Batch Scheduling of Multi-Product Pipeline Networks

S.J. Vlot

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Batch Scheduling of Multi-Product Pipeline Networks

Master of Science Thesis

For the degree of Master of Science in Systems and Control
at Delft University of Technology

S.J. Vlot

February 2017

Faculty of Mechanical, Maritime, and Materials Engineering · Delft University of Technology

The research in this thesis is carried out in cooperation with ORTEC Consulting.

Abstract

In oil supply chains, crude oil needs to be transported from oil fields to refineries, and refined products need to be transported from refineries to regional depots. On land, pipelines are the preferred mode for long-distance oil transportation, because they are safe, efficient, silent, and cheap compared to other modes of transport. Pipelines are often part of large networks, in which they connect multiple supply and demand locations. Multi-product pipelines transport batches of different products, such as gasoline, diesel, and jet fuel.

Pipeline networks should be operated such that temporal and spatial differences between supply and demand are balanced, operational limitations are satisfied, and costs are minimized. This is a rather complicated task, due to the size and complexity of pipeline networks, limited capacities of tanks and pipelines, and the existence of transportation times of several days. Because batches are pushed through pipelines, transportation times of current batches depend on injections of future batches, which is a distinctive feature of the pipeline scheduling problem. The minimization of operational cost is mainly related to transmix volumes, i.e. contaminated volumes that emerge between consecutive batches, and pumping energy.

In this thesis, we propose a novel pipeline scheduling method for solving the pipeline scheduling problem. It consists of a planning and a scheduling phase that are coupled in a hierarchical decomposition scheme. In the planning phase, global day-to-day transportation volumes are determined for each pipeline. In the scheduling phase, we use the planning output to generate complete schedules. Both phases contain a discrete-time Mixed Integer Linear Programming (MILP) problem. The MILP planning problem is solved with truncated branch and bound. The MILP scheduling problem is further decomposed using a rolling-horizon approach; the resulting subproblems are solved with branch and bound.

The pipeline scheduling method has been successfully tested on two case studies involving up to 4 products, 8 pipelines, 8 tank farms, 2 supply locations, and 5 demand locations. The proposed method is flexible in terms of network configurations, intermediate supply and demand requirements, and cost structures. Complete schedules for 30-day horizons are obtained within 3–4 minutes of computation time.

With respect to current industry practice, the novel pipeline scheduling method can greatly reduce the time required to generate schedules. Compared to current spreadsheet approaches, the proposed method is generic and less error-prone. Moreover, the obtained schedules are significantly better in terms of transmix and pumping costs.

Acknowledgments

For almost one year, I have been working on my thesis project at Delft University of Technology and ORTEC Consulting. Before I present the main findings, I would like to thank those of you who have supported me during the past year.

First, I would like to thank my supervisors at Delft University of Technology and ORTEC Consulting for giving me the opportunity to graduate on a very interesting project. In particular, I would like to thank prof.dr.ir. Bart De Schutter for his detailed and high-quality feedback, dr.ir. Noud Gademann for the interesting discussions that were often confusing and enlightening at the same time, and ir. Gregor Brandt for coaching me both on a professional and on a personal level.

Next, I would like to thank dr.ir. Ton van den Boom and dr.ir. Theresia van Essen for their interest in this thesis project and for being part of my graduation committee.

In addition, my thanks go to my colleagues at ORTEC for keeping me in touch with the real world during countless coffee breaks and Friday afternoon drinks. And last but not least, I would like to thank my family, girlfriend, and friends for their support and for taking my mind of the project when needed.

Delft, University of Technology S.J. Vlot
February 2017

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1-1 Pipelines in oil supply chains

In oil supply chains, crude oil needs to be transported from oil fields to refineries, and refined oils need to be transported from refineries to regional depots. On land, pipelines are the preferred mode for long-distance oil transportation, because they are safe, efficient, silent, and cheap compared to other modes of transport. In multi-product pipeline networks, pipelines transport batches of different products such as gasoline, diesel, and jet fuel. These pipelines connect different tank farms that act as supply, demand, or intermediary location, see Fig. 1-1.

The size of existing oil pipeline networks is enormous and still growing. In the United States, liquids pipelines stretch over 199 000 miles (320 000 km). In 2014, a total volume of 16.2 billion barrels ($2.58 \cdot 10^9$ m^3) was delivered, consisting of crude oil, refined products, and liquefied natural gas (Association of Oil Pipe Lines and American Petroleum Institute, 2015).

In pipeline networks, temporal and spatial differences between supply and demand should be balanced. In order to achieve this, product batches are pumped through the network. Planning and scheduling these batches is a rather complicated task, due to the size and complexity of existing networks, limited tank capacity in tank farms, and the existence of significant transportation times of several days. Because batches are pushed through the pipeline, transportation times of current batches depend on injections of future batches, which is a distinctive feature of the pipeline scheduling problem.

In addition to complying with supply and demand, pipeline schedules can be better or worse in terms of operational cost. First of all, batches are injected without any separation in between them, which leads to some mixing at batch interfaces. This is illustrated in Fig. 1-2, which shows a pipeline that contains three batches. The amount of contamination can be reduced by minimizing the number of batch transitions. Second, pipeline networks require energy for pumping. The required power varies with the flow rate, since pumps have different efficiencies at different flow rates. In addition, the required pumping pressure increases with the flow rate due to increasing friction losses in pipelines.

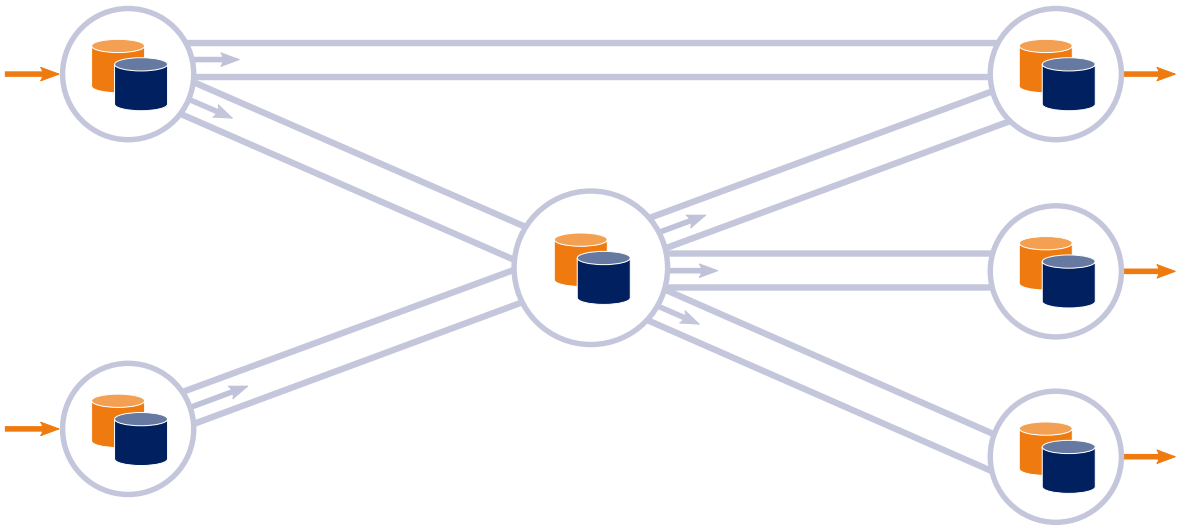


Figure 1-1: Example of a network with six pipelines and six tank farms

Figure 1-2: Example of a pipeline containing three batches

1-2 Problem statement

Given the size and dynamics of pipeline networks, planning and scheduling network operations is quite complicated. In a monthly cycle, network schedulers spend several days on this task. Some schedulers use automated scheduling rules based on expert knowledge. However, most companies still rely on basic spreadsheets. The use of mathematical optimization can improve this situation, since it offers a structured way for creating feasible and good schedules.

This project is carried out in cooperation with ORTEC Consulting, a quantitative consultancy firm interested in scheduling methods for multi-product pipeline networks. The desired deliverable is a scheduling tool that generates complete schedules for a 30-day horizon within a few minutes. The scheduling tool must be built in AIMMS, a commercial software package for building optimization-based applications. For ORTEC Consulting, there are two important considerations from a business perspective. First of all, computation time is important. Hence, the most suitable solution method is not necessarily exact. Second, the scheduling tool should be easy to configure for different clients.

This leads to the following problem statement:

Develop a fast and generic method for creating feasible and good schedules for multi-product oil pipeline networks with known supply and demand.

In this problem statement, *fast* refers to the amount of computation time required to obtain complete schedules, which should be a few minutes at most. Furthermore, the method should be *generic* in terms of network configurations and scheduling objectives. A *schedule* should describe a list of batch injections for each pipeline, specifying batch volumes, product types, start and end times, and flow rates. Schedules should be *feasible* and *good*, i.e. operational restrictions on tank levels and flow rates must be obeyed, and operational costs related to pumping energy and transmix volumes should be minimized. The method should be able to handle *networks* with at least 4 products and 5–10 long-distance pipelines. Lastly, *supply* and *demand* are known for the entire horizon, which means that supply and demand volumes are specified for each location in the network and for each product type.

1-3 Research questions

The problem statement, which has been introduced in Section 1-2, is captured in the following main research question:

How can we solve the pipeline scheduling problem using mathematical optimization?

We split the main research question into four subquestions:

- Which optimization framework is most suitable for the pipeline scheduling problem?
- Which problem representation is most suitable for the pipeline scheduling problem?
- How can we include operational cost in the optimization problem?
- What is the effect of different solution techniques on computation time and quality?

1-4 Thesis outline

This thesis report consists of five chapters. Chapter 2 covers general batch scheduling techniques, including optimization frameworks, problem representations, and solution techniques. Existing pipeline scheduling methods are categorized according to these scheduling techniques. In Chapter 3, we propose a novel pipeline scheduling method. We explain high-level choices and provide a detailed description of the optimization problems. Chapter 4 describes the results of two different case studies. Chapter 5 concludes this thesis and gives recommendations for future work.

1-5 Contributions

Existing pipeline scheduling methods are either aimed at finding optimal schedules for simplified problems, or aimed at finding feasible schedules for very large problems. Currently, there is no pipeline scheduling method for generating good or optimal schedules for medium-sized problems as described in Section 1-2. Hence, we propose a novel pipeline scheduling method aimed at finding good schedules for such problems. The proposed method is flexible in terms of network configurations, intermediate supply and demand requirements, and cost structures. Other contributions with respect to existing pipeline scheduling methods are:

- *Custom discrete-time representation*
  Most existing pipeline scheduling methods rely on continuous-time or precedence-based problem representations, because these representations yield accurate results and require less binary variables than discrete-time representations. However, it is difficult to incorporate external inputs in these representations due to a lack of fixed reference points in time. In discrete-time representations, on the other hand, it is relatively easy to include intermediate deadlines, shared use of resources, and inventory balances. In this thesis, we use a custom discrete-time representation in which batches can start and end *within* time buckets. The actual timing of these batches is determined in a post-processing step. In this way, we obtain high-resolution schedules with a small number of time buckets.

- *Incorporation of the batch front in discrete-time representations*
  Since batches are pushed through pipelines, transportation times of current batches depend on injections of future batches. Cafaro and Cerdá (2004) introduce the concept of a batch front variable to track the positions of batches in a pipeline over time. When a batch reaches the end of the pipeline, its ejection starts. In this thesis project, we adjust the continuous-time batch front representation of Cafaro and Cerdá (2004) such that it fits discrete-time problem representations.

- *Piecewise-affine representation of pumping costs*
  The energy costs related to pumping increase nonlinearly with the flow rate. In existing pipeline scheduling methods, these nonlinear pumping costs are either linearized, or included in a nonlinear optimization problem. We use a piecewise-affine pumping cost formulation such that the resulting optimization problem is accurate and can be solved within reasonable time.

Chapter 2

Batch scheduling techniques

2-1 Introduction

This chapter covers both general batch scheduling techniques and existing pipeline scheduling methods. We divide the process of solving scheduling problems in three main steps, which relate to the research questions stated in Section 1-3. These steps are:

1. Definition of the business question (decisions, goals, and boundary conditions)
2. Formulation of mathematical optimization problems that capture the business question
3. Selection of solution techniques and algorithms for solving the optimization problems

Fig. 2-1 illustrates the chapter outline, which is based on the three steps described above. Sections 2-2 to 2-4 describe general batch scheduling techniques. Section 2-2 covers two optimization frameworks that can be used to describe the pipeline scheduling problem. Section 2-3 covers the solution techniques associated with these optimization frameworks. Depending on the chosen optimization framework and solution techniques, different problem representations exist. We describe these problem representations in Section 2-4.

Section 2-5 covers existing pipeline scheduling methods. We classify these methods based on the batch scheduling techniques that are described in Sections 2-2 to 2-4.

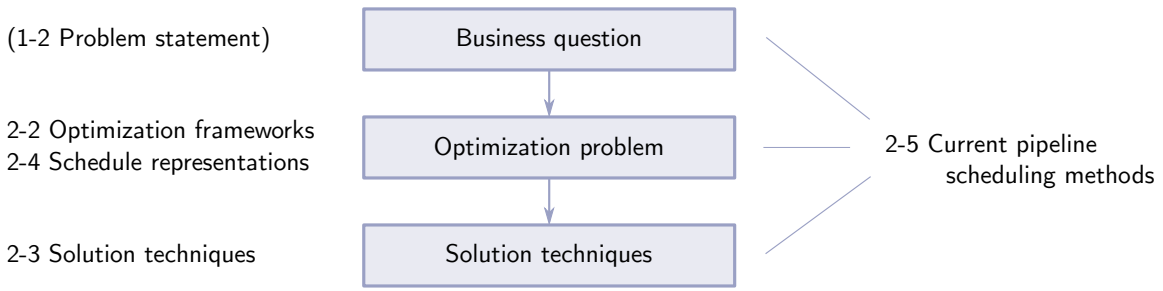


Figure 2-1: Outline of Chapter 2

2-2 Optimization frameworks

2-2-1 Mathematical programming

A mathematical program is a mathematical description of an optimization problem. It consists of variables, an objective function indicating what is best in terms of the variables, and constraints indicating what is allowed in terms of the variables.

Mathematical programming formulations are based on algebra and can be classified accordingly. Linear Programming (LP) problems are characterized by linear objective functions and constraints. Adding some nonlinear terms results in Non-Linear Programming (NLP) problems. Moreover, if some variables only take integer values, we obtain Mixed Integer Linear Programming (MILP) and Mixed Integer Non-Linear Programming (MINLP) problems.

As most scheduling problems involve discrete decisions, they are typically formulated as MILP problems. For example, we can use binary variables to describe task-machine assignments. MILP problems with binary variables are of the following form, e.g. see Grossmann (2014):

$$\begin{aligned} \min \quad & c^{\mathrm{T}} y + d^{\mathrm{T}} x \\ \text{s.t.} \quad & Ay + Bx \leq b \\ & y \in \{0,1\}^m \\ & x \in \mathbb{R}^n_{\geq 0} \end{aligned} \tag{2-1}$$

where b, c, d are vectors of coefficients and A, B are matrices of coefficients. Furthermore, x and y are vectors of continuous and binary decision variables, respectively.

2-2-2 Constraint programming

Constraint programming (Van Hentenryck, 1989) is a programming paradigm developed to solve feasibility problems in artificial intelligence applications. It has been extended to solve optimization problems and, in particular, scheduling problems (Méndez et al., 2006).

Constraint programming is more expressive than mathematical programming, because (1) it supports direct incorporation of logic and arithmetic expressions, (2) it allows variables to be indexed by each other, and (3) it offers compact formulations — global constraints — for representing specific scheduling constructions (Méndez et al., 2006). For example, *end-before-start* describes precedence relations and *cumulative* expresses capacity restrictions, e.g. in storage facilities and machines. The general form is (Lustig and Puget, 2001):

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g_j(x) = 1 \quad \forall j \in J \\ & x \in X \end{aligned} \tag{2-2}$$

where f and g are the objective and constraint functions, respectively. Furthermore, J is a set of constraint indices. The expression $g_j(x) = 1$ states that all constraints should be satisfied.

Although the terms mathematical programming and constraint programming suggest a degree of similarity, they relate to different types of programming. Constraint programming refers to *computer programming*, whereas mathematical programming refers to *making a plan*. The main implication is that a constraint program should also describe a way to solve it (Lustig and Puget, 2001). However, most constraint programming solvers contain generic search strategies. Therefore, they only require an optimization problem as an input.

2-3 Solution techniques

2-3-1 Mathematical programming

This section focuses on solving MILP problems, as scheduling problems are typically formulated as such. In general, the number of integer MILP solutions increases exponentially with the number of binary variables. Although computers and algorithms have improved significantly (Bixby and Rothberg, 2007), current exact methods cannot effectively evaluate these solutions for large instances. Instead, heuristics can be used. Heuristics cannot guarantee optimality, but often return good solutions within short computation times.

Exact methods

The most common exact solution method for MILP problems is *branch and bound* (Land and Doig, 1960; Dakin, 1965), which is the basis of commercial solvers as IBM ILOG CPLEX. Let us assume that all integer variables are binary, which is true for most scheduling problems. Then, the algorithm starts by solving a relaxed LP problem in which the binary variables are treated as continuous. The corresponding solution is a lower bound on the original minimization problem. If the relaxation does not yield integer values, a search tree is initiated. Two subproblems are solved, in which one binary variable is fixed to respectively 0 and 1. If the subproblem results in an integer solution, that particular branch is complete and its solution serves as an upper bound. If not, the process of branching repeats. The algorithm continues to solve subproblems until the lower and upper bound have the same objective value. During the search, the lower bounds are used to prune branches of the search tree.

Heuristics

Heuristics are typically classified as construction heuristics or metaheuristics.

Construction heuristics generate schedules from scratch, e.g. by applying priority rules. Priority rules basically describe in which order a queue of tasks should be processed. Haupt (1989) distinguishes rules based on arrival time (e.g. first come first serve), processing time (e.g. shortest processing time first), static due date (e.g. earliest due date first), and dynamic due date (e.g. least slack first). Single priority rules can be combined to obtain more advanced scheduling policies (Panwalkar and Iskander, 1977).

Metaheuristics are improvement heuristics, i.e. they can only improve existing solutions. After initialization, metaheuristics try to find better solutions by combining multiple existing solutions, or by performing a (stochastic) search in the neighborhood of existing solutions (Harjunkoski et al., 2014). Well-known metaheuristics are tabu search (Glover, 1986), simulated annealing (Kirkpatrick et al., 1983), and genetic algorithms (Holland, 1975).

2-3-2 Constraint programming

Constraint programming algorithms enumerate solutions in a search tree. Similar to branch and bound, a tree of subproblems is created by assigning values to discrete variables. Rather than using LP relaxations, branches are pruned using constraint propagation. In essence, an algorithm reduces the search space by checking and eliminating logic inconsistencies. If the search space in a branch becomes empty, the search continues in a different branch. For frequently occurring problem structures, high-level building blocks have been developed. These *global constraints* are propagated with specialized algorithms (Hooker, 2002, p. 186).

In general, constraint programming algorithms are aimed at finding feasible rather than optimal solutions. Optimization can be done by adding constraints that require the next solution to be better than the current one. It is possible to prove the optimality of a solution, although this can be time consuming. Proving optimality is done by showing that no better solutions exist, i.e. the feasible domain of decision variables becomes empty (Lustig and Puget, 2001).

2-3-3 Decomposition methods

If an optimization problem is too large to be solved monolithically, we can use decomposition methods to split a problem into smaller parts that are easier to solve. These parts can be described and solved using (combinations of) the techniques discussed in Sections 2-3-1 and 2-3-2.

Hierarchical and iterative decomposition

In hierarchical and iterative decomposition, the original problem is split into a master problem and a subproblem. The master problem is typically a simplified or aggregated version of the original problem. Its solution is a set of high-level decisions and forms the input of a detailed subproblem. The subproblem is solved to obtain a complete solution, e.g. a schedule. A challenge of hierarchical and iterative decomposition methods is that the master problem is often too restrictive or not restrictive enough, yielding respectively no or suboptimal results. In *iterative decomposition*, this is overcome by introducing a feedback loop that adjusts the master problem based on the results of the subproblem (Maravelias and Sung, 2009).

Two well-known iterative decomposition methods are Benders and Lagrangian decomposition. Note that these methods are not based on simplified or aggregated versions of the original problem. Instead, the master and subproblem are extracted from the original formulation by exploiting the structure of variables and constraints (Maravelias and Sung, 2009).

Time-based decomposition: rolling horizon

In time-based decomposition, the scheduling horizon is solved in stages. The main idea is to solve the problem recursively by shifting a time window, which is referred to as *rolling the horizon*. In each iteration, the problem is solved for a limited horizon. From the solution, only the decisions for the first time period(s) are implemented. These decisions form the initial conditions of the next iteration, in which the horizon is shifted (Harjunkoski et al., 2014).

A note on hybrid solution techniques

Different solution techniques have complementary strengths. Therefore, combining multiple methods can greatly improve computational performance (Harjunkoski et al., 2014).

The combination of heuristics and branch and bound might be beneficial when the problem can be decomposed into subproblems for which good heuristics exist. For example, Maravelias (2006) uses branch and bound to solve an assignment master problem and a heuristic to solve the sequencing subproblem. Since the assignment decisions are fixed in the subproblem, each machine can be sequenced independently using job-shop scheduling heuristics.

The combination of constraint programming algorithms and branch and bound can also be advantageous. Méndez et al. (2006) state that assignment decisions are efficiently solved with branch and bound because they have good continuous relaxations. Sequencing decisions, however, are better suited for constraint programming approaches since effective global sequencing constraints exist. This approach can be implemented using a logic-based Benders decomposition (Hooker, 2002).

2-4 Problem representations

Depending on the chosen optimization framework and solution techniques, different problem representations exist. Most scheduling surveys distinguish *network-oriented* and *batch-oriented* problem representations, e.g. see Méndez et al. (2006) and Harjunkoski et al. (2014). The conceptual difference between these problem representations is related to the representation of batches in shared network equipment:

- In network-oriented representations, batch interaction is constrained via shared time-grid points that are linked to network equipment. For example, batch overlap is prevented by allowing at most one active batch at each time point per machine.
- In batch-oriented representations, batch starting times are not linked to a point on a time grid. Instead, starting times are expressed by a continuous or integer variable. Constraints to prevent overlap contain additional batch sequencing variables.

Both network-oriented and batch-oriented problem representations can be used in the mathematical programming framework. Most constraint programming problems are based on batch-oriented representations, because effective global constraints are available for describing and solving these problems. An important aspect of batch-oriented representations is that batches and batch sizes must be known in advance. Network-oriented representations, on the other hand, consider batching and scheduling decisions simultaneously (Méndez et al., 2006).

In the remainder of this section, we describe three common problem representations. Sections 2-4-1 and 2-4-2 respectively cover discrete-time representations and continuous-time representations, which are both network oriented. Section 2-4-3 covers precedence-based representations, which are batch oriented.

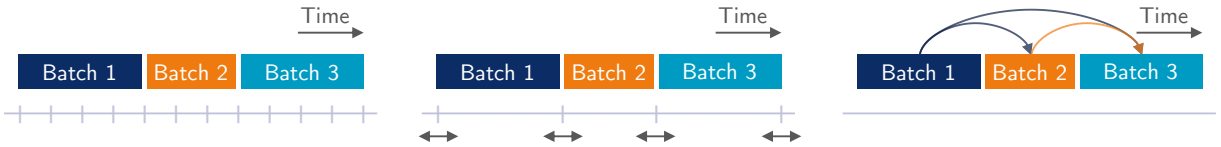
2-4-1 Discrete-time problem representations

Discrete-time representations are based on uniformly distributed time grids. Batches can start and end at these time points only, as illustrated in Fig. 2-2a.

Main decision variables

The main decision variables in discrete-time representations are (Shah et al., 1993):

$W_{ijt} \in \{0, 1\}$	Binary: indicates whether task i starts on machine j at time point t
$B_{ijt} \geq 0$	Batch size of task i that starts on machine j at time point t
$S_{st} \geq 0$	Inventory level of material s at time point t



(a) Discrete-time representation (network oriented)

(b) Continuous-time representation (network oriented)

(c) Precedence-based representation (batch oriented)

Figure 2-2: Different representations of a schedule with three consecutive batches

Strengths

The main advantage of discrete-time representations is simplicity. Due to the existence of fixed reference points, it is relatively easy to express intermediate deadlines, shared use of resources, and inventory balances. This enables straightforward integration of planning and scheduling in hierarchical decomposition approaches (Maravelias and Sung, 2009). Furthermore, costs per unit volume per unit time — such as inventory costs — result in linear expressions, because the duration of time intervals is fixed (Méndez et al., 2006). Sundaramoorthy and Maravelias (2011) state that discrete-time representations are probably the best choice for representing real-world scheduling problems.

Weaknesses

The major drawback of discrete-time representations is that dense time grids are required to obtain accurate results. As the number of time points is linked to the number of binary variables, the resulting optimization problems are typically large. In addition, most discrete-time representations assume that batch processing times are (1) constant and (2) an integer number of time steps. Nevertheless, variable processing times can be incorporated by introducing task modes with different durations (Kondili et al., 1993) or by using the mixed-time representation by Maravelias (2005), in which batches can end in between time points.

2-4-2 Continuous-time problem representations

In continuous-time representations, batches are assigned to time points with a variable timing, see Fig. 2-2b. The time values of these time points are determined during optimization.

Main decision variables

Compared to discrete-time representations, continuous-time representations involve an additional time variable:

$W_{ijt} \in \{0, 1\}$	Binary: indicates whether task i starts on machine j at time point t
$B_{ijt} \geq 0$	Batch size of task i that starts on machine j at time point t
$S_{st} \geq 0$	Inventory level of material s at time point t
$T_t \geq T_{t-1} \geq 0$	Time value of time point t

Strengths

Méndez et al. (2006) state that the main advantages of continuous-time representations are accuracy and problem size. A small number of time points can be used to represent schedules with very precise timing, which results in MILP problems with a much smaller number of binary variables compared to discrete-time approaches. Furthermore, variable processing times are easily included in the standard formulation, as time point locations are not fixed.

Weaknesses

Continuous-time representations have several disadvantages. First of all, it is difficult to incorporate external inputs, e.g. intermediate deadlines, due to a lack of fixed reference points. Second, costs defined per unit volume per unit time result in nonlinear objective functions, as time steps have variable durations. Furthermore, selecting an appropriate number of time points is difficult and important. Underestimation leads to suboptimal or even infeasible results, whereas overestimation yields large problems. Despite the smaller number of binary variables, continuous-time problems may take longer to solve than discrete-time problems, as continuous-time problems have weaker LP relaxations (Sundaramoorthy and Maravelias,

2011). An LP relaxation is considered weak when its feasible region is much larger than the feasible region of the corresponding MILP problem. Then, the LP solution yields a less effective lower bound, which reduces the possibilities for branch pruning in branch and bound.

2-4-3 Precedence-based problem representations

Precedence-based representations involve decision variables that explicitly state batch sequences, see Fig. 2-2c. Most precedence-based representations only consider assignment and sequencing decisions. Hence, batch-sizing decisions should be made in advance.

Main decision variables and constraints in mathematical programming formulations

In mathematical programming formulations, a binary sequencing variable $X_{i,i'}$ indicates whether or not batch i' is processed after batch i, and a binary assignment variable $Y_{i,j}$ indicates whether or not batch i is assigned to machine j. A continuous or integer variable T_i represents the batch starting time. Big-M constraints are required to ensure consistent sequencing (Harjunkoski et al., 2014):

$$T_i + p_i \leq T_{i'} + M(3 - X_{i,i'} - Y_{i,j} - Y_{i',j}) \qquad \forall j, i, i', i \neq i' \qquad (2\text{-}3)$$

$$T_{i'} + p_{i'} \leq T_i + M(2 + X_{i,i'} - Y_{i,j} - Y_{i',j}) \qquad \forall j, i, i', i \neq i' \qquad (2\text{-}4)$$

where p_i is the processing time of batch i and M is a parameter with a sufficiently large value, e.g. the horizon length. If batches i and i' are assigned to machine j and batch i precedes batch i', then Eq. (2-3) reduces to $T_i + p_i \leq T_{i'}$. That is, batch i' cannot start before batch i has finished. The constraint in Eq. (2-4) has a similar structure and is active when $X_{i,i'} = 0$.

Main decision variables and constraints in constraint programming formulations

In constraint programming problems, batches and machines are described as tasks and resources with associated parameters and variables. The following global constraint describes that task assignments and starting times should be chosen such that the utilization of a resource is less than or equal to its capacity (Beldiceanu and Carlsson, 2002):

$$\textbf{cumulatives}(\text{tasks}[i],\ \text{resources}[j],\ \leq) \qquad (2\text{-}5)$$

Strengths

The main advantage of precedence-based representations is that they explicitly address the sequential use of shared processing units with sequencing variables. Therefore, sequence-dependent changeover costs are easily incorporated. In case of mathematical programming, high-quality solutions can be obtained with small problem formulations, as the number of binary decision variables is not linked to the number of points on a time grid. In case of constraint programming, global constraints and specialized algorithms are available for expressing and solving scheduling problems.

Weaknesses

Batch-sizing decisions are an input in precedence-based representations. Therefore, these decisions should be made in advance. In mathematical programming formulations, inventory limitations are hard to incorporate due to a lack of global reference points in time. Moreover, Big-M constraints might slow down branch and bound. In case of constraint programming, one is very dependent on the existence of global constraints. Expressing basic scheduling elements will not be problematic. However, if more elaborate features need to be included, e.g. variable processing times, constraint programming might be less effective.

2-5 Current pipeline scheduling methods

Pipeline scheduling problems occur in different industries. Although the main focus of this thesis report is on refined oil networks, we shortly discuss methods for other networks as well since they may provide new insights.

2-5-1 Scheduling refined oil networks

Over the past years, researchers have proposed various scheduling methods for refined oil pipeline networks. To the best of our knowledge, there are three main pipeline scheduling methods for mesh-structure networks, i.e. Cafaro and Cerdá (2012), Boschetto et al. (2010), and Lopes et al. (2010). Prior to discussing these methods, we describe the first MILP approach for pipeline scheduling (Rejowski and Pinto, 2003).

