Optimization of Interplant Water Reuse in Industrial Parks

Considering Water Treatment Systems

N.A. Koldewijn

TU Delft

SWECO

Optimization of Interplant Water Reuse in Industrial Parks

Considering Water Treatment Systems

by

N.A. Koldewijn

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday January 30, 2024 at 16:00 PM.

Student number:	4543386	
Project duration:	March 9, 2023 – January 30, 2024	
Thesis committee:	Dr. ir. H. Spanjers,	TU Delft, daily supervisor
	Prof. dr. ir L.C. Rietveld,	TU Delft
	Dr. ir. G. Korevaar,	TU Delft
	J.A. Garzón Díaz MSc,	TU Delft, PhD student
	L. van der Schraaf	Sweco, daily supervisor

TU Delft

SWECO

Executive Summary

Reusing water is a crucial part of the solution for addressing the growing concern regarding the risk of water scarcity in industrialized and urbanized areas. This study introduces a tool for the design of water networks, focusing on water reuse in industrial parks. Utilizing a mixed-integer nonlinear programming (MINLP) model developed earlier, this tool is the first in water network design models that operates with open-source software, while considering water treatment systems and multiple constituents. A literature study is conducted to discover shortcomings in water network design models and to find a foundational model to use to develop the tool.

The developed tool creates a water network based on the optimization of the costs of water obtained from water sources, the costs of treatment systems, and optionally the piping costs. The treatment systems are used to regenerate the water for reuse in industrial plants and to meet environmental discharge limits. The tool develops local optimal solutions as an output. Additionally, this study is the first to integrate a water treatment systems database into a water network design model. However, this database needs to be expanded before it is usable. This study demonstrates the tool through three case studies.

N.A. Koldewijn
January 2024

Contents

Nomenclature

List of Acronyms

CAPEX capital expenditures. 24, 32, 42

EPA Environmental Protection Agency. 29

EU European Union. 1, 2

GIS geographic information systems. 45

ISIC international standard industrial classification of all economic activities. 54

IWTT Industrial Wastewater Treatment Technology. 26, 31, 44

LP linear programming. 5, 26, 27, 42

M4H Merwe-Vierhavens. 2

MILP mixed-integer linear programming. 5, 26, 27, 42

MINLP mixed-integer nonlinear programming. 5, 26, 27, 42, 44

MIP mixed-integer programming. 5

MO mathematical optimization. 3

NACE European classification of economic activities. 1, 54

NLP nonlinear programming. 5

OPEX operational expenditures. 24, 32, 42

RCN resource conservation network. 10

WN water network. 10

Glossary

direct recycling the output of an industrial plant or process is used as an input for the same industrial plant or process without applying a treatment step. 4

direct reuse the output of an industrial plant or process is used as an input for another industrial plant or process without applying a treatment step. 4

in-plant inside one industrial plant. 3, 31

industrial symbiosis physical exchange of materials, energy, water, and by-products among diversified clusters of firms. 2

interplant between different industrial plants. 3, 31

mass load the amount of mass of a specific constituent that ends up in the water after the water is used in an industrial plant or industrial process. 23

regeneration-recycling the output of an industrial plant or process is used as an input for the same industrial plant or process after a treatment step. 4, 34

regeneration-reuse the output of an industrial plant or process is used as an input for another industrial plant or process after a treatment step. 4

removal ratio the fraction of a constituent that is removed from the water during water treatment. 4, 31, 39

total water system the combination of all industrial plants and water treatment systems in an industrial park seen as one large system a single boundary, where water from water resources enters and used water is discharged. 2, 5, 21, 22

water losses water not available for reuse or recycling (in this study: evaporated water and water in products). 2, 22, 23

water network design model a set of variables and equations that is used to identify optimal connections to exchange water (in this study: specifically to exchange water between different industrial plants, while introducing water treatment systems). 3

water network design tool a tool that identifies optimal connections to exchange water (in this study: specifically to exchange water between different industrial plants, while introducing water treatment systems). 3

1

Introduction

1.1. Background

The utilization of natural water resources on a global scale due to anthropogenic activities is distributed into 72% for agricultural use, 12% for domestic use, and 16% for industrial applications. The industrial applications include all processes associated with the manufacturing industry and energy production [1]. The share of water resources used for industrial purposes differs substantially across the world. In low-income countries, the average share is approximately 3%, while this rises to 38% in high-income countries. Within the European Union (EU), the share of natural water resources used for industry is 45%, whereas in the Netherlands it reaches 74%, consisting of 12%[2] for manufacturing industries (NACE[3] C) and the remainder for energy supply (NACE D) (Centraal Bureau voor de Statistiek, 2023). More details regarding the classification of manufacturing industries and their water usage are provided in appendix A, the European classification of economic activities (NACE) in particular. (Ritchie & Roser, 2017)

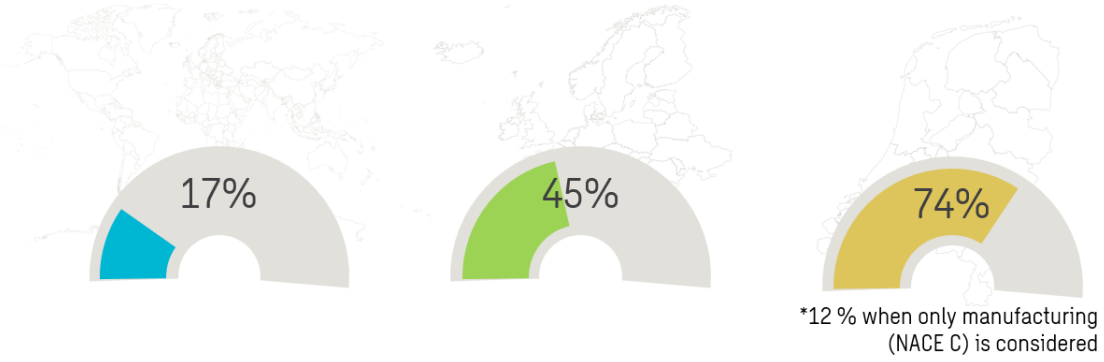


Figure 1.1: Percentages of water withdrawn from natural water resources for industrial purposes, respectively worldwide, in Europe and in the Netherlands

Across manufacturing industries, the water taken in by industrial plants serves diverse purposes[4], each requiring different water quality standards (Spanjers, 2022). The water may undergo various fates

[1] Excluding hydropower generation

[2] 991.1 million m3 for manufacturing out of 8325.8 million m3 of the total Dutch economy (salt surface water is excluded in the calculation)

[3] European classification of economic activities. Short for 'Nomenclature générale des Activités économiques dans les Communautés Européennes' in French

[4] These purposes include activities such as 'transportation, washing, pulping, serving as an ingredient, serving as a reaction medium, cooling, cleaning, sanitation use, and contributing to steam production for purposes like heating, sterilization, propulsion, and motive power' (Spanjers, 2022, slide 8)

after usage in an industrial plant as shown in figure 1.2, namely incorporation into the industrial plant's product, evaporation, or it becomes used water[5]. The evaporated water and the water in products are not available for recycling or reuse, thereby categorizing them as water losses. Contrary to evaporated water and water in products, used water has potential for either recycling within the same plant or for reuse in a different industrial plant, potentially after water treatment. Used water unsuitable for recycling or reuse is discharged. This residual water may need treatment to meet permissible discharge concentrations (or quantities) of water constituents.

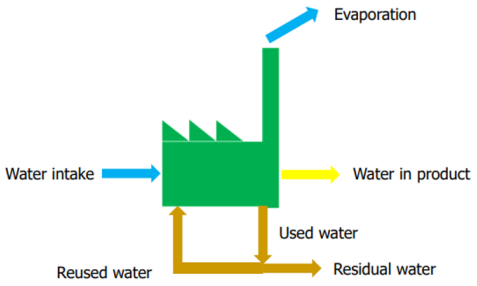


Figure 1.2: Different water in- and outflows around an industrial plant (Spanjers, 2022, slide 75)

Water demands are increasing due to population growth, industrialization and urbanization, which is a risk particularly locally and where an intensification of the water cycle due to climate change affects the availability of natural water resources. Reusing water is a crucial part of the solution for addressing the growing concern regarding the risk of water scarcity in industrialized and urbanized areas. Industrial areas, with their diverse water qualities and closely situated plants, are particularly well-suited for water reuse. The exchange of water in industrial parks, but also the exchange of energy and material resources between different industrial plants, is called industrial symbiosis[6]. Industrial symbiosis is part of industrial ecology, a field where information and knowledge flows between industries are included as well (Erkman, 1997; Massard & Erkman, 2007). An eco-industrial park arises when aforementioned exchanges are applied to an industrial park. The eco-industrial park seen as one system, where water from water resources enters and used water is discharged, is called a total water system.

One of the most well-known eco-industrial parks is located in Kalundborg, Denmark, and developed over the past five decades. At present, a collaborative network exists consisting of eighteen partners from both public and private companies across sectors sharing surpluses of energy, water, and materials with each other (Kalundborg Symbiosis, 2022). In recent years, there is a growing interest in creating more eco-industrial parks such as Kalundborg. The number of articles on industrial symbiosis has greatly increased since 2007, with China being the leading country with the largest number of publications and cases of industrial symbiosis, followed by the United States (Neves et al., 2020). Within the policy of the EU, industrial symbiosis is a key objective in the transition from a linear to a circular economy, with the aim to decrease resource consumption, including natural water resources (European Commission, 2020).

Sweco, a European architecture & engineering firm, conducted a project in the Merwe-Vierhavens (M4H) area in the Dutch city of Rotterdam, where exchanging water was a proposed solution. The M4H area is a former harbor on the edge of the city centre that is in transition to become an area where residential, employment, and manufacturing industry are combined. The existing wastewater treatment plant had limited capacity to expand, so there was a need to make a circular water system, where used urban and industrial water is reused within the area. Sweco provided consultation on water technologies that can be used to achieve a circular water system. A tool to discover possible connections between supply and demand of water within an area is desired for future projects that are of similar nature. (Clevering et al., 2022)

[5] Used water usually has a lesser quality than intake water. Sometimes used water is called 'wastewater', but that term will not be used in this study

[6] In biology, symbiosis is the interaction between two creatures living closely together, typically leading to the advantage of both (Oxford University Press, 2023). Industrial symbiosis is the 'physical exchange of materials, energy, water, and by-products among diversified clusters of firms' (Chertow, 2007, p.11)

1.2. Research Objective & Structure

The objective of this study is to design a tool that identifies optimal connections to exchange water between different industrial plants, while introducing water treatment systems where useful or needed. A visual representation of what this type of tool should conduct is shown in figure 1.3.

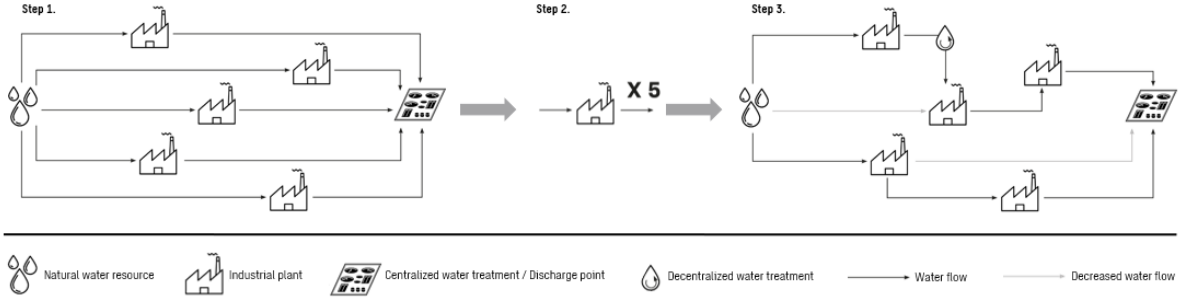


Figure 1.3: The concept of the water network design tool is explained through three steps. Initially, the industrial park consisting of five industrial plants is arranged in a linear system. These plants withdraw water from a water source and discharge it directly after usage. The second step shows the input data needed for the tool, specifically the in- and outflows of water from each industrial plant. In the third step, the tool generates a water network design, illustrating interplant water exchanges, while introducing a decentralized treatment step. This design aims to optimize water usage, resulting in an overall reduction in water withdrawal across the industrial park

Step 1 in figure 1.3 shows a linear system where five different industrial plants withdraw water from a water source. After usage in the plant, the water is either sent to a water treatment facility or it is discharged to the environment. In the second step, data of these five industrial plants about the water in- and outflows and the quality is gathered and used as an input for the tool. Step 3 shows an example of the desired output of the tool. This is a network where used water is reused in different industrial plants, potentially after regeneration, which results in an overall decrease in the water use. The tool created in this study is called 'water network design tool', and the model used within the tool is called 'water network design model'.

The study is structured into three parts. Primarily, requirements and boundaries for the model employed in the tool are defined, as shown in section 1.3. Subsequently, a literature study dedicated to reviewing existing water network design models is conducted. The literature study has two aims: the first aim is to discover shortcomings of existing water network design models; and the second aim is to find a foundational model for the water network design tool that is developed in the consecutive part. This part consists of developing a tool that identifies optimal connections to exchange water between different industrial plants (interplant [7]), while introducing water treatment systems where useful or needed. The following research questions are addressed to achieve this objective:

- What is currently lacking in water network design models?
- How can a tool facilitate the identification of interplant water exchanges, while taking water treatment systems into account?

1.3. Definitions & Project Boundaries

1.3.1. Pinch Analysis & Mathematical Optimization

There are generally two methodologies for the systematic design of a water network, that can be used within a water network design tool: water pinch analysis and mathematical optimization (MO) (Boix et al., 2015; Foo, 2012; Lawal et al., 2020; Yoo et al., 2007).

Water pinch analysis is an insight-based method to minimize the use of water from water sources, which also results in decreasing the amount of water that needs to be discharged. This is done by maximizing the reuse of water and using regeneration opportunities (Wang & Smith, 1994). Wang and

[7] the opposite of interplant is in-plant, meaning within one industrial plant

Smith introduced water pinch analysis in 1994. A review paper about water pinch analysis was written by Foo in 2009. Although Wang and Smith (1994) mentions that pinch analysis is applicable for both single constituent as well as multiple constituent cases, the method can only handle a single quality constraint at a time (Foo, 2012), so water pinch analysis becomes more difficult to use when the number of constituents increases (Yoo et al., 2007). Another disadvantage of water pinch analysis, is that it is not possible to deal with multi-objective optimization (Tiu & Cruz, 2017).

Mathematical optimization has the ability to handle multiple quality constraints at a time (Foo, 2012). More complex cases can be handled with mathematical optimization compared to water pinch analysis, such as the possibility to include constraints on the quality of water and on connections between industrial plants that are not allowed. However, an advantage of water pinch analysis over mathematical optimization, is that it provides more insights for designers (Foo, 2012). There are also examples that use hybrid approaches that include both water pinch analysis and mathematical optimization for the design of a water network[8].

In this research is chosen to use mathematical optimization, because it can handle multiple constituents simultaneously, and it is more flexible to include new functionalities in the tool when compared to water pinch analysis, especially when these functionalities are complex (Foo, 2012).

1.3.2. Water Reuse & Regeneration

The reuse or recycling of used water can be done either directly or after the treatment of the used water, a process also known as regeneration. The definitions as described by Foo (2015) and Klemes (2012) are used in this study. Recycling means that the water re-enters the same industrial process or plant as it came from and reuse means that the water is used in another industrial process or plant. Regeneration-reuse or regeneration-recycling means that the water is treated before it enters a water-using process again, in opposition to direct reuse or direct recycling. A graphical overview of these definitions is shown in figure 1.4.

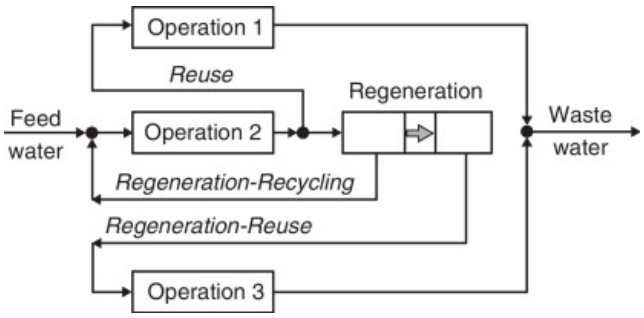


Figure 1.4: Potential water flows in an industrial plant considering three operations (industrial processes or plants): (direct) reuse, regeneration-recycling, and regeneration-reuse (Klemes, 2012)

This research focuses on incorporating water reuse, regeneration-reuse, and regeneration-recycling in the model. The objective is to develop a model that excludes direct recycling, as it often leads to the accumulation of constituents. Similarly, regeneration-recycling may result in constituent build-up if not effectively removed, so the possibility to exclude regeneration-recycling is desired as well. Excluding direct recycling and regeneration-recycling is an additional reason for the decision for choosing mathematical optimization over pinch analysis described in the previous section, as this is an example of a complex functionality that is to be included in the tool.

