collocation point $c$ and $\Lambda$ is the collocation matrix.

**Integrated problem**

The steady state value of state $n$ and manipulated variable $m$ for product $i$ is $x^{ss}_{ni}$ and $u^{ss}_{mi}$ respectively. Next we define $x^{in}_{nkl}, x^{end}_{nkl}, u^{in}_{mkl}, u^{end}_{mkl}$ which is the value of state $n$ and manipulated variable $m$ at the beginning and end of slot $k$ at line $l$. These variables are related to the scheduling variables through the following constraints:

$$\begin{aligned}
x^{in}_{nkl} &= \sum_{i=1}^{N_p} x^{ss}_{ni} Y_{ik-1l} \ \forall n, k \in \mathcal{I}_s, k > 1, l \in \mathcal{I}_l \\
x^{in}_{n1l} &= \sum_{i=1}^{N_p} x^{ss}_{ni} Y_{iN_sl} \ \forall n, l \in \mathcal{I}_l \\
u^{in}_{mkl} &= \sum_{i=1}^{N_p} u^{ss}_{mi} Y_{ik-1l} \ \forall m, k \in \mathcal{I}_s, k > 1, l \in \mathcal{I}_l \\
u^{in}_{m1l} &= \sum_{i=1}^{N_p} u^{ss}_{mi} Y_{iN_sl} \ \forall m, l \in \mathcal{I}_l \\
x^{end}_{nkl} &= \sum_{i=1}^{N_p} x^{ss}_{ni} Y_{ikl} \ \forall n, k \in \mathcal{I}_s, l \in \mathcal{I}_l \\
u^{end}_{mkl} &= \sum_{i=1}^{N_p} u^{ss}_{mi} Y_{ikl} \ \forall m, k \in \mathcal{I}_s, l \in \mathcal{I}_l
\end{aligned} \tag{5.40}$$

$$\begin{aligned}
x0^{n}_{1kl} &= x^{in}_{kl} \ \forall k, l \\
x^{n}_{N_{fe}N_{cp}kl} &= x^{end}_{kl} \ \forall n, k, l \\
u^{m}_{11kl} &= u^{in}_{kl} \ \forall n, k, l \\
u^{m}_{N_{fe}N_{cp}kl} &= u^{end}_{kl} \ \forall n, k, l
\end{aligned} \tag{5.41}$$

Table 5.1: Steady state conditions, production and inventory cost for all the products (the production rate for each product is the same in both production lines), $a_u = 0.01$

| Product | $c^{ss}(mol/L)$ | $Q^{ss}(L/h)$ | Production cost ($/kg$) | Inventory cost ($/kgh$) | Production rate ($kg/h$) | Demand rate ($kg/hr$) |
|---|---|---|---|---|---|---|
| 1 | 0.24 | 200 | 120 | 5 | 40 | 18 |
| 2 | 0.2 | 150 | 150 | 5.5 | 80 | 14 |
| 3 | 0.3032 | 130 | 130 | 7.8 | 278.8 | 17 |
| 4 | 0.32 | 1000 | 125 | 9 | 607 | 16 |

The objective of the integrated problem is to minimize the cost and the optimization problem is the following ($a_u$ is a weight coefficient):

$$\begin{aligned}\text{minimize} \quad & \sum_{i=1}^{N_p}\sum_{k=1}^{N_s}\sum_{l=1}^{N_l} c_{il}^{p}\frac{W_{ikl}}{H_l} + \sum_{i=1}^{N_p}\sum_{j=1}^{N_p}\sum_{j=1}^{N_s}\sum_{l=1}^{N_l} c_{ij}^{trans}\theta_{kl}^{t}\frac{Z_{ijkl}}{H_l} \\ & + \sum_{i=1}^{N_p}\sum_{k=1}^{N_s}\sum_{l=1}^{N_l} c_{il}^{stor}W_{ikl} \\ & + a_u \sum_{l=1}^{N_l}\sum_{k=1}^{N_s}\sum_{f=1}^{N_{fe}}\sum_{c=1}^{N_{cp}} \frac{t_{fckl}^{d}}{N_{fe}}\Lambda_{c,N_{cp}}(u_{fckl} - u_{kl}^{end})^2 \\ \text{subject to} \quad & \text{Equations 5.29,5.30, 5.31, 5.32, 5.36,5.37 5.38, 5.40, 5.41}\end{aligned} \tag{5.42}$$

### 5.3.2 Application of Nested Stochastic Blockmodeling

In this case study, we will assume that four products must be produced in two identical isothermal continuous stirred reactors (2 lines) with two slots. The dynamic behavior of the reactors is modeled by the following equation:

$$\frac{dc}{dt} = \frac{Q}{V}(c_{feed} - c(t)) + kc(t)^3, \tag{5.43}$$

where $c$ is the concentration of the reactant, $Q$ is the inlet flowrate (manipulated variable) and $V = 5000\ L, c_{feed} = 1\ mol/L, k = 2\ L^2/(hr\ mol^2)$ are the volume, inlet concentration and reaction constant parameter. The economic data of the problem are presented in Table 5.1. First, we analyze the structure of the variable graph solving the inference problem using Bayesian inference in graph-tool [235]. We present the results for $B_0 = 6$ blocks because this resulted in a decomposition which is suitable for the application of a decomposition based solution algorithm and makes intuitive sense.

As discussed earlier, in the variable graph the nodes are the variables of the problem and an edge corresponds to the constraints that couple two variables. Each edge has a weight which is equal to the number of coupling constraints. Since these weights are positive integer values we assume that the weights in the nSBM model follow a discrete geometric distribution (see Remark 3 for additional discussion on this assumption). The observed network (variable graph) is decomposed into 6 blocks (Figure 5.2) and the $\omega$ matrix for the partition of the observed graph ($\omega_0$) is:

$$\omega_0 = \begin{bmatrix} 336 & 48 & 4 & 4 & 4 & 4 \\ 48 & 177 & 1 & 1 & 1 & 1 \\ 4 & 1 & 756 & 0 & 0 & 0 \\ 4 & 1 & 0 & 756 & 0 & 0 \\ 4 & 1 & 0 & 0 & 756 & 0 \\ 4 & 1 & 0 & 0 & 0 & 756 \end{bmatrix},$$

where the colors in the entries correspond to the graph shown in Figure 5.2.

Based on the structure of this matrix the observed graph has a hybrid multi-core community structure. The two blocks in the center of the graph form the two cores and correspond to the scheduling variables. The purple block contains variables $Y_{ikl}$, $z_{ijkl}$, $x^{in}_{kl}$, $x^{end}_{kl}$, $u^{in}_{kl}$, $u^{end}_{kl}$ and the pink block contains the other scheduling variables. The variables that are related to the dynamic behavior of the system ($x_{fckl}, u_{fckl}, t^{d}_{fckl}, h^{fe}_{kl}$) for each slot and line are assigned in the other blocks. These blocks can be considered as communities, since the variables (nodes) in a block are highly coupled with the other variables in the block and loosely coupled with the variables in the two blocks in the core. This structure leads to the nested or double L shape of the $\omega_0$ matrix.

Based on the inference results three levels are identified (Figure 5.3) in the nSBM model. The graph in the first level is partitioned into three blocks. This partition reveals a core-periphery structure where the two blocks in the middle correspond to the scheduling variables and the other nodes correspond to the variables for the dynamic optimization problem in each slot and line. This structure is evident in the $L$ shape of the $\omega_1$ matrix

$$\omega_1 = \begin{bmatrix} 561 & 10 & 10 \\ 10 & 1512 & 0 \\ 10 & 0 & 1512 \end{bmatrix}$$

Although the nodes for the four dynamic optimization problems are partitioned into

Figure 5.2: Partition of the variable graph using Nested Stochastic Blockmodeling with maximum number of blocks equal to 6. The nodes with purple color are $Y_{ikl}, z_{ijkl}, x^{in}_{kl}, x^{end}_{kl}, u^{in}_{kl}, u^{end}_{kl}$, the nodes with pink color are $\theta^{t}_{kl}, W_{ikl}, H_{l}, t^{prod}_{ikl}$ and the nodes with the other colors correspond to the variables for a slot and line $x_{nfckl}, u_{mfckl}, t^{d}_{fckl}, h^{fe}_{kl}$

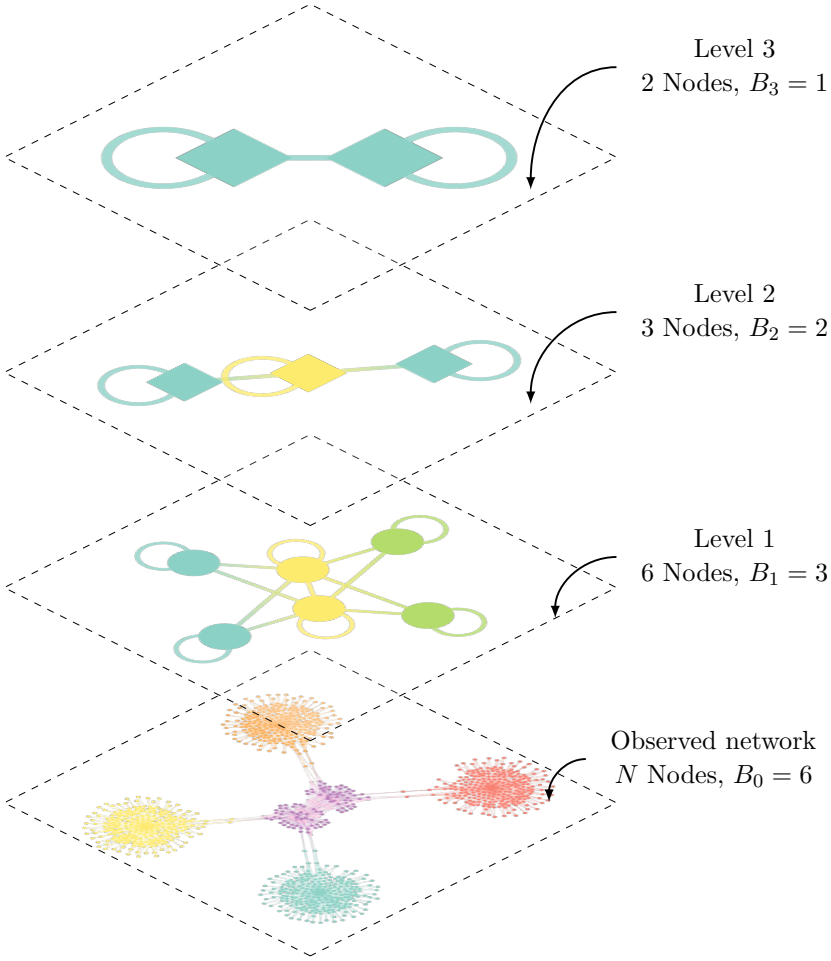


Figure 5.3: Inferred nSBM model of the variable graph for the integrated scheduling and dynamic optimization problem. The hollow cycles indicate self-edges.

two blocks (blue and green nodes), the variables are decoupled since the nodes are not connected directly, i.e. there does not exist an edge between these nodes. This partition is a 'coarser' partition of that of the observed graph. The multigraph in the second level is partitioned into two blocks. The yellow square node contains the scheduling variables and each blue node contains the variables associated with the dynamic optimization for each line. Finally, in the third level, the nodes are assigned in the same block. One node contains the scheduling and the other node the dynamic optimization variables. Based on these results, we can argue that the estimated nSBM model indeed reveals the multi-scale nature of the problem. The partition in the third level shows that two sets of variables exist, scheduling and dynamic optimization ones. In the second level, the dynamic optimization variables are decomposed further into lines, and in the first level into slots and lines. Finally, the partition of the variables graph (Level 0) reveals the complex interaction among the different variables.

We note that application of community detection and centrality analysis [198] [198] to this problem decomposes the variable graph into five communities and a hierarchy among the communities is identified. The scheduling variables are assigned into one community, and the variables for the dynamic optimization problems for each slot and line are assigned into the other communities. In this partition, all the variables in the scheduling problem are assigned in the same block, i.e. community. Therefore, community detection and centrality analysis can not identify the hybrid multi-core community structure of the variable graph. This highlights the ability of nSBM to identify the 'true' structure of the problem.

Based on the structure of the $\omega_0$, $\omega_1$ matrices different decomposition based solution approaches can be proposed. The core-periphery structure in the first or second level can be used as the basis for the application of Generalized Benders Decomposition (GBD) [108], where the variables in the core (nodes with yellow color in the multigraph of the first or second level in Fig. 5.3) are assigned in the master problem and the variables in the periphery in the subproblem. The hybrid multi-core community structure of the observed graph can be used as the basis for the application of nested GBD. The original problem is first decomposed into a master and subproblem. The variables in the two blocks in the core are assigned in the master problem and the other variables are assigned in the subproblem. The master problem is decomposed further, based on the two-block partition of the scheduling variables.

### 5.3.3 Application of Generalized Benders Decomposition based on the structure of the level 1 variable graph

Based on the structure of the $\omega_1$ matrix, we apply Generalized Benders Decomposition (GBD). A detailed explanation of the application of GBD based on this partition of the variable graph can be found in [207]. The scheduling variables (yellow nodes in the first level multigraph Figure 5.3) are assigned in the master problem and the variables associated with the dynamic optimization problem are assigned in the subproblem. The edges that couple nodes that belong to different blocks correspond to constraints that couple variables that belong in the master and subproblem. These constraints are assigned in the subproblem and the master variables present are the complicating variables. For this case study these variables are $x_{kl}^{in}, x_{kl}^{end}, u_{kl}^{in}, u_{kl}^{end}, \theta_{kl}^{t}$. Given this decomposition, the variables are decomposed in three sets. The first set contains the scheduling variables that do not affect directly the subproblem ($s_1 = \{Y_{ikl}, z_{ijkl}, t_{ikl}^{prod}, H_l\}$), the second set contains the variables for the dynamic optimization problem from each slot and line ($s_2 = \{x_{fckl}, u_{fckl}, t_{fckl}^{d}, h_{kl}^{fe}\}$), and the last set contains the complicating variables ($s_3 = \{x_{kl}^{in}, x_{kl}^{end}, u_{kl}^{in}, u_{kl}^{end}, \theta_{kl}^{t}\}$). The subproblem is solved by fixing the shared variables and is equal to:

$$
\begin{aligned}
\text{minimize} \quad & a_u \sum_{l=1}^{N_l} \sum_{k=1}^{N_s} \sum_{f=1}^{N_{fe}} \sum_{c=1}^{N_{cp}} \frac{t_{fckl}^{d}}{N_{fe}} \Lambda_{c,N_{cp}} (u_{fckl} - u_{kl}^{end})^2 \\
\text{subject to} \quad & g_{dyn} \leq 0 \,(\text{Eq. } 5.36,5.38,\, 5.37) \\
& x_{kl}^{in} = \bar{x}_{kl}^{in} \quad \forall k \in \mathcal{I}_s, l \in \mathcal{I}_l \quad : \quad \lambda_{kl}^{1} \\
& x_{kl}^{end} = \bar{x}_{kl}^{end} \quad \forall k \in \mathcal{I}_s, l \in \mathcal{I}_l \quad : \quad \lambda_{kl}^{2} \\
& u_{kl}^{in} = \bar{u}_{kl}^{in} \quad \forall k \in \mathcal{I}_s, l \in \mathcal{I}_l \quad : \quad \lambda_{kl}^{3} \\
& u_{kl}^{end} = \bar{u}_{kl}^{end} \quad \forall k \in \mathcal{I}_s, l \in \mathcal{I}_l \quad : \quad \lambda_{kl}^{4} \\
& \theta_{kl}^{t} = \bar{\theta}_{kl}^{t} \quad \forall k \in \mathcal{I}_s, l \in \mathcal{I}_l \quad : \quad \lambda_{kl}^{5}
\end{aligned}
\tag{5.44}
$$

where the bar denotes that the corresponding variable is fixed and $\lambda$ is the Lagrangean multiplier. This problem corresponds to the dynamic optimization problems for every slot and line. Hence the different problems can be solved independently and their solution depends on the values of the complicating variables ($x_{kl}^{in}, x_{kl}^{end}, u_{kl}^{in}, u_{kl}^{end}, \theta_{kl}^{t}$).

Therefore, the value function of the dynamic optimization problem, $\eta$, can be approximated by the following Benders cuts:

$$\begin{aligned}\eta \geq\ & a_u \sum_{klfc} \frac{\bar{t}^{d,p}_{fckl}}{N_{fe}} \Lambda_{c,N_{cp}} (\bar{u}^{p}_{fckl} - \bar{\hat{u}}^{end,p}_{kl})^2 \\ & - \sum_{k,l} \Bigg( \lambda^{1,p}_{kl} (x^{in}_{kl} - \bar{\hat{x}}^{in,p}_{kl}) + \lambda^{2,p}_{kl} (x^{end}_{kl} - \bar{\hat{x}}^{end,p}_{kl}) + \lambda^{3,p}_{kl} (u^{in}_{kl} - \bar{\hat{u}}^{in,p}_{kl}) \\ & \qquad + \lambda^{4,p}_{kl} (u^{end}_{kl} - \bar{\hat{u}}^{end,p}_{kl}) + \lambda^{kl}_{5,p} (\theta^{t}_{kl} - \bar{\hat{\theta}}^{t,p}_{kl}) \Bigg) \ \forall p \in P,\end{aligned} \tag{5.45}$$

where $p$ is the iteration number. The detailed derivation of the Benders cut can be found in the Supporting Information. The master problem is

$$\begin{aligned}\text{minimize} \quad & \sum_{ikl} c^{p}_{il} \frac{W_{ikl}}{H_l} + \sum_{ijkl} c^{trans}_{ij} \theta^{t}_{kl} \frac{Z_{ijkl}}{H_l} + \sum_{ikl} c^{stor}_{il} W_{ikl} + \eta \\ \text{subject to} \quad & \text{Equations 5.29,5.30,5.31,5.32,5.40,5.45}\end{aligned} \tag{5.46}$$

In order to guarantee that the dynamic optimization problems are feasible we add the following constraints in the master problem:

$$\theta^{t}_{kl} \geq \sum_{i=1}^{N_P} \sum_{j=1}^{N_P} \theta^{min}_{ij} z_{ijkl} \ \forall k, l, \tag{5.47}$$

where $\theta^{min}_{ij}$ is the minimum transition time between product $i$ and $j$. We also add the following symmetry-breaking constraint which reduces the computational time without affecting the solution [304]:

$$\sum_{i=1}^{N_p} i Y_{i1l} \leq \sum_{i=1}^{N_p} i y_{ikl} \ \forall k \in \mathcal{I}_s, k > 1, l \in \mathcal{I}_l \tag{5.48}$$

Finally, we can add operational constraints in the subproblem. In this work, we add the following constraints which constrain the change of the manipulated variables:

$$\begin{aligned}
u_{mfckl} - u_{mf-1,ckl} &\leq U_m^{max}(t^d_{fckl} - t^d_{f-1,ckl}) \ \forall m, f \geq 1, c, k, l \\
u_{mfckl} - u_{mfc-1,kl} &\leq U_m^{max}(t^d_{fckl} - t^d_{fc-1,kl}) \ \forall m, f, c \geq 1, k, l \\
u_{mfckl} - u_{mf-1,ckl} &\geq U_m^{min}(t^d_{fckl} - t^d_{f-1,ckl}) \ \forall m, f \geq 1, c, k, l \\
u_{mfckl} - u_{mfc-1,kl} &\geq U_m^{min}(t^d_{fckl} - t^d_{fc-1,kl}) \ \forall m, f, c \geq 1, k, l,
\end{aligned} \tag{5.49}$$

where $U_m^{max}, U_m^{min}$ is the maximum and minimum rate of change of manipulated variable $m$. The master problem is a Mixed Integer Nonlinear Problem (MINLP) which is solved with BARON [273] and the subproblems are Nonlinear Problems (NLPs) which are solved with IPOPT [285] in Pyomo [129]. The optimality gap tolerance is set equal to 0.1%. Finally, we note that this approach can not guarantee global optimality since the subproblem is nonconvex [256].

### 5.3.4 Application of nested Generalized Benders Decomposition based on the structure of the variable graph

Based on the structure of the $\omega_0$ matrix, nested Generalized Benders Decomposition can be applied. The original problem is first decomposed into a master and subproblem as described in the previous section. Then, the structure of the core is used to solve the master problem using GBD. The master problem is decomposed into two subproblems that we will define as MM and MS, where MM stands for Master_Master and MS for Master_Subproblem. In this approach, the MM problem contains variables $s_4 = \{Y_{ikl}, z_{ijkl}, x^{end}_{kl}, x^{in}_{kl}, u^{end}_{kl}, u^{in}_{kl}\}$ (complicating variables) and the other scheduling variables and associated constrains are assigned in the MS problem. Therefore, problem

MS is solved for fixed values of the complicating variables $s_4$ and is equal to

$$\begin{aligned}
\text{minimize} \quad & \sum_{ikl} c_{il}^{p} \frac{W_{ikl}}{H_l} + \sum_{ijkl} c_{ij}^{trans} \theta_{kl}^{t} \frac{z_{ijkl}}{H_l} + \sum_{ikl} c_{il}^{stor} W_{ikl} + \eta \\
\text{subject to} \quad & \text{Equation 5.31, 5.32, 5.47,5.45} \\
& Y_{ikl} = \bar{Y}_{ikl} \; \forall \; i \in \mathcal{I}_p, k \in \mathcal{I}_s, l \in \mathcal{I}_l \quad : \quad \mu_{ikl}^{1} \\
& z_{ijkl} = \bar{z}_{ijkl} \; \forall \; i \in \mathcal{I}_p, j \in \mathcal{I}_p, k \in \mathcal{I}_s, l \in \mathcal{I}_l \quad : \quad \mu_{ijkl}^{2} \\
& x_{kl}^{in} = \bar{x}_{kl}^{in} \; \forall \; k \in \mathcal{I}_s, l \in \mathcal{I}_l \quad : \quad \mu_{kl}^{3} \\
& x_{kl}^{end} = \bar{x}_{kl}^{end} \; \forall \; k \in \mathcal{I}_s, l \in \mathcal{I}_l \quad : \quad \mu_{kl}^{4} \\
& u_{kl}^{in} = \bar{u}_{kl}^{in} \; \forall \; k \in \mathcal{I}_s, l \in \mathcal{I}_l \quad : \quad \mu_{kl}^{5} \\
& u_{kl}^{end} = \bar{u}_{kl}^{end} \; \forall \; k \in \mathcal{I}_s, l \in \mathcal{I}_l \quad : \quad \mu_{kl}^{6},
\end{aligned} \tag{5.50}$$

where $\mu$ are the Lagrangean multipliers. The solution of this problem depends on the values of the complicating variables and the value function of this problem, $\eta_2$, can be approximated by the following Benders cut:

$$\begin{aligned}
\eta_2 \geq & \sum_{ikl} c_{il}^{p} \frac{\bar{W}_{ikl}^{q}}{\bar{H}_l^{q}} + \sum_{ijkl} c_{ij}^{trans} \bar{\theta}_{kl}^{t,q} \frac{\bar{z}_{ijkl}^{q}}{\bar{H}_l^{q}} + \sum_{ikl} c_{il}^{stor} \bar{W}_{ikl}^{q} + \bar{\eta}^{q} \\
& - \sum_{ikl} \mu_{ikl}^{1,q} (y_{ikl} - \bar{Y}_{ikl}^{q}) - \sum_{ijkl} \mu_{ijkl}^{2,q} (z_{ijkl} - \bar{z} q_{ijkl}) \\
& - \sum_{kl} \Bigg( \mu_{kl}^{3,q} (x_{kl}^{in} - \bar{x}_{ikl}^{in,q}) - \mu_{kl}^{4,q} (x_{kl}^{end} - \bar{x}_{ikl}^{end,q}) \\
& - \mu_{kl}^{5,q} (u_{kl}^{in} - \bar{u}_{ikl}^{in,q}) - \mu_{kl}^{6,q} (u_{kl}^{end} - \bar{u}_{ikl}^{end,q}) \Bigg) \; \forall q \in \mathcal{Q},
\end{aligned} \tag{5.51}$$

where $q$ is the inner iteration number. The exact derivation of this equation can be found in the Supporting Information. The MM problem is equal to:

$$\begin{aligned}
\text{minimize} \quad & \eta_2 \\
\text{subject to} \quad & \text{Equations 5.29, 5.30, 5.40, 5.48, 5.51}
\end{aligned} \tag{5.52}$$

Problem MM is a Mixed Integer Linear Problem (MILP) solved with Gurobi [124] and problem MS is a NLP solved with IPOPT [285]. The overall nested GBD algorithm is presented in Algorithm 1. In this nested GBD approach the number of constraints in problem MS at every iteration increases due to the addition of the Benders cuts that

**Data:** Optimization problem
**Result:** Upper, lower bound and variable values
Set $UB^{in} = \infty, LB^{in} = -\infty, UB^{out} = \infty, LB^{out} = -\infty$;
Set tolerance and optimality gap (tol);
**while** $(UB^{out} - LB^{out})/LB^{out} \geq 0.01\ tol$ **do**
    **while** $(UB^{in} - LB^{in})/LB^{in} \geq 0.01\ tol$ **do**
        Solve problem MM (Eq. 5.52) and obtain $LB^{in}$;
        Solve problem MS (Eq. 5.50) and obtain $UB^{in}$;
        Add benders cut in the MM problem (Eq. 5.51);
    **end**
    Set $LB^{out} = UB^{in}$;
    Solve dynamic optimization problems for each slot and line (Eq. 5.44), obtain $UB^{out}$;
    Add Benders cut to problem MS (Eq. 5.45);
    Remove Benders cuts from problem MM (Eq. 5.51);
**end**

**Algorithm 5.1:** Nested Generalized Benders Decomposition

approximate the value function of the dynamic optimization problem. Also, once the master problem is solved, the Benders cuts (Eq. 5.51) are removed, since they may not be valid for the problem in the next iteration. The optimality gap tolerance for both the inner and outer loops is set equal to 0.1%. Similarly to the case of single GBD, the nested GBD can not guarantee global optimality [256].

### 5.3.5 Results

First we solve the problem using GBD and the results are presented in Figure 5.4. The algorithm converges after 209 CPU seconds (19 iterations) and the value of the objective function is 43.36 $10^3\$/hr$. 99% of the CPU time is used for the solution of the master problem. The production results are presented in Table 5.2 and the concentration and flowrate profiles are presented in Figures 5.5,5.6. The cycle time in the first line is 26.89 hours and in the second line is 44.47 hours. The transition times in the first line are smaller compared to the second line. Furthermore, in both lines the majority of the production time is dedicated for the production of one product (product 1 in line 1 and product 2 in line 2). This is due to the lower production rate of these products.

The nested algorithm converges after 54 CPU seconds. The value of the objective function is 43.36 $10^3\$/hr$ and the evolution of the upper and lower bounds is presented

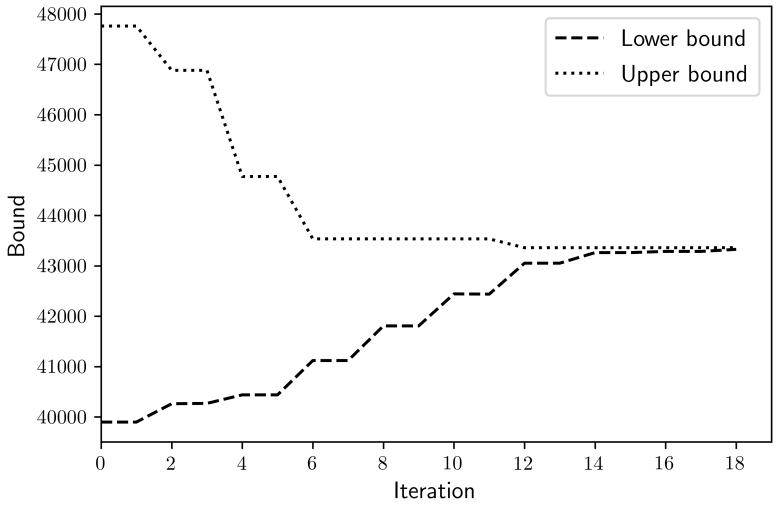


Figure 5.4: Evolution of the upper and lower bound for the single GBD algorithm.

Table 5.2: Production results

| Line 1, Cycle time 26.89 hr | | | | |
|---|---|---|---|---|
| Slot | Product | Production amount $(kg)$ | Production time $(hr)$ | Transition time $(hr)$ |
| 1 | 1 | 912 | 24.3 | 1.50 |
| 2 | 3 | 457 | 2.58 | 0.93 |
| Line 2, Cycle time 44.47 hr | | | | |
| Slot | Product | Production amount $(kg)$ | Production time $(hr)$ | Transition time $(hr)$ |
| 1 | 4 | 711 | 3.57 | 2.15 |
| 2 | 2 | 2341 | 40.90 | 4.87 |

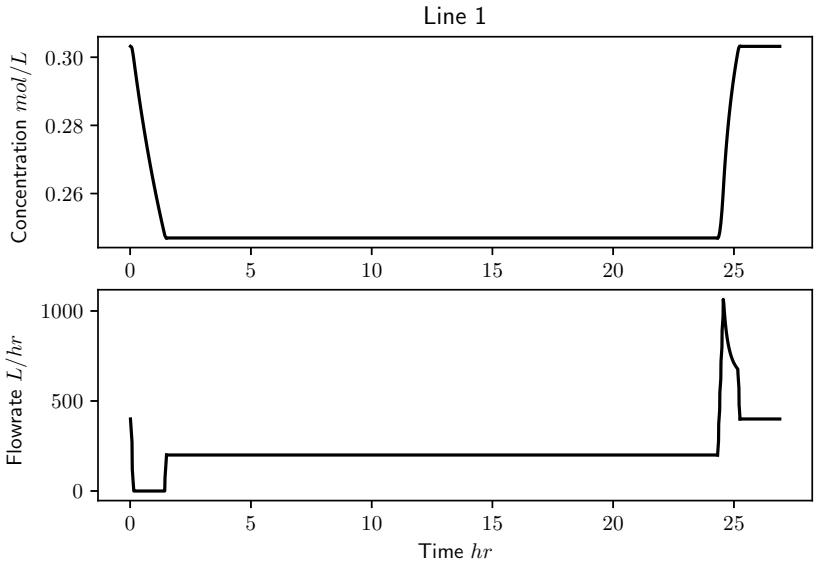


Figure 5.5: Concentration and inlet flowrate profile in line 1

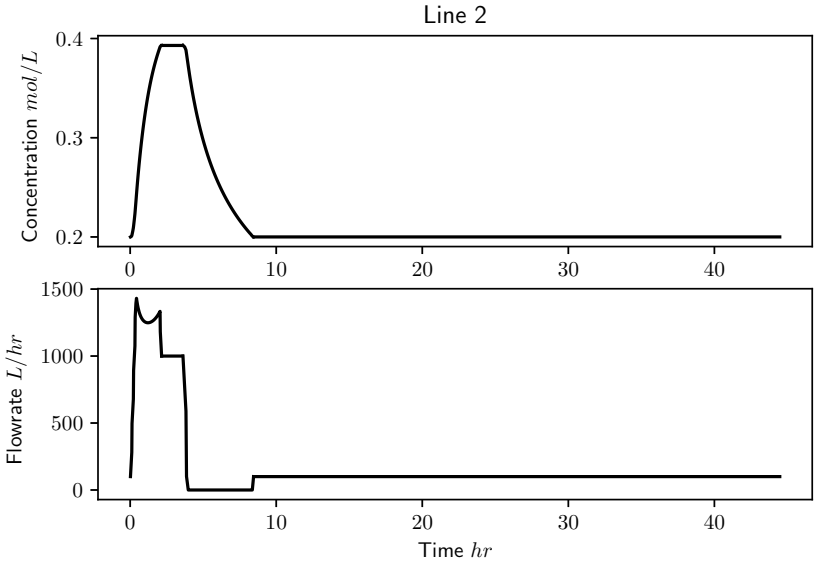


Figure 5.6: Concentration and inlet flowrate profile in line 2

in Figure 5.7. The production results are the same as the ones obtained with the single

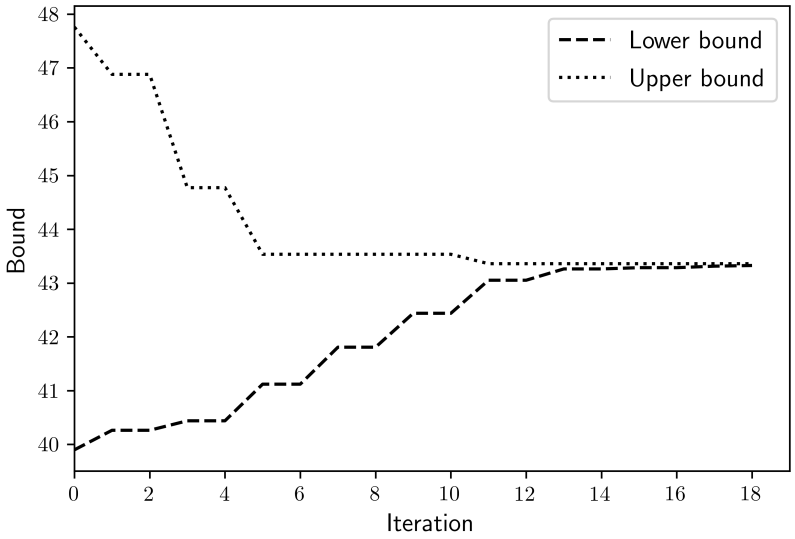


Figure 5.7: Evolution of the upper and lower bound for the nested GBD.

GBD algorithm (Table 5.2). Finally, we compare the evolution of the upper and lower bounds with CPU time for the GBD and nested GBD approach. From Figure 5.8, we see that the exploitation of the hybrid multicore-community structure of the problem reduces the computational time by 74%. Finally, we note that solving the monolithic problem with BARON, after 3000 CPU seconds the gap is 33 % and the value of the objective function is 80.7 $10^3\$/hr$. These results highlight the importance of detecting and exploiting the true underlying structure of an optimization problem.

## 5.4 Integration of Planning, Scheduling and Dynamic Optimization

### 5.4.1 Problem formulation

We now consider the problem of integration of planning, scheduling and dynamic optimization. The detailed explanation of the model can be found in [125]. We assume that the number of products is $N_p$ ($i = \{1, .., N_p\}$), the number of planning periods is $N_{pr}$ ($p = \{1, .., N_{pr}\}$), and the number of slots is $N_s$ ($k = \{1, .., N_s\}$). First we define the binary variable $W_{ikp}$ which is equal to 1 if product $i$ is produced at slot $k$ in period

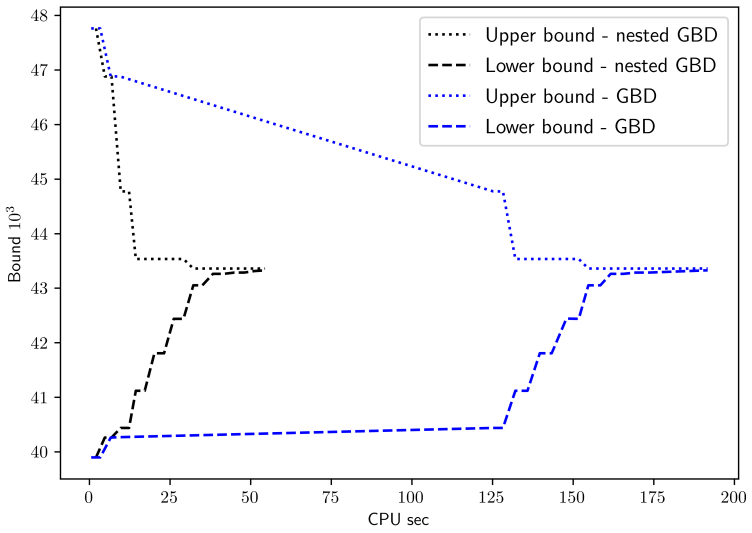


Figure 5.8: Evolution of the upper and lower bound for the single and nested decomposition algorithm.

$p$ and zero otherwise. We also define the binary variable $Z_{ijkp}$ which is equal to 1 if a product $i$ is followed by product $j$ in slot $k$ in period $p$ and the binary variable $Zp_{ijp}$ which is equal to one if transition occurs between product $i$ and $j$ between time periods. At each time slot only one product can be produced, which is enforced with the following constraints:

$$\sum_i W_{ikp} = 1 \ \forall k, p. \tag{5.53}$$

The transitions between products are modeled though the following equations:

$$\begin{aligned} Z_{ijkp} &\geq W_{ikp} + W_{j,k+1,p} - 1 \ \forall i, j \in N_p, i \neq j, k \neq N_s, p \\ Zp_{ijkp} &\geq W_{iN_sp} + W_{j,1,p+1} - 1 \ \forall i, j \in N_p, i \neq j, p \neq N_{per} \end{aligned} \tag{5.54}$$

The production time of product $i$ in slot $k$ in period $p$ is $\theta_{ikp}$ and the production time of product $i$ in period $p$ is $\hat{\theta}_{ip}$. The starting, ending, and transition time in slot $k$ in

period $p$ is $T^s_{kp}, T^e_{k,p}$, and $\theta^t_{kp}$, respectively. The timing constraints are the following:

$$\begin{aligned}
&T^s_{1,1} = 0 \\
&T^e_{k,p} = T^s_{k,p} + \sum_i \theta_{ikp} + \theta^t_{kp} \quad \forall k, p \\
&T^s_{k+1,p} = T^e_{k,p} \quad \forall k \neq N_s, p \\
&T^s_{1,p+1} = T^e_{N_s,p} \quad \forall k, p \neq N_{per} \\
&\hat{\theta}_{ip} = \sum_k \theta_{ikp} \quad \forall i, p \\
&T^e_{k,p} \leq H_p,
\end{aligned} \tag{5.55}$$

where $H_p$ is the duration of period $p$. The production rate of product $i$ is $r_i$, the amount of product $i$ produced in slot $k$ at period $p$ is $\hat{q}_{ikp}$, and the amount of product $i$ produced in period $p$ is $q_{ip}$. The production and inventory constraints are:

$$\begin{aligned}
&\hat{q}_{ikp} = r_i \hat{\theta}_{ikp} \quad \forall i, k, p \\
&q_{ip} = \sum_k \hat{q}_{ikp} \quad \forall i, p \\
&I_{ip} = I_{ip-1} + q_{ip} - S_{ip} \quad \forall i, p \\
&A_{ip} = H_p(I_{ip-1} - S_{ip-1}) + q_{ip} H_p \quad \forall i, p,
\end{aligned} \tag{5.56}$$

where $I_{ip}$ is the inventory of product $i$ in period $p$, $A_{ip}$ is the linear overestimation of the integral of inventory, and $S_{ip}$ is the amount of product $i$ sold in period $p$. The following symmetry breaking constraints are also included:

$$\begin{aligned}
&Y_{ip} \geq W_{ikp} \quad \forall i, k, p \\
&Y_{ip} \leq N_{ip} \quad \forall i, p \\
&N_{ip} \geq N - \left( \sum_i Y_{ip} - 1 \right) - M(1 - W_{i1p}) \quad \forall i, p \\
&N_{ip} \leq N - \left( \sum_i Y_{ip} - 1 \right) + M(1 - W_{i1p}) \quad \forall i, p \\
&N_{ip} = \sum_k W_{ikp} \quad \forall i, p
\end{aligned} \tag{5.57}$$

The dynamic behavior of the system is modeled as in the previous case study, the differential equations are discretized using collocation on finite elements, and the constraints

are:

$$
\begin{aligned}
x^n_{fckp} &= x0^n_{fkp} + h^{fe}_{kp} \sum_{m=1}^{N_{cp}} \Omega_{mc} \dot{x}^n_{fmkp} \quad \forall n, f, c, k, p \\
x0^n_{fkp} &= x0^n_{f-1kp} + h^{fe}_{kp} \sum_{m=1}^{N_{cp}} \Omega_{mc} \dot{x}^n_{f-1,mkp} \quad \forall n, f \geq 2, c, k, p \\
\dot{x}^n_{fckp} &= f^n(x^n_{fckp}, u^m_{fckp}) \quad \forall n, f, c, k, p \\
t^d_{fckp} &= h^{fe}_{kp}(f - 1 + \gamma_c) \quad \forall f, c, k, p.
\end{aligned}
\tag{5.58}
$$

$$
h^{fe}_{kp} = \frac{\theta^t_{kp}}{N_{fe}} \quad \forall k, p \tag{5.59}
$$

The dynamic model is integrated with the planning/scheduling problem through the following constraints:

$$
\begin{aligned}
x^{in}_{n,k,p} &= \sum_i x^{ss}_i W_{i,k,p} \quad \forall n, k, p \\
x^{end}_{n,k,p} &= \sum_i x^{ss}_i W_{i,k+1,p} \quad \forall n, k, p \\
u^{in}_{n,k,p} &= \sum_i u^{ss}_i W_{i,k,p} \quad \forall n, k, p \\
u^{end}_{n,k,p} &= \sum_i u^{ss}_i W_{i,k+1,p} \quad \forall n, k, p
\end{aligned}
\tag{5.60}
$$

$$
\begin{aligned}
x0^n_{1kp} &= x^{in}_{n,k,p} \quad \forall n, k, p \\
x^n_{N_{fe}N_{cp}kp} &= x^{end}_{n,k,p} \quad \forall n, k, p \\
u^m_{11kp} &= u^{in}_{m,k,p} \quad \forall m, k, p \\
u^m_{N_{fe}N_{cp}kp} &= u^{end}_{m,k,p} \quad \forall m, k, p
\end{aligned}
\tag{5.61}
$$

The objective is to maximize the profit which is equal to

$$
\begin{aligned}
f = & \sum_{i=1}^{N_p} \sum_{p=1}^{N_{per}} P_{ip} S_{ip} - \sum_{i=1}^{N_p} \sum_{p=1}^{N_{per}} C_{ip}^{oper} q_{ip} \\
& - C^{inv} \sum_{i=1}^{N_p} \sum_{p=1}^{N_{per}} A_{ip} - \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} \sum_{k=1}^{N_s} \sum_{p=1}^{N_{per}} C_{ij}^{trans} Z_{ijkp} \\
& - \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} \sum_{p=1}^{N_{per}} C_{ij}^{trans} Z p_{ijp} \\
& - \alpha_u \sum_{p=1}^{N_{per}} \sum_{k=1}^{N_p} \sum_{f=1}^{N_{fe}} \sum_{c=1}^{N_{cp}} N_{fe}^{-1} t_{fckp}^{d} \Lambda_{c,N_{cp}} (u_{fckp} - u_{kp}^{end})^2,
\end{aligned}
\tag{5.62}
$$

where $P_{ip}$ is the price of product $i$ in period $p$, $C_{ip}^{oper}$ is the operating cost of product $i$ in period $p$, $C^{inv}$ is the inventory cost, $C_{ij}^{trans}$ is the transition cost from product $i$ to $j$ and $a_u$ is a weight coefficient. The goal of the optimization problem is to maximize Eq. 5.62 subject to constraints 5.53, 5.54, 5.55, 5.56, 5.57, 5.58, 5.59,S.6, S.7.

### 5.4.2 Application of nested Stochastic Blockmodeling

The structure of the variable graph will be analyzed using degree-corrected nSBM and Bayesian inference in `graph-tool` [235]. We will assume that four products (4 slots) must be produced in three planning periods and the system is the same as in the previous case study (one isothermal continuous stirred reactor). Application of nSBM identifies four levels (Figure 5.9). The observed network is partitioned into 14 blocks (Figure 5.10).

The multigraph in the first level is partitioned into seven blocks and in the second level into four blocks. The number of edges between the blocks in the observed and the different multigraphs is given by $\omega_0, \omega_1, \omega_2$ respectively. This partition of the variable graph into four levels provides information about the multi-scale nature of the problem

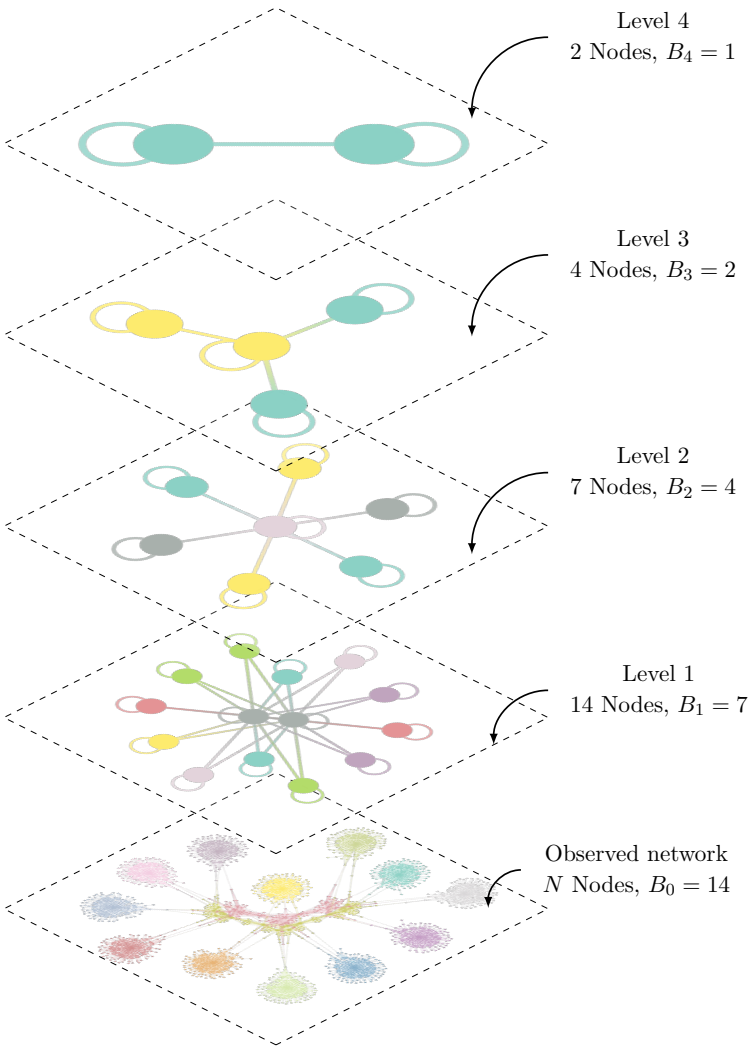


Figure 5.9: Inferred nSBM model of the variable graph for the integrated planning, scheduling and dynamic optimization problem. The hollow cycles indicate self-edges.

and the different structures that are present at different hierarchical levels.

$$\omega_0 = \begin{bmatrix} 842 & 48 & 3 & 3 & \dots & 3 \\ 48 & 742 & 1 & 1 & \dots & 1 \\ 3 & 1 & 757 & 0 & \dots & 0 \\ 3 & 1 & 0 & 757 & \dots & 0 \\ \vdots & \vdots & \vdots & & \ddots & \\ 3 & 1 & 0 & \dots & \dots & 757 \end{bmatrix} \in \mathbb{R}^{14\times 14}$$

$$\omega_1 = \begin{bmatrix} 1672 & 8 & 8 & 8 & 8 & 8 & 8 \\ 8 & 1514 & 0 & 0 & 0 & 0 & 0 \\ 8 & 0 & 1514 & 0 & 0 & 0 & 0 \\ 8 & 0 & 0 & 1514 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 1514 & 0 & 0 \\ 8 & 0 & 0 & 0 & 0 & 1514 & 0 \\ 8 & 0 & 0 & 0 & 0 & 0 & 1514 \end{bmatrix}$$

$$\omega_2 = \begin{bmatrix} 1672 & 16 & 16 & 16 \\ 16 & 3028 & 0 & 0 \\ 16 & 0 & 3028 & 0 \\ 16 & 0 & 0 & 3028 \end{bmatrix}$$

The original graph is decomposed into 14 blocks and the planning and scheduling variables are assigned in the two middle blocks. Variables $W_{ikl}$, $Z_{ijkl}$, $Zp_{ijp}$, $Y_{ip}$, $N_{ip}$, $x^{in}_{kp}$, $x^{end}_{kp}$, $u^{in}_{kp}$, $u^{end}_{kp}$ are assigned in the green block and the other planning/scheduling variables in the other blocks. The variables associated with the dynamic behavior of the problem for each slot and period $(x_{nfckp}, u_{mfckp}, h^{fe}_{kp}, t_{fckp})$ are assigned in the blocks in the periphery. From the structure of the $\omega_0$ matrix, we can determine that the graph has a hybrid multi-core community structure. The variables for the dynamic optimization problems for each slot and period are assigned into different blocks. The variables in these blocks are densely coupled denoting a community structure. However, these blocks are weakly coupled with the planning/scheduling variables, which form a multi-core structure, since these variables are assigned into two blocks and are connected with all the dynamic optimization variables.