Method by Rejowski and Pinto

The first MILP approach for scheduling multi-product pipeline systems is proposed by Rejowski and Pinto (2003, 2004). They present a discrete-time, discrete-volume MILP problem for a single pipeline with one source and multiple destinations. Schedules are optimized in terms of batch transitions, inventory costs, and pumping costs per unit volume. Problematic aspects are the discretization of pipeline volume and time, as both lead to discretization errors. Furthermore, the discrete-time approach results in a large number of binary variables, which makes this method intractable for large problem instances.

Rejowski and Pinto (2008) transform their previous work into a continuous-time formulation by introducing a variable time grid. The resulting MINLP problem contains bilinear terms in the objective function, as variable cost terms are multiplied by variable step sizes. The optimization problem also includes flow-rate-dependent pumping costs, which are described by third-degree polynomials. The authors use an outer-approximation algorithm (see Appendix A) to obtain locally optimal solutions for small problem instances.

Method by Cafaro and Cerdá

Cafaro and Cerdá (2004) present a continuous-time MILP formulation for the case introduced by Rejowski and Pinto (2003). Their method overcomes several limitations of the previous method, including the issues regarding problem size and volume discretization. As mentioned in Section 2-4-2, it is difficult to incorporate cost structures and intermediate demand deadlines in continuous-time representations. Cafaro and Cerdá (2004) deal with this by only considering demand at the end of the scheduling horizon. Moreover, costs that depend on both volume and time are approximated by costs per unit volume. Another challenge is to choose the right number of time points. Cafaro and Cerdá (2004) suggest to solve the MILP problem with an increasing number of time points until the objective value stops improving.

Cafaro and Cerdá (2012) extend their previous method to solve scheduling problems for mesh-structure networks involving up to six pipelines. Although the results look promising, it is unclear how the MILP formulation will scale for larger networks. We expect that the required number of time points will increase, assuming that batch starting times will be different in every pipeline. In another extension, Cafaro et al. (2015) incorporate nonlinear pumping costs and solve the resulting MINLP problem with outer approximation. A substantial cost improvement is observed. However, the computation time increases significantly. Therefore, the proposed MINLP approach is not suitable for real-world scheduling applications.

Method by Boschetto et al.
Boschetto et al. (2010) propose a hybrid hierarchical decomposition method for scheduling 30 pipelines of the Brazilian Petrobras network. The solution method consists of different blocks, i.e. planning, assignment, sequencing, simulation, and timing. In the planning block, a continuous-time MILP problem is solved to determine global transportation volumes. In the assignment block, a construction heuristic determines batch volumes, pipeline routes, time windows, and an initial batch sequence. In the sequencing block, a genetic algorithm improves the initial batch sequence. The simulation block determines min/max bounds on flow rates. A precedence-based MILP timing problem is solved to obtain complete schedules.

Although this method generates complete solutions for networks of impressive size, the quality of the solutions is unclear. The authors clearly aim at feasibility. However, it is not guaranteed that a 5-step hierarchical decomposition approach without feedback mechanisms will return feasible schedules. Moreover, it is not clear what should be done when infeasibilities occur.

Method by Lopes et al.
Lopes et al. (2010) propose a constraint programming method for the network studied by Boschetto et al. (2010). They decompose the overall problem into a planning and a scheduling phase with a feedback loop in case no feasible schedules exist. In the planning phase, a randomized construction heuristic generates a list of batches with details regarding products, volumes, origins, destinations, pipeline routes, and delivery deadlines. The output of the planning phase is used in the scheduling phase, in which batch sequences and pumping times are determined. The scheduling phase consists of two constraint programming problems that cover sequencing and timing decisions separately.

The selection of global constraints is extensively studied and documented. Compared to the method by Boschetto et al. (2010), finding feasible solutions is more likely due to the iterative decomposition approach.

2-5-2 Scheduling other networks

This section briefly describes the scheduling methods for crude oil, water, and gas networks.

Crude oil scheduling is equivalent to refined oil scheduling with one exception: different crude oils are blended in order to meet quality requirements. Blending terms are typically bilinear — fraction $\times$ volume —, yielding nonlinear optimization problems. Zhang and Xu (2015) describe the first MINLP formulation for scheduling a single long-distance crude oil pipeline. They propose a custom outer-approximation algorithm to solve the MINLP problem.

The scheduling methods for water and gas pipelines have a strong emphasis on the fluid dynamics aspect of pipeline networks, e.g. see D'Ambrosio et al. (2015) and Ríos-Mercado and Borraz-Sánchez (2015). The multi-product aspect is either non-existent (water networks) or neglected (gas networks). Important problem features are nonlinear flow-rate-dependent friction and pump curves that express the relation between pressure and flow rate. Furthermore, the compressibility of gas introduces additional degrees of freedom, as pipeline inflow and outflow can be controlled independently.

The resulting MINLP problems are solved with various MINLP solution methods — such as spatial branch and bound or outer approximation — or by solving a piecewise-affine approximation of the nonlinear optimization problem, see Appendix A for more details.

2-6 Summary

General batch scheduling techniques

Most scheduling problems — and optimization problems in general — are described and solved with mathematical programming or constraint programming techniques. In the mathematical programming framework, scheduling problems are typically expressed as MILP problems and solved with branch and bound, or with heuristics. Exact methods will return optimal solutions if they exist, but may require long computation times. Heuristics cannot guarantee optimality, but often return good solutions within short computation times. In constraint programming, scheduling problems are expressed with high-level global constraints. Specialized algorithms are available for solving these structures efficiently.

If a mathematical programming problem or a constraint programming problem is too large to be solved monolithically, we can use decomposition methods to split the problem into smaller parts that are easier to solve. Examples of common decomposition methods are hierarchical decomposition, iterative decomposition, and rolling-horizon decomposition.

Depending on the chosen optimization framework and solution techniques, different problem representations exist. Mathematical programming problem representations are either network oriented (time-grid based), or batch oriented (precedence based). For real-world applications, discrete-time representations are more generic and computationally less demanding than other representations. The majority of constraint programming problems is precedence based, since most global constraints are batch oriented.

Current pipeline scheduling methods

Table 2-1 provides an overview of existing refined oil pipeline scheduling methods. Currently, there are no satisfactory optimization methods for refined oil pipeline scheduling. The method by Cafaro and Cerdá (2012) is based on a continuous-time representation that cannot be extended to handle real-world situations. Hierarchical (Boschetto et al., 2010) and iterative (Lopes et al., 2010) decomposition methods have been proposed for finding feasible schedules for very large networks. However, there are no guarantees on solution quality or feasibility.

Table 2-1: Existing refined oil pipeline scheduling methods

	Business question		Optimization problem		
Article	Pipelines	Goal	Framework	Representation	Solution technique(s)
Rejowski and Pinto (2003)	1	Optimality	MILP	Discrete time	Branch and bound
Rejowski and Pinto (2008)	1	Optimality	MINLP	Continuous time	Outer approximation
Cafaro and Cerdá (2012)	6	Optimality	MILP	Continuous time	Branch and bound
Cafaro et al. (2015)	1	Optimality	MINLP	Continuous time	Outer approximation
Boschetto et al. (2010)	30	Feasibility	MILP	Precedence based	Hierarchical decomposition - Construction heuristic - Genetic algorithm - Branch and bound (2x)
Lopes et al. (2010)	30	Feasibility	Constraint programming	Precedence based	Iterative decomposition - Construction heuristic - CP search strategy (2x)

Chapter 3

A novel pipeline scheduling method

3-1 Introduction

In this chapter, we propose a novel pipeline scheduling method. It consists of a planning and a scheduling phase that are coupled in a hierarchical decomposition scheme. In the scheduling phase, we use a rolling-horizon approach to further reduce the computation time, see Fig. 3-1.

Section 3-2 covers high-level choices regarding the problem formulation and the solution techniques. Section 3-3 shows how pipeline networks are represented in the proposed method. Sections 3-4 and 3-5 describe the planning and scheduling phases, respectively.

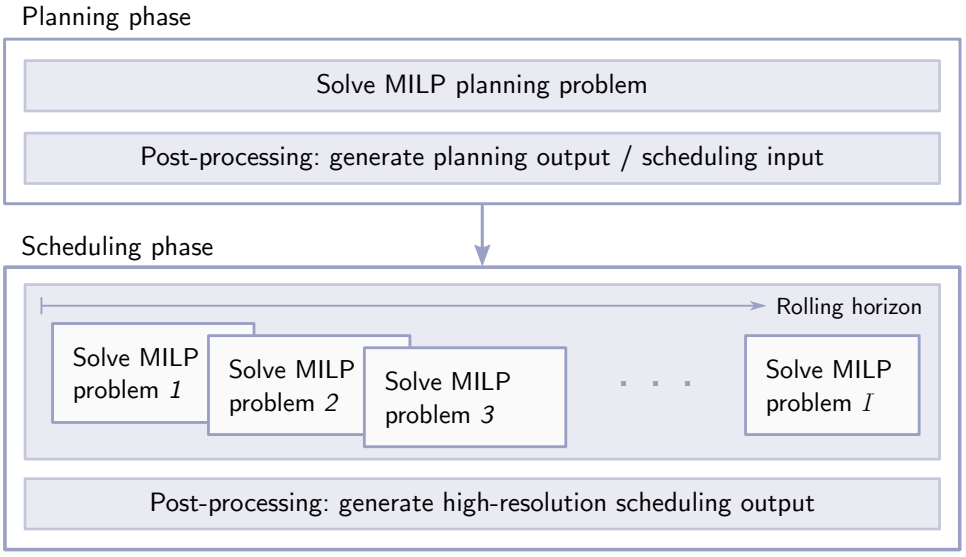


Figure 3-1: Overview of the proposed pipeline scheduling method

3-2 General methodology

In Chapter 2, we investigated general batch scheduling methods and pipeline scheduling methods. We concluded that there are no satisfactory methods to create good schedules for large networks. Therefore, we propose a novel pipeline scheduling method. It is built from scratch, but inspired by existing batch scheduling techniques.

Overall approach: hierarchical decomposition with a rolling horizon
Based on literature on existing pipeline scheduling methods (Section 2-5), we expect that some form of decomposition is required to solve the pipeline scheduling problem within reasonable time. For that reason, we split the total problem into a planning phase and a scheduling phase, and couple them in a hierarchical decomposition scheme. In the planning phase, global day-to-day transportation volumes are determined such that demand and supply requirements are met and tanks are operated within their limits. In the scheduling phase, we use the planning output to generate complete schedules. Both phases contain an optimization problem and a post-processing step. We solve the optimization problem of the planning phase monolithically. The optimization problem of the scheduling phase is too large to be solved within reasonable time. Therefore, we further decompose it using a rolling-horizon decomposition.

Optimization framework: mathematical programming
Referring to Section 2-2, there are two main optimization frameworks for expressing scheduling problems, i.e. mathematical programming and constraint programming. Both frameworks have been successfully applied to many different scheduling problems, see Harjunkoski et al. (2014). For solving mathematical programming problems, and Mixed Integer Linear Programming (MILP) problems in particular, sophisticated commercial solvers are available. These solvers are based on algorithms that make clever use of mathematical properties — e.g. lower bounds obtained from linear relaxations — to find optimal solutions, even when many feasible solutions exist. Constraint programming algorithms, on the other hand, are primarily aimed at finding feasible solutions in highly-constrained solution spaces.

The main challenge of the pipeline scheduling problem is to find a schedule that contains a small number of product transitions and requires a small amount of pumping energy to execute. We expect that many feasible schedules exist, e.g. we could inject a large number of tiny batches in each pipeline. Hence, we expect that the algorithms associated with the mathematical programming framework are most suitable for solving this problem. Therefore, we express the pipeline scheduling problem as a mathematical programming problem.

Problem representation: two discrete-time MILP problems
We formulate the optimization problems of the planning and the scheduling phase as discrete-time MILP problems. The discrete-time representation is preferred, because it offers more flexibility than other representations, see Section 2-4-1. It enables, among others, straightforward incorporation of intermediate supply and demand deadlines, and easier integration of the planning and scheduling phases in the hierarchical decomposition approach. According to Sundaramoorthy and Maravelias (2011), discrete-time problems are also better from a computational point-of-view, because they are solved faster when using branch and bound. The main reason is that discrete-time problems have tighter LP relaxations than, for example, continuous-time representations.

3-3 Representing pipeline networks

In the proposed pipeline scheduling method, networks consist of various locations that are connected by pipelines, see Fig. 3-2. These locations fulfill several roles in a pipeline network. They represent tank farms, they act as supply and demand points, and they couple primary and secondary pipelines. Tank farms are aggregated per product type, i.e. individual tanks within tank farms are not considered.

Primary and secondary pipelines

The difference between primary and secondary pipelines is that primary pipelines start at tank farms, whereas secondary pipelines start at the end of other pipelines. We can use combinations of primary and secondary pipelines to represent pipelines with intermediate exits and to represent pipelines with multiple branches, see Figs. 3-2 and 3-3.

Secondary pipelines are different from an operational perspective, because batch injections in secondary pipelines cannot be freely selected. Instead, they are directly linked to batch ejections of preceding pipelines. During this research, we observed that representing secondary pipelines as primary pipelines introduces additional degrees of freedom in the optimization problem that do not really exist, which significantly slows down branch and bound. Therefore, it is important to treat these pipelines as different entities.

Transmix and pumping cost

Two main cost drivers of pipeline operation are transmix costs and pumping costs. We express these costs with linear and piecewise-affine cost structures, respectively.

Transmix volumes appear at batch interfaces and grow during transportation due to a diffusion-like process, which is known as axial dispersion. Appendix B explains axial dispersion in more detail. Depending on the composition, transmix volumes are sold as a different product or reprocessed in a nearby refinery. As costs are incurred per product transition, we represent transmix costs with a linear cost structure.

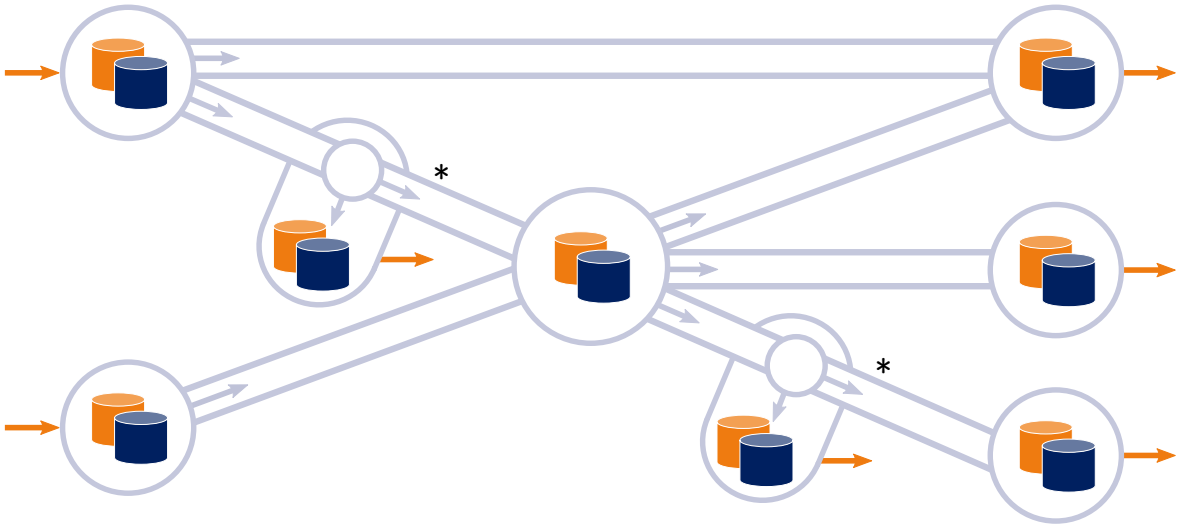


Figure 3-2: Example of a network with six primary and two secondary (*) pipelines

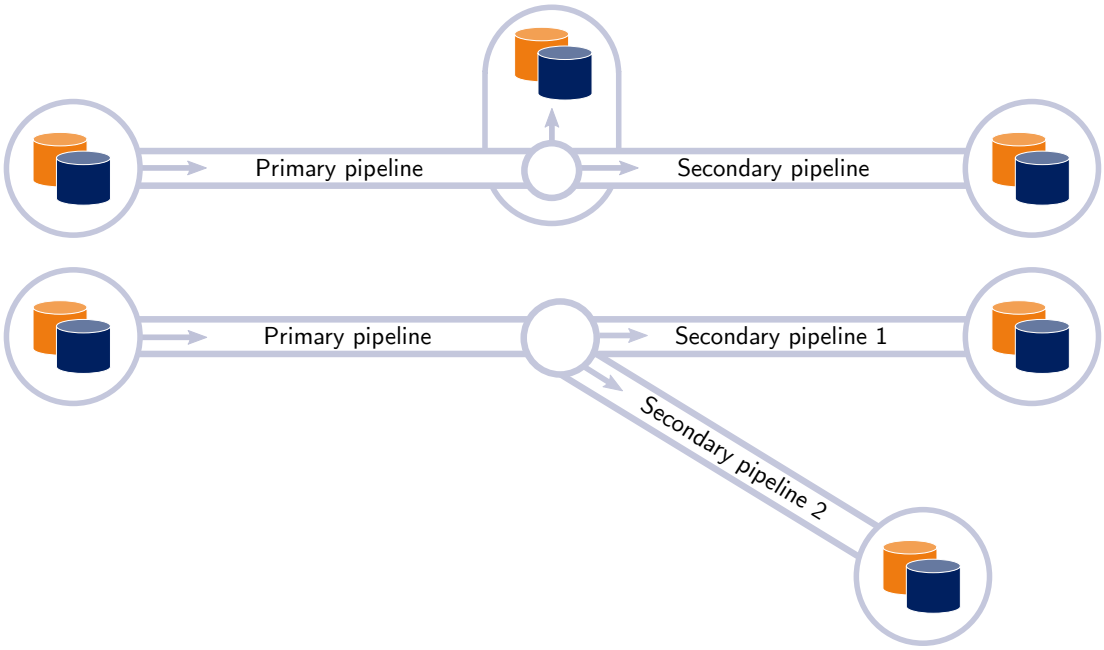


Figure 3-3: Two pipeline configurations with secondary pipelines

The energy required for pumping depends on both the pump efficiency and the network flow characteristics. Pumps should overcome the pressure drops in pipelines that are caused by friction losses. These friction losses increase nonlinearly with the flow rate. We expect that taking the flow-rate dependency of pumping costs into account will have a beneficial effect on the pumping costs of the resulting schedule. However, incorporating the nonlinear dynamics — as proposed by Cafaro et al. (2015) — leads to Mixed Integer Non-Linear Programming (MINLP) problems, which are hard to solve within reasonable time. Therefore, we approximate the pumping costs with piecewise-affine functions. Fig. 3-4 shows an example of a piecewise-affine approximation with an arbitrary number of line segments. Appendix C describes the required computations for obtaining such approximations.

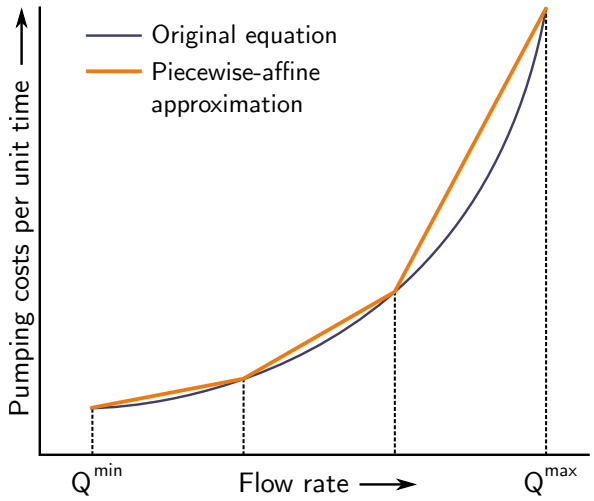


Figure 3-4: Example of a piecewise-affine pumping cost approximation with three line segments

Sets

We describe pipeline networks with the following sets:

N	Locations
P	Products
L	Pipelines
$L_n^{\text{start}} \subseteq L$	Pipelines that start at location n
$L_n^{\text{end}} \subseteq L$	Pipelines that end at location n
$L^{\text{prim}} \subseteq L$	Primary pipelines
$L^{\text{sec}} \subset L$	Secondary pipelines
$L_l^{\text{sec}} \subseteq L^{\text{sec}}$	Secondary pipelines that are connected to primary or secondary pipeline l
M	Line segments used in the piecewise-affine approximation of pumping costs

Parameters

Furthermore, we use the following parameters:

V_l^{pipe}	Total pipeline volume (capacity) of pipeline l
Q_l^{min}	Minimum injection flow rate in pipeline l
Q_l^{max}	Maximum injection flow rate in pipeline l
$R_{n,p}^{\text{min}}$	Minimum stock level of product p at location n
$R_{n,p}^{\text{max}}$	Maximum stock level of product p at location n
V_l^{min}	Minimum batch volume in pipeline l
V_l^{max}	Maximum batch volume in pipeline l

A note on notation

In the remainder of this chapter, we use the following conventions: capital letters denote sets and parameters, whereas lowercase letters denote indices and variables. Indices are always located in subscripts. Superscripts are used to distinguish different parameters and subsets.

The MILP planning and scheduling problems are based on the sets and parameters introduced here. In addition, there are several variables that occur in both MILP problems but have a slightly different meaning. Therefore, we add a ‘p’ or an ‘s’ to the superscripts of all variables to make a clear distinction between the planning and scheduling variables, respectively.

3-4 Planning phase

3-4-1 Introduction

The main goal of the planning phase is to create a list of transportation volumes that are required to meet supply and demand for the entire time horizon. The output contains:

- Planned injection volumes per pipeline, product, and time period
- Planned supply realizations per location, product, and time period
- Planned demand realizations per location, product, and time period

The planning phase is based on a discrete-time MILP problem, which is solved monolithically, and a post-processing step. The latter refines the output of the former. In the remainder of this introduction, we provide a high-level overview of the planning phase.

Planning time buckets

For a specified time horizon, we initialize a set of uniformly distributed planning time buckets, e.g. 30 planning time buckets of 1 day. We neglect dynamics within planning time buckets, such as fluctuations of tank levels. As this might lead to feasibility issues in the scheduling phase, some conservatism is included in the tank capacity constraints.

Aggregated planning batches

For each primary pipeline and each planning time bucket, we initialize exactly one aggregated planning batch. During optimization, one or multiple products are assigned to these planning batches, which is visualized in Fig. 3-5a. In the planning phase, the main decisions are related to choosing the products and volumes that are contained in each aggregated planning batch. In the scheduling phase, the contents of each planning batch are sequenced such that the number of product transitions is minimized, see Fig. 3-5b.

Transportation times

In the planning phase, we assume that batch transportation times are constant. We approximate them using average pipeline flow rates. This results in a simpler problem formulation, in which the ejection timing of each planning batch can be calculated in advance.

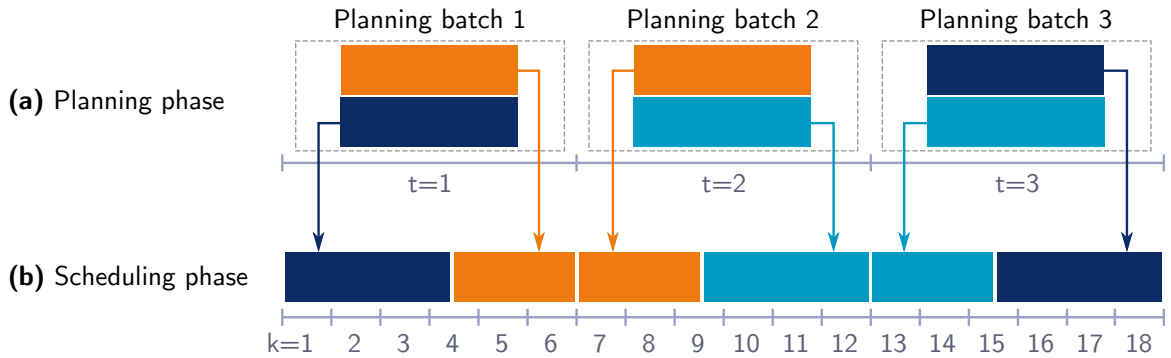


Figure 3-5: Time buckets and batches in the planning and scheduling phase (blocks with the same color represent batches of the same product)

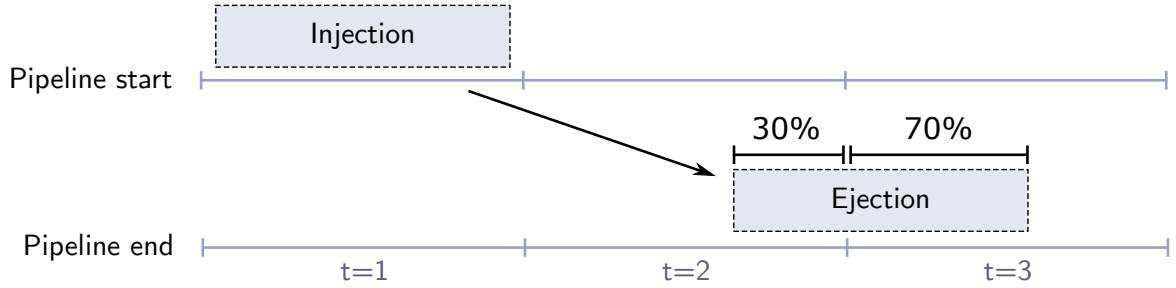


Figure 3-6: Relation between injection timing and ejection timing (transportation time is 1.7 planning time buckets)

We store the ejection timing in the parameter $Y_{l,h,t}$, which expresses the fraction of planning batch h that is ejected from pipeline l in planning time bucket t. The post-processing step corrects the approximation error caused by this simplification.

Batch ejections will typically take place in two planning time buckets, since transportation times are generally not an integer number of planning time buckets. When a planning batch h is injected in pipeline l in planning time bucket $t = 1$ and the average transportation time is 1.7 planning time buckets, then 30% of the batch ejection takes place in the second planning time bucket ($Y_{l,h,t=2} = 0.3$) and 70% takes place in the third planning time bucket ($Y_{l,h,t=3} = 0.7$), see Fig. 3-6.

Supply and demand

Both supply and demand volumes are specified on beforehand and should be fulfilled as much as possible. We do not consider backlog functionality; when supply or demand cannot be realized, it is simply lost.

Objective function

The objective function of the MILP planning problem consists of costs related to pipeline shutdowns, pumping costs, transmix costs, lost supply, and lost demand. The costs of pipeline shutdowns are based on the number of planning time buckets in which a pipeline is inactive and on the total number of pipeline shutdowns. We use piecewise-affine functions to describe the pumping costs. Furthermore, we approximate the number of batch transitions by the total number of products in each aggregated planning batch. Supply and demand that cannot be realized is incorporated with a linear cost structure, defined per unit volume.

Post-processing

Since the MILP planning problem is based on fixed transportation times, we perform a post-processing step to obtain the actual ejection timing of each planning batch. The post-processing step simulates the injection of each planning batch. The actual ejection timing of each planning batch can be calculated by imposing a volume balance on each pipeline.

3-4-2 Sets

In addition to the sets in Section 3-3, the MILP planning problem is based on these sets:

T	Planning time buckets
$T^{\text{in}}_{l,h} \subseteq T$	Injection time domain of planning batch h in pipeline l
$T^{\text{ej}}_{l,h} \subseteq T$	Ejection time domain of planning batch h in pipeline l
H	Planning batches
$H_l \subseteq H$	Planning batches allocated to pipeline l
$H^{\text{LF}}_l \subseteq H_l$	Line-fill batches in pipeline l, i.e. batches that are contained in pipeline l at the start of the planning horizon

3-4-3 Parameters

In addition to the parameters in Section 3-3, the MILP planning problem is based on the parameters in this section. They are divided into general parameters and cost parameters.

General parameters

$\Delta T \in \mathbb{R}^+$	Duration of one planning time bucket
$Y_{l,h,t} \in [0,1]$	Volume fraction of batch $h \in H_l$ ejected from pipeline l during planning time bucket $t \in T^{\text{ej}}_{l,h}$
$V^{\text{LF}}_{l,h,p} \in \mathbb{R}^+$	Volume of product p in line-fill batch $h \in H^{\text{LF}}_l$ in pipeline l
$S_{n,p,t} \in \mathbb{R}^+$	Supply volume of product p during planning time bucket t at location n
$D_{n,p,t} \in \mathbb{R}^+$	Demand volume of product p during planning time bucket t at location n

Cost parameters

$C^{\text{pump,marg}}_{l,m} \in \mathbb{R}$	Marginal pumping costs in pipeline l in piecewise-affine segment m
$C^{\text{pump,stat}}_{l,m} \in \mathbb{R}$	Static pumping costs in pipeline l in piecewise-affine segment m
$C^{\text{trans}}_{l} \in \mathbb{R}^+$	Costs of injecting one extra product in a planning batch in pipeline l
$C^{\text{down}}_{l} \in \mathbb{R}^+$	Costs per time bucket of downtime in pipeline l
$C^{\text{shut}}_{l} \in \mathbb{R}^+$	Costs per shutdown of pipeline l
$C^{\text{sup}}_{n,p} \in \mathbb{R}^+$	Costs of unfulfilled supply of product p at location n
$C^{\text{dem}}_{n,p} \in \mathbb{R}^+$	Costs of unfulfilled demand of product p at location n

3-4-4 Variables

This section describes all planning-related variables and their required initial conditions.

Location variables

$r^{\mathrm{p}}_{n,p,t} \in \mathbb{R}^+$	Stock level of product p at location n at the start of planning time bucket t
$s^{\mathrm{p}}_{n,p,t} \in \mathbb{R}^+$	Realized supply of product p at location n during planning time bucket t
$d^{\mathrm{p}}_{n,p,t} \in \mathbb{R}^+$	Realized demand of product p at location n during planning time bucket t
$s^{\mathrm{p,u}}_{n,p,t} \in \mathbb{R}^+$	Unfulfilled supply of product p at location n during planning time bucket t
$d^{\mathrm{p,u}}_{n,p,t} \in \mathbb{R}^+$	Unfulfilled demand of product p at location n during planning time bucket t

Pipeline variables

$z^{\mathrm{p}}_{l,t} \in \{0,1\}$	Indicates whether pipeline l is active during planning time bucket t (1) or not (0)
$\psi^{\mathrm{p}}_{l,t} \in \mathbb{R}^+$	Pumping costs related to pipeline l during planning time bucket t
$\sigma^{\mathrm{p}}_{l,t} \in [0,1]$	Indicates[1] whether pipeline $l \in L^{\mathrm{prim}}$ is shut down at the beginning of planning time bucket t (1) or not (0)

Batch variables

$v^{\mathrm{p,in}}_{l,h,p,t} \in \mathbb{R}^+$	Volume injection of product p in batch $h \in H_l$ in pipeline l during planning time bucket $t \in T^{\mathrm{in}}_{l,h}$
$v^{\mathrm{p,ej}}_{l,h,p,t} \in \mathbb{R}^+$	Volume ejection of product p from batch $h \in H_l$ from pipeline l during planning time bucket $t \in T^{\mathrm{ej}}_{l,h}$
$\theta^{\mathrm{p}}_{l,h,p} \in \{0,1\}$	Indicates whether batch $h \in H_l$ in pipeline l contains product p (1) or not (0)

Initial condition

$r^{\mathrm{p}}_{n,p,t=1}$	Initial stock level of product p at location n

3-4-5 Constraints

This section describes all planning-related constraints.