The removal of constituents from the water by regeneration, is often modelled by using either a fixed outlet concentration from the water treatment step or the removal ratio. The removal ratio is the fraction of a constituent that is removed from the water during water treatment. This value is calculated by dividing the mass of a specific constituent that is removed during water treatment by the total amount of mass of the constituent that entered the water treatment step.

[8]This is shown in the result of the literature study in appendix D

1.3.3. Optimization Objective

Optimization models can be categorized into minimizing and maximizing models. In the context of optimization in the design of water networks, a variable is usually minimized, namely fresh water use, wastewater production, water regeneration, costs, or a combination of multiple variables. The optimization objective is the variable that is optimized in the water network design model. In previously developed models, the optimization objective was either the sum of water inflows into the total water system or the sum of the costs. A disadvantage of optimizing solely based on water inflows is the potential of designing a total water system that proves excessively costly to construct and operate due to uneconomic water treatment systems. Consequently, it is chosen to optimize based on the total costs, containing at least the costs of the water that is used, and the costs of the water treatment systems.

1.3.4. Types of Mathematical Optimization

There are different mathematical optimization techniques, such as linear programming (LP), nonlinear programming (NLP), mixed-integer linear programming (MILP), and mixed-integer nonlinear programming (MINLP), as shown in figure 1.5. LP is a technique to find the best outcome in a mathematical model containing linear equations, optionally taking linear constraints into account. In NLP, nonlinear equations or nonlinear constraints are considered as well. In mixed-integer programming (MIP), there are variables that are both continuous and discrete[9]. In water network design models, binary variables (which are integer) are often used to in- or exclude certain connections, costs, allowable flow rates, limited piping connections, as well as safety, control and geographical constraints (Chew & Foo, 2009; Gunaratnam et al., 2005). (Nemati-Amirkolaii, 2021)

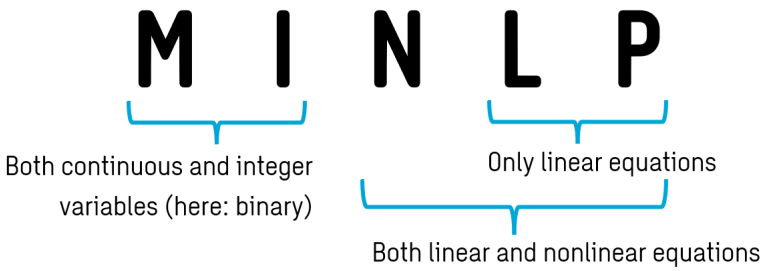


Figure 1.5: Explanation of different types of mathematical optimization problems: LP, NLP & MINLP

The function containing the optimization objective, the objective function, can be convex or non-convex. Convex means that the function has a global optimum and therefore has one optimal solution, and non-convex means that there are multiple local optimal points. Convex problems only contain convex functions and constraints, and are in general easier to solve than non-convex problems. The result of the model will depend on the initial conditions for non-convex problems. Therefore, in non-convex mathematical models a global optimum often cannot be guaranteed. LP problems are always non-convex, but NLP may be either convex or non-convex. Problems including binary variables are non-convex in general, but convex problems are possible.

In this research, it is desired to have a model that is flexible in a way that it is possible to in- and exclude certain connections to remove regeneration-reuse and regeneration-recycling for example. In addition, nonlinear constraints and nonlinear equations need to be handled, because of bilinear terms in mass balances and possibly nonlinear cost equations to estimate costs. Therefore, the model in this research will be a MINLP model, that is non-convex in most cases.

[9] In this study, variables are quantities that can change while using the tool, while parameters are fixed beforehand. Continuous variables are variables that can take on every value, while discrete variables can take on distinct, separate values. Integer variables are variables that can only take whole numbers, and binary variables are a special case of integer variables that can only take 0 or 1.

2

Methods

2.1. Literature Study

The literature study commenced by examining earlier literature reviews to discover deficiencies in water network design models. The literature review conducted by Jeżowski (2010) is the most relevant literature review for the design of water networks in industrial parks within the scope of this research. Jeżowski (2010) includes literature until 2009, so the literature study presented here focuses on developments after that period. There is one key difference: the literature study of Jeżowski (2010) focused on both pinch analysis and mathematical optimization methods and the literature study presented here only focuses on mathematical optimization methods as explained in the Introduction.

The methodology applied by Jeżowski (2010) is utilized in this study, although slightly adjusted to the aim of this research. Each paper is reviewed based on three sections, providing a comprehensive overview to find information:

- I (Scope), which provides information about what flows are included in the network design, such as whether it contains energy and material flows besides water flows, or whether it considers only one or multiple constituents in the model, or if water treatment systems are included. (Jeżowski, 2010)
- II (Method), which contains information about the type of model that is used, such as LP, MILP and/or MINLP, and what type of software is used to solve the model, and what type of objective function is used. If a paper uses a very specific method to solve the model, it is mentioned in this paragraph. (Jeżowski, 2010)
- ”III (Special remarks/features), which provides some important information that does not fall into sections I and II; the remarks can be those formulated in the original paper or by the author of this review.” (Jeżowski, 2010, p.1)

Furthermore, an additional review method is used to give a clearer overview of the content of each model, utilizing the approach shown in Behera et al. (2020). The method involves constructing a table wherein each row displays a different paper containing a model to be reviewed, and each column represents categories indicating whether certain elements are included in the paper. It is especially useful for the second goal of the literature review as described in section 1.2, namely to select a foundational model for this study, but the method of Behera et al. (2020) is also useful for the other aim: to identify shortcomings in existing water network design models. The papers were judged on the following elements and the closed questions related to each:

- Scientific article: Is the water network design model presented in a scientific article?
- Tool: Is the water network design model applied in a way that it is user-friendly?
- Software model: Is the water network described as a set of equations that are used to produce a software model?
- Water: Is the exchange of water considered in the model?
- Material: Is the exchange of materials considered in the model?
- Energy: Is the transfer of energy considered in the model?
- Water treatment: Are water treatment (also know as regeneration) systems considered in the model in order to improve the quality of the water for a specific use?
- Single constituent: Is only one constituent in the water considered in the water network design model?
- Multiple constituents: Are multiple constituents in the water considered in the water network design model?
- Centralized treatment: Is centralized treatment considered in the water network design model?
- Decentralized treatment: Are decentralized treatment systems considered in the water network design model?
- Pinch analysis: Is the model based on water pinch analysis?
- Mathematical optimization: Is the model based on mathematical optimization?
- Water minimization: Is the optimization (possibly) based on the minimization of water use?
- Economically minimized: Is the optimization (possibly) based on the minimization of costs?
- MINLP: Is the model a mixed-integer nonlinear programming model?
- Industry type/Case study specific: Is the model made for one specific type of industry or one specific case study?
- Site layout considered: Is the site layout considered in the study? Site layout includes the use of the location or distances between industrial plants.
- Case study/-ies: Is the model applied to one or more case studies?

2.2. Water Network Design Tool

2.2.1. Used Software & Model Concept

The optimization platform used for this tool is JupyterLab version 3.5.3. This is an open-source, web-based software that is possibly, but not necessarily, accessible via Anaconda Navigator. The programming language used to make the tool is Python version 3.10.9 and the code was run on an Intel(R) Core(TM) i7-8665U CPU @ 1.90GHz 2.11 GHz. The following Python packages are used in the tool:

- numpy 1.26.1
- pandas 1.5.3
- networkx 2.8.4
- graphviz 0.20.1
- amplpy 0.12.1 (AMPL & modules)
- matplotlib 3.8.2
- Pyomo 6.6.1 (Var, NonNegativeReals, ConcreteModel, Binary, Constraint, Reals, Objective, minimize, SolverFactory & value from pyomo.environ)

Last mentioned package, Pyomo, is used for the optimization modelling of the tool developed in this study. The following (convex) MINLP problem served as an example for this research to create the water network design model in Python.

$$(x-4)^2 - x \leq 50(1-y)$$
$$xlog(x) + 5 \leq 50y$$

Where, x is a continuous variable between 1 and 10, and y is a binary variable, which means that it can only take the values 0 or 1. The objective is to minimize x. The Python code that is used to solve this problem is shown below. In this example, the MindtPy solver is used, with glpk and ipopt as subsolvers, which are installed by using conda ("Getting Started with Pyomo", n.d.). The following lines of code are used to solve the problem and served as a concept to develop the tool in this research (Bernal & Peng, n.d.):

```
#Required imports
from pyomo.environ import *

#Create a simple model
model = ConcreteModel()

model.x = Var(bounds=(1.0,10.0),initialize=5.0)
model.y = Var(within=Binary)

model.c1 = Constraint(expr=(model.x-4.0)**2 - model.x <= 50.0*(1-model.y))
model.c2 = Constraint(expr=model.x*log(model.x)+5.0 <= 50.0*(model.y))

model.objective = Objective(expr=model.x, sense=minimize)

#Solve the model using MindtPy
SolverFactory('mindtpy').solve(model, mip_solver='glpk', nlp_solver='ipopt').write()
```

In the code shown above, a model is created with $model = ConcreteModel()$. Subsequently, the variables x and y with their boundaries are added to the model, and the equations are added to the model with the $Constraint$ option, which is followed by the addition of the objective. Lastly, the model is solved with the $SolverFactory().solve()$, where the used solver is defined within the first brackets. The output of this example is shown in appendix G, section G.1.

The solver used in the tool developed in this research is not MindtPy, but SCIP. SCIP is a non-commercial solver that can be used for solving non-convex MINLP (“SCIP”, n.d.), and is offered for free by a tool called AMPL[1]. AMPL connects commercial, but also open-source solvers like SCIP, that can be used in Pyomo after installation (“AMPL”, n.d.). Installing AMPL on your computer is possible at the following website https://portal.ampl.com/user/ampl/license/list and the installation of the AMPL modules for Python is explained here: https://dev.ampl.com/ampl/python/modules.html

The input data for the case studies is imported into Python by using a Microsoft Excel file. Microsoft Excel version 2302 was used in this study. This file contains eight tabs, wherein the data is placed. An example of the Excel file for the Petroleum Refinery Case Study presented in this study is shown in appendix E.

2.2.2. Design Process

The development process of the tool is roughly divided into three stages, as shown in figure 2.1. The first stage consists of defining the design concept of the tool, the second stage is the development of the model and subsequently the tool, and after that a never-ending loop starts of maintaining and upgrading the tool. The phases shown in the figure are inspired by the software development life cycle shown by Martyniuk (2022), but are applied in a different way to display the design process of this research.

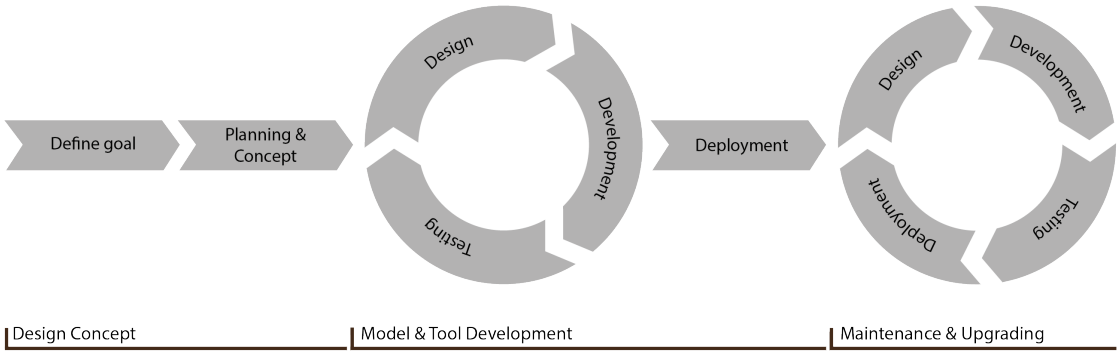


Figure 2.1: Design Stages of Water Network Design Tool Development. Stage 1 is the ’Design Concept’ stage, that consists of defining the goal, planning and concept. Stage 2, the ’Model & Tool Development’ stage consists of a loop of having a design idea, developing the idea and test this, respectively. The loop is followed by deployment of the tool. Stage 3, ’Maintenance & Upgrading’, consists of the same phases as stage 2, however, deployment is included in the loop.

[1]Short for ’A Mathematical Programming Language’

3

Literature Study

3.1. Definitions in Literature

In this research, the focus is on the reuse of water, potentially after regeneration. The exchange of water between different firms on a larger scale leads to a water network (WN). In literature multiple synonyms are used across different papers for this term, such as: inter-plant/interplant water integration (Chew & Foo, 2009; Rubio-Castro et al., 2013), inter-plant/interplant water network (Chen et al., 2010; X. Wang et al., 2019), total site water integration, total site water reuse (Fadzil, Alwi, Manan, et al., 2018), total water system (Gunaratnam et al., 2005), water allocation network (De-León Almaraz et al., 2016; Zhou et al., 2015), water conservation network (Sotelo-Pichardo et al., 2011), water exchange network (Aviso, 2014; Aviso et al., 2010), water integration (Rubio-Castro et al., 2010), water network (de Faria et al., 2009; Foo, 2009; Hong et al., 2018; Jeżowski, 2010; Khor et al., 2014; Liao et al., 2007), water reuse network (Yoo et al., 2007), water-using network (Li et al., 2015; Z. Y. Liu et al., 2009; Pan et al., 2012; Su et al., 2012), and water-using system (Alva-Argáez et al., 2007).

More broadly, terms 7such as (interplant) resource conservation network (RCN) (Chew, Foo, Ng, et al., 2010; Foo, 2012; Ng et al., 2009), mass exchange network (Short et al., 2018) or process integration (Foo, 2012) are used to describe the exchange of resources between different industries. In literature, exchanges not only involve water networks, but also the transfer of heat and hydrogen are frequently described. An industrial park where materials, energy and water are exchanged to reduce waste and pollution, leading to efficient resource sharing, is called an eco-industrial park.

The composition of water, containing various constituents, poses challenges in the exchange of water. The term 'constituent' is often interchangeably used with 'contaminant' or 'impurity'. The term 'constituent' is preferred in this research for its neutrality. Unlike 'contaminant' and 'impurity', which suggest a waste connotation, 'constituent' is in line with a circular economy, where constituents are viewed as resources available for reuse.

3.2. Scope of Literature Study

As explained in section 1.2, the literature research in this study has two aims. The first one is to identify shortcomings in existing water network design models. The second aim of the literature research is to find a foundational model to build on further in order to complete one or more of the identified shortcomings in the tool developed in this study.

This literature research can be considered as a continuation or extension of the literature research of Jeżowski (2010). That research includes papers until late-to-mid 2009, so the literature review in this study will continue with papers written in 2010 and later, as well as the papers written in 2009 that were not considered by Jeżowski (2010). However, there is one key difference in the literature research of Jeżowski (2010) and the one presented in this paper, namely that the literature research in this paper focuses on mathematical optimization models for the design of water networks. Consequently, the

papers that are solely based on pinch analysis are not included in the literature review in this study.

In the first stages of the research done in this paper, however, some papers from before 2010 and papers that are solely based on pinch analysis were reviewed in the method used by Behera et al. (2020). These are shown in the table in appendix D, in addition to the papers discussed more thoroughly in the next section according to the method of Jeżowski (2010).

3.3. Literature Annotations

In this section, studies on the design of water networks using mathematical optimization are discussed in alphabetical order. In every literature annotation, the complete citation of the study is shown, followed by a review based on three sections according to the method of Jeżowski (2010), as explained in section 2.1. The results of the studies shown in this section are included in the table in appendix D, which contains a summarized overview of each study.