In the first level, the planning and scheduling variables are assigned into the same block and the dynamic optimization variables are assigned into different blocks leading

Figure 5.10: Partition of the variable graph for the integrated planning, scheduling and dynamic optimization problem

Figure 5.11: Partition of the first level multigraph for the integrated planning, scheduling and dynamic optimization problem

to a core-periphery structure, highlighted by the $L$ shape of the $\omega_1$ matrix. Similarly, the multigraph in the second level has a core-periphery structure. In this multigraph, the planning/scheduling variables are in the middle node and the variables for the dynamic optimization problems are in the periphery. This graph is a coarser partition of the level 1 graph, and although each node in the periphery corresponds to the variables of two dynamic optimization problems, these variables are decoupled since in the first level the nodes in the periphery are not coupled directly, i.e. there does not exist an edge between the nodes in the periphery.

### 5.4.3 Application of Generalized Benders Decomposition based on the core-periphery structure of the first level multigraph

The core-periphery structure of the multigraph in level 1 can be used as the basis for the application of GBD. The variables in the core (grey nodes in Figure 5.11) correspond to the planning/scheduling variables and are assigned in the master problem. The variables for the dynamic optimization problems for each slot and period, and the associated constraints, are assigned in the subproblem. The complicating variables are $x^{in}_{kp}$, $x^{end}_{kp}$, $u^{in}_{kp}$, $u^{end}_{kp}$, $\theta^{t}_{kp}$. The subproblem is solved for fixed values of the complicating variables and is

$$
\begin{aligned}
\text{minimize} \quad & \alpha_u \sum_{p=1}^{N_{per}} \sum_{k=1}^{N_s} \sum_{f=1}^{N_{fe}} \sum_{c=1}^{N_{cp}} N_{fe}^{-1} t^{d}_{fckp} \Omega_{c,N_{cp}} (u_{fckp} - u^{end}_{kp})^2 \\
\text{subject to} \quad & \text{Equations 5.58,5.59, S.7} \\
& x^{in}_{nkp} = \bar{x}^{in}_{nkp} \quad \forall n,k,p \qquad : \quad \gamma^{1}_{kl} \\
& x^{end}_{nkp} = \bar{x}^{end}_{nkp} \quad \forall n,k,p \qquad : \quad \gamma^{2}_{kl} \\
& u^{in}_{mkp} = \bar{u}^{in}_{mkp} \quad \forall m,k,p \qquad : \quad \gamma^{3}_{kl} \\
& u^{end}_{mkp} = \bar{u}^{end}_{mkp} \quad \forall m,k,p \qquad : \quad \gamma^{4}_{kl} \\
& \theta^{t}_{kp} = \bar{\theta}^{t}_{kp} \quad \forall k,p \qquad : \quad \gamma^{5}_{kl},
\end{aligned}
\tag{5.63}
$$

where $\gamma$ is the Lagrangean multiplier of each constraint. This problem can be solved independently for every slot and period. The value function of this problem is approximated by the following Benders cut:

$$
\begin{aligned}
\eta \geq & \alpha_u \sum_{p=1}^{N_{per}} \sum_{k=1}^{N_p} \sum_{f=1}^{N_{fe}} \sum_{c=1}^{N_{cp}} N_{fe}^{-1} \bar{t}^{d,v}_{fckp} \Omega_{c,N_{cp}} (\bar{u}^{v}_{fckp} - \bar{u}^{end,,v}_{kp})^2 \\
& - \sum_{p=1}^{N_{per}} \sum_{k=1}^{N_s} \Big( \gamma^{1,v}_{k,p} (x^{in}_{kp} - \bar{x}^{in,v}_{kp}) + \gamma^{2,v}_{k,p} (x^{end}_{kp} - \bar{x}^{end,v}_{kp}) \\
& + \gamma^{3,v}_{k,p} (u^{in}_{kp} - \bar{u}^{in,v}_{kp}) + \gamma^{4,v}_{k,p} (u^{end}_{kp} - \bar{u}^{end,v}_{kp}) \\
& + \gamma^{5,v}_{k,p} (\theta^{t}_{kp} - \bar{\theta}^{t,v}_{kp}) \Big) \quad \forall v \in \mathcal{V},
\end{aligned}
\tag{5.64}
$$

where the superscript $v$ is the iteration number. The master problem is

$$\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{N_p}\sum_{p=1}^{N_{per}}\left(P_{ip}S_{ip} - C_{ip}^{oper}q_{ip} - C^{inv}A_{ip}\right) \\
& - \sum_{i=1}^{N_p}\sum_{j=1}^{N_p}\sum_{k=1}^{N_s}\sum_{p=1}^{N_{per}} C_{ij}^{trans}Z_{ijkp} \\
& - \sum_{i=1}^{N_p}\sum_{j=1}^{N_p}\sum_{p=1}^{N_{per}} C_{ij}^{trans}Zp_{ijp} - \eta \\
\text{subject to} \quad & \text{Eq. 5.53, 5.54, 5.55, 5.56, 5.57, S.6, S.9}
\end{aligned} \tag{5.65}$$

In order to guarantee that the dynamic optimization problem is feasible we add the following constraint in the master problem:

$$\begin{aligned}
\theta_{kp}^{t} &\geq \sum_{i=1}^{N_p}\sum_{j=1}^{N_p} Z_{ijkp}\theta_{ij}^{min} \;\; \forall k, p, k \neq N_s \\
\theta_{N_s,p}^{t} &\geq \sum_{i=1}^{N_p}\sum_{j=1}^{N_p} Zp_{ijp}\theta_{ij}^{min} \;\; \forall p, p \neq N_{per}
\end{aligned} \tag{5.66}$$

Finally, we add operational constraints similar to Eq. 5.49. The exact derivation of the master, subproblem and Benders cut can be found in the Supplementary material. The master problem is a MILP solved with Gurobi [124] and the subproblem is a NLP solved with IPOPT [285] in Pyomo [129]. The problem is solved to 0.1% optimality gap.

### 5.4.4 Results

We solve the integrated problem, using the GBD approach proposed in the previous section and the economic parameters of the optimization problem are presented in Tables 7.1, 7.2, 7.3. The algorithm converges after 294 CPU seconds, the evolution of the upper and lower bounds is presented in Figure 5.12, and the value of the objective function is 1.247 $10^7$\$. The solution of the master problem accounts for 76% of the total CPU time. Monolithic solution with BARON [273] can not find a feasible solution after 500 CPU seconds.

The production results are presented in Table 5.6 and the profiles of the concentration and inlet flowrate for each slot and period are presented in Fig. 5.13, 5.14, 5.15.

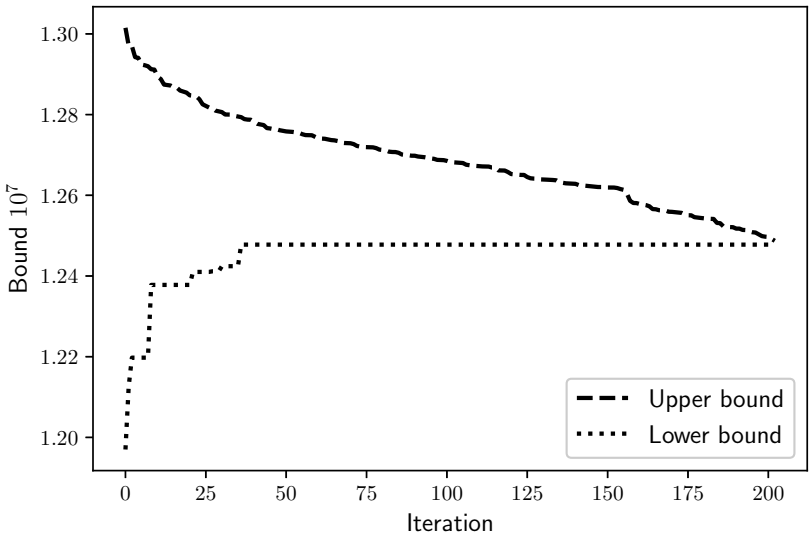


Figure 5.12: Convergence of Generalized Benders Decomposition based on the core-periphery structure of the first level multigraph for the integrated planning, scheduling and dynamic optimization problem.

No transition occurs between the different time periods, leading to a reduction in the transition cost. The production of the products satisfies the demand. Furthermore, we note the production profiles of products $C, D$. Product $C$ is overproduced in the first period, since its operating cost is lower compared to the other two periods. Product $D$ is overproduced in the last period where its operating cost is lower and price higher compared to the other periods.

From these results, we can argue that application of GBD based on the core-periphery structure of level 1 of the variable graph enables an efficient solution of the integrated problem.

**Remark 5.1.** We note that in this case study, similar to the one in the integration of scheduling and dynamic optimization, we can apply a nested GBD approach, based on the hybrid multi-core community structure of the variable graph. We did not apply nested GBD since the master problem is a MILP which can be solved efficiently with Gurobi. For problems with larger number of products/periods, application of nested GBD might be necessary in order to further reduce the computational time.

Table 5.3: Operating conditions and product price for the integrated planning, scheduling and dynamic optimization problem

| Product | $c^{ss}(mol/L)$ | $Q^{ss}(L/h)$ | Production rate |
|---|---|---|---|
| A | 0.24 | 200 | 150 |
| B | 0.2 | 150 | 80 |
| C | 0.30 | 130 | 278 |
| D | 0.32 | 1000 | 607 |

Table 5.4: Operating and transition cost for the integrated planning, scheduling and dynamic optimization problem, $C^{inv} = 0.026, a_u = 1$

| | $C^{oper}$ | | | $C^{trans}$ | | | |
|---|---|---|---|---|---|---|---|
| Product | $p = 1$ | $p = 2$ | $p = 3$ | A | B | C | D |
| A | 13 | 13 | 13 | 0 | 100 | 600 | 120 |
| B | 22 | 12 | 12 | 150 | 0 | 50 | 80 |
| C | 35 | 45 | 45 | 200 | 150 | 0 | 100 |
| D | 29 | 19 | 19 | 90 | 100 | 120 | 0 |

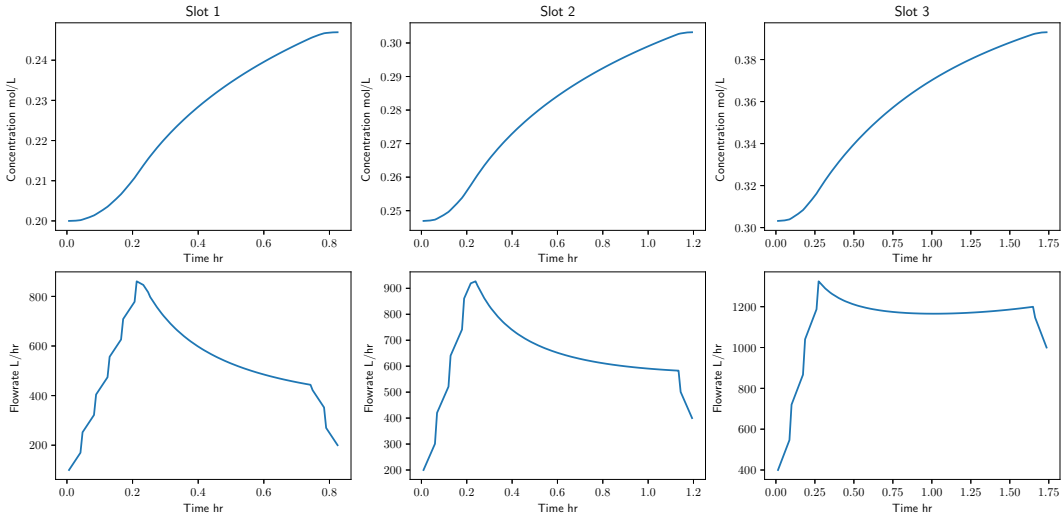


Figure 5.13: Concentration and inlet flowrate profile for the first period

Table 5.5: Product demand for the integrated planning, scheduling and dynamic optimization problem

| | Demand ($mol/week$) | | | Price ($/mol$) | | |
|---|---|---|---|---|---|---|
| Prod. | $p=1$ | $p=2$ | $p=3$ | $p=1$ | $p=2$ | $p=3$ |
| A | 6000 | 8000 | 7000 | 200 | 220 | 200 |
| B | 5000 | 3600 | 6000 | 160 | 140 | 150 |
| C | 7000 | 9000 | 7000 | 130 | 150 | 140 |
| D | 4000 | 11000 | 11000 | 110 | 110 | 120 |

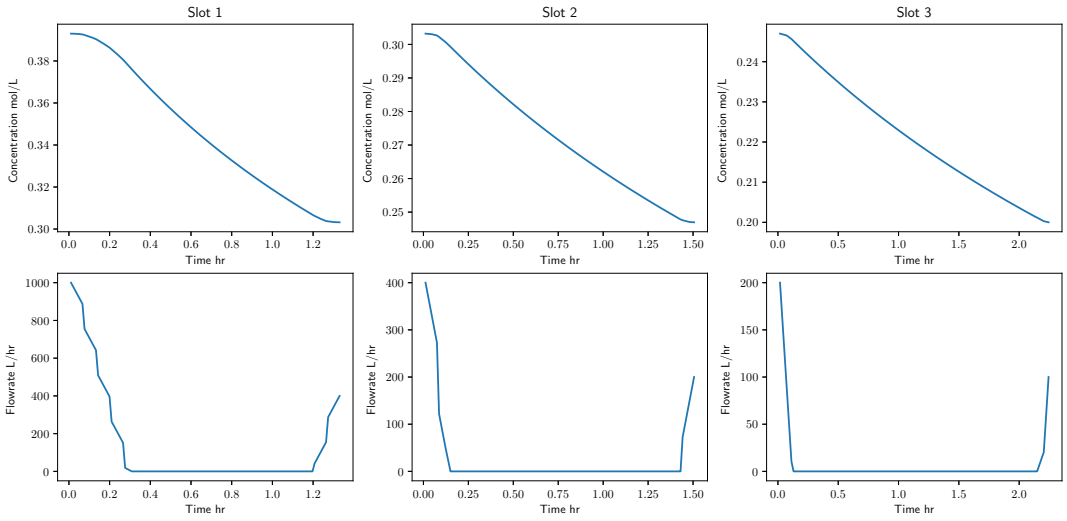


Figure 5.14: Concentration and inlet flowrate profile for the second period

## 5.5 Conclusions and Further Remarks

The integration of process operations leads to large scale optimization problems whose monolithic solution is challenging. In this work we proposed nested Stochastic Block-modeling and Bayesian inference as a framework to detect the underlying hierarchical block structure and the hierarchy itself of such optimization problems. We applied this framework to representative problems on integration of scheduling and dynamic optimization, and planning, scheduling and dynamic optimization. The inference and solution results highlight the inherent ability of the proposed approach to detect the multi-scale nature of these problems and the complex block structures that are present

Table 5.6: Production results for the integrated planning, scheduling and dynamic optimization problem.

| Period 1 | | | | |
|---|---|---|---|---|
| Slot | Product | Production amount $(mol)$ | Production time $(hr)$ | Transition time $(hr)$ |
| 1 | B | 5000 | 62.50 | 0.82 |
| 2 | A | 6000 | 39.83 | 1.19 |
| 3 | C | 15418 | 55.31 | 1.73 |
| 4 | D | 4000 | 6.59 | 0 |
| Period 2 | | | | |
| Slot | Product | Production amount $(mol)$ | Production time $(hr)$ | Transition time $(hr)$ |
| 1 | D | 11000 | 18.12 | 1.33 |
| 2 | C | 581.85 | 2.08 | 1.50 |
| 3 | A | 8000 | 53.11 | 2.24 |
| 4 | B | 5000 | 62.5. | 0 |
| Period 3 | | | | |
| Slot | Product | Production amount $(mol)$ | Production time $(hr)$ | Transition time $(hr)$ |
| 1 | B | 4600 | 57.5. | 1.08 |
| 2 | A | 8000 | 53.11 | 1.16 |
| 3 | C | 9000 | 32.29 | 2.01 |
| 4 | D | 29089 | 47.92 | 0 |

in the different hierarchical levels. Furthermore, we showed that the exploitation of the structure at different hierarchical levels enables an efficient solution of such problems using decomposition based solution algorithms. Finally, the following general remarks can be made.

**Remark 5.2.** The decomposition that is obtained with this approach is supported by statistical evidence, and when the decomposition with the minimum description length is selected, it is optimal from a network structure perspective. Despite the improvement in computational time noted compared to the monolithic solution of the problem, it is not guaranteed that the obtained decomposition is optimal with respect to the computational time or convergence rate. This problem will be addressed in future work.

**Remark 5.3.** For the integrated scheduling and dynamic optimization problem, we applied weighted nested SBM using a discrete geometric model for the edge weights. In

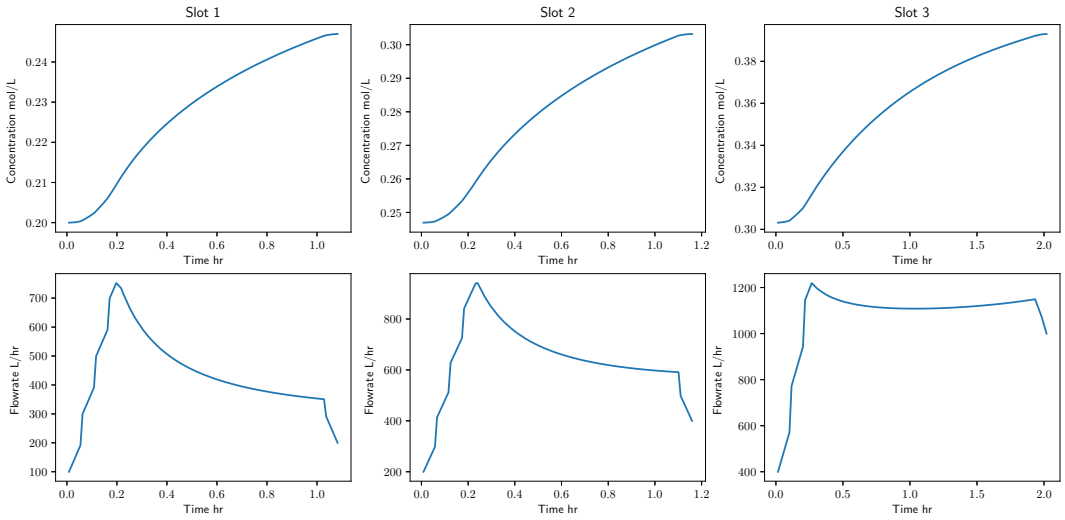


Figure 5.15: Concentration and inlet flowrate profile for the third period

general, different models can be selected, such as discrete binomial and Poisson, and real exponential and normal [238]. Since the selection of the model for the weights is an assumption, one can try different models and test the suitability of the resulting decompositions for the solution of the problem.

# Chapter 6
# Learning to initialize Generalized Benders Decomposition via active learning

## 6.1 Introduction

Decomposition-based optimization algorithms have been used widely to solve complex and large-scale optimization problems in a broad range of applications in chemical engineering, such as production scheduling and planning, supply chain management, mixed integer optimal control, and real-time operation. These algorithms exploit the underlying structure of a problem and decompose it into a number of easier-to-solve subproblems. Typical examples include Benders [27] and Generalized Benders Decomposition [108], Lagrangean decomposition [119], Alternating Direction Method of Multipliers (ADMM) [48], and cross decomposition [281].

Despite the wide success of these algorithms, their efficiency over monolithic methods is not known a-priory and their implementation, especially in an online setting, is challenging due to the many steps involved in their implementation and the underlying computational complexity of every step. Automatically determining when to use and how to implement a decomposition-based solution algorithm can simplify their implementation and reduce the solution time for complex optimization problems. In the companion paper [205] we proposed a graph classification approach to determine when to use a decomposition-based solution algorithm. In this paper, we focus on the implementation of decomposition-based solution methods.

The application of a decomposition-based solution algorithm has three steps: (1) problem decomposition, (2) coordination scheme, and (3) initialization of the algorithm. In the first step, the original problem is decomposed into a number of easier-to-solve subproblems. This step requires knowledge of the underlying structure of the problem. Automatic decomposition approaches either represent the problem as a graph and employ methods from network science to learn the underlying structure [6, 198, 207, 199, 145] or use machine learning (ML) to analyze the computational efficiency of different decompositions [24, 23]. The coordination step determines the exchange of information between the different subproblems. For distributed algorithms, the coordination determines the update of the dual variables whereas for hierarchical algorithms it involves the addition of cuts. Finally, the last step is the initialization of the algorithm. Unlike monolithic algorithms which may require an initial guess for the variables, decomposition-based

algorithms require additional information regarding the values of the dual variables for distributed algorithms or an initial set of cuts for hierarchical algorithms. The configuration of these steps can have an important effect on the computational performance of a decomposition-based solution algorithm, however, determining the best configuration for a given problem is nontrivial.

These steps can be viewed as hyperparameters of the solution algorithm. Therefore, given an optimization problem and a decomposition-based solution algorithm, one must find the optimal values of the algorithm parameters such that a desired performance function, e.g., the solution time, is optimized. Formally this is known as the algorithm configuration problem and is stated as follows [88, 258]:

**Problem 6.1.** (Algorithm configuration) Given an optimization problem $P$, an optimization algorithm $\alpha$ with parameter $n \in \mathcal{N}$, and a performance function $m : \mathcal{P} \times \Pi \mapsto \mathcal{M}$, determine the optimal values of the parameters $n^*$ that optimize $m$

$$n^* \in \arg\min_{n \in \mathcal{N}} \quad m(n, P). \tag{6.1}$$

The algorithm configuration problem has three components, the optimization problem $P$ which belongs in some class of optimization problems $\mathcal{P}$, the parameter space, i.e., all possible values of the parameters represented by $\mathcal{N}$, and the performance space $\mathcal{M}$ which is a metric to compare the different configurations. We note that the version of the problem presented above is known as the per-instance algorithm configuration problem since the optimal parameters are determined only for a given problem $P$. Alternatively, one can find the optimal values of the parameters for a class or set of optimization problems. The algorithm configuration can be either static, i.e., the parameters of the algorithms remain fixed during the solution process, or dynamic, where the parameters adapt as the solution procedure evolves. The solution of the algorithm configuration problem is challenging since optimization solvers, either monolithic or decomposition-based, have multiple algorithmic steps and each step can have different parameters. Furthermore, optimization solvers employ a number of heuristics; although these can accelerate the solution on average, their efficacy for a given problem is not known a-priory. Therefore, finding the optimal configuration of an optimization solver is a challenging black-box optimization problem since the number of possible combinations of parameters can be very large, their optimal values may change significantly for different classes of optimization problems, and the evaluation of a given configuration can be computationally expensive.

Automated algorithm configuration approaches aim to either solve the algorithm configuration problem in a computationally efficient way or approximate its solution. In the former approach, Bayesian optimization and derivative-free methods have been employed to tune optimization algorithms by optimizing directly the black-box performance function [181, 59, 140, 139, 138]. In the latter approach, ML tools have been used to approximate the performance function $m$ with a surrogate one $\hat{m}$ and then identify the optimal values of the parameters $n^*$. These methods have been traditionally applied for tuning monolithic solvers, either by considering all the parameters simultaneously [143] or specific algorithmic steps such as branching [183, 152, 14, 80, 122, 123, 179], cutting plane methods [272, 136, 231], estimating active constraints [197, 32, 33], and improving primal heuristics [83]. ML tools have also been used to tune decomposition-based solution methods. For hierarchical decomposition-based methods, such as column generation and Benders Decomposition, machine learning is used to aid the cut selection process using classification techniques [210, 147, 170], whereas for distributed algorithms, such as ADMM, machine learning is used to determine the values of the dual variables and penalty parameters [35, 300]. In both cases, ML is used in the coordination step of these algorithms.

These ML approaches are based on a handcrafted feature representation of an optimization problem which is the input to the surrogate model $\hat{m}$ [265, 141, 28, 58], and involve two steps; the first is offline where multiple problems are solved for different values of the parameters in order to create a training dataset used to learn a surrogate model for the performance function. Once the surrogate model is trained, it is used online to identify the best set of parameters for a given problem. The main limitation of these ML approaches is data availability since generating a large training data set can be computationally expensive.

In this work, we focus on the *initialization* of decomposition-based solution methods. Specifically, we focus on cutting plane-based hierarchical solution methods, such as Benders and Generalized Benders Decomposition. These algorithms are based on the observation that if a subset of the variables, called complicating variables, is fixed then the problem is either easier to solve or has a special structure. Thus the original problem is decomposed into two problems; a master problem which considers the complicating variables and a subproblem which considers the non-complicating variables and whose solution depends on the values of the complicating variables. The solution of the master problem and the subproblem is coordinated via the addition of optimality and feasibility

cuts, which inform the master problem about the bounds and the feasibility of the problem respectively. In the standard application of the algorithm, the master problem is initially solved without any cuts. Since the cuts contain information about the effect of the complicating variables on the subproblem, the addition of cuts in the first iteration can potentially lead to a reduction in the computational time since fewer iterations might be necessary. However, the addition of a large number of cuts may increase the computational complexity of the master problem which might increase the solution time. Hence it is important to identify the number of cuts that balances the amount of information added to the master problem with the increase in computational complexity. This balance is important for online applications where an optimization problem is solved repeatedly to compensate for updated process information. Depending on the application, either only the parameters of the optimization problem change, such as in model predictive control applications [271], or the parameters and the number of variables and constraints can change, such as in online scheduling applications [200, 249].

In this paper, we propose an ML approach to learn how to initialize Generalized Benders Decomposition. The proposed approach has two steps. In the first, ML is used to learn a surrogate model that estimates the solution time of the optimization problem for a given number of cuts added to the master problem in the first iteration of the algorithm. In the second step, the surrogate model is used online to determine the optimal number of cuts that should be added in the master problem in the first iteration. We apply the proposed approach to a case study on the real-time operation of process systems, where a mixed integer economic model predictive control problem is solved. Specifically, we assume that the system is an isothermal continuously stirred tank reactor (CSTR) that can manufacture a number of products, and multiple disturbances can affect the operation of the system. The mixed integer economic model predictive control problem is solved using a hybrid multicut Generalized Benders Decomposition proposed in [201], initialized using the proposed approach. The results show that (1) the optimal initialization can be achieved in an automated way without human intervention, (2) the proper initialization can lead to a significant reduction in solution time, and (3) active learning can guide the learning process either during the initial development of such frameworks or for cases where generating the training dataset is computationally expensive.

The rest of the paper is organized as follows: In Section 6.2 we present the Generalized Benders Decomposition algorithm and the different acceleration techniques, in

Section 6.3 we pose the initialization of Generalized Benders Decomposition as an algorithm configuration problem, in Section 6.4 we present the proposed approach, and in Section 6.5 we present the case study and the numerical results. Finally, in Section 6.6, given the results in the first part of this two-series paper [205] we present a unified framework for automated decomposition-based solution algorithm selection and configuration.

## 6.2 Generalized Benders Decomposition

### 6.2.1 Standard implemetation

We will assume that the following problem (denoted as $P$) must be solved:

$$\begin{aligned} P(p) := \underset{z,x,y}{\text{minimize}} \quad & f_1(z,x;p_m) + f_2(x,y;p_s) \\ \text{subject to} \quad & g_1(z,x;p_m) \leq 0 \\ & h_1(z,x;p_m) = 0 \\ & g_2(x,y;p_s) \leq 0 \\ & h_2(x,y;p_s) = 0 \\ & z \in \mathbb{Z}^{n_z}, x \in \mathbb{R}^{n_x^c} \times \mathbb{Z}^{n_x^d}, y \in \mathbb{R}^{n_y}, \end{aligned} \tag{6.2}$$

where $p = [p_m, p_s]^\top$ are the parameters of the problem, and $z \in \mathbb{Z}^{n_z}$, $x \in \mathbb{R}^{n_x^c} \times \mathbb{Z}^{n_x^d}$,$y \in \mathbb{R}^{n_y}$ are the variables. The solution of this problem depends on the values of the parameters $p$. In this problem, we observe that if the variables $z, x$ are fixed, then the resulting problem is a continuous optimization problem that depends on the values of the variables $x$, which are the complicating variables, and the parameters $p_s$. Given this structure, we can apply Generalized Benders Decomposition, by assigning the $z, x$ variables and the associated constraints in the master problem and the other variables and constraints in the subproblem. Under this decomposition, parameters $p_m$ affect only the master problem and parameters $p_s$ affect only the subproblems. The

subproblem is

$$\begin{aligned} \mathcal{S}(x, p_s) := \underset{\bar{x}, y}{\text{minimize}} \quad & f_2(\bar{x}, y; p_s) \\ \text{subject to} \quad & g_2(\bar{x}, y; p_s) \leq 0 \\ & h_2(\bar{x}, y; p_s) = 0 \\ & \bar{x} = x \quad : \quad \lambda \\ & \bar{x} \in \mathbb{R}^{n_x^c + n_x^d}, y \in \mathbb{R}^{n_y}, \end{aligned} \tag{6.3}$$

where $\lambda$ are the Lagrangean multipliers for the equality constraint $\bar{x} = x$. The solution of this problem depends on the values of the complicating variables $x$ and the parameters $p_s$, and for a given value of $x = \bar{x}$ the value function of the subproblem can be approximated as follows:

$$\mathcal{S}(x, p_s) \geq \mathcal{S}(\bar{x}, p_s) - \bar{\lambda}(x - \bar{x}), \tag{6.4}$$

where $\bar{\lambda}$ is equal to the value of the Lagrangean multiplier at the optimal solution of the subproblem when solved for $x = \bar{x}$. Note that we assume that the subproblem is always feasible for all values of $x$. The master problem is:

$$\begin{aligned} \mathcal{M}(\cdot) := \underset{z, x, \eta}{\text{minimize}} \quad & f_1(z, x; p_1) + \eta \\ \text{subject to} \quad & g_1(z, x; p_2) \leq 0 \\ & h_1(z, x; p_3) = 0 \\ & \eta \geq f_2(\bar{x}^l, \bar{y}^l; p_4) - \lambda^l(x - \bar{x}^l) \ \forall l \in \mathcal{L} \\ & z \in \mathbb{Z}^{n_z}, x \in \mathbb{R}^{n_x^c} \times \mathbb{Z}^{n_x^d}, \end{aligned} \tag{6.5}$$

where $\mathcal{M}(\cdot) = \mathcal{M}(p_m, \mathcal{L})$, $l$ is the iteration number and the set $\mathcal{L}$ denotes the index of the Benders cuts. The steps of GBD are presented in Algorithm 1.

### 6.2.2 Acceleration techniques for Benders decomposition

The algorithm alternates between the solution of the master problem and the subproblem, therefore the computational performance depends on the complexity of the master problem and subproblem, i.e., the solution time per iteration, the number of infeasible subproblems that must be solved, and the quality of cuts that are generated during the solution. Two approaches can be followed to handle these issues. The first one is based on the theoretical aspects of the algorithm and the underlying geometry of the

**Data:** Optimization problem
**Result:** Upper, lower bound and variable values
Set $UB = \infty$, $LB = -\infty$;
Set tolerance and optimality gap (tol);
Initialize the algorithm;
**while** $(UB - LB)/LB \geq tol/100$ **do**
    Solve the master problem (Eq. 6.5) and obtain $LB, x$;
    Solve the subproblem (Eq. 6.3) and obtain $y$;
    Add Benders cut (Eq. 6.4);
    Update the upper bound $f_1(z, x; p_m) + f_2(x, y; p_s)$;
**end**

**Algorithm 6.1:** Generalized Benders Decomposition

problem. Common strategies in this approach are problem reformulation and decomposition [74, 185], the addition of valid inequalities in the master problem to reduce the number of infeasible subprolems [252], multicut implementation [295] for stochastic optimization problems, cut generation and management [185, 269, 253, 225, 282], and regularization/stabilization of the master problem [250, 178]. The second approach involves the use of machine learning. For example, machine learning has been used to develop a classifier to determine which cuts should be added in the master problem during the multicut implementation of the algorithm for the solution of two-stage mixed integer stochastic optimization problems [147, 170]. Machine learning has also been used to approximate the solution of the subproblem reducing the solution time for cases where the subproblem is computationally complex, such as two-stage stochastic optimization problems [168] and mixed integer model predictive control problems [202]. These acceleration methods, both the ones based on the underlying geometry and the ones using machine learning, reduce the computational time either by reducing the solution time per iteration or by reducing the number of iterations.

## 6.3 Initialization of GBD as an algorithm configuration problem

We will assume that problem $P(p)$ must be solved repeatedly given new values of the parameters $p$. The problem that we will address is the following:

**Problem 6.2.** Given an optimization problem $P(p)$ determine the optimal number of

cuts to add in the master problem in the first iteration of Generalized Benders Decomposition, such that the CPU time is minimized.

We can pose this as a per-instance algorithm configuration problem as follows

$$\underset{n \in \mathcal{N}_{cuts}}{\text{minimize}} \; m(n, P(p)), \tag{6.6}$$

where the optimization problem is $P(p)$, the parameter space $\mathcal{N}_{cuts}$ represents the cuts that can be added, and the performance function is the solution time $\mathcal{M} = \mathbb{R}_+$. The solution of the algorithm configuration problem for Generalized Benders Decomposition has three main challenges. The first, which is common in algorithm configuration problems, is that the performance function $m$ is not known a-priory. The second issue is related to the parameter space, i.e., all the cuts that can potentially be used to initialize the master problem. In general, selecting which cuts to use is a challenging problem that arises in the solution of many classes of problems such as quadratic [19, 189], and mixed integer linear programming problems [210, 60]. Regarding Generalized Benders Decomposition, the number of cuts that can be evaluated depends on the number and type of complicating variables. For cases where the complicating variables are integer and $n_c$ cuts must be added, the cut selection process leads to a combinatorial optimization problem since one must select $n_c$ cuts from all possible cuts. The situation is more complex when the complicating variables are continuous, since in this case an infinite number of cuts can be added even for one complicated variable. Finally, all the parameters of the problem can change simultaneously, thus the cuts must be evaluated continuously as the parameter of the problem change. Hence, although the addition of cuts in the master problem might reduce the number of iterations required for convergence, the solution time might increase since multiple subproblems must be solved to evaluate the initial set of cuts.

We will assume that (1) the parameters of the subproblem do not change, (2) all the complicating variables are continuous, and (3) $n_c$ cuts can be added by discretizing the domain of the complicating variables ($x \in [x^{lb}, x^{ub}]$) into $n_c$ uniform points. The first assumption guarantees that the Benders cuts must be evaluated only once since even if the parameters of the master problem change, the Benders cuts are still valid estimators of the value function of the subproblem, since the parameters of the subproblem do not change. Therefore, the process of adding these cuts to the master problem does not affect the total solution time. The second and third assumptions determine the cut

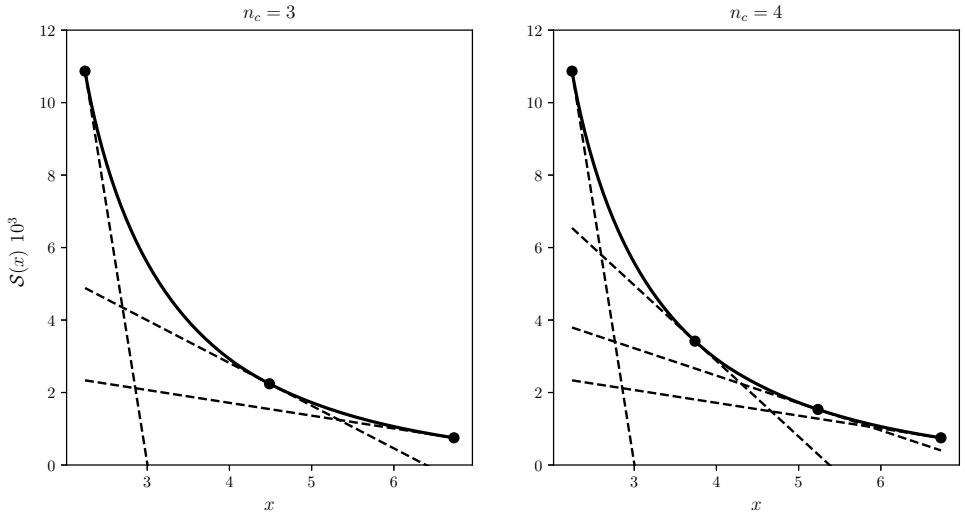


Figure 6.1: Domain discretization for the case study considered in Section 6.5 for a transition from product 1 to 2 for three and four number of cuts ($n_c$). The solid line is the value function, $x \in [2.24, 6.73]$ is the complicating variable, the dotted lines are the value function approximations, i.e., Benders cuts, evaluated ad the points indicated by the dots.

selection strategy. In this work, the number of cuts determines the number of points that will be used to approximate the value function of the subproblem via Benders cuts. An example is presented in Fig. 6.1, where the value function corresponds to the system discussed in Section 6.5 and the approximation with three and four cuts is presented. This setting enables us to simplify the cut selection process for the case of continuous complicating variables. Notice that in this setting the cuts are uniformly distributed in the domain of the complicating variables.

## 6.4 Learning to initialize via supervised and active learning

The goal is to learn a surrogate model $\hat{m}$ which approximates the solution time and can be used to identify the optimal number of cuts to add in the master problem,

$$n^* \in \arg\min_{n \in \mathbb{Z}_+} \ \hat{m}(n, \nu(\mathcal{P}(p))), \tag{6.7}$$

where $\nu(P(\mathbf{p}))$ are some features of problem $P(p)$. We propose two ML approaches for learning the surrogate model.

### 6.4.1 Supervised learning approach

The estimation of the parameters of a surrogate model requires data that capture the relation between the features of an optimization problem $\nu(P(p))$ and the number of cuts $n$ with the solution time. Such data can be generated using the procedure presented in Algorithm 6.2, where first random values of the parameters of the master problem ($p_m$) are generated based on some underlying probability distribution. Next, the optimization problem $P(p_i)$ is solved for a fixed number of cuts $n_i$, and the solution time $y_i$ is obtained. Finally, the features of the problem $\nu(P(p_i))$ are obtained and the tuplet $(n_i, \nu(P(p_i)))$ and the solution time $y_i$ are stored. Once this procedure is completed, we obtain the dataset $\mathcal{D} = \{(n_i, \nu(P(p_i))), y_i\}_{i=1}^{N_{data}}$. This dataset can be used to learn the parameters of the surrogate model by optimizing some loss function such as the squared error between the model prediction and the data. Once the learning step is completed, the surrogate model can be used to learn how many cuts to add for online applications are presented in Algorithm 6.3, where given new values of the parameters of the optimization problem, the features of the problem are obtained and the optimal number of cuts is computed by optimizing the surrogate model. Finally, the cuts are added to the master problem and Generalized Benders Decomposition is implemented.

### 6.4.2 Active learning approach

The main limitation of the supervised learning approach is the computational time required to generate the training data, since for every value of the parameters $p_i$ and number of cuts $n_i$ the problem must be solved to obtain the solution time. This approach can be computationally expensive, even intractable for complex optimization problems. To resolve this we propose the application of active learning [260], a commonly used approach in machine learning tasks where the features of the data are known but obtaining their label is costly or time-consuming. Unlike supervised learning where all the data are available for training, in active learning the model itself determines which data should be labeled and thus be used for training the surrogate model.

The active learning paradigm has three components. The first is the unlabeled data which can be generated de novo (known as membership query synthesis) [8], can become available in an online setting (stream-based selective sampling) [68], or can be

**Data:** Optimization problem, number of data points $N_{data}$, number of discretization points $N_{cuts}$, upper and lower bounds for complicating variables $x \in [x^{lb}, x^{ub}]$, $\check{p} = [p_4, p_5, p_6]$
**Result:** Surrogate model $\hat{m}$

$i = 1$;
**while** $i \leq N_{data}$ **do**
  Generate parameters $p_1, p_2, p_3 \rightarrow p_i = [p_1, p_2, p_3, \check{p}]$;
  **for** $j = 2 : N_{cuts}$ **do**
    Solve problem $\mathcal{P}(p_i)$ using $j$ cuts for each $x$;
    Obtain CPU time $y_i$;
    Obtain features of the problem $s_j = (\nu(\mathcal{P}(p_i)), j)$;
    Append data $\{s_j, y_j\}$;
  **end**
  $i = i + 1$;
**end**
Using data $\{s_i, y_i\}_{i=1}^{N_{data}(N_{cuts}-1)}$ learn parameters of a surrogate model $\hat{m}$;

**Algorithm 6.2:** Learning the relation between number of cuts and CPU time for a general optimization problem for continuous complicating variables

gathered at once (pool-based sampling) [171]. The second aspect is the query strategy, which determines which data should be labeled. This decision is taken by considering the informativeness of the available unlabeled data. Typical query strategies are uncertainty sampling [171], query by committee [262], and expected model change [261]. We refer the reader to [260] for a detailed discussion of the different sampling methods. The last component is an oracle which generates the label for a given input. Typical example of an oracle is a human expert, a computer simulation or the outcome of an experiment.

In this work, we will use the pool-based active learning paradigm with uncertainty-based sampling, where the data point for which the model is the least certain is labeled. The basic steps of the application of active learning for learning the solution time of Generalized Benders Decomposition are presented in Fig. 6.2.

### Generation of pool of labels and initial training set

For the application of active learning, first, the pool of labels is created. We generate random values of the parameter $p_i$ and for every parameter the features of the optimization problem $\nu(P(p_i))$ are obtained and a number of cuts $n_i$ is selected. This forms the pool of features $\mathcal{C}_p = \{s_i\}_{i=1}^{N_{pool}}$ $(s_i = \{n_i, \nu(P(p_i))\})$. Next a small set of training

dataset is obtained by sampling $N_{initial}$ data points from the pool and evaluating the solution time $y_i$. This is the initial training set $\mathcal{C} = \{s_i, y_i\}_{i=1}^{N_{initial}}$ which we will refer to as labeled training set.

Given the unlabeled pool $\mathcal{C}_p$ and the labeled dataset $\mathcal{C}$, the surrogate model considers all the data in the pool, and the data point $s = \{n_s, \nu(P(p_s))\}$ for which the prediction is the least certain about is selected for labeling.

**Undertainty based sampling using Gaussian Processes**

The selection of the data point requires quantification of the prediction uncertainty. Gaussian Process Regression (GPR) is a non-parameteric Bayesian approach which can provide uncertainty measures [290]. GPR is based on the Gaussian Process which is a stochastic process that defines a distribution over functions. Specifically, given observations of the input variables $X = [x_1, ..., x_N]$ and measurements of the output $Y = [y_1, ..., y_N]$, the relationship between $X$ and $Y$ is modelled as a Gaussian multivariate distribution. GPR seeks to learn a mapping $f : X \mapsto Y$, i.e., $y = f(x)$, with mean $m(x)$ and covariance $k(x, x')$ where ($\mathbb{E}$ is the expected value)

$$\begin{aligned} m(x) &= \mathbb{E}[f(x)] \\ k(x, x') &= \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))]. \end{aligned} \tag{6.8}$$

This is done under the assumption that the data are indepedent and the probability to observe an output given the observations can be factored over cases in the training set. The Gaussian Process, is written as $f(x) \sim \mathcal{GP}(m(x), k(x, x'))$. Different kernel functions $k(\cdot, \cdot)$ can be used, however in this paper we use the Matern kernel given by the following equation

$$k(\cdot, \cdot) = \frac{1}{\Gamma(\nu)2^{\nu-1}} \left( \frac{\sqrt{2\nu}}{\ell} d(\cdot, \cdot) \right)^{\nu} K_v \left( \frac{\sqrt{2\nu}}{\ell} d(\cdot, \cdot) \right), \tag{6.9}$$

where $d(\cdot, \cdot)$ is the Eucledian distance between features $s_i$ and $s_j$, $\Gamma(\cdot)$ is the gamma function, $K_v(\cdot)$ is the modified Bessel function, and $\ell, \nu$ are tunable hyperparameters. During training the parameters of the mean and kernel function are estimated based on the available data. This step estimates the posterior distribution over functions that best explain the data. This posterior distribution is also Gaussian and is used to make predictions for a new data point. We refer the reader to [290] for detailed explanation

of these steps.

### Oracle

The oracle is the Generalized Benders Decomposition algorithm which given the parameters of an optimization problem and a number of cuts, solves the optimization problem and returns the solution time. We note that although here we consider the standard version of Generalized Benders Decomposition as the oracle, in principle, other vesions can be incorporated, such as multicut implementation, partial Benders decomposition etc.

### Active learning loop

The active learning loop is presented in Algorithm 6.4. The inputs are the pool of unlabeled data $\mathcal{C}_p$, the initial training set $C$, and the surrogate model $\hat{m}$ which is a Gaussian Process with Matern kernel. First, the model is trained using the initial training dataset. Next, the model is used to predict the solution time and undertainty arount the prediction for all the unlabeled data and idenitfy the datapoint $s$ with the maximum uncertainty. This data point is passed to the Generalized Benders Decomposition algorithm and the solution time (the label) is recored , the labeled datapoint $\{s, y\}$ is appended in the training dataset, and the datapoint with label $s$ is removed form the pool. The surrogate model is trained again using the new training dataset and this loop continues until the maximum munber of iterations is reached. The outcome of this approach is the surrogate model $\hat{m}$.