Stock balance

Stock levels increase due to pipeline ejections and realized supply, and decrease due to pipeline injections and realized demand:

$$r^{\mathrm{p}}_{n,p,t} = r^{\mathrm{p}}_{n,p,t-1} + \sum_{l \in L^{\mathrm{end}}_n} \sum_{h \in H_l} v^{\mathrm{p,ej}}_{l,h,p,t-1} - \sum_{l \in L^{\mathrm{start}}_n} \sum_{h \in H_l} v^{\mathrm{p,in}}_{l,h,p,t-1} + s^{\mathrm{p}}_{n,p,t-1} - d^{\mathrm{p}}_{n,p,t-1} \quad \forall n \in N, \forall p \in P, \forall t \in T, t > 1 \tag{3-1}$$

[1] Although 0 and 1 are the only meaningful values in this context, this variable can be expressed as a continuous 0–1 variable in the MILP formulation, since its value is forced to 0 or 1 by the involved constraints.

Conservative stock bounds

To prevent feasibility problems when generating schedules, we use conservative stock bounds: at the beginning of each planning time bucket, there should be enough stock to complete *all* stock-decreasing operations and there should be enough idle capacity to complete *all* stock-increasing operations. The following two constraints describe this:

$$R^{\min}_{n,p} \leq r^{\mathrm{p}}_{n,p,t-1} - \sum_{l \in L^{\mathrm{start}}_n} \sum_{h \in H_l} v^{\mathrm{p,in}}_{l,h,p,t-1} - d^{\mathrm{p}}_{n,p,t-1} \qquad \forall n \in N, \forall p \in P, \forall t \in T, t > 1 \quad (3\text{-}2)$$

$$r^{\mathrm{p}}_{n,p,t-1} + \sum_{l \in L^{\mathrm{end}}_n} \sum_{h \in H_l} v^{\mathrm{p,ej}}_{l,h,p,t-1} + s^{\mathrm{p}}_{n,p,t-1} \leq R^{\max}_{n,p} \quad \forall n \in N, \forall p \in P, \forall t \in T, t > 1 \quad (3\text{-}3)$$

Supply and demand

Supply and demand volumes are specified on beforehand and should be fulfilled as much as possible. We add violation variables to make the optimization problem robust against infeasible inputs:

$$s^{\mathrm{p}}_{n,p,t} + s^{\mathrm{p,u}}_{n,p,t} = S_{n,p,t} \qquad \forall n \in N, \forall p \in P, \forall t \in T \quad (3\text{-}4)$$

$$d^{\mathrm{p}}_{n,p,t} + d^{\mathrm{p,u}}_{n,p,t} = D_{n,p,t} \qquad \forall n \in N, \forall p \in P, \forall t \in T \quad (3\text{-}5)$$

Batch contents

Product volumes in aggregated batches are semi-continuous. Only if a product is included, the corresponding volume should be between a lower and an upper bound:

$$V^{\min}_l \theta^{\mathrm{p}}_{l,h,p} \leq \sum_{t \in T^{\mathrm{in}}_{l,h}} v^{\mathrm{p,in}}_{l,h,p,t} \qquad \forall l \in L, \forall h \in H_l, \forall p \in P \quad (3\text{-}6)$$

$$\sum_{t \in T^{\mathrm{in}}_{l,h}} v^{\mathrm{p,in}}_{l,h,p,t} \leq V^{\max}_l \theta^{\mathrm{p}}_{l,h,p} \qquad \forall l \in L, \forall h \in H_l, \forall p \in P \quad (3\text{-}7)$$

Batch ejection timing

Batch transportation times are fixed. The parameter $Y_{l,h,t}$, which indicates the volume fraction of batch h that is ejected in planning time bucket t, is calculated in advance based on average transportation times. In the constraints, we distinguish new batches and line-fill batches, i.e. batches that are contained in the pipeline at the start of the horizon:

$$v^{\mathrm{p,ej}}_{l,h,p,t} = Y_{l,h,t} \sum_{t' \leq t} v^{\mathrm{p,in}}_{l,h,p,t'} \qquad \forall l \in L, \forall h \in H_l \setminus H^{\mathrm{LF}}_l, \forall p \in P, \forall t \in T^{\mathrm{ej}}_{l,h} \quad (3\text{-}8)$$

$$v^{\mathrm{p,ej}}_{l,h,p,t} = Y_{l,h,t} \cdot V^{\mathrm{LF}}_{l,h,p} \qquad \forall l \in L, \forall h \in H^{\mathrm{LF}}_l, \forall p \in P, \forall t \in T \quad (3\text{-}9)$$

Pipeline flow bounds

Pipeline flow is semi-continuous, i.e. it is either zero or between a lower and an upper bound. We impose bounds on the sum of batch injections:

$$(Q^{\min}_l \cdot \Delta T) z^{\mathrm{p}}_{l,t} \leq \sum_{h \in H_l} \sum_{p \in P} v^{\mathrm{p,in}}_{l,h,p,t} \qquad \forall l \in L^{\mathrm{prim}}, \forall t \in T \quad (3\text{-}10)$$

$$\sum_{h \in H_l} \sum_{p \in P} v^{\mathrm{p,in}}_{l,h,p,t} \leq (Q^{\max}_l \cdot \Delta T) z^{\mathrm{p}}_{l,t} \qquad \forall l \in L^{\mathrm{prim}}, \forall t \in T \quad (3\text{-}11)$$

For secondary pipelines, the flow bounds are different. If the preceding pipeline operates at low speed, then the part of the ejection redirected to the secondary pipeline should be large enough to meet the minimum flow-rate requirements of the secondary pipeline. Likewise, if the preceding pipeline operates at high speed and the maximum flow rate of the secondary pipeline is lower than that, then the ejection of the preceding pipeline cannot be completely redirected to the secondary pipeline. Since the MILP planning problem is based on aggregated batches and fixed transportation times, we cannot use the constraints in Eqs. (3-10) and (3-11) to guarantee these conditions. Instead, we introduce two constraints on product level:

$$\frac{Q_{l'}^{\min}}{Q_{l}^{\min}} \sum_{t \in T_{l,h}^{\mathrm{ej}}} v_{l,h,p,t}^{\mathrm{p,ej}} \leq \sum_{t \in T_{l',h}^{\mathrm{in}}} v_{l',h,p,t}^{\mathrm{p,in}} \qquad \forall l \in L, \forall l' \in L_{l}^{\mathrm{sec}}, \forall h \in H_l \cap H_{l'}, \forall p \in P \tag{3-12}$$

$$\sum_{t \in T_{l',h}^{\mathrm{in}}} v_{l',h,p,t}^{\mathrm{p,in}} \leq \frac{Q_{l'}^{\max}}{Q_{l}^{\max}} \sum_{t \in T_{l,h}^{\mathrm{ej}}} v_{l,h,p,t}^{\mathrm{p,ej}} \qquad \forall l \in L, \forall l' \in L_{l}^{\mathrm{sec}}, \forall h \in H_l \cap H_{l'}, \forall p \in P \tag{3-13}$$

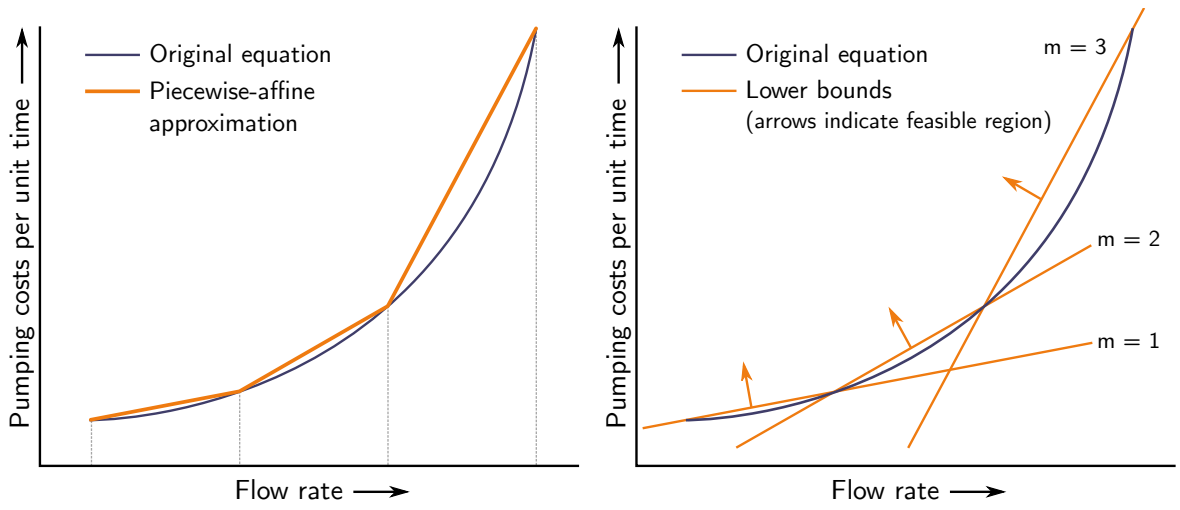
The constraints in Eqs. (3-12) and (3-13) ensure that the injection in secondary pipeline l' is always within the range of allowed flow rates $[Q_{l'}^{\min}, Q_{l'}^{\max}]$, regardless of the flow rate in primary pipeline l.

Pumping costs

We approximate the pumping costs with piecewise-affine functions, as illustrated in Fig. 3-7a and further explained in Appendix C. Since the piecewise-affine functions are convex, we can include them in the problem formulation without any additional binary variables. The line segments, see Fig. 3-7b, are lower bounds on the pumping costs $\psi_{l,t}^{\mathrm{p}}$:

$$C_{l,m}^{\mathrm{pump,marg}} \sum_{p \in P} \sum_{h \in H_l} v_{l,h,p,t}^{\mathrm{p,in}} + C_{l,m}^{\mathrm{pump,stat}} \cdot \Delta T \leq \psi_{l,t}^{\mathrm{p}} \qquad \forall l \in L, \forall m \in M, \forall t \in T \tag{3-14}$$

As $\psi_{l,t}^{\mathrm{p}}$ is minimized, its value will always coincide with the piecewise-affine function. Depending on the flow rate, the constraint in Eq. (3-14) is binding for at least one line segment.



(a) Piecewise-affine approximation **(b)** Line segments as lower bounds

Figure 3-7: Expressing piecewise-affine approximations with line segments as lower bounds

Number of shutdowns

If pipeline l is active in the previous planning time bucket $(t-1)$ but not in the current planning time bucket t, the following constraint forces $\sigma^{\mathrm{p}}_{l,t}$ to 1:

$$z^{\mathrm{p}}_{l,t-1} - z^{\mathrm{p}}_{l,t} \leq \sigma^{\mathrm{p}}_{l,t} \qquad \forall l \in L^{\mathrm{prim}}, \forall t \in T, t > 1 \tag{3-15}$$

$$1 - z^{\mathrm{p}}_{l,t=1} \leq \sigma^{\mathrm{p}}_{l,t=1} \qquad \forall l \in L^{\mathrm{prim}} \tag{3-16}$$

3-4-6 Objective function

The objective function of the MILP planning problem consists of six terms:

$$\begin{aligned} &\underbrace{\sum_{l \in L^{\mathrm{prim}}} \sum_{h \in H} \sum_{p \in P} C^{\mathrm{trans}}_{l} \theta^{\mathrm{p}}_{l,h,p}}_{\text{transmix}} + \underbrace{\sum_{l \in L} \sum_{t \in T} \psi^{\mathrm{p}}_{l,t}}_{\text{pumping}} + \underbrace{\sum_{l \in L^{\mathrm{prim}}} \sum_{t \in T} C^{\mathrm{down}}_{l} (1 - z^{\mathrm{p}}_{l,t})}_{\text{pipeline downtime}} \\ &+ \underbrace{\sum_{l \in L^{\mathrm{prim}}} \sum_{t \in T} C^{\mathrm{shut}}_{l} \sigma^{\mathrm{p}}_{l,t}}_{\text{pipeline shutdowns}} + \underbrace{\sum_{n \in N} \sum_{p \in P} \sum_{t \in T} C^{\mathrm{sup}}_{n,p} s^{\mathrm{p,u}}_{n,p,t}}_{\text{unfulfilled supply}} + \underbrace{\sum_{n \in N} \sum_{p \in P} \sum_{t \in T} C^{\mathrm{dem}}_{n,p} d^{\mathrm{p,u}}_{n,p,t}}_{\text{unfulfilled demand}} \end{aligned} \tag{3-17}$$

3-5 Scheduling phase

3-5-1 Introduction

In the scheduling phase, we use the output of the planning phase to generate complete schedules. Similar to the planning phase, the scheduling phase consists of a discrete-time MILP problem and a post-processing step that refines the MILP solution.

We decompose the MILP scheduling problem using a rolling-horizon approach. In every iteration, the MILP problem is solved for a limited time horizon. Subsequently, the MILP solution for the first scheduling time buckets is fixed. In addition, the MILP solution serves as an initial condition for the next MILP problem, in which the horizon is shifted. We repeat these steps until the entire scheduling horizon is solved.

The remainder of this introduction provides a high-level overview of the scheduling phase.

Scheduling time buckets

We initialize a set of uniformly distributed scheduling time buckets for the scheduling horizon. We assume that an integer number of scheduling time buckets fits in one planning time bucket. For example, when each planning time bucket has a duration of 24 hours, the scheduling time buckets can be 4 or 6 hours, but not 5 hours.

Scheduling batches

We initialize scheduling batches for each primary pipeline, for each subset of scheduling time buckets that corresponds to one planning time bucket, and for each product type. In contrast to planning batches, scheduling batches contain a single product.

Fig. 3-5 in Section 3-4-1 illustrates the relation between planning batches and scheduling batches. In essence, each planning batch contains one or more scheduling batches that are not yet sequenced. The scheduling batches that correspond to a particular planning batch, can only be injected in the scheduling time buckets that correspond to that particular planning batch. Referring to the example in Fig. 3-5, scheduling batches that originate from planning batch 1 can only be injected in scheduling time buckets $k = 1$ up to $k = 6$, since these scheduling time buckets correspond to planning time bucket $t = 1$.

Post-processing

To overcome discretization errors that occur in most discrete-time representations, we allow batches to start and end *within* scheduling time buckets. Referring to Fig. 3-8, if the injection of batch 2 starts in the third scheduling time bucket, then volumes of both batch 1 and batch 2 can be injected in that time bucket. The post-processing step computes the actual switching time using volume-based interpolation. For example, if 40% of the total injection volume in the third scheduling time bucket corresponds to batch 2, then the post-processed starting time of batch 2 is at 60% of the third scheduling time bucket.

Transportation times

In contrast to the planning problem, the scheduling problem does include dynamic transportation times. Inspired by Cafaro and Cerdá (2004), we use a batch front variable to track the positions of batches in a pipeline over time. The batch front variable represents the volumetric front coordinate of a batch with respect to the start of the pipeline. When a batch reaches the end of the pipeline, its ejection starts.

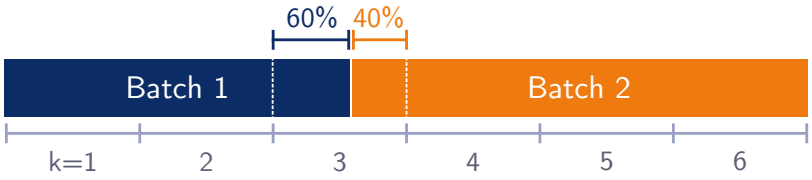


Figure 3-8: Example of a batch that starts within a scheduling time bucket

Objective function

As the pipeline activation and the fulfillment of supply and demand are determined in the planning phase, the objective function of the MILP scheduling problem only contains two terms, i.e. transmix costs and pumping costs. Similar to the MILP planning problem, we approximate the pumping costs with piecewise-affine functions. The transmix costs are based on the number of product transitions in the batch injection sequence.

3-5-2 Sets

In addition to the sets in Section 3-3, the MILP scheduling problem is based on these sets:

K	Scheduling time buckets
$K^{\text{ip}}_{l,b} \subseteq K$	In-pipe time domain of scheduling batch b in pipeline l
$K^{\text{in}}_{l,b} \subseteq K^{\text{ip}}_{l,b}$	Injection time domain of scheduling batch b in pipeline l
$K^{\text{ej}}_{l,b} \subseteq K^{\text{ip}}_{l,b}$	Ejection time domain of scheduling batch b in pipeline l
B	Scheduling batches
$B_p \subseteq B$	Scheduling batches of product p
$B_l \subseteq B$	Scheduling batches allocated to pipeline l
$B_h \subseteq B$	Scheduling batches that correspond to aggregated planning batch h
$B^{\text{LF}}_l \subseteq B_l$	Line-fill batches in pipeline l, i.e. batches that are contained in pipeline l at the start of the scheduling horizon

3-5-3 Parameters

In addition to the parameters in Section 3-3, the MILP scheduling problem is based on the parameters mentioned below. They are divided into general, planning, and cost parameters.

General parameters

$\Delta K \in \mathbb{R}^+$	Duration of one scheduling time bucket
$V^{\text{LF}}_{l,b,p} \in \mathbb{R}^+$	Volume of product p in line-fill batch $b \in B^{\text{LF}}_l$ in pipeline l

Planning parameters

$V^{\text{plan}}_{l,h,p} \in \mathbb{R}^+$	Planned volume of product p in planning batch $h \in B_l$ in pipeline l
$Z^{\text{plan}}_{l,k} \in \{0,1\}$	Planned pipeline activation during scheduling time bucket k in pipeline l
$S^{\text{plan}}_{n,p,k} \in \mathbb{R}^+$	Planned supply of product p during scheduling time bucket k at location n
$D^{\text{plan}}_{n,p,k} \in \mathbb{R}^+$	Planned demand of product p during scheduling time bucket k at location n

Cost parameters

$C_{l,m}^{\text{pump,marg}} \in \mathbb{R}$	Marginal pumping energy costs in pipeline l in piecewise-affine segment m
$C_{l,m}^{\text{pump,stat}} \in \mathbb{R}$	Static pumping energy costs in pipeline l in piecewise-affine segment m
$C_{l}^{\text{trans}} \in \mathbb{R}^{+}$	Costs per product transition in pipeline l
$C_{l}^{\text{down}} \in \mathbb{R}^{+}$	Costs per time bucket of downtime in pipeline l
$C_{l}^{\text{shut}} \in \mathbb{R}^{+}$	Costs per shutdown of pipeline l

3-5-4 Variables

This section describes all scheduling-related variables and their required initial conditions.

Location variables

$r_{n,p,k}^{\text{s}} \in \mathbb{R}^{+}$	Inventory level of product p at location n at the start of scheduling time bucket k

Pipeline variables

$\sigma_{l,k}^{\text{s}} \in [0,1]$	Indicates[2] whether pipeline l is shut down at the beginning of scheduling time bucket k (1) or not (0)
$\theta_{l,k}^{\text{s}} \in [0,1]$	Indicates[2] whether a new batch injection is started in pipeline $l \in L^{\text{prim}}$ in scheduling time bucket k which contains a different product than the previous batch (1) or not (0)
$\psi_{l,k}^{\text{s}} \in \mathbb{R}^{+}$	Pumping costs in pipeline l during scheduling time bucket k

Batch timing variables

$x_{l,b,k}^{\text{s,start}} \in \{0,1\}$	Indicates whether the injection of batch $b \in B_l$ starts in pipeline l in scheduling time bucket $k \in K_{l,b}^{\text{in}}$ (1) or not (0)
$x_{l,b,k}^{\text{s,cont}} \in [0,1]$	Indicates[2] whether the injection of batch $b \in B_l$ continues in pipeline l in scheduling time bucket $k \in K_{l,b}^{\text{in}}$ (1) or not (0)
$y_{l,b,k}^{\text{s,start}} \in \{0,1\}$	Indicates whether the ejection of batch $b \in B_l$ starts in pipeline l in scheduling time bucket $k \in K_{l,b}^{\text{ej}}$ (1) or not (0)
$y_{l,b,k}^{\text{s,cont}} \in [0,1]$	Indicates[2] whether the ejection of batch $b \in B_l$ continues in pipeline l in scheduling time bucket $k \in K_{l,b}^{\text{ej}}$ (1) or not (0)
$w_{l,b,k}^{\text{s,transit}} \in [0,1]$	Indicates[2] whether batch $b \in B_l$ is in transit at the start of scheduling time bucket $k \in K_{l,b}^{\text{ip}}$ (1) or not (0)
$w_{l,b,k}^{\text{s,pipe}} \in [0,1]$	Indicates[2] whether batch $b \in B_l$ is in pipeline l at the start of scheduling time bucket $k \in K_{l,b}^{\text{ip}}$ (1) or not (0)

[2]Although 0 and 1 are the only meaningful values in this context, this variable can be expressed as a continuous 0–1 variable in the MILP formulation, since its value is forced to 0 or 1 by the involved constraints.

Batch volume variables

$v^{\mathrm{s,in}}_{l,b,k} \in \mathbb{R}^+$	Injection volume of batch $b \in B_l$ in pipeline l during scheduling time bucket $k \in K^{\mathrm{in}}_{l,b}$
$v^{\mathrm{s,ej}}_{l,b,k} \in \mathbb{R}^+$	Ejection volume of batch $b \in B_l$ from pipeline l during scheduling time bucket $k \in K^{\mathrm{ej}}_{l,b}$
$v^{\mathrm{s}}_{l,b,k} \in \mathbb{R}^+$	Volume of batch $b \in B_l$ in pipeline l at the start of scheduling time bucket $k \in K^{\mathrm{ip}}_{l,b}$
$f^{\mathrm{s}}_{l,b,k} \in \mathbb{R}^+$	Volumetric front coordinate of batch $b \in B_l$ in pipeline l at the start of scheduling time bucket $k \in K^{\mathrm{ip}}_{l,b}$

Initial conditions

$r^{\mathrm{s}}_{n,p,k=1}$	Initial stock level of product p at location n
$x^{\mathrm{s,cont}}_{l,b,k=1}$	Indicates whether the injection of batch $b \in B_l$ continues in pipeline l in the first scheduling time bucket (1) or not (0). This variable equals 1 for the latest line-fill batch in each pipeline, i.e. the batch that is closest to the pipeline start.
$y^{\mathrm{s,cont}}_{l,b,k=1}$	Indicates whether the ejection of batch $b \in B_l$ continues in pipeline l in the first scheduling time bucket (1) or not (0). This variable equals 1 for the earliest line-fill batch in each pipeline, i.e. the batch that is closest to the pipeline end.
$w^{\mathrm{s,transit}}_{l,b,k=1}$	Indicates whether batch $b \in B_l$ is in transit in pipeline l in the first scheduling time bucket (1) or not (0)
$v^{\mathrm{s}}_{l,b,k=1}$	Batch volume of line-fill batches $b \in B^{\mathrm{LF}}_l$ in pipeline l at the start of the first scheduling time bucket
$f^{\mathrm{s}}_{l,b,k=1}$	Batch front of line-fill batches $b \in B^{\mathrm{LF}}_l$ in pipeline l at the start of the first scheduling time bucket

Explanation of the batch volume and batch timing variables

Fig. 3-9 illustrates the desired behavior of the batch volume variables and the batch timing variables of one batch — batch 3 — in relation to other batches. The dashed vertical lines mark important events, i.e. injection start, injection end, ejection start, and ejection end.

Fig. 3-9a shows the scheduling time bucket in which the injection of batch 3 starts. The injection of batch 3 continues (Fig. 3-9b) until the injection of batch 4 starts. The same behavior holds for the ejection timing. First, the ejection of batch 3 starts, see Fig. 3-9c. The ejection of batch 3 continues (Fig. 3-9d) until the ejection of batch 4 starts.

The batch front is used to relate the injection and ejection timing. Fig. 3-9e plots the batch fronts of multiple batches. The area between these curves illustrates the pipeline content over time. When the injection of batch 3 starts, its batch front starts increasing. The batch front of batch 3 keeps increasing until the other end of the pipeline is reached. At that point, the ejection of batch 3 starts. The batch front remains constant until batch 3 is fully ejected.

Fig. 3-9f shows the batch volume over time. The batch volume increases between injection start and injection end, and decreases between ejection start and ejection end.

Furthermore, Fig. 3-9g indicates whether or not batch 3 is in transit at the start of a particular scheduling time bucket, i.e. injection has started but ejection has not yet started. Fig. 3-9h indicates whether or not batch 3 is in the pipeline at the start of a particular scheduling time bucket, i.e. injection has started but ejection has not yet ended.

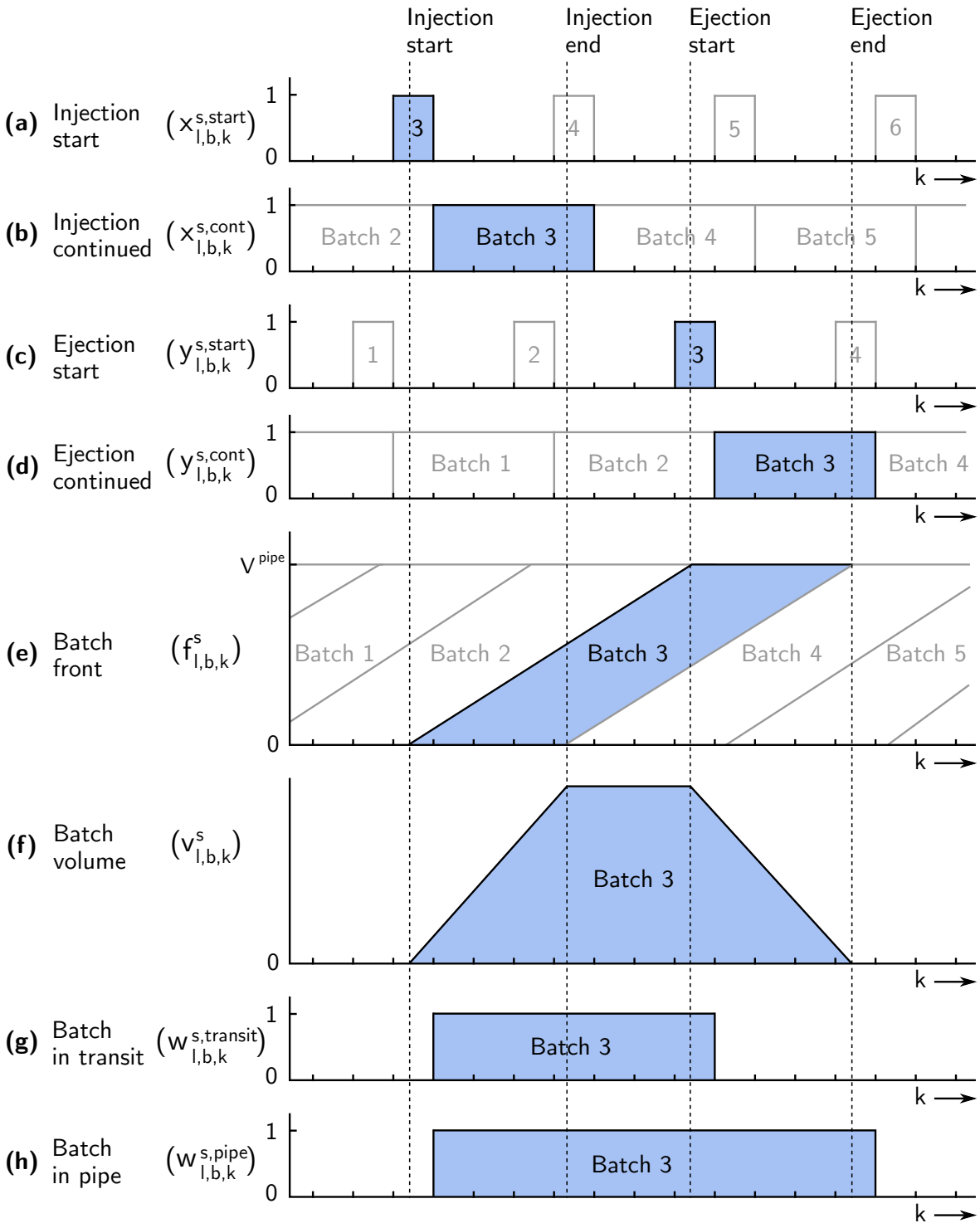


Figure 3-9: Illustration of the batch volume and batch timing variables

3-5-5 Constraints

This section describes all scheduling-related constraints.