1. Abraham, E. J., Al-Mohannadi, D. M., & Linke, P. (2022). Resource integration of industrial parks over time. *Computers and Chemical Engineering*, *164*. https://doi.org/10.1016/j.compchemeng.2022.107886

   I: RCN design; (very basic) regeneration unit included; single constituent; multiple resources; resource and cash flows included

   II: MILP; Microsoft Excel: LINDO "Whats'sBest! 16.0.2.6" solver; NPV (next present value) objective function; multi-period resource integration approach

   III: Water, materials and energy considered; case study specific; performance over time considered; carbon capture utilization and storage included

2. Ahmed, R., Shehab, S., Al-Mohannadi, D. M., & Linke, P. (2020). Synthesis of integrated processing clusters. *Chemical Engineering Science*, *227*. https://doi.org/10.1016/j.ces.2020.115922

   I: RCN design; (very basic) regeneration unit included; single constituent

   II: MILP; Microsoft Excel: LINDO "Whats'sBest! 16.0.2.2" solver; annual gross profit objective function

   III: Water, materials and energy considered; generic method; CO_2 utilization and sequestration; processing clusters identification; life-cycle analysis included

3. Alnouri, S. Y., Linke, P., & El-Halwagi, M. M. (2016). Synthesis of industrial park water reuse networks considering treatment systems and merged connectivity options. *Computers and Chemical Engineering*, *91*, 289–306. https://doi.org/10.1016/j.compchemeng.2016.02.003

   General remark: this is an extension of Annotation 17 (Alnouri et al., 2014)

   I: WN design; regeneration units included; multiple constituents

   II: MINLP model; Microsoft Excel: LINDO "What's Best 9.0.5.0"; total annualized cost minimization

   III: Centralized and decentralized treatment options considered; merging strategies for common pipe segments considered; considers infrastructure layout map

4. Aviso, K. B., Tan, R. R., & Culaba, A. B. (2010). Designing eco-industrial water exchange networks using fuzzy mathematical programming. *Clean Technologies and Environmental Policy*, *12*, 353–363. https://doi.org/10.1007/s10098-009-0252-1

   I: WN design; regeneration units included; multiple constituents

   II: NLP model; Microsoft Excel: LINDO "LINGO 11.0"; fuzzy mathematical optimization; maximize level of satisfaction achieved by each company

   III: Fuzzy mathematical optimization used for considering individual benefits, maximize satisfaction of the least satisfied participant

5. Aviso, K. B. (2014). Design of robust water exchange networks for eco-industrial symbiosis. *Process Safety and Environmental Protection*, *92*, 160–170. https://doi.org/10.1016/j.psep.2012.12.001

I: WN design; no regeneration units included; multiple constituents

II: LP model; Microsoft Excel: LINDO ”LINGO 12.0”; freshwater consumption minimization

III: Method to obtain near optimal solution that is robust for potential changes, considering multiple scenarios

6. Behera, C. R., Al, R., Gernaey, K. V., & Sin, G. (2020). A process synthesis tool for wwtp – an application to design sustainable energy recovery facilities. *Chemical Engineering Research and Design*, *156*, 353–370. https://doi.org/10.1016/j.cherd.2020.02.014

   I: Wastewater treatment plant design; regeneration units included; multiple constituents

   II: Monte Carlo based optimization; MATLAB: Simulink ; multi-objective optimization (total annualized cost, maximizing net energy recovery & minimizing greenhouse gas emissions)

   III: Process synthesis tool: SPDLab; dimensioning of technologies considered; plant layout generation considered

7. Bishnu, S. K., Linke, P., Alnouri, S. Y., & El-Halwagi, M. (2014). Multiperiod planning of optimal industrial city direct water reuse networks. *Industrial and Engineering Chemistry Research*, *53*, 8844–8865. https://doi.org/10.1021/ie5008932

   I: WN design; no regeneration units included; multiple constituents

   II: MINLP model; Microsoft Excel: LINDO ”What’s Best 9.0!”; freshwater minimization; multiperiod model for cost optimization

   III: Industrial city considering layout changes in time

8. Boix, M., Montastruc, L., Pibouleau, L., Azzaro-Pantel, C., & Domenech, S. (2011). A multiobjective optimization framework for multicontaminant industrial water network design. *Journal of Environmental Management*, *92*, 1802–1808. https://doi.org/10.1016/j.jenvman.2011.02.016

   I: WN design; regeneration units included; multiple constituents

   II: MINLP model; GAMS: COIN-BONMIN 0.9 optimizer; multi-objective optimization framework (GEC (global equivalent cost) contains: freshwater flow rate at network entrance, water flow rate at inlet of regeneration units & number of interconnections)

   III: Based on maximum inlet and outlet concentrations of regeneration units; multiple criteria decision making

9. Boix, M., Montastruc, L., Pibouleau, L., Azzaro-Pantel, C., & Domenech, S. (2012). Industrial water management by multiobjective optimization: From individual to collective solution through eco-industrial parks. *Journal of Cleaner Production*, *22*, 85–97. https://doi.org/10.1016/j.jclepro.2011.09.011

   I: WN design; regeneration units included; single constituent

   II: MILP model; GAMS: CPLEX 11.2.1; bi-objective optimization (fresh water flow at the network entrance & water flow rate at inlets of regeneration units, while constant number of connections)

   III: Regeneration units defined by outlet concentration

10. Boix, M., Négny, S., Montastruc, L., & Mousqué, F. (2023). Flexible networks to promote the development of industrial symbioses: A new optimization procedure. *Computers and Chemical Engineering*, *169*. https://doi.org/10.1016/j.compchemeng.2022.108082

    I: WN design; no regeneration units included; no constituents considered

    II: MILP model; ILOG: CPLEX solver; net present cost minimized; single and two-step approach

    III: Flexibility included in the model: minimum of maximal deviations for the nominal inlet parameters in the network; piping length considered

11. Chen, C. L., Hung, S. W., & Lee, J. Y. (2010). Design of inter-plant water network with central and decentralized water mains. *Computers and Chemical Engineering*, *34*, 1522–1531. https://doi.org/10.1016/j.compchemeng.2010.02.024

    I: WN design; regeneration units included; multiple constituents

II: MINLP model; GAMS: DICOPT; cost minimization

III: Use of central and decentralized water mains; considers storage tanks; optimization based on fresh water minimization or total annualized cost minimization); piping length and pumping cost considered

12. Chew, I. M. L., & Foo, D. C. Y. (2009). Automated targeting for inter-plant water integration. *Chemical Engineering Journal*, *153*, 23–36. https://doi.org/10.1016/j.cej.2009.05.026

    I: WN design; central regeneration included; single constituent

    II: Both insight-based and optimization based (hybrid approach); minimum flow rate or minimum cost optimization

13. Chew, I. M. L., Foo, D. C. Y., Bonhivers, J. C., Stuart, P., Alva-Argaez, A., & Savulescu, L. E. (2013). A model-based approach for simultaneous water and energy reduction in a pulp and paper mill. *Applied Thermal Engineering*, *51*, 393–400. https://doi.org/10.1016/j.applthermaleng.2012.08.070

    I: WN design; no regeneration considered; single contaminant

    II: Both insight-based and optimization based (hybrid approach); MINLP model; LINGO Global solver; total operating cost or total annualized cost minimization

    III: Both water and energy considered; specific for pulp and paper mill; sensitivity analysis included; especially useful for in-plant reuse/recycling due to complexity

14. Chin, H. H., Varbanov, P. S., Klemeš, J. J., & Tan, R. R. (2022). Accounting for regional water recyclability or scarcity using machine learning and pinch analysis. *Journal of Cleaner Production*, *368*. https://doi.org/10.1016/j.jclepro.2022.133260

    I: WN design; no regeneration considered; multiple constituents

    II: Machine learning & pinch analysis combined; clustering or classification of water quality

    III: Includes artificial intelligence in the design of a water network

15. De-León Almaraz, S., Boix, M., Azzaro-Pantel, C., Montastruc, L., & Domenech, S. (2015). Design of a multi-contaminant water allocation network using multi-objective optimization. *Computer Aided Chemical Engineering*, *37*, 911–916. https://doi.org/10.1016/B978-0-444-63578-5.50147-X

    General remark: NLP follows the method of Feng et al. (2008) and Boix et al. (2011)

    I: WN design; regeneration units included; multiple constituents

    II: NLP & MINLP; GAMS: ”IPOPT” solver; multi-objective optimization (GEC (global equivalent cost) & flow rates); MINLP solved by using lexicographic and ϵ-constraints methods

    III: Use of limiting pollutant; fixed maximal input and output concentrations for each constituent and each treatment unit

16. De-León Almaraz, S., Boix, M., Montastruc, L., Azzaro-Pantel, C., Liao, Z., & Domenech, S. (2016). Design of a water allocation and energy network for multi-contaminant problems using multi-objective optimization. *Process Safety and Environmental Protection*, *103*, 348–364. https://doi.org/10.1016/j.psep.2016.03.015

    General remark: this is an extension of Annotation 15 (De-León Almaraz et al., 2015) to include HEN

    I: WN & HEN design; regeneration units included; multiple constituents

    II: Two step approach (WAN followed by HEN); WN: multi-objective optimization (GEC (global equivalent cost) & number of network connections) by using lexicographic and ϵ-constraints methods; HEN: pinch analysis (energy minimization) or mathematical programming (combined with WAN)

    III: Water and energy considered; multi-criteria problem solved with □-constraint method; HEN solved by pinch analysis or mathematical programming; fixed maximal input and output concentrations for each constituent and each treatment unit

17. Alnouri, S. Y., Linke, P., & El-Halwagi, M. (2014). Water integration in industrial zones: A spatial representation with direct recycle applications. *Clean Technologies and Environmental Policy*, *16*, 1637–1659. https://doi.org/10.1007/s10098-014-0739-2

    I: WN design; no regeneration units included; multiple constituents

    II: NLP model; Microsoft Excel: LINDO ”What’s Best 9.0.5.0”; two objectives: (i) freshwater consumption and wastewater discharge minimized (ii) piping and freshwater costs minimized

    III: Water networks within an industrial city plot; infrastructure layout mapped

18. Foong, S. Z., & Ng, D. K. (2021). Simultaneous design and integration of multiple processes for eco-industrial park development. *Journal of Cleaner Production*, *298*. https://doi.org/10.1016/j.jclepro.2021.126797

    I: RCN design; centralized regeneration considered; no constituents

    II: MILP; Microsoft Excel: LINDO ”LINGO 16.0”; economic performance maximized

    III: Considers water, materials & energy; includes feasible operation range analysis (FORA);

19. Fouladi, J., AlNouss, A., & Al-Ansari, T. (2021). Optimising the sustainability performance of an industrial park: An energy-water-food nexus. *Computer Aided Chemical Engineering*, *50*, 1505–1510. https://doi.org/10.1016/B978-0-323-88506-5.50232-1

    General remarK: expansion of Alnouri et al. (2014)

    I: RCN design; regeneration considered; multiple constituents

    II: Microsoft Excel: LINDO ”What’s Best” solver; multi-objective optimization; total annual cost minimized

    III: Energy-water-food nexus; exergetic approach; coupled to Aspen chemical process simulator

20. Hong, X., Liao, Z., Sun, J., Jiang, B., Wang, J., & Yang, Y. (2018). Energy and water management for industrial large-scale water networks: A systematic simultaneous optimization approach. *ACS Sustainable Chemistry and Engineering*, *6*, 2269–2282. https://doi.org/10.1021/acssuschemeng.7b03740

    I: HIWN (heat integrated water network) design; regeneration considered; multiple constituents

    II: MINLP model; GAMS; COINIPOPT solver (NLP) & DICOPT/CONOPT/CPLEX solver (MINLP); total annual cost minimization

    III: Considers water & energy; three step solution approach

21. Huang, X., Luo, X., Chen, J., Yang, Z., Chen, Y., Ponce-Ortega, J. M., & El-Halwagi, M. M. (2018). Synthesis and dual-objective optimization of industrial combined heat and power plants compromising the water-energy nexus. *Applied Energy*, *224*, 448–468. https://doi.org/10.1016/j.apenergy.2018.04.095

    I: RTCHP (RO–TMD coupling water desalination method); regeneration considered; single constituent

    II: Dual objective NLP model; GAMS; BARON solver

    III: Water-energy system; use of Rankine cycle; Pareto optimal

22. Le, T. M., Kujawa-Roeleveld, K., Tran, D. T., & Rijnaarts, H. H. (2022). Data envelopment analysis as a tool to assess the water demand minimization potential in industrial zones in the vietnamese delta. *Water Resources and Industry*, *28*. https://doi.org/10.1016/j.wri.2022.100181

    I: Water reduction; regeneration considered; multiple constituents

    II: LP; CCR (Charnes-Cooper-Rhodes) & BCC (Banker-Charnes-Cooper)

    III: Data envelopment analysis; specific for Vietnamese industry

23. Leong, Y. T., Tan, R. R., & Chew, I. M. L. (2014). Optimization of chilled and cooling water systems in a centralized utility hub. *Energy Procedia*, *61*, 846–849. https://doi.org/10.1016/j.egypro.2014.11.979

    I: HEN (heat exchange network), more specific: CCWS (chilled cooling water system); no regeneration considered; no constituents considered

    II: -

    III: Sensitivity analysis included

24. Liao, Z., Rong, G., Wang, J., & Yang, Y. (2011). Systematic optimization of heat-integrated water allocation networks. *Industrial and Engineering Chemistry Research*, *50*, 6713–6727. https://doi.org/10.1021/ie1016392

    I: WAHEN (water allocation heat exchange network); regeneration considered; single constituent

    II: MILP & MINLP model; GAMS; CPLEX & DICOPT solver (MILP & MINLP); total annual cost minimization

25. Lim, S. R., & Park, J. M. (2010). Interfactory and intrafactory water network system to remodel a conventional industrial park to a green eco-industrial park. *Industrial and Engineering Chemistry Research*, *49*, 1351–1358. https://doi.org/10.1021/ie9014233

    I: WN design; inter- and intrafactory; EIPWNS (eco-industrial park water network system);

    II: NLP; GAMS; MINOS solver; multiobjective optimization (total annualized costs & environmental impact)

    III: LCA (life cycle assessment) included

26. Nemati-Amirkolaii, K., Romdhana, H., & Lameloise, M. L. (2021). A novel user-friendly tool for minimizing water use in processing industry. *Cleaner Engineering and Technology*, *4*. https://doi.org/10.1016/j.clet.2021.100260

    I: WN model; no regeneration; multiple constituents

    II: MINLP model; GAMS & Python; water minimization

    III: Tool; GEC (global equivalent cost); TOPSIS

27. Ng, D. K. S., Foo, D. C. Y., & Tan, R. (2009). Automated targeting technique for single-impurity resource conservation networks. part 1: Direct reuse/recycle. *Industrial and Engineering Chemistry Research*, *48*, 7637–7646. https://doi.org/10.1021/ie900120y

    I: RCN design; no regeneration; single constituent

    II: LP model; Microsoft Excel: LINDO ”LINGO 10.0”; minimization of flow rate or costs

    III: Water & materials; applicable to hydrogen networks; automated pinch analysis

28. Ng, D. K. S., Chew, I. M. L., Tan, R. R., Foo, D. C. Y., Ooi, M. B., & El-Halwagi, M. M. (2014). Rcnet: An optimisation software for the synthesis of resource conservation networks. *Process Safety and Environmental Protection*, *92*, 917–928. https://doi.org/10.1016/j.psep.2013.10.006

    I: RCN design; no regeneration; single constituent

    II: LP model; Microsoft Excel: LINDO ”What’s Best” solver; maximize resource conservation

    III: Spreadsheet-based software tool (RCNet); based on pinch analysis & mathematical programming

29. O’Dwyer, E., Chen, K., Wang, H., Wang, A., Shah, N., & Guo, M. (2020). Optimisation of wastewater treatment strategies in eco-industrial parks: Technology, location and transport. *Chemical Engineering Journal*, *381*. https://doi.org/10.1016/j.cej.2019.122643

    I: Whole-system design; regeneration considered; multiple constituents

    II: MILP model; GAMS; CPLEX solver; minimization of total costs

    III: Water & materials considered (phosphorous, nitrogen & sludge); site layout considered

30. Pan, C., Shi, J., & Liu, Z. Y. (2012). An iterative method for design of water-using networks with regeneration recycling. *AIChE Journal*, *58*, 456–465. https://doi.org/10.1002/aic.12595

    General remark: expansion of Z. Y. Liu et al. (2009)

    I: WN design; regeneration considered; multiple constituents

    II: Iterative method; Microsoft Excel; minimize fresh water consumption

    III: Treatment-subnetwork designed before total water network

31. Poplewski, G., Jezowski, J. M., & Jezowska, A. (2011). Water network design with stochastic optimization approach. *Chemical Engineering Research and Design*, *89*, 2085–2101. https://doi.org/10.1016/j.cherd.2010.12.016

    I: WN design; regeneration considered; multiple constituents

    II: MINLP & NLP model; GAMS: DICOPT solver; minimize fresh water consumption

    III: Meta-heuristic optimization or adaptive random search (ARS)

32. Rubio-Castro, E., Ponce-Ortega, J. M., Nápoles-Rivera, F., El-Halwagi, M. M., Serna-González, M., & Jiménez-Gutiérrez, A. (2010). Water integration of eco-industrial parks using a global optimization approach. *Industrial and Engineering Chemistry Research*, *49*, 9945–9960. https://doi.org/10.1021/ie100762u

    I: WN design; regeneration units included; single constituent

    II: MINLP & MILP model; GAMS: DICOPT solver (for MINLP) and CPLEX solver (for MILP); total annual cost minimization; new discretization approach used

    III: Removal ratio used to describe regeneration effectiveness; piping length considered

33. Rubio-Castro, E., Ponce-Ortega, J. M., Serna-González, M., Jiménez-Gutiérrez, A., & El-Halwagi, M. M. (2011). A global optimal formulation for the water integration in eco-industrial parks considering multiple pollutants. *Computers and Chemical Engineering*, *35*, 1558–1574. https://doi.org/10.1016/j.compchemeng.2011.03.010

    I: WN design; regeneration units included; multiple constituents

    II: MINLP model; GAMS: CPLEX solver; total annual cost minimization

    III: Both in-plant and inter-plant modifications; replacement of treatment units, shared treatment units; piping length considered

34. Rubio-Castro, E., Ponce-Ortega, J. M., Serna-González, M., & El-Halwagi, M. M. (2012). Optimal reconfiguration of multi-plant water networks into an eco-industrial park. *Computers and Chemical Engineering*, *44*, 58–83. https://doi.org/10.1016/j.compchemeng.2012.05.004

    General remark: The model of Rubio-Castro et al. (2010) is used partly for this model