## 6.5 Application to mixed integer economic model predictive control for real time operation of chemical processes

In this section, we apply the proposed method for learning to initialize Generalized Benders Decomposition for the solution of mixed integer economic model predictive control problems that arise in the operation of chemical processes. Specifically, we consider a continuous manufacturing system whose operation can be affected by disturbances in the scheduling, e.g., change in demand, and control level, e.g., change in the inlet conditions of the process, as presented in Fig. 6.3. Once a disturbance affects the system

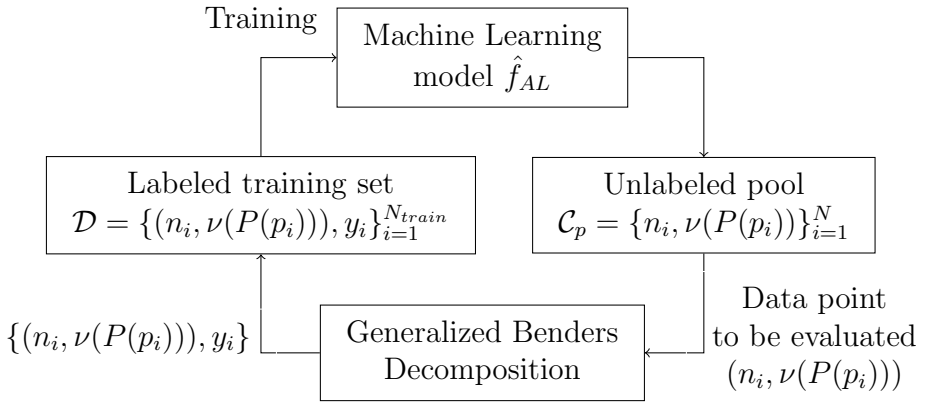


Figure 6.2: Learning to initialize Generalized Benders Decomposition via active learning framework

**Data:** Surrogate model $\hat{m}$, Optimization problem $P$, value of parameters $p$
**Result:** Problem solution
1 Compute the features of the problem $\nu(P(p))$;
2 Determine the optimal number of cuts $n^*_{cuts} = \arg\min_{n \in \mathcal{N}_{cuts}} \hat{m}(N, \nu(P(p)))$;
3 Add $n^*_{cuts}$ cuts in the master problem;
4 Solve the optimization problem using Algorithm 1;

**Algorithm 6.3:** Regression based initialization of Generalized Benders decomposition

an optimization problem is solved to determine the production sequence and the transitions between the products. In this section, we present the optimization model, the decomposition-based solution approach, the data generation process, and the evaluation of the different learning approaches.

### 6.5.1 Optimization model

#### Scheduling model

We will consider the case where an isothermal CSTR is used to produce $N_p$ products over a time horizon of $H$ hours which is discretized into $N_s$ slots ($N_s = N_p$). We will assume that while the system is following a nominal schedule a disturbance affects the

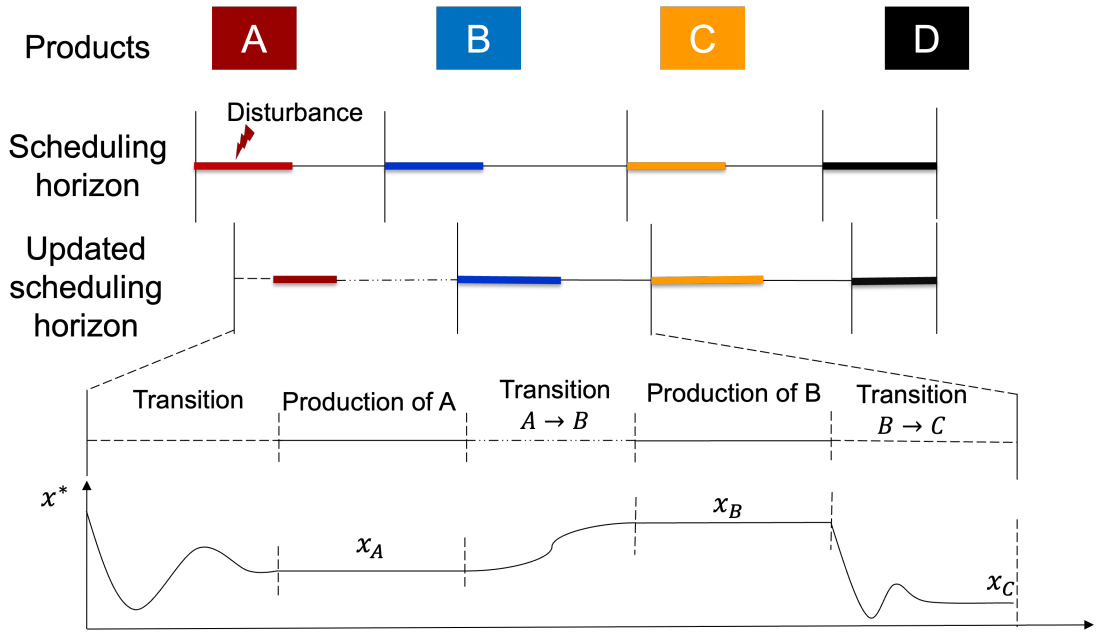


Figure 6.3: Schematic of rescheduling

**Data:** Number of evaluations $N$, initial labeled dataset $\mathcal{C} = \{s, y\}$, Pool of labels $\mathcal{C}_p$, Surrogate model $\hat{m}$
**Result:** Surrogate model $\hat{m}$
Train surrogate model $\hat{m}$ on the initial dataset $\mathcal{C}$;
**while** $i \leq N$ **do**
  Select a data point with features $s$ from pool $\mathcal{C}_p$ based on maximum uncertainty sampling strategy $s \in \arg\max_{s \in \mathcal{C}_p} \sigma(s)$;
  Evaluate label $y$ for $s$ using GBD;
  Append data $\{s, y\}$ in set $\mathcal{C} = \mathcal{C} \cup \{s, y\}$;
  Remove data-point $s$ from pool $\mathcal{C}_p = \mathcal{C}_p \setminus \{s\}$;
  Train surrogate model $\hat{m}$ using set $\mathcal{C}$;
  $i = i + 1$;
**end**

**Algorithm 6.4:** Learning surrogate model for solution time via active learning

system at time $T_0$, as presented in Fig. 6.3. Under this setting, every slot $k$ except the first one has two regimes; a production regime where a product is manufactured and a transition regime where a transition occurs from the operating point of the product manufactured at slot $k$ to the operating point of the product manufactured at slot $k+1$.

The first slot has three regimes; a transition regime that captures the transition from some intermediate state (where the system is due to the disturbance) to the operating point of the product manufactured in the first slot, a production regime, and another transition regime which considers the transition from the product manufactured at the first slot to the product manufactured in the second slot.

We define a binary variable $W_{ik}$ which is equal to one if product $i$ is manufactured in slot $k$ and zero otherwise. Also, we define a binary variable $Z_{ijk}$ which is equal to one if a transition occurs from product $i$ to $j$ in slot $k$, and variable $\hat{Z}_i$ which is equal to one if a transition occurs from an intermediate state to product $i$ in the first slot. At every time point, only one product can be manufactured. The logic constraints are:

$$\begin{aligned} &\sum_{i=1}^{N_p} W_{ik} = 1 \ \forall k = 1, .., N_s \\ &Z_{ijk} \geq W_{ik} + W_{j,k+1} - 1 \ \forall i, j, k \neq N_s \\ &\hat{Z}_i = W_{i1} \ \forall i \end{aligned} \tag{6.10}$$

The starting time of slot $k$ is $T_k^s$, the ending time $T_k^e$, the production time of product $i$ in slot $k$ is $\Theta_{ik}$, and the transition time in slot $k$ is $\theta_k^t$. The timing constraints are:

$$\begin{aligned} T_k^e &= T_k^s + \sum_{i=1}^{N_p} \Theta_{ik} + \theta_k^t \ \forall k = 1, .., N_s \\ T_{k+1}^s &= T_k^e \ \forall k = 1, .., N_s - 1 \\ T_{N_s}^e &= H - T_0 \\ \Theta_{ik} &\leq H W_{ik} \ \forall i = 1, .., N_p, k = 1, .., N_s \\ \theta_1^t &= \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} Z_{ij1} \theta_{ij1} + \sum_{i=1}^{N_p} \hat{Z}_i \hat{\theta}_i \\ \theta_k^t &= \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} Z_{ijk} \theta_{ijk} \ \forall k = 2, .., N_s \\ \theta_{ijk} &\geq \theta_{ij}^{min} \ \forall i, j, k \\ \hat{\theta}_i &\geq \hat{\theta}_i^{min} \ \forall i \end{aligned} \tag{6.11}$$

where $\theta_{ijk}$ is the transition time from product $i$ to $j$ in slot $k$, $\hat{\theta}_i$ is the transition time from an intermediate state to the steady state of product $i$, $\theta_{ij}^{min}$ is the minimum

transition time from product $i$ to $j$, and $\hat{\theta}_i^{min}$ is the minimum transition time from the intermediate state to the steady state of product $i$. The production rate of product $i$ is $r_i$, the production amount of product $i$ in slot $k$ is $q_{ik}$, and the inventory of product $i$ in slot $k$ is $I_{ik}$. The production constraints are

$$\begin{aligned} I_{ik} &= I_{ik-1} + r_i\Theta_{ik} - S_{ik} \ \forall i = 1, .., N_p, k = 2, .., N_p \\ I_{ik} &= I_i^0 + r_i\Theta_{ik} - S_{ik} \ \forall i = 1, .., N_p, k = 1, \end{aligned} \tag{6.12}$$

where $I_i^0$ is the initial inventory of product $i$. The demand of product $i$ is $d_i$ and the due date for every product is in the end of the time horizon. The demand satisfaction constraints are

$$S_{iN_s} \geq d_i \ \forall i = 1, .., N_p \tag{6.13}$$

**Dynamic model**

We will assume that the system is described by a system of differential equations

$$\dot{x} = F(x, u), \tag{6.14}$$

where $x \in \mathbb{R}^{n_x}$, $u \in \mathbb{R}^{n_u}$ are the state and manipulated variables, and $F : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \mapsto \mathbb{R}^{n_x}$ are vector functions. We will consider simultaneously all transitions between the products and discretize the differential equations using the method of orthogonal collocation on finite elements (using $N_f$ finite elements and $N_c$ collocation points). We define state variable $x_{ijkfc}$ and manipulated variable $u_{ijkfc}$ for a transition from product $i$ to $j$ in slot $k$, finite element $f$ and collocation point $c$. We also define variable $\hat{x}_{ifc}$ and manipulated variable $\hat{u}_{ifc}$ for a transition from the intermediate state to product $i$ in finite element $f$ and collocation point $c$ in the first slot. The discretized differential equations for transitions between products are

$$\begin{aligned} x_{ijkfc} &= F_d(x_{ijkfc}, u_{ijkfc}, \theta_{ijk}) \ \forall i, j, k, f, c \\ x_{ijk11} &= x_i^{ss} \ \forall i, j, k \\ x_{ijkN_fN_c} &= x_j^{ss} \ \forall i, j, k \\ u_{ijk11} &= u_i^{ss} \ \forall i, j, k \\ u_{ijkN_fN_c} &= u_j^{ss} \ \forall i, j, k \end{aligned} \tag{6.15}$$

and the equations for the transition from the intermediate state to product $i$ are:

$$\begin{aligned}
\hat{x}_{ifc} &= F_d(\hat{x}_{ifc}, \hat{u}_{ifc}, \hat{\theta}_i) \ \forall i, l \\
\hat{x}_{i11} &= x^* \ \forall i \\
\hat{x}_{iN_fN_c} &= x_i^{ss} \ \forall i \\
\hat{u}_{i11} &= u^* \ \forall i \\
\hat{u}_{iN_fN_c} &= u_i^{ss} \ \forall i
\end{aligned} \tag{6.16}$$

where $x_i^{ss}$, $u_i^{ss}$ are the steady state values of the state and manipulated variables of product $i$. A detailed expression for the discretized differential equations can be found at [201].

### Objective function

The objective function has three terms; the first is the profit $\Phi_1$, the second is the transition cost between products $\Phi_2$, and the third is the transition cost from the intermediate state $\Phi_3$. These terms are equal to:

$$\begin{aligned}
\Phi_1 &= \sum_{i,k} p_{ik} S_{ik} - C_{ik}^{oper} q_{ik} - C^{inv} I_{ik} - \sum_{ijk} C_{ijk}^{trans} Z_{ijk} \\
\Phi_2 &= \sum_{ijk} Z_{ijk} \alpha_u \left( \sum_{fc} N_f^{-1} t_{ijfck}^d \Lambda_{cN_c} (u_{ijfck} - u_j^{ss})^2 \right) \\
\Phi_3 &= \sum_{i} \hat{Z}_i \alpha_u \left( \sum_{fc} N_f^{-1} \hat{t}_{ifc}^d \Lambda_{cN_c} (\hat{u}_{ifc} - u_j^{ss})^2 \right)
\end{aligned} \tag{6.17}$$

where $p_i, C_i^{op}$ are the price and operating cost of product $i$, $C^{inv}$ is the inventory cost, $C_{ij}^{tr}$ is the transition cost from product $i$ to $j$, $a_u$ is a weight coefficient, and $\Lambda$ is the collocation matrix.

### Mixed Integer Model Predictive Problem and decomposition-based solution

The mixed integer MPC problem is:

$$\begin{aligned}
P(p): \text{minimize} \ & \ \Phi_1 - \Phi_2 - \Phi_3 \\
\text{subject to} \ & \ \text{Eq. } 6.10, 6.11, 6.12, 6.13, 6.15, 6.16.
\end{aligned} \tag{6.18}$$

where $p = \{\{d_i\}_{i=1}^{N_p}, \{I_i^0\}_{i=1}^{N_p}, T_0, \{\theta_{ij}^{min}\}_{i=1,j=1}^{N_p N_p}, \{\hat{\theta}_i\}_{i=1}^{N_p}, \{r_i\}_{i=1}^{N_p}, \{x_i^{ss}\}_{i=1}^{N_p}, \{u_i\}_{i=1}^{N_p}, x^*\}$. This problem has three sets of parameters; the ones related to the scheduling part of the problem $\hat{p} = \{\{d_i\}_{i=1}^{N_p}, \{I_i^0\}_{i=1}^{N_p}, T_0, \{\theta_{ij}^{min}\}_{i=1,j=1}^{N_p N_p}, \{\hat{\theta}_i\}_{i=1}^{N_p}, \{r_i\}_{i=1}^{N_p}\}$, ones related to the dynamic behavior of the system for transitions between products $\check{p} = \{\{x_i^{ss}\}_{i=1}^{N_p}, \{u_i^{ss}\}_{i=1}^{N_p}\}$, and the ones related to the transition from the intermediate state to the steady state of the different products $\tilde{p} = x^*$.

We will rewrite the mixed integer economic MPC problem (Eq. 6.18) as follows

$$\begin{aligned} \text{maximize} \quad & \Phi_1(w) - \sum_{ijk} Z_{ijk} f_{dyn}^{ijk}(\omega_{ijk}, \theta_{ijk}) - \sum_i \hat{Z}_i f_{dyn}^i(\hat{\omega}_i, \hat{\theta}_i) \\ \text{subject to} \quad & g_{sched}(w, \theta_{ijk}, \hat{\theta}_i; \hat{p}) \leq 0 \\ & g_{ijk}^{dyn}(\theta_{ijk}, \omega_{ijk}; \check{p}) \leq 0 \ \forall i, j, k \\ & \hat{g}_i^{dyn}(\hat{\theta}_i, \hat{\omega}_{ijk}; \hat{p}) \leq 0 \ \forall i \end{aligned} \tag{6.19}$$

where $w = \{W_{ik}, Z_{ijk}, \hat{Z}_i, T_k^s, T_k; \Theta_{ik}, \theta_k^t, S_{ik}\}$ are scheduling variables, $\omega_{ijk} = \{x_{ijkfc}, u_{ijkfc}\}$ are variables associated with the dynamic behavior of the system for a transition from product $i$ to product $j$ in slot $k$, and $\hat{\omega}_i = \{\hat{x}_{ifc}, \hat{u}_{ifc}\}$ are variables associated with the dynamic behavior of the system for a transition from the intermediate state to the product $i$. $g_{sched}$ are the scheduling constraints (Eq. 6.10, 6.11, 6.12, 6.13), $g_{ijk}^{dyn}$ are the discretized differential equations for transitions between products (Eq. 6.15), and $\hat{g}_i^{dyn}$ are the discretized differential equations for transition from the intermediate state to product $i$ (Eq. 6.16)

If we fix the scheduling variables $w$ and the transition times $\theta_{ijk}, \hat{\theta}_i$ then the dynamic optimization problems for all the transitions can be solved independently. We define as $\phi_{ijk}$ the value function of the dynamic optimization problem for a transition from product $i$ to product $j$ in slot $k$. The dynamic optimization problem for this transition is

$$\begin{aligned} \phi_{ijk}(\theta_{ijk}) := \text{minimize} \quad & f_{dyn}^{ijk}(\omega_{ijk}, \tilde{\theta}_{ijk}) \\ \text{subject to} \quad & g_{dyn}(\tilde{\theta}_{ijk}, \omega_{ijk}) \leq 0 \\ & \tilde{\theta}_{ijk} = \theta_{ijk} \quad : \lambda_{ijk}. \end{aligned} \tag{6.20}$$

where $\lambda_{ijk}$ is the Lagrangean multiplier for the equality constraint. Similarly, we define the value function for a transition from the intermediate state to product $i$, $\hat{\phi}_i$, and the

dynamic optimization problem for this transition is

$$\begin{aligned} \hat{\phi}_i(\hat{\theta}_i, \check{p}) := \text{minimize} \quad & f_{dyn}^{ijk}(\hat{\omega}_i, \check{\theta}_i) \\ \text{subject to} \quad & g_{dyn}(\check{\theta}_i, \hat{\omega}_i; \check{p}) \leq 0 \\ & \check{\theta}_i = \hat{\theta}_i \quad : \hat{\lambda}_i. \end{aligned} \tag{6.21}$$

We note that these dynamic optimization problems are always feasible, since the transition times are bounded from below by the minimum transition times, i.e., $\theta_{ijk} \geq \theta_{ij}^{min}$, $\hat{\theta}_i \geq \hat{\theta}_i^{min}$. The value functions $\phi_{ijk}, \hat{\phi}_i$ can be approximated with Benders cuts given by the following equations [108]

$$\begin{aligned} \eta_{ijk} \geq & \phi_{ijk}(\bar{\theta}_{ijk}^l) - \lambda_{ijk}^l(\theta_{ijk} - \bar{\theta}_{ijk}^l) \\ \hat{\eta}_i \geq & \hat{\phi}_i(\bar{\hat{\theta}}_i^l) - \hat{\lambda}_i^l(\hat{\theta}_i - \bar{\hat{\theta}}_i^l), \end{aligned} \tag{6.22}$$

where $l \in \mathcal{L}$ denotes the number of points used to approximate the value functions. The original problem can be reformulated as

$$\begin{aligned} \text{maximize} \quad & \Phi_1(w) - \sum_{ijk} Z_{ijk}\eta_{ijk} - \sum_i \hat{Z}_i\hat{\eta}_i \\ \text{subject to} \quad & g_{sched}(w, \theta_{ijk}, \hat{\theta}_i) \leq 0 \\ & \eta_{ijk} \geq \phi_{ijk}^l(\bar{\theta}_{ijk}^l) - \lambda_{ijk}^l(\theta_{ijk} - \bar{\theta}_{ijk}^l) \ \forall i, j, k, l \\ & \hat{\eta}_i \geq \hat{\phi}_i(\bar{\hat{\theta}}_i) - \hat{\lambda}_i^l(\hat{\theta}_i - \bar{\hat{\theta}}_i^l) \ \forall i, l. \end{aligned} \tag{6.23}$$

To solve the problem we use a hybrid multicut Generalized Benders Decomposition algorithm proposed in [201]. In this algorithm, the solution of the master problem provides the production sequence and the transition times, and Benders cuts are added to approximate the transition cost. In this case, the dynamic optimization problems between the products depend only on the transition time, whereas the transition from the intermediate state depends on the transition time and the concentration of the intermediate state. Therefore, the initialization of the algorithm considers only the optimal number of cuts added to approximate the transitions between the products, i.e. how many points should be used to approximate $\phi_{ijk}$. We use the same number of cuts for all transitions.

**Data:** Number of data points $N_{data}$, Maximum number of discretization points $N_{cuts}$, upper and lower bounds for complicating variables $\theta_{ijk} \in [\theta_{ij}^{min}, 5\theta_{ij}^{min}]$, Demand distribution for every product, Distribution of inlet concentration in the reactor, Scheduling horizon $H$

**Result:** Pool of unlabeled data $\mathcal{C}_p$

$\mathcal{C}_p = \{\}$;
$i = 0$;
**while** $i \leq N_{data}$ **do**
    Select at random a time point $T_0 \sim U(0, H)$;
    Introduce a disturbance in the inlet concentration of the rector $c_0 \sim U(0.8, 1.2)$;
    Generate new demand for every product $d_i \sim U(\underline{d}_i, \bar{d}_i)$;
    Compute inventory $I_0$ at time $T_0$;
    Get the value of the concentration in the reactor $x^*$;
    Obtain minimum transition times $\hat{\theta}_i^{min}$ from $x^*$ to $x_i^{ss}$;
    Form and solve the master problem;
    **if** *Master problem is feasible* **then**
        Select at random $\pi$ cuts to add in the master problem $\pi \sim U(2, N_{cuts})$;
        Computer features $s$ and append the data point in the pool $\mathcal{C}_p$;
    **else**
        The demand can not be satisfied at the end of the time horizon due to the disturbance, data point is not considered
    **end**
**end**

**Algorithm 6.5:** Unlabeled data generation procedure for creating the pool of unlabeled data for active learning

### 6.5.2 Application of active learning approach

For the application of active learning, first, we generate the pool of unlabeled data as presented in Algorithm 6.5. We assume that at a random time point $T_0$ between 0 and the end of the scheduling horizon, the demand of all the products and the inlet concentration of the reactor change simultaneously based on some probability distributions. The statistics of the demand are presented in Table 6.2 and we assume that the inlet concentration follows a uniform distribution with a low value of 0.8 and a high value of 1.2. Once the disturbance occurs, we compute the inventory, based on the realization of the initial schedule that was followed for $T_0$ hours and the concentration $x^*$ inside the reactor. Next, the minimum transition time from the intermediate state $x^*$ to the

steady state $x_i^{ss}$ is computed for all the products. Given this information, we formulate a new master problem where the scheduling horizon is $H - T_0$ and check its feasibility. If the master problem is infeasible, then the disturbance that was generated can not be rejected such that the system can meet the demand at the end of the scheduling horizon. If the master problem is feasible, then $n_c \sim U(2, N_{cuts})$ cuts are selected randomly and the features of the problem and the number of cuts are added in the unlabeled pool $\mathcal{C}_p$. Note that the feasibility of the mixed integer MPC problem can be determined by the feasibility of the master problem without any cuts since the subproblems are always feasible and the only source of infeasibility in the mixed integer MPC problem is due to the inability to satisfy the demand.

The features of the problem, in this case, are the time point $T_0$, the concentration in the reactor $x^*$, the inlet flowrate $Q$, the demand of the products $\{d_i\}_{i=1}^{N_{prod}}$, inventory of the products $\{I_i^0\}_{i=1}^{N_{prod}}$, state of the system, i.e., production or transition, and the number of cuts added $n_{cuts}$. For the state of the system, we use one-hot encoding, i.e., $state \in \{0, 1\}$. Overall the features $s_i$ for data point $i$ are:

$$s_i = [T_0, x^*, Q, \{d_i\}_{i=1}^{N_{prod}}, \{I_i^0\}_{i=1}^{N_{prod}}, state, n_{cuts}]$$

These features form the pool $\mathcal{C}_p = \{s_i\}_{i=1}^{N_{pool}}$ of data points with $N_{pool} = 49000$. Next, we generate a small number of data points ($N_{initial} = 10$) and evaluate the CPU time. We allow 100 evaluations $N = 100$, i.e., 100 data points are labeled. The computational time for obtaining the data is 726 seconds. The active learning appraoch is implemented using scikit-learn [232] and the mulitcut Generalized Benders Decomposition is implemented in Pyomo [129], the master problem is solved using Gurobi [124] and the subproblem is solved with IPOPT [285].

### 6.5.3 Comparison of active and supervised learning

We compare the proposed active learning approach, denoted as GP-AL, with a random sampling of 110 points from the pool using different surrogates models such as Gaussian Process (GP) with Matern kernel, Neural Network (NN), Random Forest (RF) and Decision Tree (DT). The hyperparameters of the Gaussian Process, Random Forest, and the Decision Tree were set equal to their default values while the neural network has 3 layers, 150 neurons per layer, and `tanh` as activation function.

We generate 100 new disturbances that are not part of the pool and were not used for

training in either the active or supervised learning approach. The solution time statistics for the different models are presented in Table 6.1. From the results, we observe that the active learning approach leads on average to 66.5% reduction in CPU time compared to the standard application of GBD, whereas the Gaussian process, neural network, random forest, and decision tree lead to $53\%, 43\%, 51\%$ and 33% reduction respectively.

Additionally, for the 100 random disturbances considered, the solution time obtained from the active learning approach is always lower than the solution time without the addition of cuts. The minimum percentage reduction in solution time for the active learning approach is 0.09%, whereas for the surrogate models learned via random sampling, the minimum reduction is negative, i.e., the solution time of the proposed approach is higher than the original implementation of Generalized Benders Decomposition. These results indicate that the optimal number of cuts identified by optimizing the surrogate model trained via random sampling is highly suboptimal. Although these results can be justified by the fact that only 110 data points were used for training, they also highlight the importance of selecting the proper data points to label in cases where obtaining the labels is computationally expensive.

### 6.5.4 Application of supervised learning

We also consider the case where the labeling cost is not significant. Specifically, for every data point in the pool $\mathcal{C}_p$ with features $s_i$ generated in the previous section, we solve the optimization problem and obtain the solution time $y_i$, leading to a data set $\mathcal{D} = \{s_i, y_i\}_{i=1}^{49000}$. We use this data set for training three surrogate modes; a decision tree, a random forest, and a neural network using scikit-learn [232]. For the Decision Tree and the Random Forest, we used the default values of the hyperparameters. The neural network had 3 layers with 150 neurons, the activation function was tanh, the learning rate was equal to $10^{-4}$, and the regularization parameter $\alpha$ was set equal to 0.01. Note that in this case, we do not train Gaussian Processes since the dataset has 49000 data points and the Gaussian processes have cubic complexity on the number of samples.

Once the surrogate models were trained, we generate 100 random disturbances that change simultaneously the demands and the inlet concentration as in the previous section. These disturbances are different than the disturbances generated in the previous section. We solve the optimization problem for every disturbance by initializing the Generalized Benders Decomposition algorithm using the number of cuts suggested by

Table 6.1: Computational time for the proposed approach for different surrogate models. NC refers to solving the problem without the addition of cuts in the first iteration.

| Solution statistics | Initalization strategy | | | | | |
|---|---|---|---|---|---|---|
| | NC | GP-AL | GP | NN | RF | DT |
| Aver. CPU time | 13.7 | 4.31 | 6.22 | 7.67 | 6.34 | 9.11 |
| Aver. red. | - | 66.5 | 53.44 | 43.52 | 51.61 | 33.64 |
| Aver. fold red. | - | 3.33 | 2.49 | 2.18 | 2.38 | 1.75 |
| Max. red. (%) | - | 81.3 | 81.41 | 79.57 | 81.96 | 77.55 |
| Min. red. (%) | - | 0.09 | -2.66 | -0.02 | -26.47 | -18.82 |
| Max fold red. | - | 5.35 | 5.38 | 4.48 | 5.54 | 4.45 |
| Min fold red. | - | 1.00 | 0.97 | 0.99 | 0.79 | 0.84 |

Table 6.2: Distribution of the demand

| Product | Nominal value | Distribution (Uniform) | |
|---|---|---|---|
| | | low | high |
| 1 | 600 | -100 | 100 |
| 2 | 550 | -15 | 15 |
| 3 | 600 | -30 | 30 |
| 4 | 1200 | -20 | 20 |
| 5 | 2000 | -400 | 400 |

optimizing the different surrogate models.

The total CPU time for the different disturbances is presented in Fig. 6.4 and the solution time statistics in Table 6.3. From the results, we observe that the average total CPU time without the addition of cuts (No cuts) is 14.7 seconds. The selection of the optimal number of cuts to add to the master problem leads to a 70% reduction in CPU time. From the three surrogate models, the neural network shows the maximum improvement in total CPU time, although the average reduction is similar for all surrogate models. Furthermore, the minimum reduction is positive for all models, indicating that the solution time with the proposed initialization is lower than the original implementation of the algorithm without cuts. Finally, the time presetned in Table 6.3 is the total time required to determine the optimal number of cuts and solve the problem. For case study considered, the time to determine the optimal number of cuts is in the order of $10^{-2}$ seconds for the decision tree and the neural network and in the order of $10^{-1}$ seconds for the random forest (see Fig. 6.5). Thus, the process of determining the number of cuts and adding them is significantly smaller than the solution time.

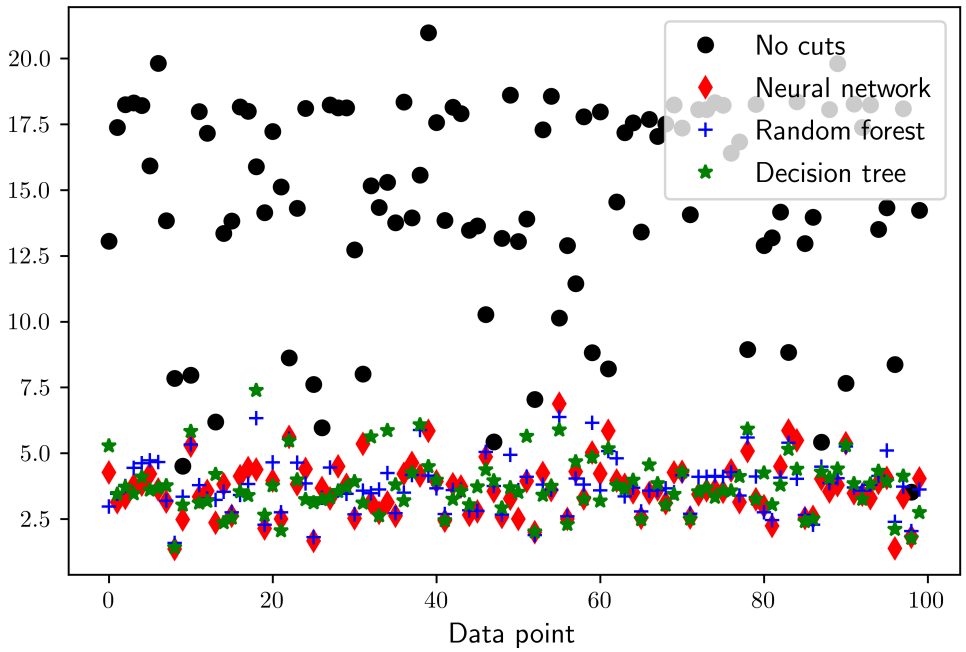


Figure 6.4: Solution time of the proposed approach with different surrogate models for 49000 training data points.

## 6.6 Conclusions and discusion

The repeated solution of large-scale decision-making problems arises frequently in the operation of process systems. The efficient solution of such problems with monolithic optimization algorithms can be challenging, especially in online settings. Although decomposition-based solution methods have been widely used to solve large-scale optimization problems, their off-the-shelf implementation is nontrivial. In this work, we proposed a machine learning-based approach for optimally initializing cutting plane-based decomposition-based solution algorithms, such as Generalized Benders. We use active learning to guide the generation of labeled data for learning a surrogate model that predicts the solution time of Generalized Benders Decomposition for a given problem and the initial set of cuts added in the master problem. The proposed approach is applied to a case study on the solution of mixed integer economic model predictive control problems that arise in the real-time operation of process systems. The numerical results show that the optimal initialization of the algorithm can significantly reduce the

Table 6.3: Computational time for the proposed approach for different surrogate models trained via supervised learning

| Solution statistics | Initalization strategy | | | |
|---|---|---|---|---|
| | No cuts | Neural networks | Random forests | Decision trees |
| Average CPU time (sec) | 14.7 | 3.63 | 3.78 | 3.74 |
| Average reduction (%) | - | 71.71 | 70.50 | 70.53 |
| Average fold reduction | - | 4.23 | 4.01 | 4.10 |
| Max. reduction (%) | - | 84.85 | 83.88 | 86.40 |
| Min. reduction (%) | - | 25.66 | 17.30 | 21.13 |
| Max fold reduction | - | 6.60 | 6.20 | 7.35 |
| Min fold reduction | - | 1.34 | 1.20 | 1.26 |

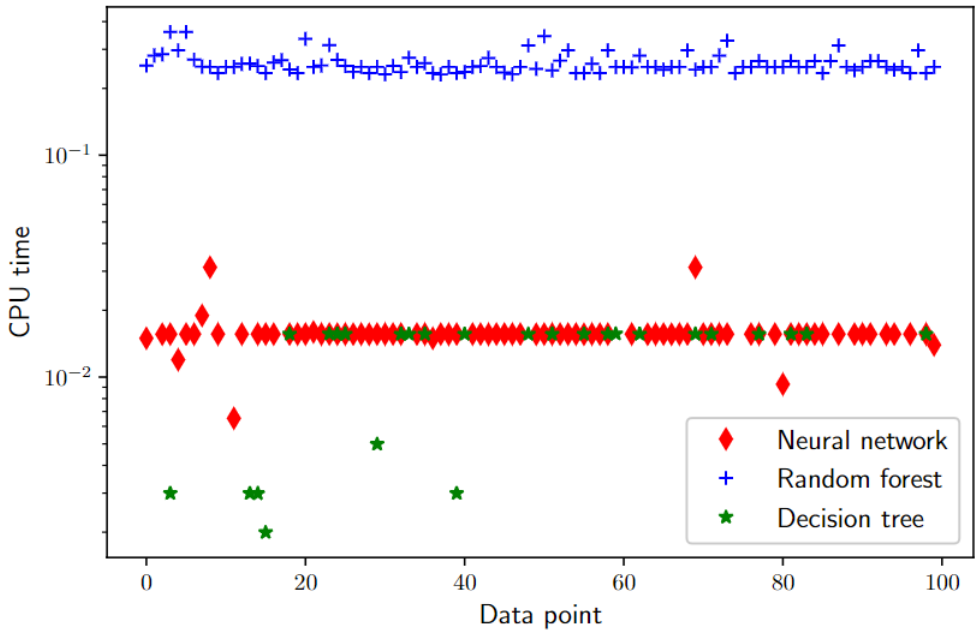


Figure 6.5: CPU time to determine the optimal number of cuts to add for the different surrogate models.

solution time up to 70 % and the computation of the optimal number of cuts using the learn surrogate model does not incur additional computational cost. Finally, these results highlight the ability of active learning to guide the data generation process for cases where obtaining the solution time is computationally expensive.

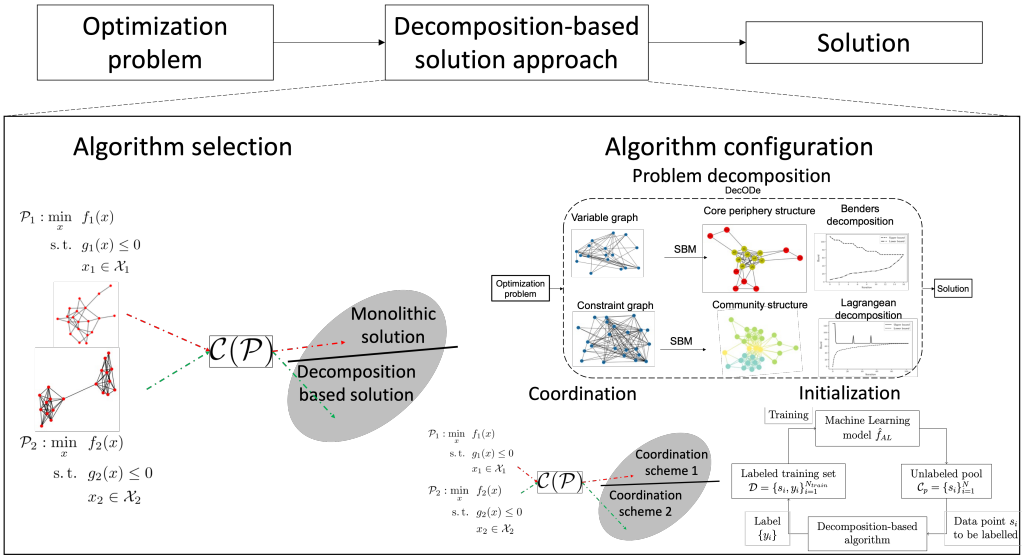


Figure 6.6: Learning to Decompose (L2D) framework for automated decomposition-based solution algorithm selection and configuration via artificial intelligence and network science

**Remark 6.1.** In this paper, we used a pool-based active learning approach with uncertainty-based sampling. However different active learning paradigms have been proposed. These paradigms can potentially be exploited for learning how to initialize decomposition-based and monolithic-based solution algorithms for different applications. For example, stream-based selective sampling can be used for learning online the surrogate models for the performance function, thus enabling the online learning of the optimal initialization.

In general, the solution of an optimization problem with decomposition-based solution methods poses four questions; (1) Should a decomposition-based method be selected over a monolithic one, (2) how to decompose the optimization problem, (3) Which coordination scheme should be used, and (4) how to initialize the algorithm? The results presented in this paper and in [205] in tandem with our previous work on learning the underlying structure of an optimization problem [6, 207, 199] show that artificial intelligence tools in combination with network science can be used to create an automated framework for decomposition-based solution algorithm selection and configuration, called Learning to Decompose (L2D).

The overall framework is presented in Fig. 6.6 and has two parts. The first focuses on algorithm selection, where given an optimization problem graph classification techniques can be used to determine whether, and potentially which, decomposition-based solution algorithm should be used. In the second part, the configuration of the selected decomposition-based solution algorithm is considered. The configuration has three aspects, problem decomposition, coordination, and initialization. For the problem decomposition, structure detection algorithms can be implemented to learn the underlying structure of the problem and use it as the basis for the application of the decomposition-based solution algorithm using DecODe [207].

Regarding coordination, although it was not considered in this paper or in [205], it is an important aspect for decomposition-based solution algorithms. For hierarchical decomposition-based methods, the coordination is done via cuts and the different coordination schemes correspond to different cut generation and management techniques. For distributed algorithms, the coordination considers the different methods to update the dual variables or Lagranean multiples. Consider for example the case of Lagrangean decomposition, where the dual variables can be updated using subgradient, cutting plane, bundle, and trust region methods [69]. Determining which coordination scheme is the best, especially for mixed integer nonlinear programming problems is not obvious. ML tools such as classification and regression, possibly coupled with geometric deep learning, might aid the selection of the coordination scheme as presented in Fig. 6.6. Finally, for the initialization, supervised learning can be used to learn surrogate models for the computational performance of the algorithm which can subsequently be used to optimally initialize the algorithm. Finally, these steps can either be performed sequentially given an optimization problem or can be performed individually.

This framework, and any ML-based approach, requires data for learning the parameters of the surrogate models used in the different tasks. Although multiple libraries of optimization problems exist [1, 2, 109], these are unlabeled data since for a given problem the solution time for a given algorithm and a given configuration is not known. Building a library where not only the optimization problem but also the solution time, and possibly other information regarding the solution, is stored would significantly reduce the required time for building such an ML-based approach for decomposition-based solution algorithms. Finally, the computational effort required to obtain the labels can be reduced either via active learning or semi-supervised learning [18].

# Part II
# From structure detection to improved computational performance

# Chapter 7
# A multicut Generalized Benders Decomposition approach for the integration of process operations and dynamic optimization for continuous systems

## 7.1 Introduction

The optimal operation of process systems depends on the solution of a wide class of optimization problems which are typically considered independently. However, fast changing economic environments render this approach suboptimal and motivate the integration of different decision levels [114, 75]. A typical example is the integration of process operations, i.e. production planning and scheduling, with dynamic optimization [17, 67, 82]. In these problems, production decisions such as execution of a task and allocation of resources are made simultaneously with optimal control decisions leading to increased profitability.

In order to achieve this integration two solution approaches have been proposed in the literature [17]. In the "top-down" approach, the dynamic behavior of the system is incorporated in the planning/scheduling problem, the problem is solved once and the results provide the production sequence which is the set point for the control level [97, 222, 125]. In the "bottom up" approach the integrated problem is solved in a rolling horizon manner resulting in a closed loop implementation [305, 229, 66, 249, 54].

We can argue that the main limitation in both approaches is the solution of the resulting optimization problem. The main challenges arise due to the inherently nonlinear behavior of most process systems, which in conjunction with the multiple time scales of the different decision making problems lead to large scale Mixed Integer Nonlinear Programs (MINLP). The monolithic solution of such problems is challenging due to the presence of continuous and discrete variables which are coupled through nonlinear constraints [25]. In order to improve the tractability of these problems two paths can be followed. In the first, the optimization problem is simplified using surrogate models, typically approximating the nonlinear dynamic behavior of the system [230, 306, 52, 65, 264, 57]. The alternative approach is to use decomposition based

solution algorithms and exploit the structure of the full optimization problem. Typical examples of this approach include the application of Lagrangean [275, 209] and Benders [221, 63, 64] decomposition. Although these algorithms can potentially reduce the computational time, their application is challenging too. First, a decomposition of the optimization problem itself is necessary. Its structure however, as it relates to the requisite solution algorithm, is not always evident. In recent research in our group we have proposed the application of Stochastic Blockmodeling (SBM) and Bayesian inference as a tool to learn the underlying structure of the problem [207, 199]. In this approach the optimization problem is represented as a graph, and application of statistical inference allows learning the structure of the problem which can guide us towards the selection of the most appropriate decomposition based solution algorithm.

The second issue is related to the convergence of decomposition based solution algorithms which depends strongly on the problem formulation. For the integration of planning, scheduling, and dynamic optimization for continuous systems, two problem formulations have been proposed: one based on slots [125] and another one based on the Traveling Salesman Problem [57]. In this work we will focus on the slot based formulation, where the planning horizon is discretized into periods, and each period is discretized into slots. The values of the state and manipulated variables for each slot and period depend on the production sequence and the transition time. We have recently shown that application of *nested* SBM [199] can reveal the underlying structure of such a problem and can be used as the basis for the application of Generalized Benders Decomposition (GBD) [199, 108]. However, global optimality can not be guaranteed due to the nonconvexity of the dynamic optimization problems that consider the dynamic transition of the system between the different products.

In this work, we propose a new formulation of the integrated problem for single stage continuous processes. Specifically, motivated by the formulation of the integrated scheduling and dynamic optimization problem for batch systems [63], we consider all the transitions between the products for all the slots and periods simultaneously. We analyze the structure of the problem using SBM and Bayesian inference. Based on the inference results, we find that the planning/scheduling and dynamic optimization constraints are coupled only through the transition times. Therefore, the cost associated with the dynamic optimization problem for each transition, slot, and period depends only on the transition time and can be replaced by its value function in the objective function. This structure lends itself to the application of multicut Generalized Benders decomposition,

which solves the problem in reduced computational time compared to other formulations of the problem and decomposition based solution algorithms [125, 199]. In order to reduce further the computational time we propose a hybrid multicut Generalized Benders decomposition algorithm where the cut associated with one transition in one slot and period is added for all slots and periods. We show that this approach leads to further reduction in computational time without affecting the solution quality. The rest of the document is organized as follows: In Section 7.2 we present the integrated optimization problem, in Section 7.3 we analyze the structure of the integrated problem and in Section 7.4 we present the decomposition based solution algorithms. Finally in Sections 7.5,7.6 we analyze the performance of the proposed algorithms.

## 7.2 Problem formulation

### 7.2.1 Production planning and scheduling

In this section we will present the planning and scheduling model for a single stage single line continuous process proposed in [84]. We assume that $N_p$ ($i = \{1, .., N_p\}$) products must be produced over a planning horizon which is discretized into $N_{pr}$ ($p = \{1, .., N_{pr}\}$) periods, which are further discretized into $N_s$ ($k = \{1, .., N_s\}$) slots. First we define a binary variable $W_{ikp}$ which is equal to 1 if product $i$ is produced at slot $k$ in period $p$ and zero otherwise. We also define the variable $Z_{ijkp} \in \{0, 1\}$ which is equal to 1 if a product $i$ is followed by product $j$ in slot $k$ in period $p$, and the variable $Zp_{ijp}$ which is equal to one if transition occurs between product $i$ and $j$ between time periods. At each time slot only one product can be produced, which is enforced with the following constraints:

$$\sum_i W_{ikp} = 1 \ \forall k, p. \tag{7.1}$$

Based on the sequence of the products at the end of each slot, either a transition occurs from product $i$ to $j$ or the same product is produced in the next slot. The transition between products for every slot and period is modeled though the following equations:

$$\begin{aligned} &Z_{ijkp} \geq W_{ikp} + W_{j,k+1,p} - 1 \ \forall i, j, k \neq N_s, p \\ &Zp_{ijkp} \geq W_{iN_sp} + W_{j,1,p+1} - 1 \ \forall i, j, p, \end{aligned} \tag{7.2}$$

where the first constraint considers the transitions between the slots in a period, and the second constraint considers the transitions between the periods. Due to the possible transitions, each slot is composed of a production and a transition regime. The production time of product $i$ in slot $k$ in period $p$ is $\Theta_{ikp}$ and the transition time in slot $k$ in period $p$ is $\check{\theta}^t_{kp}$. The timing constraints are the following:

$$\begin{aligned}
&T^s_{1,1} = 0 \\
&T^e_{k,p} = T^s_{k,p} + \sum_i \Theta_{ikp} + \check{\theta}^t_{kp} \quad \forall k, p \\
&T^s_{k+1,p} = T^e_{k,p} \quad \forall k \neq N_s, p \\
&T^s_{1,p+1} = T^e_{N_s,p} \quad \forall k, p \neq N_{per} \\
&\Theta_{ikp} \leq W_{ikp} H_p \quad \forall i, k, p \\
&\hat{\Theta}_{ip} = \sum_k \Theta_{ikp} \quad \forall i, p \\
&T^e_{k,p} \leq p H_p \quad \forall k, p,
\end{aligned} \tag{7.3}$$

where $T^s_{kp}$ is the starting tine of slot $k$ in period $p$, $T^e_{kp}$ is the ending time in slot $k$ and period $p$, $H_p$ is the duration of period $p$ and $\hat{\Theta}_{ip}$ is the production time of product $i$ in period $p$. We assume that the demand of product $i$ in period $p$ ($d_{ip}$) must be satisfied in the end of the period. The production rate of product $i$ is $r_i$, the amount of product $i$ produced in slot $k$ at period $p$ is $\hat{q}_{ikp}$ and the amount of product $i$ produced in period $p$ is $q_{ip}$. The production and inventory constraints are:

$$\begin{aligned}
&\hat{q}_{ikp} = r_i \Theta_{ikp} \quad \forall i, k, p \\
&q_{ip} = \sum_k \hat{q}_{ikp} \quad \forall i, p \\
&I_{ip} = I_{ip-1} + q_{ip} - S_{ip} \quad \forall i, p \\
&A_{ip} = H_p (I_{ip-1} - S_{ip-1}) + q_{ip} H_p \quad \forall i, p \\
&S_{ip} \geq d_{ip} \quad \forall i, p,
\end{aligned} \tag{7.4}$$

where $I_{ip}$ is the inventory of product $i$ in period $p$, $A_{ip}$ is the linear overestimation of the integral of inventory, $S_{ip}$ is the amount of product $i$ sold in period $p$. Finally, the

following symmetry breaking constraints are included:

$$
\begin{aligned}
& Y_{ip} \geq W_{ikp} \quad \forall i,k,p \\
& Y_{ip} \leq N_{ip} \leq \bar{N} Y_{ip} \quad \forall i,p \\
& N_{ip} \geq N - \left( \sum_i Y_{ip} - 1 \right) - M(1 - W_{i1p}) \quad \forall i,p \\
& N_{ip} \leq N - \left( \sum_i Y_{ip} - 1 \right) + M(1 - W_{i1p}) \quad \forall i,p \\
& N_{ip} = \sum_k W_{ikp} \quad \forall i,p
\end{aligned}
\tag{7.5}
$$

where $Y_{ip} \in \{0,1\}$ is equal to 1 if product $i$ is assigned in period $p$ and zero otherwise, $\bar{N}$ is the number of slots, $M$ is a parameter, and $N_{ip}$ is equal to the number of slots that product $i$ is manufactured in period $p$. The first constraint (in Eq. 7.5) ensures that every product is assigned to at least one slot every period, the second constraint guarantees that if a product is manufactured in a period then at least one slot must be used, and the last constraint enforces that a product is manufactured in consecutive slots if necessary. We refer the reader to [84] for a detailed explanation of these constraints.