Stock balance

Stock levels increase due to pipeline ejections and planned supply, and decrease due to pipeline injections and planned demand. The following constraint, which is similar to Eq. (3-1) in the MILP planning problem, describes this:

$$r^{\mathrm{s}}_{n,p,k} = r^{\mathrm{s}}_{n,p,k-1} + \sum_{l \in L^{\mathrm{end}}_n} \sum_{b \in B_l \cap B_p} v^{\mathrm{s,ej}}_{l,b,k-1} - \sum_{l \in L^{\mathrm{start}}_n} \sum_{b \in B_l \cap B_p} v^{\mathrm{s,in}}_{l,b,k-1} + S^{\mathrm{plan}}_{n,p,k-1} - D^{\mathrm{plan}}_{n,p,k} \quad \forall n \in N, \forall p \in P, \forall k \in K, k > 1 \tag{3-18}$$

Conservative stock bounds

To prevent feasibility problems during pipeline operation, we use conservative stock bounds that are equivalent to Eqs. (3-2) and (3-3) in the MILP planning problem:

$$R^{\min}_{n,p} \leq r^{\mathrm{s}}_{n,p,k-1} - \sum_{l \in L^{\mathrm{start}}_n} \sum_{b \in B_l \cap B_p} v^{\mathrm{s,in}}_{l,b,k-1} - D^{\mathrm{plan}}_{n,p,k-1} \quad \forall n \in N, \forall p \in P, \forall k \in K, k > 1 \tag{3-19}$$

$$r^{\mathrm{s}}_{n,p,k-1} + \sum_{l \in L^{\mathrm{end}}_n} \sum_{b \in B_l \cap B_p} v^{\mathrm{s,ej}}_{l,b,k-1} + S^{\mathrm{plan}}_{n,p,k-1} \leq R^{\max}_{n,p} \quad \forall n \in N, \forall p \in P, \forall k \in K, k > 1 \tag{3-20}$$

Overall pipeline flow

The pipeline must be completely filled, thus the injection volume equals the ejection volume:

$$\sum_{b \in B_l} v^{\mathrm{s,in}}_{l,b,k} = \sum_{b \in B_l} v^{\mathrm{s,ej}}_{l,b,k} \qquad \forall l \in L, \forall k \in K \tag{3-21}$$

Pipeline flow is semi-continuous, i.e. it is either zero, or between a lower and an upper bound:

$$(Q^{\min}_l \cdot \Delta K) Z^{\mathrm{plan}}_{l,k} \leq \sum_{b \in B_l} v^{\mathrm{s,in}}_{l,b,k} \qquad \forall l \in L, \forall k \in K \tag{3-22}$$

$$\sum_{b \in B_l} v^{\mathrm{s,in}}_{l,b,k} \leq (Q^{\max}_l \cdot \Delta K) Z^{\mathrm{plan}}_{l,k} \qquad \forall l \in L, \forall k \in K \tag{3-23}$$

Injections in secondary pipelines cannot be larger than ejections from preceding pipelines:

$$\sum_{l' \in L^{\mathrm{sec}}_l} v^{\mathrm{s,in}}_{l',b,k} \leq v^{\mathrm{s,ej}}_{l,b,k} \qquad \forall l \in L, \forall b \in B_l, \forall k \in K^{\mathrm{ej}}_{l,b} \tag{3-24}$$

Batch injection timing

The variables $x^{\mathrm{s,start}}_{l,b,k}$ and $x^{\mathrm{s,cont}}_{l,b,k}$ express the injection timing, see Section 3-5-4. In secondary pipelines, the injection timing is directly linked to the ejection timing of the previous pipeline:

$$x^{\mathrm{s,start}}_{l',b,k} = y^{\mathrm{s,start}}_{l,b,k} \qquad \forall l \in L, \forall l' \in L^{\mathrm{sec}}_l, \forall b \in B_l \cap B_{l'}, \forall k \in K^{\mathrm{ej}}_{l,b} \cap K^{\mathrm{in}}_{l',b} \tag{3-25}$$

$$x^{\mathrm{s,cont}}_{l',b,k} = y^{\mathrm{s,cont}}_{l,b,k} \qquad \forall l \in L, \forall l' \in L^{\mathrm{sec}}_l, \forall b \in B_l \cap B_{l'}, \forall k \in K^{\mathrm{ej}}_{l,b} \cap K^{\mathrm{in}}_{l',b} \tag{3-26}$$

Eqs. (3-27) to (3-31) include all timing-related constraints for primary pipelines:

$$\sum_{k \in K^{\text{in}}_{l,b}} x^{\text{s,start}}_{l,b,k} \leq 1 \qquad \forall l \in L^{\text{prim}}, \forall b \in B_l \tag{3-27}$$

$$\sum_{b \in B_l} x^{\text{s,start}}_{l,b,k} \leq 1 \qquad \forall l \in L^{\text{prim}}, \forall k \in K \tag{3-28}$$

$$\sum_{b \in B_l} x^{\text{s,cont}}_{l,b,k} = 1 \qquad \forall l \in L^{\text{prim}}, \forall k \in K \tag{3-29}$$

$$x^{\text{s,cont}}_{l,b,k} \leq x^{\text{s,cont}}_{l,b,k-1} + x^{\text{s,start}}_{l,b,k-1} \qquad \forall l \in L^{\text{prim}}, \forall b \in B_l, \forall k \in K^{\text{in}}_{l,b}, k > 1 \tag{3-30}$$

$$x^{\text{s,start}}_{l,b,k-1} \leq x^{\text{s,cont}}_{l,b,k} \qquad \forall l \in L^{\text{prim}}, \forall b \in B_l, \forall k \in K^{\text{in}}_{l,b}, k > 1 \tag{3-31}$$

Eqs. (3-27) and (3-28) state that each batch injection starts at most once, and that at most one batch injection starts in each scheduling time bucket. Exactly one batch injection continues in each scheduling time bucket (Eq. (3-29)). The constraint in Eq. (3-30) ensures that a batch only continues when it has started. Eq. (3-31) states that a batch injection continues for at least one scheduling time bucket after injection start. The combination of Eqs. (3-29) and (3-31) ensures that batch injections are properly ended.

Subsequently, the injection volumes are linked to the injection timing:

$$v^{\text{s,in}}_{l,b,k} \leq (Q^{\max}_l \cdot \Delta K)(x^{\text{s,start}}_{l,b,k} + x^{\text{s,cont}}_{l,b,k}) \qquad \forall l \in L, \forall b \in B_l, \forall k \in K^{\text{in}}_{l,b} \tag{3-32}$$

If an injection starts, the total injection volume should be at least the minimum batch size:

$$V^{\min}_l \sum_{k \in K^{\text{in}}_{l,b}} x^{\text{s,start}}_{l,b,k} \leq \sum_{k \in K^{\text{in}}_{l,b}} v^{\text{s,in}}_{l,b,k} \qquad \forall l \in L, \forall b \in B_l \tag{3-33}$$

Batch ejection timing

We use a similar set of constraints for the batch ejection timing:

$$\sum_{k \in T} y^{\text{s,start}}_{l,b,k} \leq 1 \qquad \forall l \in L, \forall b \in B_l \tag{3-34}$$

$$\sum_{b \in B_l} y^{\text{s,start}}_{l,b,k} \leq 1 \qquad \forall l \in L, \forall k \in K \tag{3-35}$$

$$\sum_{b \in B_l} y^{\text{s,cont}}_{l,b,k} = 1 \qquad \forall l \in L, \forall k \in K \tag{3-36}$$

$$y^{\text{s,cont}}_{l,b,k} \leq y^{\text{s,cont}}_{l,b,k-1} + y^{\text{s,start}}_{l,b,k-1} \qquad \forall l \in L, \forall b \in B_l, \forall k \in K, k > 1 \tag{3-37}$$

$$y^{\text{s,start}}_{l,b,k-1} \leq y^{\text{s,cont}}_{l,b,k} \qquad \forall l \in L, \forall b \in B_l, \forall k \in K^{\text{ej}}_{l,b}, k > 1 \tag{3-38}$$

$$v^{\text{s,ej}}_{l,b,k} \leq (Q^{\max}_l \cdot \Delta K)(y^{\text{s,start}}_{l,b,k} + y^{\text{s,cont}}_{l,b,k}) \qquad \forall l \in L, \forall b \in B_l, \forall k \in K^{\text{ej}}_{l,b} \tag{3-39}$$

Furthermore, batch ejections can only take place after injection:

$$y^{\text{s,start}}_{l,b,k} \leq \sum_{k' \leq k} x^{\text{s,start}}_{l,b,k'} \qquad \forall l \in L, \forall b \in B_l \setminus B^{\text{LF}}_l, \forall k \in K^{\text{ej}}_{l,b} \tag{3-40}$$

Auxiliary timing variables

For the constraints related to the batch front, two auxiliary variables are required to indicate whether a batch is currently in transit or in the pipeline, see Section 3-5-4 and Fig. 3-9.

The in-transit variable is active when injection has started but ejection has not yet started. The following constraint describes this:

$$w^{\mathrm{s,transit}}_{l,b,k} = w^{\mathrm{s,transit}}_{l,b,k-1} + x^{\mathrm{s,start}}_{l,b,k-1} - y^{\mathrm{s,start}}_{l,b,k-1} \qquad \forall l \in L, \forall b \in B_l, \forall k \in K^{\mathrm{ip}}_{l,b}, k > 1 \tag{3-41}$$

The in-pipeline variable is active when injection has started but ejection has not yet ended. In terms of the other variables, the in-pipeline variable is the sum of the in-transit variable and the ejection-continue variable:

$$w^{\mathrm{s,pipe}}_{l,b,k} = w^{\mathrm{s,transit}}_{l,b,k} + y^{\mathrm{s,cont}}_{l,b,k} \qquad \forall l \in L, \forall b \in B_l, \forall k \in K^{\mathrm{ip}}_{l,b} \tag{3-42}$$

Batch volume balance

Batch volumes increase due to volume injections and decrease due to volume ejections:

$$v^{\mathrm{s}}_{l,b,k} = v^{\mathrm{s}}_{l,b,k-1} + v^{\mathrm{s,in}}_{l,b,k-1} - v^{\mathrm{s,ej}}_{l,b,k-1} \qquad \forall l \in L, \forall b \in B_l, \forall k \in K^{\mathrm{ip}}_{l,b}, k > 1 \tag{3-43}$$

If a batch is not in the pipeline, its volume is zero. Otherwise, the volume is at most $V^{\max}_l$:

$$v^{\mathrm{s}}_{l,b,k} \leq V^{\max}_l w^{\mathrm{s,pipe}}_{l,b,k} \qquad \forall l, b \in B_l, k \tag{3-44}$$

Linking injection and ejection

Recall that batches are pushed through the pipeline, which implies that the batch ejection timing is linked to future batch injections. The batch front location captures these dynamics. The batch front location is the volumetric front coordinate of a batch measured from the pipeline start. When the batch front reaches the end of the pipeline, its ejection should start. The batch front and the relation to other variables is illustrated in Fig. 3-9 and explained in Section 3-5-4. To describe the batch front, we consider the following aspects:

- The batch front of a batch is only non-zero if the batch is in the pipeline:

$$f^{\mathrm{s}}_{l,b,k} \leq V^{\mathrm{pipe}}_l \cdot w^{\mathrm{s,pipe}}_{l,b,k} \qquad \forall l \in L, \forall b \in B_l, \forall k \in K^{\mathrm{ip}}_{l,b} \tag{3-45}$$

- Directly after injection start, the batch front is equal to the first injection volume: $f^{\mathrm{s}}_{l,b,k} = v^{\mathrm{s,in}}_{l,b,k-1}$. The following constraints capture this:

$$v^{\mathrm{s,in}}_{l,b,k-1} - (Q^{\max}_l \cdot \Delta K)(1 - x^{\mathrm{s,start}}_{l,b,k-1}) \leq f^{\mathrm{s}}_{l,b,k} \qquad \forall l \in L, \forall b \in B_l, \forall k \in K^{\mathrm{ip}}_{l,b}, k > 1 \tag{3-46}$$

$$f^{\mathrm{s}}_{l,b,k} \leq f^{\mathrm{s}}_{l,b,k-1} + v^{\mathrm{s,in}}_{l,b,k-1} + (Q^{\max}_l \cdot \Delta K)(1 - x^{\mathrm{s,start}}_{l,b,k-1}) \qquad \forall l \in L, \forall b \in B_l, \forall k \in K^{\mathrm{ip}}_{l,b}, k > 1 \tag{3-47}$$

- In subsequent scheduling time buckets, the front increases due to injections of all batches, and decreases due to own ejections: $f^{\mathrm{s}}_{l,b,k} = f^{\mathrm{s}}_{l,b,k-1} + \sum_{(b' \in B_l)} v^{\mathrm{s,in}}_{l,b',k-1} - v^{\mathrm{s,ej}}_{l,b,k-1}$. The following constraints capture this:

$$f^{\mathrm{s}}_{l,b,k-1} + \sum_{b' \in B_l} v^{\mathrm{s,in}}_{l,b',k-1} - v^{\mathrm{s,ej}}_{l,b,k-1} - (V^{\mathrm{pipe}}_l + Q^{\max}_l \cdot \Delta K)(1 - w^{\mathrm{s,pipe}}_{l,b,k} + x^{\mathrm{s,start}}_{l,b,k-1}) \leq f^{\mathrm{s}}_{l,b,k}$$

$$\forall l \in L, \forall b \in B_l, \forall k \in K^{\mathrm{ip}}_{l,b}, k > 1 \tag{3-48}$$

$$f^{\mathrm{s}}_{l,b,k} \leq f^{\mathrm{s}}_{l,b,k-1} + \sum_{b' \in B_l} v^{\mathrm{s,in}}_{l,b',k-1} - v^{\mathrm{s,ej}}_{l,b,k-1} \qquad \forall l \in L, \forall b \in B_l, \forall k \in K^{\mathrm{ip}}_{l,b}, k > 1 \tag{3-49}$$

  Note that we cannot use these constraints for the scheduling time bucket in which the current batch injection starts, since injections in the previous batch should not increase the batch front of the current batch.
- Ejection can only start if the batch front will exceed the pipeline volume in the next scheduling time bucket. The ejection volume at ejection start is bounded by the amount of overshoot:

$$v^{\mathrm{s,ej}}_{l,b,k} \leq f^{\mathrm{s}}_{l,b,k} + \sum_{b' \in B_l} v^{\mathrm{s,in}}_{l,b',k} - V^{\mathrm{pipe}}_l \cdot y^{\mathrm{s,start}}_{l,b,k} \qquad \forall l \in L, \forall b \in B_l, \forall k \in K^{\mathrm{ej}}_{l,b} \tag{3-50}$$

Stick to the plan

We use the planning output to restrict the number of feasible solutions of the MILP scheduling problem. In primary pipelines, the sum of all batch injections should equal the planned injection volume:

$$\sum_{b \in B_l \cap B_p \cap B_h} \sum_{k \in K^{\mathrm{in}}_{l,b}} v^{\mathrm{s,in}}_{l,b,k} = V^{\mathrm{plan}}_{l,h,p} \qquad \forall l \in L^{\mathrm{prim}}, \forall h \in H_l \setminus H^{\mathrm{LF}}_l, \forall p \in P \tag{3-51}$$

In primary pipelines, each batch injection always takes place in a single planning time bucket. Therefore, each planned batch injection is fully completed at the end of the corresponding planning time bucket. In secondary pipelines, batch injections are linked to ejections from preceding pipelines. As these ejections are typically not aligned with the planning time buckets — recall the situation in Fig. 3-6 — planned batch injections in secondary pipelines are often not fully completed at the end of the scheduling horizon. These partial injections would violate the equality constraint in Eq. (3-51). Therefore, we introduce alternative constraints for secondary pipelines, in which we limit the pipeline inflow and the tank inflow at the start of secondary pipelines:

$$\sum_{b \in B_l \cap B_p \cap B_h} \sum_{k \in K^{\mathrm{in}}_{l,b}} v^{\mathrm{s,in}}_{l,b,k} \leq V^{\mathrm{plan}}_{l,h,p} \qquad \forall l \in L^{\mathrm{sec}}, \forall h \in H_l \setminus H^{\mathrm{LF}}_l, \forall p \in P \tag{3-52}$$

$$\underbrace{\underbrace{\sum_{b \in B_l \cap B_p \cap B_h} \sum_{k \in K^{\mathrm{ej}}_{l,b}} v^{\mathrm{s,ej}}_{l,b,k}}_{\text{preceding pipe outflow}} - \underbrace{\sum_{l' \in L^{\mathrm{sec}}_l} \sum_{b \in B_{l'} \cap B_p \cap B_h} \sum_{k \in K^{\mathrm{in}}_{l',b}} v^{\mathrm{s,in}}_{l',b,k}}_{\text{secondary pipe inflow}}}_{\text{tank inflow}} \leq \underbrace{V^{\mathrm{plan}}_{l,h,p} - \sum_{l' \in L^{\mathrm{sec}}_l} V^{\mathrm{plan}}_{l',h,p}}_{\text{planned tank inflow}} \tag{3-53}$$

$$\forall l \in L, \forall h \in H_l, \forall p \in P$$

Transmix detection

If a new batch injection of product p starts in scheduling time bucket k and the current batch injection contains a different product, then we force the transmix variable $\theta^{\mathrm{s}}_{l,k}$ to 1 with the following constraint:

$$-1 + \sum_{b \in B_l \cap B_p} x^{\mathrm{s,start}}_{l,b,k} + \sum_{b \in B_l \setminus B_p} x^{\mathrm{s,cont}}_{l,b,k} \leq \theta^{\mathrm{s}}_{l,k} \qquad \forall l \in L, \forall p \in P, \forall k \in K \tag{3-54}$$

Pumping costs

Equivalent to Eq. (3-14) in the MILP planning problem, we approximate the pumping costs with piecewise-affine functions. The line segments m are lower bounds on the pumping costs:

$$C^{\mathrm{pump,marg}}_{l,m} \sum_{b \in B_l} v^{\mathrm{s,in}}_{l,b,k} + C^{\mathrm{pump,stat}}_{l,m} \cdot \Delta K \leq \psi^{\mathrm{s}}_{l,k} \qquad \forall l \in L, \forall m \in M, \forall k \in K \tag{3-55}$$

3-5-6 Objective function

The objective function of the MILP scheduling problem only contains two terms, i.e. transmix costs and pumping costs:

$$\underbrace{\sum_{l \in L^{\mathrm{prim}}} \sum_{k \in K} C^{\mathrm{trans}}_{l} \theta^{\mathrm{s}}_{l,k}}_{\text{transmix}} + \underbrace{\sum_{l \in L} \sum_{k \in K} \psi^{\mathrm{s}}_{l,k}}_{\text{pumping}} \tag{3-56}$$

3-6 Summary

In this chapter, we have proposed a novel pipeline scheduling method. It consists of a planning and a scheduling phase that are coupled in a hierarchical decomposition scheme. Both phases contain an MILP problem and a post-processing step. Two main cost drivers of pipeline operation are transmix volumes and pumping energy. We have expressed these costs using linear and piecewise-affine cost structures, respectively.

In the planning phase, global day-to-day transportation volumes are determined such that demand and supply requirements are met, and tanks are operated within their limits. Batch sequences and dynamic transportation times are not yet considered. The planning phase contains a discrete-time MILP problem, which is solved monolithically, and a post-processing step that refines the MILP solution.

In the scheduling phase, the planning output is used to generate complete schedules. Similar to the planning phase, the scheduling phase consists of a discrete-time MILP problem and a post-processing step that refines the MILP solution. The MILP scheduling problem is solved with a rolling-horizon approach to further reduce the computation time.

Chapter 4

Case studies

4-1 Introduction

In this chapter, we test the novel pipeline scheduling method on two different case studies. We have implemented the method in AIMMS 4.30 and solve the Mixed Integer Linear Programming (MILP) problems with IBM ILOG CPLEX 12.7 (64-bit) on an Intel Core i7 (2.0 GHz) CPU with 8 GB of RAM.

The first case study (Section 4-2) is based on the mesh-structure network introduced by Cafaro and Cerdá (2012). After explaining the case details and describing the base case solution, we test the pipeline scheduling method for different tank sizes and supply-demand scenarios. We repeat the experiments with different pumping cost structures — i.e. no costs, affine costs, and piecewise-affine costs — in both the MILP planning problem and the MILP scheduling problem. The main goal of these experiments is to investigate how the pipeline scheduling method performs in these different situations and to see the effects of adopting a piecewise-affine pumping cost structure in terms of solution quality and computation time.

The second case study (Section 4-3) considers a tree-structure network with an increasing number of pipelines. The case study illustrates that the pipeline scheduling method works on a variety of network configurations. Furthermore, we investigate the quality of the decomposition approach by solving the tree-structure network with different solution methods. We compare the results of the proposed pipeline scheduling method to the results obtained without rolling-horizon decomposition and without hierarchical decomposition.

Section 4-4 covers an alternative transmix representation for the MILP planning problem. We test the alternative transmix representation on both case studies to investigate whether the quality of the obtained solutions improves.

4-2 Mesh-structure network

4-2-1 Case details

The network considered in this section is inspired by the mesh-structure network of Cafaro and Cerdá (2012). The network transports four different products and consists of eight locations — of which two supply locations and five demand locations — that are connected by six primary pipelines and two secondary pipelines, see Fig. 4-1. Pipelines PL1 and PL4 have intermediate exits. The orange arrows indicate supply and demand locations.

Although the network is the same as in Cafaro and Cerdá (2012), we solve a different case. The method by Cafaro and Cerdá (2012) cannot handle intermediate supply and demand deadlines. Therefore, the authors assume that there is no incoming supply within their scheduling horizon of 10 days and that all demand volumes are due at the end of the horizon. The rate of fulfillment is limited by a maximum flow rate. Although it is possible to incorporate such demand structures in our pipeline scheduling method, we have not included it yet, as it is not in the scope of this thesis project. Instead, we assume that supply and demand volumes are fixed, see Section 1-2. Therefore, we have decided to generate a different set of input data.

Tables 4-1 to 4-5 contain the input data for this case study. We express volumes in kilobarrels (1 kbbl $\approx$ 158.99 m^3), which is common in the oil industry. Table 4-1 contains the main pipeline parameters. Table 4-2 lists the available storage capacity at each location. Table 4-3 describes the supply and demand scenarios. The pumping cost coefficients are calculated according to Appendix C with the input parameters listed in Table 4-4. The resulting coefficients are included in Tables D-1 and D-2 in Appendix D-1-1. Table 4-5 contains the values of the other cost parameters. For details regarding the initial pipeline contents, see Table D-3.

We solve the case study using planning time buckets with a duration of 24 hours and scheduling time buckets with a duration of 4 hours, for a 30-day horizon. The rolling horizon is based on two parameters, i.e. the scheduling window, which expresses the number of scheduling time buckets for which the problem is solved in one iteration, and the roll period, which expresses by how much the horizon is shifted after one iteration. We set these parameters to 4 and 2 days, respectively. Using larger parameter values only results in an increase in computation time, whereas using smaller parameter values yields worse schedules.

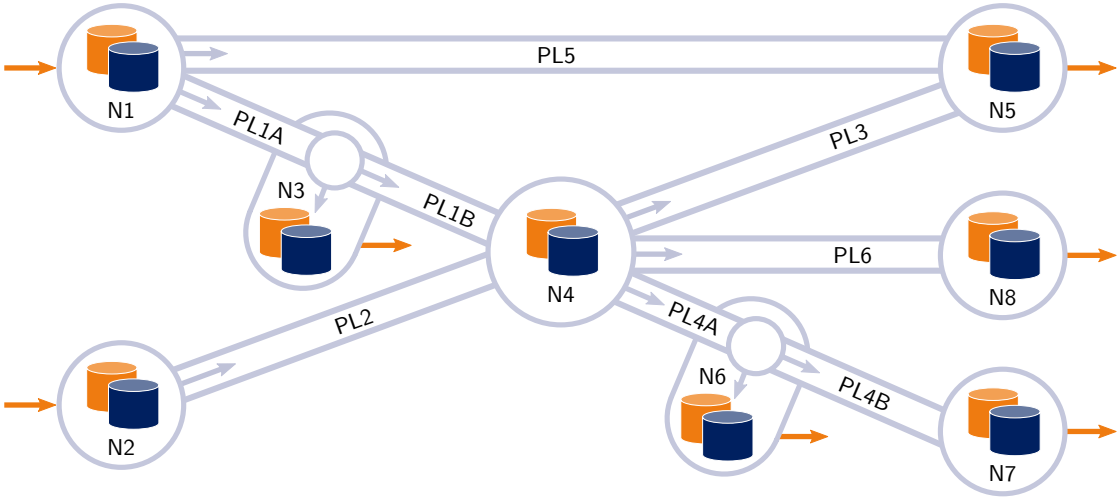


Figure 4-1: Mesh-structure network

Table 4-1: Pipeline characteristics

				Capacity [kbbl/day]		Batch size [kbbl]	
Pipeline	Length [km]	Diameter [inch]	Volume [kbbl]	Min	Max	Min	Max
PL1A	200	20	255	120	150	30	150
PL1B	100	20	127	90	120	24	120
PL2	250	20	319	130	160	32	160
PL3	250	20	319	100	120	24	120
PL4A	150	20	191	100	120	24	120
PL4B	150	20	191	70	120	24	120
PL5	411	12	189	30	40	8	40
PL6	164	12	75	27	40	8	40

Table 4-2: Aggregated tank capacity and opening stock

	Product availability					
Location	P1	P2	P3	P4	Capacity (for available products) [kbbl]	Opening stock [kbbl]
N1	x	x	x		450	225
N2		x		x	350	175
N3	x	x	x		250	125
N4	x	x	x	x	450	225
N5	x	x			350	175
N6				x	250	125
N7	x	x	x	x	350	175
N8				x	250	125

Table 4-3: Supply and demand parameters

	Supply at location [kbbl/day]		Demand at location [kbbl/day]				
Products	N1	N2	N3	N5	N6	N7	N8
P1	95	-	15	55	-	25	-
P2	35	80	10	80	-	25	-
P3	40	-	10	-	-	30	-
P4	-	80	-	-	10	30	40

Table 4-4: Input parameters for the pumping cost estimation

Parameter	Description	Value	
ϵ	Pipeline roughness	0.002	[inch]
η	Pump efficiency	75	[%]
C^{E}	Energy cost	0.20	[$/kWh]
ρ	Average density	800	[kg/m^3]
ν	Average kinematic viscosity	$0.70 \cdot 10^{-6}$	[m^2/s]
g	Gravitational acceleration	9.81	[m/s^2]
$\lvert M \rvert$	Number of line segments	4	[-]

Table 4-5: Cost parameters

Parameter	Description	Value	
C_l^{trans}	Costs per product transition	20 000	[$]
C_l^{shut}	Costs per pipeline shutdown	50 000	[$]
C_l^{down}	Costs per time bucket of downtime	100 000	[$]
$C_{n,p}^{\text{sup}}$	Costs of unfulfilled supply	200 000	[$/kbbl]
$C_{n,p}^{\text{dem}}$	Costs of unfulfilled demand	200 000	[$/kbbl]

4-2-2 Case solution

We solved the case study described in the previous section with the pipeline scheduling method introduced in Chapter 3. Table 4-6 shows the main results.

During the experiments, we obtained feasible solutions within short computation times and with acceptable optimality gaps of less than 5%. However, solving the MILP planning problem to optimality required much more time. In most cases, the branch and bound algorithm did not converge within one hour of computing. Recalling the problem statement in Section 1-2, we are mainly interested in a pipeline scheduling method that returns good solutions within short computation times. Therefore, we suggest to solve the MILP planning problem with truncated branch and bound, i.e. to terminate the optimization before optimality has been proven. When we limit the computation time to 180 seconds, we obtain a solution with an optimality gap of 2.90%, which we consider acceptable.

The MILP scheduling problem is solved in 22.28 seconds of computation time. The final solution consists of $763k pumping costs and $2040k transmix costs. Figs. 4-2 and 4-3 illustrate the results. Fig. 4-2 shows the stock levels at location N4. The stock levels are well within the tank capacity limits, which indicates that inflow and outflow are in balance. Fig. 4-3 partly visualizes the schedule of pipeline PL1A. The Gantt chart shows the injection timing of batches at location N1 and the ejection timing of these batches at location N3. The area chart directly above the Gantt chart shows the pipeline contents over time. The white line at the left side of each colored band represents the batch front. It takes approximately two days to transport batches from the start to the end of the pipeline. Note that batches of the same product are combined as much as possible to limit the number of product transitions.

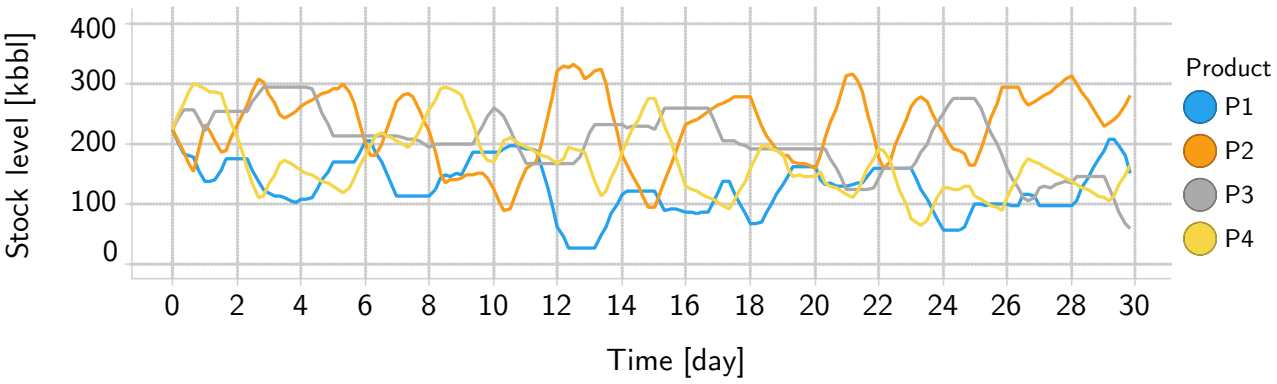


Figure 4-2: Stock levels at location N4

Table 4-6: Case solution

MILP planning problem	
Total number of variables	6 523
Number of binary variables	946
Number of constraints	9 668
Computation time [s]	180.05
Optimality gap [%]	2.90

MILP scheduling problem	
Total number of variables (per iteration)	7 749
Number of binary variables (per iteration)	771
Number of constraints (per iteration)	17 471
Computation time (per iteration) [s]	1.59
Computation time (total) [s]	22.28

Performance indicators	
Pumping cost (MILP solution) [$ ×1000]	763.74
Pumping costs (actual) [$ ×1000]	763.20
Transmix costs [$ ×1000]	2 040.00
Number of pipeline shutdowns	0
Number of downtime days	0
Supply fulfillment [%]	100
Demand fulfillment [%]	100

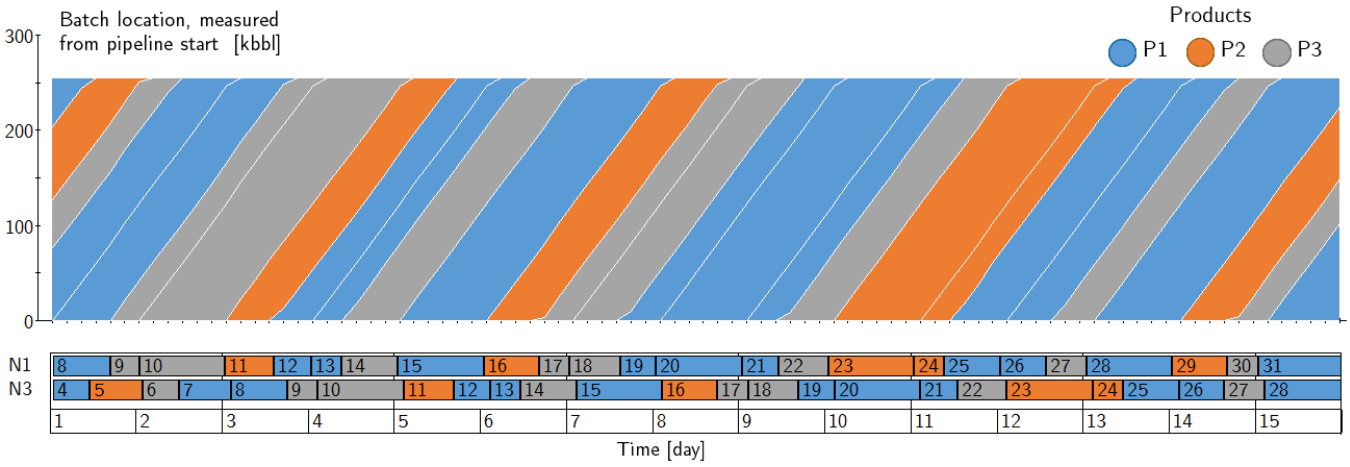


Figure 4-3: Pumping schedule of pipeline PL1A

4-2-3 Tank experiment

In the tank experiment, we vary the storage capacity in the network from 80% to 120% of the original capacity while keeping other parameters constant. We solve these scenarios with different pumping cost structures in both the MILP planning problem and the MILP scheduling problem.