    I: WN design; regeneration units included; multiple constituents

    II: MINLP model; GAMS: DICOPT/CONOPT solver (second and third, MINLP model) and CPLEX solver (first, linear model); total annual cost minimization

    III: Both in-plant and inter-plant modifications; three step approach; both economic and environmental benefits considered; considers power consumption of pumps in costs; piping length considered

35. Rubio-Castro, E., Ponce-Ortega, J. M., Serna-González, M., El-Halwagi, M. M., & Pham, V. (2013). Global optimization in property-based interplant water integration. *AIChE Journal*, *59*, 813–833. https://doi.org/10.1002/aic.13874

    I: WN design; regeneration units included; multiple constituents

    II: MINLP model; GAMS: CPLEX solver (relaxation of MINLP: MILP); total annual cost minimization

    III: Both in-plant and inter-plant modifications; Convex relaxation of MINLP method for global optimization by using subdomains; properties of streams integration (e.g. pH, density, viscosity, toxicity, and color); piping length considered

36. Short, M., Isafiade, A. J., Biegler, L. T., & Kravanja, Z. (2018). Synthesis of mass exchanger networks in a two-step hybrid optimization strategy. *Chemical Engineering Science*, *178*, 118–135. https://doi.org/10.1016/j.ces.2017.12.019

    I: Mass exchange network design (for packed columns); no regeneration; single constituent

    II: MINLP, MILP & NLP; GAMS; DICOPT solver (MINLP), CPLEX solver (MILP) & CONOPT solver (NLP)

    III: Two-step hybrid optimization strategy; considers Reynolds number

37. Sotelo-Pichardo, C., Ponce-Ortega, J. M., El-Halwagi, M. M., & Frausto-Hernández, S. (2011). Optimal retrofit of water conservation networks. *Journal of Cleaner Production*, *19*, 1560–1581. https://doi.org/10.1016/j.jclepro.2011.05.011

    I: WN retrofit; regeneration considered; single constituent

    II: MINLP model; BARON solver; minimize total annual costs

    III: Considers already existing & new treatment systems

38. Su, W. N., Li, Q. H., Liu, Z. Y., & Pan, C. H. (2012). A new design method for water-using network of multiple contaminants with single internal water main. *Journal of Cleaner Production*, *29-30*, 38–45. https://doi.org/10.1016/j.jclepro.2012.01.041

    General remark: uses concept of Z. Y. Liu et al. (2009)

    I: WN design; no regeneration; multiple constituents

    II: CPD (Concentration potential of the demands) method; minimize water use

    III: Single internal water main

39. Tiu, B. T. C., & Cruz, D. E. (2017). An milp model for optimizing water exchanges in eco-industrial parks considering water quality. *Resources, Conservation and Recycling*, *119*, 89–96. https://doi.org/10.1016/j.resconrec.2016.06.005

    I: WN design; regeneration considered; single constituent

    II: MILP model; GAMS; multi-objective optimization; minimizes economic cost and environmental impact

    III: Energy & water considered

40. Tokos, H., & Pintarič, Z. N. (2012). Development of a minlp model for the optimization of a large industrial water system. *Optimization and Engineering*, *13*, 625–662. https://doi.org/10.1007/s11081-011-9162-2

    I: WN design; regeneration considered; single constituent

    II: MINLP model; GAMS; DICOPT solver; minimize total annual costs

    III: Model specific for production and packaging area of breweries

41. Wang, X., Fan, X., & Liu, Z. Y. (2019). Design of interplant water network of multiple contaminants with an interplant water main. *Chemical Engineering Transactions*, *72*, 295–300. https://doi.org/10.3303/CET1972050

    General remark: uses concept of Z. Y. Liu et al. (2009)

    I: WN design; no regeneration; multiple constituents

    II: CPD method (iterative method); minimize freshwater consumption

    III: Internal water main

42. Zhou, L., Liao, Z., Wang, J., Jiang, B., Yang, Y., & Yu, H. (2015). Simultaneous optimization of heat-integrated water allocation networks using the mathematical model with equilibrium constraints strategy. *Industrial and Engineering Chemistry Research*, *54*, 3355–3366. https://doi.org/10.1021/ie501960e

    I: WAHEN (water allocation and heat exchange network); regeneration considered; single constituent

II: MINLP, NLP & MPEC (mathematical model with equilibrium constraints) model; GAMS; NLPEC solver (MINLP), CONOPT solver (NLP) & DICOPT solver (MINLP); minimize network costs

III: Equilibrium constraints strategy

3.4. Literature Study Analysis

The literature research of Boix et al. (2015) about optimization methods for the design of eco-industrial parks, revealed several shortcomings. One of the shortcomings discovered was the lack of multi-objective optimization, which was studied by Boix et al. (2011), but was later also applied by De-León Almaraz et al. (2015), De-León Almaraz et al. (2016), Tiu and Cruz (2017), and 2021 (2021). Multi-objective optimization can be about optimizing environmental, economic and social aspects simultaneously. Some papers optimize based on the global equivalent cost (Boix et al., 2011; De-León Almaraz et al., 2016; De-León Almaraz et al., 2015; Feng et al., 2008; Nemati-Amirkolaii et al., 2021) that include multiple criteria such as the sum of water retrieved from natural resources, sum of water inflow rates at water treatment systems, number of connections between industrial plants, and some take both environmental and economic aspects into account. Environmental impacts can be estimated via a life cycle assessment, where various impact categories can be calculated and minimized in multi-objective optimization, such as greenhouse gas emissions, eutrophication, land use change, water footprint, fossil fuel use, and solid waste production (Mu et al., 2019). Another application of multi-objective optimization, is by optimizing the needs of the different plants as separate objectives. In most tools, the total water network is optimized at a whole, but the advantages and disadvantages for the creation of such a network for the industrial plants are not considered separately. The social aspects mentioned earlier are not taken into account often. The only example found is by Hipólito-Valencia et al. (2014) who included the generation of jobs as an objective (Boix et al., 2015).

Boix et al. (2015), Jeżowski (2010), and Duhbacı et al. (2021) mentioned that there is a lack of models that optimize water and a heat network simultaneously. The table in appendix D shows that there are models that include both water and energy (Abraham et al., 2022; Ahmed et al., 2020; Behera et al., 2020; Boix et al., 2023; Chew et al., 2013; De-León Almaraz et al., 2016; Foong & Ng, 2021; Fouladi et al., 2021; Hong et al., 2018; Huang et al., 2018; Leong et al., 2014; Liao et al., 2011; C. Liu et al., 2015). However, when this is narrowed down to water network design models that consider both multiple constituents and water treatment systems, only the model of Hong et al. (2018) is left.

The inclusion of materials in water network design models is sometimes applied. When the optimization of water, energy, and materials are all considered in an optimization model, this could lead to the full optimization of eco-industrial parks. There are a few papers that consider all three in a model, namely Abraham et al. (2022), Ahmed et al. (2020), Foong and Ng (2021), Fouladi et al. (2021), and C. Liu et al. (2015). One paper that is noteworthy for combining water and materials in a model, is the one of O'Dwyer et al. (2020). This paper has a special focus on the recovery of materials from water.

One other shortcoming mentioned by Boix et al. (2015), are models that take flexibility of eco-industrial parks into account. One model that takes flexibility into account, is the one of Montastruc et al. (2013) (Boix et al., 2015). The number of connections of in the eco-industrial park is included in the multi-objective optimization of this model.

Capelleveen et al. (2017) wrote about 'information systems facilitating the identification of industrial symbiosis'. They recommended a few directions for identification tools for industrial symbiosis. One of the recommendations is to develop more product and service oriented software. The tools are often custom made instead of software that is accessible for everyone. Capelleveen et al. (2017) mentioned that the software made for a specific purpose can often easily be applied in different settings.

Another recommendation is to develop one platform that links data, industrial symbiosis cases, and IS tools (Capelleveen et al., 2017). There are multiple platforms that serve as a 'marketplace' that brings the supply and demand of water together, such as De Waterbank (2023), Water Europe Marketplace (n.d.), Symbiosis4growth (2023), Sharebox (n.d.), IS Data (n.d.), and International Syngergies (n.d.). However, these platforms usually do not link data with water network design tools.

Some of the reviewed papers created a tool from their water network design model. One of them is Ng et al. (2014), who created a tool called RCNet. This spreadsheet-based tool takes both pinch analysis and

mathematical programming into consideration, but only works for a single water constituent. Another one is Short et al. (2018), who created MExNets, a Python-based model specifically for packed columns and is currently under development. GAMS software is needed for full functionality of the tool (Short, n.d.). Nemati-Amirkolaii et al. (2021) made a tool in Python to reuse water with multiple constituents, but it does not include water treatment systems.

Most models were optimized by using GAMS software (GAMS Software GmbH, n.d.) or LINGO (LINDO Systems Inc., 2016), as shown in the literature annotations in the previous section. These are both commercial models and require payment before they can be used. There is a lack in open-source models and tools that design water networks while considering multiple constituents and treatment systems, as visible in table 3.1.

Yeo et al. (2019) proposed to use digitized knowledge, coupled with machine learning algorithms to seek new industrial symbiosis connections. This adoption of intelligent learning techniques is also mentioned by Capelleveen et al. (2017). The amount of databases are growing and these could be used to assist in finding new possibilities to exchange water between different plants. The use of databases in finding industrial symbiosis connections is not applied in any of the reviewed models.

The papers reviewed in the previous section, were also judged based on whether the site layout of the industrial area was considered in the water network design model. The site layout is relevant for estimating the costs related to creating the network. The price of piping increases with the distance between different industrial plants and the availability of natural resources, including water resources, has an influence on the price of resources. Site layout was considered in many papers, but to different extents. Some papers only considered the distance between plants, while others included the spatial design of whole industrial parks.

The lack of mathematical models for large-scale problems considering both heat networks and water networks with regeneration-reuse and multiple constituents, is mentioned by Ibrić et al. (2022). This is also visible in the table in appendix D, and in table 3.1 in the next section.

The review paper of Jeżowski (2010) highlighted some other shortcomings, namely 'batch-wise water networks, uncertain data and disturbances, interplant integration, and mechanisms of increasing possibilities of locating a global or 'good' local optimum'. In water network design models, it is often assumed that the plants run continuously, and that the concentrations of the constituents are constant. The batch-wise operation of plants is not considered, neither is storage. The models reviewed in this paper also take these assumptions, so this is still a knowledge gap that is also mentioned later by Lawal et al. (2020). The second knowledge gap discovered by (Jeżowski, 2010) (2010), 'uncertain data and disturbances', is about how networks react under disturbances, such as controllability issues, seasonal changes, or the impact of malfunction or closing of a plant. These are important factors to consider, in order to build a resilient and flexible network, something that is also addressed by Boix et al. (2015). A lack of models that consider interplant integration is mentioned by (Jeżowski, 2010) (2010) as shown above, but this gap has been filled, since a lot of models reviewed in this paper consider interplant integration. However, one issue described is not solved, namely the assumption that the different plants can work perfectly together and that all information is available. The last gap that was mentioned is that improvements are needed on finding global or 'good' local optimums in the optimization of the water network design, especially in combination with heat networks.

3.5. Selection of Foundational Water Network Design Model

The selection of the foundational model for this research was done by using the table shown in appendix D, which was made in Microsoft Excel. As described in section 1.2, the model should be a software-based model incorporating decentralized treatment systems, multiple constituents, and cost optimization that includes at least water and water treatment costs. It is desired to develop a MINLP model, since this is the most flexible option for accommodating new functionalities, so this restriction is also taken into account. To ensure reproducibility of the model, the equations should be given and described extensively for understanding. This is often only the case in scientific articles, so this restriction is added as well. Table 3.1 shows the models from the table shown in appendix D after applying aforementioned restrictions.

Table 3.1: Models found in literature that are suitable to function as a foundational model in this study

Source	Scientific article Software model Decentralized water treatment Multiple constituents Mathematical optimization Economically minimized MINLP	Tool	Materials	Energy	Pinch analysis	Data included
Chen et al., 2010	✓	-	-	-	-	-
Gunaratnam et al., 2005	✓	-	-	-	-	-
Hong et al., 2018	✓	-	-	✓	-	-
Poplewski et al., 2011	✓	-	-	-	-	-
Rubio-Castro et al., 2011	✓	-	-	-	-	-
Rubio-Castro et al., 2012	✓	-	-	-	-	-
Rubio-Castro et al., 2013	✓	-	-	-	-	-
This study	✓	✓	**-**	**-**	**-**	✓

The models from Hong et al. (2018), Rubio-Castro et al. (2011), Rubio-Castro et al. (2012), and Rubio-Castro et al. (2013) were not chosen due to the complexity of the models that is not necessary for the scope of this study. Hong et al. (2018) included heat integration, which increases the complexity of the model, and the other three studies included both interplant and in-plant water reuse, wheras the focus of this study is only on interplant water reuse. The model of Chen et al. (2010) included water mains in the model and that is not desired for the model in this study.

The specifications of the two remaining options, Gunaratnam et al. (2005) and Poplewski et al. (2011), are both suitable. Gunaratnam et al. (2005) uses a more simple approach for the optimization procedure and includes the distances between different industrial plants for the optimization of the costs, which is the reason that this model is preferred over Poplewski et al. (2011). Additionally, Poplewski et al. (2011) does not consider environmental discharge limits and always includes piping costs, while Gunaratnam et al. (2005) presents the opportunity to in- or exclude both.

4

Water Network Design Model: Gunaratnam et al. (2005)

4.1. Model Concept Explanation

The model presented in the paper of Gunaratnam et al. (2005) provides an automated method to design water networks, using mass balances and both linear and nonlinear constraints. The model contains both continuous and integer variables, which makes the model a mixed-integer nonlinear programming (MINLP) model. The model couples water sources (supply) to water sinks (demand) and introduces water treatment systems to meet quality standards for reuse or discharge. The objective function is to minimize the costs of the total water network, that consist of the costs of the water from water sources, the treatment costs, and optionally the costs of the pipe connections. The sets, variables, parameters, equations, and constraints that are used in the model are shown in appendix B. The equations presented in Gunaratnam et al. (2005) contained errors, but the appendix in this document displays the corrected versions.

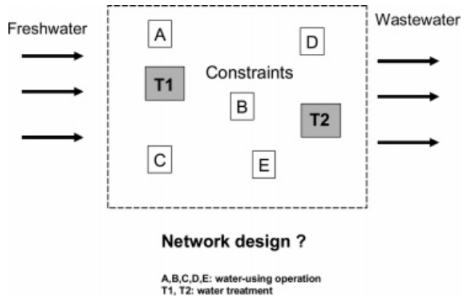


Figure 4.1: "Design problem of a total water system" (Gunaratnam et al., 2005, p.589). This figure shows the boundaries of a total water system, containing five industrial plants or water-using operations (A, B, C, D, and E) and two treatment systems (T1 and T2). There are multiple inflows of water from water sources into the total water systems, and multiple used water streams for discharge. Gunaratnam et al. (2005) often uses different terms than presented in this paper, as shown in this figure.

The design problem of a water network consists of industrial plants [1] and water treatment systems [2]. Each industrial plant has a water inflow and outflow that are of different qualities and quantities. Figure 4.1 shows the design problem of a total water system where, in this case, five industrial plants (A, B, C, D & E) and two water treatment systems (T1 & T2) are present within the boundaries of the total water system. When the system is considered as a whole, there can be one or multiple inflows of water and one or multiple outflows of used water. All these flows have quality standards, and the inflows have different costs.

[1] called water-using systems or operations by Gunaratnam et al. (2005)
[2] called water-treating systems or treatment unit by Gunaratnam et al. (2005)

A superstructure with all possible connections is shown in figure 4.2 when a system with two water sources[3], two industrial plants (O1 & O2), one water treatment system (TP), and two used water [4] discharge points are considered. In the figure, mixers and splitters are represented by small circles and squares, respectively.

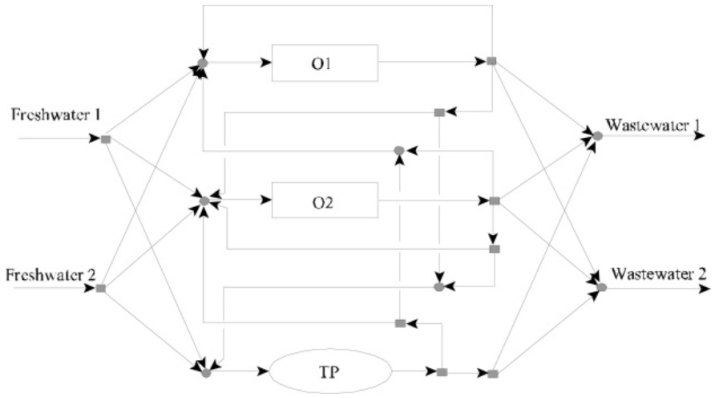


Figure 4.2: "Combined superstructure representation" (Gunaratnam et al., 2005, p.590). In this figure, two industrial plants (O1 and O2), two water sources (freshwater 1 and 2), two used water streams (wastewater 1 and 2), and one treatment system (TP) are shown. All possible connections are presented, including mixers (circles) and splitters (squares). In this superstructure, a possible connection between a water source and treatment system is shown, while that possibility is excluded in the tool presented in this study.