### 7.2.2 Dynamic model

We assume that the dynamic behavior of the system is described by a system of ordinary differential equations

$$\dot{x}(t) = F(x,u), \tag{7.6}$$

where $x \in \mathbb{R}^n$ are the states of the system, $u \in \mathbb{R}^m$ are the manipulated variables and $F : \mathbb{R}^{n+m} \rightarrow \mathbb{R}^n$ are vector functions. The differential equations are discretized using the method of orthogonal collocation on finite elements using $N_{fe}$ ($f = \{1,..,N_{fe}\}$) finite elements and $N_c$ ($c = \{1,..,N_c\}$) collocation points. In this work, we will consider simultaneously all the possible transitions and will define the variables $x^n_{ijfckp}$ and $u^m_{ijfckp}$ as the values of the $n^{th}$ state and $m^{th}$ manipulated variable for a transition from product $i$ to product $j$ at finite element $f$, collocation point $c$, slot $k$ and period $p$. We will also define $\theta_{ijkp}$ as the transition time for the transition from product $i$ to $j$ in slot $k$ and

period $p$. The discretized differential equations are:

$$
\begin{aligned}
& x^n_{ijfckp} = x0^n_{ijfkp} + h^{fe}_{kp} \sum_{m=1}^{N_{cp}} \Omega_{mc} \dot{x}^n_{ijfmkp} \quad \forall n,i,j,f,c,k,p \\
& h^{fe}_{ijkp} = \frac{\theta_{ijkp}}{N_{fe}} \quad \forall i,j,k,p \\
& x0^n_{ijfkp} = x0^n_{ij,f-1kp} + h^{fe}_{ijkp} \sum_{m=1}^{N_{cp}} \Omega_{mc} \dot{x}^n_{ij,f-1,mkp} \quad \forall n,i,j,f \geq 2,c,k,p \\
& \dot{x}^n_{ijfckp} = F(x^n_{ijfckp}, u^m_{ijfckp}) \quad \forall n,i,j,f,c,k,p \\
& t^d_{ijfckp} = h^{fe}_{ijkp}(f-1+\gamma_c) \quad \forall i,j,f,c,k,p \\
& x0_{ij1kp} = x^{ss}_i \quad \forall i,j,k,p \\
& x_{ijN_{fe}N_{cp}kp} = x^{ss}_j \quad \forall i,j,k,p \\
& u_{ij11kp} = u^{ss}_i \quad \forall i,j,k,p \\
& u_{ijN_{fe}N_{cp}kp} = u^{ss}_j \quad \forall i,j,k,p,
\end{aligned}
\tag{7.7}
$$

where $x^{ss}_i, u^{ss}_i$ are the steady state values of the state and manipulated variables for product $i$, $\Omega$ is the collocation matrix, and $\gamma$ are the Radau roots. We note that in this formulation, the values of the state and manipulated variables at the beginning and end of each slot and period do not depend on the binary variables $W_{ikp}$ or $Z_{ijkp}$. This is different from [125, 57] where the values of the states/manipulated variables at the beginning and end of each slot and period depend on the values of the binary variables.

### 7.2.3 Integrated problem

The objective function of the integrated optimization problem is:

$$
\begin{aligned}
\Phi = & \sum_{i,p} \left( P_{ip} S_{ip} - C^{oper}_{ip} q_{ip} - C^{inv} A_{ip} \right) \\
& - \sum_{i,j,k,p} C^{trans}_{ij} Z_{ijkp} - \sum_{i,j,p} C^{trans}_{ij} Zp_{ijp} \\
& - \sum_{i,j,p,k} Z_{ijkp} \left( \alpha_u \sum_{f,c} N^{-1}_{fe} t^d_{ijfckp} \Omega_{c,N_{cp}} (u_{ijfckp} - u^{ss}_j)^2 \right) \\
& - \sum_{i,j,p} Zp_{ijkp} \left( \alpha_u \sum_{f,c} N^{-1}_{fe} t^d_{ijfcN_sp} \Omega_{c,N_{cp}} (u_{ijfcN_sp} - u^{ss}_j)^2 \right),
\end{aligned}
\tag{7.8}
$$

where $P_{ip}$ is the price of product $i$ in period $p$, $C_{ip}^{oper}$ is the operating cost of product $i$ in period $p$, $C^{inv}$ is the inventory cost, $C_{ij}^{trans}$ is the transition cost from product $i$ to $j$, and $a_u$ is a weight coefficient. The first term represents the sales, the second and third terms are related to the operating and inventory costs, and the fourth and fifth terms represent the fixed transition cost between the slots and periods. The last two terms represent the transition cost. In these terms, the cost of a transition from product $i$ to $j$ in slot $k$ and period $p$ is multiplied by $Z_{ijkp}$ (and by $Zp_{ijp}$ if we consider the transition between the time periods). Therefore, if a transition does not occur, i.e. $Z_{ijkp} = 0$ or $Zp_{ijp} = 0$, then the cost does not contribute to the objective. Finally, the transition time for each slot and period depends on the transitions that occur and the following equations are added:

$$\begin{aligned} \check{\theta}_{kp}^{t} &= \sum_{i=1}^{N_p}\sum_{i=1}^{N_p} \theta_{ijkp} Z_{ijkp} \ \forall k, k \neq N_s, p \\ \check{\theta}_{N_sp}^{t} &= \sum_{i=1}^{N_p}\sum_{i=1}^{N_p} \theta_{ijN_sp} Zp_{ijp} \ \forall p, p \neq N_{per} \\ \theta_{ijkp} &\geq \theta_{ij}^{min} \ \forall i, j, k, p, \end{aligned} \tag{7.9}$$

where $\theta_{ij}^{min}$ is the minimum transition time for a transition between product $i$ and $j$. The integrated optimization problem is:

$$\begin{aligned} &\text{maximize} \quad \Phi \\ &\text{subject to} \quad \text{Eq. 7.1, 7.2, 7.3, 7.4, 7.5, 7.7, 7.9} \end{aligned} \tag{7.10}$$

This is the general problem formulation for the integration of planning, scheduling, and dynamic optimization for a single stage single unit system. In this formulation, each slot is composed of a production and a transition regime. Therefore, if the number of slots is equal to the number of products, in the last slot of the last period no transition occurs. Hence the variables and constraints associated with all the transitions for this slot can be removed from the optimization problem.

## 7.3 Problem decomposition

In this section we will analyze the structure of the integrated problem using SBM and Bayesian inference. We refer the reader to the Supplementary material for an

introduction to SBM and to [207, 199] for a detailed explanation on the application of this approach to optimization problems. For illustration, we consider an isothermal continuous stirred tank reactor where an irreversible reaction occurs ($A \rightarrow 3B$). The dynamic behavior of the system is described by the following equation [125]:

$$\frac{dc(t)}{dt} = \frac{Q(t)}{V}(c_{feed} - c(t)) - kc(t)^3, \tag{7.11}$$

where $c$ is the concentration of the reactant, $Q$ is the inlet flowrate (manipulated variable) and $V = 5000\ L, c_{feed} = 1\ mol/L, k = 2\ L^2/(hr\ mol^2)$ are the reactor volume, inlet concentration and reaction rate constant, respectively. In order to keep the graph simple, we will assume that two products must be produced in three planning periods and the dynamic model is discretized using 20 finite elements with 3 collocation points. We apply degree corrected SBM in the constraint graph of the problem using graph-tool [235]. In this graph, the nodes are the constraints of the problem and the edges are the variables that couple two constraints. Based on the inference results (Fig. 7.1), twenty five blocks are identified. The planning/scheduling constraints are assigned in one block (nodes in the middle of the graph), and the constraints that are associated with the dynamic behavior of the system for each transition, slot, and period are assigned into different blocks. These blocks are connected with the block in the center of the graph through one edge, the coupling variables. In this case the coupling variables are the transition times $\theta_{ijkp}$. This *hybrid core community structure* in the graph is also manifested in the L shape of the $\omega$ (block density) matrix inferred for the SBM:

$$\omega = \begin{bmatrix} 1392 & 1 & 1 & \dots & 1 \\ 1 & 6794 & 0 & \dots & 0 \\ 1 & 0 & 6794 & \dots & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ 1 & 0 & \dots & \dots & 6794 \end{bmatrix} \in \mathbb{R}^{25\times 25},$$

where the $\omega_{ij}$ entry of this matrix is equal to the number of edges between the nodes in block $i$ and the nodes in block $j$. The planning/scheduling constraints form the core, which is connected with all the communities, i.e. the constraints of the dynamic optimization problems, through the variables $\theta_{ijkp}$.

We note that although the structure of this problem is the same as the one identified using nested SBM in our previous work [199] based on the problem formulation proposed

Figure 7.1: Inference results on the constraint graph of the integrated optimization problem

in [125], the coupling variables differ. In [199] the coupling variables are the transition time, initial and final states, and manipulated variables. In this case the coupling variables are only the transition times $\theta_{ijkp}$. This coupling of the variables/constraints can be attributed to the modeling of the transitions. The state and manipulated variables depend on the transition times through the discretization of the differential equations.

## 7.4 Decomposition based solution algorithm

### 7.4.1 Problem reformulation based on the identified structure from SBM

Given the structure of the problem the variables can be decomposed into three sets. The first contains the variables that are associated with planning and scheduling decisions, the second set contains the variables that are associated with transitions between products, and the last set contains the coupling variables (transition times) that couple the planning/scheduling decisions with the dynamic optimization decisions. Given this partition of the variables, if we fix the planning/scheduling and the coupling variables, then the dynamic optimization problems for the transitions are independent and depend only on the transition time $\theta_{ijk}$. The dynamic optimization problem for a transition

from product $i$ to $j$ in slot $k$ and period $p$ is:

$$
\begin{aligned}
\text{maximize} \quad & -\alpha_u \sum_{f=1}^{N_{fe}} \sum_{c=1}^{N_{cp}} N_{fe}^{-1} t^d_{ijfckp} \Omega_{c,N_{cp}} (u_{ijfckp} - u_j^{ss})^2 \\
\text{subject to} \quad & x^n_{ijfckp} = x0^n_{ijfkp} + h^{fe}_{kp} \sum_{m=1}^{N_{cp}} \Omega_{mc} \dot{x}^n_{ijfmkp} \quad \forall n, f, c \\
& h^{fe}_{ijkp} = \frac{\hat{\theta}_{ijkp}}{N_{fe}} \\
& x0^n_{ijfkp} = x0^n_{ij,f-1kp} + h^{fe}_{ijkp} \sum_{m=1}^{N_{cp}} \Omega_{mc} \dot{x}^n_{ij,f-1,mkp} \quad \forall n, f \geq 2, c \\
& \dot{x}^n_{ijfckp} = f(x^n_{ijfckp}, u^m_{ijfckp}) \quad \forall n, f, c \\
& t^d_{ijfckp} = h^{fe}_{ijkp}(f - 1 + \gamma_c) \quad \forall f, c \\
& x0_{ij1kp} = x_i^{ss} \\
& x_{ijN_{fe}N_{cp}kp} = x_j^{ss} \\
& u_{ij11kp} = u_i^{ss} \\
& u_{ijN_{fe}N_{cp}kp} = u_j^{ss} \\
& \hat{\theta}_{ijkp} = \theta_{ijkp}.
\end{aligned}
\tag{7.12}
$$

We will write this problem as:

$$
\begin{aligned}
\text{maximize} \quad & -f^{dyn}_{ijkp} \\
\text{subject to} \quad & g^{dyn}_{ijkp} \leq 0 \\
& \hat{\theta}_{ijkp} = \theta_{ijkp}
\end{aligned}
\tag{7.13}
$$

which is equivalent to minimizing $f^{dyn}_{ijkp}$ subject to $g^{dyn}_{ijkp} \leq 0$ and $\hat{\theta}_{ijkp} = \theta_{ijkp}$. The solution of this problem depends on the value of the transition time $\theta_{ijkp}$. We define as $\phi_{ijkp}$ the value function of a transition from product $i$ to $j$ in slot $k$ and period $p$, which is equal to the optimal value of the objective function of the following optimization problem:

$$
\begin{aligned}
\text{minimize} \quad & f^{dyn}_{ijkp} \\
\text{subject to} \quad & g^{dyn}_{ijkp} \leq 0 \\
& \hat{\theta}_{ijkp} = \theta_{ijkp} \quad : \lambda_{ijkp}
\end{aligned}
\tag{7.14}
$$

where $\lambda_{ijkp}$ is the Lagrange multiplier for the equality constraint $\hat{\theta}_{ijkp} = \theta_{ijkp}$. The value function depends only on the transition time, i.e. the coupling variable.

The objective function of the integrated problem can be decomposed into two parts. The first contains the cost associated with the planning/ scheduling variables ($\Phi_1$) and the second part considers the transition costs ($\Phi_2$):

$$\begin{aligned}\Phi_1 = &\sum_{i,p}\left(P_{ip}S_{ip} - C^{oper}_{ip}q_{ip} - C^{inv}A_{ip}\right)\\ &- \sum_{i,j,k,p} C^{trans}_{ij}Z_{ijkp} - \sum_{i,j,p} C^{trans}_{ij}Zp_{ijp}\end{aligned}$$

$$\begin{aligned}\Phi_2 = &\sum_{i,j,k,p} Z_{ijkp}\left(\alpha_u \sum_{f,c} N_{fe}^{-1} t^d_{ijfckp}\Omega_{c,N_{cp}}(u_{ijfckp} - u^{ss}_j)^2\right)\\ &+ \sum_{i,j,p} Zp_{ijkp}\left(\alpha_u \sum_{f,c} N_{fe}^{-1} t^d_{ijfcN_sp}\Omega_{c,N_{cp}}(u_{ijfcN_sp} - u^{ss}_j)^2\right).\end{aligned}$$

Based on the value function of the dynamic optimization problem, the integrated problem is written as follows:

$$\begin{aligned}&\text{maximize} \quad \Phi_1 - \sum_{ijkp} Z_{ijkp}\phi_{ijkp}(\theta_{ijkp}) - \sum_{ijp} Zp_{ijp}\phi_{ijN_sp}(\theta_{ijN_sp})\\ &\text{subject to} \quad \text{Equations 7.1, 7.2, 7.3, 7.4, 7.5,7.9.}\end{aligned} \tag{7.15}$$

This problem is equivalent to the original integrated problem (Eq. 7.10) since $\phi_{ijkp}$ is only a function of the transition time $\theta_{ijkp}$ as identified by the application of SBM. Therefore, if a transition does not occur, i.e. $Z_{ijkp} = 0$, then $Z_{ijkp}\phi_{ijkp}(\theta_{ijkp}) = 0$ even if $\phi_{ijkp}(\theta_{ijkp}) \neq 0$, so the cost associated with this transition does not affect the profit. If the transition occurs, i.e. $Z_{ijkp} = 1$, then the cost, which is equal to $\phi_{ijkp}(\theta_{ijkp})$, affects the profit.

### 7.4.2 Solution algorithm

The above problem can not be solved directly since the value functions are not known explicitly. We will follow a cutting plane method to solve the problem. Specifically, we will approximate the value functions with hyperplanes. From [107], it is known that if the value function $\phi_{ijkp}$ is convex and we solve the problem in Eq. 7.14 for $\bar{\theta}_{ijkp} = \theta_{ijkp}$,

we can outer approximate $\phi_{ijkp}$ as follows:

$$\phi_{ijkp}(\theta_{ijkp}) \geq \phi_{ijkp}(\bar{\theta}_{ijkp}) + \partial\phi_{ijkp}(\bar{\theta}_{ijkp})(\theta_{ijkp} - \bar{\theta}_{ijkp}) \;\; \forall \theta_{ijkp} \geq \theta_{ij}^{min} \tag{7.16}$$

where $\partial\phi(\bar{\theta}_{ijkp})$ is the subgradient of the value function for $\theta_{ijkp} = \bar{\theta}_{ijkp}$ and is equal to the negative of the optimal Lagrange multiplier $\lambda_{ijkp}$ for the equality constraint $\theta_{ijkp} = \bar{\theta}_{ijkp}$.

Given this approximation, if we solve the dynamic optimization problems for different values of $\theta_{ijkp}$ we can approximate the function $\phi_{ijkp}$ and the problem:

$$\underset{\theta_{ijkp}}{\text{minimize}} \;\; \phi_{ijkp}(\theta_{ijkp}) \tag{7.17}$$

is equivalent to [105, 106]:

$$\begin{aligned} \underset{\theta_{ijkp}}{\text{minimize}} \;\; & \eta_{ijkp} \\ \text{subject to} \;\; & \eta_{ijkp} \geq \phi_{ijkp}^{v}(\bar{\theta}_{ijkp}^{v}) - \lambda^{v}(\theta_{ijkp} - \bar{\theta}_{ijkp}^{v}) \forall v = 1, ..., \mathcal{V} \end{aligned} \tag{7.18}$$

where $\mathcal{V} = \{v_1, v_2, .., v_n\}$ is the number of points used to approximate the value function and the overbar denotes a fixed value. The integrated optimization problem can be written as:

$$\begin{aligned} \text{maximize} \;\; & \Phi_1 - \sum_{ijkp} Z_{ijkp}\eta_{ijkp} - \sum_{ijp} Zp_{ijp}\eta_{ijN_sp} \\ \text{subject to} \;\; & \text{Equations 7.1, 7.2, 7.3, 7.4, 7.5,7.9} \\ & \eta_{ijkp} \geq \phi_{ijkp}^{v} - \lambda_{ijkp}^{v}(\theta_{ijkp} - \bar{\theta}_{ijkp}^{v}) \;\; \forall i, j, k, p, v \in \mathcal{V} \end{aligned} \tag{7.19}$$

We note that the tangent approximation corresponds to the Benders cuts [108]. We will follow a Generalized Benders Decomposition approach to solve this problem. The master problem is the relaxed planning/scheduling problem and the subproblems are independent dynamic optimization problems for each transition, slot, and period. The dynamic optimization problems are solved only for the transitions that occur, i.e. $(i, j, k, p) \in \{(i, j, k, p) | Z_{ijkp} = 1\}, (i, j, p) \in \{(i, j, p) | Zp_{ijp} = 1\}$. Once the subproblems are solved, the following Benders cuts are added in the master problem:

$$\eta_{ijkp} \geq \phi_{ijkp} - \lambda_{ijkp}(\theta_{ijkp} - \bar{\theta}_{ijkp})$$

**Data:** Optimization problem
**Result:** Upper, lower bound and variable values
1 Set $UB = \infty, LB = -\infty$;
2 Set tolerance and optimality gap (tol);
3 Initialize the algorithm by fixing the production sequence and obtain the transitions that occur, transition times, Lagrangean multipliers $\lambda_{ijkp}$, $\phi_{ijkp}$ and add tangent approximation (Line 10);
4 **while** $(UB - LB)/LB \geq tol/100$ **do**
5 Solve the master problem (Eq. 7.19), obtain $UB$ and transitions;
6 Solve the dynamic optimization problems (Eq. 7.14) that correspond to $Z_{ijkp} = 1, Zp_{ijp} = 1$;
7 Obtain Lagrangean multipliers $\lambda_{ijkp}$, $\phi_{ijkp}$, $\Phi_1$,$\Phi_2$;
8 Update lower bound $LB = \max\{LB, \Phi_1 - \Phi_2\}$;
9 Add following tangent approximation in the relaxed problem:

$$\text{if } Z_{ijkp} = 1 \text{ add } \eta_{ijkp} \geq \phi_{ijkp} - \lambda_{ijkp}(\theta_{ijkp} - \bar{\theta}_{ijkp})$$
$$\text{if } Zp_{ijp} = 1 \text{ add } \eta_{ijN_sp} \geq \phi_{ijN_sp} - \lambda_{ijN_sp}(\theta_{ijN_sp} - \bar{\theta}_{ijN_sp})$$

10 **end**

**Algorithm 7.1:** Multicut Generalized Benders Decomposition based on the learnt structure

if $Z_{ijkp} = 1$ and

$$\eta_{ijN_sp} \geq \phi_{ijN_sp} - \lambda_{ijN_sp}(\theta_{ijN_s} - \bar{\theta}_{ijN_sp})$$

if $Zp_{ijp} = 1$, where the overbar denotes a fixed value. The steps that are followed to solve the problem are presented in Algorithm 7.1. In this approach at every iteration one cut is added for every transition that occurs, therefore multiple cuts are added per iteration. This approach is known in the literature as multicut Benders decomposition [41].

We can exploit further the structure of the problem to accelerate its solution. Specifically, since $\eta$ is defined for all $(ijkp)$, the approximation of the transition from $i$ to $j$ in slot 1 is also valid in all the other slots and periods. Therefore, we propose an accelerated hybrid multicut algorithm where in each iteration we add the constraints:

$$\eta_{ijk'p'} \geq \phi_{ijkp} - \lambda_{ijkp}(\theta_{ijk'p'} - \bar{\theta}_{ijkp}) \ \forall i, j, k' \in N_s, p' \in N_{per}$$

if $Z_{ijkp} = 1$ and

$$\eta_{ijk'p'} \geq \phi_{ijN_sp} - \lambda_{ijN_sp}(\theta_{ijk'p'} - \bar{\theta}_{ijN_sp}) \; \forall i, j, k' \in N_s, p' \in N_{per}$$

if $Zp_{ijp} = 1$, where the overbar denotes a fixed value. In this approach, at each iteration, for each transition that occurs multiple cuts are added. We note that the addition of multiple cuts per iteration is a common strategy in decomposition based solution methods for MINLP problems [269, 160]. In these cases the solution of the master problem provides a pool of solutions which can be used to solve multiple subproblems which lead to the addition of multiple cuts per iteration. In the proposed *hybrid* multicut GBD approach, the subproblems are solved based on the global solution of the master problem but the cut for a transition between product $i$ and $j$ in slot $k$ and period $p$ is used to approximate the value functions for the specific transition for all slots and periods. The steps that are followed to solve the problem are presented in Algorithm 7.2. For both algorithms (multicut and hybrid multicut), the relaxed problem is a Mixed Integer Nonlinear Program (MINLP) with bilinear terms in the objective function ($Z_{ijkp}\eta_{ijkp}$, $Zp_{ijp}\eta_{ijN_sp}$) and in Eq. 7.9 ($Z_{ijkp}\theta_{ijkp}$, $Zp_{ijp}\theta_{ijN_sp}$) solved with Gurobi [124]. We note that this problem can be transformed into a Mixed Integer Linear Program (MILP) by linearizing the bilinear terms. Specifically we can replace every bilinear term, e.g. $Z_{ijkp}\eta_{ijkp}$ with $\eta \in [\underline{\eta}_{ij}, \bar{\eta}_{ij}]$ as follows

$$\begin{aligned}
\min\{0, \underline{\eta}_{ij}\} \leq \delta_{ijkp} \leq \bar{\eta}_{ij} \\
\underline{\eta}_{ij} Z_{ijkp} \leq \delta_{ijkp} \leq \bar{\eta}_{ij} Z_{ijkp} \\
\eta_{ijkp} - (1 - Z_{ijkp})\bar{\eta}_{ij} \leq \delta_{ijkp} \leq \eta_{ijkp} - (1 - Z_{ijkp})\underline{\eta}_{ij} \\
\delta_{ijkp} \leq \eta_{ijkp} + (1 - Z_{ijkp})\underline{\eta}_{ij}.
\end{aligned} \tag{7.20}$$

The computation of $\underline{\eta}_{ij}, \bar{\eta}_{ij}$ is presented in the Supplementary material. Similarly we can linearize the $Z_{ijkp}\theta_{ijkp}$ bilinear terms. Although this leads to a MILP, we found that for small planning horizons solving the master problem with the bilinear terms (using Gurobi) is faster than solving the MILP model. Hence, in the first case study (Section 7.5) the master problem is a MINLP whereas in the second case study (Section 7.6) we use the MILP model. The subproblems are nonlinear optimization problems solved with IPOPT [285] and the values of the dual variables that are used are the ones returned by IPOPT. The algorithm is implemented in Python using Pyomo [129].

**Data:** Optimization problem
**Result:** Upper, lower bound and variable values
1 Set $UB = \infty, LB = -\infty$;
2 Set tolerance and optimality gap (tol);
3 Initialize the algorithm by fixing the production sequence and obtain the transitions that occur, transition times, Lagrangean multipliers $\lambda_{ijkp}$, $\phi_{ijkp}$ and add tangent approximation (Line 10);
4 **while** $(UB - LB)/LB \geq tol/100$ **do**
5 Solve the master problem (Eq. 7.19), obtain $UB$ and transitions;
6 Solve the dynamic optimization problems (Eq. 7.14) that correspond to $Z_{ijkp} = 1, Zp_{ijp} = 1$;
7 Obtain Lagrangean multipliers $\lambda_{ijkp}$, $\phi_{ijkp}$, $\Phi_1$,$\Phi_2$;
8 Update lower bound $LB = \max\{LB, \Phi_1 - \Phi_2\}$;
9 Add the following tangent approximation in the relaxed problem:;
10 **if** $Z_{ijkp} = 1$ **then**
11 add $\eta_{ijk'p'} \geq \phi_{ijkp} - \lambda_{ijkp}(\theta_{ijk'p'} - \bar{\theta}_{ijkp})\ \forall i, j, k' \in N_s, p' \in N_{per}$
12 **end**
13 **if** $Zp_{ijp} = 1$ **then**
14 add $\eta_{ijk'p'} \geq \phi_{ijN_sp} - \lambda_{ijN_sp}(\theta_{ijk'p'} - \bar{\theta}_{ijN_sp})\ \forall i, j, k' \in N_s, p' \in N_{per}$
15 **end**
16 **end**

**Algorithm 7.2:** Hybrid Multicut Generalized Benders Decomposition

**Remark 7.1.** In the proposed algorithms the hyperplanes that approximate a value function are valid under the assumption that the value function is convex. For the isothermal CSTR considered in Section 3, when four products can be produced, the value function for each transition from product 1 to the other products is shown in Figure 7.2 and the Benders cut for the transition from product 1 to product 2 is shown in Fig 7.3 (the steady state values of the concentration and inlet flowrate are presented in Table 7.1). From these figures we observe that the value functions are convex and the Benders cuts are valid understimators.

**Remark 7.2.** Given the convexity of the value functions, the proposed multicut and hybrid multicut algorithms can be used to solve the problem to global optimality since the Benders cuts are valid understimators of the value functions. However, the solution of the integrated problem to global optimality requires that the subproblems are also solved to global optimality. Although this can be achieved using global optimization

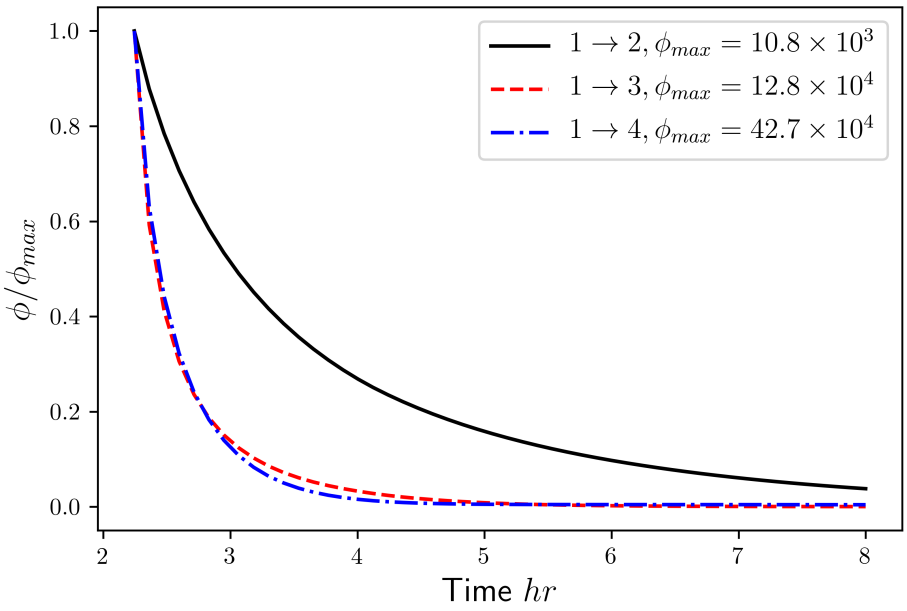


Figure 7.2: Transition cost for a transition from product 1 to products 2,3,4. The x axis is the transition time and the y axis the scaled cost. The steady state values of the state and manipulated variable are given in Table 7.1.

solvers like BARON [154], this can increase the CPU time in cases where the subproblems have a large number of variables/constraints and nonconvex terms. In the case studies presented in this paper we solve the subproblems with IPOPT and therefore global optimality cannot be guaranteed. We also note that in the general case, where the value functions are not convex the proposed algorithms can still be applied but global optimality cannot be guaranteed even if the subproblems are solved to global optimality.

**Remark 7.3.** The idea of replacing the transition cost with its value function has been previously pursued in literature. In [264], the dynamic optimization problem was solved for different values of the transition time and a flexible recipe approach was followed to solve the integrated planning, scheduling, and dynamic optimization problem. Similarly in [57] a surrogate model was used to approximate the transition cost. A similar approach was followed in [65], where a cooperative game theoretical approach was followed and the problem was formulated as a two level game. The first

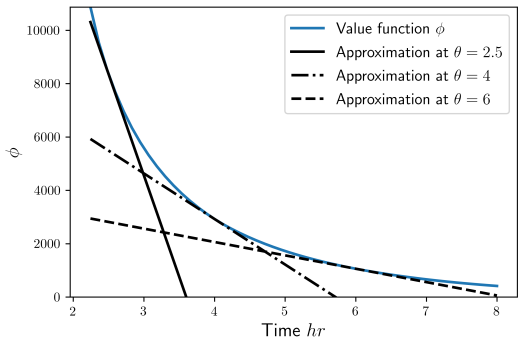


Figure 7.3: Approximation of the value function for three different values of $\theta$.

level agent solves the scheduling problem and the second level agent solves the dynamic optimization problem by considering only its objective value. The value function of the dynamic optimization problem was approximated using piece-wise linear functions. In the above approaches the transition cost and value of the objective function obtained are approximate. In our approach, the proposed algorithm provides the exact solution of the problem.

## 7.5 Case study 1: Isothermal CSTR

In the first case study we consider the isothermal CSTR whose dynamic behavior is described by Eq. 7.11. We assume that four products must be produced in two planning periods, each composed of four slots. The economic data of the problem, adapted from [125], can be found in Tables 7.1,7.2,7.3. We solve the problem with the proposed formulation and multicut and hybrid multicut decomposition based solution algorithms and compare them with the application of Generalized Benders Decomposition based on the structure found in [199] using the formulation of the integrated problem proposed in [125]. The formulation of the integrated problem and formulation of GBD [199] can be found in the Supplementary material. In this case study, we constrain the rate of change of the manipulated variable for the dynamic optimization problems through the

following constraints:

$$
\begin{aligned}
&u^m_{ijfckp} - u^m_{ijf-1,ckp} \leq U^{max}_m (t^d_{ijfckp} - t^d_{ijf-1,ckp}) \ \forall m, i, j, f \geq 1, c, k, p \\
&u^m_{ijfckp} - u^m_{ijfc-1,kp} \leq U^{max}_m (t^d_{ijfckp} - t^d_{ijfc-1,kp}) \ \forall m, i, j, f, c \geq 1, k, p \\
&u^m_{ijfckp} - u^m_{ijf-1,ckp} \geq U^{min}_m (t^d_{ijfckp} - t^d_{ijf-1,ckp}) \ \forall m, i, j, f \geq 1, c, k, p \\
&u^m_{ijfckp} - u^m_{ijfc-1,kp} \geq U^{min}_m (t^d_{ijfckp} - t^d_{ijfc-1,kp}) \ \forall m, i, j, f, c \geq 1, k, p,
\end{aligned}
\tag{7.21}
$$

where $U^{max}_m, U^{min}_m$ is the maximum and minimum change for manipulated variable $m$. The dynamic model is discretized using 20 finite elements with 3 collocation points. For all algorithms the optimality gap was set equal to 0.1%, and the initial production sequence was $A \rightarrow B \rightarrow C \rightarrow D$ for both periods. The master problem in the first iteration has 592 variables and 445 constraints and the dynamic optimization problem for each transition, slot and period has 265 variables and 326 constraints.

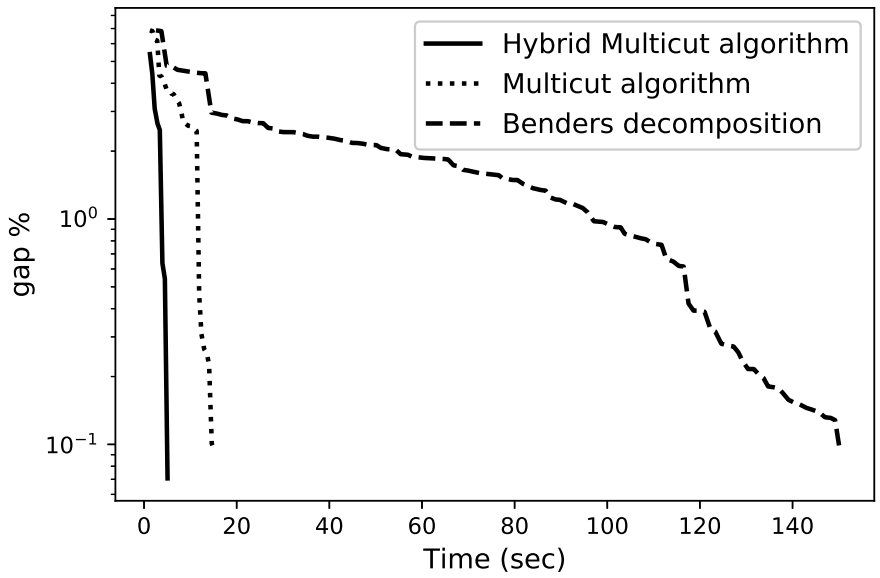


Figure 7.4: Evolution of the gap for the proposed algorithms and GBD (Benders decomposition) based on [125],[199] for the first case study.

The evolution of the optimality gap and the upper and lower bound with the CPU time for the different algorithms is presented in Fig. 7.4, 7.5. Based on the results, GBD based on the formulation and decomposition from [125, 199] converges after 150

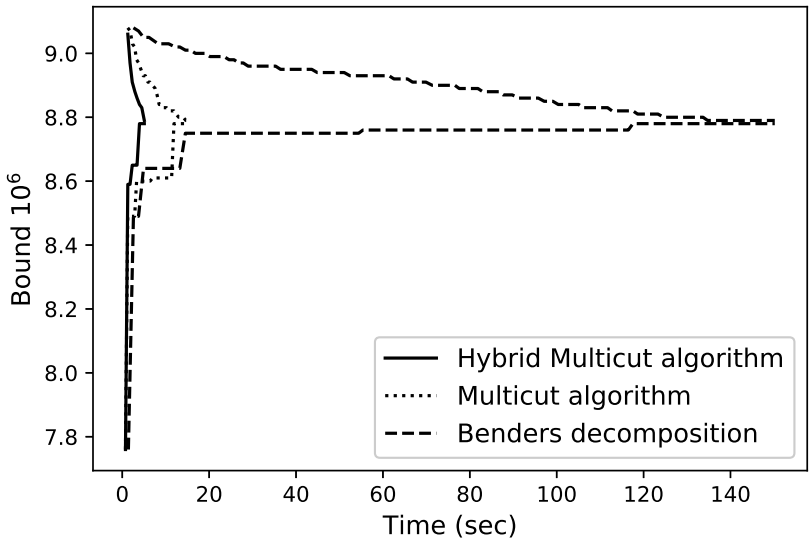


Figure 7.5: Evolution of the upper and lower bounds for the proposed algorithms and GBD (Benders decomposition) based on [125],[199] for the first case study.

Table 7.1: Operating conditions and product price for the first case study.

| Product | $c^{ss}(mol/L)$ | $Q^{ss}(L/h)$ | Production rate |
|---|---|---|---|
| A | 0.24 | 200 | 150 |
| B | 0.2 | 100 | 80 |
| C | 0.30 | 400 | 278 |
| D | 0.39 | 1000 | 607 |

Table 7.2: Operating and transition cost for the first case study, $C^{inv} = 0.026, a_u = 1$.

| | $C^{oper}$ | | $C^{trans}$ | | | |
|---|---|---|---|---|---|---|
| Product | $p = 1$ | $p = 2$ | A | B | C | D |
| A | 13 | 13 | 0 | 100 | 60 | 120 |
| B | 22 | 12 | 150 | 0 | 50 | 80 |
| C | 35 | 45 | 200 | 150 | 0 | 100 |
| D | 29 | 19 | 90 | 100 | 120 | 0 |

CPU seconds (131 iterations) and the value of the objective function is $8.781 \times 10^6$. The multicut algorithm converges after 15 CPU seconds (28 iterations) leading to a

Table 7.3: Product demand for the first case study.

| | Demand (mol/week) | | Price ($/mol) | |
|---|---|---|---|---|
| Prod. | $p = 1$ | $p = 2$ | $p = 1$ | $p = 2$ |
| A | 8000 | 9000 | 200 | 220 |
| B | 4000 | 3600 | 160 | 140 |
| C | 7000 | 8000 | 130 | 150 |
| D | 6000 | 11000 | 110 | 110 |

90% reduction in CPU time and the objective function is $8.782 \times 10^6$. From Fig. 7.5 we observe that the multicut algorithm reduces the bounds faster even though initially the lower bound obtained from GBD is better. The hybrid multicut algorithm converges after 5 seconds (8 iterations) leading to a 96% reduction in CPU time compared to GBD and 66% compared to the multicut algotirhm. The solution that is obtained is the same as the one obtained from the multicut algorithm. The difference in the solution time is due to the fact that in the hybrid multicut algorithm, in each iteration more information is added in the master problem through the multiple cuts for each transition. In GBD and the multicut algorithm, in each iteration the Benders cut provides information only about the specific production sequence and transition time. The production results obtained from the proposed algorithm are presented in Table 7.4, and the concentration and inlet flowrate profiles are presented in Fig. 7.6. From the production results, we see that the demand is satisfied and product $D$ is overproduced in the second period. Additionally, product $C$ is overproduced in the first period in order to satisfy the demand in the second period where the operating cost is higher, compared to the first period. Finally, no transition occurs between the two periods leading to a reduction in the cost and increase in the profit.

## 7.6 Case study 2: MMA polymerization reactor

In the second case study we will consider a methyl methacrylate (MMA) polymerization process. The states of the system are the concentration of the monomer $c_m$, the concentration of the initiator $c_i$, concentration of dead chains $D_0$ and mass concentration of dead chains $D_1$. The input is the flowrate of the initiator $F_i$ and the output is the molecular weight of each product $Y$. Based on the value of the initiator inlet flowrate, products with different molecular weight can be produced. The dynamic behavior of

Table 7.4: Production results for the for the first case study.

| Period 1 | | | | |
|---|---|---|---|---|
| Slot | Product | Production amount $(mol)$ | Production time $(hr)$ | Transition time $(hr)$ |
| 1 | B | 4000 | 50.0 | 0.98 |
| 2 | A | 8000 | 53.3 | 1.21 |
| 3 | C | 14176 | 51 | 1.58 |
| 4 | D | 6000 | 9.8 | 0 |
| Period 2 | | | | |
| Slot | Product | Production amount $(mol)$ | Production time $(hr)$ | Transition time $(hr)$ |
| 1 | D | 29968 | 49.3 | 1.87 |
| 2 | C | 823 | 2.9 | 1.55 |
| 3 | A | 9000 | 60 | 2.24 |
| 4 | B | 4000 | 50 | 0 |

the system is described by the following differential equations [64]:

$$\begin{aligned}
&\frac{dc_m(t)}{dt} = -(k_p + k_{fm})Pc_m(t)\sqrt{c_i(t)} + \frac{F(c_m^{in} - c_m(t))}{V} \\
&\frac{dc_i(t)}{dt} = -k_i c_i(t) + \frac{F_i(t)c_i^{in} - Fc_i(t)}{V} \\
&\frac{dD_0(t)}{dt} = (0.5k_{tc} + k_{td})P^2 c_i(t) + k_{fm}Pc_m(t)\sqrt{c_i(t)} - \frac{FD_0(t)}{V} \\
&\frac{dD_1(t)}{dt} = M_m(k_p + k_{fm})Pc_m(t)\sqrt{c_i(t)} - \frac{FD_1(t)}{V} \\
&Y(t) = D_1(t)/D_0(t) \\
&P = \sqrt{\frac{2f^* k_f}{k_{td} + k_{tc}}}
\end{aligned} \tag{7.22}$$

The parameters of the dynamic model can be found in Table 7.5 [64]. We will assume that four products must be produced in three planning periods. The steady state values of the states, inlet flowrate and molecular weight are presented in Table 7.6. In this case, the dynamic model is discretized using 20 finite elements and 3 collocation points. The dynamic optimization problems for each slot has 741 variables and 689 constraints and in the first iteration the master problem has 1368 variables and 2966 constraints. We solve the problem using the proposed multicut and hybrid multicut algorithms and

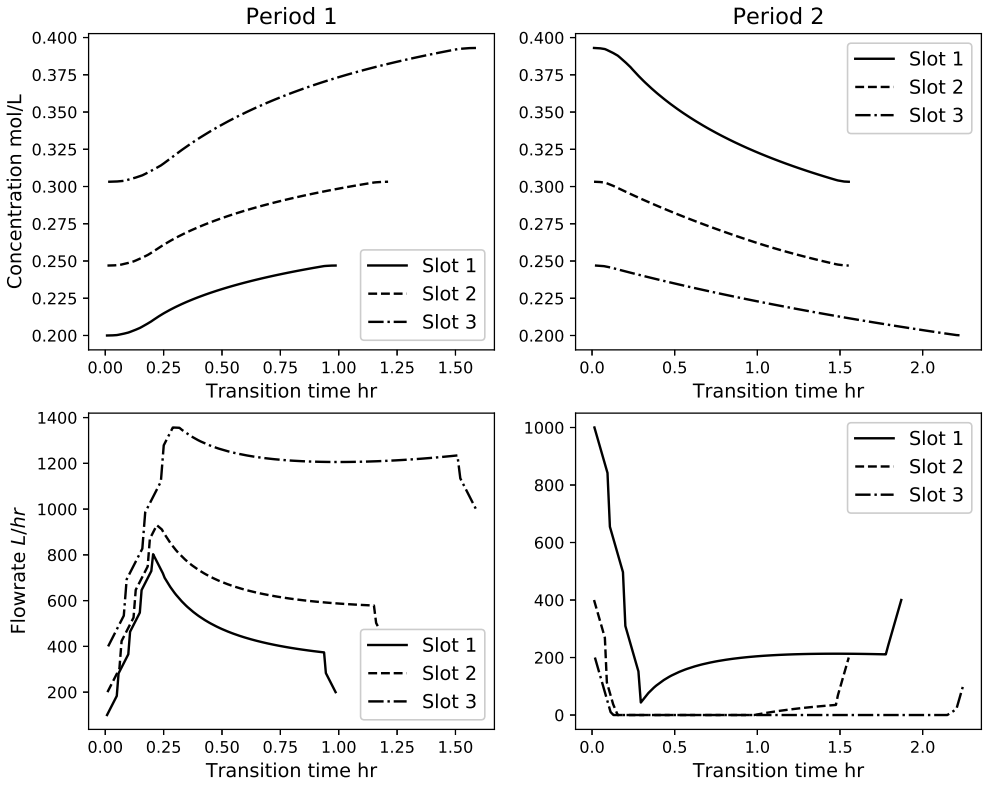


Figure 7.6: Concentration and inlet flowrate profiles for each slot and period for the first case study.

GBD based on [125, 199]. The algorithms are initialized by fixing the product sequence to $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ for all periods and the optimality gap tolerance is set to 0.1%. The economic data of the problem are presented in Tables 7.6,7.7,7.8.

The evolution of the optimality gap and the upper and lower bound with the CPU time for the different algorithms is presented in Fig. 7.7, 7.8. The multicut algorithm converges after 90 CPU seconds (17 iterations), the hybrid multicut algorithm converges after 35 seconds (7 iterations) and the value of the objective function is $5.06 \times 10^6$. GBD based on [125, 199] converges after 425 CPU seconds (71 iterations) and the value of the objective function is $4.84 \times 10^6$. The production sequence that is obtained from this algorithm is the same as the initial guess, leading to lower value of the objective function compared to the solution obtained by the multicut and hybrid multicut algorithm. In this case, the hybrid multicut algorithm reduces the CPU time by 91 %, the multicut algorithm by 78% and the value of the objective is improved by 4.5%. Similar to the

Table 7.5: Parameters of the dynamic model for the MMA rector

| Parameter | Value |
|---|---|
| F | 10 $m^3/h$ |
| V | 10 $m^3$ |
| $f^*$ | 0.58 |
| $k_p$ | 2.5 $10^6$ $m^3/(kmol\ h)$ |
| $k_{td}$ | 1.09 $10^{11}$ $m^3/(kmol\ h)$ |
| $k_{fm}$ | 2.45 $10^3$ $m^3/(kmol\ h)$ |
| $k_i$ | 1.02 $10^{-1}$ $h^{-1}$ |
| $c_i^{in}$ | 8 $kmol/m^3$ |
| $c_m^{in}$ | 6 $kmol/m^3$ |
| $M_m$ | 100.12 kg/kmol |

Table 7.6: Steady state and production rate values for the MMA rector

| | Products | | | |
|---|---|---|---|---|
| Steady state value | 1 | 2 | 3 | 4 |
| $c_m$ | 3.07 | 3.22 | 3.33 | 3.43 |
| $c_i$ | 0.14 | 0.12 | 0.1 | 0.09 |
| $D_0$ | 0.019 | 0.016 | 0.014 | 0.012 |
| $D_1$ | 292 | 277 | 267 | 257 |
| $Y$ | 15000 | 17000 | 18500 | 20000 |
| $F_i$ | 0.2 | 0.16 | 0.14 | 0.12 |
| Prod. rate ($kg/hr$) | 123 | 76.4 | 50.8 | 35.2 |

Table 7.7: Operating and transition cost for the second case study, $C^{inv} = 0.026, a_u = 10^6$

| | $C^{oper}$ | | | $C^{trans}$ | | | |
|---|---|---|---|---|---|---|---|
| Product | $p=1$ | $p=2$ | $p=3$ | A | B | C | D |
| 1 | 23 | 10 | 15 | 0 | 150 | 120 | 180 |
| 2 | 30 | 25 | 20 | 160 | 0 | 180 | 160 |
| 3 | 45 | 55 | 40 | 150 | 100 | 0 | 90 |
| 4 | 50 | 29 | 20 | 110 | 100 | 120 | 0 |

previous case study the multicut and hybrid multicut algorithms improve the upper and lower bounds faster compared to GBD. Also, the addition of multiple cuts per transition in the hybrid multicut algorithm leads to improved performance compared to the multicut algorithm (see Fig. 7.7). The production results obtained with the proposed

Table 7.8: Product demand and price for the second case study.

| | Demand (kg/week) | | | Price ($/kg) | | |
|---|---|---|---|---|---|---|
| Prod. | $p = 1$ | $p = 2$ | $p = 3$ | $p = 1$ | $p = 2$ | $p = 3$ |
| 1 | 2500 | 2200 | 2150 | 300 | 280 | 250 |
| 2 | 2200 | 1400 | 1800 | 180 | 150 | 160 |
| 3 | 3000 | 3500 | 2500 | 160 | 190 | 180 |
| 4 | 1000 | 2000 | 1500 | 120 | 120 | 130 |

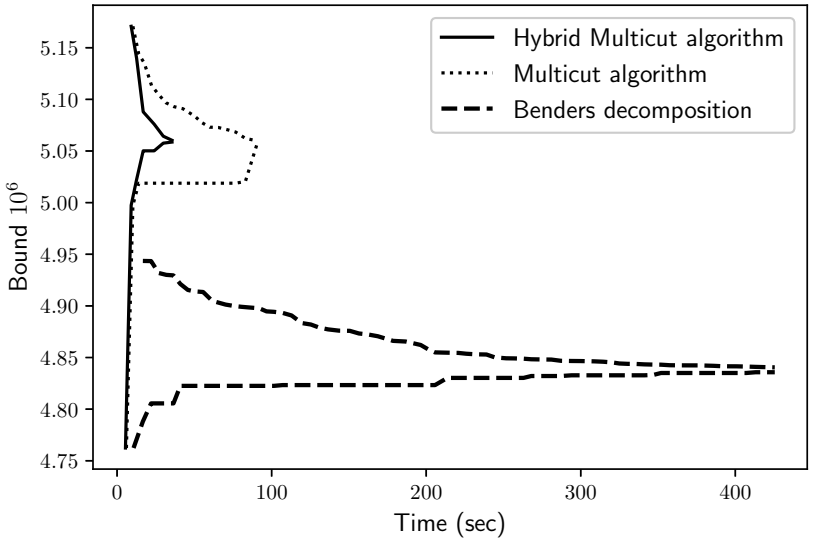


Figure 7.7: Evolution of the upper and lower bound for the different algorithms for the second case study.

approach are presented in Table 7.9 and the profiles of the output and manipulated variables are presented in Fig 7.10. From the production results, we observe that no transitions occur between the time periods leading to a reduction in the transition cost. Product $C$ is overproduced in the first period where the operating cost is lower compared to the second time period and product $D$ is overproduced in the last period.