Experiments with different pumping cost structures in the MILP planning problem
To test different cost structures in the planning phase, we solve the MILP planning problem with no, affine, and piecewise-affine pumping costs for different tank sizes. Subsequently, we solve the MILP scheduling problem with piecewise-affine pumping costs. Afterwards, we obtain the actual pumping costs by evaluating the nonlinear equations.

The resulting schedules do not contain any pipeline shutdowns or unfulfilled supply and demand. Hence, the total costs consist of pumping and transmix costs only. Fig. 4-4 shows the sum of the actual pumping costs and the transmix costs for different tank scenarios and planning cost structures. For all cost structures, the costs tend to decrease when tank capacities increase. However, the lines contain some unexpected jumps. By further investigating the cost components, we discover that these jumps are mainly due to transmix costs, see Fig. D-1 in Appendix D-1-2. This could indicate that the transmix representation in the MILP planning problem is too simple. Hence, we test an alternative transmix representation at the end of this chapter, see Section 4-4.

The optimality gaps obtained with different planning cost structures tend to increase when tank sizes decrease. The optimality gaps for the base case are around 3%. For small tank sizes, the optimality gaps increase to 6–7%, see Fig. D-2.

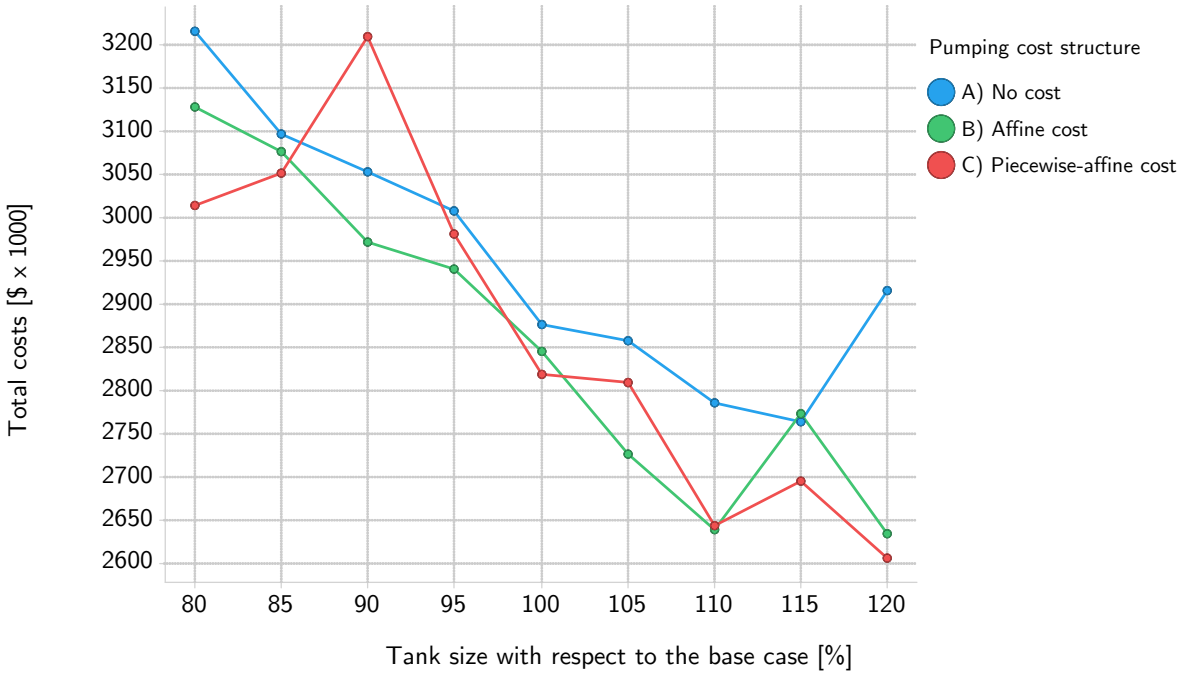


Figure 4-4: Total costs for different tank scenarios and planning cost structures

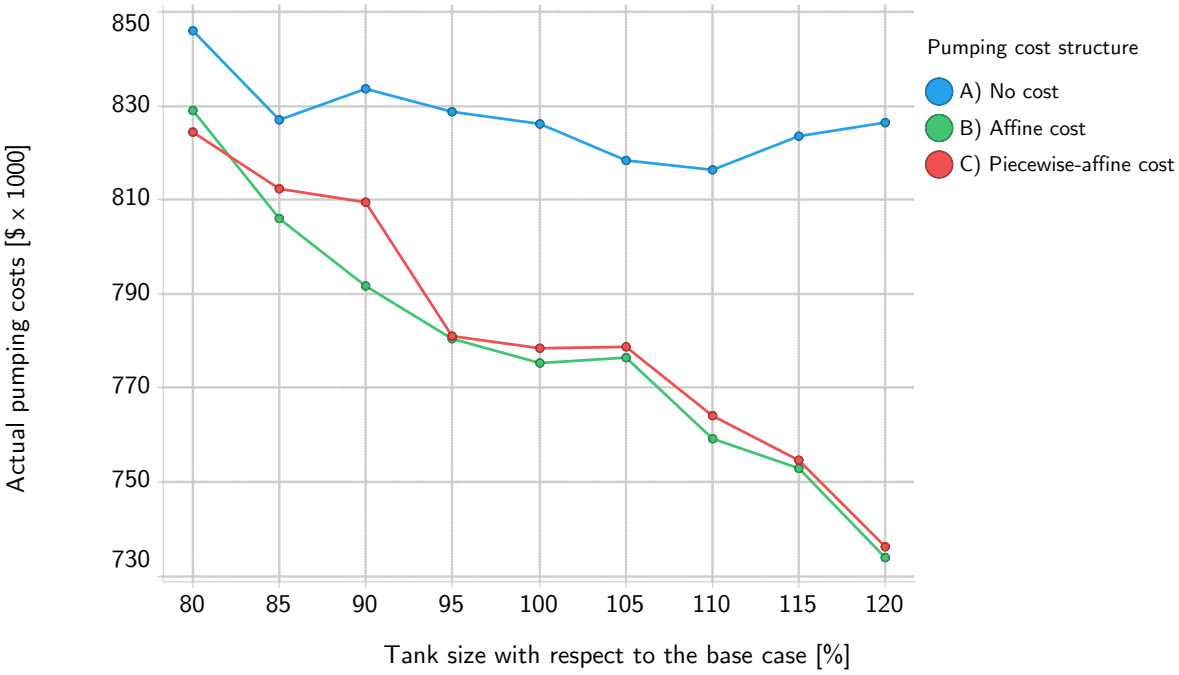


Figure 4-5: Pumping cost for different tank scenarios and planning cost structures

Fig. 4-5 shows the pumping costs for different tank scenarios and planning cost structures. The graph clearly indicates the advantage of including pumping costs in the MILP planning problem, as it results in a cost decrease of 3–11%. It is surprising that the affine cost structure results in slightly better schedules than the piecewise-affine cost structure. We might explain this by investigating the optimality gaps; the solutions obtained with a piecewise-affine cost structure have larger optimality gaps, particularly at tank sizes of 90%, see Fig. D-2. An experiment with longer computation times indeed shows that the piecewise-affine cost structure yields equally good or slightly better schedules than the affine pumping cost structure.

Experiments with different pumping cost structures in the MILP scheduling problem

To test different pumping cost structures in the scheduling phase, we solve the MILP scheduling problem with no, affine, and piecewise-affine pumping costs. We solve the MILP planning problem with a piecewise-affine cost structure. Similar to the previous experiment, we obtain the actual pumping costs by evaluating the original nonlinear equations.

The transmix costs, which are equal for all pumping cost structures, again dominate the total costs. There are some differences in terms of pumping cost, see Fig. 4-6. The curves are very similar for all cost structures, since the injection volumes are based on the same planning output. The pumping costs obtained with a piecewise-affine cost structure are approximately 2% lower than the pumping costs obtained with other cost structures. The small increase in pumping costs at tank sizes of 105% seems strange. However, the pumping costs are only one part of the total costs. Overall, there is still a decrease in cost.

The MILP scheduling problem of the base case is solved in 15–20 seconds of computation time. For scenarios with less tank capacity, the computation times increase to 35–45 seconds. The time differences between different cost structures are small. For more details, see Fig. D-3.

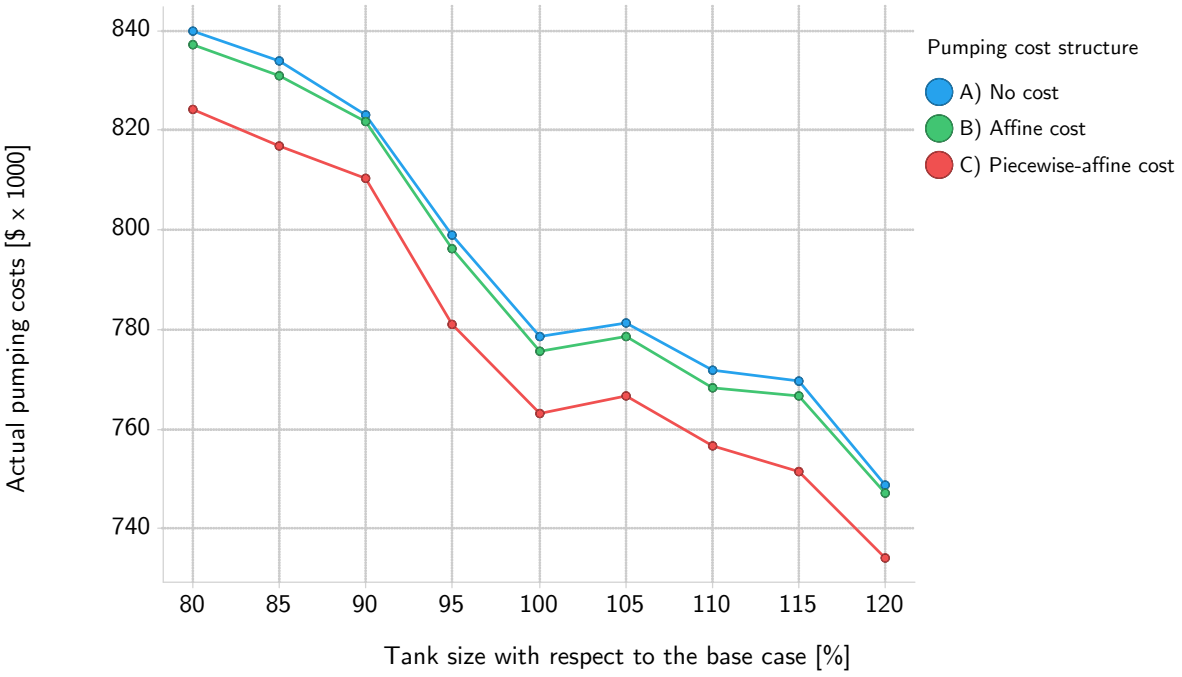


Figure 4-6: Pumping cost for different tank scenarios and scheduling cost structures

4-2-4 Supply-demand experiment

In the supply-demand experiment, we test the pipeline scheduling method for different supply and demand scenarios while keeping the network capacity, i.e. tank and pipeline capacity, constant. Again, we vary the base case in a range of 80% to 120% and solve with different pumping cost structures as explained in Section 4-2-3.

Experiments with different pumping cost structures in the MILP planning problem

Fig. 4-7 shows the total costs for different supply-demand scenarios and planning cost structures. The costs tend to increase when there is more supply and demand to be fulfilled. Similar to the tank experiment, we observe unexpected jumps in costs, which are again due to jumps in the transmix costs. As already mentioned in Section 4-2-3, we propose an alternative transmix representation in Section 4-4. The affine and piecewise-affine pumping cost structures result in the lowest actual pumping costs, see Fig. D-4 in Appendix D-1-3.

Fig. 4-8 shows that pipelines must be shut down for small supply-demand scenarios. There are some small differences between the results obtained with different cost structures in low supply-demand scenarios. Furthermore, not all supply and demand can be fulfilled for high supply-demand scenarios, see Fig. D-5. These supply and demand volumes are lost.

Experiments with different pumping cost structures in the MILP scheduling problem

The results of the experiment with different scheduling cost structures are included in Figs. D-6 and D-7 in Appendix D-1-3. Again, the piecewise-affine cost structure results in the lowest actual pumping costs. The relative difference with respect to other cost structures is approximately 2%, which is in line with the observations in the tank experiment (see Section 4-2-3).

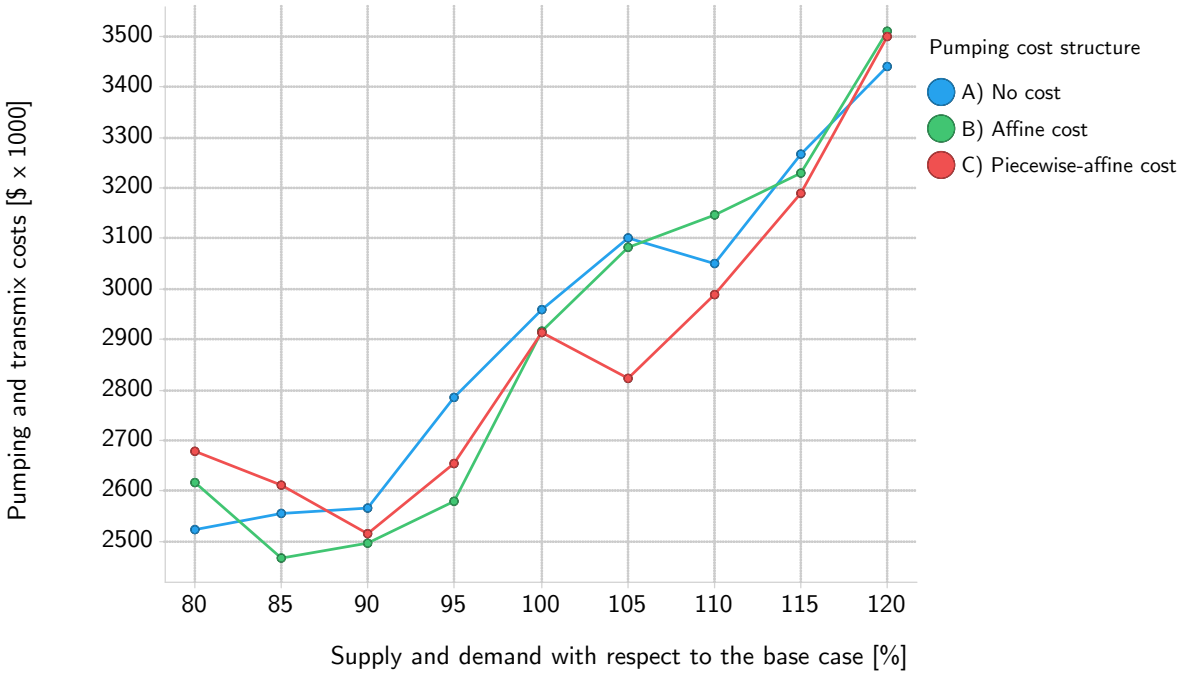


Figure 4-7: Total cost for different supply-demand scenarios and planning cost structures

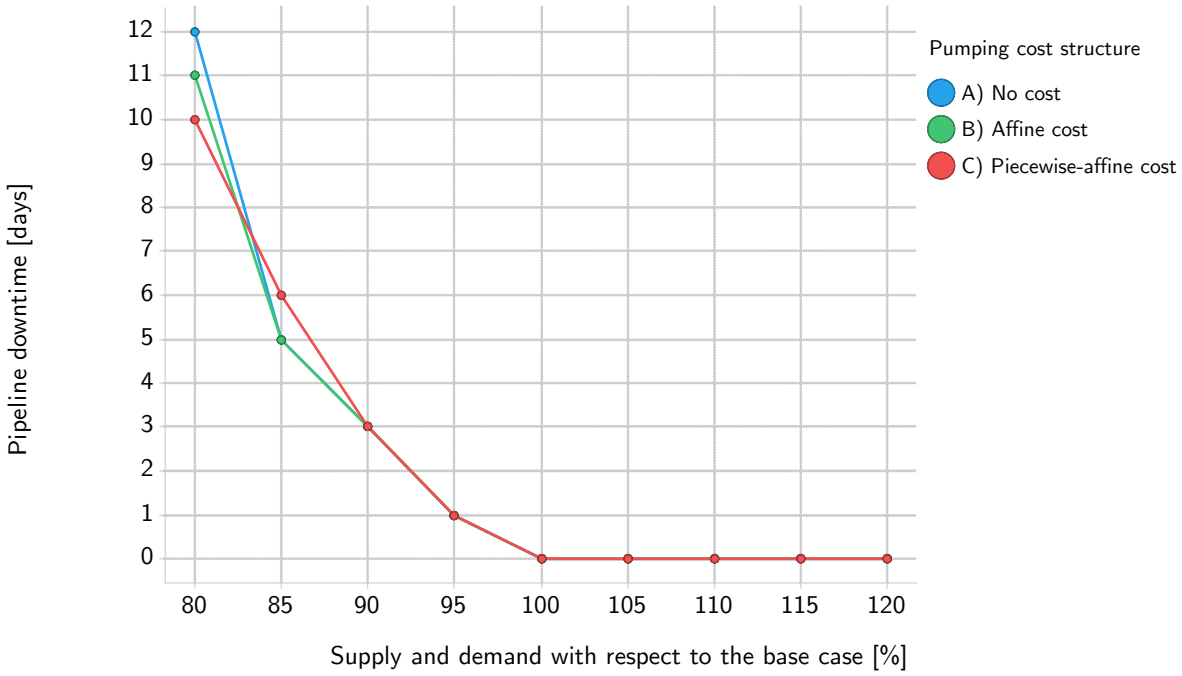


Figure 4-8: Number of downtime days for different supply-demand scenarios and planning cost structures

4-3 Tree-structure network

4-3-1 Case details

In this case study, we consider a tree-structure network with an increasing number of pipelines. The different network configurations are visualized in Fig. 4-9 and described here:

- Network 1 consists of two locations and a single pipeline. Supply arrives at location N1 and demand is picked up at location N2.
- In network 2, we add pipeline PL2 and shift the demand from location N2 to N3.
- In network 3, we add pipeline PL3 and shift half of the supply to location N4.
- In network 4, we split pipeline PL2 into pipelines PL2A and PL2B with an intermediate demand location at location N5.
- In network 5, we shift the demand at location N5 to location N6. We connect location N5 and location N6 using pipeline PL4.
- In network 6, we branch pipeline PL2A into an additional pipeline PL2C that ends at location N7. We shift half of the demand at location N3 to location N7.

The largest network consists of seven locations — of which two supply, three demand, and two intermediary locations — and six pipelines that transport four different products. Table 4-7 lists pipeline dimensions, capacity limits, and batch limits. Table 4-8 describes storage capacities and opening stock levels. The total supply volume is 40 kbbl per product per day, which we evenly distribute among the available supply locations. Next to that, we evenly distribute the nominal demand volumes of 40 kbbl per product per day among the available demand locations. To simulate fluctuations in demand over time, we multiply the nominal demand volumes by a factor γ, which is drawn from a continuous uniform distribution between 0.5 and 1.5 for each planning time bucket, demand location, and product. The randomized demand volumes are used as an input in the MILP planning problem.

We use the same cost parameter values as in the previous case study, see Tables 4-4 and 4-5. The pumping cost coefficients are calculated according to Appendix C. The resulting coefficients are included in Table D-4 in Appendix D-2. Each pipeline contains 25% of each product at the start of the horizon. We solve this case using planning time buckets with a duration of 24 hours and scheduling time buckets with a duration of 6 hours. We again set the rolling-horizon parameters, i.e. the scheduling window and the roll period, to 4 and 2 days, respectively.

Table 4-7: Pipeline characteristics

				Capacity [kbbl/day]		Batch size [kbbl]	
Pipeline	Length [km]	Diameter [inch]	Volume [kbbl]	Min	Max	Min	Max
PL1	150	20	191	75	90	22.5	90
PL2A	200	20	255	150	180	45.0	180
PL2B	200	20	255	50	60	15.0	60
PL2C	100	20	127	50	60	15.0	60
PL3	150	20	191	75	90	22.5	90
PL4	100	20	127	50	60	15.0	60

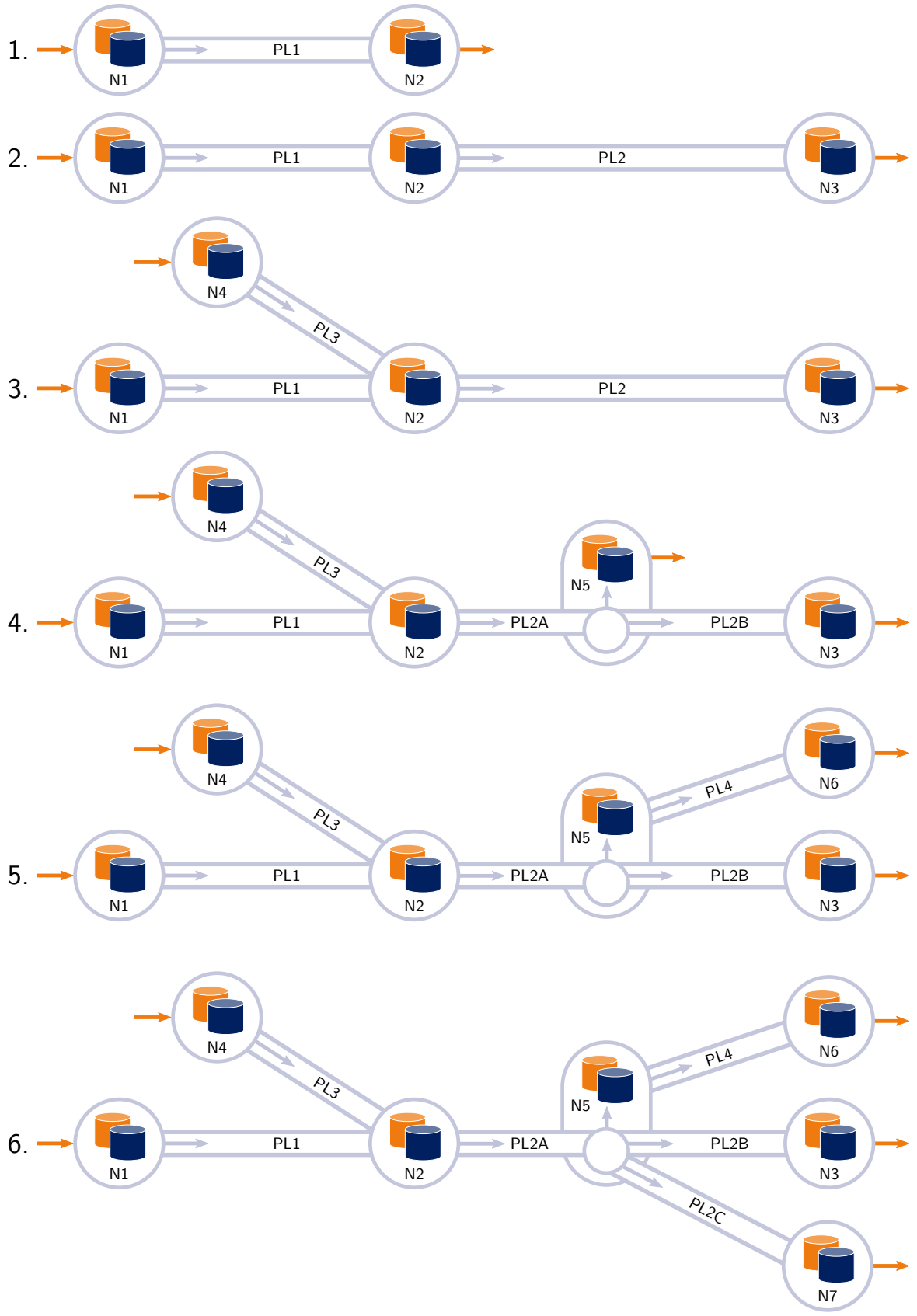


Figure 4-9: Tree-structure networks with an increasing number of pipelines

Table 4-8: Aggregated tank capacity and opening stock

	Product availability					
Location	P1	P2	P3	P4	Capacity per product [kbbl]	Opening stock per product [kbbl]
N1	x	x	x	x	150	75
N2	x	x	x	x	300	150
N3	x	x	x	x	150	75
N4	x	x	x	x	150	75
N5	x	x	x	x	150	75
N6	x	x	x	x	150	75
N7	x	x	x	x	150	75

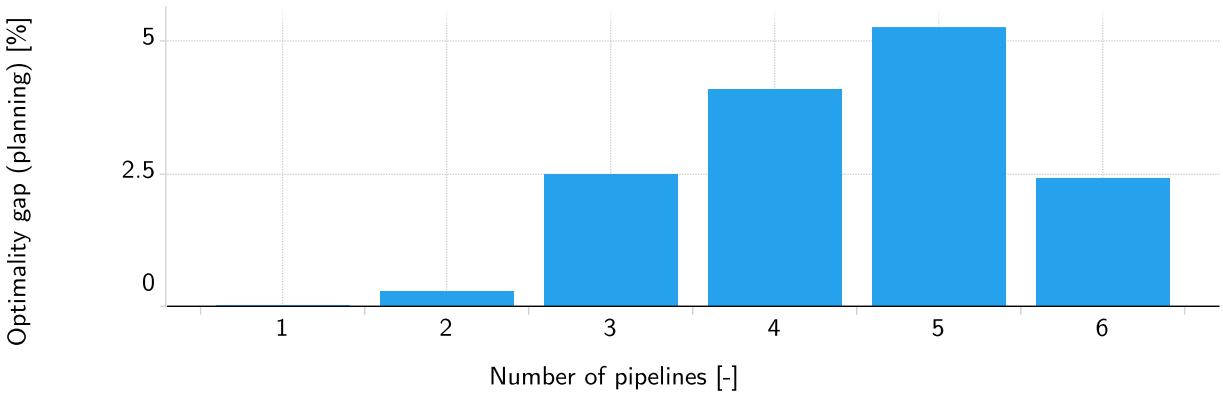


Figure 4-10: Optimality gaps for an increasing number of pipelines (MILP planning problem)

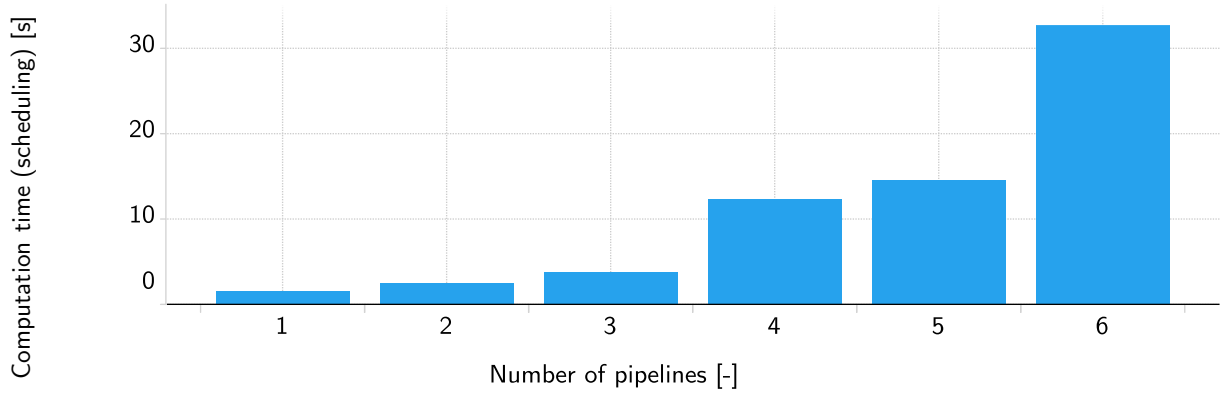


Figure 4-11: Computation times for an increasing number of pipelines (MILP scheduling problem)

4-3-2 Case solution

Table 4-9 shows the results of the case study described in the previous section. The first observation is that all network configurations are successfully solved. Due to the differences between network configurations, it is not straightforward to compare the costs of these networks. However, it is possible to investigate the differences in terms of optimality gaps and computation times. These results are included in Figs. 4-10 and 4-11, respectively.

As in the previous case study, we solve the MILP planning problem with truncated branch and bound. After 180 seconds of computation time, we obtain solutions with acceptable optimality gaps, see Table 4-9 and Fig. 4-10. The optimality gaps range from 0.00% to 5.26% and tend to increase when we increase the number of pipelines.

Fig. 4-11 shows the computation time required to solve the MILP scheduling problems. The computation time clearly increases when we increase the number of pipelines. The difference from three to four pipelines and from five to six pipelines is particularly large. Note that these steps have one common characteristic, i.e. a secondary pipeline is added to the network. Although not documented here, we also observed in other experiments that computation times significantly increase when secondary pipelines are included in the network. Based on these observations, it seems that secondary pipelines have a larger influence on the computation time than primary pipelines. Additional dependencies in secondary pipelines might explain this. Rather than a single injection-ejection relation, the ejection timing of secondary pipelines depends on both the injection and the ejection timing of preceding pipelines.