In the following section, an explanation of the model equations and constraints is shown. An explanation of the equations, where the equations themselves are shown beside the explanation can be found in Gunaratnam et al. (2005), but keep in mind that these equations contain mistakes, of which the correct versions are shown in appendix B.

4.2. Model Equations & Constraints

4.2.1. Mass Balances

The model consists of mass balances around the operations, mixers, splitters, and linear and nonlinear constraints, which are all shown in appendix B. A steady state is assumed for all industrial plants (or water-using operations) and water treatment systems (or treatment operation), together called operations, so a build-up of water or constituents is not possible according to the model. Figure 4.3 displays the variables that are considered around a general operation. In this figure, the operation is shown as 'U', a mixer before the operation is added as a circle, and a splitter after the operation as a square.

Equation B.1 is a mass balance around the total water system, as shown in figure 4.3 (equation 1). It is the sum of all in-going water flows from water sources ($F_{s,u}^{w}$) from water source 's' to operation 'u', minus the used water flows that are discharged ($F_{u,e}^{out}$) from operation 'u' to discharge point 'e', which is equal to the water losses of the operations (F_{u}^{loss}). The total flow through an operation (F_{u}^{t}) is the summation of inflows, minus the loss as illustrated by constraints B.7 and B.8.

The mass balances around the various operations are formulated by equation B.2; figure 4.3 (equation 2) illustrates the balance's boundaries. The inflow(s) from water resource(s), water discharge flow(s), and the water loss of the specific operation 'u' are considered, just like in equation B.1. In addition, the incoming flows from another operation 'ua' to operation 'u' are added ($F_{ua,u}^{ua}$), and the outgoing flows of operation 'u' to another operation 'ua' are subtracted ($F_{u,ua}^{ua}$) from the mass balance.

The in-going ($M_{c,u}^{in}$) and out-going ($M_{c,u}^{out}$) mass flows of the different constituents 'c' for operations 'u' are calculated by using the balance over the mixer (circle in figure 4.3) and the splitter (square in figure 4.3) respectively. This is done by multiplying the water flows with the concentrations of the contaminant

[3]Gunaratnam et al. 2005 calls water sources 'freshwater sources', but the water source could as well be another water source, like sea water or treated urban water

[4]wastewater in figure 4.2

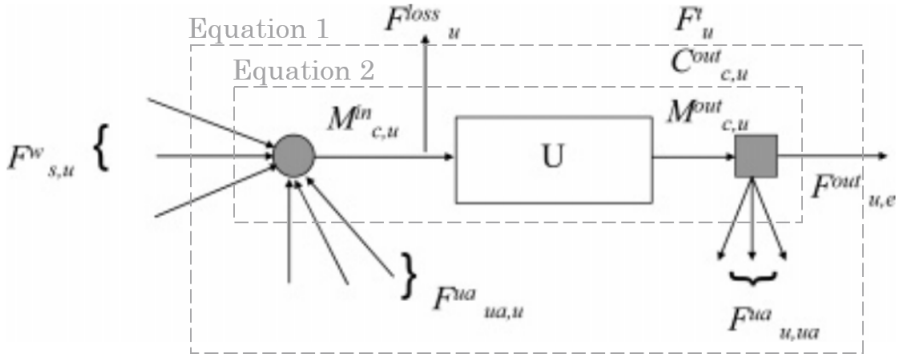


Figure 4.3: "Schematic representation of a basic operation" and mass balance visualization (Gunaratnam et al., 2005, p.595). An operation is shown as 'U' and the arrows represent water streams. The circle represents a mixer, where all in-going water flows from water sources ($F^{w}_{s,u}$) and water reuse flows ($F^{ua}_{ua,u}$) are combined. The square represents a splitter, where the used water from the operation is divided over reuse streams ($F^{ua}_{u,ua}$) and discharge flows ($F^{out}_{u,e}$). Equation 1 is a mass balance over the whole system containing all operations, and equation 2 is a mass balance over the streams of a single operation. Mass balances of constituents are produced by calculating the total mass entering the operation ($M^{in}_{c,u}$) and the total mass leaving the operation ($M^{out}_{c,u}$), which depends on the concentration of a constituent leaving an operation ($c^{out}_{c,u}$). The water losses are defined by F^{loss}_{u} and the total water through an operation is F^{t}_{u}. An overview of the symbols is shown in appendix H.

in the specific flows, as shown in equations B.5 and B.6 in appendix B. The mass balances over the water-using operations consist of the in-going mass plus the mass that enters the water in the operation (mass load), minus the out-going mass and the mass lost via the water losses (equation B.5). Equation B.6 is the mass balance over the water treatment systems, and consists of the in-going mass minus the mass losses and the out-going mass. The in-going mass is multiplied by one minus the removal ratio of the specific contaminant in the treatment operation, to calculate the amount of in-going mass left after treatment.

4.2.2. Constraints

Constraints are incorporated into the model to assist in identifying a feasible water network design based on the data and limitations. The constraints contain both continuous and binary variables. The binary variables, which are defined for every possible connection, provide an easy way to include restrictions on certain connections.

The limits on the total water flow rates going through the operations are defined in constraints B.7 and B.8. These inequality constrains are defined by setting a lower and upper limit on these flow rates, respectively. The minimum water flow rate through water-using operations is calculated by dividing the mass load that enters the water in the operation by the difference between the maximum concentration in the outflow and the maximum concentration in the inflow of a specific constituent. This is done for each constituent, and the lowest value is taken for the minimum water flow rate. A maximum water flow rate is introduced to prevent the model from calculating options that are very unrealistic. It is recommended to assume a maximum value for the water flow rate yourself and add an additional 50% to this value, to ensure that the value is taken broadly.

Constraint B.9 is included to give an upper limit to the mass flow into an operation for a specific constituent. This guarantees that water with a certain quality enters an operation. The limit on the discharge of constituents to discharge points is defined by constraint B.10, preventing that water with poor quality enters the environment.

Upper and lower limits to the separate water flows entering and leaving the operations, are defined by constraints B.11 to B.16. These limits are determined as one thinks best. The lower limit on a water flow rate introduces a threshold to eliminate small water flow rates, which is useful for simplifying the final design of a water network. In addition, constraints B.11 to B.16 couple the continuous variables of the water flows to the binary variables. An additional constraint is introduced to control the complexity of the water network, namely constraint B.17. This constraint sets a maximum number of water flows that can enter and leave an operation by using the binary variables defined earlier. Finally, constraint B.23 is introduced to prevent direct recycling.

The objective function, which is minimized to find the optimal solution, is the sum of the costs of the water coming from water resources, the costs of water treatment systems, and optionally the piping costs (equation B.28). The costs of the water are calculated by multiplying the costs of the water (in $ per ton water), the total water flows (in ton water per hour), and the operation time (in hours per year), as shown in equation B.24. The estimation of the water treatment costs is done based on an empirical equation that uses the total water flow through the treatment system to estimate the costs, which is an unique function for each treatment method. In Gunaratnam et al. (2005), the cost functions are separately defined for the capital expenditures (CAPEX) and operational expenditures (OPEX). A general cost function to estimate the water treatment costs is presented in appendix B (equation B.25).

4.3. Optional Equations

4.3.1. Elimination of Regeneration-Recycling

Constraints B.18 to B.22 are optionally included to eliminate regeneration-recycling. This is desired, since regeneration-recycling may result in the build-up of specific constituents that are not or not well removed in the treatment operations.

In order to eliminate regeneration-recycling, a distinction between streams that feed a treatment system and streams that are fed by a treatment system have to be made, which has not been done up to this point. Kuo and Smith (1998) introduced a method to divide the flows that are connected to treatment systems into these two groups: streams that feed a treatment system (group G1) and streams that are fed by a treatment system (group G2). Constraints B.18 to B.20, and B.22 are introduced to ensure that all streams that are connected to a treatment system belong to one of the two groups. Constraint B.21 ensures that regeneration-recycling connections are not possible.

4.3.2. Piping Costs

The costs of the piping depend on the cross-sectional area of the pipes and the flow velocity of the water through the pipes. The cross-sectional area of the pipe is calculated by using equation B.26. The piping cost is then calculated with equation B.27, which includes the distance between the pipes. Equations B.26 and B.27 are shown for the piping costs of the pipes from water sources to operations, but piping cost for the other connections are calculated in the same way.

4.4. Solution Strategy

The model described in section 4.1 results in mixed-integer nonlinear programming (MINLP). As explained before, the nonlinearity is caused by the power terms in the water treatment cost functions and the bilinear terms in the mass balances, since two unknown variables are multiplied (flow and concentration). The appearance of both continuous and binary variables transforms the nonlinear programming model into a mixed integer nonlinear programming model.

MINLP needs an initialization point to be solved, because the nonlinear terms may result in suboptimal local solutions and may cause convergence failures Gunaratnam et al. (2005). Gunaratnam et al. (2005) presents a method to find an initialization point, which is shown in figure 4.4. This is an iterative method, which consists of using mixed-integer linear programming (MILP) and linear programming (LP) alternately.

The MILP and LP both contain an relaxed form of the mass balances, equations B.29 and B.30, by adding mass loss and mass gain terms. In the MILP, the outflow concentrations are fixed and flows are calculated by minimizing the annual costs. In addition, the cost functions are linearized over a range of effluent flow rates to remove the last nonlinear terms. Subsequently, the flows in the LP are fixed to the flows calculated by the previous MILP and new concentrations are calculated. This is done by minimizing the sum of the mass loss and mass gain terms (M^{terms}, equation B.36), that were introduced in the mass balances. The calculated concentrations in the LP are then used as an input for the MILP again, and the iteration continuous until the M^{terms} is lower than the specified criterion (ϵ). The final result of the flows and concentrations will be the initialization for the MINLP in the final stage.

In MILP and LP, not all equations and constraints present in the MINLP are used. The MILP contains equations B.29, B.30, B.31, B.32, which are slightly adjusted functions based on B.5, B.6, B.3, B.4 respectively. Furthermore, the MILP contains equations B.1, B.2, B.7 - B.9, B.11 - B.23, and equation B.33 which is similar to equation B.28. The LP only contains equations B.34 and B.35, which are projections of equations B.3 and B.4, in addition to equations B.29, B.30, and B.36. A more detailed explanation of the solution stategy is written by Gunaratnam et al. (2005).

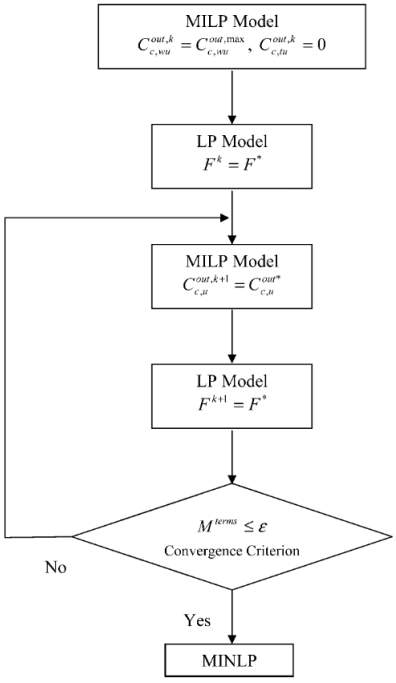


Figure 4.4: ”Solution strategy” (Gunaratnam et al., 2005, p.595). The solution strategy begins by using the MILP model, by fixing the outgoing concentrations of the constituents leaving water-using operations ($c_{c,wu}^{out,k}$) to the maximum possible concentration ($c_{c,wu}^{out,max}$). The outgoing concentration of constituents leaving treatment units is fixed to zero. The flow rates calculated by the MILP model (F^*) are used to fix the flow rates in the LP model (F^k). The output of the LP model are concentrations ($c_{c,u}^{out*}$) that are used to fix the concentration in a new run of the MILP model ($c_{c,u}^{out,k+1}$). The flow rates calculated by this MILP model (F^*) are used as an input for a new run of the LP model (F^{k+1}). The M^{terms}, which is the sum of all mass loss and mass gain terms introduced in the MILP and LP models, is calculated. If the value is below the convergence criterion, the concentrations and flows are used to initialize the MINLP, but if the value is above the convergence criterion, another iteration of the MILP and LP model is executed.

5

Water Network Design Tool

5.1. Explanation of Developed Tool

The tool is developed according to the stages described in section 2.2 in the Methods as shown in figure 2.1, by using the model of Gunaratnam et al. (2005) explained in the previous chapter. The first stage, the 'Design Concept' stage, consisted of defining the goal, planning, and concept. The results of this stage are described in the Introduction, sections 1.2 and 1.3. The consecutive stage, called 'Model & Tool Development', consists of the recurring phases of making a design or idea, and subsequently developing and testing it. Firstly, this included developing the MILP, LP, and MINLP in separate files until feasible results were obtained. These were combined into a single file by putting the MILP, LP, and MINLP into different Python functions. These functions are used in the solution strategy consisting of the initiation, iteration, and final MINLP as described in section 4.4. Some additional functionalities were added to enhance the user-friendliness of the tool, such as the visualisation of the MINLP results, calculating the water savings, the introduction of 'switches', and the coupling to the Industrial Wastewater Treatment Technology (IWTT). The third stage shown in figure 2.1 starts after the first deployment of the tool, when new functionalities are to be added, required adjustments linked to updates of packages or software are to be made, and potential mistakes are to be corrected. The four phases of design, development, testing and deployment will be repeated each time for these changes. This stage starts after the study presented here is shared in the TU Delft repository.

The tool is developed in Python. The code is divided into multiple parts that are also shown in the code in appendix F, namely:

- 'IMPORT PACKAGES'
- 'FUNCTIONS (PARTLY) EXTERNAL'
- 'INPUT'
- 'IMPORT OF DATA AND DATA VISUALISATION'
- 'INDUSTRIAL WASTEWATER TREATMENT TECHNIQUES (IWTT)'
- 'DEFINE PARAMETERS BASED ON INPUT'
- 'DEFINE LISTS TO STORE RESULTS'
- 'MILP FUNCTION'
- 'LP FUNCTION'
- 'MINLP FUNCTION'
- 'INITIATION'

- 'ITERATION'
- 'FINAL MINLP'
- 'WATER SAVINGS'
- 'MINLP RESULTS VISUALISATION'

In the JupyterLab file, each of these parts was placed in a different cell. In this file, all cells are hidden, except for the 'Input', which makes the file easy to use. The build-up and function of all parts of the Python code is described in the following sections.

5.1.1. Import Packages

The Python packages that are used in the tool are imported in this part. An overview of these packages is shown in section 2.2, chapter 2.

5.1.2. Functions (Partly) External

In this part, two functions are defined that are used later in the tool. The first function was posted on Stack Overflow (kcoskun, 2021) and was unchanged. It is used in the tool to place the flow labels in the middle of the lines for the visualisation of the final network. The Networkx package, which is used to make the visualisation, does not provide this function for curved edges.

The second function shown in this section is used for the linearization procedure of the cost functions of the water treatment systems for the MILP function. The code used in this function is mostly copied from Stack Overflow (cottontail, 2023), but is adjusted to work in the tool. The function makes a regression line through a number of data points that lay on the original water treatment function that is to be linearized. The interval on which this is done, is defined on the 'Flow Limits' tab of the Excel input file (as shown in Appendix E, figure E.6). The number of scatter points is defined as 10 in the Python file under 'Define Parameters Based on Input' and can be changed if desirable.

5.1.3. Input

This is the only cell that is visible when using the tool, as shown in figure 5.1, which is the example for the first case study presented later in this report. All of the remaining cells are hidden, but every cell can be accessed by clicking on the three dots as visible in figure 5.1.

In the input cell, the name of the Excel file that contains the data is given first. This file should be in the same folder as the JupyterLab file of the tool. Next, the size of the sets used in the optimization model are presented, which are the number of constituents, the number of water sources, the number of discharge points, the number of water-using operations, and the number of treatment units. Subsequently, the yearly operation time of the industrial plants is defined, just like the annualization factor that is the fraction of the capital costs of the treatment systems that should be accounted for in one year. This is followed by defining what solvers are used to solve the MILP, LP, and MINLP. The location of the place on your computer where the solver is installed has to be given as well. The maximum number of iterations is defined to make sure that the iterations stop in case convergence does not occur. The convergence criterion, so the lower limit the M^{terms} should reach before starting the MINLP, is defined thereafter.

Next, some 'switches' are introduced, in order to include or exclude certain options. Setting the switch to 0 means that the specific function is inactivated or not included, and 1 is the opposite. The first four switches are defined to in- or exclude water reuse, regeneration recycling, environmental discharge limits, and pipe connection costs, respectively. The following three switches are to activate or inactivate certain parts of the solution strategy, described in section 4.4. The initiation (the first MILP & LP), the iteration loop, and the final MINLP can be activated separately. The final five switches determine what results are shown. The input data from the Excel file is made visible by activating the show excel data switch. The final visualisation of the MINLP is shown when activating the switch thereafter. The last three switches are to print the calculated variables of the MINLP, MILP, and LP subsequently. When the MILP and LP switches are activated, the calculated variables of every iteration are shown.