Finally, we solve the problem for different numbers of planning periods and the convergence results are presented in Table 7.10. In all the cases the algorithms are initialized by fixing the production sequence to $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ for all periods and the optimality gap tolerance is set to 0.1%. The economic data can be found in the Supplementary Material.

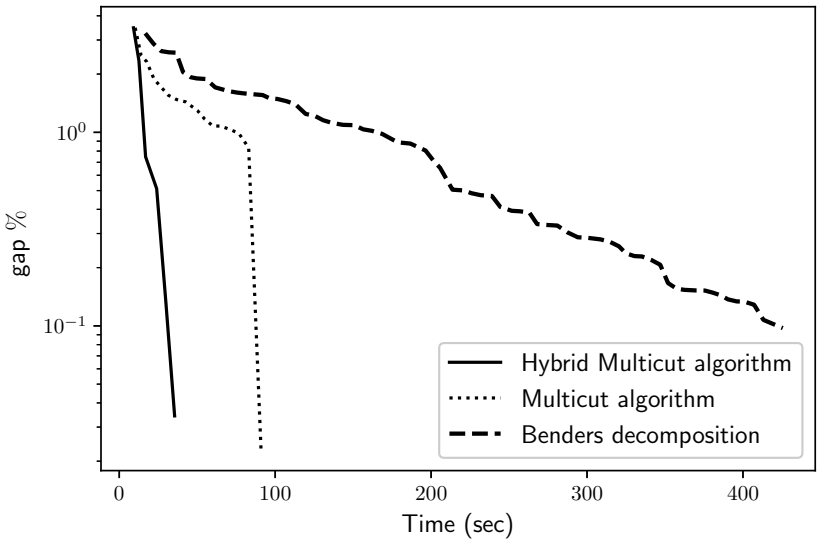


Figure 7.8: Evolution of the optimality gap for the different algorithms for the second case study.

From the results we observe that for all cases the proposed algorithms solve the problem faster than GBD and the solution that is obtained is better, i.e. the value of the objective function is higher. Specifically, the multicut algorithm reduces the CPU time up to 94% (for 7 planning periods) and the hybrid multicut algorithm up to 98% (for 9 planning periods). Both algorithms provide better results since the value of the objective function is increased up to 6.4% for 7 planning periods. For all algorithms an increase in the number of planning periods leads to an increase in the CPU time. For the proposed algorithms this increase is caused by the increase in the number of variables/constraints in the master problem. For example, the solution of the master problem requires 43% of the CPU time for the hybrid multicut algorithm for 6 planning periods and 53% of the CPU time for the multicut algorithm for 10 planning periods, whereas for the GBD algorithm the solution of the master problem requires (on average) 2.5% of the CPU time. However, the overall CPU time for the proposed algorithms is lower since fewer iterations are required, i.e. the master problem and subproblems are solved fewer times. Comparing the performance of the hybrid multicut and multicut algorithm, from Fig. 7.9 we observe that the solution time of the master problem per iteration is similar, however since the hybrid multicut algorithm requires fewer iterations

Table 7.9: Production results for the second case study.

| Period 1 | | | | |
|---|---|---|---|---|
| Slot | Product | Production amount ($kg$) | Production time ($hr$) | Transition time ($hr$) |
| 1 | 4 | 1000 | 28.36 | 1.10 |
| 2 | 3 | 4013 | 78.86 | 1.14 |
| 3 | 2 | 2200 | 28.76 | 1.41 |
| 4 | 1 | 3481 | 28.28 | 0 |
| Period 2 | | | | |
| Slot | Product | Production amount ($kg$) | Production time ($hr$) | Transition time ($hr$) |
| 1 | 1 | 3481 | 28.28 | 1.95 |
| 2 | 2 | 2200 | 28.76 | 1.69 |
| 3 | 3 | 2486 | 48.85 | 1.66 |
| 4 | 4 | 2000 | 56.72 | 0 |
| Period 3 | | | | |
| Slot | Product | Production amount ($kg$) | Production time ($hr$) | Transition time ($hr$) |
| 1 | 4 | 2000 | 56.72 | 1.10 |
| 2 | 3 | 3500 | 68.77 | 1.25 |
| 3 | 2 | 1000 | 13.07 | 1.41 |
| 4 | 1 | 3156 | 25.64 | 0 |

the total CPU time for the solution of the problem and the solution of the master problem is lower.

We note that the dynamic optimization problems that are solved in each iteration for all algorithms are independent and can be solved in parallel. Although this will lead to a reduction in the total CPU time, it will not affect the computational performance of the hybrid multicut algorithm compared to GBD, since this parallelization will reduce the computational time for the solution of the subproblems. The solution time of the master problem will not be affected. From Table 7.10 we can see that for the cases considered the total solution time of the master problem in the hybrid multicut algorithm is lower than the total solution time of the master problem in the GBD algorithm. Therefore, the solution time of the hybrid multicut algorithm will still be lower than the solution time of GBD. Finally, based on these results we note that for a large number of products/planning periods the limiting step in the proposed algorithm will be the solution of the master problem. In such cases one must employ different acceleration

Table 7.10: Convergence results for the MMA reactor for different planning periods.

| | 4 weeks | 5 weeks | 6 weeks | 7 weeks | 8 weeks | 9 weeks | 10 weeks |
|---|---|---|---|---|---|---|---|
| Hybrid Multicut algorithm | | | | | | | |
| Master problem[a] | | | | | | | |
| Variables | 1824 (400)[b] | 2280 (500)[b] | 2736 (600)[b] | 3192 (700)[b] | 3648 (800)[b] | 4104 (900)[b] | 4560 (1000)[b] |
| Constraints | 3959 | 4952 | 5945 | 6938 | 7931 | 8924 | 9917 |
| Objective ($\times 10^6$) | 7.57 | 9.66 | 10.70 | 12.24 | 14.7 | 17.64 | 21.58 |
| CPU time (sec) | 56 | 72 | 123 | 100 | 161 | 128 | 247 |
| Master prob. | 14.2 | 20 | 54 | 28.8 | 62.5 | 41.3 | 106 |
| Subproblem | 41.8 | 52 | 68 | 71.2 | 98.5 | 86.7 | 141 |
| Iterations | 7 | 6 | 7 | 6 | 7 | 5 | 6 |
| Multicut algorithm | | | | | | | |
| Master problem[a] | | | | | | | |
| Variables | 1824 (400)[b] | 2280 (500)[b] | 2736 (600)[b] | 3192 (700)[b] | 3648 (800)[b] | 4104 (900)[b] | 4560 (1000)[b] |
| Constraints | 3959 | 4952 | 5945 | 6938 | 7931 | 8924 | 9917 |
| Objective ($\times 10^6$) | 7.57 | 9.66 | 10.70 | 12.24 | 14.7 | 17.64 | 21.5 |
| CPU time (sec) | 156 | 199 | 464 | 289 | 681 | 613 | 1193 |
| Master prob. | 42.58 | 72.94 | 212.32 | 111 | 306 | 256 | 612 |
| Subproblem | 113.15 | 126.06 | 251.68 | 178 | 375 | 357 | 571 |
| Iterations | 23 | 21 | 28 | 17 | 27 | 19 | 30 |
| Generalized Benders Decomposition from [125, 199] | | | | | | | |
| Master problem[a] | | | | | | | |
| Variables | 865 (400)[b] | 1081 (500)[b] | 1297 (600)[b] | 1513 (700)[b] | 1729 (800)[b] | 1945 (900)[b] | 2161 (1000)[b] |
| Constraints | 865 | 1086 | 1307 | 1507 | 1725 | 1943 | 2616 |
| Objective ($\times 10^6$) | 7.27 | 9.27 | 10.2 | 11.5 | 14 | 16.7 | 20.6 |
| CPU time (sec) | 908 | 1812 | 2087 | 5095 | 6800 | 9748 | 11246 |
| Master prob. | 22 | 42 | 47 | 106 | 256 | 247 | 314 |
| Subproblem | 886 | 1170 | 2040 | 4989 | 6544 | 9488 | 10932 |
| Iterations | 96 | 165 | 179 | 252 | 418 | 424 | 451 |

[a] First iteration
[b] Binary variables

techniques for the solution of the master problem, such as using decomposition based solution algorithms like Benders [27] or bilevel [90, 264] decomposition.

## 7.7 Conclusions

The optimal operation of process systems depends on the solution of problems that involve different time scales, the integration of which leads to large scale optimization problems. In this work, we proposed a new formulation of the integrated planning, scheduling, and dynamic optimization problem for continuous systems. In this approach, all the possible transitions are considered simultaneously. Using SBM we analyzed the structure of the problem and proposed a multicut and a hybrid multicut Generalized Benders Decomposition algorithm. Through case studies we showed that the proposed

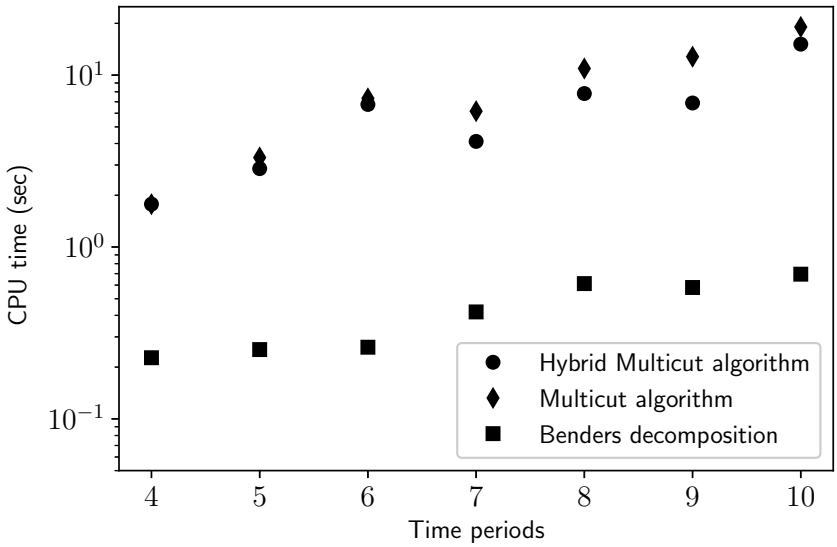


Figure 7.9: Average computational time per iteration for the solution of the master problem for different planning periods.

problem formulation and multicut and hybrid multicut GBD algorithms can reduce the necessary CPU time and find a better solution compared to the application of GBD using other formulations. Finally, we point out some extensions of the proposed problem formulation and decomposition based solution approach in the remarks below.

**Remark 7.4.** In this work we considered the case of planning, scheduling, and dynamic optimization. The same problem formulation and solution approach can be followed in the case of the integration of scheduling and dynamic optimization for continuous systems for different problems such as cyclic/ non-cyclic schedules and single and parallel lines.

**Remark 7.5.** In the case studies considered, the production planning and scheduling problem was modelled using a slot based formulation. We can also follow a Traveling Salesman Problem formulation [57, 182]. In that case the time horizon is discretized into periods and in each period each transition can occur at most once. We can define $\phi_{ijp}$ as the value function for the dynamic optimization cost for the transition from product $i$ to $j$ in slot $p$. The value of $\phi_{ijp}$ will depend on the value of the transition time. Therefore, we can follow the same hybrid multicut GBD algorithm to solve the

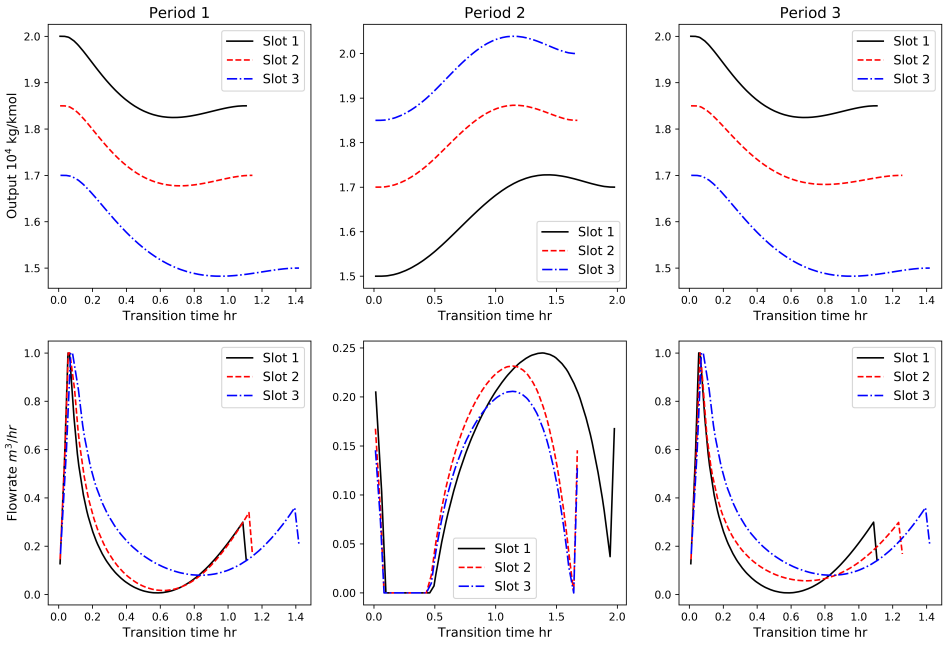


Figure 7.10: Inlet flowrate and output profile for the second case study.

problem.

# Chapter 8
# Efficient solution of mixed integer model predictive control problems via Benders decomposition

## 8.1 Introduction

Model Predictive Contol (MPC) is a common strategy for controlling and operating a system such that a desired objective function is optimized subject to constraints that describe the behavior of the system [246]. MPC is usually applied to continuous dynamical systems. The efficient solution of the MPC dynamic optimization problem is especially challenging when the optimization problem contains both discrete and continuous variables.

Discrete variables can arise either due to the hybrid nature of the dynamic system (e.g., piece-wise affine dynamics) or due to the presence of discrete variables related to the operation of the system. Typical examples of the latter include the operation of energy systems where a generator is either on or off [79] and the operation of chemical processes that can operate only at specific operating points [75]. Such cases are becoming common in the chemical industry due to converging time scales of scheduling and real-time operation caused by fast-changing economic conditions [17, 249]. These problems are formally known as mixed-integer optimal control problems [191] and are formulated as Mixed Integer Dynamic Optimization problems (MIDO). A common approach to solving such problems is to discretize the differential equations that describe the system and then use mathematical optimization techniques. This approach approximates the original MIDO problem with a Mixed Integer Programming (MIP) problem.

Solving such MIP problems online can be a daunting computational task and thus a major limitation for the implementation of mixed integer MPC. Branch and Bound is a standard (off-line) solution strategy for MIP problems, where branching is performed either on the integer variables or on the spatial domain of the feasible region for nonconvex nonlinear problems [273, 132]. This approach can guarantee global optimality, yet it is generally slow for online applications. Multiparametric programming can be used to efficiently determine the optimal solution of the MIP problem [223, 10]. However, computing the critical regions, i.e., the optimal solution, for every value of the parameters of the MIP problem is computationally expensive for nonlinear systems with many states. Another solution approach is to exploit the underlying structure of the problem and use decomposition-based solution algorithms, such as the combinatorial integral

approximation (CIA) [251], dual dynamic programming [162], and Generalized Benders Decomposition (GBD) [208, 200, 192, 288].

GBD is based on the observation that if a subset of the variables of the optimization problem, called complicating variables, are fixed, then the problem can be solved faster. GBD decomposes the MIP problem into a master problem which considers the binary (and possibly some continuous) variables and a subproblem which is a continuous optimization problem. This decomposition is based on the underlying structure of the problem, which in general can be inferred using structure detection methods from network science [207, 199]. The master problem and subproblem are solved iteratively and coordinated via cuts which inform the former about the effect of the complicating variables on the latter. The convergence of GBD can be slow and is guaranteed only when the value function of the subproblem is convex [108].

Recently, machine learning (ML) has been used to improve the computational performance of optimization algorithms [28]. In the context of mixed integer MPC, ML can be used to approximate the values of the binary variables [190, 303] and active constraints [56, 33]. This approach transforms the MIP problem into a continuous optimization problem that can be solved faster. Alternatively, ML can be used to accelerate the solver itself, either by warm-starting or improving algorithmic steps such as branching and cut selection for GBD [204].

In this work, we aim to reduce the computational time related to the solution of the master problem and the subproblem, and thus enable the efficient online application of GBD in mixed integer MPC problems. Specifically, we propose a branch and check algorithm where the master problem is solved only once and Benders cuts are added whenever an integer feasible solution is found during branch and bound. We use machine learning to approximate the information that is required for the construction of cuts added to the master problem, therefore, alleviating the need for iterative solution of the subproblem. The proposed algorithm can be applied to a broad class of optimization problems that frequently arise in mixed-integer MPC applications, such as Mixed Integer Linear (MILP), Quadratic (MIQP) and Nonlinear (MINLP) Programming problems. We illustrate the application and advantages of the proposed approach through two case studies; one on the dynamic real time operation of chemical processes and one on mixed integer model predictive control.

## 8.2 Machine learning based branch and check Generalized Benders Decomposition algorithm

### 8.2.1 Generalized Benders Decomposition

Consider the following optimization problem

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g(x) \leq 0 \\ & x \in \mathbb{R}^{n_x^c} \times \mathbb{Z}^{n_x^d} \end{aligned} \tag{8.1}$$

where $x$ are the decision variables that can be both continuous and discrete. The first step in the implementation of GBD is to decompose the problem and determine the complicating variables. Let's assume, for simplicity, that the constraints of the problem are partitioned into two sets $g_1, g_2$ using some structure detection approach [207, 199]. Given this partition of the constraints, the variables are partitioned into three sets: variables $u$ that are present only in constraints $g_1$, variables $z$ that are present only in $g_2$, and variables $y$ that are present in both constraints, i.e., complicating variables. Given this partition of the variables and constraints, the above problem can be written as

$$\begin{aligned} \underset{u,y,z}{\text{minimize}} \quad & f_1(u,y) + f_2(z,y) \\ \text{subject to} \quad & g_1(u,y) \leq 0 \\ & g_2(z,y) \leq 0 \\ & u \in \mathbb{R}^{n_u}, y \in \mathbb{R}^{n_y^c} \times \mathbb{Z}^{n_y^d}, z \in \mathbb{R}^{n_z}, \end{aligned} \tag{8.2}$$

where $n_u + n_y^c + n_z = n_x^c$, $n_y^d = n_x^d$, and the objective is also decomposed into two terms. Note that in this case, the complicating variables are both continuous and discrete. Fixing the $y$ variables in Eq. 8.2, we obtain the subproblem

$$\begin{aligned} \mathcal{S}(y) := \underset{\bar{y},z}{\text{minimize}} \quad & f_2(\bar{y},z) \\ \text{subject to} \quad & g_2(\bar{y},z;p_2) \leq 0 \\ & \bar{y} = y \quad : \quad \lambda \\ & \bar{y} \in \mathbb{R}^{n_y^c + n_y^d}, z \in \mathbb{R}^{n_z}, \end{aligned} \tag{8.3}$$

where $\lambda$ are the Lagrange multipliers for the equality constraints $\bar{y} = y$. The solution of this problem, in terms of objective function value and the Lagrange multipliers, depends on the value of the complicating variables $y$. Given the value function of the subproblem $\mathcal{S}(y)$, the original problem in Eq. 8.2 can be written as

$$\begin{aligned} \underset{u,y}{\text{minimize}} \quad & f_1(u,y) + \mathcal{S}(y) \\ \text{subject to} \quad & g_1(u,y) \leq 0 \\ & u \in \mathbb{R}^{n_u}, y \in \mathbb{R}^{n_y^c} \times \mathbb{Z}^{n_y^d}. \end{aligned} \tag{8.4}$$

This problem can not be solved directly, since $\mathcal{S}(y)$ is not known explicitly. An approach to overcome this obstacle is to approximate $\mathcal{S}$ with hyperplanes generated using information from the dual of the subproblem [108]. Specifically, if the subproblem is solved for $\bar{y} = y$, then $\mathcal{S}(y)$ can be approximated from below as follows [107]:

$$\mathcal{S}(y) \geq \mathcal{S}(\bar{y}) + \partial\mathcal{S}(\bar{y})(y - \bar{y}), \tag{8.5}$$

where $\partial\mathcal{S}(\bar{y})$ is the subgradient of the value function for $\bar{y} = y$ and is equal to the negative of the optimal Lagrange multiplier $\lambda$, i.e., $\partial\mathcal{S}(\bar{y}) = -\lambda$. This inequality is known as Benders cut [108], and the problem

$$\underset{y}{\text{minimize}} \quad \mathcal{S}(y) \tag{8.6}$$

is equivalent to:

$$\begin{aligned} \underset{y}{\text{minimize}} \quad & \eta \\ \text{subject to} \quad & \eta \geq \mathcal{S}(\bar{y}^l) - \lambda^l(y - \bar{y}^l) \; \forall l \in \mathcal{L} \\ & y \in \mathbb{R}^{n_y^c} \times \mathbb{Z}^{n_y^d}, \end{aligned} \tag{8.7}$$

where $\mathcal{L} = \{1, ..., \ell\}$ is the number of points used to approximate the value function and the overbar denotes a fixed value. Given this approximation of $\mathcal{S}$ the problem in Eq. 8.4 can be written as

$$\begin{aligned} \underset{u,y}{\text{minimize}} \quad & f_1(u,y) + \eta \\ \text{subject to} \quad & g_1(u,y) \leq 0 \\ & \eta \geq \mathcal{S}(\bar{y}^l) - \lambda^l(y - \bar{y}^l) \; \forall l \in \mathcal{L} \\ & u \in \mathbb{R}^{n_u}, y \in \mathbb{R}^{n_y^c} \times \mathbb{Z}^{n_y^d}. \end{aligned} \tag{8.8}$$

We note that the Benders cuts are generated in cases where the subproblem (Eq. 8.3) is feasible for given values of the complicating variables $y$. For cases where the subproblem is infeasible, a feasibility cut can be generated (see [108] for details on the generation of these cuts).

This approximation is exact for cases where the value function $\mathcal{S}(y)$ is convex [108], otherwise the solution of this problem cannot be guaranteed to be the globally optimal solution of the original problem (Eq. 8.2). However, even for the case where $\mathcal{S}(y)$ is convex, the solution to this problem can be challenging since the number of approximations that must be added can be large (e.g., consider the case where the complicating variables are only continuous and thus an infinite number of approximations are added). To overcome this, the cuts are added iteratively as dictated by the solution of the problem in Eq. 8.8 which is called the master problem. Specifically, in the first iteration, the master problem is solved without any cuts. This solution provides a lower bound to the optimal solution of the original problem and the values of the complicating variables. The subproblem is solved for the given values of the complicating variables providing an upper bound and the information for the generation of the Benders cuts. Once the subproblem is solved, the approximations are added to the master problem which is solved again. This procedure is repeated until the bounds converge.

### 8.2.2 Branch and check solution approach

The repeated solution of the master problem in the standard application of GBD described above is computationally expensive for online applications or for problems where the master problem is complex. Branch and check has been proposed as an approach to overcome this limitation [277]. In branch and check the solution of the master problem starts from the LP (or continuous) relaxation of the master problem which generates a set of open nodes $R$ and standard node selection techniques are used to select a node $\rho$. If the solution of the LP relaxation at node $\rho$ is integral, then the subproblem (Eq. 8.3) is solved for $\bar{y} = y^\rho$ and the cut

$$\eta \geq \mathcal{S}(y^\rho) - \lambda(y^\rho)(y - y^\rho) \tag{8.9}$$

is added to all the open nodes $R$ in the branch and bound tree and the solver continues the solution of the problem by adapting the set of open nodes.

### 8.2.3 Machine learning based branch and check Generalized Benders Decomposition

A major limitation in the branch and check GBD algorithm described above is the need to solve the subproblem every time an integer feasible solution is found. This can be limiting in cases where the subproblem is computationally expensive or the original problem is decomposed into multiple subproblems. We address this limitation by proposing a machine learning approach to alleviate the need to solve the subproblem multiple times.

Given an integer feasible solution of the master problem at a node $\rho$, the solution of the subproblem provides $\mathcal{S}(y^\rho)$ and $\lambda(y^\rho)$, information which is later used to form the cut and add it in the master problem. We will approximate $\mathcal{S}(y), \lambda(y)$ with surrogate models $\tilde{\mathcal{S}}(y), \tilde{\lambda}(y)$. These models can be obtained offline by solving the subproblem for different values of the complicating variables $y_i$ and obtaining $\mathcal{S}(y_i)$ and $\lambda(y_i)$. Through this procedure, we can obtain two datasets; one that can be used to learn the value function $\{y_i, \mathcal{S}(y_i)\}_{i=1}^{N_{train}}$ and $\{y_i, \lambda(y_i)\}_{i=1}^{N_{train}}$ which can be used to learn a function to approximate the value of the Lagrange multipliers. Given these surrogate models, once an integer feasible solution is found at a node $\rho$ in the branch and bound tree, the following cut is added to the open nodes

$$\eta \geq \tilde{S}(y^\rho) - \tilde{\lambda}(y^\rho)(y - y^\rho). \tag{8.10}$$

and the solver continues the solution of the problem (see Algorithm 8.1).

This algorithm can be applied to a wide class of mixed integer MPC applications, where the underlying optimization problem is an MILP ($f$,$g$ are affine), an MIQP ($f$ is quadratic positive definite and $g$ is affine), or an MINLP (which is either convex, i.e., the continuous relaxation is convex, or the value function of the subproblem is convex).

Note that although the proposed algorithm combines mathematical optimization with machine learning, it is different than other ML-based techniques [190, 303, 56, 33, 204] which focus on the initialization of the solution method. In the proposed approach, ML is used to accelerate the algorithmic step related to the solution of the subproblem. A similar approach has been proposed in [168] for the solution of linear two-stage stochastic optimization problems using Benders Decomposition.

Note also that the solution obtained by the proposed approach is not guaranteed to

be the same as the one obtained by the application of standard GBD or a monolithic solution approach, since both the value function and the dual variables are approximated. However, the approximation does not affect the feasibility of the problem itself which is determined by the feasibility of the master problem, i.e., the cuts approximate the value function of the subproblem which does not affect the feasible region of the master problem.

## 8.3 Application to dynamic real time optimization of chemical processes

We apply the proposed method to a case study regarding the operation of chemical processes. We will consider the case where an isothermal continuously stirred tank reactor (CSTR) is used to manufacture $N_p$ products and a dynamic real time optimization problem is solved to determine the production sequence while considering the dynamic behavior of the system given the price, production cost, and demand for each product. The problem formulation for this case is similar to the model presented in Chapter 7 and in [200]. We present here the main components of the model.

### 8.3.1 Mathematical optimization model

#### Scheduling problem

The scheduling horizon is $H$ hours and is discretized into $N_s$ slots ($N_s = N_p$). We define a binary variable $W_{ik}$ which is equal to one if product $i$ is manufactured in slot $k$ and zero otherwise. We also define binary variable $Z_{ijk}$ which is equal to one if a transition occurs from product $i$ to product $j$ in slot $k$ and zero otherwise. We will assume that at every time point, only one product is manufactured. The logic constraints related to the production sequence and transitions are

$$\begin{aligned} &\sum_{i=1}^{N_p} W_{ik} = 1 \ \forall k \\ &Z_{ijk} \geq W_{ik} + W_{j,k+1} - 1 \ \forall i, j, k \neq N_s. \end{aligned} \tag{8.11}$$

We define $T_k^s$ and $T_k^e$ as the starting and ending time of slot $k$ ($T_1^s = 0, T_{N_s}^e = H$). The production time of product $i$ in slot $k$ is $\Theta_{ik}$, the transition time between product $i$ and

**Data:** Optimization problem, surrogate models $\tilde{\mathcal{S}}$ and $\tilde{\lambda}$
**Result:** Solution of the optimization problem

Start Branch and Bound algorithm by solving continuous relaxation of the master problem;
Obtain open nodes $R$;
**while** $R \neq \emptyset$ **do**
    Select a node $\rho$ from $R$ using node selection techniques;
    Solve continuous relaxation at $\rho$ and obtain $y^\rho$;
    **if** $y^\rho$ *is integer* **then**
        Approximate the value function $\tilde{\mathcal{S}}(y^\rho)$;
        Approximate Lagrange multipliers $\tilde{\lambda}(y^\rho)$;
        Add Benders cut to all open nodes in $R$;

$$\eta \geq \tilde{S}(y^\rho) - \tilde{\lambda}(y^\rho)(y - y^\rho)$$

        Add MIP-based cuts Update existing nodes in $R$;
    **else**
        Partition domain of $y$ variables;
        Add MIP-based cuts;
        Update existing nodes in $R$;
    **end**
**end**

**Algorithm 8.1:** Machine Learning based Branch and Check Generalized Benders Decomposition

$j$ in slot $k$ is $\theta_{ijk}$, and the timing constraints are:

$$\begin{aligned}
&T_k^e = T_k^s + \sum_i \Theta_{ik} + \sum_{ij} \theta_{ijk} Z_{ijk} \ \forall k \\
&T_{k+1}^s = T_k^e \ \forall k \neq N_s \\
&\Theta_{ik} \leq W_{ik} H \ \forall i, k \\
&\theta_{ijk} \geq \theta_{ij}^{min} \ \forall i, j, k,
\end{aligned} \tag{8.12}$$

where $\theta_{ij}^{min}$ is the minimum transition time from product $i$ to product $j$. Given these constraints, each slot has two regimes, a production regime where a product is manufactured with duration $\Theta_{ik}$ and a transition regime with duration $\theta_{ijk}$. The amount of product $i$ manufactured in slot $k$ is $q_{ik}$ and the inventory level is $I_{ik}$. The inventory and

demand satisfaction constraints are:

$$\begin{aligned} &I_{ik} = I_{i,k-1} + r_i\Theta_{ik} - S_{ik} \ \forall i, k \geq 1 \\ &S_{i,N_s} \geq d_i \ \forall i, \end{aligned} \tag{8.13}$$

where $r_i$ is the production rate of product $i$ and $d_i$ is the demand.

**Dynamic model**

We assume that an irreversible reaction occurs ($A \rightarrow 3B$) and multiple products can be manufactured by adjusting the inlet flowrate. The dynamic behavior of the system is described by the following differential equation

$$\frac{dc(t)}{dt} = \frac{F(t)}{V}(c_{feed} - c(t)) - k_r c(t)^3, \tag{8.14}$$

where $c(t)$ is the concentration of $A$, $F(t)$ is the inlet flowrate, and $V, c_{feed}, k_r$ are parameters which correspond to the reactor volume, inlet concentration and reaction constant, and are equal to 5000 $L$, 1 $mol/L$, 2 $L^2/(hr\ mol^2)$. The differential equation is discretized using orthogonal collocation on finite elements (using $N_{fe}$ finite elements and $N_l$ collocation points).

We consider simultaneously all the transitions between the products and we define $c_{ijklp}, F_{ijklp}$ as the concentration and inlet flowrate for a transition from product $i$ to $j$ in slot $k$ at finite element $f$ and collocation point $p$. The algebraic equations that describe the dynamic behavior of the system during a transition are represented as (see [201] for a detailed description)

$$G(c_{ijklp}, F_{ijklp}, t_{ijklp}, \theta_{ijk}) \leq 0 \ \forall i, j, k, l, p, \tag{8.15}$$

**Optimization problem**

The objective function has two terms. The first represents the sales minus the operating and inventory cost and the second term is the transition cost ($\Phi_2$). The costs are

$$\begin{aligned}\Phi_1 &= \sum_{ik} P_{ik} S_{ik} - \sum_{ik} C^{op}_{ik} q_{ik} - C^{inv} I_{ik} - \sum_{ijk} C^{tr}_{ij} Z_{ijk} \\ \Phi_2 &= \sum_{ijk} Z_{ijk} \underbrace{\alpha_u \Bigg( \sum_{lp} N_l^{-1} t_{ijklp} \Lambda_{p,N_p} (F_{ijklp} - F^{ss}_j)^2 \Bigg)}_{f^{dyn}_{ijk}},\end{aligned} \tag{8.16}$$

where $P_{ik}$ is the price of product $i$ in slot $k$, $C^{op}_{ik}$ is the operating cost of product $i$ in slot $k$, $C^{inv}$ is the inventory cost, $C^{tr}_{ij}$ is the fixed transition cost from product $i$ to $j$, $\alpha_u$ is a weight coefficient, and $\Lambda$ is the collocation matrix. Overall, the resulting problem is a mixed integer economic MPC problem and the optimization problem is

$$\begin{aligned}\mathcal{P}(p) := \text{maximize} \quad & \Phi_1 - \Phi_2 \\ \text{subject to} \quad & \text{Eq. } 8.11, 8.12, 8.13, 8.15,\end{aligned} \tag{8.17}$$

where $p = \{\{d_i\}_{i=1}^{N_p}, \{P_i\}_{i=1}^{N_p}, \{C^{op}_{ik}\}_{i=1\,j=1}^{N_p\ N_s}, C^{inv}, \{C^{tr}_{ij}\}_{i=1\,j=1}^{N_p\ N_p}\}$. This is an MINLP problem that can not be solved efficiently monolithically.

### 8.3.2 Decomposition of the optimization problem

The above problem can be solved using GBD by noting that if the scheduling-related variables are fixed, then the problem is decomposed into a number of independent nonlinear optimization problems. Specifically, if we fix the $W_{ik}, Z_{ijk}, T^s_k, T^e_k, \Theta_{ik}, \theta_{ijk}, I_{ik}, S_{ik}$ variables, then problem in Eq. 8.17 can be written as

$$\begin{aligned}\text{maximize} \quad & -\sum_{ijk} Z_{ijk} f^{dyn}_{ijk} \\ \text{subject to} \quad & G(c_{ijklp}, F_{ijklp}, t_{ijklp}, \theta_{ijk}) \leq 0 \ \forall i, j, k, l, p.\end{aligned} \tag{8.18}$$

From Eq. 8.18 we observe that the problem can be decomposed into independent subproblems and the solution of each subproblem depends on the values of the transition times $\theta^t_{ijk}$. Therefore, we define the subproblem as

The optimal solution of this problem and the value of the dual variable for the

equality constraint $\bar{\theta} = \theta_{ijk}$ depend on the transition time, i.e., $\mathcal{S}(\theta^t_{ijk}), \lambda_{ijk}(\theta_{ijk})$. Given this subproblem, we can reformulate the original problem in Eq. 8.17 as follows

$$\begin{aligned} &\text{maximize} \;\; \Phi_1 - \sum_{ijk} Z_{ijk}\mathcal{S}_{ijk}(\theta^t_{ijk}) \\ &\text{subject to} \;\; \text{Eq. } 8.11, 8.12, 8.13. \end{aligned} \tag{8.19}$$

Since the value functions $\mathcal{S}_{ijk}$ are not known explicitly, the above problem cannot be solved directly. However, $\mathcal{S}_{ijk}$ is convex (see [201]), therefore, we can approximate the transition costs using Benders cuts as follows

$$\begin{aligned} \max \;\; & \Phi_1 - \sum_{ijk} Z_{ijk}\eta_{ijk} \\ \text{s.t.} \;\; & \text{Eq. } 8.11, 8.12, 8.13 \\ & \eta_{ijk} \geq \mathcal{S}(\theta^l_{ijk}) - \lambda^l_{ijk}(\theta^l_{ijk})(\theta_{ijk} - \theta^l_{ijk}) \;\; \forall l \in \mathcal{L} \end{aligned} \tag{8.20}$$

Given this decomposition of the mixed integer economic MPC problem, the master problem is an MINLP with bilinear terms ($Z_{ijk}\eta_{ikl}, Z_{ijk}\theta_{ijk}$) and every subproblem is a Nonlinear Programming problem. We note that the bilinear terms in the master problem can be linearized exactly, however, the linearization adds more constraints and variables which might slow down the solver. In previous work we found that for small number of products, it is faster to solve the MINLP master problem [201], therefore, we do not linearize the bilinear terms. Given this decomposition, we can apply the machine learning based algorithm presented in the previous section.

### 8.3.3 Learning the surrogate models and implementation

We learn the value functions and Lagrange multipliers for every transition using supervised learning. For the generation of the training data, we discretize the transition time into $N_{data} = 1000$ points between $[\theta^{min}_{ij}, 5\theta^{min}_{ij}]$. For every value of the transition time we solve the subproblem using IPOPT [285], we obtain $\mathcal{S}(\theta_i)$ and $\lambda_i(\theta_i)$ and form two datasets $\{\theta_i, \mathcal{S}(\theta_i)\}_{i=1}^{N_{data}}$, $\{\theta_i, \lambda_i(\theta_i)\}_{i=1}^{N_{data}}$.

We use neural networks to approximate $\mathcal{S}, \lambda$ using Scikit-learn [232]. All the neural networks have the same architecture, 3 layers with 50 neurons and `relu` as activation function. We use Adam [156] with learning rate equal to $10^{-4}$ (constant), $L_2$ regularization term equal to $10^{-4}$, the maximum number of iterations is $10^4$, and the solver

Table 8.1: Mean ($\mu$) and standard deviation ($\sigma$) of price and demand parameters for the different products

| Product | Price ($/mol) $\mu$ ($\sigma$) | Demand ($mol) $\mu$ ($\sigma$) |
|---|---|---|
| 1 | 200 (20) | 600 (20) |
| 2 | 160 (16) | 550 (10) |
| 3 | 130 (13) | 600 (30) |
| 4 | 110 (11) | 1220 (50) |
| 5 | 140 (14) | 2000 (20) |

Table 8.2: Operating and transition cost, $\alpha_u = 1$, $C^{inv} = 0.026(\$/mol)$

| Product | $C^{op}$ ($/mol) | $C^{trans}$($) | | | | |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| 1 | 33 | 0 | 100 | 60 | 120 | 150 |
| 2 | 22 | 150 | 0 | 50 | 80 | 100 |
| 3 | 35 | 200 | 150 | 0 | 100 | 150 |
| 4 | 29 | 90 | 100 | 120 | 0 | 100 |
| 5 | 25 | 150 | 100 | 150 | 140 | 0 |

stops if the objective does not improve for $10^3$ iterations. We implement the proposed algorithm in Pyomo [129] using callbacks in Gurobi [124] with `LazyCuts`.

### 8.3.4 Computational results

We generate 20 random instances of the problem assuming that five products must be manufactured and the price and demand of each product follow a uniform distribution with parameters presented in Table 8.1 and 8.2. We solve the problem using the proposed approach and the multicut GBD algorithm proposed in [201]. We note that the initialization of GBD can affect the solution time. In this case study, the initialization for the proposed algorithm and the GBD algorithm proposed in [201] is the same since the master problem for both algorithms is the same (the master problem in the first iteration of GBD has no cuts).

From the results we observe that the average solution time of the proposed algorithm is 0.56 seconds and the standard deviation is 0.11 seconds, whereas the average solution time of GBD proposed in [201] is 10.84 seconds and the standard deviation is 0.84 seconds (see Fig. 8.1). These results show that the proposed approach achieves on average 94% reduction in solution time. Furthermore, we analyze the optimal value of

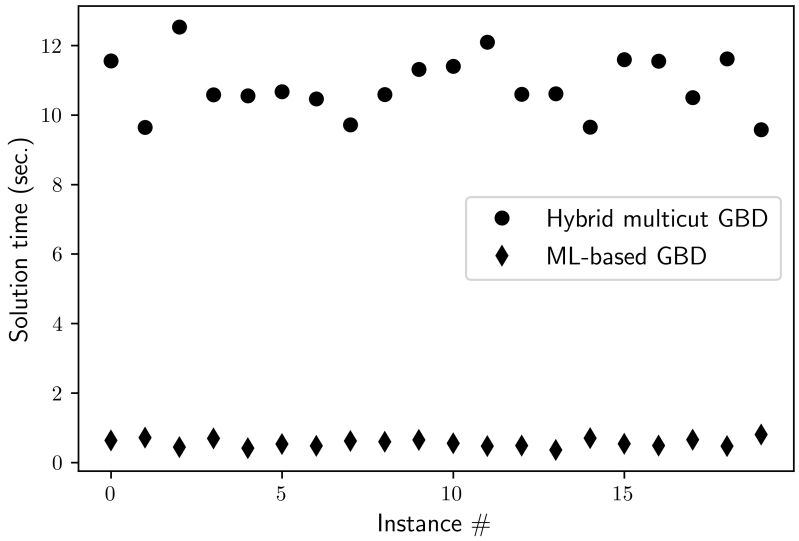


Figure 8.1: Solution time of the proposed method (Algorithm 8.1) and GBD from [201]

the objective function that is obtained with both methods. From Fig. 8.2 we observe that the percentage error is less than 0.1%, while the average error is 0.04%.

## 8.4 Application to mixed integer economic model predictive control

In this section, we apply the proposed approach for the solution of mixed integer model predictive control problems. We use the model presented in Chapter 6, where the operation of the system can be affected by disturbances in the scheduling (i.e., change in demand) or control level (i.e., change in inlet conditions). In this section, we present parts of the optimization model that are necessary for describing the application of the method. We refer the reader to Chapter 6 and [201] for a detailed explanation of the optimization model.

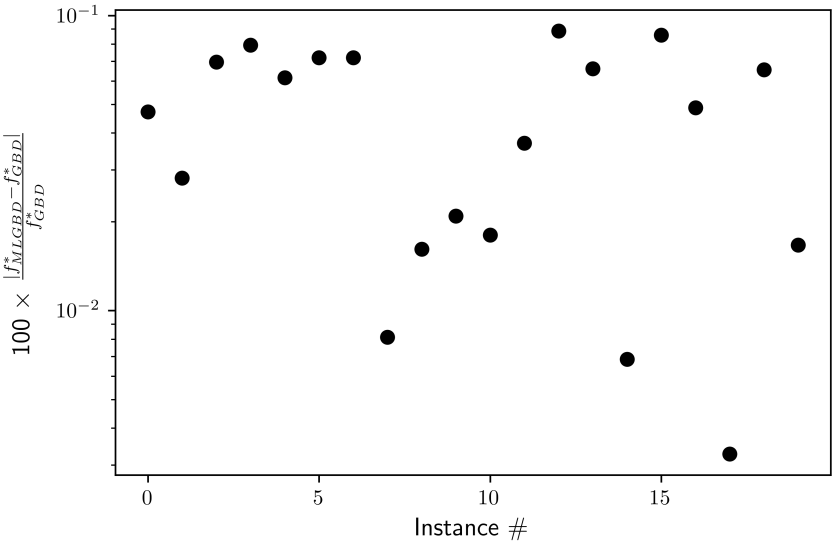


Figure 8.2: Percentage error between the optimal solution obtained with the proposed method ($f^*_{MLGBD}$) and GBD ($f^*_{GBD}$) proposed in [201].

### 8.4.1 Optimization model and decomposition-based solution

The mixed integer model predictive control problem has the following form

$$\begin{aligned} P(p, x_0) := \underset{w, \theta_{ijk}, \hat{\theta}_i}{\text{maximize}} \quad & \Phi_1(w; p) - \sum_{ijk} Z_{ijk} \mathcal{S}_{ijk}(\theta_{ijk}) - \sum_{i} \hat{Z}_i \hat{S}_i(\hat{\theta}, x_0) \\ \text{subject to} \quad & g_{sched}(w, \theta_{ijk}, \hat{\theta}_i; p) \leq 0 \end{aligned} \tag{8.21}$$

where $w$ are scheduling variables, $\theta_{ijk}$ is the transition time from product $i$ to $j$ in slot $k$, $\hat{\theta}_i$ is the transition time from the intermediate state $x_0$ to product $i$, $p$ are parameters of the optimization problem, $\mathcal{S}_{ijk}(\theta_{ijk})$ is equal to

$$\begin{aligned} \mathcal{S}_{ijk}(\theta_{ijk}) := \underset{x_{ijkfc}, u_{ijkfc}, \tilde{\theta}_{ijk}}{\text{minimize}} \quad & f^{ijk}_{dyn}(x_{ijkfc}, u_{ijkfc}, \tilde{\theta}_{ijk}) \\ \text{subject to} \quad & g_{dyn}(\tilde{\theta}_{ijk}, x_{ijkfc}, u_{ijkfc}) \leq 0 \\ & \tilde{\theta}_{ijk} = \theta_{ijk} \quad : \lambda_{ijk}, \end{aligned} \tag{8.22}$$

and $\hat{\mathcal{S}}_i(\hat{\theta}_i, x_0)$ is equal to

$$\begin{aligned}\hat{\mathcal{S}}_i(\hat{\theta}_i, x_0) := \underset{\hat{x}_{ifc}, \hat{u}_{ifc}, \check{\theta}_i}{\text{minimize}} \quad & f_{dyn}^{ijk}(\hat{x}_{ifc}, \hat{u}_{ifc}, \check{\theta}_i) \\ \text{subject to} \quad & g_{dyn}(\check{\theta}_i, \hat{x}_{ifc}, \hat{u}_{ifc}) \leq 0 \\ & \check{\theta}_i = \hat{\theta}_i \quad : \hat{\lambda}_i(\hat{\theta}_i, x_0) \\ & x_{i11} = x_0\end{aligned} \tag{8.23}$$

The value functions can be approximated via Benders cuts as follows

$$\begin{aligned}\eta_{ijk} \geq & \phi_{ijk}(\bar{\theta}_{ijk}^l) - \lambda_{ijk}^l(\bar{\theta}_{ijk}^l)(\theta_{ijk} - \bar{\theta}_{ijk}^l) \\ \hat{\eta}_i \geq & \hat{\phi}_i(\bar{\hat{\theta}}_i^l, x_0) - \hat{\lambda}_i^l(\bar{\hat{\theta}}_i^l, x_0)(\hat{\theta}_i - \bar{\hat{\theta}}_i^l),\end{aligned} \tag{8.24}$$

where $l \in \mathcal{L}$ denotes the number of points used to approximate the value functions. The original problem can be reformulated as

$$\begin{aligned}P(p, x_0) := \text{maximize} \quad & \Phi_1(w; p) - \sum_{ijk} Z_{ijk}\eta_{ijk} - \sum_i \hat{Z}_i\hat{\eta}_i \\ \text{subject to} \quad & g_{sched}(w, \theta_{ijk}, \hat{\theta}_i; p) \leq 0 \\ & \eta_{ijk} \geq \phi_{ijk}^l(\bar{\theta}_{ijk}^l) - \lambda_{ijk}^l(\theta_{ijk} - \bar{\theta}_{ijk}^l) \ \forall i, j, k, l \\ & \hat{\eta}_i \geq \hat{\phi}_i(\bar{\hat{\theta}}_i^l, x_0) - \hat{\lambda}_i^l(\bar{\hat{\theta}}_i^l, x_0)(\hat{\theta}_i - \bar{\hat{\theta}}_i^l) \ \forall i, l.\end{aligned} \tag{8.25}$$

Given this problem formulation, we can apply the proposed machine learning branch and check Generalized Benders Decomposition algorithm.