Table 4-9: Case solution

	Number of pipelines					
	1	2	3	4	5	6
MILP planning problem						
Total number of variables	1 879	2 541	3 323	4 353	5 003	6 289
Number of binary variables	170	340	510	662	832	984
Number of constraints	3 863	4 661	5 579	6 727	7 513	8 917
Computation time [s]	23.25	180.06	180.03	180.06	180.05	180.05
Optimality gap [%]	0.00	0.28	2.49	4.10	5.26	2.41
MILP scheduling problem						
Total number of variables (per iteration)	1 065	1 685	2 392	3 679	4 335	6 684
Number of binary variables (per iteration)	67	132	201	323	400	669
Number of constraints (per iteration)	3 347	4 692	6 155	8 744	10 147	14 102
Computation time (per iteration) [s]	0.11	0.18	0.27	0.88	1.04	2.34
Computation time (total) [s]	1.58	2.48	3.82	12.38	14.54	32.73
Performance indicators						
Pumping cost (MILP scheduling) [$ ×1000]	177.20	618.15	505.95	339.76	350.78	289.02
Pumping costs (actual) [$ ×1000]	177.16	617.93	505.80	339.65	350.69	288.91
Transmix costs [$ ×1000]	600.00	1 200.00	1 840.00	2 200.00	2 780.00	2 740.00
Number of pipeline shutdowns	0	0	0	0	0	0
Number of downtime days	0	0	0	0	0	0
Supply fulfillment [%]	100	100	100	100	100	100
Demand fulfillment [%]	100	100	100	100	100	100

4-3-3 Decomposition experiment

The proposed pipeline scheduling method is based on a MILP planning problem and a MILP scheduling problem that are coupled in a hierarchical decomposition scheme. Furthermore, we solve the MILP scheduling problem with a rolling-horizon approach. The main purpose of using these decomposition methods is to reduce the computation time. However, the quality of the generated schedules might decrease due to these decomposition methods.

In this section, we investigate the quality and performance of different solution methods by performing three experiments, see Fig. 4-12. In the first experiment (Fig. 4-12a), we solve the case study with the hierarchical and rolling-horizon decomposition. The resulting schedule is used as an upper bound in the second experiment (Fig. 4-12b), in which we solve the MILP scheduling problem without the rolling-horizon decomposition. In the third experiment (Fig. 4-12c), we solve the entire case monolithically by dropping the stick-to-the-plan constraints in the MILP scheduling problem, i.e. Eqs. (3-51) to (3-53) in Section 3-5. Again, the schedule obtained in the first experiment is used as an upper bound.

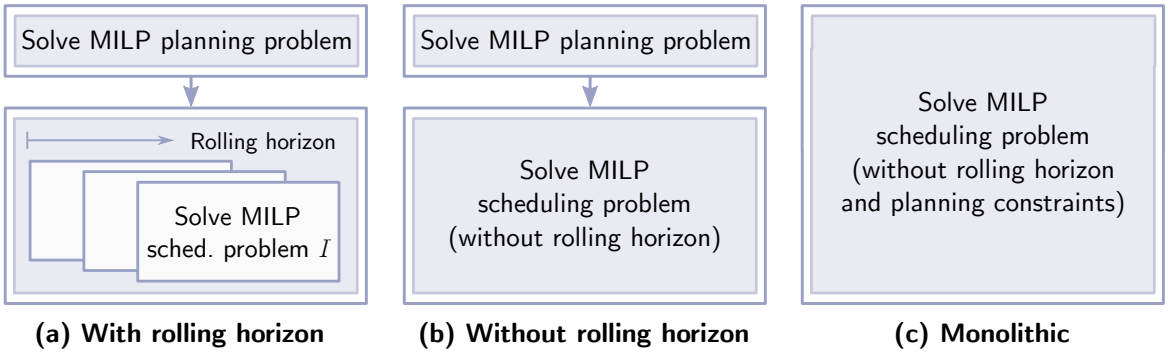
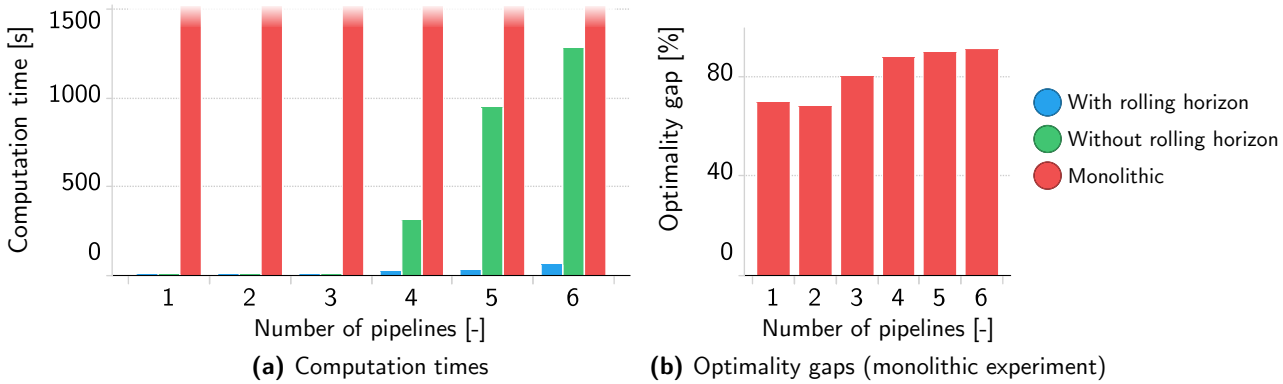


Figure 4-12: Overview of the decomposition experiments (post-processing is not visualized)

The experiments are performed with a maximum computation time of 24 hours. Figs. 4-13 to 4-15 show the main results. Fig. 4-13a visualizes the computation times of the experiments. Although we provide an initial solution, it takes a long time to solve the MILP scheduling problem without rolling horizon, particularly for networks with more than three pipelines. When we solve the problem monolithically, optimality cannot be guaranteed within 24 hours. The remaining optimality gaps are in the range of 70–90%, see Fig. 4-13b. These results already indicate the computational benefits of using a decomposition method.

Figs. 4-14 and 4-15 show the solution quality obtained with different solution methods. The differences between experiments with and without rolling horizon in terms of pumping costs and transmix costs are negligible. Thus, the rolling-horizon decomposition is fast and it does not deteriorate solution quality. On the other hand, the results obtained in the monolithic experiment are slightly better than the results obtained in other experiments. On average, the pumping costs are 6% lower when we use a monolithic solution method. The transmix costs decrease by 8%. Moreover, the solutions obtained with the monolithic solution method could improve even further when longer computation times are used, since the associated optimality gaps are large. Hence, although the hierarchical decomposition approach is fast, it does lead to a decrease in solution quality.



(a) Computation times **(b)** Optimality gaps (monolithic experiment)

Figure 4-13: Computation times and optimality gaps obtained with different solution methods

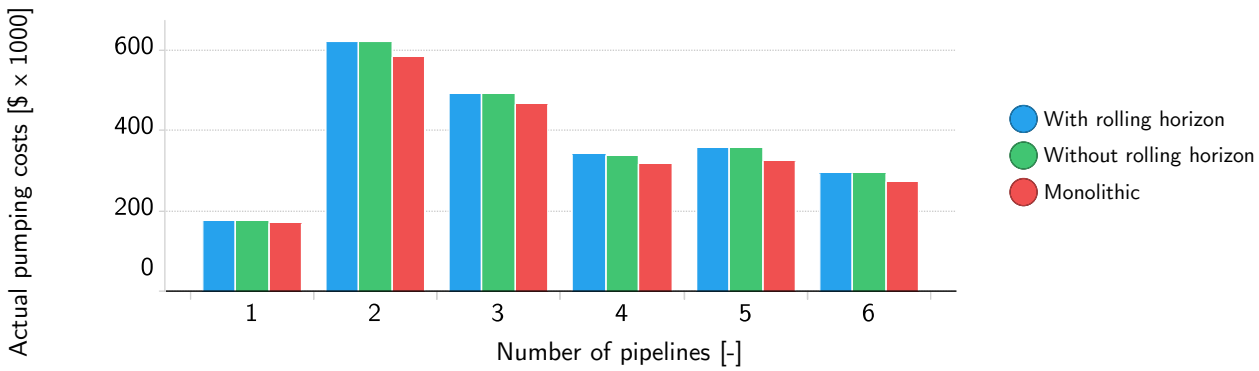


Figure 4-14: Pumping costs obtained with different solution methods

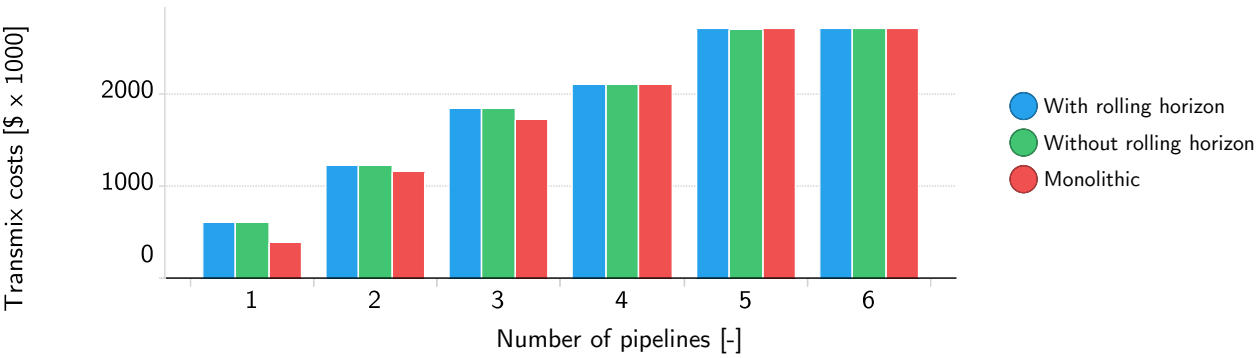


Figure 4-15: Transmix costs obtained with different solution methods

4-4 Transmix experiment

Based on the results of the tank and supply-demand experiments of the mesh-structure network (Sections 4-2-3 and 4-2-4), we expect that the MILP planning formulation can be improved by extending the current transmix representation. In the original formulation of Section 3-4, we approximate the number of product transitions by the number of products that are contained *within* aggregated planning batches. In that case, we neglect product transitions *between* aggregated planning batches. We expect that including these product transitions in the MILP planning problem will yield better schedules. This experiment is a first attempt to improve the transmix cost structure.

Adjustments to the MILP planning problem

We replace the transmix cost term in the planning objective, Eq. (3-17), by:

$$\sum_{l \in L^{\text{prim}}} \sum_{h \in H} \sum_{p \in P} C_l^{\theta} \theta'_{l,h,p} \tag{4-1}$$

where the variable $\theta'_{l,h,p} \in [0,1]$ indicates to what extent we penalize the presence of product p in aggregated planning batch h in pipeline l. It is preferred to include products in the current batch h that are also present in the previous batch $(h-1)$, because these products can be potentially merged in the scheduling phase. To express this, we distinguish three cases:

$$\theta'_{l,h,p} = \begin{cases} 0 & \text{if } \theta^{\text{p}}_{l,h,p} = 0 \\ 1-\lambda & \text{if } \theta^{\text{p}}_{l,h,p} = 1 \text{ and } \theta^{\text{p}}_{l,h-1,p} = 1 \\ 1 & \text{if } \theta^{\text{p}}_{l,h,p} = 1 \text{ and } \theta^{\text{p}}_{l,h-1,p} = 0 \end{cases} \tag{4-2}$$

where the parameter $\lambda \in [0,1]$ is the inter-batch discount factor. We incorporate Eq. (4-2) in the MILP planning problem using the following constraint:

$$\theta^{\text{p}}_{l,h,p} - \lambda \theta^{\text{p}}_{l,h-1,p} \leq \theta'_{l,h,p} \qquad \forall l \in L^{\text{prim}}, \forall h \in H, \forall p \in P \tag{4-3}$$

We include Eqs. (4-1) and (4-3) in the MILP planning formulation of Section 3-4 and solve both case studies with the adjusted MILP planning problem. We change the restriction on computation time to 300 seconds to make sure that the obtained optimality gaps are still acceptable. During the experiments described in Sections 4-2 and 4-3, we observed that most aggregated planning batches contain two products. Thus, if the current planning batch contains a product that is also contained in the preceding planning batch, there is chance of 50% that we can combine them. Therefore, we set the inter-batch discount factor to $\lambda = 1/2$.

Case study results of the tree-structure network

Figs. 4-16 and 4-17 show the optimality gaps of the MILP planning solution and the transmix costs of the resulting schedule, respectively. Referring to Fig. 4-16, the optimality gaps obtained with the alternative formulation are large (20%–30%) compared to the original formulation, despite the increased amount of allowed computation time. Nevertheless, the alternative formulation results in schedules with less transmix costs for most networks, see Fig. 4-17. On average, the transmix costs decrease by 5%. When we use shorter computation times, e.g. 180 seconds, we do not observe these improvements. When we use longer computation times, e.g. 1 hour, the transmix costs decrease by another 7%.

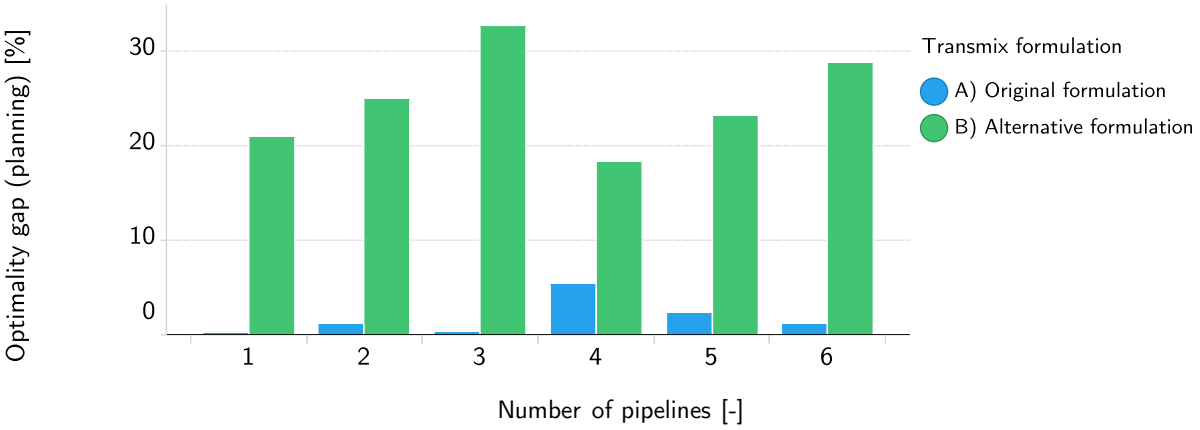


Figure 4-16: Optimality gaps obtained with two transmix formulations (tree-structure network)

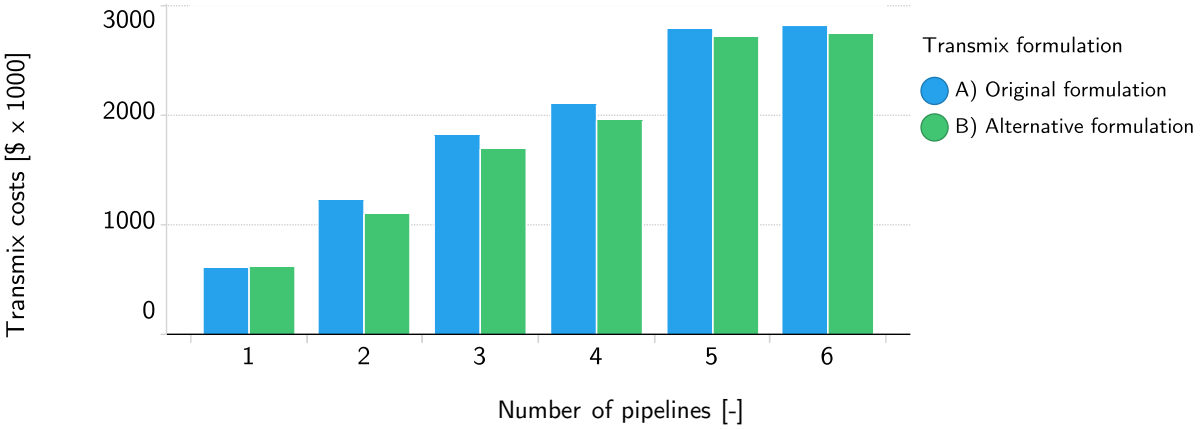


Figure 4-17: Transmix costs obtained with two transmix formulations (tree-structure network)

Case study results of the mesh-structure network

We repeat the tank experiment of the mesh-structure network with the alternative transmix formulation. Again, long computation times of 300 seconds are required to obtain results with an acceptable optimality gap, see Fig. D-8 in Appendix D-3. For small tank sizes, the transmix costs decrease by approximately 10%. For very large tanks, this difference increases to almost 40%, see Fig. D-9. In case of large tank scenarios, the aggregated planning batches will typically contain a single product. Therefore, most product transitions are present between successive planning batches. Hence, incorporating inter-batch transitions in the MILP planning problem is particularly beneficial for scenarios with large tanks.

Final remarks

The first results obtained with the alternative transmix formulation are promising. However, more research is required to improve it, as the optimality gaps are large and the computation times required to reach these gaps are too long. Further improving the alternative transmix formulation remains for future research.

4-5 Summary

The novel pipeline scheduling method has been successfully tested on two case studies. We have decided to solve the MILP planning problem using truncated branch and bound with a time limit of 180 seconds, because near-optimal solutions are found within that amount of time but proving optimality requires much longer. The MILP scheduling problem is solved within 10–60 seconds of computation time. Thus, the pipeline scheduling method returns complete schedules within 3–4 minutes of computation time.

The first case study is inspired by the mesh-structure network of Cafaro and Cerdá (2012). We solved this case with varying tank sizes and supply-demand scenarios. When tank sizes increase, both pumping costs and transmix costs tend to decrease. When supply and demand increase, pumping costs also increase. The transmix costs both increase in low and high supply-demand scenarios. The best schedules were obtained with an affine or a piecewise-affine pumping cost structure in the MILP planning problem, and with a piecewise-affine pumping cost structure in the MILP scheduling problem.

In the second case study, we considered a tree-structure network with an increasing number of pipelines. It seems that secondary pipelines have a larger influence on the computation time than primary pipelines. Next to that, we investigated the quality of the decomposition approach by solving the case study with rolling horizon, without rolling horizon, and completely monolithic. The hierarchical decomposition method and the rolling-horizon decomposition method significantly reduce the computation time required to solve the pipeline scheduling problem. The differences in solution quality obtained with and without rolling-horizon decomposition are negligible. On the other hand, slightly better solutions are obtained with the monolithic solution method. On average, the pumping costs are 6% lower when we use a monolithic solution method. The transmix costs are decreased by 8%. Hence, the hierarchical decomposition approach does lead to a decrease in solution quality.

In the last experiment, we tested an alternative transmix representation for the MILP planning problem. In the alternative transmix representation, the interaction between aggregated planning batches is incorporated in the problem formulation. The preliminary results are promising. The transmix costs decreased by 10–40% in the first case study, and by 5% in the second case study. However, the required computation times and the resulting optimality gaps are significantly larger than in the original transmix representation. Hence, we should further improve the transmix formulation in future research.

Chapter 5

Conclusions and Recommendations

5-1 Conclusions

Given the size and complexity of pipeline networks, planning and scheduling network operations is a complicated task. Referring to the problem statement in Section 1-2, the research in this thesis has been aimed at developing a fast and generic method for creating feasible and good schedules for multi-product oil pipeline networks with known supply and demand. We captured this in the following main research question:

How can we solve the pipeline scheduling problem using mathematical optimization?

We have split the research question into four subquestions, which we answer in this section.

Which optimization framework is most suitable for the pipeline scheduling problem?
There are two main optimization frameworks that can be used to represent the pipeline scheduling problem, i.e. mathematical programming and constraint programming. Mathematical programming is the most suitable framework for this problem, because the algorithms associated with this framework are better at finding good or optimal schedules when many feasible solutions exist, which is the case for the pipeline scheduling problem. Within the mathematical programming framework, we have selected a Mixed Integer Linear Programming (MILP) formulation to represent the pipeline scheduling problem. The results indicate that the choice for mathematical programming indeed leads to fast and good results.

Which problem representation is most suitable for the pipeline scheduling problem?
There are three main problem representations in the mathematical programming framework, i.e. discrete-time, continuous-time, and precedence-based representations. Of these three options, discrete-time representations provide the most flexibility. Discrete-time representations enable, among others, straightforward incorporation of intermediate supply and demand deadlines, and easier integration of planning and scheduling phases in decomposition approaches. Furthermore, optimization problems with discrete-time representations are better from a computational point-of-view, as they are solved faster when using branch and bound.

The pipeline scheduling problem is split into a planning phase and a scheduling phase that are coupled in a hierarchical decomposition scheme. In the planning phase, relatively large time buckets are used, e.g. with a duration of 24 hours, and several simplifying assumptions are made. For example, transportation times are fixed and batches are not yet sequenced. The scheduling phase involves smaller time buckets, e.g. with a duration of 4 hours. Moreover, batch sequencing and dynamic transportation times are considered in the scheduling phase. A batch front variable links the timing of batch injections to the timing of batch ejections. To overcome discretization errors that occur in most discrete-time representations, we have introduced a time-bucket approach in which batches can start and end *within* time buckets. Afterwards, the actual timing of these events is determined with volume-based interpolation. In this way, we obtain high-resolution schedules with a small number of time buckets.

How can we include operational cost in the optimization problem?
The main operational costs of pipeline networks are related to transmix volumes and pumping energy. Transmix volumes emerge when two different products are sequentially pumped into a pipeline. These products mix due to a diffusion-like process, known as axial dispersion. Transmix costs are incurred per product transition. We have incorporated this with a linear cost structure in which costs are assigned to the number of product transitions.

Due to friction losses in pipelines, energy is required to keep the pipeline running. These friction losses increase nonlinearly with the flow rate. We have approximated these nonlinearities using piecewise-affine functions with 4 segments. In the MILP scheduling problem, we observed an average cost decrease of 2% with respect to solutions obtained without piecewise-affine pumping cost approximations. In the MILP planning problem, on the other hand, the affine and piecewise-affine representations result in similar pumping costs.

What is the effect of different solution techniques on computation time and quality?
The proposed pipeline scheduling method is based on a MILP planning problem and a MILP scheduling problem that are coupled in a hierarchical decomposition scheme. Furthermore, we solve the MILP scheduling problem with a rolling-horizon approach. These decompositions are favorable from a computation point-of-view. However, the quality of the generated schedules might decrease due to these decomposition methods. Experiments show that the quality loss due to the rolling-horizon decomposition is negligible. Furthermore, we solved the total pipeline scheduling problem monolithically. After a day of computing, the intermediate solutions of the monolithic approach show an average decrease of 6% in pumping costs and 8% in transmix costs. Hence, the proposed hierarchical decomposition approach does lead to a decrease in solution quality.

Concluding remarks
With respect to current industry practice, the novel pipeline scheduling method can greatly reduce the time required to generate schedules. Compared to current spreadsheet approaches, the mathematical programming formulation is generic and less error-prone. Moreover, the obtained schedules are better in terms of operational cost.

In conclusion, we have successfully developed and tested a novel pipeline scheduling method. The proposed method is flexible in terms of network configurations, intermediate supply and demand requirements, and cost structures. Complete schedules for 30-day horizons are obtained in competitive computation times of 3–4 minutes.

5-2 Recommendations

Although the proposed pipeline scheduling method shows promising results, there are always aspects that can be improved. This section describes recommendations for future work.

Problem extensions

The following problem extensions can be the subject of future research:

- *Tank allocation*
  In addition to batch volumes and timing, the source and destination tanks of each batch should be specified to enable schedule execution. However, the proposed pipeline scheduling method does not consider tank allocation. In future research, it would be interesting to incorporate tank allocation in the problem formulation, e.g. by adding a tank allocation phase to the hierarchical decomposition method. Since the tank allocation problem will be mainly a feasibility problem, we could use a constraint programming approach or a construction heuristic to solve this.

- *Variable energy tariffs*
  When energy tariffs vary over time, e.g. based on peak hours, it might be rewarding to adjust the pipeline flow rate throughout the day. We could incorporate variable energy tariffs in the problem formulation by adding a time index to the corresponding cost parameters. Note that this illustrates the flexibility of the selected discrete-time representation; it is much harder to incorporate time-dependent energy tariffs in continuous-time and precedence-based representations, as there is no fixed time grid.

- *Supply and demand backlog*
  The current problem formulation does not include supply and demand backlog. When supply or demand cannot be met, it is simply lost. However, this situation does not correspond to reality. If demand cannot be fulfilled, it is typically delivered on the next day or replaced by a product of a higher grade. If supply consists of incoming ships and a cargo cannot be unloaded, ships are delayed until storage capacity becomes available. In future work, we could include this behavior in the MILP planning problem.

- *Inventory costs*
  The current problem formulation does not include inventory holding costs, since external factors — supply and demand — determine how much of each product is contained in the network. The underlying assumption is that the inventory holding costs are equal in the entire network. If this assumption does not hold, we could include these costs in the objective function to see whether this has an effect on the resulting schedules.

Case studies

We can address the following aspects to make the case studies more realistic:

- *Improve parameter values*
  Out of all cost parameters, only the pumping costs are based on accurate estimations. Although we have derived a transmix volume based on laws of physics, the corresponding costs are unknown. Also the costs of pipeline shutdowns are unclear. When pipeline operation is resumed, there will be costs related to fluid acceleration. In addition, batch mixing might increase at lower flow rates, yielding larger transmix volumes.

- *Tweak piecewise-affine pumping cost approximation*
  In the case studies in Chapter 4, we have approximated the pumping costs by piecewise-affine function with four line segments that are evenly distributed over the interval of allowed flow rates. The line segments connect five breakpoints that are located on the original nonlinear curve. Hence, the approximation is an overestimation of the actual costs. We could tweak the piecewise-affine approximation by solving a least-squares problem in which we minimize the squared approximation error of the pumping costs.

Transmix representation

Furthermore, we can improve the transmix representation in terms of cost and volume:

- *Improve transmix cost structure*
  In the MILP planning problem, we approximate transmix costs using the number of products that are contained in aggregated planning batches. However, the product transitions between aggregated planning batches are neglected. We expect to obtain better schedules when these inter-batch product transitions are incorporated in the MILP planning problem. The alternative transmix representation of Section 4-4 already improves the transmix cost structure. The results are promising. However, the optimality gaps are large and the corresponding computation times are too long. In future work, we could further improve the proposed transmix representation or investigate other transmix representations.

- *Incorporate transmix volume losses*
  The proposed MILP formulations do not include the cause of transmix costs, i.e. volume loss due to mixing at batch interfaces. The MILP formulations can be improved by taking these volume losses into account. In the simplest form, we can multiply batch volumes by a constant loss factor. Alternatively, we can add a variable loss term to the batch volume balance, which becomes positive based on the product properties of preceding and succeeding batches.

Solution quality

Next to further investigating the schedule quality obtained with the current decomposition method, two suggestions for future work are:

- *Less conservative MILP planning problem*
  The MILP planning and scheduling problems contain conservative tank bounds to prevent feasibility issues during schedule generation and execution. Since the time discretization in the planning phase is rather coarse, the conservatism might result in schedules with unnecessary high operational costs. In future work, we could investigate the effect of (partly) relaxing these conservative tank bounds. This might result in tank capacity violations during scheduling. These violations could be resolved by adding a feedback loop from the scheduling phase to the planning phase in which the MILP planning problem is tightened based on the MILP scheduling solution.

- *Less strict MILP scheduling problem*
  The current MILP scheduling problem strictly obeys the planned injection volumes. In future work, it might be interesting to investigate whether allowing more scheduling freedom leads to better schedules. For example, shifting some volume to the previous or next day may result in schedules with less transmix costs.

Long-term problem extensions

The scope of the pipeline scheduling problem can be further increased by considering the following long-term problem extensions:

- *Scheduling in dynamic environments*
  The proposed pipeline scheduling method is based on deterministic inputs. However, these inputs may change during schedule execution. In future work, we might investigate the effects of disruptions on the network. Consequently, we could develop strategies for coping with these disruptions, both during schedule generation — i.e. proactive, robust scheduling — and during schedule execution — i.e. reactive rescheduling.

- *Optimization of oil supply chains*
  Pipeline networks are typically part of larger oil supply chains. For example, supply and demand locations can be connected to ports, truck terminals, railway stations, and refineries. Future work could consider the interaction of these different entities in a larger supply chain network. In particular, it might be interesting to incorporate a larger part of the oil supply chain in a single optimization problem. The problem can be further extended by considering the related decision-making and information-sharing processes, which typically involve multiple companies with different objectives.

Applications in other fields of research

Parts of the proposed pipeline scheduling method might be applicable to crude oil scheduling and production scheduling:

- *Scheduling of crude oil networks*
  Although the proposed pipeline scheduling method is aimed at scheduling pipeline networks for refined products, it could be extended to schedule crude oil pipeline networks. In crude oil pipeline networks, different crude oils are blended in order to meet quality requirements. When crude blending takes place in tanks only — i.e. batches are not blended in pipelines — then the scheduling problem for crude oil pipelines is similar to the pipeline scheduling problem considered in this thesis. Tank allocation and crude blending in these tanks can be captured in an additional optimization problem.

- *Custom discrete-time representation for production scheduling*
  In this thesis, we have introduced a custom discrete-time representation in which batches can start and end within time buckets. A post-processing step computes the actual timing based on batch sizes. In future work, we could investigate whether similar representations exist for production scheduling. If this is not the case, we could identify potential applications for the custom discrete-time representation. For example, the representation could be applied to scheduling problems in which batch processing times are large and the desired resolution of time is high.

Appendix A

Mixed Integer Non-Linear Programming

The general form of Mixed Integer Non-Linear Programming (MINLP) is as follows (Grossmann, 2002):

$$\begin{aligned} \min \quad & f(x,y) \\ \text{s.t.} \quad & g_j(x,y) \leq 0 \quad \forall j \in J \\ & x \in X \\ & y \in Y \end{aligned} \tag{A-1}$$

where f and g are differentiable functions, J is a set of constraint indices, and x and y are continuous and discrete variables, respectively. The set X describes a convex continuous domain. The set Y is a set of integer points, which are in most cases binary.

MINLP problems can be solved with nonlinear solvers, such as outer approximation and spatial branch and bound. An alternative approach is to use piecewise-affine relaxations and approximations in order to transform MINLP problems into linear problems that can be solved with efficient Mixed Integer Linear Programming (MILP) solvers. Piecewise-affine approaches are particularly interesting for MINLP problems with nonconvex relaxations, as the corresponding MILP problems have convex, linear relaxations. A typical disadvantage of piecewise-affine approaches is that they introduce additional binary variables in the problem formulation.