Figure 5.1: Python Input Petroleum Refinery Case Study, scenario 1

5.1.4. Import of Data and Data Visualisation

The data from the Excel input file is imported in this part, and is displayed in case the 'show excel data switch' is activated.

5.1.5. Industrial Wastewater Treatment Techniques (IWTT)

In this part, the database of the IWTT is uploaded in the tool first. Therefore, the csv files of the database should be in the same folder as the JupyterLab file. The csv files can be downloaded via the website of the Environmental Protection Agency (EPA) ("Industrial Wastewater Treatment Technology Database (IWTT) | US EPA", 2023). Then, the columns of the datasets that are not needed are removed in Python. The multiple files are combined into one dataframe that contains all relevant data. The following step is making a list of the industries that are part of the specific case study, which are defined in the Excel file. With this information, only the data from IWTT of these industries will be displayed. There is an option to save the data in Excel files, which is especially helpful in case that the data is too much to display in JupyterLab. There is an option build in this cell, that stops to run the code from this point. This is useful when a user of the tool only want to check the dataset, and does not want to run the model yet.

5.1.6. Define Parameters Based on Input

This is the part where the parameters that are defined earlier in the Excel file, are linked to the parameters that are defined in the equations of the optimization model.

5.1.7. Define Lists to Store Results

The lists to store the results of the costs and the M^{terms} are defined here, so the results of the different iterations can be printed after solving the problem. This is needed, because the results of the MILP and LP are deleted after every iteration.

5.1.8. MILP Function

In this section, a function is defined that contains all equations and constraints for the mixed integer linear programming. The structure is the same as the example shown in section 2.2 in chapter 2. First, the model is defined outside the function. Then, the MILP function is defined, that takes a list as an input that contains the concentrations of the the different concentrations of the water-using and treatment operations. Within the function, a few lists are defined to store results than will function as an input for the LP function later.

Then, all variables are added to the model, followed by the model equations and constraints. The MILP model is solved after that, and is followed by the option to print the results and by storing some of the results in lists. All variables, equations, and constraints are deleted in the final part of the function, which is required to be able to run the function again in the iteration. There are three outputs of the function. The first one is a list of the flows that serves as an input for the LP in the iteration. The second one are the binary variables of these flows, and is used for the MINLP initialization. The final output is a list of the total flows through the operations, which is only used when the the iteration switch is off.

5.1.9. LP Function

The linear programming function is similar as the MILP function in the structure, but different equations are used, as described in section 4.1. The input of the LP function is a list of flows, which contains the flows from water sources, the discharge flows, and the reuse flows. The linear programming function has one output, which is a list of the concentrations of the different constituents in the outflows of the operations.

5.1.10. MINLP Function

The structure of the MINLP function is the same as the MILP and LP described before, except that the equations are different as described in section 4.1, and that the variables, equations and constraints are not deleted in the end, since the MINLP function is always only run once.

The MINLP has four different inputs. The first input is a list of initialization values for the freshwater, discharge, and reuse flows. The second one is a list that gives the binary variables of aforementioned list. The next input is a list with the initialization values of the concentrations of the different constituents in the outflows of the operations. The fourth input is a list of the total flows through the operations, which is only used in case that the iteration is inactivated. The MINLP function has one output, the final total costs.

5.1.11. Initiation

The initiation is the first step of the solution strategy. The MILP function is run for the first time. The concentrations of the constituents in the outflows of the water-using operations are set to the maximum value, and the concentration of the constituents in the outflows of the treatment operations are set to 0. Gunaratnam et al. (2005) mentioned that these concentrations of the treatment operations may be set to the environmental discharge limit in case the iterations does not result in a value of M^{terms} under the convergence criterion. This change can be made here if needed. The LP function is also run for the first time, with the results from the MILP function as an input. These results will be used for the first iteration in the next part. The results of the initiation are printed.

5.1.12. Iteration

After the initiation, the iteration starts, which is the second step of the solution strategy. It start with the MILP function with the LP result from the initiation as an input. The output of this MILP, is used as an input in the next LP. This circle continuous until the M^{terms} is lower than the convergence criterion, as explained in section 4.1. The convergence criterion is set to 0.01 in this research. The results of the functions are printed during the iterations.

5.1.13. Final MINLP

The third and last step of the solution strategy, is running the MINLP with the results from the initialization procedure, that consists of the initiation and iteration steps. The results of the solver are shown when the MINLP function provides an optimal solution.

5.1.14. Water Savings

In this part, the savings of water use and the production of discharge water are calculated. This is done by comparing it to a linear system, so where the (minimum amount of) water is used in the operation only once before discharge. The results are printed in sentences.

5.1.15. MINLP Results Visualisation

This part of the code visualises the network in a graph, by using the networkx Python package. First, the flows are stored in lists, which are used to make edges in the graph. This is followed by creating nodes for every water source, water-using operation, treatment operation, and discharge point. Lastly, the figure containing the network is drawn and labels are added. Examples of these visualisations are shown in the next section.

5.2. Case Study Results

The case studies are adopted from Gunaratnam et al. (2005) to test whether the developed tool gave similar results as the model of aforementioned paper. The IWTT database is therefore not used in case studies 1, 2 & 3.

5.2.1. Petroleum Refinery Case Study

Input Petroleum Refinery Case Study

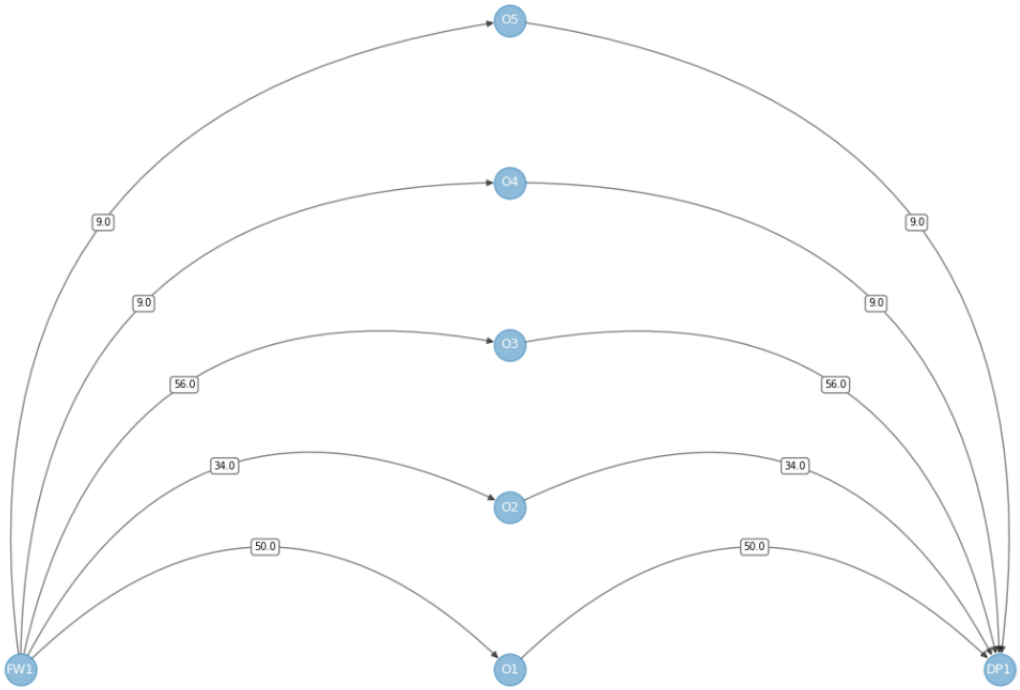


Figure 5.2: Initial situation Petroleum Refinery Case Study. The initial, linear situation are five industrial processes (O1, O2, O3, O4, and O5) that all use water from the same water source (FW1). After usage, the water from all industrial processes is discharged to a the same discharge point (DP1).

In this case study, a design problem for a simplified petroleum refinery is considered. This case study is not for designing an interplant water network, but for an in-plant water network. However, it is still relevant for testing the performance of the tool. The plant consists of five water-using operations, which are shown in table 5.1. The initial, linear system containing the minimum flows is shown in figure 5.2. Three constituents are considered, namely hydrocarbon (HC), hydrogen sulfide (H_2S), and suspended solids (SS), and three treatment operations: steam-stripping column (T1), biological treatment unit (T2), and API[1] separator (T3). The removal ratios for all constituents and treatment operations are shown in table 5.2.

Table 5.1: ”Water-using operations for case study 1” (Gunaratnam et al., 2005, p.596)

Operation	Description	Operation	Description
O1	steam stripping	O4	VDU
O2	HDS-1	O5	HDS-2
O3	desalter		

[1]API stands for American Petroleum Institute

Table 5.2: ”Performance of the treatment units for case study 1” (Gunaratnam et al., 2005, p.596)

	Removal ratio		
Operation	HC	H_2S	SS
T1	0.00	0.999	0.00
T2	0.70	0.90	0.98
T3	0.95	0.00	0.50

The costs of the treatment operations are estimated by using functions where the CAPEX and OPEX are a function of the total water inflow of the treatment operation. The functions for the CAPEX and OPEX of each operation are shown below.

$$CAPEX: Cost_{T1}^{TU}(\$) = 16800 * {F_{tu}^{t}}^{0.7} \tag{5.1}$$

$$OPEX: Cost_{T1}^{TU}(\$/h) = F_{tu}^{t} \tag{5.2}$$

$$CAPEX: Cost_{T2}^{TU}(\$) = 12600 * {F_{tu}^{t}}^{0.7} \tag{5.3}$$

$$OPEX: Cost_{T2}^{TU}(\$/h) = 0.0067 * F_{tu}^{t} \tag{5.4}$$

$$CAPEX: Cost_{T3}^{TU}(\$) = 4800 * {F_{tu}^{t}}^{0.7} \tag{5.5}$$

$$OPEX: Cost_{T3}^{TU}(\$/h) = 0 \tag{5.6}$$

To obtain the annual treatment costs, the CAPEX is multiplied by an annualization factor, which is 0.1 in this case, and the OPEX is multiplied by the yearly operating time of the plant site, which is equal to 8600 h. The costs of the water source are \$0.2 per ton, and this water is assumed to be free of constituents. Furthermore, the environmental limits for HC, H_2S, and SS are 20 ppm, 5 ppm, and 100 ppm respectively.

Table 5.3 shows the operating data for the water-using operations for case study 1, which includes the minimum and maximum allowable in- and outflow concentrations for each operation and constituent, the limiting mass load, and the limiting flow rate. The maximal total flow through the operations was estimated to be 150 ton/h. The upper limit for the flows from water sources, discharge flows, and reuse flows was set to 100 ton/h, and the lower limit for the flows from water sources was set to 1 ton/h, but for the discharge and reuse flows to 9 ton/h. The limits for the linearization procedure of the treatment operations were set to 10 and 150 ton/h. The maximum amount of water sources an operation can receive, which consists of reuse flows and flows from water sources, is set to three.

Table 5.3: ”Operating data for the water-using operations for case study 1” (Gunaratnam et al., 2005, p.596)

Operation	Contaminant	$C_{c,u}^{in,max}$ (ppm)	$C_{c,u}^{out,max}$ (ppm)	$L_{c,u}^{ml}$ (g/h)	$F_{c,u}^{ml}$ (ton/h)
O1	HC	0	15	750	50
	H2S	0	400	20000	50
	SS	0	35	1750	50
O2	HC	20	120	3400	34
	H2S	300	12500	414800	34
	SS	45	180	4590	34
O3	HC	120	220	5600	56
	H2S	20	45	1400	56
	SS	200	9500	520800	56
O4	HC	0	20	160	8
	H2S	0	60	480	8
	SS	0	20	160	8
O5	HC	50	150	800	8
	H2S	400	8000	60800	8
	SS	60	120	480	8

The distances between the operations and the distance from the discharge point, are shown in table 5.4. These are used to estimate the piping costs in scenario 3, which are assumed to be made of carbon steel. The parameters of the costs for piping work are 3603.4 for $a_{s,u}^{fw}$, $a_{u,ua}^{ua}$, and $a_{u,e}^{out}$, and is equal to 124.6 for $b_{s,u}^{fw}$, $b_{u,ua}^{ua}$, and $b_{u,e}^{out}$. The flow velocity through all the pipes in the network is assumed to be 1 m/s.

Table 5.4: ”Distance matrix (m) for case study 1” (Gunaratnam et al., 2005, p.596)

	O1	O2	O3	O4	O5	T1	T2	T3	Discharge
S1	30	25	70	50	90	200	500	600	2000
O1	0	30	80	150	400	90	150	200	1200
O2	30	0	60	100	165	100	150	150	1000
O3	80	60	0	50	75	120	90	350	800
O4	150	100	50	0	150	250	170	400	650
O5	400	165	75	150	0	300	120	200	300
T1	90	100	120	250	300	0	125	80	250
T2	150	150	90	170	120	125	0	35	100
T3	200	150	350	400	200	80	35	0	100

Output Petroleum Refinery Case Study, Scenario 1

In scenario 1 of the Petroleum Refinery Case Study, regeneration-recycling was allowed and the costs of piping were not taken into account. The result of the optimized water network design is shown in figure 5.3. FW1 is the water source, O1 until O5 are the five industrial processes, T1 until T3 are the three treatment units, and DP1 is the discharge point. The arrows show the water flows and the number in the middle of the arrow shows the water flow rate in tons per hour. The result shows that regeneration recycling is possible in this scenario, as the water from industrial process O3 is treated in treatment systems T3 and T2, and then recycled to industrial process O3.

The initialization was finished in three iterations and the MINLP within 49.3 seconds. The price of the optimized network is $591926. The use of water from water sources and the amount of discharged water decreased with 61.7% in comparison with a linear system where the minimum water flow goes through the operations and where the water is not reused. The complete result of the whole run is shown in appendix G.

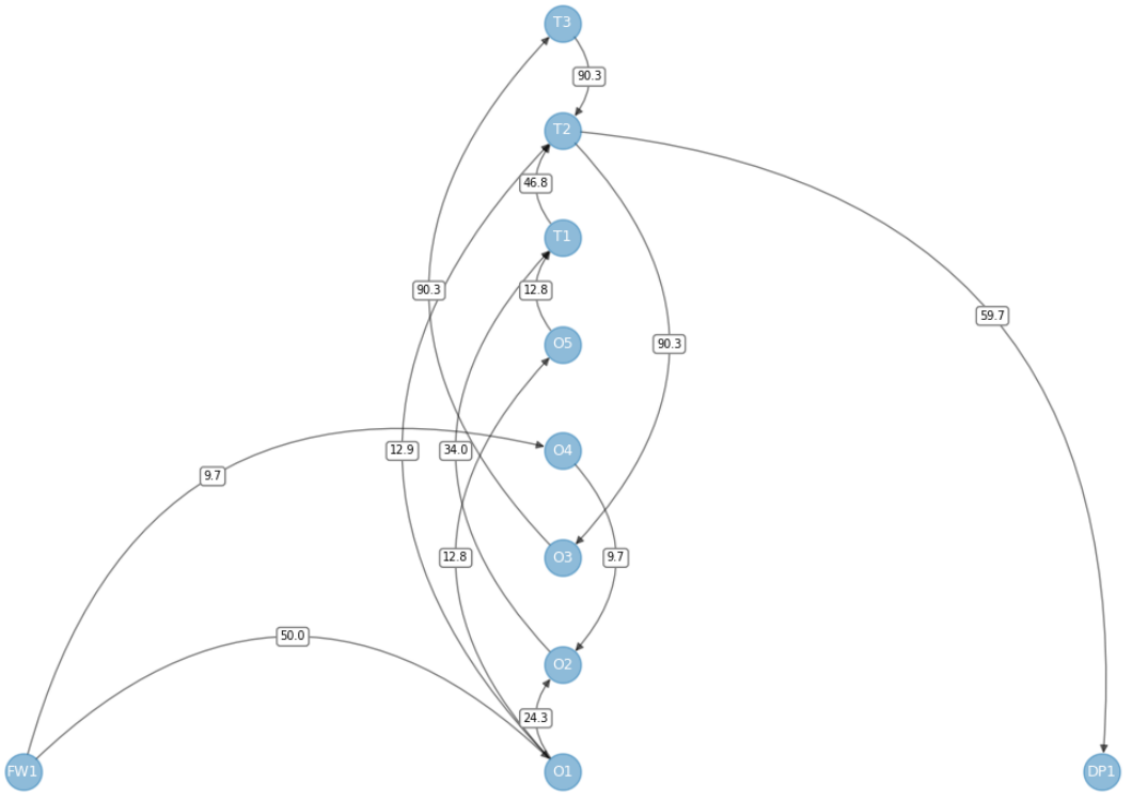


Figure 5.3: Results Petroleum Refinery Case Study by allowing regeneration recycling without piping costs (scenario 1). This figure shows five industrial processes (O1, O2, O3, O4, and O5) and three water treatment systems (T1, T2, and T3). Processes O1 and O4 use water from the water source (FW1). The water from process O1 is partly reused in processes O2 and O5, the remaining is treated in treatment system T2. The water used in operation O4 is reused in process O2. The water used in processes O2 and O5 is treated in treatment system T2. The water treated in treatment system O2 is partly reused in process O3, which is treated in treatment system O3 and subsequently in treatment system O2 after usage. The water from treatment system O2 that is not reused, is discharged to a discharge point (DP1). The water flow rates are shown in the middle of each arrow in tons of water per hour.