### 8.4.2 Learning the surrogate models

The learning procedure for the transitions between products is the same as the procedure presented in Setion 8.3.3 since the value functions depend only on the transition times $\theta_{ijk}$. We define $\tilde{\mathcal{S}}_{ijk}$ and $\tilde{\lambda}_{ijk}$ as the surrogate model used to predict the transition cost and Lagrangean multiplier for a transition from product $j$ to $j$ in slot $k$.

The value functions for the subproblems that consider the transition from the intermediate state $x_0$ to the steady state of a product depend on the transition time and $x_0$. For every product, we generate a random value for $x_0$ and we compute the minimum transition time $\hat{\theta}_{min}$. Next, we solve multiple dynamic optimization problems for different values of the transition time $\hat{\theta} \in \{\hat{\theta}_{min}, 2\hat{\theta}_{min}\}$ and obtain the transition cost.

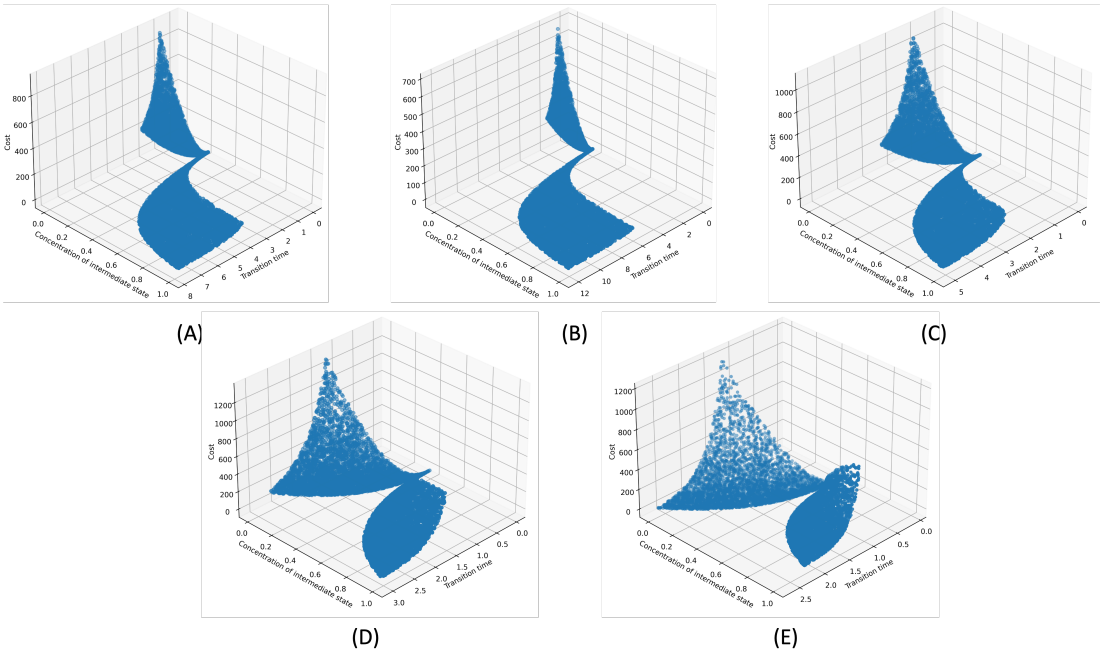


Figure 8.3: Transition cost from the intermediate state to the steady state of different products for different transition times. A, B, C, D, E refer to product 1,2,3,4,5 respectively. The x-axis is the transition time, the y-axis the initial concentration in the reactor ($x_0$), and the z-axis is the transition cost divided by 1000.

The value functions for the five products are presented in Fig. 8.3.

From this figure we observe that all the value functions have similar forms, there is a combination of initial concentration $x_0$ and transition time where the cost is very small. This corresponds to the case where $x_0$ is very close to the steady-state concentration of a product $x_{ss}$, thus the minimum transition time from $x_0$ to $x_{ss}$ and the associated cost are very small. The data presented in Fig. 8.3 can be used to learn a surrogate model, such as a neural network, $\tilde{\hat{S}}(\hat{\theta}, x_0)$ for the transition cost and $\tilde{\hat{\lambda}}(\hat{\theta}, x_0)$ for the Lagrangean multiplier. However, we observed that the accuracy of these surrogate models is low. To increase the accuracy of the prediction, for every product we create two surrogate models, one that is used when $x_0 > x_{ss}$ and one for $x_0 < x_{ss}$. In this approach, every surrogate model approximates a part of the value functions presented in Fig. 8.3. The procedure used to generate the data for learning the surrogate models and the implementation of the machine learning branch and check the Generalized Benders Decomposition algorithm are presented in Algorithm 8.2 and Algorithm 8.3.

Table 8.3: Soluton time statistics for different surrogate models for the solution of mixed integer MPC problems. NN-GBD, DT-GBD, RF-GBD refer to the implementation of the proposed algorithm using Neural Networks, Decision Trees, and Random Forests as surrogate models.

| | GBD | NN-GBD | DT-GBD | RF-GBD |
|---|---|---|---|---|
| Average solution time (sec) | 19.04 | 0.82 | 0.54 | 4.19 |
| Standard deviation | 2.74 | 0.25 | 0.15 | 1.40 |
| Average error | - | 0.19 | 0.16 | 0.18 |
| Average percentage reduction (%) | - | 95.69 | 97.16 | 78.00 |
| Fold reduction | - | 23 | 35 | 4.5 |

### 8.4.3 Computational results

We assume that the reactor is following a nominal schedule, obtained by the solution of the problem in the first part of the chapter, and at some time point in the first 10 hours of operation a disturbance changes in the inlet concentration in the reactor and the demand of the products. The probability distribution of the disturbance is the same as in the previous case study. We train three surrogate models, a Neural Network, a Decision Tree, and a Random forest, as presented in Algorithm 8.2. We generate 20 disturbances that were not used for training, to test the efficiency of the proposed approach. The solution time statistics are presented in Table 8.3 and a boxplot of the solution time is presented in Fig. 8.4. From the results, we observe that the proposed approach leads on average to 95% and 97% reduction in solution time when neural networks and random forests are used as surrogate models. Specifically, the average solution time of Generalized Benders Decomposition from [201] is 19.04 seconds, whereas the average solution time for the neural network, decision tree, and random forest is 0.82, 0.54, and 4.19 seconds respectively. These results show that the proposed approach can lead, on average, to a 35-fold reduction in solution time when the decision tree is used as a surrogate model.

Also, we analyze the error in the optimal solution that is obtained using Generalized Benders Decomposition from [201] and the proposed approach. The error is computed as follows

$$Error_i = 100 \times \frac{|f^*_{GBD} - f^*_{i-GBD}|}{f^*_{GBD}} \ \forall i = \{NN, DT, RF\}. \tag{8.26}$$

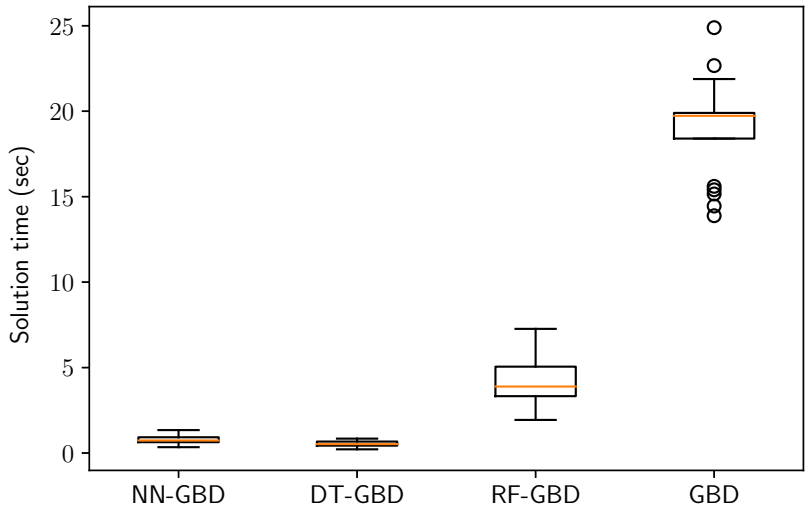


Figure 8.4: Boxplot of solution time for multicut Genralized Benders Decomposition from [201] and the proposed method using different surrogate models, Neural Network (NN-GBD), Decision Tree (DT-GBD), and Random Forest (RF-GBD).

The percentage error is presented in Fig. 8.5 and in Table 8.3. We observe that all the surrogate models have similar average error and the decision tree shows the minimum average error of 0.16%.

## 8.5 Conclusions

In this chapter, we proposed an ML-based branch and check Generalized Benders Decomposition algorithm for the efficient solution of a wide class of mixed integer programming problems that arise in mixed integer Model Predictive Control applications. The proposed approach combines branch and check Generalized Benders Decomposition with machine learning to reduce the computational time for the solution of the master problem and subproblem. The reduction in solution time with the proposed approach is achieved by approximating the information that is necessary to construct the Benders cuts, i.e., value function and Lagranean multipliers. Application to a problem in the operation of a chemical process shows that the proposed approach leads to a significant reduction in solution time while the solution accuracy is not significantly affected. These results show that the proposed approach can significantly accelerate the solution of mixed integer MPC problems, especially for cases where the underlying problem is

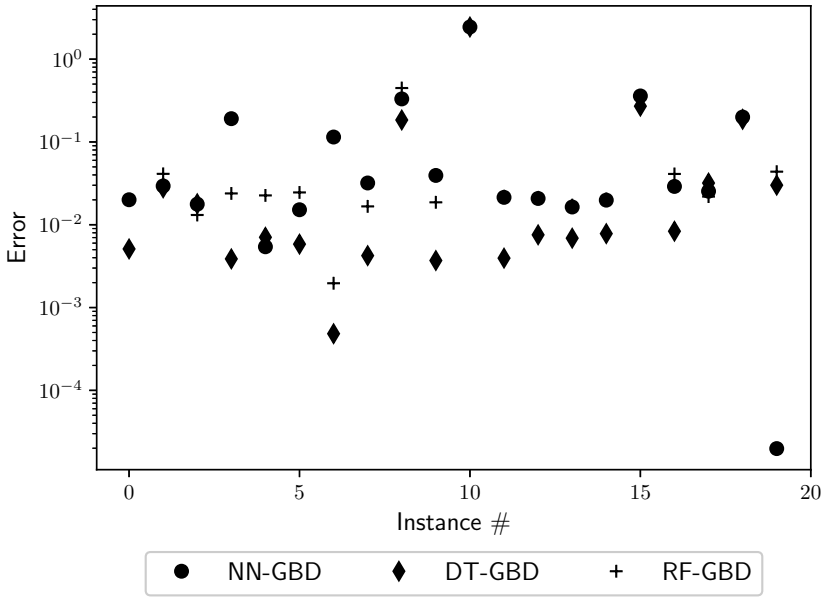


Figure 8.5: Percentage error between the optimal solution obtained using the multicut Genralized Benders Decomposition from [201] and the proposed method using different surrogate models, Neural Network (NN-GBD), Decision TRee (DT-GBD), and Random Forest (RF-GBD).

a Mixed Integer Nonlinear Programming problem while incurring small errors in the optimal solution of the problem. Finally, we make the following remarks.

**Remark 8.1.** In this case, the cuts that are approximated are optimality cuts since the subproblem is always feasible. Therefore, even if the prediction accuracy of the surrogate model is poor, the solution that will be obtained might be highly suboptimal, but feasible. This is important for online, and potentially, industrial applications where feasibility is usually more important than optimality.

**Remark 8.2.** We note that the proposed approach can be used for the solution of a problem where the subproblem might be infeasible for a given value of the complicating variables. However, the solution that is obtained is not guaranteed to be either optimal or feasible.

**Data:** Set of products $\mathcal{P}$, Steady-state concentration of products $x_{ss}$, Number of points $N_d$
**Result:** Surrogate models

**for** $p \in \mathcal{P}$ **do**
  epoch=0;
  $\mathcal{D}_l^c = \{\ \}$;
  $\mathcal{D}_u^c = \{\ \}$;
  $\mathcal{D}_l^\lambda = \{\ \}$;
  $\mathcal{D}_u^\lambda = \{\ \}$;
  **while** *epoch* $\leq N$ **do**
    Get random value for $x_0 \sim U(0,1)$;
    Get $\hat{\theta}_{min}$ from $x_0$ to $x_{ss}(p)$;
    Select randomly $\hat{\theta} \sim U(\hat{\theta}_{min}, 2\hat{\theta}_{min})$;
    Solve dynamic optimization problem from $x_0$ to $x_{ss}(p)$ with transition time $\hat{\theta}$ and obtain cost $c$ and Lagrangean multipliers $\lambda$;
    **if** $x_0 \leq x_{ss}(p)$ **then**
      Append data point with features $x_0, \hat{\theta}$ and label $c$ to dataset $\mathcal{D}_l$: $\mathcal{D}_l^c = \mathcal{D}_l^c \cup ((\hat{\theta}, x_0), c)$;
      Append data point with features $x_0, \hat{\theta}$ and label $\lambda$ to dataset $\mathcal{D}_l^\lambda$: $\mathcal{D}_l^\lambda = \mathcal{D}_l^\lambda \cup ((\hat{\theta}, x_0), \lambda)$;
    **else**
      Append data point with features $x_0, \hat{\theta}$ and label $c$ to dataset $\mathcal{D}_l$: $\mathcal{D}_u^c = \mathcal{D}_u^c \cup ((\hat{\theta}, x_0), c)$;
      Append data point with features $x_0, \hat{\theta}$ and label $\lambda$ to dataset $\mathcal{D}_u^\lambda$: $\mathcal{D}_u^\lambda = \mathcal{D}_l^\lambda \cup ((\hat{\theta}, x_0), \lambda)$;
    **end**
    epoch = epoch +1;
  **end**
  Learn surrogate model $\tilde{\hat{\mathcal{S}}}_p^u$ using data set $\mathcal{D}_u^c$;
  Learn surrogate model $\tilde{\hat{\mathcal{S}}}_p^l$ using data set $\mathcal{D}_l^c$;
  Learn surrogate model $\tilde{\hat{\lambda}}_p^u$ using data set $\mathcal{D}_u^\lambda$;
  Learn surrogate model $\tilde{\hat{\lambda}}_p^l$ using data set $\mathcal{D}_l^\lambda$;
**end**

**Algorithm 8.2:** Learning surrogate models for approximating value function and Lagrangean multipliers for transition from intermediate state to the steady state of the different products

**Data:** Master problem, surrogate models $\tilde{\hat{\mathcal{S}}}_p^u$, $\tilde{\hat{\mathcal{S}}}_p^l$, $\tilde{\hat{\lambda}}_p^u$, $\tilde{\hat{\lambda}}_p^l$, $x_0$, $x_{ss}$
**Result:** Solution of the optimization problem

Start Branch and Bound algorithm and obtain open nodes $R$;
**while** $R \neq \emptyset$ **do**
  Select a node $\rho$ from $R$ using node selection techniques;
  Solve continuous relaxation at $\rho$ and obtain $w^\rho$, $\theta_{ijk}^\rho$, $\hat{\theta}_i^\rho$;
  **if** *Binary variables are integer* **then**
    Approximate the value function and Lagrange multipliers for transitions between products that occur $\mathcal{C}_c = \{(i,j,k)|Z_{ijk} = 1\}$, $\tilde{\mathcal{S}}(\theta_{ijk})$, $\tilde{\lambda}(\theta_{ijk})$;
    Add Benders cut to all open nodes in $R$;

$$\eta \geq \tilde{S}(\theta_{ijk}^\rho) - \tilde{\lambda}(\theta_{ijk}^\rho)(\theta_{ijk} - \theta_{ijk}^\rho) \quad \forall (i,j,k) \in \mathcal{C}_c$$

    Determine the product $p$ that is manufactured in the first slot;
    Obtain transition time $\hat{\theta}_p^\rho$;
    **if** $x_0 \leq x_{ss}(p)$ **then**
      Approximate the value function and Lagrange multipliers using $\tilde{\hat{\mathcal{S}}}_p^l, \tilde{\hat{\lambda}}_p^l$ for the transition from $x_0$ to $p$ and add Benders cut to all open nodes in $R$;

$$\hat{\eta}_p \geq \tilde{\hat{S}}_p^l(\hat{\theta}_p^\rho) - \tilde{\hat{\lambda}}_p^l(\hat{\theta}_p^\rho)(\hat{\theta}_p - \hat{\theta}_p^\rho)$$

    **else**
      Approximate the value function and Lagrange multipliers using $\tilde{\hat{\mathcal{S}}}_p^u, \tilde{\hat{\lambda}}_p^u$ for the transition from $x_0$ to $p$ and add Benders cut to all open nodes in $R$;

$$\hat{\eta}_p \geq \tilde{\hat{S}}_p^u(\hat{\theta}_p^\rho) - \tilde{\hat{\lambda}}_p^u(\hat{\theta}_p^\rho)(\hat{\theta}_p - \hat{\theta}_p^\rho)$$

    **end**
    Add MIP-based cuts Update existing nodes in $R$;
  **else**
    Partition domain of $y$ variables;
    Add MIP-based cuts;
    Update existing nodes in $R$;
  **end**
**end**

**Algorithm 8.3:** Machine Learning based Branch and Check Generalized Benders Decomposition for mixed-integer model predictive control for the CSTR case

# Chapter 9
# Conclusions and Future directions

This thesis is focused on the solution of complex and large-scale optimization problems using decomposition-based solution methods. The first part was focused on developing an automated framework, based on artificial intelligence and network science, for the implementation of decomposition-based solution algorithms for the solution of generic optimization problems. The second part of the thesis was focused on using the structure detection results to improve the computational performance of decomposition-based solution methods by (1) reformulating an optimization problem such that its structure is ideal for the application of decomposition-based solution methods, and (2) using machine learning to accelerate specific steps of the algorithm. The framework developed in this thesis, first shows that artificial intelligence can accelerate decomposition-based solution methods and opens the way to explaining the computational performance of decomposition-based solution algorithms. Also, it highlights the need for better labeling algorithms which will reduce the computational time associated with the generation of training datasets and thus will reduce the time needed to develop such artificial intelligence-based acceleration techniques. We expand on these points in the rest of this chapter. It is hoped that the methods proposed here and the following discussion are of interest to the optimization community and practitioners who use optimization and may inspire beneficial further explorations.

## 9.1 Understanding optimization algorithms

Optimization solvers, both monolithic and decomposition-based, are generally treated as black-box systems whose exact implementation is not known to the user. Explaining and understanding the computational performance of a solver for a given optimization problem and a set of parameters can provide insights regarding the efficiency and limitations of the solver and can decode the role of the problem formulation on the computational performance. This is a complex task, since solvers have multiple algorithmic steps, employ a number of heuristics, and the problem itself might have a large number of variables and constraints. Thus automated methods are essential to understand and improve the problem formulation and the optimization algorithm itself. One approach to achieve this is to use tools from explainable artificial intelligence (xAI) to explain the

predictions of a surrogate model [9, 247, 118] that predicts the computational performance of an algorithm for a given problem. However, the application of these methods requires the appropriate abstraction of an optimization problem such that it can be used as the basis for machine learning models. This abstraction should capture detailed information about the structural and functional coupling among the variables and the constraints of the problem. The results from Chapter 2 show that a graph neural network can be used as a surrogate model to make predictions about the performance of optimization algorithms while considering detailed structural and functional coupling among the variables and the constraints of the problem. Therefore, this approach opens the path to explaining the computational performance of optimization algorithms in general as presented in the following future research directions.

**Future direction 1: Why should a decomposition-based solution algorithm be used?** The graph classification approach proposed in Chapter 2 can predict whether a decomposition-based solution algorithm should be selected over a monolithic one. Given this approach, a natural question to ask is: *Given an optimization problem, why is it solved faster with a decomposition-based solution algorithm?* Although, at first, someone might try to use the number of variables and constraints to answer this question, these features can not capture the inherent complexity of decomposition-based solution methods and detailed information about the problem formulation. The performance of decomposition-based solution algorithms depends on multiple factors, such as problem decomposition, coordination scheme, initialization, and problem formulation. Given the graph neural network approach used for algorithm configuration, one can utilize recent advances in explainable artificial intelligence for graph neural networks [294, 184, 135, 87], adapt them for the case of optimization algorithms, and use them for understanding why a given problem is solved faster using a decomposition-based solution algorithm.

**Future direction 2: Algorithm selection for generic optimization problems** The work presented in Chapter 2 lays the foundation for selecting the best solution strategy for generic optimization problems that arise in chemical engineering applications. As discussed in Chapter 2, the selection of the best solution strategy, with standard machine learning models, requires the extraction of a set of features. However, obtaining a handcrafted set of features is time-consuming and requires domain knowledge about the

underlying problem and the solution algorithm. Furthermore, developing such features for nonlinear and mixed integer nonlinear optimization problems is nontrivial. However, the graph neural network approach presented in Chapter 2 can be easily extended to consider detailed functional information about two or more variables in a constraint, such as affine, product, power, unary, and reciprocal expression. Furthermore, different types of graphs can be used, such as bipartite variable-constraint, constraint, and variable graph. Given this representation, a graph classifier can be developed to predict the best solution strategy for linear (simplex or interior point method), mixed integer linear (branch and cut or decomposition-based method), nonlinear (interior point or active-set based method), and mixed integer nonlinear programming problems where different algorithms might employ different convex relaxations. Similar approaches can be used to identify the best solution strategy for the solution of stochastic programming problems, robust optimization, and disjunctive programming problems.

**Future direction 3: Selection of problem formulation and relaxations** The graph classification approach can also serve as the basis for selecting problem formulations or relaxations. The problem formulation determines the complexity of the optimization model, i.e., the number of variables and constraints, the interaction pattern among the variables and constraints, and the type of constraints, e.g., convex or nonconvex. This step is usually guided by intuition and for a given decision-making problem multiple problem formulations may be possible [212, 268, 101, 169]. Determining the computational performance of an optimization algorithm for a given problem formulation can be time-consuming since the problem must first be solved to optimality. Furthermore, for complex decision-making problems, determining which part of the problem formulation mostly affects the solution time is nontrivial and requires detailed knowledge of optimization theory, algorithms, and their implementation. Identifying the best problem formulation, reformulation or relaxation can be important for the solution of online-optimization problems, such as production scheduling [270, 268, 101, 169], solution of disjunctive programming problems [278], and in global optimization algorithms [151].

## 9.2 Accelerating learning efficiency for online configuration of decomposition-based methods

In Chapters 2, 6, 8 different machine learning approaches were proposed to determine when to use a decomposition-based solution method and how to accelerate decomposition-based solution methods. These methods require data that capture the computational performance of an algorithm for a given problem. Obtaining such data can be computationally expensive, thus efficient sampling methods are necessary to guide the data generation process. The results in Chapter 6 show that pool-based active learning with uncertainty-based sampling can guide the data generation process. However, based on the application, other approaches can be followed, such as stream-based sampling. Although active learning can reduce the computational time for generating the training set, a potentially large number of data points must be labeled. This can be avoided using self-supervised learning [18], where given a small dataset of labeled data, the model itself learns representations that can be potentially used for other tasks, such as classification (e.g., predicting the best solution strategy) or regression (e.g., predicting solution time). Finally, transfer learning [289] can be used to reduce the size of the training set when such machine learning models are developed for mixed integer nonlinear programming problems which are particularly difficult to solve. This transfer learning approach can potentially exploit the similarities during the solution of mixed integer optimization problems where some aspects of the solution strategy like branching are common in both the case of MILP and MINLP problems. The development and adaptation of active learning, semi-supervised, and transfer learning approaches to optimization problems can reduce the computational time required for generating training datasets and can open new research directions as presented in the rest of this section.

**Future direction 5: Optimal decomposition of mathematical optimization problems** The results in Chapters 3, 4, 5 show that methods from network science can be used to learn the underlying structure of an optimization problem which can subsequently be used as the basis for decomposition-based solution algorithms. Despite the improvements in solution time documented through the case studies, the decomposition is optimal from a network science perspective. However, the structure detection algorithm does not differentiate between continuous/integer variables, convex/nonconvex

constraints, etc. Therefore, the decomposition that is obtained is not guaranteed to be optimal with respect to the solution time or convergence rate of the decomposition-based solution algorithm. Finding the optimal decomposition of an optimization problem for the implementation of a decomposition algorithm can be posed as an algorithm configuration problem as follows:

**Problem 9.1.** (Optimal decomposition) Given an optimization problem $p$ what is the optimal partition of the variables/constraints $b$ such that a desired objective $f$ (such as CPU time or optimality gap for a given CPU time) is minimized when the problem is solved using decomposition based solution algorithm $\alpha$, i.e., $b^* \in \arg\min_b \ f_\alpha(b)$

As discussed in Chapter 6 this is a black-box optimization problem. A possible solution to this problem is to use a graph neural network to predict the solution time of a given decomposition, which corresponds to differents partitions of the graph representation of the problem. Therefore, the block affiliation of every variable and constraint can be used as an extra feature for the nodes in the graph. However, a large training dataset is necessary since for a given problem, multiple decompositions must be evaluated. Therefore, the application of active learning, semi-supervised, and transfer learning, can aid the efficient generation of training dataset and the learned model can be subsequently used to identify the optimal decomposition of a given problem for a given decomposition-based solution algorithm.

**Future direction 6: Reconstructing the topology of black-box optimization problems via Bayesian Stochastic Blockmodeling.** The work presented in this thesis assumes that the graph representation of an optimization problem, i.e., the optimization model is given. Yet, this is assumption not necessarily true for all applications, since the underlying system might not be known exactly, and thus the optimization problem might a black-box one. Learning the structure of the black-box problem can potentially lead to a reduction in the number of required function evaluations. However, generating data for learning the structure of the problem can be costly. Recently it has been shown that learning simultaneously the topology and structure of a network [239, 242] improves the accuracy of the reconstruction, compared to only learning the topology. This is achieved using Stochastic blockmodeling and simultaneously learning a Bayesian network and the block affiliation of the nodes. This approach can be used to learn the interaction pattern between the variables of a black-box optimization

problem as well as the structure, which can be subsequently used for the application of decomposition-based solution methods.

# References


[1] *MINLPLib: A library of mixed-integer and continuous nonlinear programming instances*, 2018.

[2] *NLP and MINLP Test Problems*, 2022.

[3] F. Albalawi, H. Durand, and P. D. Christofides, *Distributed economic model predictive control for operational safety of nonlinear processes*, AIChE J., 63 (2017), pp. 3404–3418.

[4] A. Allman, M. J. Palys, and P. Daoutidis, *Scheduling-informed optimal design of systems with time-varying operation: A wind-powered ammonia case study*, AIChE Journal, 65 (2019), p. e16434.

[5] A. Allman, W. Tang, and P. Daoutidis, *Towards a generic algorithm for identifying high-quality decompositions of optimization problems*, in Computer Aided Chemical Engineering, vol. 44, Elsevier, 2018, pp. 943–948.

[6] ——, *Decode: a community-based algorithm for generating high-quality decompositions of optimization problems*, Optimization and Engineering, 20 (2019), pp. 1067–1084.

[7] C. J. Anderson, S. Wasserman, and K. Faust, *Building stochastic blockmodels*, Soc. Netw., 14 (1992), pp. 137–161.

[8] D. Angluin, *Queries and concept learning*, Machine learning, 2 (1988), pp. 319–342.

[9] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, et al., *Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI*, Information fusion, 58 (2020), pp. 82–115.

[10] D. Axehill, T. Besselmann, D. M. Raimondo, and M. Morari, *A parametric branch and bound approach to suboptimal explicit hybrid mpc*, Automatica, 50 (2014), pp. 240–246.

[11] C. Aykanat, A. Pinar, and Ü. V. Çatalyürek, *Permuting sparse rectangular matrices into block-diagonal form*, SIAM Journal on scientific computing, 25 (2004), pp. 1860–1879.

[12] H. Aytug, *Feature selection for support vector machines using generalized benders decomposition*, Eur. J. Oper. Res., 244 (2015), pp. 210–218.

[13] M. Bagajewicz and V. Manousiouthakis, *On the generalized benders decomposition*, Comput. Chem. Eng., 15 (1991), pp. 691–700.

[14] M.-F. Balcan, T. Dick, T. Sandholm, and E. Vitercik, *Learning to branch*, in International conference on machine learning, PMLR, 2018, pp. 344–353.

[15] M. Baldea and P. Daoutidis, *Dynamics and nonlinear control of integrated process systems*, Cambridge University Press, 2012.

[16] M. Baldea, J. Du, J. Park, and I. Harjunkoski, *Integrated production scheduling and model predictive control of continuous processes*, AIChE J., 61 (2015), pp. 4179–4190.

[17] M. Baldea and I. Harjunkoski, *Integrated production scheduling and process control: A systematic review*, Comput. Chem. Eng., 71 (2014), pp. 377–390.

[18] R. Balestriero, M. Ibrahim, V. Sobal, A. Morcos, S. Shekhar, T. Goldstein, F. Bordes, A. Bardes, G. Mialon, Y. Tian, et al., *A cookbook of self-supervised learning*, arXiv preprint arXiv:2304.12210, (2023).

[19] R. Baltean-Lugojan, P. Bonami, R. Misener, and A. Tramontani, *Scoring positive semidefinite cutting planes for quadratic optimization via trained neural networks*, preprint: http://www. optimization-online. org/DB_ HTML/2018/11/6943. html, (2019).

[20] A.-L. Barabási, *Network science*, Cambridge University Press, 2016.

[21] M. J. Barber, *Modularity and community detection in bipartite networks*, Phys. Rev. E, 76 (2007), p. 066102.

[22] T. Bartz-Beielstein and S. Markon, *Tuning search algorithms for real-world applications: A regression tree based approach*, in Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753), vol. 1, IEEE, 2004, pp. 1111–1118.

[23] S. Basso and A. Ceselli, *A data driven dantzig–wolfe decomposition framework*, Math. Prog. Comput., 15 (2023), pp. 153–194.

[24] S. Basso, A. Ceselli, and A. Tettamanzi, *Random sampling and machine learning to understand good decompositions*, Annals of Oper. Res., 284 (2020), pp. 501–526.

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

[26] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski, *Robust optimization*, vol. 28, Princeton university press, 2009.

[27] J. F. Benders, *Partitioning procedures for solving mixed-variables programming problems*, Numer. Math., 4 (1962), pp. 238–252.

[28] Y. Bengio, A. Lodi, and A. Prouvost, *Machine learning for combinatorial optimization: a methodological tour d'horizon*, Eur. J. Oper. Res., 290 (2021), pp. 405–421.

[29] B. Benyahia, R. Lakerveld, and P. I. Barton, *A plant-wide dynamic model of a continuous pharmaceutical process*, Ind. Eng. Chem. Res., 51 (2012), pp. 15393–15412.

[30] M. Bergner, A. Caprara, A. Ceselli, F. Furini, M. E. Lübbecke, E. Malaguti, and E. Traversi, *Automatic Dantzig-Wolfe reformulation of mixed integer programs*, Math. Prog., 149 (2015), pp. 391–424.

[31] D. Bertsimas, D. B. Brown, and C. Caramanis, *Theory and applications of robust optimization*, SIAM review, 53 (2011), pp. 464–501.

[32] D. Bertsimas and B. Stellato, *The voice of optimization*, Machine Learning, 110 (2021), pp. 249–277.

[33] ——, *Online mixed-integer optimization in milliseconds*, INFORMS J Comput ., 34 (2022), pp. 2229–2248.

[34] A. Bhosekar and M. G. Ierapetritou, *Advances in surrogate based modeling, feasibility analysis, and optimization: A review*, Comput. Chem. Eng., 108 (2018), pp. 250–267.

[35] D. Biagioni, P. Graf, X. Zhang, A. S. Zamzam, K. Baker, and J. King, *Learning-accelerated admm for distributed dc optimal power flow*, IEEE Control Syst. Lett., 6 (2020), pp. 1–6.

[36] L. T. Biegler, *An overview of simultaneous strategies for dynamic optimization*, Chemical Engineering and Processing: Process Intensification, 46 (2007), pp. 1043–1053.

[37] ——, *Nonlinear programming: concepts, algorithms, and applications to chemical processes*, SIAM, 2010.

[38] L. T. Biegler, I. E. Grossmann, and A. W. Westerberg, *Systematic methods for chemical process design*, Prentice Hall, 1997.

[39] L. T. Biegler, Y. Lang, and W. Lin, *Multi-scale optimization for process systems engineering*, Comput. Chem. Eng., 60 (2014), pp. 17–30.

[40] L. T. Biegler and V. M. Zavala, *Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization*, Comput. Chem. Eng., 33 (2009), pp. 575–582.

[41] J. R. Birge and F. Louveaux, *Introduction to stochastic programming*, Springer, 2nd ed., 2011.

[42] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, *Fast unfolding of communities in large networks*, J. Stat. Mech. Theor. Exp., 2008 (2008), p. P10008.

[43] P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, et al., *An algorithmic framework for convex mixed integer nonlinear programs*, Discrete optimization, 5 (2008), pp. 186–204.

[44] P. Bonami, A. Lodi, and G. Zarpellon, *A classifier to decide on the linearization of mixed-integer quadratic problems in cplex*, Oper. Res., (2022).

[45] L. Bottou, F. E. Curtis, and J. Nocedal, *Optimization methods for large-scale machine learning*, SIAM Review, 60 (2018), pp. 223–311.

[46] F. Boukouvala and C. A. Floudas, *ARGONAUT: AlgoRithms for Global Optimization of coNstrAined grey-box compUTational problems*, Optim. Lett., 11 (2017), pp. 895–913.

[47] F. Boukouvala, R. Misener, and C. A. Floudas, *Global optimization advances in mixed-integer nonlinear programming, minlp, and constrained derivative-free optimization, cdfo*, Eur. J. Oper. Res., 252 (2016), pp. 701–727.

[48] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al., *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Found. Trends. Mach. Learn., 3 (2011), pp. 1–122.

[49] W. Bradley, J. Kim, Z. Kilwein, L. Blakely, M. Eydenberg, J. Jalvin, C. Laird, and F. Boukouvala, *Perspectives on the integration between first-principles and data-driven modeling*, Comput. Chem. Eng., (2022), p. 107898.

[50] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, *Geometric deep learning: Grids, groups, graphs, geodesics, and gauges*, arXiv preprint arXiv:2104.13478, (2021).

[51] R. Bunel, A. De Palma, A. Desmaison, K. Dvijotham, P. Kohli, P. Torr, and M. P. Kumar, *Lagrangian decomposition for neural network verification*, in Conference on Uncertainty in Artificial Intelligence, PMLR, 2020, pp. 370–379.

[52] B. Burnak, J. Katz, N. A. Diangelakis, and E. N. Pistikopoulos, *Simultaneous process scheduling and control: a multiparametric programming-based approach*, Ind. Eng. Chem. Res., 57 (2018), pp. 3963–3976.

[53] M. R. Bussieck, A. S. Drud, and A. Meeraus, *Minlplib—a collection of test models for mixed-integer nonlinear programming*, INFORMS J. Comput., 15 (2003), pp. 114–119.

[54] A. Caspari, C. Tsay, A. Mhamdi, M. Baldea, and A. Mitsos, *The integration of scheduling and control: Top-down vs. bottom-up*, J Process Control, 91 (2020), pp. 50–62.

[55] P. M. Castro, I. E. Grossmann, and Q. Zhang, *Expanding scope and computational challenges in process scheduling*, Comput. Chem. Eng., 114 (2018), pp. 14–42.

[56] A. Cauligi, P. Culbertson, E. Schmerling, M. Schwager, B. Stellato, and M. Pavone, *Coco: Online mixed-integer control via supervised learning*, IEEE Robot. Autom, 7 (2021), pp. 1447–1454.

[57] V. M. Charitopoulos, V. Dua, and L. G. Papageorgiou, *Traveling salesman problem-based integration of planning, scheduling, and optimal control for continuous processes*, Ind. Eng. Chem. Res., 56 (2017), pp. 11186–11205.

[58] T. Chen, X. Chen, W. Chen, H. Heaton, J. Liu, Z. Wang, and W. Yin, *Learning to optimize: A primer and a benchmark*, arXiv preprint arXiv:2103.12828, (2021).

[59] W. Chen, Z. Shao, K. Wang, X. Chen, and L. T. Biegler, *Random sampling-based automatic parameter tuning for nonlinear programming solvers*, Ind. Eng. Chem. Res., 50 (2011), pp. 3907–3918.

[60] C. Chi, A. M. Aboussalah, E. B. Khalil, J. Wang, and Z. Sherkat-Masoumi, *A deep reinforcement learning framework for column generation*, arXiv preprint arXiv:2206.02568, (2022).

[61] P. D. Christofides, R. Scattolini, D. Muñoz de la Peña, and J. Liu, *Distributed model predictive control: A tutorial review and future research directions*, Comput. Chem. Eng., 51 (2013), pp. 21–41.

[62] Y. Chu and F. You, *Integration of scheduling and control with online closed-loop implementation: Fast computational strategy and large-scale global optimization algorithm*, Comput. Chem. Eng., 47 (2012), pp. 248–268.

[63] ——, *Integrated scheduling and dynamic optimization of complex batch processes with general network structure using a generalized benders decomposition approach*, Ind. Eng. Chem. Res., 52 (2013), pp. 7867–7885.

[64] ——, *Integration of production scheduling and dynamic optimization for multi-product CSTRs: Generalized Benders decomposition coupled with global mixed-integer fractional programming*, Comput. Chem. Eng., 58 (2013), pp. 315–333.

[65] ——, *Integrated planning, scheduling, and dynamic optimization for batch processes: Minlp model formulation and efficient solution methods via surrogate modeling*, Ind. Eng. Chem. Res., 53 (2014), pp. 13391–13411.

[66] ——, *Moving horizon approach of integrating scheduling and control for sequential batch processes*, AIChE J., 60 (2014), pp. 1654–1671.

[67] ——, *Model-based integration of control and operations: Overview, challenges, advances, and opportunities*, Comput. Chem. Eng., 83 (2015), pp. 2–20.

[68] D. Cohn, L. Atlas, and R. Ladner, *Improving generalization with active learning*, Machine learning, 15 (1994), pp. 201–221.

[69] A. J. Conejo, E. Castillo, R. Minguez, and R. Garcia-Bertrand, *Decomposition techniques in mathematical programming: engineering and science applications*, Springer Science & Business Media, 2006.

[70] A. J. Conejo, F. J. Nogales, and F. J. Prieto, *A decomposition procedure based on approximate newton directions*, Math. Prog., 93 (2002), pp. 495–515.

[71] M. Conforti, G. Cornuéjols, G. Zambelli, et al., *Integer programming*, vol. 271, Springer, 2014.

[72] B. A. Conway, *A survey of methods available for the numerical optimization of continuous dynamic systems*, J. Optim. Theor. Appl., 152 (2012), pp. 271–306.

[73] A. Cozad, N. V. Sahinidis, and D. C. Miller, *Learning surrogate models for simulation-based optimization*, AIChE J., 60 (2014), pp. 2211–2227.

[74] T. G. Crainic, W. Rei, M. Hewitt, and F. Maggioni, *Partial Benders decomposition strategies for two-stage stochastic integer programs*, vol. 37, CIRRELT, 2016.

[75] P. Daoutidis, J. H. Lee, I. Harjunkoski, S. Skogestad, M. Baldea, and C. Georgakis, *Integrating operations and control: A perspective and roadmap for future research*, Comput. Chem. Eng., 115 (2018), pp. 179–184.

[76] P. Daoutidis, W. Tang, and A. Allman, *Decomposition of control and optimization problems by network structure: Concepts, methods, and inspirations from biology*, AIChE J., 65 (2019), p. e16708.

[77] P. Daoutidis, W. Tang, and S. S. Jogwar, *Decomposing complex plants for distributed control: Perspectives from network theory*, Comput. Chem. Eng., 114 (2018), pp. 43–51.

[78] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová, *Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications*, Phys. Rev. E, 84 (2011), p. 066106.

[79] K. Deng, Y. Sun, S. Li, Y. Lu, J. Brouwer, P. G. Mehta, M. Zhou, and A. Chakraborty, *Model predictive control of central chiller plant with thermal energy storage via dynamic programming and mixed-integer linear programming*, IEEE Trans. Autom. Sci., 12 (2014), pp. 565–579.

[80] G. Di Liberto, S. Kadioglu, K. Leo, and Y. Malitsky, *Dash: Dynamic approach for switching heuristics*, Eur. J. Oper. Res., 248 (2016), pp. 943–953.

[81] L. S. Dias and M. G. Ierapetritou, *Integration of scheduling and control under uncertainties: Review and challenges*, Chem. Eng. Res. Des., 116 (2016), pp. 98–113.

[82] ——, *From process control to supply chain management: An overview of integrated decision making strategies*, Comput. Chem. Eng., 106 (2017), pp. 826–835.

[83] J.-Y. Ding, C. Zhang, L. Shen, S. Li, B. Wang, Y. Xu, and L. Song, *Accelerating primal solution findings for mixed integer programs based on solution prediction*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, 2020, pp. 1452–1459.

[84] M. E. Dogan and I. E. Grossmann, *A decomposition method for the simultaneous planning and scheduling of single-stage continuous multiproduct plants*, Ind. Eng. Chem. Res., 45 (2006), pp. 299–315.

[85] J. Du, J. Park, I. Harjunkoski, and M. Baldea, *A time scale-bridging approach for integrating production scheduling and process control*, Comput. Chem. Eng., 79 (2015), pp. 59–69.

[86] M. A. Duran and I. E. Grossmann, *An outer-approximation algorithm for a class of mixed-integer nonlinear programs*, Math. Program., 36 (1986), pp. 307–339.

[87] A. Duval and F. D. Malliaros, *GraphSVX: Shapley value explanations for graph neural networks*, in Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2021, pp. 302–318.

[88] K. Eggensperger, M. Lindauer, and F. Hutter, *Pitfalls and best practices in algorithm configuration*, Journal of Artificial Intelligence Research, 64 (2019), pp. 861–893.

[89] M. M. El-Halwagi and V. Manousiouthakis, *Synthesis of mass exchange networks*, AIChE J., 35 (1989), pp. 1233–1244.

[90] M. Erdirik-Dogan and I. E. Grossmann, *Simultaneous planning and scheduling of single-stage multi-product continuous plants with parallel lines*, Comput. Chem. Eng., 32 (2008), pp. 2664–2683.

[91] M. C. Ferris and J. D. Horn, *Partitioning mathematical programs for parallel solution*, Math. Program., 80 (1998), pp. 35–61.

[92] M. Fey and J. E. Lenssen, *Fast graph representation learning with pytorch geometric*, arXiv preprint arXiv:1903.02428, (2019).

[93] M. L. Fisher, *An applications oriented guide to lagrangian relaxation*, Interfaces, 15 (1985), pp. 10–21.

[94] ——, *The lagrangian relaxation method for solving integer programming problems*, Manage Sci., 50 (2004), pp. 1861–1871.

[95] R. Fletcher and S. Leyffer, *Solving mixed integer nonlinear programs by outer approximation*, Math. Program., 66 (1994), pp. 327–349.

[96] O. E. Flippo and A. H. Rinnooy Kan, *Decomposition in general mathematical programming*, Math. Program., 60 (1993), pp. 361–382.

[97] A. Flores-Tlacuahuac and I. E. Grossmann, *Simultaneous cyclic scheduling and control of a multiproduct CSTR*, Ind. Eng. Chem. Res., 45 (2006), pp. 6698–6712.

[98] ——, *Simultaneous scheduling and control of multiproduct continuous parallel lines*, Ind. Eng. Chem. Res., 49 (2010), pp. 7909–7921.

[99] C. A. Floudas, *Nonlinear and mixed-integer optimization: fundamentals and applications*, Oxford University Press, 1995.

[100] C. A. Floudas and C. E. Gounaris, *A review of recent advances in global optimization*, J. Glob. Optim., 45 (2009), pp. 3–38.

[101] C. A. Floudas and X. Lin, *Continuous-time versus discrete-time approaches for scheduling of chemical processes: A review*, Comput. Chem. Eng., 28 (2004), pp. 2109–2129.

[102] S. Fortunato and D. Hric, *Community detection in networks: A user guide*, Phys. Rep., 659 (2016), pp. 1–44.

[103] P. I. Frazier, *A tutorial on bayesian optimization*, arXiv preprint arXiv:1807.02811, (2018).

[104] D. J. Garcia and F. You, *Supply chain design and optimization: Challenges and opportunities*, Comput. Chem. Eng., 81 (2015), pp. 153–170.

[105] A. M. Geoffrion, *Elements of large-scale mathematical programming part I: Concepts*, Manage Sci., 16 (1970), pp. 652–675.

[106] ——, *Elements of large scale mathematical programming part II: Synthesis of algorithms and bibliography*, Manage Sci., 16 (1970), pp. 676–691.

[107] ——, *Duality in nonlinear programming: a simplified applications-oriented development*, SIAM review, 13 (1971), pp. 1–37.

[108] ——, *Generalized Benders decomposition*, J. Optim. Theor. Appl., 10 (1972), pp. 237–260.

[109] A. Gleixner, G. Hendel, G. Gamrath, T. Achterberg, M. Bastubbe, T. Berthold, P. Christophel, K. Jarck, T. Koch, J. Linderoth, et al., *Miplib 2017: data-driven compilation of the 6th mixed-integer programming library*, Math. Prog. Comput., 13 (2021), pp. 443–490.

[110] A. Goldenberg, A. X. Zheng, S. E. Fienberg, E. M. Airoldi, et al., *A survey of statistical network models*, Found. Trend. Mach. Learn., 2 (2010), pp. 129–233.

[111] C. P. Gomes and B. Selman, *Algorithm portfolios*, Artif Intell, 126 (2001), pp. 43–62.

[112] J. Gong and F. You, *Optimal processing network design under uncertainty for producing fuels and value-added bioproducts from microalgae: Two-stage adaptive robust mixed integer fractional programming model and computationally efficient solution algorithm*, AIChE J., 63 (2017), pp. 582–600.

[113] D. Grattarola, D. Zambon, F. M. Bianchi, and C. Alippi, *Understanding pooling in graph neural networks*, I IEEE Trans Neural Netw Learn Syst, (2022).