Outer approximation

Outer approximation is based on an iterative sequence of solving an MILP master problem and a Non-Linear Programming (NLP) subproblem (Duran and Grossmann, 1986). The values of the binary values are determined in the master problem. Therefore, the resulting subproblem is a continuous NLP problem.

The outer approximation is initialized by solving the NLP relaxation of the MINLP problem. Subsequently, the MINLP problem is linearized based on the NLP solution, which is known

as the MILP master problem. Solutions to the master problem are local lower bounds of the optimal solution (D'Ambrosio et al., 2015). Next, the binary variables are fixed and the corresponding NLP subproblem is solved. If a feasible solution is found, it is an upper bound of the optimal solution. A new iteration is initiated by linearizing the MINLP around the new upper bound. The algorithm is terminated when the lower and upper bounds have converged.

According to D'Ambrosio et al. (2015), outer approximation only works well for convex problems, as local lower bounds are valid global lower bounds for convex problems. This assumption does not hold for nonconvex problems. In that case, the algorithm might converge towards a local optimum rather than the global optimum.

Spatial branch and bound

Spatial branch and bound is an optimization algorithm aimed at solving MINLP problems with nonconvex relaxations to global optimality. The total problem is iteratively divided into subproblems by branching on both integer and continuous variables. Locally optimal solutions are used as upper bounds. Lower bounds are obtained by solving convex relaxations of the subproblems. By gradually refining the subproblems, tighter relaxations are obtained. The branching process is continued until the relaxations are tight enough to provide almost feasible solutions, also referred to as ϵ-feasibility (D'Ambrosio et al., 2015).

Piecewise-affine reformulation

MINLP problems can be transformed into MILP problems using piecewise-affine approximations and relaxations. The difference between these two options is illustrated in Fig. A-1. In case of a piecewise-affine approximation, the nonlinear function is replaced by line segments. In case of a piecewise-affine relaxation, the nonlinear function is replaced by envelopes. The relaxed function value must be within one of these envelopes. Additional binary variables are used to indicate which line segment or envelope is activated.

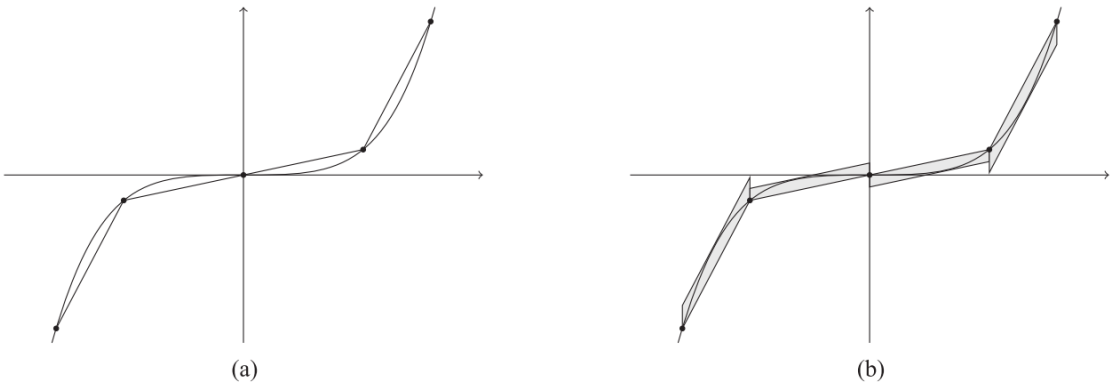


Figure A-1: Difference between piecewise-affine approximations and relaxations (D'Ambrosio et al., 2015)

Appendix B

Transmix estimation

This appendix describes the estimation of typical transmix volumes and costs. Section B-1 explains the dispersion model. Section B-2 discusses a numerical example.

B-1 The dispersion model

During pipeline transport, batch transitions will slowly spread in longitudinal direction, a phenomenon also known as axial dispersion. Axial dispersion can be modeled with an adapted version of Fick's diffusion law (Levenspiel, 1999, p. 295–296):

$$\frac{\partial C}{\partial t} = \mathbf{D}\frac{\partial^2 C}{\partial x^2} \tag{B-1}$$

where $C(x,t)$ is the presence of the transition as a function of time t and location x. Furthermore, $\mathbf{D}$ is the axial dispersion coefficient[1]. In dimensionless form, the equation becomes:

$$\frac{\partial C}{\partial \theta} = \left(\frac{\mathbf{D}}{uL}\right)\frac{\partial^2 C}{\partial \zeta^2} - \frac{\partial C}{\partial \zeta} \tag{B-2}$$

with dimensionless time $\theta = t/\bar{t}$, dimensionless location $\zeta = (ut + x)/L$, mean transportation time $\bar{t}$, flow velocity u, and pipeline length L. The parameter $\left(\frac{\mathbf{D}}{uL}\right)$, also known as the vessel dispersion number, indicates the amount of axial dispersion.

For small values of dispersion ($\left(\frac{\mathbf{D}}{uL}\right) < 0.01$), the transmix region remains bell-shaped and symmetric. In that case, the solution of Eq. (B-2) is an error curve that is similar to the the probability density function of a normal distribution (Levenspiel, 1999, p. 296):

$$C = \frac{1}{\sqrt{4\pi\left(\frac{\mathbf{D}}{uL}\right)}}\exp\left[-\frac{(1-\theta)^2}{4\left(\frac{\mathbf{D}}{uL}\right)}\right] \tag{B-3}$$

[1]For consistency with Fig. B-1, we use a bold $\mathbf{D}$ to denote the axial dispersion coefficient.

In this error curve, the standard deviation of the dimensionless transportation time is:

$$\sigma_\theta = \frac{\sigma_t^2}{\bar{t}^2} = 2\frac{\mathbf{D}}{uL} \tag{B-4}$$

where $\bar{t}$ is the average transportation time, and σ_t is the standard deviation of the transportation time. Then, the volumetric standard deviation σ_V is:

$$\sigma_V = \sigma_t \cdot u \cdot \pi \left(\frac{d}{2}\right)^2 \tag{B-5}$$

Substituting Eq. (B-4) in Eq. (B-5) yields:

$$\sigma_V = \sqrt{2\frac{\mathbf{D}}{uL}} \cdot \bar{t} \cdot u \cdot \pi \left(\frac{d}{2}\right)^2 \tag{B-6}$$

$$= \sqrt{2\frac{\mathbf{D}}{uL}} \cdot L \cdot \pi \left(\frac{d}{2}\right)^2 \tag{B-7}$$

B-2 Transmix cost estimation

In this section, we estimate the volume and costs of a typical transmix volume.

Estimating the transmix volume

The example discussed in this section is based on the parameter values of Table B-1.

Table B-1: Pipeline and fluid parameters for transmix estimation

Quantity		Value (non-SI)		Value (SI)	
d	Pipeline diameter	20	inch	0.51	m
L	Pipeline length	300	km	$300 \cdot 10^3$	m
q	Flow rate	150	kbbl/day	0.276	m^3/s
u	Flow velocity			1.36	m/s
ν	Kinematic viscosity	0.70	cSt	$0.70 \cdot 10^{-6}$	m^2/s

The transmix volume that should be removed depends on the purity requirements before and after the batch transition. For example, if the pipeline flow is considered pure when it contains at most 5% of another product — which should hold for both sides of the symmetric transmix volume — then we should remove 90% of the transmix volume. Since 90% of the surface beneath a normally-distributed bell curve corresponds to $2 \cdot 1.645$ standard deviations, we should remove:

$$V_{\text{transmix}} = 2 \cdot 1.645 \cdot \sigma_V \tag{B-8}$$

We need to know the value of the vessel dispersion number $\frac{\mathbf{D}}{uL}$ to compute the volumetric standard deviation σ_V. Fig. B-1 visualizes a related parameter, i.e. $\frac{\mathbf{D}}{ud}$, for different Reynolds numbers. Using this parameter, we can compute the vessel dispersion number: $\frac{\mathbf{D}}{uL} = \frac{\mathbf{D}}{ud} \cdot \frac{d}{L}$.

For the situation described in Table B-1, the Reynolds number is $\text{Re} = \frac{ud}{\nu} = 9.88 \cdot 10^5$. According to Fig. B-1, the parameter $\frac{\mathbf{D}}{ud}$ is approximately 0.20, yielding a vessel dispersion number of $3.39 \cdot 10^{-7}$. Substituting this number in Eq. (B-7) and then in Eq. (B-8) results in a transmix of 165 m^3 (1.04 kbbl), which is approximately 1% of the average batch size.

Figure B-1: Correlation for the dispersion of fluids flowing in pipes (Levenspiel, 1999, p. 310)

Estimating the transmix cost

Transmix volumes are sold as a different product or reprocessed in a nearby refinery. The costs related to this are difficult to estimate, since they are highly dependent on the actual network, transmix compositions, sales opportunities, and the location of nearby refineries. For example, in different scientific articles e.g. Cafaro and Cerdá (2004, 2012), transmix costs range from \$4 000 to \$40 000.

To provide an illustrative calculation, let us take the pipeline case of Table B-1. Suppose that the transmix volume contains regular and premium fuels. Since the quality of premium fuel is higher than the quality of regular fuel, the transmix volume is sold as regular fuel. The cost difference between these fuels is approximately \$0.40 per gallon (U.S. Department of Energy, 2015), which is \$106 per m^3. Assuming that half of the transmix volume originates from premium fuel, the corresponding downgrading costs are $\frac{1}{2} \cdot 165 \left[m^3\right] \cdot 106 \left[\frac{\$}{m^3}\right] = 8745 \left[\$\right]$.

Appendix C

Pumping cost estimation

This appendix covers the estimation of pumping energy and its associated costs. Section C-1 describes the energy balance for steady incompressible flow. Section C-2 discusses losses in pipelines. Section C-3 provides equations for calculating the pumping power. Section C-4 describes the piecewise-affine approximation of pumping costs.

C-1 Pipeline flow

Pipeline flow can be described with the steady-flow energy equation for incompressible fluids, e.g. see (White, 2011, p. 193):

$$\underbrace{\left(\frac{p_{\text{out}}}{\rho g}+\frac{u_{\text{out}}^2}{2g}+z_{\text{out}}\right)}_{\text{outflow}}-\underbrace{\left(\frac{p_{\text{in}}}{\rho g}+\frac{u_{\text{in}}^2}{2g}+z_{\text{in}}\right)}_{\text{inflow}}=h_{\text{pump}}-h_{\text{loss}} \tag{C-1}$$

with static pressures p_{in}, p_{out}, flow velocities u_{in}, u_{out}, elevation heights z_{in}, z_{out}, pump head[1] h_{pump}, head loss h_{loss}, material density ρ, and gravitational acceleration g. The energy equation expresses that the total amount of static pressure, kinetic, and potential energy between two points in a system is increased by pumps and decreased by energy losses.

We assume that oil has the same static pressure in source and destination tanks, i.e. $p_{\text{in}} = p_{\text{out}}$. Furthermore, if the pipeline diameter is constant, the inflow velocity equals the outflow velocity. Consequently, the terms in Eq. (C-1) related to static pressure and kinetic energy cancel. Thus, pumps should overcome elevation changes and pipeline losses:

$$h_{\text{pump}} = z_2 - z_1 + h_{\text{loss}} \tag{C-2}$$

The right-hand side of Eq. (C-2) corresponds to the *system curve* in Fig. C-1. Section C-2 describes the computation of h_{loss}. The left-hand side of Eq. (C-2) corresponds to the *pump*

[1] In fluid dynamics, head is a different measure for liquid pressure: $h = \frac{p}{\rho g}$.

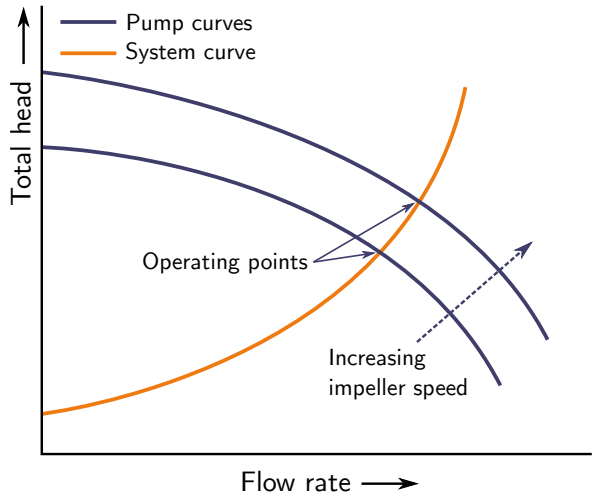


Figure C-1: Pump and system curves

curves in Fig. C-1, which are further described in Section C-3. The equality in Eq. (C-2) implies that the entire system of pipelines and pumps always operates at an intersection of the system and pump curves.

C-2 Head losses

Losses in pipelines are classified as major or minor (White, 2011, p. 399). Major losses correspond to friction losses over the entire pipeline length, whereas minor losses are local losses due to flow at pipeline entrances, exists, fittings, and valves. The total losses are:

$$h_{\text{loss}} = h_{\text{friction}} + \sum_{i \in I} h_{\text{minor},i} = \frac{u^2}{2g}\left(f\frac{L}{D} + \sum_{i \in I} K_i\right) \tag{C-3}$$

with Darcy friction factor f, pipeline length L, pipeline diameter D, and local-loss coefficients K_i for local losses $i \in I$. For large values of L/D, which is the case for long pipelines, the minor losses are negligible.

We calculate the friction losses in pipelines with the Darcy-Weisbach equation:

$$h_{\text{friction}} = f\frac{L}{D}\frac{u^2}{2g} \tag{C-4}$$

The Darcy friction factor f in Eq. (C-4) is typically calculated by performing several iterations of the implicit Colebrook-White equation (Colebrook, 1939):

$$\frac{1}{\sqrt{f}} = -2\log_{10}\left(\frac{\epsilon}{3.7D} + \frac{2.51}{\text{Re}\sqrt{f}}\right) \tag{C-5}$$

with pipeline roughness ϵ and Reynolds number Re. Alternatively, we can obtain f with approximate explicit formulations, such as the equation by Swamee and Jain (1976):

$$f = \frac{0.25}{\left[\log_{10}\left(\frac{\epsilon}{3.7D} + \frac{5.74}{\text{Re}^{0.9}}\right)\right]^2} \tag{C-6}$$

C-3 Pumping power

Pipeline networks are commonly equipped with centrifugal pumps that are driven by electric motors. Fig. C-1 shows typical performance curves for two different impeller speeds. By varying the impeller speed, we can reach different operating points on the system curve. The power required to operate the pump at these operating points is:

$$P = \frac{\rho \cdot g \cdot h_{\text{pump}} \cdot Q}{\eta} \tag{C-7}$$

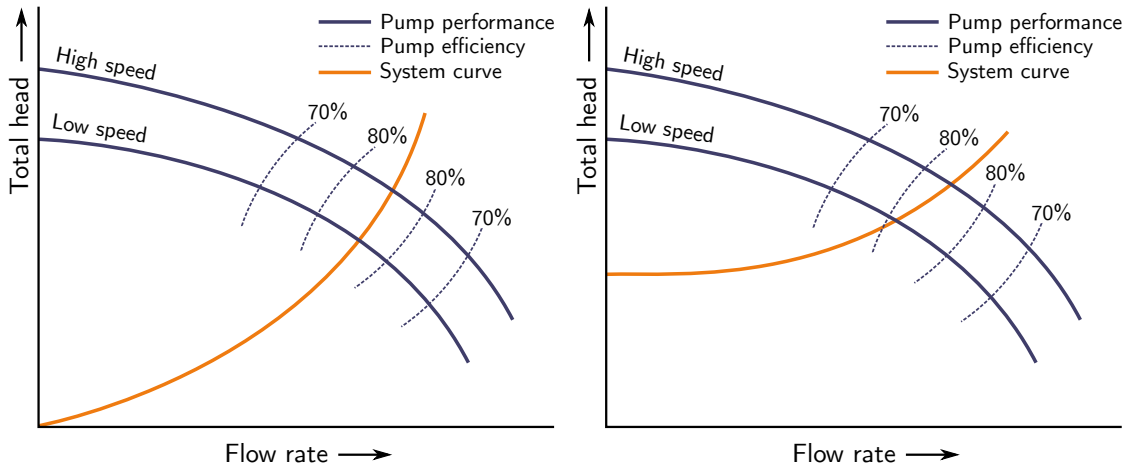
with electric power P, fluid density ρ, gravitational acceleration g, pump head h_{pump}, flow rate[2] Q, and pump efficiency η. The corresponding pumping costs per unit time are:

$$J = C^{\text{E}} \cdot P \tag{C-8}$$

in which C^{E} represents the costs per unit energy.

As explained in Section C-1, the required pump head h_{pump} depends on elevation differences and friction losses at a specific operating point. Next to the required pump head, the pump efficiency η is also dependent on the location of the operating point. Fig. C-2 shows typical level curves of pump efficiency at different operating points. For systems with a small static head, i.e. small elevation changes, the system curve is aligned with the efficiency curves (Fig. C-2a). In that case, the pump efficiency will not change much at different flow rates. However, for systems with a large static head, the system curve does cross the efficiency curves, see Fig. C-2b. Hence, the pump efficiency will vary in that case.

The pump efficiency and the energy use are typically determined by doing measurements on the actual system. Since there was no access to such data during this thesis, we assume a constant pump efficiency in the interval of allowed pipeline flow rates. When the static head is very large, we should reconsider this assumption.



(a) System curve with no static head **(b) System curve with large static head**

Figure C-2: Pump curves and efficiency for different system curves

[2]The relation between velocity and flow rate is $u = \frac{Q}{\pi (D/2)^2}$, with pipeline diameter D.

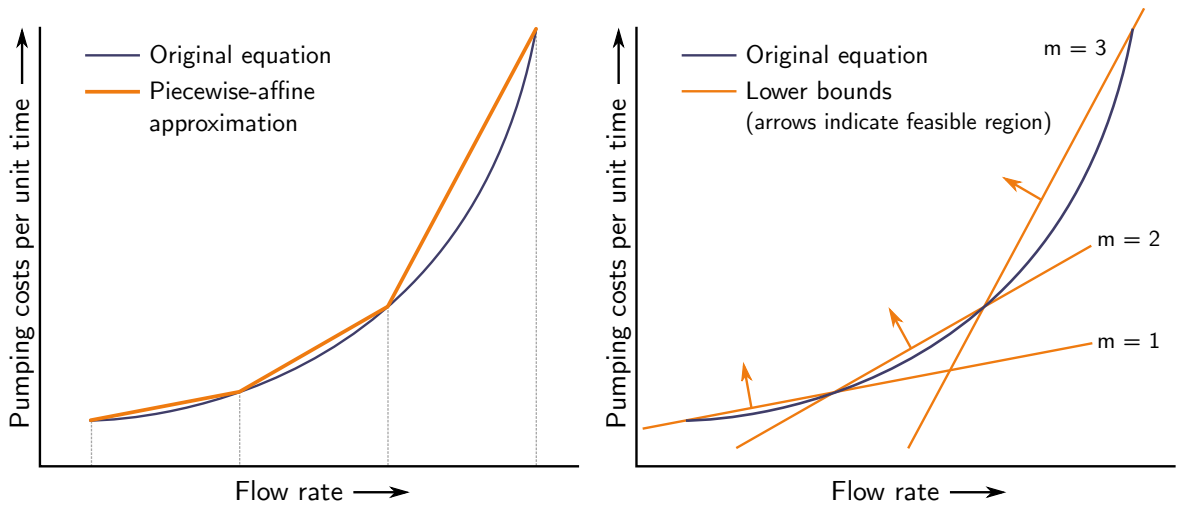
C-4 Piecewise-affine approximation of pumping costs

Piecewise-affine functions consist of multiple line segments. Fig. C-3a shows an example of a piecewise-affine approximation with three line segments. The line segments $m \in M$ for pipeline $l \in L$ are parameterized by $C_{l,m}^{\text{pump,marg}}$ and $C_{l,m}^{\text{pump,stat}}$, which represent the slopes and intercepts of these lines, respectively. Then, in case of the scheduling subproblem, the piecewise-affine pumping costs are expressed by the following set of equations:

$$\psi_{l,k}^{\text{s}} = \begin{cases} C_{l,1}^{\text{pump,marg}} \sum_{b \in B_l} v_{l,b,k}^{\text{s,in}} + C_{l,1}^{\text{pump,stat}} \cdot \Delta K & \text{if } Q_{l,1}^{\text{start}} \leq \sum_{b \in B_l} v_{l,b,k}^{\text{s,in}} < Q_{l,1}^{\text{end}} \\ C_{l,2}^{\text{pump,marg}} \sum_{b \in B_l} v_{l,b,k}^{\text{s,in}} + C_{l,2}^{\text{pump,stat}} \cdot \Delta K & \text{if } Q_{l,2}^{\text{start}} \leq \sum_{b \in B_l} v_{l,b,k}^{\text{s,in}} < Q_{l,2}^{\text{end}} \\ \vdots & \\ C_{l,|M|}^{\text{pump,marg}} \sum_{b \in B_l} v_{l,b,k}^{\text{s,in}} + C_{l,|M|}^{\text{pump,stat}} \cdot \Delta K & \text{if } Q_{l,|M|}^{\text{start}} \leq \sum_{b \in B_l} v_{l,b,k}^{\text{s,in}} \leq Q_{l,|M|}^{\text{end}} \end{cases} \tag{C-9}$$

A similar set of equations holds for the planning subproblem.

The flow rate intervals, slopes, and intercepts can be computed in different ways. In this thesis, we decided to distribute the line segments uniformly over the interval of allowed flow rates. Furthermore, we determine the slopes and intercepts by connecting points that are located on the nonlinear curve, see Fig. C-3a. The main reason to select this approach is its simplicity. Furthermore, the actual pumping costs are calculated in the post-processing step of the scheduling phase. Therefore, the accuracy of the piecewise-affine approximation is of less importance. Nevertheless, it might be interesting to investigate different approaches in future research. For example, we could obtain the slopes and intercepts by solving a least-squares problem in which we minimize the squared approximation error of the pumping costs.



(a) Piecewise-affine approximation **(b)** Line segments as lower bounds

Figure C-3: Expressing piecewise-affine approximations with line segments as lower bounds

The piecewise-affine pumping cost approximation is based on the following steps:

- Initialization of breakpoints and line segments
- Cost calculation at breakpoints
- Calculation of slope and intercept parameters
- Incorporation of piecewise-affine pumping costs in the MILP formulations

In the remainder of this section, we describe these steps in detail.

Initialization of breakpoints and line segments

For all pipelines $l \in L$, the start and end points of each line segment $m \in M$ are:

$$Q_{l,m}^{\text{start}} = Q_l^{\min} + (m-1)\frac{Q_l^{\max} - Q_l^{\min}}{|M| - 1} \tag{C-10}$$

$$Q_{l,m}^{\text{end}} = Q_l^{\min} + m \cdot \frac{Q_l^{\max} - Q_l^{\min}}{|M| - 1} \tag{C-11}$$

where $|M|$ is the cardinality of set M.

Cost calculation at breakpoints

To obtain the costs that correspond to flow rates $Q_{l,m}^{\text{start}}$ and $Q_{l,m}^{\text{end}}$, we repeat the following steps for all pipelines $l \in L$ and for all line segments $m \in M$:

1. Compute the Darcy friction factor f using the implicit Colebrook-White equation (Eq. (C-5)) or the explicit Swamee-Jain equation (Eq. (C-6))
2. Compute the head loss due to friction using the Darcy-Weisbach equation (Eq. (C-4))
3. Compute the required power and the corresponding costs using Eqs. (C-7) and (C-8)

We store the resulting energy costs per unit time in the parameters $J_{l,m}^{\text{start}}$ and $J_{l,m}^{\text{end}}$.

Calculation of slope and intercept parameters

The piecewise-affine functions are parameterized by $C_{l,m}^{\text{pump,marg}}$ and $C_{l,m}^{\text{pump,stat}}$, which represent the slopes and intercepts of the line segments, respectively. The corresponding values are:

$$C_{l,m}^{\text{pump,marg}} = \frac{J_{l,m}^{\text{end}} - J_{l,m}^{\text{start}}}{Q_{l,m}^{\text{end}} - Q_{l,m}^{\text{start}}} \tag{C-12}$$

$$C_{l,m}^{\text{pump,stat}} = J_{l,m}^{\text{start}} - C_{l,m}^{\text{pump,marg}} \cdot Q_{l,m}^{\text{start}} \tag{C-13}$$

Incorporation of piecewise-affine pumping costs in the Mixed Integer Linear Programming (MILP) formulations

As the piecewise-affine functions are convex, we can incorporate them without any binary variables. This is done by bounding the pumping costs ψ from below by each line segment, see Fig. C-3b. In the planning and MILP scheduling problems, the corresponding constraints are included in Eqs. (3-14) and (3-55), respectively. These constraints are:

$$C_{l,m}^{\text{pump,marg}} \sum_{p \in P} \sum_{h \in H_l} v_{l,h,p,t}^{\text{p,in}} + C_{l,m}^{\text{pump,stat}} \cdot \Delta T \leq \psi_{l,t}^{\text{p}} \qquad \forall l \in L, \forall m \in M, \forall t \in T$$

$$C_{l,m}^{\text{pump,marg}} \sum_{b \in B_l} v_{l,b,k}^{\text{s,in}} + C_{l,m}^{\text{pump,stat}} \cdot \Delta K \leq \psi_{l,k}^{\text{s}} \qquad \forall l \in L, \forall m \in M, \forall k \in K$$

Appendix D

Additional case study data

D-1 Mesh-structure network

D-1-1 Case details

Table D-1: Pumping cost parameters used in affine experiments

Pipeline l	Segment m	$C_{l,m}^{\text{pump,marg}}$ [\$/kbbl]	$C_{l,m}^{\text{pump,stat}}$ [\$/day]
PL1A	1	111.09	-9688.06
PL1B	1	34.35	-2300.67
PL2	1	159.31	-14962.53
PL3	1	93.55	-6672.26
PL4A	1	56.13	-4003.36
PL4B	1	43.16	-2446.85
PL5	1	224.66	-5009.57
PL6	1	82.87	-1727.96

Table D-2: Pumping cost parameters used in piecewise-affine experiments

Pipeline l	Segment m	$C_{l,m}^{\text{pump,marg}}$ [\$/kbbl]	$C_{l,m}^{\text{pump,stat}}$ [\$/day]
PL1A	1	93.63	-7592.86
PL1A	2	104.86	-9023.86
PL1A	3	116.70	-10623.39
PL1A	4	129.18	-12400.76
PL1B	1	27.50	-1683.99
PL1B	2	31.86	-2108.98
PL1B	3	36.53	-2599.62
PL1B	4	41.52	-3160.55
PL2	1	135.92	-11922.31
PL2	2	150.99	-13994.54
PL2	3	166.84	-16292.94
PL2	4	183.47	-18829.16
PL3	1	81.52	-5469.27
PL3	2	89.31	-6286.99
PL3	3	97.45	-7181.96
PL3	4	105.93	-8157.63
PL4A	1	48.91	-3281.56
PL4A	2	53.59	-3772.19
PL4A	3	58.47	-4309.18
PL4A	4	63.56	-4894.58
PL4B	1	27.86	-1376.08
PL4B	2	37.19	-2145.35
PL4B	3	47.82	-3155.72
PL4B	4	59.77	-4439.57
PL5	1	179.97	-3668.89
PL5	2	208.40	-4593.03
PL5	3	238.87	-5659.57
PL5	4	271.38	-6878.60
PL6	1	60.80	-1132.04
PL6	2	74.59	-1549.41
PL6	3	89.77	-2057.71
PL6	4	106.31	-2665.78

Table D-3: Initial pipeline contents

Pipeline	Sequence	Product	Volume of total pipeline [%]
PL1A	1	P1	20
PL1A	2	P2	30
PL1A	3	P3	20
PL1A	4	P1	30
PL1B	1	P3	25
PL1B	2	P1	25
PL1B	3	P2	25
PL1B	4	P3	25
PL2	1	P4	30
PL2	2	P2	20
PL2	3	P4	20
PL2	4	P2	30
PL3	1	P1	25
PL3	2	P2	25
PL3	3	P1	25
PL3	4	P2	25
PL4A	1	P1	25
PL4A	2	P4	25
PL4A	3	P2	25
PL4A	4	P4	25
PL4B	1	P2	30
PL4B	2	P1	20
PL4B	3	P3	30
PL4B	4	P4	20
PL5	1	P2	25
PL5	2	P1	25
PL5	3	P2	30
PL5	4	P1	20
PL6	1	P4	100

D-1-2 Tank experiment

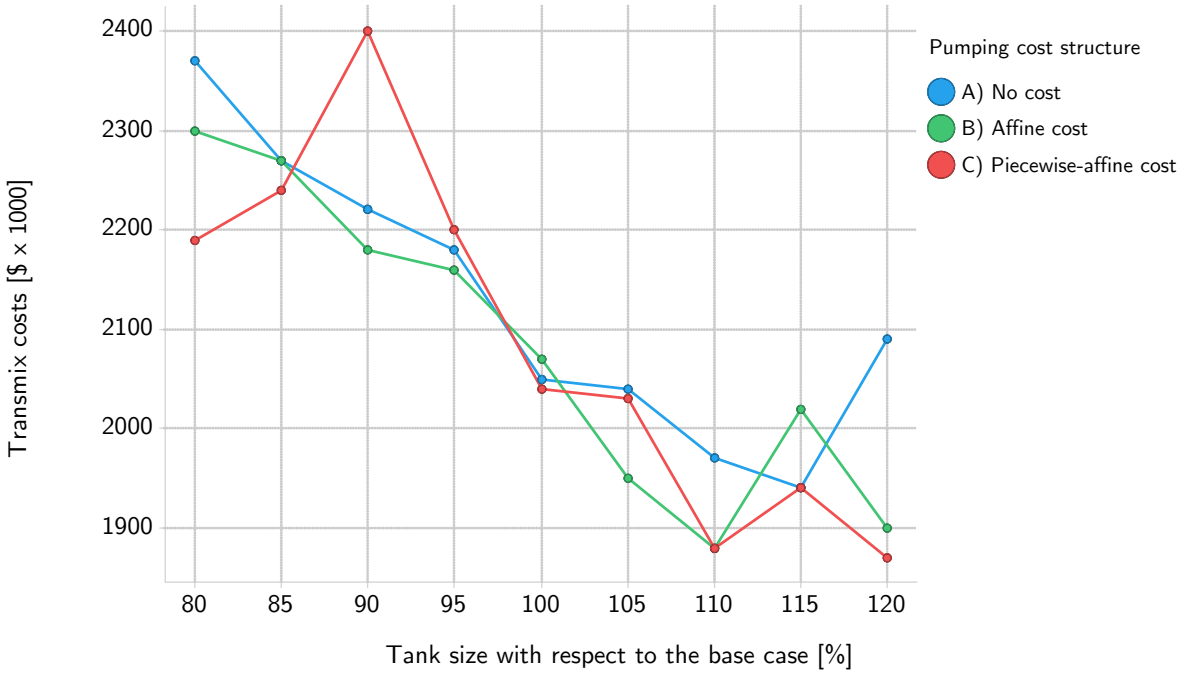


Figure D-1: Transmix costs for different tank scenarios and planning cost structures