Output Petroleum Refinery Case Study, Scenario 2

Scenario 2 of the Petroleum Refinery Case Study still does not include the costs of piping, but regeneration-recycling excluded. Figure 5.4 shows the optimized network design of this scenario.

In the first run, when the initial outgoing concentrations of the treatment units were set to 0, the MINLP did not give a result, as shown in appendix G. After waiting for 1000s, it was decided to set the initial outgoing concentration of the treatment units equal to the environmental discharge limits, like suggested in Gunaratnam et al. (2005).

For this run, two iterations were needed to reach a M^{terms} lower than 0.01. The MINLP took 293.8 seconds to complete and resulted in an optimized network with an estimated price of $679414. The use of water from water sources and discharge of used water reduced with 61.7% in this scenario, when compared to a linear system. The results of the different iterations, and the values of all variables are shown in appendix G.

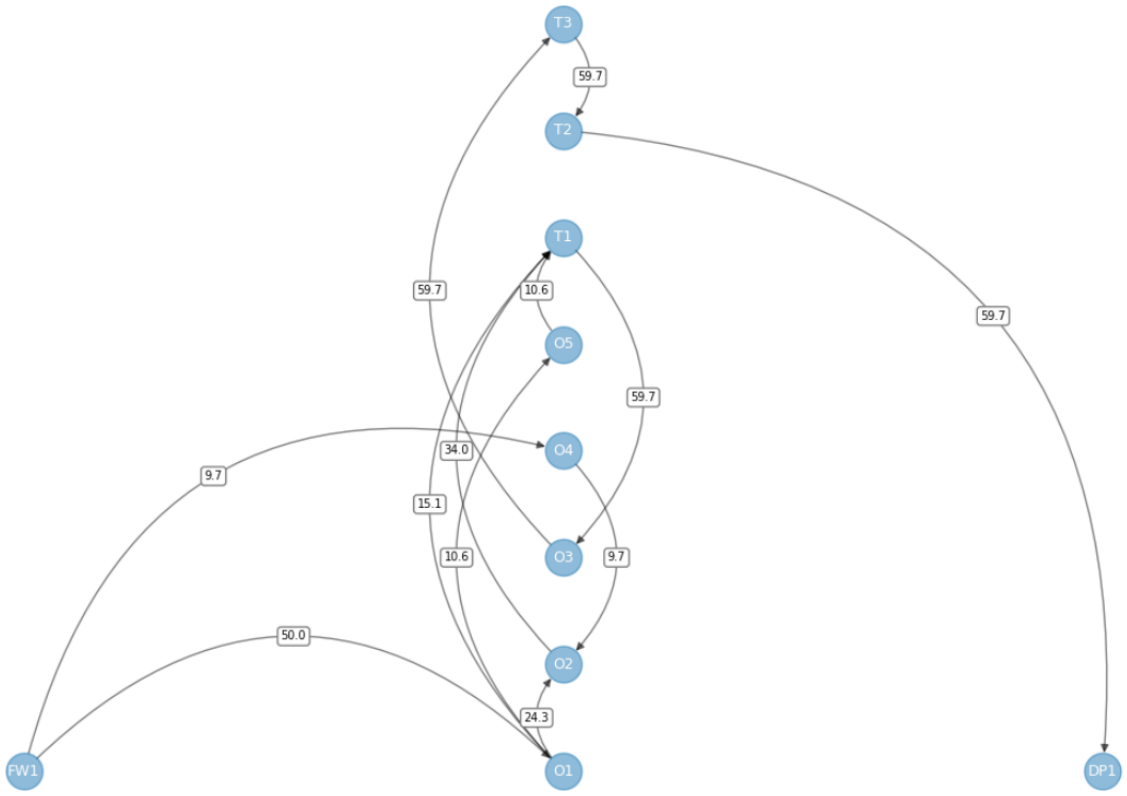


Figure 5.4: Results Petroleum Refinery Case Study, regeneration-recycling not allowed and piping costs not included (scenario 2). This figure shows five industrial processes (O1, O2, O3, O4, and O5) and three water treatment systems (T1, T2, and T3). Processes O1 and O4 use water from the water source (FW1). The water from process O1 is partly reused in operations O2 an O5, the remaining is treated in treatment system T1. The water used in process O4 is reused in process O2 and the water used in process O5 is treated in treatment system T1. The water treated in treatment system T1 is reused in process O3. After usage in process O3, all water is treated in treatment systems T3 and T2, respectively, before it is discharged to discharge point 1. The water flow rates are shown in the middle of each arrow in tons of water per hour.

Output Petroleum Refinery Case Study, Scenario 3

In scenario 3 of the Petroleum Refinery Case Study, the piping costs are included and regeneration recycling is not allowed. The result of this scenario is presented in figure 5.5.

This scenario took four iterations to converge. The MINLP took about 41 seconds to reach the optimal solution and gave a network with an estimated price of $862984. The use of water from water sources and water that is discharged decreased with 51.6% when compared to a linear system. The complete results of the iterations and the MINLP is shown in appendix G.

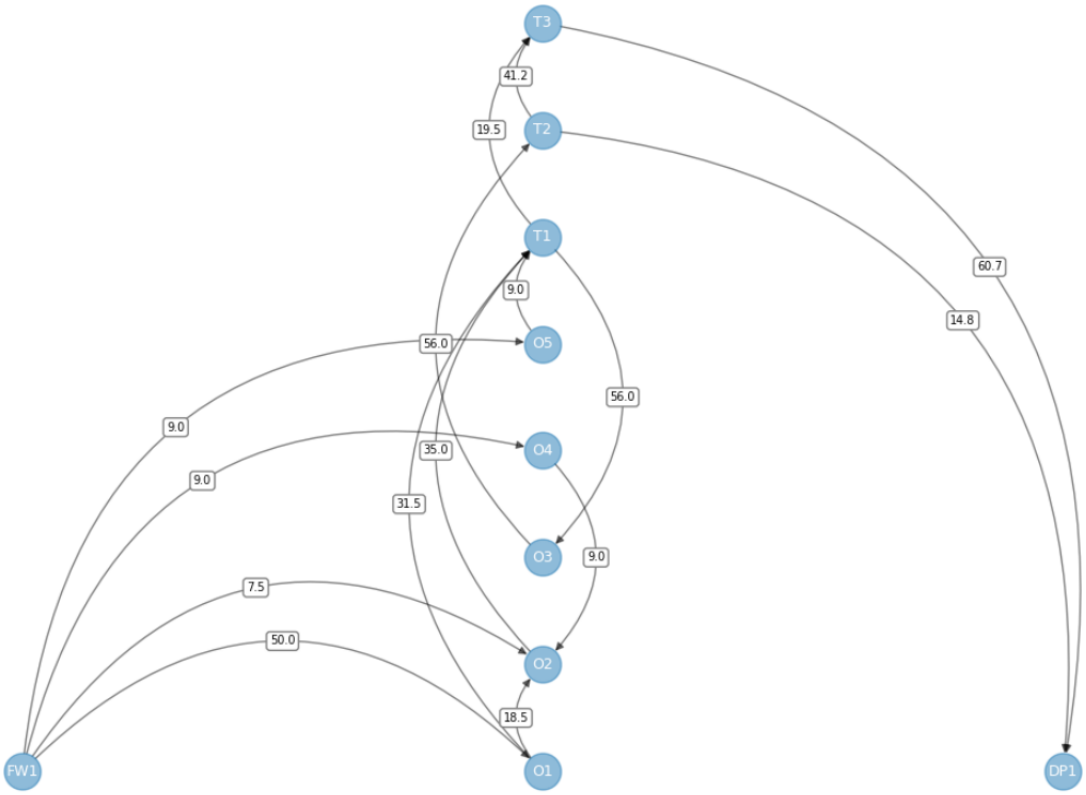


Figure 5.5: Results Petroleum Refinery Case Study, regeneration recycling not allowed and piping costs are included (scenario 3). This figure shows five industrial processes (O1, O2, O3, O4, and O5) and three water treatment systems (T1, T2, and T3). Industrial processes O1, O2, O4, and O5 use water from the water source (FW1). The water used in process O1 is partly reused in process O2 and partly treated in treatment system T1. The used water of process O4 is reused fully in process O2. The water used in process O2 is treated in treatment system T1, just like the water used in process O5. Part of the water treated in treatment system T1 is used in process O3, the remaining is treated in treatment system T3. The water used in process O3 is treated in treatment system T2. Part of the water treated in treatment system T2 is discharged directly to the discharge point (DP1), while the remaining is treated in treatment system T3. After treatment in treatment system T3, the water is discharged. The water flow rates are shown in the middle of each arrow in tons of water per hour.

5.2.2. Water Reuse Case Study

Input Water Reuse Case Study

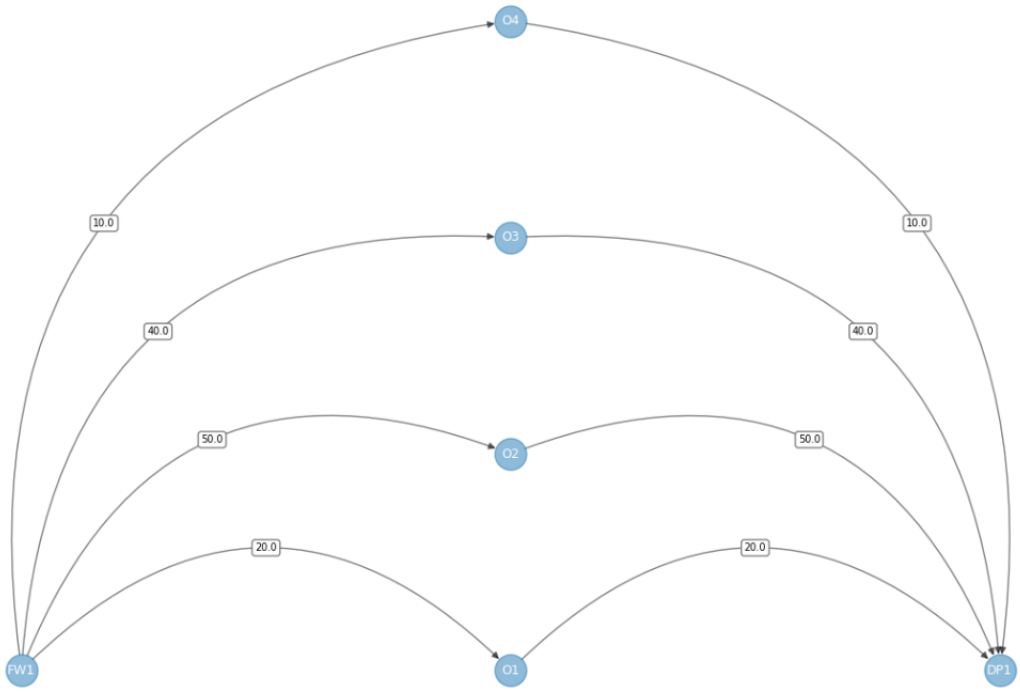


Figure 5.6: Input Water Reuse Case Study. The initial, linear situation are four water-using operations (O1, O2, O3, and O4) that all use water from the same water source (FW1). After usage, the water from all water-using operations is discharged to the same discharge point (DP1).

The case study presented in this section consists of four water-using operations, but does not include treatment systems. The limiting water data for the water-using operations, considering a single, unspecified constituent, is shown in table 5.5. This case study was used earlier by Wang and Smith (1994), but is also used by Gunaratnam et al. (2005).

Since treatment systems and piping costs are both excluded in this case study, the total annualized costs consist only of the costs of water from water sources. Therefore, the optimization in costs would result in the same network as the minimization of water use. The costs of the water source are $0.75 in this study, the annual operating hours are 8600 hours, and the annualization factor is 0.1. It was assumed that the water source was free of the unspecified contaminant.

The maximum total flow through the operations was set to 110 tons per hour and the maximum amount of receiving flows was set to 2. The upper limits for the flows from the water source, discharge flows, and reuse flows were set to 101 ton/h, and the lower limit was set to 9 ton/h.

Table 5.5: ”Limiting water data for case study 2” (Gunaratnam et al., 2005, p.597)

Operation	$C_{c,u}^{in,max}$ (ppm)	$C_{c,u}^{out,max}$ (ppm)	$L_{c,u}^{ml}$ (g/h)	$F_{c,u}^{ml}$ (ton/h)
O1	0	100	2000	20
O2	50	100	5000	50
O3	50	800	30000	40
O4	400	800	4000	10

The initiation was done, without the iteration, and the total flows through the operations were fixed from the initiation in the MINLP.

Output Water Reuse Case Study

The optimized network of the water reuse case study is shown in figure 5.7. The MINLP took 0.17 seconds to complete. The estimated price of the network is $580500, and the use of water from the water source and the discharged wastewater decreased with 25% when compared to the initial, linear system. The complete results of the initiation and the MINLP is shown in appendix G.

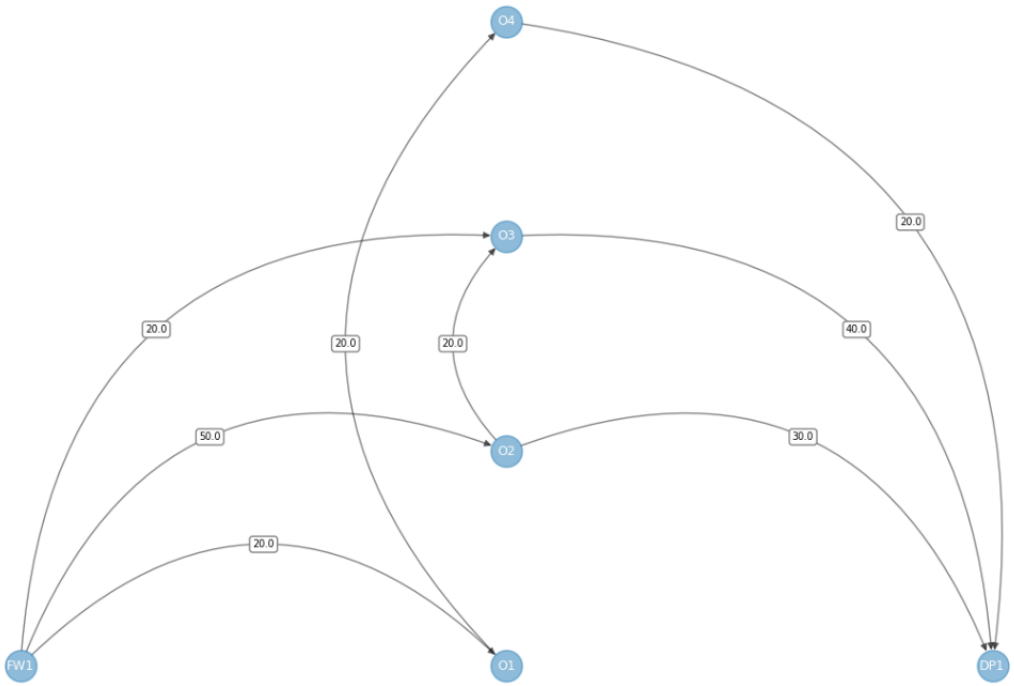


Figure 5.7: Results Water Reuse Case Study. The four water-using operations are shown as O1, O2, O3, and O4. Operations O1, O2, and O3 use water from the water source (FW1). O3 partly uses used water from operation 2. Operation 4 reuses the water from O1 completely. The water used in O3 and O4 is discharged to the discharge point (DP1), as well as the remaining water of operation O2. The water flow rates are shown in the middle of each arrow in tons of water per hour.

5.2.3. Water Treatment Case Study

Input Water Treatment Case Study

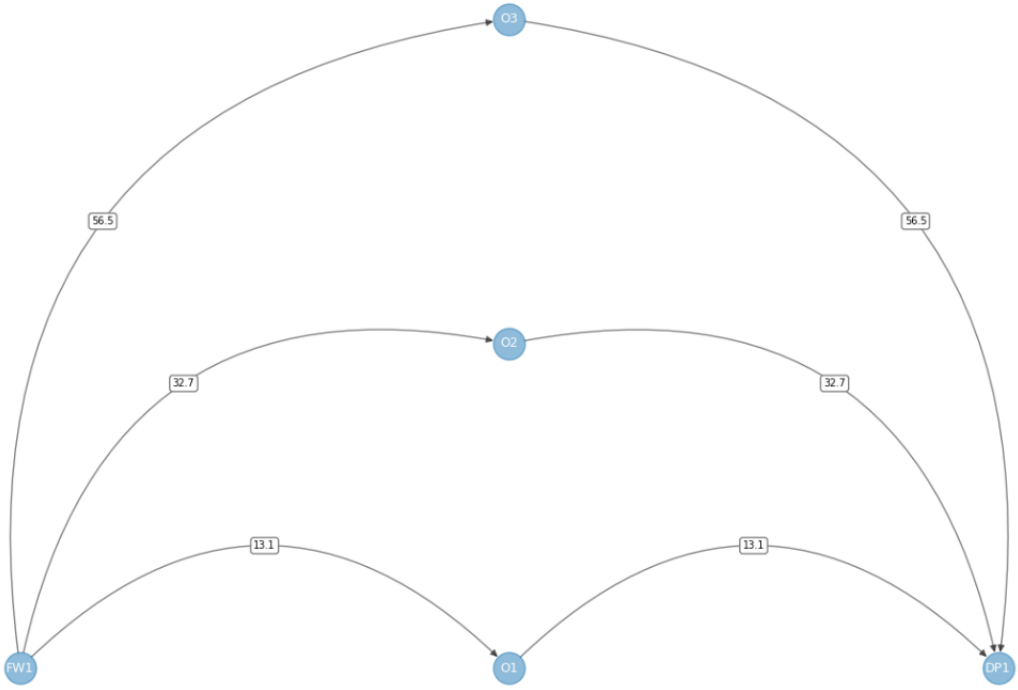


Figure 5.8: Input Water Treatment Case Study. The initial, linear situation contains three water-using operations (O1, O2, and O3) that all use water from the same water source (FW1). After usage, the used water of each operation is discharged to a common discharge point (DP1).