[114] I. Grossmann, *Enterprise-wide optimization: A new frontier in process systems engineering*, AIChE J., 51, pp. 1846–1857.

[115] I. E. Grossmann, *Advances in mathematical programming models for enterprise-wide optimization*, Comput. Chem. Eng., 47 (2012), pp. 2–18.

[116] ——, *Global Optimization in engineering design*, Springer, 2013.

[117] I. E. Grossmann, R. M. Apap, B. A. Calfa, P. García-Herreros, and Q. Zhang, *Recent advances in mathematical programming techniques for the optimization of process systems under uncertainty*, Comput. Chem. Eng., 91 (2016), pp. 3–14.

[118] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, *A survey of methods for explaining black box models*, ACM computing surveys (CSUR), 51 (2018), pp. 1–42.

[119] M. Guignard and S. Kim, *Lagrangean decomposition: A model yielding stronger lagrangean bounds*, Mathematical programming, 39 (1987), pp. 215–228.

[120] A. Gupta and C. D. Maranas, *A hierarchical lagrangean relaxation procedure for solving midterm planning problems*, Ind. Eng. Chem. Res., 38 (1999), pp. 1937–1947.

[121] O. K. Gupta and A. Ravindran, *Branch and bound experiments in convex nonlinear integer programming*, Manage Sci., 31 (1985), pp. 1533–1546.

[122] P. Gupta, M. Gasse, E. Khalil, P. Mudigonda, A. Lodi, and Y. Bengio, *Hybrid models for learning to branch*, Advances in neural information processing systems, 33 (2020), pp. 18087–18097.

[123] P. Gupta, E. B. Khalil, D. Chetélat, M. Gasse, Y. Bengio, A. Lodi, and M. P. Kumar, *Lookback for learning to branch*, arXiv preprint arXiv:2206.14987, (2022).

[124] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual*, 2021. Accessed: 2021-08-31.

[125] M. A. Gutiérrez-Limón, A. Flores-Tlacuahuac, and I. E. Grossmann, *Minlp formulation for simultaneous planning, scheduling, and control of short-period single-unit processing systems*, Ind. Eng. Chem. Res., 53 (2014), pp. 14679–14694.

[126] A. Hagberg, P. Swart, and D. S Chult, *Exploring network structure, dynamics, and function using networkx*, tech. rep., Los Alamos National Lab., Los Alamos, NM (United States), 2008.

[127] W. Hamilton, Z. Ying, and J. Leskovec, *Inductive representation learning on large graphs*, Advances in neural information processing systems, 30 (2017).

[128] C. L. Hanselman and C. E. Gounaris, *A mathematical optimization framework for the design of nanopatterned surfaces*, AIChE Journal, 62 (2016), pp. 3250–3263.

[129] W. E. Hart, C. D. Laird, J.-P. Watson, D. L. Woodruff, G. A. Hackebeil, B. L. Nicholson, and J. D. Siirola, *Pyomo – optimization modeling in Python*, Springer, 2nd ed., 2017.

[130] M. M. F. Hasan, F. Boukouvala, E. L. First, and C. A. Floudas, *Nationwide, regional, and statewide $CO_2$ capture, utilization, and sequestration supply chain network optimization*, Ind. Eng. Chem. Res., 53 (2014), pp. 7489–7506.

[131] M. B. Hastings, *Community detection as an inference problem*, Phys. Rev. E, 74 (2006), p. 035102.

[132] P. Hespanhol, R. Quirynen, and S. Di Cairano, *A structure exploiting branch-and-bound algorithm for mixed-integer model predictive control*, in 2019 18th European Control Conference (ECC), IEEE, 2019, pp. 2763–2768.

[133] P. W. Holland, K. B. Laskey, and S. Leinhardt, *Stochastic blockmodels: First steps*, Soc. Netw., 5 (1983), pp. 109–137.

[134] L. Huang, J. Jia, B. Yu, B.-G. Chun, P. Maniatis, and M. Naik, *Predicting execution time of computer programs using sparse polynomial regression*, Advances in neural information processing systems, 23 (2010).

[135] Q. Huang, M. Yamada, Y. Tian, D. Singh, and Y. Chang, *GraphLime: Local interpretable model explanations for graph neural networks*, IEEE Trans Knowl Data Eng, (2022).

[136] Z. Huang, K. Wang, F. Liu, H.-L. Zhen, W. Zhang, M. Yuan, J. Hao, Y. Yu, and J. Wang, *Learning to select cuts for efficient mixed-integer programming*, Pattern Recognit., 123 (2022), p. 108353.

[137] F. Hutter, Y. Hamadi, H. H. Hoos, and K. Leyton-Brown, *Performance prediction and automated tuning of randomized and parametric algorithms*, in International conference on principles and practice of constraint programming, Springer, 2006, pp. 213–228.

[138] F. Hutter, H. H. Hoos, and K. Leyton-Brown, *Automated configuration of mixed integer programming solvers*, in International Conference on Integration of Artificial Intelligence (AI) and Oper. Res. (OR) Techniques in Constraint Programming, Springer, 2010, pp. 186–202.

[139] ——, *Sequential model-based optimization for general algorithm configuration*, in International conference on learning and intelligent optimization, Springer, 2011, pp. 507–523.

[140] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle, *ParamILS: an automatic algorithm configuration framework*, Journal of Artificial Intelligence Research, 36 (2009), pp. 267–306.

[141] F. Hutter, L. Xu, H. H. Hoos, and K. Leyton-Brown, *Algorithm runtime prediction: Methods & evaluation*, Artif Intell, 206 (2014), pp. 79–111.

[142] M. G. Ierapetritou and C. A. Floudas, *Effective continuous-time formulation for short-term scheduling. 1. Multipurpose batch processes*, Ind. Eng. Chem. Res., 37 (1998), pp. 4341–4359.

[143] G. Iommazzo, C. d'Ambrosio, A. Frangioni, and L. Liberti, *Learning to configure mathematical programming solvers by mathematical programming*, in International Conference on Learning and Intelligent Optimization, Springer, 2020, pp. 377–389.

[144] R. R. Iyer and I. E. Grossmann, *A bilevel decomposition algorithm for long-range planning of process networks*, Ind. Eng. Chem. Res., 37 (1998), pp. 474–481.

[145] J. Jalving, Y. Cao, and V. M. Zavala, *Graph-based modeling and simulation of complex systems*, arXiv preprint arXiv:1812.04983, (2018).

[146] J. Jalving, S. Shin, and V. M. Zavala, *A graph-based modeling abstraction for optimization: Concepts and implementation in plasmo. jl*, arXiv preprint arXiv:2006.05378, (2020).

[147] H. Jia and S. Shen, *Benders cut classification via support vector machines for solving two-stage stochastic programs*, INFORMS Journal on Optimization, 3 (2021), pp. 278–297.

[148] Y. Jiang, M. A. Rodriguez, I. Harjunkoski, and I. E. Grossmann, *Optimal supply chain design and management over a multi-period horizon under demand uncertainty. Part II: A Lagrangean decomposition algorithm*, Comput. Chem. Eng., 62 (2014), pp. 211–224.

[149] B. Karrer and M. E. J. Newman, *Stochastic blockmodels and community structure in networks*, Phys. Rev. E, 83 (2011), p. 016107.

[150] P. Kerschke, H. H. Hoos, F. Neumann, and H. Trautmann, *Automated algorithm selection: Survey and perspectives*, Evolutionary computation, 27 (2019), pp. 3–45.

[151] A. Khajavirad and N. V. Sahinidis, *A hybrid lp/nlp paradigm for global optimization relaxations*, Math. Prog. Comput., 10 (2018), pp. 383–421.

[152] E. Khalil, P. Le Bodic, L. Song, G. Nemhauser, and B. Dilkina, *Learning to branch in mixed integer programming*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30, 2016.

[153] T. Khaniyev, S. Elhedhli, and F. S. Erenay, *Structure detection in mixed-integer programs*, INFORMS J. Comput., 30 (2018), pp. 570–587.

[154] M. R. Kilinç and N. V. Sahinidis, *Exploiting integrality in the global optimization of mixed-integer nonlinear programming problems with baron*, Optim. Methods Softw., 33 (2018), pp. 540–562.

[155] S. H. Kim and F. Boukouvala, *Surrogate-based optimization for mixed-integer nonlinear problems*, Comput. Chem. Eng., 140 (2020), p. 106847.

[156] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980, (2014).

[157] T. N. Kipf and M. Welling, *Semi-supervised classification with graph convolutional networks*, arXiv preprint arXiv:1609.02907, (2016).

[158] E. Kondili, C. C. Pantelides, and R. W. H. Sargent, *A general algorithm for short-term scheduling of batch operations – I. MILP formulation*, Comput. Chem. Eng., 17 (1993), pp. 211–227.

[159] L. Kotthoff, *Algorithm selection for combinatorial search problems: A survey*, in Data mining and constraint programming, Springer, 2016, pp. 149–190.

[160] J. Kronqvist, A. Lundell, and T. Westerlund, *The extended supporting hyperplane algorithm for convex mixed-integer nonlinear programming*, J. Glob. Optim., 64 (2016), pp. 249–272.

[161] M. Kruber, M. E. Lübbecke, and A. Parmentier, *Learning when to use a decomposition*, in International conference on AI and OR techniques in constraint programming for combinatorial optimization problems, Springer, 2017, pp. 202–210.

[162] R. Kumar, M. J. Wenzel, M. N. ElBsat, M. J. Risbeck, K. H. Drees, and V. M. Zavala, *Dual dynamic programming for multi-scale mixed-integer MPC*, Comput. Chem. Eng., 148 (2021), p. 107265.

[163] L. Lao, A. Aguirre, A. Tran, Z. Wu, H. Durand, and P. D. Christofides, *CFD modeling and control of a steam methane reforming reactor*, Chem. Eng. Sci., 148 (2016), pp. 78–92.

[164] N. H. Lappas and C. E. Gounaris, *Multi-stage adjustable robust optimization for process scheduling under uncertainty*, AIChE J., 62 (2016), pp. 1646–1667.

[165] C. L. Lara and I. E. Grossmann, *Global optimization for a continuous location-allocation model for centralized and distributed manufacturing*, in Computer Aided Chemical Engineering, vol. 38, Elsevier, 2016, pp. 1009–1014.

[166] C. L. Lara, D. S. Mallapragada, D. J. Papageorgiou, A. Venkatesh, and I. E. Grossmann, *Deterministic electric power infrastructure planning: Mixed-integer programming model and nested decomposition algorithm*, Eur. J. Oper. Res., 271 (2018), pp. 1037–1054.

[167] D. B. Larremore, A. Clauset, and A. Z. Jacobs, *Efficiently inferring community structure in bipartite networks*, Phys. Rev. E, 90 (2014), p. 012805.

[168] E. Larsen, E. Frejinger, B. Gendron, and A. Lodi, *Fast continuous and integer l-shaped heuristics through supervised learning*, arXiv preprint arXiv:2205.00897, (2022).

[169] H. Lee and C. T. Maravelias, *Combining the advantages of discrete-and continuous-time scheduling models: Part 1. Framework and mathematical formulations*, Comput. Chem. Eng., 116 (2018), pp. 176–190.

[170] M. Lee, N. Ma, G. Yu, and H. Dai, *Accelerating generalized benders decomposition for wireless resource allocation*, IEEE Trans. Wirel., 20 (2020), pp. 1233–1247.

[171] D. D. Lewis and W. A. Gale, *A sequential algorithm for training text classifiers: Corrigendum and additional data*, in ACM SIGIR Forum, vol. 29, ACM New York, NY, USA, 1995, pp. 13–19.

[172] J. Li, S. E. Demirel, and M. M. F. Hasan, *Process synthesis using block superstructure with automated flowsheet generation and optimization*, AIChE J., 64 (2018), pp. 3082–3100.

[173] L.-J. Li and R.-J. Zhou, *Optimization of large-scale water transfer networks: Conic integer programming model and distributed parallel algorithms*, AIChE J., 63 (2017), pp. 1566–1581.

[174] X. Li, Y. Chen, and P. I. Barton, *Nonconvex generalized benders decomposition with piecewise convex relaxations for global optimization of integrated process design and operation problems*, Ind. Eng. Chem. Res., 51 (2012), pp. 7287–7299.

[175] X. Li, Q. Qu, F. Zhu, J. Zeng, M. Yuan, K. Mao, and J. Wang, *Learning to reformulate for linear programming*, arXiv preprint arXiv:2201.06216, (2022).

[176] Z. Li and M. G. Ierapetritou, *Integrated production planning and scheduling using a decomposition framework*, Chem. Eng. Sci., 64 (2009), pp. 3585–3597.

[177] ——, *Production planning and scheduling integration through augmented Lagrangian optimization*, Comput. Chem. Eng., 34 (2010), pp. 996–1006.

[178] J. Linderoth and S. Wright, *Decomposition algorithms for stochastic programming on a computational grid*, Comput Optim Appl, 24 (2003), pp. 207–250.

[179] D. Liu, M. Fischetti, and A. Lodi, *Learning to search in local branching*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, 2022, pp. 3796–3803.

[180] J. Liu, D. Muñoz de la Peña, and P. D. Christofides, *Distributed model predictive control of nonlinear process systems*, AIChE J., 55 (2009), pp. 1171–1184.

[181] J. Liu, N. Ploskas, and N. V. Sahinidis, *Tuning baron using derivative-free optimization algorithms*, J. Glob. Optim., 74 (2019), pp. 611–637.

[182] S. Liu, J. M. Pinto, and L. G. Papageorgiou, *A TSP-based milp model for medium-term planning of single-stage continuous multiproduct plants*, Ind. Eng. Chem. Res., 47 (2008), pp. 7733–7743.

[183] A. Lodi and G. Zarpellon, *On learning and branching: a survey*, Top, 25 (2017), pp. 207–236.

[184] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang, *Parameterized explainer for graph neural network*, Advances in neural information processing systems, 33 (2020), pp. 19620–19631.

[185] T. L. Magnanti and R. T. Wong, *Accelerating benders decomposition: Algorithmic enhancement and model selection criteria*, Oper. Res., 29 (1981), pp. 464–484.

[186] R. S. Mah, *Chemical process structures and information flows*, Elsevier, 2013.

[187] C. T. Maravelias, *General framework and modeling approach classification for chemical production scheduling*, AIChE J., 58 (2012), pp. 1812–1828.

[188] C. T. Maravelias and C. Sung, *Integration of production planning and scheduling: Overview, challenges and opportunities*, Comput. Chem. Eng., 33 (2009), pp. 1919–1930.

[189] A. Marousi and A. Kokossis, *On the acceleration of global optimization algorithms by coupling cutting plane decomposition algorithms with machine learning and advanced data analytics*, Comput. Chem. Eng., 163 (2022), p. 107820.

[190] D. Masti, T. Pippia, A. Bemporad, and B. De Schutter, *Learning approximate semi-explicit hybrid mpc with an application to microgrids*, IFAC-PapersOnLine, 53 (2020), pp. 5207–5212.

[191] R. D. McAllister and J. B. Rawlings, *Advances in mixed-integer model predictive control*, in 2022 American Control Conference (ACC), IEEE, 2022, pp. 364–369.

[192] S. Menta, J. Warrington, J. Lygeros, and M. Morari, *Learning solutions to hybrid control problems using benders cuts*, in Learning for Dynamics and Control, PMLR, 2020, pp. 118–126.

[193] M. D. Mesarovic, *Multilevel systems and concepts in process control*, Proc. IEEE, 58 (1970), pp. 111–125.

[194] M. D. Mesarovic, D. Macko, and Y. Takahara, *Theory of hierarchical, multilevel, systems*, vol. 68, Elsevier, 2000.

[195] N. F. Michelena and P. Y. Papalambros, *A hypergraph framework for optimal model-based decomposition of design problems*, Comput Optim Appl, 8 (1997), pp. 173–196.

[196] R. Misener and C. A. Floudas, *Antigone: algorithms for continuous/integer global optimization of nonlinear equations*, J. Glob. Optim., 59 (2014), pp. 503–526.

[197] S. Misra, L. Roald, and Y. Ng, *Learning for constrained optimization: Identifying optimal active constraint sets*, INFORMS J Comput ., 34 (2022), pp. 463–480.

[198] I. Mitrai and P. Daoutidis, *Decomposition of integrated scheduling and dynamic optimization problems using community detection*, J. Process Control, 90 (2020), pp. 63–74.

[199] ——, *Efficient solution of enterprise-wide optimization problems using nested stochastic blockmodeling*, Ind. Eng. Chem. Res., 60 (2021), pp. 14476–14494.

[200] ——, *An adaptive multi-cut decomposition based algorithm for integrated closed loop scheduling and control*, in Computer Aided Chemical Engineering, vol. 49, Elsevier, 2022, pp. 475–480.

[201] ——, *A multicut generalized benders decomposition approach for the integration of process operations and dynamic optimization for continuous systems*, Comput. Chem. Eng., 164 (2022), p. 107859.

[202] ——, *Efficient solution of mixed integer model predictive control problems via benders decomposition*, Under Review, (2023).

[203] ——, *A graph classification approach to determine when to decompose optimization problems*, in Computer Aided Chemical Engineering, vol. 52, Elsevier, 2023, pp. 655–660.

[204] ——, *Learning to initialize generalized benders decomposition via active learning*, in FOCAPO/CPC, San Antonio, Texas, 2023.

[205] ——, *Taking the human out of decomposition-based optimization via artificial intelligence: Part I. Learning when to decompose*, Under review, (2023).

[206] ——, *Taking the human out of decomposition-based optimization via artificial intelligence: Part II. Learning to initialize*, Under review, (2023).

[207] I. Mitrai, W. Tang, and P. Daoutidis, *Stochastic blockmodeling for learning the structure of optimization problems*, AIChE J., 68 (2022), p. e17415. DOI:10.1002/aic.17415.

[208] M. Mohideen, J. Perkins, and E. Pistikopoulos, *Towards an efficient numerical procedure for mixed integer optimal control*, Comput. Chem. Eng., 21 (1997), pp. S457–S462.

[209] D. Mora-Mariano, M. A. Gutiérrez-Limón, and A. Flores-Tlacuahuac, *A lagrangean decomposition optimization approach for long-term planning, scheduling and control*, Comput. Chem. Eng., 135 (2020), p. 106713.

[210] M. Morabit, G. Desaulniers, and A. Lodi, *Machine-learning–based column selection for column generation*, Transp. Sci., 55 (2021), pp. 815–831.

[211] M. Morari, Y. Arkun, and G. Stephanopoulos, *Studies in the synthesis of control structures for chemical processes: Part I: Formulation of the problem. process decomposition and the classification of the control tasks. analysis of the optimizing control structures*, AIChE J., 26 (1980), pp. 220–232.

[212] W. T. Morris, *On the art of modeling*, Manage Sci., 13 (1967), pp. B–707.

[213] S. Mouret, I. E. Grossmann, and P. Pestiaux, *A new Lagrangian decomposition approach applied to the integration of refinery planning and crude-oil scheduling*, Comput. Chem. Eng., 35 (2011), pp. 2750–2766.

[214] D. Müller, M. Illner, E. Esche, T. Pogrzeba, M. Schmidt, R. Schomäcker, L. T. Biegler, G. Wozny, and J.-U. Repke, *Dynamic real-time optimization under uncertainty of a hydroformylation mini-plant*, Comput. Chem. Eng., 106 (2017), pp. 836–848.

[215] V. Nair, S. Bartunov, F. Gimeno, I. von Glehn, P. Lichocki, I. Lobov, B. O'Donoghue, N. Sonnerat, C. Tjandraatmadja, P. Wang, et al., *Solving mixed integer programs using neural networks*, arXiv preprint arXiv:2012.13349, (2020).

[216] National Academies of Sciences Engineering and Medicine, *New directions for chemical engineering*, (2022).

[217] M. E. Newman and G. Reinert, *Estimating the number of communities in a network*, Phys. Rev. Lett., 117 (2016), p. 078301.

[218] M. E. J. Newman, *Equivalence between modularity optimization and maximum likelihood methods for community detection*, Phys. Rev. E, 94 (2016), p. 052315.

[219] M. E. J. Newman, *Networks*, Oxford University Press, 2nd ed., 2018.

[220] M. E. J. Newman and M. Girvan, *Finding and evaluating community structure in networks*, Phys. Rev. E, 69 (2004), p. 026113.

[221] Y. Nie, L. T. Biegler, C. M. Villa, and J. M. Wassick, *Discrete time formulation for the integration of scheduling and dynamic optimization*, Ind. Eng. Chem. Res., 54 (2015), pp. 4303–4315.

[222] Y. Nie, L. T. Biegler, and J. M. Wassick, *Integrated scheduling and dynamic optimization of batch processes using state equipment networks*, AIChE J., 58 (2012), pp. 3416–3432.

[223] R. Oberdieck and E. N. Pistikopoulos, *Explicit hybrid model-predictive control: The exact solution*, Automatica, 58 (2015), pp. 152–159.

[224] F. Oliveira, I. E. Grossmann, and S. Hamacher, *Accelerating Benders stochastic decomposition for the optimization under uncertainty of the petroleum product supply chain*, Comput. Oper. Res., 49 (2014), pp. 47–58.

[225] R. Pacqueau, F. Soumis, and L.-N. Hoang, *A fast and accurate algorithm for stochastic integer programming, applied to stochastic shift scheduling*, Groupe d'études et de recherche en analyse des décisions, 2012.

[226] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., *Pytorch: An imperative style, high-performance deep learning library*, Advances in neural information processing systems, 32 (2019).

[227] B. P. Patil, E. Maia, and L. A. Ricardez-Sandoval, *Integration of scheduling, design, and control of multiproduct chemical processes under uncertainty*, AIChE J., 61 (2015), pp. 2456–2470.

[228] R. Pattison, C. R. Touretzky, T. Johansson, M. Baldea, and I. Harjunkoski, *Moving horizon scheduling of an air separation unit under fast-changing energy prices*, IFAC-PapersOnLine, 49 (2016), pp. 681–686.

[229] R. C. Pattison, C. R. Touretzky, I. Harjunkoski, and M. Baldea, *Moving horizon closed-loop production scheduling using dynamic process models*, AIChE J., 63 (2017), pp. 639–651.

[230] R. C. Pattison, C. R. Touretzky, T. Johansson, I. Harjunkoski, and M. Baldea, *Optimal process operations in fast-changing electricity markets: framework for scheduling with low-order dynamic models and an air separation application*, Ind. Eng. Chem. Res., 55 (2016), pp. 4562–4584.

[231] M. B. Paulus, G. Zarpellon, A. Krause, L. Charlin, and C. Maddison, *Learning to cut by looking ahead: Cutting plane selection via imitation learning*, in International conference on machine learning, PMLR, 2022, pp. 17584–17600.

[232] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research, 12 (2011), pp. 2825–2830.

[233] T. P. Peixoto, *Entropy of stochastic blockmodel ensembles*, Physical Review E, 85 (2012), p. 056122.

[234] ——, *Efficient monte carlo and greedy heuristic for the inference of stochastic block models*, Physical Review E, 89 (2014), p. 012804.

[235] T. P. Peixoto, *The graph-tool python library*, 2014. Accessed: 2021-08-31.

[236] T. P. Peixoto, *Hierarchical block structures and high-resolution model selection in large networks*, Physical Review X, 4 (2014), p. 011047.

[237] ——, *Nonparametric bayesian inference of the microcanonical stochastic block model*, Phys. Rev. E, 95 (2017), p. 012317.

[238] ——, *Nonparametric weighted stochastic block models*, Physical Review E, 97 (2018), p. 012306.

[239] ——, *Reconstructing networks with unknown and heterogeneous errors*, Physical Review X, 8 (2018), p. 041011.

[240] ——, *Bayesian stochastic blockmodeling*, Advances in network clustering and blockmodeling, (2019), pp. 289–332.

[241] ——, *Bayesian stochastic blockmodeling*, in Advances in network clustering and blockmodeling, P. Doreian, V. Batagelj, and A. Ferligoj, eds., Wiley, 2019, ch. 11, pp. 289–332.

[242] ——, *Network reconstruction and community detection from dynamics*, Physical review letters, 123 (2019), p. 128301.

[243] J. Pihera and N. Musliu, *Application of machine learning to algorithm selection for tsp*, in 2014 IEEE 26th International Conference on Tools with Artificial Intelligence, IEEE, 2014, pp. 47–54.

[244] E. N. Pistikopoulos, A. Barbosa-Povoa, J. H. Lee, R. Misener, A. Mitsos, G. V. Reklaitis, V. Venkatasubramanian, F. You, and R. Gani, *Process systems engineering–the generation next?*, Comput. Chem. Eng., 147 (2021), p. 107252.

[245] E. N. Pistikopoulos and N. A. Diangelakis, *Towards the integration of process design, control and scheduling: Are we getting closer?*, Comput. Chem. Eng., 91 (2016), pp. 85–92.

[246] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: theory, computation, and design*, vol. 2, Nob Hill Publishing Madison, WI, 2017.

[247] M. T. Ribeiro, S. Singh, and C. Guestrin, *" why should i trust you?" explaining the predictions of any classifier*, in Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp. 1135–1144.

[248] J. R. Rice, *The algorithm selection problem*, in Advances in computers, vol. 15, Elsevier, 1976, pp. 65–118.

[249] M. J. Risbeck, C. T. Maravelias, and J. B. Rawlings, *Unification of closed-loop scheduling and control: State-space formulations, terminal constraints, and nominal theoretical properties*, Comput. Chem. Eng., 129 (2019), p. 106496.

[250] A. Ruszczyński and A. Świetanowski, *Accelerating the regularized decomposition method for two stage stochastic linear problems*, Eur. J. Oper. Res., 101 (1997), pp. 328–342.

[251] S. Sager, H. G. Bock, and M. Diehl, *The integer approximation error in mixed-integer optimal control*, Math. Program., 133 (2012), pp. 1–23.

[252] G. K. Saharidis, M. Boile, and S. Theofanis, *Initialization of the benders master problem using valid inequalities applied to fixed-charge network problems*, Expert Systems with Applications, 38 (2011), pp. 6627–6636.

[253] G. K. Saharidis and M. G. Ierapetritou, *Improving benders decomposition using maximum feasible subsystem (mfs) cut generation strategy*, Comput. Chem. Eng., 34 (2010), pp. 1237–1245.

[254] N. Sahinidis, *BARON user manual v. 2018.11.15.* `https://minlp.com/downloads/docs/baron%20manual.pdf`, 2020. retrieved on Dec. 3, 2018.

[255] N. V. Sahinidis, *BARON: A general purpose global optimization software package*, J. Glob. Optim., 8 (1996), pp. 201–205.

[256] N. V. Sahinidis and I. E. Grossmann, *Convergence properties of generalized Benders decomposition*, Comput. Chem. Eng., 15 (1991), pp. 481–491.

[257] J. Sansana, M. N. Joswiak, I. Castillo, Z. Wang, R. Rendall, L. H. Chiang, and M. S. Reis, *Recent trends on hybrid modeling for industry 4.0*, Comput. Chem. Eng., 151 (2021), p. 107365.

[258] E. Schede, J. Brandt, A. Tornede, M. Wever, V. Bengs, E. Hüllermeier, and K. Tierney, *A survey of methods for automated algorithm configuration*, arXiv preprint arXiv:2202.01651, (2022).

[259] A. M. Schweidtmann and A. Mitsos, *Deterministic global optimization with artificial neural networks embedded*, J. Optim. Theor. Appl., (2018), pp. 1–24.

[260] B. Settles, *Active learning literature survey*, (2009).

[261] B. Settles, M. Craven, and S. Ray, *Multiple-instance active learning*, Advances in neural information processing systems, 20 (2007).

[262] H. S. Seung, M. Opper, and H. Sompolinsky, *Query by committee*, in Proceedings of the fifth annual workshop on Computational learning theory, 1992, pp. 287–294.

[263] N. K. Shah and M. G. Ierapetritou, *Lagrangian decomposition approach to scheduling large-scale refinery operations*, Comput. Chem. Eng., 79 (2015), pp. 1–29.

[264] H. Shi, Y. Chu, and F. You, *Novel optimization model and efficient solution method for integrating dynamic optimization with process operations of continuous manufacturing processes*, Ind. Eng. Chem. Res., 54 (2015), pp. 2167–2187.

[265] K. Smith-Miles and L. Lopes, *Measuring instance difficulty for combinatorial optimization problems*, Computers & Oper. Res., 39 (2012), pp. 875–889.

[266] K. Smith-Miles and J. van Hemert, *Discovering the suitability of optimisation algorithms by learning from evolved instances*, Annals of Mathematics and Artificial Intelligence, 61 (2011), pp. 87–104.

[267] S. Sra, S. Nowozin, and S. J. Wright, *Optimization for machine learning*, Mit Press, 2012.

[268] H. Stefansson, S. Sigmarsdottir, P. Jensson, and N. Shah, *Discrete and continuous time representations and mathematical models for large production scheduling problems: A case study from the pharmaceutical industry*, Eur. J. Oper. Res., 215 (2011), pp. 383–392.

[269] L. Su, L. Tang, and I. E. Grossmann, *Computational strategies for improved minlp algorithms*, Comput. Chem. Eng., 75 (2015), pp. 40–48.

[270] A. Sundaramoorthy and C. T. Maravelias, *Computational study of network-based mixed-integer programming approaches for chemical production scheduling*, Ind. Eng. Chem. Res., 50 (2011), pp. 5023–5040.

[271] W. Tang, A. Allman, D. B. Pourkargar, and P. Daoutidis, *Optimal decomposition for distributed optimization in nonlinear model predictive control through community detection*, Comput. Chem. Eng., 111 (2018), pp. 43–54.

[272] Y. Tang, S. Agrawal, and Y. Faenza, *Reinforcement learning for integer programming: Learning to cut*, in International conference on machine learning, PMLR, 2020, pp. 9367–9376.

[273] M. Tawarmalani and N. V. Sahinidis, *A polyhedral branch-and-cut approach to global optimization*, Math. Program., 103 (2005), pp. 225–249.

[274] S. Terrazas-Moreno, A. Flores-Tlacuahuac, and I. E. Grossmann, *Simultaneous cyclic scheduling and optimal control of polymerization reactors*, AIChE J., 53 (2007), pp. 2301–2315.

[275] ——, *Lagrangean heuristic for the scheduling and control of polymerization reactors*, AIChE J., 54 (2008), pp. 163–182.

[276] S. Terrazas-Moreno, P. A. Trotter, and I. E. Grossmann, *Temporal and spatial lagrangean decompositions in multi-site, multi-period production planning problems with sequence-dependent changeovers*, Comput. Chem. Eng., 35 (2011), pp. 2913–2928.

[277] E. S. Thorsteinsson, *Branch-and-check: A hybrid framework integrating mixed integer programming and constraint logic programming*, in Principles and Practice of Constraint Programming—CP 2001: 7th International Conference, CP 2001 Paphos, Cyprus, November 26–December 1, 2001 Proceedings 7, Springer, 2001, pp. 16–30.

[278] F. Trespalacios and I. E. Grossmann, *Review of mixed-integer nonlinear and generalized disjunctive programming methods*, Chem. Ing. Tech., 86 (2014), pp. 991–1012.

[279] C. Tsay and M. Baldea, *Integrating production scheduling and process control using latent variable dynamic models*, Control Eng. Pract., 94 (2020), p. 104201.

[280] S. A. van den Heever, I. E. Grossmann, S. Vasantharajan, and K. Edwards, *A lagrangean decomposition heuristic for the design and planning of offshore hydrocarbon field infrastructures with complex economic objectives*, Ind. Eng. Chem. Res., 40 (2001), pp. 2857–2875.

[281] T. J. Van Roy, *Cross decomposition for mixed integer programming*, Math. Program., 25 (1983), pp. 46–63.

[282] T. Varelmann, J. I. Otashu, K. Seo, A. W. Lipow, A. Mitsos, and M. Baldea, *A decoupling strategy for protecting sensitive process information in cooperative optimization of power flow*, AIChE J., 68 (2022), p. e17429.

[283] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, *Graph attention networks*, arXiv preprint arXiv:1710.10903, (2017).

[284] J. P. Vielma, SIAM Review, 57 (2015), pp. 3–57.

[285] A. Wächter and L. T. Biegler, *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*, Math. Prog., 106 (2006), pp. 25–57.

[286] W. Wan, J. P. Eason, B. Nicholson, and L. T. Biegler, *Parallel cyclic reduction decomposition for dynamic optimization problems*, Comput. Chem. Eng., 120 (2019), pp. 54–69.

[287] J. Wang and T. Ralphs, *Computational experience with hypergraph-based methods for automatic decomposition in discrete optimization*, in Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems: 10th International Conference, CPAIOR 2013, Yorktown Heights, NY, USA, May 18-22, 2013. Proceedings 10, Springer, 2013, pp. 394–402.

[288] J. Warrington, *Learning continuous q-functions using generalized benders cuts*, in 2019 18th European Control Conference (ECC), IEEE, 2019, pp. 530–535.

[289] K. Weiss, T. M. Khoshgoftaar, and D. Wang, *A survey of transfer learning*, Journal of Big data, 3 (2016), pp. 1–40.

[290] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*, vol. 2, MIT press Cambridge, MA, 2006.

[291] D. P. Word, J. Kang, J. Akesson, and C. D. Laird, *Efficient parallel solution of large-scale nonlinear dynamic optimization problems*, Comput. Optim. Appl., 59 (2014), pp. 667–688.

[292] L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown, *Hydra-mip: Automated algorithm configuration and selection for mixed integer programming*, in RCRA workshop on experimental evaluation of algorithms for solving problems with combinatorial explosion at the international joint conference on artificial intelligence (IJCAI), 2011, pp. 16–30.

[293] Y. Ye, I. E. Grossmann, and J. M. Pinto, *Mixed-integer nonlinear programming models for optimal design of reliable chemical plants*, Comput. Chem. Eng., 116 (2018), pp. 3–16.

[294] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, *GNNexplainer: Generating explanations for graph neural networks*, Advances in neural information processing systems, 32 (2019).

[295] F. You and I. E. Grossmann, *Multicut benders decomposition algorithm for process supply chain planning under uncertainty*, Annals of Oper. Res., 210 (2013), pp. 191–211.

[296] Y. Yuan, Z. Li, and B. Huang, *Nonlinear robust optimization for process design*, AIChE J., 64 (2018), pp. 481–494.

[297] Z. Yuan, B. Chen, G. Sin, and R. Gani, *State-of-the-art and progress in the optimization-based simultaneous design and control for chemical processes*, AIChE J., 58 (2012), pp. 1640–1659.

[298] V. M. Zavala, C. D. Laird, and L. T. Biegler, *Interior-point decomposition approaches for parallel solution of large-scale nonlinear parameter estimation problems*, Chem. Eng. Sci., 63 (2008), pp. 4834–4845.

[299] B. Zeng and L. Zhao, *Solving two-stage robust optimization problems using a column-and-constraint generation method*, Oper. Res. Lett., 41 (2013), pp. 457–461.

[300] S. Zeng, A. Kody, Y. Kim, K. Kim, and D. K. Molzahn, *A reinforcement learning approach to parameter selection for distributed optimal power flow*, Electr. Power Syst. Res., 212 (2022), p. 108546.

[301] Q. Zhang and I. E. Grossmann, *Enterprise-wide optimization for industrial demand side management: Fundamentals, advances, and perspectives*, Chem. Eng. Res. Des., 116 (2016), pp. 114–131.

[302] X. Zhang, T. Martin, and M. E. J. Newman, *Identification of core-periphery structure in networks*, Phys. Rev. E, 91 (2015), p. 032803.

[303] J.-J. Zhu and G. Martius, *Fast non-parametric learning to accelerate mixed-integer programming for hybrid model predictive control*, IFAC-PapersOnLine, 53 (2020), pp. 5239–5245.

[304] J. Zhuge and M. Ierapetritou, *Simultaneous scheduling and control with closed loop implementation on parallel units*, in Proceedings Foundations of Computer-Aided Process Operations (FOCAPO) 2012, Proceedings Foundations of Computer-Aided Process Operations (FOCAPO), Savannah, USA, 2012.

[305] J. Zhuge and M. G. Ierapetritou, *Integration of scheduling and control with closed loop implementation*, Ind. Eng. Chem. Res., 51 (2012), pp. 8550–8565.

[306] ——, *Integration of scheduling and control for batch processes using multi-parametric model predictive control*, AIChE J., 60 (2014), pp. 3169–3183.

[307] ——, *An integrated framework for scheduling and control using fast model predictive control*, AIChE J., 61 (2015), pp. 3304–3319.

# Appendices

# Appendix A
# Supplementary material - Decomposition of integrated scheduling and dynamic optimization problems using community detection

## A.1 Formulation of the master and primal problems for the Benders decomposition for a single multiproduct CSTR

The formulation of the Benders decomposition algorithm is based on [99]. The problem has a hierarchical structure, with the scheduling problem in the first layer and the dynamic optimization subproblems in the second layer. The shared variables between the scheduling and dynamic optimization subproblem in slot $k$ are:

$$c_k^{in}, c_k^{end}, q_k^{in}, q_k^{end}, \theta_k^t, T_c$$

The primal problem in slot $k$ (dynamic optimization problem) is solved for fixed transition time, set-point values for the states and the manipulated variables and total

operating time. The formulation for slot $k$ is:

$$D_k : \max_{Q} -\frac{C^{feed}}{\bar{T}_c} \sum_f h_f \bar{\theta}_k^t \sum_c \gamma_c \, Q_{f,c,k}$$

$$c_{f,c,k} = c0_{f,k} + \bar{\theta}_k^t h_{fk} \sum_{l=1}^{N_{cp}} \Omega_{lc} \dot{c}_{f,l,k} \quad \forall f, c, k \; : \; \lambda^1_{f,c,k}$$

$$c0_{f,k} = c0_{f-1,k} + \bar{\theta}_k^t h_{f-1,k} \sum_{l=1}^{N_{cp}} \Omega_{l,N_c} \dot{c}_{f-1,l,k} \quad \forall f \geq 2, k \; : \; \lambda^2_{f,k}$$

$$\dot{c}_{f,c,k} = \frac{Q_{f,c,k}}{V}(1 - c_{f,c,k}) - k c^3_{f,c,k} \quad \forall f, c, k$$

$$t_{f,c,k} = (f - 1 + \gamma_c)\frac{\bar{\theta}_k^t}{N_{fe}} \quad \forall f, c, k \; : \; \lambda^3_{f,c,k}$$

$$u_{f,c,k} - u_{f,c-1,k} \leq u^c_{cont} \; \forall f, k, c \; \neq 1$$

$$u_{f,c,k} - u_{f,c-1,k} \geq -u^c_{cont} \; \forall f, k, c \; \neq 1$$

$$u_{f,1,k} - u_{f-1,N_{cp},k} \leq u^f_{cont} \; \forall f, k \; \neq 1$$

$$u_{f,1,k} - u_{f-1,N_{cp},k} \geq -u^f_{cont} \; \forall f, k \; \neq 1$$

$$c0_{1,k} = \bar{c^k_{in}} \; : \; \lambda^4_k$$

$$c_{20,3,k} = \bar{c^k_{end}} \; : \; \lambda^5_k$$

$$Q_{1,1,k} = \bar{q^k_{in}} \; : \; \lambda^6_k$$

$$Q_{20,3,k} = \bar{q^k_{end}} \; : \; \lambda^7_k$$

where $\lambda^i_{fck}$ is the lagrange multiplier for contraint $i$ at finite element $f$, collocation point $c$ and slot $k$. Concerning the operating constraints the bounds on the change of the manipulated variable between collocation points is $u^c_{cont} = 200$ and between finite elements $u^f_{cont}$=500. The Lagrangean function for this specific formulation is

$(h_{fk} = 1/N_f)$:

$$
\begin{aligned}
L(z, y, \bar{w}, \bar{\lambda}) = \sum_{i=1}^{Np} \frac{C_i^p W_i}{Tc} - \sum_{i=1}^{Np} \frac{C_i^s (G_i - W_i/T_c)\Theta_i}{2} - \frac{1}{T_c} \sum_{k=1}^{N_s} \sum_{f=1}^{N_f} \frac{\theta_k}{N_f} \sum_{c=1}^{N_c} \bar{Q}_{f,c,k} \gamma_c \\
+ \sum_{k=1}^{N_s} \left( (\bar{c_{in}^k} - c_{in}^k)\bar{\lambda_k^4} + (\bar{c_{end}^k} - c_{end}^k)\bar{\lambda_k^5} + (\bar{q_{in}^k} - q_{in}^k)\bar{\lambda_k^6} + (\bar{q_{end}^k} - q_{end}^k)\bar{\lambda_k^7} \right) \\
+ \sum_{k=1}^{N_s} \left\{ \sum_{f=1}^{N_f} \sum_{c=1}^{N_c} \bar{\lambda_{fck}^1} \left( \bar{c}_{fck} - \bar{c0}_{fk} - \theta_k^t h_{fk} \sum_{l=1}^{N_c} \Omega_{lc} \bar{\dot{c}}_{flk} \right) \right. \\
+ \sum_{f \geq 2}^{N_f} \bar{\lambda_{fk}^2} \left( \bar{c0}_{fck} - \bar{c0}_{(f-1)k} - \theta_k^t h_{fk} \sum_{l=1}^{N_c} \Omega_{l,N_c} \bar{\dot{c}}_{(f-1)lk} \right) \\
\left. + \sum_{f=1}^{N_f} \sum_{c=1}^{N_c} \bar{\lambda_{fck}^3} \left( \bar{t}_{fck} - \frac{\theta_k^t}{N_f} (f - 1 + \gamma_c) \right) \right\}
\end{aligned}
$$

The master problem is:

$$
\begin{aligned}
\max \quad & \eta \\
\text{s.t.} \quad & \eta \leq L(x, y) \\
& \text{Eq. } 3.2 - 3.13, 3.22 - 3.27
\end{aligned}
\tag{A.1}
$$

## A.2 Appendix B: Graph theory concepts

### A.2.1 Centrality measures

A graph $G$ is composed by nodes $(V)$ and edges $(E)$ that capture the interactions among the nodes. A graph can be represented by its adjacency matrix which has elements $A_{ij} = 1$ if an edge exist between node $i$ and $j$ and 0 otherwise. If the nodes represent different types of objects (A ,B) then the graph is bipartite and the set of nodes is $V = A \cup B$ with $A \cap B = \emptyset$. If we project the bipartite graph into one set of nodes then an undirected graph $G_v(A, E_A)$ is obtained where the nodes are the type A nodes of the bipartite graph and the edges capture the interaction between the type A nodes. If the projection is performed on the other set $(B)$ then the $G_c(B, E_B)$ undirected graph is obtained.

In an undirected graph a walk is a sequence of nodes that are connected by an edge.

The length of the walk is the number of edges in the walk and the shortest path is the shortest walk between a given pair of nodes. If the graph is disconnected then the distance between the nodes in different components is infinite. The position of a node in the graph is related with its importance in the graph. The importance of a node is quantified using different metrics; in this work the closeness and betweenness centrality will be used. Closeness centrality is a measure of the mean distance from a node to other nodes. If $d_{ij}$ is the shortest path from node i to node j, then the mean distance ($l_i$) from node i to all other nodes is:

$$l_i = \frac{1}{n-1} \sum_j d_{ij} \tag{A.2}$$

and the closeness centrality ($C_i$) of node i is the inverse of the mean distance:

$$C_i = \frac{n-1}{\sum_j d_{ij}} \tag{A.3}$$

This definition of closeness centrality is used since we have only one component in the graph. Betweenness centrality captures the extent to which a node lies in the shortest paths between other nodes. The number of shortest paths from node $s$ to node $t$ that pass through node $i$ is $\sigma(s,t|i)$ and the total number of shortest paths from $s$ to $t$ is $\sigma(s,t)$. The betweenness centrality is equal to:

$$C_B(i) = \sum_{s,t \in V} \frac{\sigma(s,t|i)}{\sigma(s,t)} \tag{A.4}$$

These two values will be used to evaluate the importance of a node in the graph.

### A.2.2 Community detection

The edges of a graph capture the interactions between the nodes. Based on these interactions a graph can be partitioned into communities, i.e sets of nodes which are more densely connected compared to the rest of the graph. The community structure of a graph is quantified using a metric called modularity and computed by the following equation:

$$Q = \frac{1}{2m} \sum_{ij} \big(A_{ij} - P_{ij}\big) b_i b_j \tag{A.5}$$

where $A_{ij}$ is the $(i, j)$ entry of the adjacent matrix of the graph and $b_i, b_j$ are integers declaring the group membership of nodes $i$ and $j$ respectively. Modularity compares the number of connections between the nodes in a community with the number of connections if the edges were chosen at random with probability $P_{ij}$. Usually this probability is computed using the configuration model for generating random graphs. In this model the degree $k_i$ of node $i$ follows a specific degree distribution given by the entries of the adjacent matrix. Therefore, given the degree distribution of each node the goal is to find the group membership of the nodes such that the modularity is maximized, solving the following optimization problem:

$$\max_b \frac{1}{2m} \sum_{ij} \left( A_{ij} - \frac{k_i k_j}{2m} \right) b_i b_j$$

This is an integer programming problem and is proven to be NP-hard. For its solution, a two-step greedy optimization algorithm is used [42]. Initially, each node is assigned to its own community. In the first step, the nodes are assigned to communities of their neighbors such that the modularity is increased. When a local optimum of the modularity has been reached then the second step of the algorithm is performed, where a new graph is created where the nodes represent the nonempty communities of the graph obtained in the end of the first step. Then the first step of the algorithm is applied again to the new graph. These two steps are applied iteratively until the modularity cannot increase further. A detailed analysis of the algorithm can be found in [42].