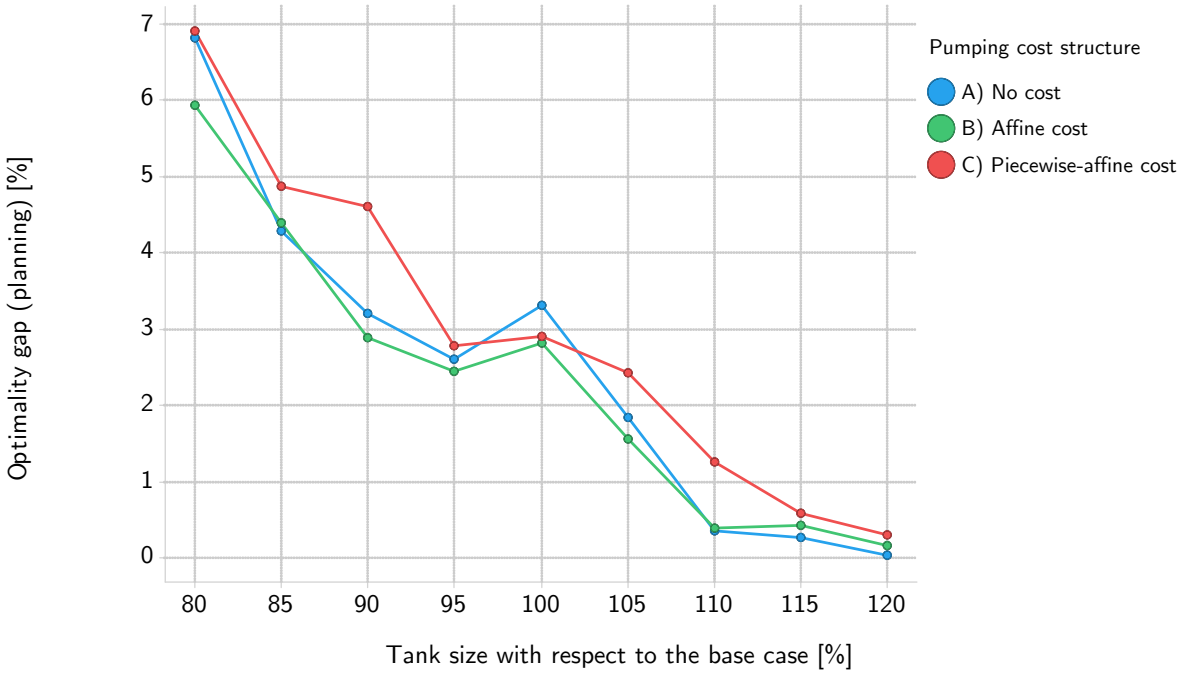


Figure D-2: Optimality gap for different tank scenarios and planning cost structures

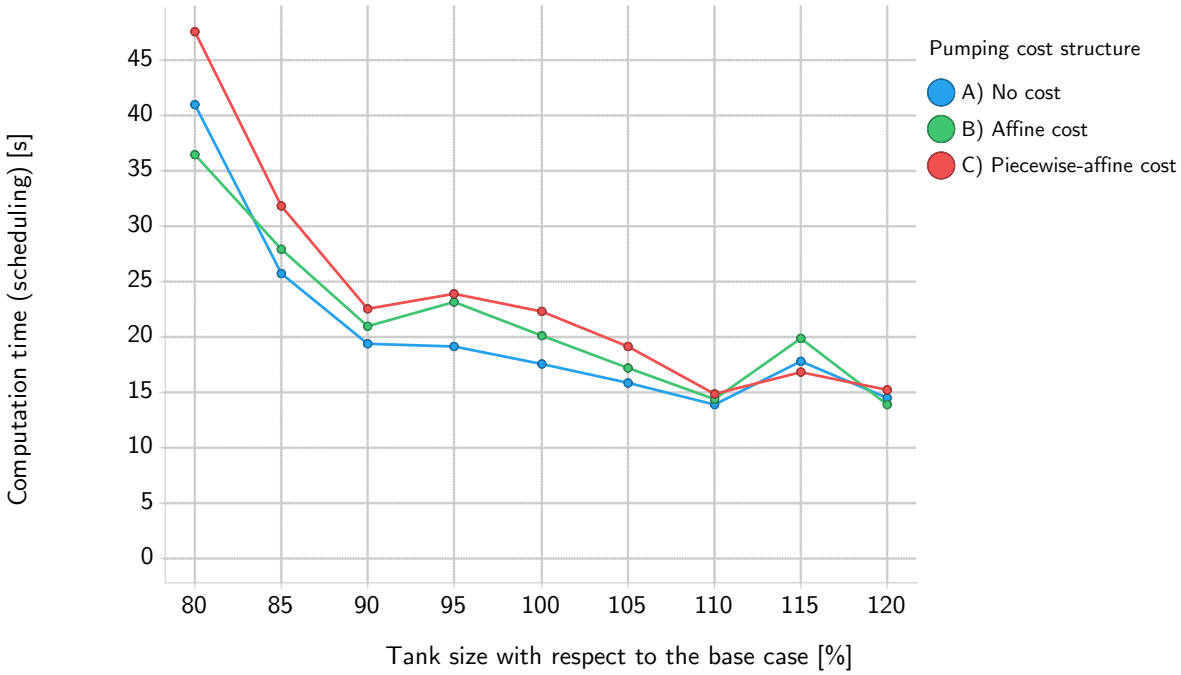


Figure D-3: Computation time for different tank scenarios and scheduling cost structures

D-1-3 Supply-demand experiment

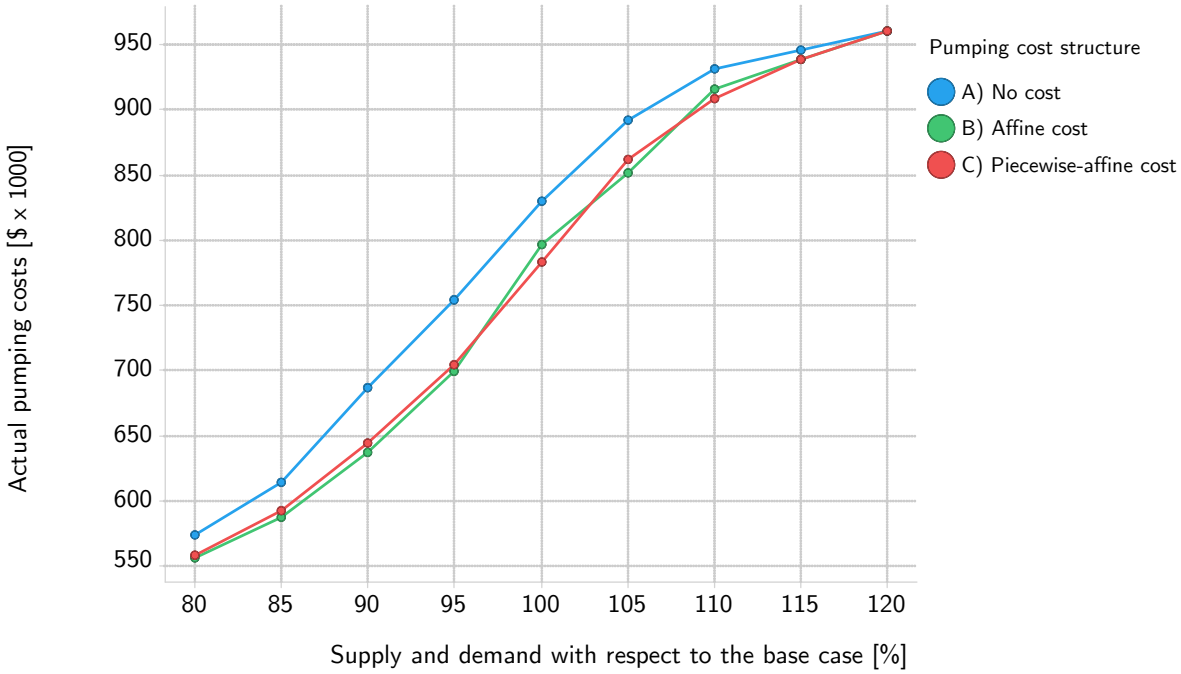


Figure D-4: Pumping cost for different supply-demand scenarios and planning cost structures

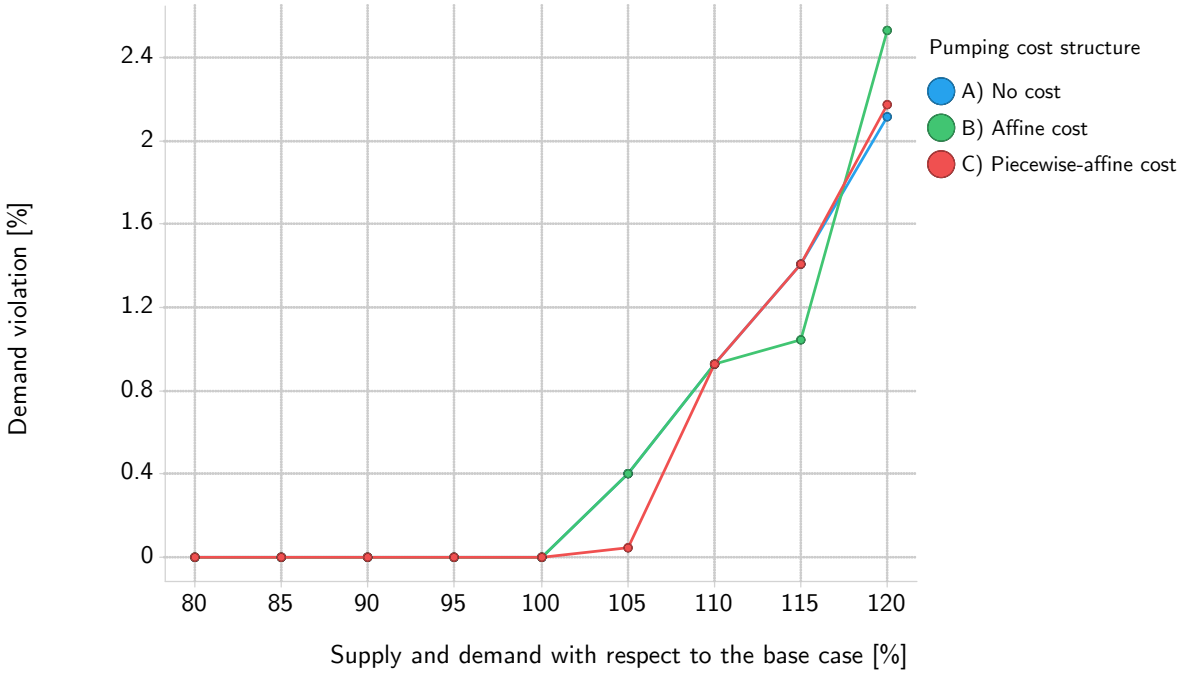


Figure D-5: Violations for different supply-demand scenarios and planning cost structures

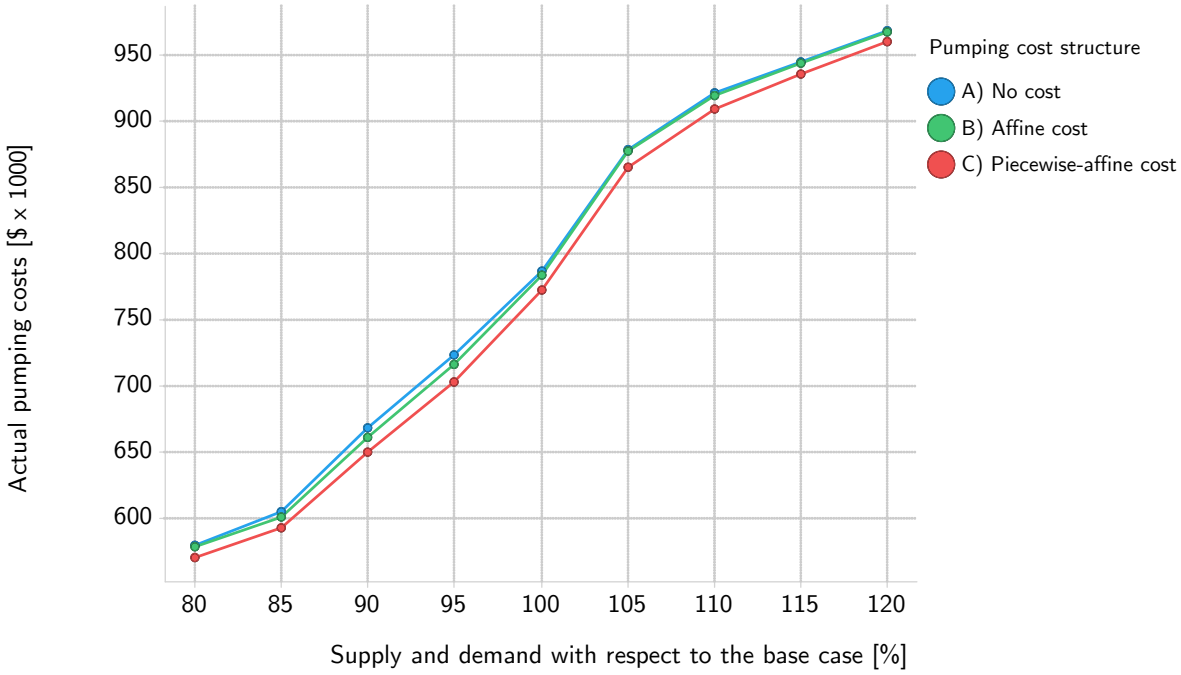


Figure D-6: Pumping cost for different supply-demand scenarios and scheduling cost structures

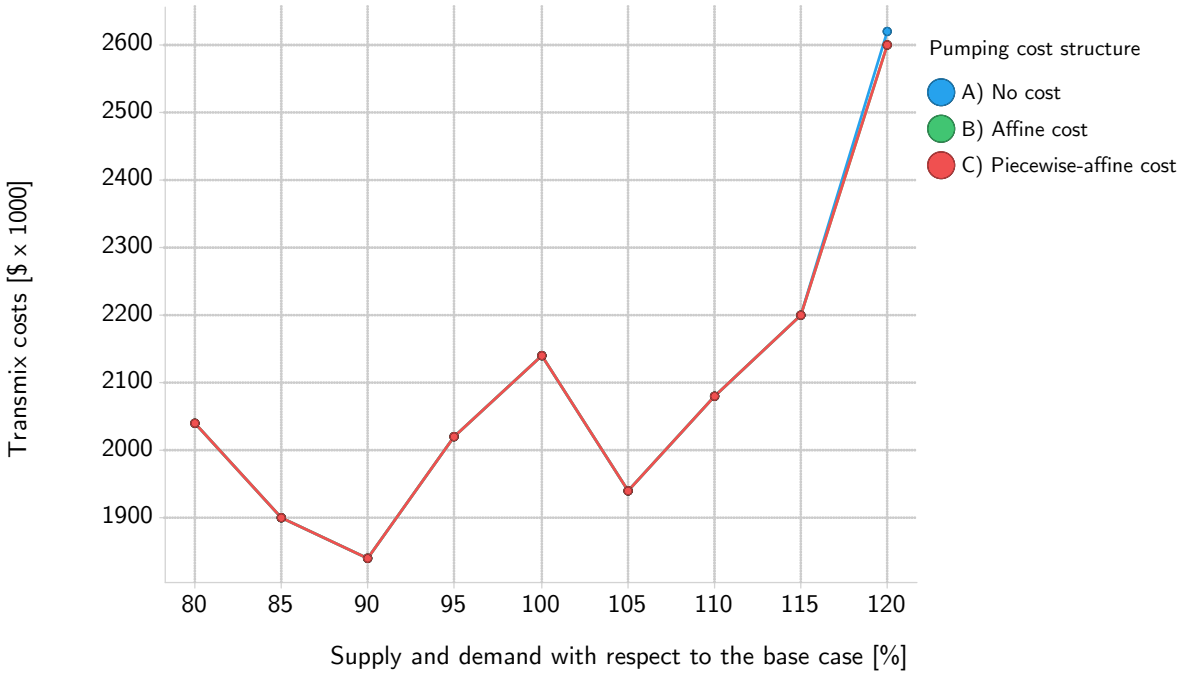


Figure D-7: Transmix cost for different supply-demand scenarios and scheduling cost structures

D-2 Tree-structure network

Table D-4: Pumping cost parameters

Pipeline l	Segment m	$C_{l,m}^{\text{pump,marg}}$ [\$/kbbl]	$C_{l,m}^{\text{pump,stat}}$ [\$/day]
PL1	1	28.25	-1417.90
PL1	2	30.93	-1628.89
PL1	3	33.72	-1859.68
PL1	4	36.64	-2111.15
PL2	1	28.25	-1417.90
PL2	2	30.93	-1628.89
PL2	3	33.72	-1859.68
PL2	4	36.64	-2111.15
PL3A	1	142.27	-14365.29
PL3A	2	156.00	-16526.27
PL3A	3	170.34	-18893.01
PL3A	4	185.31	-21474.80
PL3B	1	17.49	-583.14
PL3B	2	19.13	-669.31
PL3B	3	20.84	-763.48
PL3B	4	22.62	-866.00
PL3C	1	8.74	-291.57
PL3C	2	9.56	-334.65
PL3C	3	10.42	-381.74
PL3C	4	11.31	-433.00
PL4	1	8.74	-291.57
PL4	2	9.56	-334.65
PL4	3	10.42	-381.74
PL4	4	11.31	-433.00

D-3 Transmix experiment

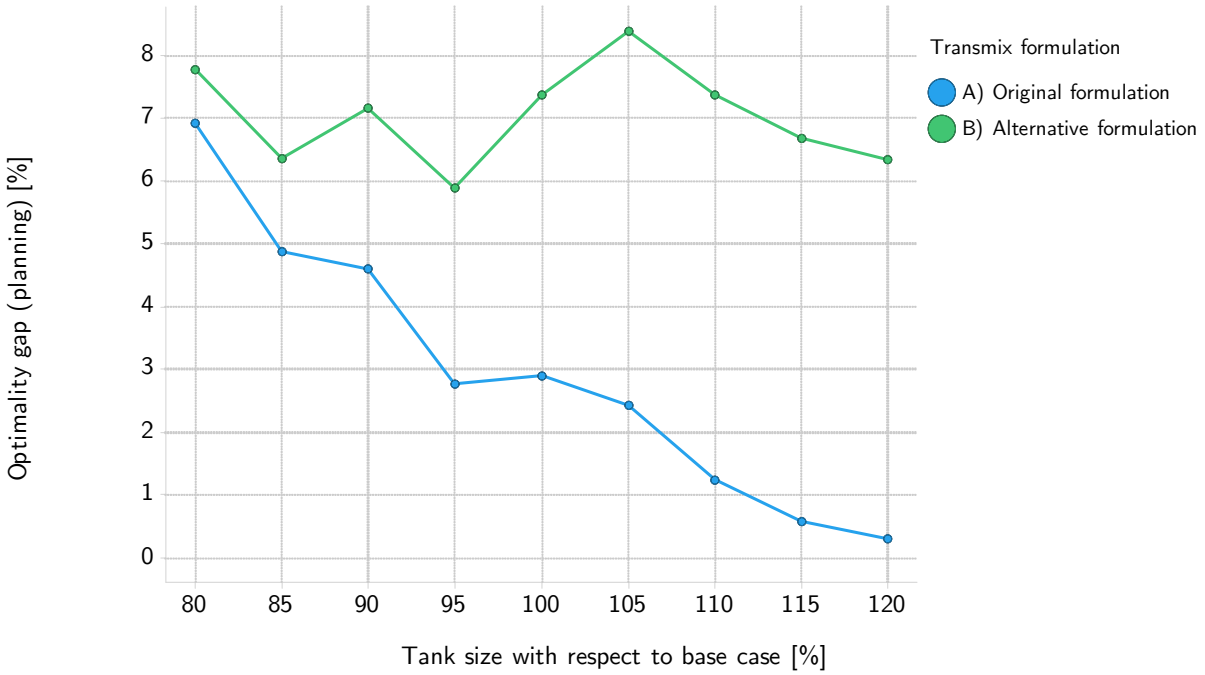


Figure D-8: Optimality gap compared to the original formulation (mesh-structure network)

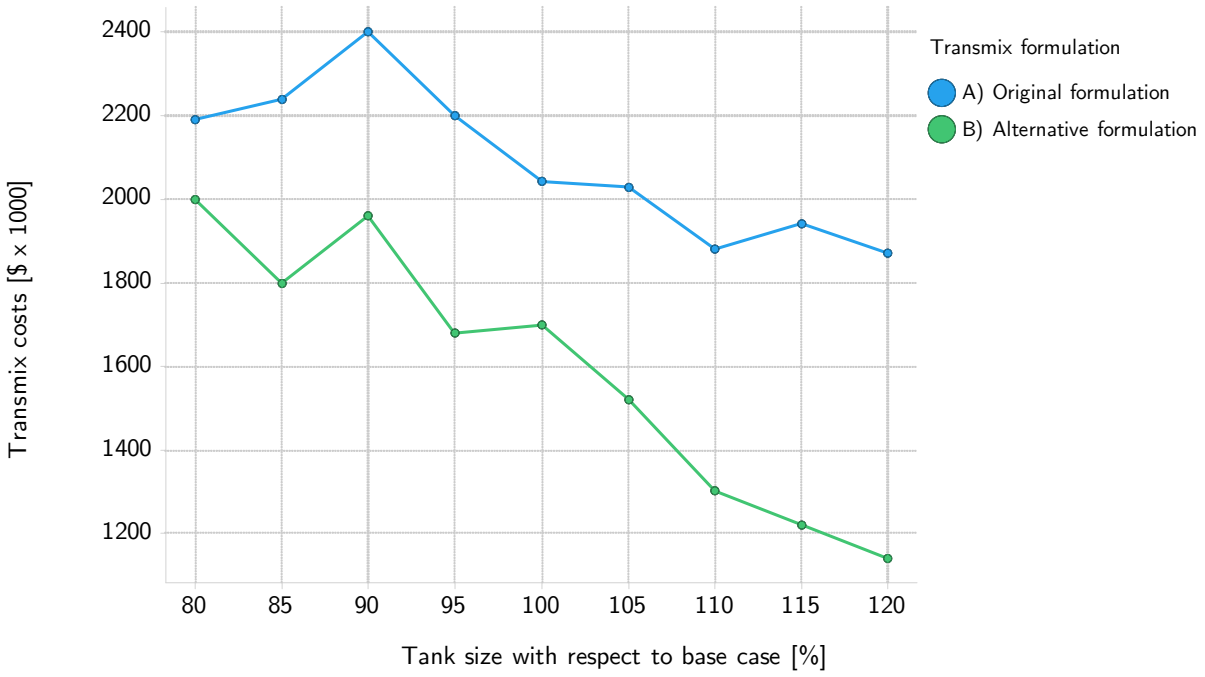


Figure D-9: Transmix costs compared to the original formulation (mesh-structure network)

Bibliography

Association of Oil Pipe Lines and American Petroleum Institute. U.S. Liquids Pipeline Usage & Mileage Report, 2015.

N. Beldiceanu and M. Carlsson. A new multi-resource cumulatives constraint with negative heights. In *International Conference on Principles and Practice of Constraint Programming*, pages 63–79. Springer, 2002.

R. Bixby and E. Rothberg. Progress in computational mixed integer programming - a look back from the other side of the tipping point. *Annals of Operations Research*, 149(1):37–41, 2007.

S. N. Boschetto, L. Magatão, W. M. Brondani, F. Neves-Jr, L. V. Arruda, A. P. Barbosa-Póvoa, and S. Relvas. An operational scheduling model to product distribution through a pipeline network. *Industrial & Engineering Chemistry Research*, 49(12):5661–5682, 2010.

D. C. Cafaro and J. Cerdá. Optimal scheduling of multiproduct pipeline systems using a non-discrete MILP formulation. *Computers & Chemical Engineering*, 28(10):2053–2068, 2004.

D. C. Cafaro and J. Cerdá. Rigorous scheduling of mesh-structure refined petroleum pipeline networks. *Computers & Chemical Engineering*, 38:185–203, 2012.

V. G. Cafaro, D. C. Cafaro, C. A. Méndez, and J. Cerdá. MINLP model for the detailed scheduling of refined products pipelines with flow rate dependent pumping costs. *Computers & Chemical Engineering*, 72:210–221, 2015.

C. F. Colebrook. Turbulent flow in pipes, with particular reference to the transition region between the smooth and rough pipe laws. *Journal of the Institution of Civil engineers*, 12 (8):393–422, 1939.

R. J. Dakin. A tree-search algorithm for mixed integer programming problems. *The Computer Journal*, 8(3):250–255, 1965.

C. D'Ambrosio, A. Lodi, S. Wiese, and C. Bragalli. Mathematical programming techniques in water network optimization. *European Journal of Operational Research*, 243(3):774–788, 2015.

M. A. Duran and I. E. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36(3):307–339, 1986.

F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549, 1986.

I. E. Grossmann. Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and Engineering*, 3(3):227–252, 2002.

I. E. Grossmann. Challenges in the application of mathematical programming in the enterprise-wide optimization of process industries. *Theoretical Foundations of Chemical Engineering*, 48(5):555–573, 2014.

I. Harjunkoski, C. T. Maravelias, P. Bongers, P. M. Castro, S. Engell, I. E. Grossmann, J. Hooker, C. Méndez, G. Sand, and J. Wassick. Scope for industrial applications of production scheduling models and solution methods. *Computers & Chemical Engineering*, 62:161–193, 2014.

R. Haupt. A survey of priority rule-based scheduling. *Operations-Research-Spektrum*, 11(1): 3–16, 1989.

J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

J. Hooker. *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*. John Wiley & Sons, 2002.

S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, et al. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

E. Kondili, C. Pantelides, and R. Sargent. A general algorithm for short-term scheduling of batch operations - I. MILP formulation. *Computers & Chemical Engineering*, 17(2): 211–227, 1993.

A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society*, pages 497–520, 1960.

O. Levenspiel. *Chemical Reaction Engineering*. Wiley, 1999.

T. M. T. Lopes, A. A. Ciré, C. C. de Souza, and A. V. Moura. A hybrid model for a multiproduct pipeline planning and scheduling problem. *Constraints*, 15(2):151–189, 2010.

I. J. Lustig and J.-F. Puget. Program does not equal program: Constraint programming and its relationship to mathematical programming. *Interfaces*, 31(6):29–53, 2001.

C. T. Maravelias. Mixed-time representation for state-task network models. *Industrial & Engineering Chemistry Research*, 44(24):9129–9145, 2005.

C. T. Maravelias. A decomposition framework for the scheduling of single-and multi-stage processes. *Computers & Chemical Engineering*, 30(3):407–420, 2006.

C. T. Maravelias and C. Sung. Integration of production planning and scheduling: Overview, challenges and opportunities. *Computers & Chemical Engineering*, 33(12):1919–1930, 2009.

C. A. Méndez, J. Cerdá, I. E. Grossmann, I. Harjunkoski, and M. Fahl. State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Computers & Chemical Engineering*, 30(6):913–946, 2006.

S. S. Panwalkar and W. Iskander. A survey of scheduling rules. *Operations Research*, 25(1): 45–61, 1977.

R. Rejowski and J. M. Pinto. Scheduling of a multiproduct pipeline system. *Computers & Chemical Engineering*, 27(8):1229–1246, 2003.

R. Rejowski and J. M. Pinto. Efficient MILP formulations and valid cuts for multiproduct pipeline scheduling. *Computers & Chemical Engineering*, 28(8):1511–1528, 2004.

R. Rejowski and J. M. Pinto. A novel continuous time representation for the scheduling of pipeline systems with pumping yield rate constraints. *Computers & Chemical Engineering*, 32(4):1042–1066, 2008.

R. Z. Ríos-Mercado and C. Borraz-Sánchez. Optimization problems in natural gas transportation systems: A state-of-the-art review. *Applied Energy*, 147:536–555, 2015.

N. Shah, C. Pantelides, and R. Sargent. A general algorithm for short-term scheduling of batch operations - I. computational issues. *Computers & Chemical Engineering*, 17(2): 229–244, 1993.

A. Sundaramoorthy and C. T. Maravelias. Computational study of network-based mixed-integer programming approaches for chemical production scheduling. *Industrial & Engineering Chemistry Research*, 50(9):5023–5040, 2011.

P. K. Swamee and A. K. Jain. Explicit equations for pipe-flow problems. *Journal of the Hydraulics Division*, 102(5):657–664, 1976.

U.S. Department of Energy. U.S. Gasoline and Diesel Retail Prices. `https://www.eia.gov/dnav/pet/pet_pri_gnd_dcus_nus_a.htm`, 2015. Retrieved January 2, 2017.

P. Van Hentenryck. *Constraint Satisfaction in Logic Programming*, volume 5. MIT Press Cambridge, 1989.

F. M. White. *Fluid Mechanics*. McGraw-Hill, 2011.

S. Zhang and Q. Xu. Refinery continuous-time crude scheduling with consideration of long-distance pipeline transportation. *Computers & Chemical Engineering*, 75:74–94, 2015.

Glossary

List of Acronyms

LP	Linear Programming
MILP	Mixed Integer Linear Programming
MINLP	Mixed Integer Non-Linear Programming
NLP	Non-Linear Programming

List of Symbols

All sets, parameters, and variables are described in Chapter 3:

- General sets and parameters are described in Section 3-3
- Planning-related sets, parameters, and variables are described in Section 3-4
- Scheduling-related sets, parameters, and variables are described in Section 3-5