## A.3 Appendix C: Notation

Variables and parameters in the integrated scheduling and dynamic optimization problem:

Parameters

$C_i^p$ : Price of product $i$ (\$/kg)

$C_i^s$ : Inventory cost of product $i$ (\$/kg)

$C^m$ : Cost of manipulated variable $m$

$D_i$ : Demand of product $i$

$\theta^{max}$ : Maximum time that product $i$ can be manufactured

$\Omega$ : Collocation matrix

$\gamma$ : Radau roots

$x_{ni}^{ss}$: Steady state value of state $n$ for product $i$

$u_{mi}^{ss}$: Steady state value of flowrate $m$ for product $i$

$V$ : Reactor volume

$k$ : Reaction constant

$h_{fk}$: Length of finite element $f$ in slot $k$

Variables

$W_i$ : Amount of product $i$ manufactured

$T_c$: Total cyclic time

$G_i$: Production rate of product $i$

$\Theta_i$: Production time of product $i$

$\theta_k^t$: Transition time in slot $k$

$u_{fck}$: Manipulated variable in finite element $f$ in collocation point $c$ and slot $k$

$y_{ik}$: Binary variable, 1 if product $i$ is produced in slot $k$ 0 otherwise

$\theta_{ik}$: Production time of product $i$ in slot $k$

$p_k$: Processing time in slot $k$

$t_k^s$: Starting time of slot $k$

$t_k^e$: Ending time of slot $k$

$x_{nfck}$: State $n$ in finite element $f$ in collocation point $c$ and slot $k$

$u_{mfck}$: State $n$ in finite element $f$ in collocation point $c$ and slot $k$

$x0_{nfk}$: State $n$ in finite element $f$ and slot $k$

$t_{fck}$: Time in finite element $f$ in collocation point $c$ and slot $k$

$x_{nk}^{in}$: Value of the state variable $n$ at the beginning of slot $k$

$x_{nk}^{end}$: Value of the state variable $n$ at the end of slot $k$

# Appendix B
# Supporting information for: Efficient Solution of Enterprise-wide Optimization Problems Using Nested Stochastic Blockmodeling

## B.1 Formulation of the master and subproblem for the application of GBD based on the structure of the level 1 multigraph

Based on the core-periphery structure of the first level multigraph, the variables are decomposed into three sets. The first set contains variables $s_1 = \{Y_{ikl}, z_{ijkl}, t^{prod}_{ikl}, H_l\}$, the second variables $s_2 = \{x_{fckl}, u_{fckl}, t^d_{fckl}, h^{fe}_{kl}\}$, and the last set contains the complicating variables $s_3 = \{x^{in}_{kl}, x^{end}_{kl}, u^{in}_{kl}, u^{end}_{kl}, \theta^t_{kl}\}$. We create a copy of these variables, $\hat{x}^{in}_{kl}, \hat{x}^{end}_{kl}, \hat{u}^{in}_{kl}, \hat{u}^{end}_{kl}, \hat{\theta}^t_{kl}$, and constraints 41, 37 (in the paper) are written as

$$\begin{aligned} x^n_{11kl} &= \hat{x}^{in}_{kl} \ \forall k,l \\ x^n_{N_{fe}N_{cp}kl} &= \hat{x}^{end}_{kl} \ \forall n,k,l \\ u^m_{11kl} &= \hat{u}^{in}_{kl} \ \forall n,k,l \\ u^m_{N_{fe}N_{cp}kl} &= \hat{u}^{end}_{kl} \ \forall n,k,l \\ h^{fe}_{kl} &= \frac{\hat{\theta}^t_{kl}}{N_{fe}} \forall k,l. \end{aligned} \tag{S.1}$$

We reformulate the integrated problem as follows:

$$\text{minimize} \quad \sum_{ikl} c^p_{il} \frac{W_{ikl}}{H_l} + \sum_{ijkl} c^{trans}_{ij} \theta^t_{kl} \frac{Z_{ijkl}}{H_l} + \sum_{ikl} c^{stor}_{il} W_{ikl} + a_u \sum_{klfc} \frac{t^d_{fckl}}{N_{fe}} \Lambda_{c,N_{cp}} (u_{fckl} - \hat{u}^{end}_{kl})^2$$

$$\begin{aligned} \text{subject to} \quad & \text{Equations 29,30,31,32,36,38,40,S.1} \\ & \hat{x}^{in}_{kl} = x^{in}_{kl} \ \ \forall k,l \\ & \hat{x}^{end}_{kl} = x^{end}_{kl} \ \ \forall k,l \\ & \hat{u}^{in}_{kl} = u^{in}_{kl} \ \ \forall k,l \\ & \hat{u}^{end}_{kl} = u^{end}_{kl} \ \ \forall k,l \\ & \hat{\theta}^t_{kl} = \theta^t_{kl} \ \ \forall k,l. \end{aligned}$$

If the scheduling variables are fixed, then the optimization problem is:

$$\begin{aligned}
\text{minimize} \quad & a_u \sum_{klfc} \frac{t^d_{fckl}}{N_{fe}} \Lambda_{c,N_{cp}} (u_{fckl} - \hat{u}^{end}_{kl})^2 \\
\text{subject to} \quad & g_{dyn} \leq 0 \,(\text{Eq. } 36{,}38{,}\text{S.1}) \\
& \hat{x}^{in}_{kl} = \bar{x}^{in}_{kl} \;\; \forall k \in \mathcal{I}_s, l \in \mathcal{I}_l \quad : \quad \lambda^1_{kl} \\
& \hat{x}^{end}_{kl} = \bar{x}^{end}_{kl} \;\; \forall k \in \mathcal{I}_s, l \in \mathcal{I}_l \quad : \quad \lambda^2_{kl} \\
& \hat{u}^{in}_{kl} = \bar{u}^{in}_{kl} \;\; \forall k \in \mathcal{I}_s, l \in \mathcal{I}_l \quad : \quad \lambda^3_{kl} \\
& \hat{u}^{end}_{kl} = \bar{u}^{end}_{kl} \;\; \forall k \in \mathcal{I}_s, l \in \mathcal{I}_l \quad : \quad \lambda^4_{kl} \\
& \hat{\theta}^t_{kl} = \bar{\theta}^t_{kl} \;\; \forall k \in \mathcal{I}_s, l \in \mathcal{I}_l \quad : \quad \lambda^5_{kl}
\end{aligned} \tag{S.2}$$

If we define $\eta$ as the value function of the dynamic optimization subproblem the original problem can be written as:

$$\begin{aligned}
\text{minimize} \quad & \sum_{ikl} c^p_{il} \frac{W_{ikl}}{H_l} + \sum_{ijkl} c^{trans}_{ij} \theta^t_{kl} \frac{Z_{ijkl}}{H_l} + \sum_{ikl} c^{stor}_{il} W_{ikl} \\
& + \eta(\{x^{in}_{kl}\}, \{x^{end}_{kl}\}, \{u^{in}_{kl}\}, \{u^{end}_{kl}\}, \{\theta^t_{kl}\}) \\
\text{subject to} \quad & \text{Equations } 29{,}30{,}31{,}32{,}40
\end{aligned}$$

This problem can not be solved directly since the value function is not known. It can be approximated as follows:

$$\begin{aligned}
\eta = \max_{\sigma,\lambda} \Bigg\{ \min \Bigg( & a_u \sum_{klfc} \frac{t^d_{fckl}}{N_{fe}} \Omega_{c,N_{cp}} (u_{fckl} - \hat{u}^{end}_{kl})^2 - \sigma^\top g_{dyn} \\
& - \sum_{k,l} \Big( \lambda^1_{kl} (x^{in}_{kl} - \hat{x}^{in}_{kl}) + \lambda^2_{kl} (x^{end}_{kl} - \hat{x}^{end}_{kl}) \\
& + \lambda^3_{kl} (u^{in}_{kl} - \hat{u}^{in}_{kl}) - \lambda^4_{kl} (u^{end}_{kl} - \hat{u}^{end}_{kl}) \\
& + \lambda^{kl}_5 (\theta^t_{kl} - \hat{\theta}^t_{kl}) \Big) \Bigg) \Bigg\},
\end{aligned}$$

where $\sigma, \lambda$ are the Lagrangean multipliers. This expression can be relaxed as follows:

$$\eta \geq \min \Bigg( a_u \sum_{klfc} \frac{t^d_{fckl}}{N_{fe}} \Omega_{c,N_{cp}} (u_{fckl} - \hat{u}^{end}_{kl})^2 - \sigma^\top g_{dyn} \\ - \sum_{k,l} \Bigg( \lambda^1_{kl} (x^{in}_{kl} - \hat{x}^{in}_{kl}) + \lambda^2_{kl} (x^{end}_{kl} - \hat{x}^{end}_{kl}) \\ + \lambda^3_{kl} (u^{in}_{kl} - \hat{u}^{in}_{kl}) - \lambda^4_{kl} (u^{end}_{kl} - \hat{u}^{end}_{kl}) \\ + \lambda^{kl}_5 (\theta^t_{kl} - \hat{\theta}^t_{kl}) \Bigg) \Bigg).$$

If we solve the dynamic optimization problem for given values of the comlicating variables, and substitute the dual variables ($\bar{\sigma}, \bar{\lambda}, \bar{\sigma}^\top g_{dyn} = 0$) and variables that belong in the subproblem ($\bar{x}_{fckl}, \bar{u}_{fckl}, \bar{h}^{fe}_{kl}, \hat{\bar{x}}^{in}_{kl}, \hat{\bar{x}}^{end}_{kl}, \hat{\bar{u}}^{in}_{kl}, \hat{\bar{u}}^{end}_{kl}, \hat{\bar{\theta}}^t_{kl}$) with their optimal values we obtain the Benders cut:

$$\eta \geq a_u \sum_{klfc} \frac{\bar{t}^{d,p}_{fckl}}{N_{fe}} \Lambda_{c,N_{cp}} (\bar{u}^p_{fckl} - \bar{\hat{u}}^{end,p}_{kl})^2 \\ - \sum_{k,l} \Bigg( \lambda^{1,p}_{kl} (x^{in}_{kl} - \bar{\hat{x}}^{in,p}_{kl}) + \lambda^{2,p}_{kl} (x^{end}_{kl} - \bar{\hat{x}}^{end,p}_{kl}) + \lambda^{3,p}_{kl} (u^{in}_{kl} - \bar{\hat{u}}^{in,p}_{kl}) \\ + \lambda^{4,p}_{kl} (u^{end}_{kl} - \bar{\hat{u}}^{end,p}_{kl}) + \lambda^{kl}_{5,p} (\theta^t_{kl} - \bar{\hat{\theta}}^{t,p}_{kl}) \Bigg) \ \forall p \in P.$$

## B.2 Formulation of the nested GBD based on the structure of the variable graph of the integrated scheduling and dynamic optimization problem for parallel lines

Based on the multi-core community structure of the variable graph of the problem, the full space model, is first decomposed into a master and subproblem as above. The master problem is decomposed further into problems MM and MS. Variables $Y_{ikl}$,$z_{ijkl}$,$x^{end}_{kl}, x^{in}_{kl}, u^{end}_{kl}, u^{in}_{kl}$ are assigned in problem MM (complicating variables) and the other variables are assigned in problem MS. We create a copy of the complicating

variables , $\tilde{Y}_{ikl}$,$\tilde{z}_{ijkl}$, $\tilde{x}^{end}_{kl}$,$\tilde{x}^{in}_{kl}$,$\tilde{u}^{end}_{kl}$,$\tilde{u}^{in}_{kl}$, and the master problem can be written as:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{ikl} c^{p}_{il}\frac{W_{ikl}}{H_l} + \sum_{ijkl} c^{trans}_{ij}\theta^{t}_{kl}\frac{\tilde{z}_{ijkl}}{H_l} + \sum_{ikl} c^{stor}_{il} W_{ikl} + \eta \\
\text{subject to} \quad & \text{Equations 29,30,40,45,47} \\
& t^{prod}_{ikl} \leq t_{max}\tilde{Y}_{ikl} \ \forall\ i \in \mathcal{I}_p, k \in \mathcal{I}_s, l \in \mathcal{I}_l \\
& H_l = \sum_{i=1}^{N_p}\sum_{k=1}^{N_s} t^{prod}_{ikl} \ \forall l \in \mathcal{I}_l \\
& W_{ikl} = r_{il}(t^{prod}_{ikl} - \theta^{t}_{kl}\tilde{Y}_{ikl}) \ \forall\ i \in \mathcal{I}_p, k \in \mathcal{I}_s, l \in \mathcal{I}_l \qquad \text{(S.3)} \\
& \tilde{Y}_{ikl} = Y_{ikl} \ \forall\ i \in \mathcal{I}_p, k \in \mathcal{I}_s, l \in \mathcal{I}_l \\
& \tilde{z}_{ijkl} = z_{ijkl} \ \ \forall\ i \in \mathcal{I}_p, j \in \mathcal{I}_p, k \in \mathcal{I}_s, l \in \mathcal{I}_l \\
& \tilde{x}^{in}_{kl} = x^{in}_{kl} \ \forall\ k \in \mathcal{I}_s, l \in \mathcal{I}_l \\
& \tilde{x}^{end}_{kl} = x^{end}_{kl} \ \forall\ k \in \mathcal{I}_s, l \in \mathcal{I}_l \\
& \tilde{u}^{in}_{kl} = u^{in}_{kl} \ \forall\ k \in \mathcal{I}_s, l \in \mathcal{I}_l \\
& \tilde{u}^{end}_{kl} = u^{end}_{kl} \ \forall\ k \in \mathcal{I}_s, l \in \mathcal{I}_l
\end{aligned}
$$

If we fix the complicating variables $Y_{ikl}$, $z_{ijkl}$, $x^{end}_{kl}$,$x^{in}_{kl}$,$u^{end}_{kl}$,$u^{in}_{kl}$, we obtain problem MS:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{ikl} c^{p}_{il}\frac{W_{ikl}}{H_l} + \sum_{ijkl} c^{trans}_{ij}\theta^{t}_{kl}\frac{\tilde{z}_{ijkl}}{H_l} + \sum_{ikl} c^{stor}_{il} W_{ikl} + \eta \\
\text{subject to} \quad & \text{Equation 45,47} \\
& t^{prod}_{ikl} \leq t_{max}\tilde{Y}_{ikl} \ \forall\ i \in \mathcal{I}_p, k \in \mathcal{I}_s, l \in \mathcal{I}_l \\
& H_l = \sum_{i=1}^{N_p}\sum_{k=1}^{N_s} t^{prod}_{ikl} \ \forall l \in \mathcal{I}_l \\
& W_{ikl} = r_{il}(t^{prod}_{ikl} - \theta^{t}_{kl}\tilde{Y}_{ikl}) \ \forall\ i \in \mathcal{I}_p, k \in \mathcal{I}_s, l \in \mathcal{I}_l \qquad \text{(S.4)} \\
& \tilde{Y}_{ikl} = \bar{Y}_{ikl} \ \forall\ i \in \mathcal{I}_p, k \in \mathcal{I}_s, l \in \mathcal{I}_l \quad : \ \mu^{1}_{ikl} \\
& \tilde{z}_{ijkl} = \bar{z}_{ijkl} \ \forall\ i \in \mathcal{I}_p, j \in \mathcal{I}_p, k \in \mathcal{I}_s, l \in \mathcal{I}_l \quad : \ \mu^{2}_{ijkl} \\
& \tilde{x}^{in}_{kl} = \bar{x}^{in}_{kl} \ \forall\ k \in \mathcal{I}_s, l \in \mathcal{I}_l \quad : \ \mu^{3}_{kl} \\
& \tilde{x}^{end}_{kl} = \bar{x}^{end}_{kl} \ \forall\ k \in \mathcal{I}_s, l \in \mathcal{I}_l \quad : \ \mu^{4}_{kl} \\
& \tilde{u}^{in}_{kl} = \bar{u}^{in}_{kl} \ \forall\ k \in \mathcal{I}_s, l \in \mathcal{I}_l \quad : \ \mu^{5}_{kl} \\
& \tilde{u}^{end}_{kl} = \bar{u}^{end}_{kl} \ \forall\ k \in \mathcal{I}_s, l \in \mathcal{I}_l \quad : \ \mu^{6}_{kl}
\end{aligned}
$$

If we denote the scheduling constraints (except the last 6) as $g_{sched}$ and the objective function as $f_{sched}$, then the value of this optimization problem can be approximated as follows

$$\begin{aligned}\eta_2 = \max_{\delta,\mu} \Bigg\{ \min \quad \Bigg( & f_{sched} - \delta^\top g_{sched} - \sum_{i,k,l} \mu^1_{kl}(Y_{ikl} - \bar{Y}_{ikl}) \\ & - \sum_{ijkl} \mu^2_{ijkl}(z_{ijkl} - \bar{z}_{ijkl}) - \sum_{k,l} \mu^3_{kl}(x^{in}_{kl} - \bar{x}^{in}_{kl}) \\ & - \sum_{kl} \lambda^4_{kl}(x^{end}_{kl} - \bar{x}^{end}_{kl}) - \sum_{kl} \lambda^5_{kl}(u^{in}_{kl} - \bar{u}^{in}_{kl}) \\ & - \sum_{kl} \mu^{kl}_5(u^{end}_{kl} - \bar{u}^{end}_{kl}) \Bigg) \Bigg\}, \end{aligned} \tag{B.1}$$

where $\delta, \mu$ are the Lagrangean multipliers. If we fix the shared variables and solve the dynamic optimziation problem, we can substitute the multipliers and the variables that belong in problem MS with their optimal values, and we obtain the Benders cut (Eq. 51).

## B.3 Formulation of the GBD based on the first level core-periphery structure of the integrated planning, scheduling and dynamic optimization problem

In this case the variables are decomposed into three sets; variables that affect only the planning/scheduling problem, variables that affect the dynamic optimization problem and complicating variables $x^{in}_{kp}, x^{end}_{kp}, u^{in}_{kp}, u^{end}_{kp}, \theta^t_{kl}$. We create a copy of these variables and follow the same steps as in the first Section.

# Appendix C

# Supplementary material - A multicut Generalized Benders Decomposition approach for the integration of process operations and dynamic optimization for continuous systems

## C.1 Computational results for random values of the parameters of the problem for the MMA polymerization reactor

The parameters of the problems that can change are: demand, price and operating cost. For each parameter we assume that it is a random variable with mean ($\mu$) and standard deviation ($\sigma$) presented in the Table C.1.

Table C.1: Mean and standard deviation of the parameters of the integrated problem for the second case study.

| Period 1 | | | |
|---|---|---|---|
| Product | Demand ($\mu/\sigma$) | Price ($\mu/\sigma$) | Operating cost ($\mu/\sigma$) |
| 1 | 2500/200 | 300/50 | 23/5 |
| 2 | 2200/300 | 180/15 | 30/5 |
| 3 | 3000/250 | 160/30 | 45/5 |
| 4 | 1000/50 | 120/25 | 50/5 |
| Period 2 | | | |
| Product | Demand ($\mu/\sigma$) | Price ($\mu/\sigma$) | Operating cost ($\mu/\sigma$) |
| 1 | 2200/250 | 280/50 | 10/5 |
| 2 | 1400/200 | 150/15 | 25/5 |
| 3 | 3500/300 | 190/30 | 55/5 |
| 4 | 2000/100 | 120/25 | 29/5 |
| Period 3 | | | |
| Product | Demand ($\mu/\sigma$) | Price ($\mu/\sigma$) | Operating cost ($\mu/\sigma$) |
| 1 | 2150/200 | 250/45 | 15/5 |
| 2 | 1800/150 | 160/35 | 20/5 |
| 3 | 2500/200 | 180/15 | 40/5 |
| 4 | 1500/20 | 130/5 | 20/5 |

We generated 50 instances and the solution time statistics are presented in Table C.2.

Table C.2: Solution time statistics for random instances of the second case study

| Algorithm | Average sol. time (sec) | Standard deviation |
|---|---|---|
| Multicut | 96 | 18.3 |
| Hybrid Multicut | 42 | 5.6 |
| GBD | 443 | 84.5 |

From these results we observe that on average, the multicut algorithm reduced the CPU time by 78%, the hybrid multicut algorithm by 90 % compared to GBD and 56% compared to the multicut algorithm.

## C.2 Computation of minimum transition time

The minimum transition time between product $i$ and $j$ is computed by solving the following problem:

$$
\begin{aligned}
\theta_{ij}^{min} = \text{minimize} \quad & \theta \\
\text{subject to} \quad & x_{fc}^{n} = x0_{f}^{n} + h^{fe} \sum_{m=1}^{N_{cp}} \Omega_{mc} \dot{x}_{ijfmkp}^{n} \quad \forall n, f, c \\
& h^{fe} = \frac{\theta}{N_{fe}} \\
& x0_{f}^{n} = x0_{f-1}^{n} + h^{fe} \sum_{m=1}^{N_{cp}} \Omega_{mc} \dot{x}_{f-1,m}^{n} \quad \forall n, f \geq 2, c \\
& \dot{x}_{fc}^{n} = f(x_{fc}^{n}, u_{fc}^{m}) \quad \forall n, f, c \\
& t_{fc}^{d} = h^{fe}(f - 1 + \gamma_{c}) \quad \forall f, c \\
& x0_{1} = x_{i}^{ss} \\
& x_{N_{fe}N_{cp}} = x_{j}^{ss} \\
& u_{11} = u_{i}^{ss} \\
& u_{N_{fe}N_{cp}} = u_{j}^{ss}
\end{aligned}
\tag{C.1}
$$

## C.3 Computation of bounds for the linearization of the bilinear terms in the second case study

The proposed formulation has two sets of bilinear terms. The first $Z_{ijkp}\eta_{ijkp}$ appears in the objective function and the second $Z_{ijkp}\theta_{ijkp}$ appears in constraint 9 (in the paper). Using the linearization presented in the paper, for the first set of bilinear terms we get

$$\begin{aligned} 0 \leq \delta_{ijkp} &\leq \bar{\eta}_{ij} \\ \underline{\eta}_{ij} Z_{ijkp} \leq \delta_{ijkp} &\leq \bar{\eta}_{ij} Z_{ijkp} \\ \eta_{ijkp} - (1 - Z_{ijkp})\bar{\eta}_{ij} \leq \delta_{ijkp} &\leq \eta_{ijkp} - (1 - Z_{ijkp})\underline{\eta}_{ij} \\ \delta_{ijkp} &\leq \eta_{ijkp} + (1 - Z_{ijkp})\underline{\eta}_{ij}. \end{aligned} \tag{C.2}$$

In these equations $\bar{\eta}_{ij}$ is the upper bound on the transition cost from product $i$ to product $j$. For the MMA polymerization reactor, from Fig. 7 (in the manuscript) we observe that the value functions are convex, and the maximum value is obtained when the transition time is equal to the minimum transition time. Therefore, the upper bound

is obtained by solving the following problem:

$$
\begin{aligned}
\bar{\eta}_{ij} = \text{minimize} \quad & \sum_{f=1}^{N_{fe}} \sum_{c=1}^{N_{cp}} N_{fe}^{-1} t_{fc}^{d} \Omega_{c,N_{cp}} (u_{fc} - u_{j}^{ss})^2 \\
\text{subject to} \quad & x_{fc}^{n} = x0_{f}^{n} + h^{fe} \sum_{m=1}^{N_{cp}} \Omega_{mc} \dot{x}_{ijfmkp}^{n} \quad \forall n, f, c \\
& h^{fe} = \frac{\theta}{N_{fe}} \\
& x0_{f}^{n} = x0_{f-1}^{n} + h^{fe} \sum_{m=1}^{N_{cp}} \Omega_{mc} \dot{x}_{f-1,m}^{n} \quad \forall n, f \geq 2, c \\
& \dot{x}_{fc}^{n} = f(x_{fc}^{n}, u_{fc}^{m}) \quad \forall n, f, c \\
& t_{fc}^{d} = h^{fe}(f - 1 + \gamma_{c}) \quad \forall f, c \\
& x0_{1} = x_{i}^{ss} \\
& x_{N_{fe}N_{cp}} = x_{j}^{ss} \\
& u_{11} = u_{i}^{ss} \\
& u_{N_{fe}N_{cp}} = u_{j}^{ss} \\
& \theta = \theta_{ij}^{min}
\end{aligned}
\tag{C.3}
$$

The lower bound can be obtained by relaxing the last constraint in the above problem.

$$
\begin{aligned}
\bar{\eta}_{ij} = \text{minimize} \quad & \sum_{f=1}^{N_{fe}} \sum_{c=1}^{N_{cp}} N_{fe}^{-1} t_{fc}^{d} \Omega_{c,N_{cp}} (u_{fc} - u_j^{ss})^2 \\
\text{subject to} \quad & x_{fc}^{n} = x0_f^n + h^{fe} \sum_{m=1}^{N_{cp}} \Omega_{mc} \dot{x}_{ijfmkp}^{n} \quad \forall n, f, c \\
& h^{fe} = \frac{\theta}{N_{fe}} \\
& x0_f^n = x0_{f-1}^n + h^{fe} \sum_{m=1}^{N_{cp}} \Omega_{mc} \dot{x}_{f-1,m}^{n} \quad \forall n, f \geq 2, c \\
& \dot{x}_{fc}^{n} = f(x_{fc}^{n}, u_{fc}^{m}) \quad \forall n, f, c \\
& t_{fc}^{d} = h^{fe}(f - 1 + \gamma_c) \quad \forall f, c \\
& x0_1 = x_i^{ss} \\
& x_{N_{fe}N_{cp}} = x_j^{ss} \\
& u_{11} = u_i^{ss} \\
& u_{N_{fe}N_{cp}} = u_j^{ss}
\end{aligned}
\tag{C.4}
$$

## C.4 Stochastic Blockmodeling and statistical inference

In this section we present the stochastic model and inference approach for the simple Stochastic Blockmodel for brevity. We refer the reader to [240, 238] for a detailed explanation of SBM and to [201, 199] for the application of this approach to optimization problems.

We consider an undirected graph $G(V, E)$ with $N$ nodes ($N = |V|$), $M$ edges ($M = |E|$) and adjacency matrix $A \in \mathbb{R}^{N \times N}$, where $A_{ij} = A_{ji}$ is equal to the number of edges between node $i$ and $j$. We assume that the nodes are divided into $B$ blocks, and define a partition vector $b \in \mathbb{R}^N$, where $b_i \in \{1, .., B\}$ denotes the group membership of node $i$. We also define the matrix $\omega \in \mathbb{R}^{B \times B}$, where $\omega_{rs}$ is equal to the number of edges between the nodes in block $r$ and the nodes in block $s$, and $\omega_{rr}$ is equal to twice the number of edges in block $r$. If we assume that the number of edges between a node in block $r$ and a node in block $s$ is a Poisson random variable with expected value $\omega_{rs}$,

then the probability to observe a graph with adjacency matrix $A$ is

$$P(A|b,\omega) = \prod_{1\leq i<j\leq N} \frac{\omega_{b_i b_j}^{A_{ij}}}{A_{ij}!} e^{-\omega_{b_i b_j}} \prod_{i=1}^{N} \frac{(\frac{1}{2}\omega_{b_i b_i})^{A_{ii}}}{(A_{ii}/2)!} e^{-\frac{1}{2}\omega_{b_i b_i}}, \tag{S.1}$$

where the first term considers the probability of an edge between two nodes and the second term is the probability of a self edge. From this equation it is evident that the structure of the graph depends on the group membership of the nodes $b$ since the probability of an edge between the nodes or a self-edge depends solely on the group membership of the nodes.

Given the above equation the inverse problem must be solved: given a graph with adjacency matrix $A$ find the partition of the nodes and the matrix $\omega$ that lead to the generation of this graph. In this work we will focus on the second method. Direct maximization of $P(A|\omega, b)$ leads to overfitting, since an increase in the number of the blocks leads to an increase in the probability. This can be avoided using a Bayesian inference approach. Specifically, from Bayes' rule we obtain

$$P(b|A) = \frac{P(A|b)P(b)}{P(A)}, \tag{S.2}$$

where the numerator depends on the model parameters since

$$P(A|b)P(b) = P(A|\omega, b)P(\omega|b)P(b) = P(A|\omega, b)P(\omega, b). \tag{S.3}$$

Given these equations, the original task of maximizing $P(A|b)$ is equivalent to minimizing $-P(b|A)$ which is equivalent to

$$\underset{b,\omega}{\text{minimize}} - \log_2 P(A|\omega, b) - \log_2 P(\omega, b). \tag{S.4}$$

This objective has an information theoretical interpretation. The first term is the amount of bits that is necessary to encode the observed data, i.e. graph, and the second term is the amount of bits necessary to encode the model itself. Therefore, an increase in the number of the blocks leads to a reduction in the first term but the second term increases since the model becomes more complex. Hence, this approach avoids overfitting. Finally, before solving the problem, we must define the prior distributions $P(\omega|b), P(b)$. We refer the reader to [240] for a detailed derivation of these

priors. Given the above information, the problem is solved using a Markov Chain Monte Carlo (MCMC) approach where the estimation of the posterior is achieved by random node move proposals. This approach can infer both the partition of the nodes and the number of the blocks. We refer the reader to [240, 237] for a detailed explanation of the algorithm. Finally, we must note that the inference results might differ between runs, since the algorithm is stochastic.

Two main extensions of this model that we will use are the degree corrected [240, 237] and weighted SBM [238]. In the basic SBM model, nodes with high degree tend to be assigned in the same block. This can be avoided by considering the expected degrees of the nodes as additional parameters of the model which are sampled from their own distribution. The second variant considers the case where a weight is associated with the edges. We can assume that the observed weights depend on some parameters which are sampled from their own distributions. Different models can be used for the priors for the weights; we refer the reader to [238] for detailed explanations. Given the prior distribution of the expected degrees and weights, the same MCMC approach can be followed to solve the inference problem.

## C.5 Formulation of the integrated problem based on [125] and GBD formulation based on [199]

In this section we will present the integrated problem proposed in [125] and GBD algorithm based on [199]. The planning/scheduling problem is the same as in Section 2.1 (in the paper). Similarly as in Section 2.2 (in the paper) the dynamic behavior of the system is described by a system of ordinary differential equations Eq. 6 (in the paper) which are discretized using the method of orthogonal collocation on finite elements and

are equal to:

$$
\begin{aligned}
x^n_{fckp} &= x0^n_{fkp} + h^{fe}_{kp} \sum_{m=1}^{N_{cp}} \Omega_{mc} \dot{x}^n_{fmkp} \quad \forall n, f, c, k, p \\
x0^n_{fkp} &= x0^n_{f-1kp} + h^{fe}_{kp} \sum_{m=1}^{N_{cp}} \Omega_{mc} \dot{x}^n_{f-1,mkp} \quad \forall n, f \geq 2, c, k, p \\
\dot{x}^n_{fckp} &= f^n(x^n_{fckp}, u^m_{fckp}) \quad \forall n, f, c, k, p \\
t^d_{fckp} &= h^{fe}_{kp}(f - 1 + \gamma_c) \quad \forall f, c, k, p \\
h^{fe}_{kp} &= \frac{\theta^t_{kp}}{N_{fe}} \quad \forall k, p.
\end{aligned}
\tag{S.5}
$$

The integration of the planning/scheduling problem with the dynamic behavior of the system is done through the following constraints:

$$
\begin{aligned}
x^{in}_{n,k,p} &= \sum_i x^{ss}_i W_{i,k,p} \quad \forall n, k, p \\
x^{end}_{n,k,p} &= \sum_i x^{ss}_i W_{i,k+1,p} \quad \forall n, k, p \\
u^{in}_{n,k,p} &= \sum_i u^{ss}_i W_{i,k,p} \quad \forall n, k, p \\
u^{end}_{n,k,p} &= \sum_i u^{ss}_i W_{i,k+1,p} \quad \forall n, k, p
\end{aligned}
\tag{S.6}
$$

$$
\begin{aligned}
x0^n_{1kp} &= x^{in}_{n,k,p} \quad \forall n, k, p \\
x^n_{N_{fe}N_{cp}kp} &= x^{end}_{n,k,p} \quad \forall n, k, p \\
u^m_{11kp} &= u^{in}_{m,k,p} \quad \forall m, k, p \\
u^m_{N_{fe}N_{cp}kp} &= u^{end}_{m,k,p} \quad \forall m, k, p
\end{aligned}
\tag{S.7}
$$

The objective of the integrated problem is to maximize the profit which is equal to

$$
\begin{aligned}
f = & \sum_{i=1}^{N_p} \sum_{p=1}^{N_{per}} \left( P_{ip} S_{ip} - C_{ip}^{oper} q_{ip} - C^{inv} A_{ip} \right) \\
& - \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} \sum_{k=1}^{N_s} \sum_{p=1}^{N_{per}} C_{ij}^{trans} Z_{ijkp} - \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} \sum_{p=1}^{N_{per}} C_{ij}^{trans} Zp_{ijp} \\
& - \alpha_u \sum_{p=1}^{N_{per}} \sum_{k=1}^{N_p} \sum_{f=1}^{N_{fe}} \sum_{c=1}^{N_{cp}} N_{fe}^{-1} t_{fckp}^{d} \Lambda_{c,N_{cp}} (u_{fckp} - u_{kp}^{end})^2,
\end{aligned}
\tag{S.8}
$$

Application of Nested SBM to the variable graph of the optimization problem reveals a hybrid core-community structure [199]. The variables associated with the dynamic behavior of the problem for each slot and period ($x_{nfckp}$, $u_{mfckp}$, $h_{kp}^{fe}$, $t_{fckp}^{d}$) are assigned in different blocks and the planning/scheduling variables are assigned in the same block. In this case the complicating variables are $x_{kp}^{in}$, $x_{kp}^{end}$, $u_{kp}^{in}$, $u_{kp}^{end}$, $\theta_{kp}^{t}$. Hence, the subproblems are the dynamic optimization problems for each slot and period, solved for fixed values of the complicating variables and the master problem is the planning/scheduling problem with the Benders cuts, which are given by the following equation:

$$
\begin{aligned}
\eta \geq & \alpha_u \sum_{p=1}^{N_{per}} \sum_{k=1}^{N_p} \sum_{f=1}^{N_{fe}} \sum_{c=1}^{N_{cp}} N_{fe}^{-1} \bar{t}_{fckp}^{d,v} \Omega_{c,N_{cp}} (\bar{u}_{fckp}^{v} - \bar{u}_{kp}^{end,,v})^2 \\
& - \sum_{p=1}^{N_{per}} \sum_{k=1}^{N_s} \left( \gamma_{k,p}^{1,v} (x_{kp}^{in} - \bar{x}_{kp}^{in,v}) + \gamma_{k,p}^{2,v} (x_{kp}^{end} - \bar{x}_{kp}^{end,v}) \right. \\
& + \gamma_{k,p}^{3,v} (u_{kp}^{in} - \bar{u}_{kp}^{in,v}) + \gamma_{k,p}^{4,v} (u_{kp}^{end} - \bar{u}_{kp}^{end,v}) \\
& \left. + \gamma_{k,p}^{5,v} (\theta_{kp}^{t} - \bar{\theta}_{kp}^{t,v}) \right) \ \forall v \in \mathcal{V},
\end{aligned}
\tag{S.9}
$$

where the superscript $v$ is the iteration number and $\gamma$ are the Lagrnagean multipliers. We refer the reader to [199] for a detailed explanation of the structure of the problem and derivation of Generalized Benders Decomposition.

## C.6 Data for 4 planning periods

The economic data of the problem are given in Tables C.3,C.4,C.5.

Table C.3: Operating and transition cost for the second case study for 4 planning periods, $C^{inv} = 0.026, a_u = 10^6$.

| | $C^{oper}$ | | | |
|---|---|---|---|---|
| Product | $p = 1$ | $p = 2$ | $p = 3$ | $p = 4$ |
| 1 | 23 | 10 | 15 | 25 |
| 2 | 30 | 25 | 20 | 18 |
| 3 | 45 | 55 | 40 | 45 |
| 4 | 50 | 29 | 20 | 25 |

Table C.4: Product demand for the second case study for 4 planning periods.

| | Demand (kg/week) | | | |
|---|---|---|---|---|
| Product | $p = 1$ | $p = 2$ | $p = 3$ | $p = 4$ |
| 1 | 2500 | 2200 | 2150 | 2000 |
| 2 | 2200 | 1400 | 1800 | 2000 |
| 3 | 3000 | 3500 | 2500 | 2800 |
| 4 | 1000 | 2200 | 1500 | 1300 |

## C.7 Data for 5 planning periods

The economic data of the problem are given in Tables C.6,C.7,C.8.

Table C.5: Product price for the second case study for 4 planning periods.

| | Price ($/kg) | | | |
|---|---|---|---|---|
| Product | $p = 1$ | $p = 2$ | $p = 3$ | $p = 4$ |
| 1 | 300 | 280 | 250 | 260 |
| 2 | 180 | 150 | 160 | 165 |
| 3 | 160 | 190 | 180 | 170 |
| 4 | 120 | 120 | 130 | 140 |

Table C.6: Operating and transition cost for the second case study for 5 planning periods, $C^{inv} = 0.026, a_u = 10^6$.

| | $C^{oper}$ | | | | |
|---|---|---|---|---|---|
| Product | $p = 1$ | $p = 2$ | $p = 3$ | $p = 4$ | $p = 5$ |
| 1 | 23 | 10 | 15 | 25 | 15 |
| 2 | 30 | 25 | 20 | 18 | 28 |
| 3 | 45 | 55 | 40 | 45 | 50 |
| 4 | 50 | 29 | 20 | 25 | 10 |

## C.8 Data for 6 planning periods

The economic data of the problem are given in Tables C.9,C.10,C.11.

Table C.7: Product demand for the second case study for 5 planning periods.

| | Demand (kg/week) | | | | |
|---|---|---|---|---|---|
| Product | $p = 1$ | $p = 2$ | $p = 3$ | $p = 4$ | $p = 5$ |
| 1 | 2500 | 2200 | 2150 | 2000 | 2000 |
| 2 | 2200 | 1400 | 1800 | 2000 | 2200 |
| 3 | 3000 | 3500 | 2500 | 2800 | 2200 |
| 4 | 1000 | 2200 | 1500 | 1300 | 1500 |

Table C.8: Product price for the second case study for 5 planning periods.

| | Price ($\$/kg$) | | | | |
|---|---|---|---|---|---|
| Product | $p = 1$ | $p = 2$ | $p = 3$ | $p = 4$ | $p = 5$ |
| 1 | 300 | 280 | 250 | 260 | 240 |
| 2 | 180 | 150 | 160 | 165 | 185 |
| 3 | 160 | 190 | 180 | 170 | 190 |
| 4 | 120 | 120 | 130 | 140 | 150 |

## C.9 Data for 7 planning periods

The economic data for 7 planning periods are given in Tables C.12,C.13,C.14.

Table C.9: Operating and transition cost for the second case study for 6 planning periods, $C^{inv} = 0.026, a_u = 10^6$

| | $C^{oper}$ | | | | | |
|---|---|---|---|---|---|---|
| Product | $p = 1$ | $p = 2$ | $p = 3$ | $p = 4$ | $p = 5$ | $p = 6$ |
| 1 | 23 | 10 | 15 | 25 | 15 | 18 |
| 2 | 30 | 25 | 20 | 18 | 28 | 20 |
| 3 | 45 | 55 | 40 | 45 | 50 | 35 |
| 4 | 50 | 29 | 20 | 25 | 10 | 20 |

Table C.10: Product demand for the second case study for 6 planning periods.

| | Demand (kg/week) | | | | | |
|---|---|---|---|---|---|---|
| Product | $p = 1$ | $p = 2$ | $p = 3$ | $p = 4$ | $p = 5$ | $p = 6$ |
| 1 | 2500 | 2200 | 2150 | 2000 | 2200 | 2400 |
| 2 | 2200 | 1400 | 1800 | 2500 | 3500 | 3500 |
| 3 | 3000 | 3500 | 2500 | 2800 | 2250 | 2800 |
| 4 | 1000 | 2200 | 1500 | 2500 | 1600 | 2000 |

## C.10 Data for 8 planning periods

The economic data for 8 planning periods are given in Tables C.15,C.16,C.17.

Table C.11: Product price for the second case study for 6 planning periods.

| | Price ($/kg) | | | | | |
|---|---|---|---|---|---|---|
| Product | $p = 1$ | $p = 2$ | $p = 3$ | $p = 4$ | $p = 5$ | $p = 6$ |
| 1 | 300 | 280 | 250 | 260 | 240 | 290 |
| 2 | 180 | 150 | 160 | 165 | 185 | 260 |
| 3 | 160 | 190 | 180 | 170 | 190 | 220 |
| 4 | 120 | 120 | 130 | 140 | 150 | 190 |

Table C.12: Operating and transition cost for the second case study for 7 planning periods, $C^{inv} = 0.026, a_u = 10^6$

| | $C^{oper}$ | | | | | | |
|---|---|---|---|---|---|---|---|
| Product | $p = 1$ | $p = 2$ | $p = 3$ | $p = 4$ | $p = 5$ | $p = 6$ | $p = 7$ |
| 1 | 23 | 10 | 15 | 25 | 15 | 18 | 16 |
| 2 | 30 | 25 | 20 | 18 | 28 | 20 | 25 |
| 3 | 45 | 55 | 40 | 45 | 50 | 35 | 40 |
| 4 | 50 | 29 | 20 | 25 | 10 | 20 | 25 |

## C.11 Data for 9 planning periods

The economic data for 9 planning periods are given in Tables C.18,C.19,C.20.

Table C.13: Product demand for the second case study for 7 planning periods.

| | Demand (kg/week) | | | | | | |
|---|---|---|---|---|---|---|---|
| Product | $p = 1$ | $p = 2$ | $p = 3$ | $p = 4$ | $p = 5$ | $p = 6$ | $p = 7$ |
| 1 | 2500 | 2200 | 2150 | 2000 | 2200 | 2400 | 2350 |
| 2 | 2200 | 1400 | 1800 | 2500 | 3500 | 3500 | 3250 |
| 3 | 3000 | 3500 | 2500 | 2800 | 2250 | 2800 | 2500 |
| 4 | 1000 | 2200 | 1500 | 2500 | 1600 | 2000 | 2400 |

Table C.14: Product price for the second case study for 7 planning periods.

| | Price ($/$kg$) | | | | | | |
|---|---|---|---|---|---|---|---|
| Product | $p = 1$ | $p = 2$ | $p = 3$ | $p = 4$ | $p = 5$ | $p = 6$ | $p = 7$ |
| 1 | 300 | 280 | 250 | 260 | 240 | 290 | 310 |
| 2 | 180 | 150 | 160 | 165 | 185 | 260 | 255 |
| 3 | 160 | 190 | 180 | 170 | 190 | 220 | 200 |
| 4 | 120 | 120 | 130 | 140 | 150 | 190 | 100 |

## C.12 Data for 10 planning periods

The economic data for 10 planning periods are given in Tables C.21,C.22,C.23.

Table C.15: Operating and transition cost for the second case study for 8 planning periods, $C^{inv} = 0.026, a_u = 10^6$

| | $C^{oper}$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Product | $p = 1$ | $p = 2$ | $p = 3$ | $p = 4$ | $p = 5$ | $p = 6$ | $p = 7$ | $p = 8$ |
| 1 | 23 | 10 | 15 | 25 | 15 | 18 | 16 | 19 |
| 2 | 30 | 25 | 20 | 18 | 28 | 20 | 25 | 20 |
| 3 | 45 | 55 | 40 | 45 | 50 | 35 | 40 | 42 |
| 4 | 50 | 29 | 20 | 25 | 10 | 20 | 25 | 22 |

Table C.16: Product demand for the second case study for 8 planning periods.

| | Demand (kg/week) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Product | $p = 1$ | $p = 2$ | $p = 3$ | $p = 4$ | $p = 5$ | $p = 6$ | $p = 7$ | $p = 8$ |
| 1 | 2500 | 2200 | 2150 | 2000 | 2000 | 2400 | 2350 | 2000 |
| 2 | 2200 | 1400 | 1800 | 2300 | 3500 | 3500 | 1500 | 2600 |
| 3 | 3000 | 3500 | 2500 | 2800 | 2800 | 2800 | 2000 | 2500 |
| 4 | 1000 | 2000 | 1600 | 2500 | 2000 | 2000 | 2400 | 1850 |

Table C.17: Product price for the second case study for 8 planning periods.

| | Price ($\$/kg$) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Product | $p = 1$ | $p = 2$ | $p = 3$ | $p = 4$ | $p = 5$ | $p = 6$ | $p = 7$ | $p = 8$ |
| 1 | 300 | 280 | 250 | 260 | 240 | 290 | 310 | 290 |
| 2 | 180 | 150 | 160 | 165 | 185 | 260 | 255 | 240 |
| 3 | 160 | 190 | 180 | 170 | 190 | 220 | 200 | 260 |
| 4 | 120 | 120 | 130 | 140 | 150 | 190 | 100 | 140 |

Table C.18: Operating and transition cost for the second case study for 9 planning periods, $C^{inv} = 0.026, a_u = 10^6$

| | $C^{oper}$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Product | $p = 1$ | $p = 2$ | $p = 3$ | $p = 4$ | $p = 5$ | $p = 6$ | $p = 7$ | $p = 8$ | $p = 9$ |
| 1 | 23 | 10 | 15 | 25 | 15 | 18 | 16 | 19 | 15 |
| 2 | 30 | 25 | 20 | 18 | 28 | 20 | 25 | 20 | 19 |
| 3 | 45 | 55 | 40 | 45 | 50 | 35 | 40 | 42 | 45 |
| 4 | 50 | 29 | 20 | 25 | 10 | 20 | 25 | 22 | 20 |

Table C.19: Product demand for the second case study for 9 planning periods.

| | Demand (kg/week) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Product | $p=1$ | $p=2$ | $p=3$ | $p=4$ | $p=5$ | $p=6$ | $p=7$ | $p=8$ | $p=9$ |
| 1 | 2500 | 2200 | 2150 | 2000 | 2200 | 2400 | 2350 | 2350 | 2200 |
| 2 | 2200 | 1400 | 1800 | 2500 | 3500 | 3500 | 3250 | 2400 | 2500 |
| 3 | 3000 | 3500 | 2500 | 2800 | 2250 | 2800 | 2500 | 2500 | 2400 |
| 4 | 1000 | 2000 | 1500 | 2500 | 1600 | 2000 | 2000 | 1800 | 1000 |

Table C.20: Product price for the second case study for 9 planning periods.

| | Price ($\$/kg$) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Product | $p=1$ | $p=2$ | $p=3$ | $p=4$ | $p=5$ | $p=6$ | $p=7$ | $p=8$ | $p=9$ |
| 1 | 300 | 280 | 250 | 260 | 240 | 290 | 310 | 290 | 320 |
| 2 | 180 | 150 | 160 | 165 | 185 | 260 | 255 | 240 | 235 |
| 3 | 160 | 190 | 180 | 170 | 190 | 220 | 200 | 260 | 275 |
| 4 | 120 | 120 | 130 | 140 | 150 | 190 | 100 | 140 | 150 |

Table C.21: Operating and transition cost for the second case study for 10 planning periods, $C^{inv} = 0.026, a_u = 10^6$

| | $C^{oper}$ | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Product | $p=1$ | $p=2$ | $p=3$ | $p=4$ | $p=5$ | $p=6$ | $p=7$ | $p=8$ | $p=9$ | $p=10$ |
| 1 | 23 | 10 | 15 | 25 | 15 | 18 | 16 | 19 | 15 | 18 |
| 2 | 30 | 25 | 20 | 18 | 28 | 20 | 25 | 20 | 19 | 22 |
| 3 | 45 | 55 | 40 | 45 | 50 | 35 | 40 | 42 | 45 | 38 |
| 4 | 50 | 29 | 20 | 25 | 10 | 20 | 25 | 22 | 20 | 23 |

Table C.22: Product demand for the second case study for 10 planning periods.

| | Demand (kg/week) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Product | $p=1$ | $p=2$ | $p=3$ | $p=4$ | $p=5$ | $p=6$ | $p=7$ | $p=8$ | $p=9$ | $p=10$ |
| 1 | 2500 | 2200 | 2150 | 2000 | 2200 | 2400 | 2350 | 2350 | 2200 | 2300 |
| 2 | 2200 | 1400 | 1800 | 2500 | 3500 | 3500 | 3250 | 2400 | 2500 | 2500 |
| 3 | 3000 | 3500 | 2500 | 2800 | 2250 | 2800 | 2500 | 2500 | 2400 | 2000 |
| 4 | 1000 | 2000 | 1500 | 2500 | 1600 | 2000 | 2000 | 1800 | 1000 | 1100 |

Table C.23: Product price for the second case study for 10 planning periods.

| | Price ($/kg) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Product | $p = 1$ | $p = 2$ | $p = 3$ | $p = 4$ | $p = 5$ | $p = 6$ | $p = 7$ | $p = 8$ | $p = 9$ | $p = 10$ |
| 1 | 300 | 280 | 250 | 260 | 240 | 290 | 310 | 290 | 320 | 315 |
| 2 | 180 | 150 | 160 | 165 | 185 | 260 | 255 | 240 | 235 | 245 |
| 3 | 160 | 190 | 180 | 170 | 190 | 220 | 200 | 260 | 275 | 255 |
| 4 | 120 | 120 | 130 | 140 | 150 | 190 | 100 | 140 | 150 | 180 |