The case study presented in this section consists of three water-using operations, three treatment operations, and three constituents, and is taken from Gunaratnam et al. (2005). Water reuse is not allowed in this case study, so the treatment operations are used exclusively to meet the environmental discharge limits of the constituents. The constituents are hydrogen sulfide (H_2S), oil, and suspended solids (SS) and have an environmental limit of 2 ppm, 2 ppm and 5 ppm, respectively. Table 5.6 shows the limiting data for the three operations in this case study. The used water treatment systems for this case study are the same as for the Petroleum Refinery Case Study, presented in the same order. The removal ratios for every constituent are shown in table 5.7.

Table 5.6: "Limiting water data for Water Treatment Case Study" (Gunaratnam et al., 2005, p.598)

Operation	Contaminant	$C_{c,u}^{in,max}$ (ppm)	$C_{c,u}^{out,max}$ (ppm)	$L_{c,u}^{ml}$ (g/h)	$F_{c,u}^{ml}$ (ton/h)
O1	H_2S	0	390	5109	13.1
	oil	0	10	131	13.1
	SS	0	25	327.5	13.1
O2	H_2S	0	16780	548706	32.7
	oil	0	110	3597	32.7
	SS	0	40	1308	32.7
O3	H_2S	0	25	1412.5	56.5
	oil	0	100	5650	56.5
	SS	0	35	1977.5	56.5

The distances between the operations and the water sources and discharge point, are shown in table 5.8. The cost parameters are the same as the Petroleum Refinery Case Study, except for the annualization factor that is equal to 0.4 in this case study. The lower limit of the flows of water from the water

Table 5.7: ”Performance of the treatment units for Water Treatment Case Study” (Gunaratnam et al., 2005, p.598)

	Removal ratio		
Operation	H_2S	Oil	SS
T1	0.999	0	0
T2	0.9	0.9	0.97
T3	0	0.95	0.2

source, discharge flows, and reuse flows are set to 9 ton/h and the upper limit to 150 ton/h. The limits for the linearization procedure of the treatment units were set to 10 and 150 ton/h.

Table 5.8: ”Distance matrix (m) for case study 3” (Gunaratnam et al., 2005, p.598)

	O1	**O2**	**O3**	**T1**	**T2**	**T3**	**Discharge**
O1	0	0	0	30	150	50	170
O2	0	0	0	40	125	80	90
O3	0	0	0	35	120	40	100
T1	30	40	35	0	80	30	35
T2	150	25	120	80	0	65	55
T3	50	80	40	30	65	0	40
Discharge	170	90	100	35	55	40	0

Output Water Treatment Case Study

The network that resulted from the optimization of the water treatment case study is shown in figure 5.9. After one iteration, the M^{terms} reached a value below 0.01. The MINLP took less than one second to run, and resulted in a network with an estimated cost of $857792. There is no decrease in the use of water and the production of used water, because this is a linear system as water reuse was not allowed in this case study. More details on the results are shown in appendix G.

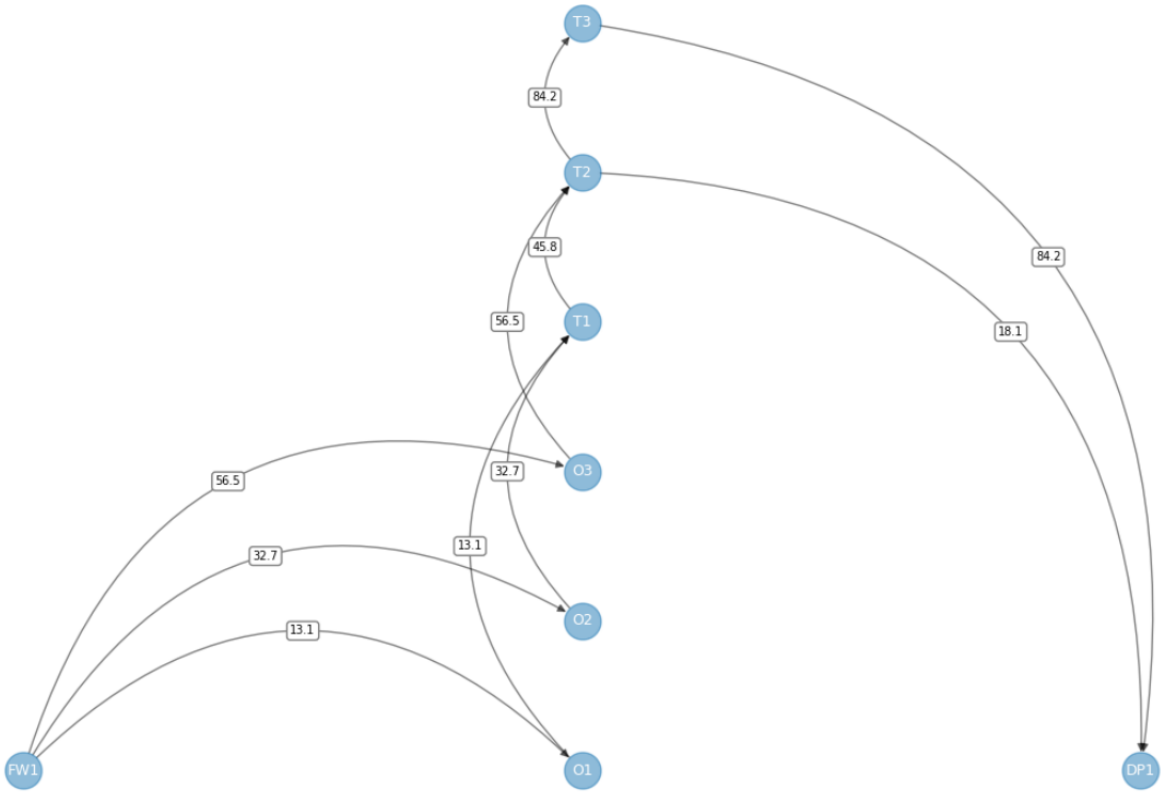


Figure 5.9: Results Water Treatment Case Study. This figure shows three water-using operations (O1, O2, and O3) and three water treatment systems (T1, T2, and T3). The operations all use water from the water source (FW1), since water reuse is not allowed. The water used by operations O1 and O2 is treated by treatment system 1 and treatment system 2, respectively. The water used in operation 3 is treated directly by treatment system 2. Part of the water treated in treatment system 2 is treated by treatment system 3 before discharge, while the remaining is discharged directly to the discharge point (DP1).

6

Discussion

6.1. Water Network Design Tool Choices

The tool was implemented using Python with the Pyomo package, as shown in the Methods. Python was chosen over for example GAMS and LINDO software, because it is open-source. Compared to other programming languages, Python is relatively easy to learn and it is widely used, making it an accessible platform to create the tool in.

The selected solver for the optimization of the LP, MILP, and MINLP was SCIP. Although alternative open-source solvers could have been used for LP, such as ipopt, gurobi, and cplex. Couenne is an open-source solver that solves MINLP models, but it resulted in unfeasible solutions for the model presented in this study. Running the model by using different solvers for LP and MILP did not result in different results.

One drawback of using a solver in this manner, is that the final step of the model operates as a black box, offering limited insight into the specific points of failure if the model becomes infeasible. This is a common issue in mathematical optimization, as mentioned in section 1.3.

The objective function of the model is defined as a cost function, containing costs of water from the water sources, costs of water treatment systems (CAPEX and OPEX), and optionally pipe costs. This generates more realistic networks than networks that solely minimize based on costs of water from water sources. However, if the emphasis should be more on minimizing the use of water and decreasing the production of used water, then the price of the water can be increased, to obtain a network where minimizing water is more essential.

6.2. Case Study Results & Solution Strategy

Table 6.1 gives a comparison of the calculated costs in this study with those calculated by Gunaratnam et al. (2005). While the expectation was that these would correspond, this is not the case, except for the Water Reuse Case Study. Nonetheless, all case studies show results in the same order of magnitude as Gunaratnam et al. (2005). The difference in the results is the most largest in scenario 3 in the Petroleum Refinery Case Study and the Water Treatment Case Study, as shown in table 6.1. These cases include the piping costs in the calculations, unlike the other case studies.

The smaller differences in the cases that exclude the piping costs, could arise from different factors. One potential cause is shortcomings in Gunaratnam et al. (2005). Mistakes were identified in the equations and the data presented in the paper, suggesting the possibility of additional inaccuracies.

Additionally, certain values were unspecified in Gunaratnam et al. (2005), leading to assumptions in this study regarding the upper and lower limits on the use of water from water sources, discharge and reuse flow rates, and the convergence criterion. Furthermore, the linearization procedure for the cost functions in the MILP was not elucidated, and the exact input requirements for the MINLP were

Table 6.1: Calculated costs in comparison with the results of Gunaratnam et al. (2005)

	Total annualized costs ($)	Costs in Gunaratnam et al. (2005)($)	Difference (%)
Petroleum Refinery Case Study			
Scenario 1	591926	578217	2.37
Scenario 2	679414	654245	3.85
Scenario 3	862984	683369	26.28
Water Reuse Case Study	580500	580500	0
Water Treatment Case Study	857792	705156	21.65

not specified. Through trial-and-error, it was discoverd that using concentrations from the MILP/LP iteration as initializing values, did not result in a MINLP solution. However, fixing the water flows from water sources from the iteration in the MINLP, with a boundary of plus and minus 2, enabled the MINLP model to provide an optimized solution.

Another potential explanation for the different results in table 6.1, is the use of different solvers. While SCIP was used to optimize the LP, MILP, and MINLP in this study, Gunaratnam et al. (2005) used OSL for the LP and MILP, and DICOPT for the MINLP. DICOPT operates on an outer-approximation algorithm (“DICOPT”, n.d.), whereas SCIP utilizes a branch-and-cut algorithm. The difference in solving methods could account for the varying results.

In the Water Reuse Case Study, the solution strategy presented in section 4.1, did not result in a feasible solution. The iteration failed to convert, resulting in no initialization point for the MINLP, so the case study could not be solved. However, an alternative strategy was applied to solve the Water Reuse Case Study. The binary variables of the water flows from the MILP in the initiation stage were directly used in the MINLP, and the total flows through the water-using operations were fixed. This resulted in a solution with the same costs as presented in (Gunaratnam et al., 2005) (2005).

Consequently, the solution strategy by Gunaratnam et al. (2005) implemented in the tool, does not always give the optimal solution. As mentioned in section 1.3, when solving non-convex MINLP, a global optimum often cannot be guaranteed, as seen in this study where local optima are presented. Exploring alternative initialization strategies may result in better local optimal solution, or even a global optimum.

6.3. Industrial Wastewater Treatment Techniques (IWTT) Database

Initially, the objective was to integrate the IWTT database with the model, so that based on selected industries and parameters, the treatment steps would automatically be coupled. However, the database lacks sufficient data to fulfill this purpose. Each industry presents multiple treatment trains, but these trains consist of data for different constituents. Therefore, the IWTT database serves as a resource for inspiration on potential treatment trains for different industries in this tool. The database is currently linked to the tool for this specific purpose. In future case studies where the tool is utilized, most data of the removal ratios of certain constituents that is needed will likely be missing, so these should be estimated based on literature or experience. Ideally, the database will be completed for all possible water constituents and treatment systems.

Additionally, cost estimations are not included in the IWTT database. The cost functions as described in section 4.1 are needed to estimate the costs of the water treatment systems as a function of the flow. However, while searching for these functions in literature, it is observed that they do not always conform to the structure presented in equation B.25 in appendix H (Guo et al., 2014; Hilbig et al., 2020; Molinos-Senante et al., 2012; Sharma, 2010; Sipala et al., 2003; Tsagarakis et al., 2003). Therefore, adjustments are required in the model to accommodate alternative forms of cost equations, or the cost equations themselves need to be modified to align with the structure in B.25.

7

Conclusions, Recommendations & Future Research

7.1. Key Conclusions

The primary objective of this research is to design a tool that identifies optimal connections to exchange water between different industrial plants, while introducing water treatment systems where useful or needed. It was decided to develop a tool that considers multiple constituents and one that is flexible if new functionalities are to be added, therefore the decision was made to develop a model using mixed-integer nonlinear programming (MINLP).

The literature study revealed a lack of open-source water network design tools that include water treatment systems and handle multiple constituents. Furthermore, there was a lack of tools integrating databases into their water network design model. This study introduces an open-source tool that includes water treatment systems and addressess multiple constituents. The solution strategy in the tool does not guarantee to provide the global optimal solution, but it provides local optimal solutions, as demonstrated in the presented case studies. This research also explores the utility of a database with water treatment systems, the IWTT database, in aiding in the design of industrial water networks. In the following sections some enhancements on the tool and possibilities for expansions are described.

7.2. Tool Enhancements

7.2.1. Database Enhancement

The current IWTT database, though partially usable, lacks data for many constituents and removal ratios. This leads to reliance on assumptions, especially in cost equations. To enhance the database, continuous updates are recommended, but using another database or creating a new one is also possible. Extending the collaboration with more companies and universities could provide valuable data for the database. Additionally, the inclusion of this data in (yearly) reports of companies could enhance the accessibility of the data and thus make it easier to extend the database.

7.2.2. Model Refinements in Existing Functions

Various improvements can be made to the existing model. Firstly, the initialization method, as described in the discussion. Additionally, the optimization process could be expanded to include different MINLP solvers to compare them. Comparing different methods of quantifying the removal of treatment operations, such as fixed outlet removal, can also be considered instead of the removal ratio. Furthermore, the inclusion of water quality characteristics in cost functions and the influence of thermal and pressure effects can be investigated. The tool could be enhanced with a visual interface as well, potentially in the form of a website, allowing it to be more user-friendly and easier to use.

7.3. Tool Expansions

There are several ideas for expanding the tool. Firstly, coupling site layout considerations to a further extent than done by Gunaratnam et al. (2005), who solely uses the distances between different plants. It could be coupled with geographic information systems (GIS) or perhaps functions could be included that automatically identify optimal connections, using external sources like bedrijvenopdekaart.nl, a website that links the location of companies to the type of industry (bedrijvenopdekaart.nl, n.d.).

One of the most crucial expansions is coupling heat integration into the tool through energy balances. This may significantly impact the costs and the environmental impact in the optimization of water networks. IN addition, future research should explore the dynamic coupling of energy conservation within the model, for example with the availability of electricity and water in different times and periods. Furthermore, the exchange of materials among different industries could be considered, especially the recovery of materials from water. This could be achieved by linking the tool to a database that identifies materials produced in specific industries.

The tool currently gives the most optimized option. Future research could explore presenting multiple scenarios to provide alternatives, so they can be compared. Furthermore, the tool could be enhanced by including environmental impact calculations, like lice cycle assessments. These could be included in the objective function as well. As well as including sensitivity analysis, to know what happens to the network when the situation changes, which is important for the resilience of the system.

To make the model more realistic, additional limitations could be included, such as pH range and concentration range for specific treatment systems. This data is often presented in the IWTT database already. Another way to make the tool more realistic, is by making it able to consider batch-wise processes instead of continuous processes only. This includes considering the inclusion of water storage options, exploring its costs and feasibility, including vessels or tanks, and aquifer storage and recovery.

8

Acknowledgement

I would like to express my deepest appreciation to my thesis committee. The meetings and feedback moments took place in a very pleasant and constructive atmosphere. A special thanks to Henri Spanjers, the chair of my committee, for offering numerous opportunities and providing valuable feedback. His efforts and the time he invested helped me a lot to improve the project. I always enjoyed our meetings and working on the industry water course was an enjoyable experience as well. I'm also very grateful to Laurens van der Schraaf, my supervisor from Sweco. He gave me the best possible introduction to the company, and helped me build a network within Sweco, leading to my first 'real' job there.

I extend my appreciation to my other thesis supervisors, Gijsbert Korevaar, Luuk Rietveld, and Alexander Garzón Díaz, whose valuable feedback helped me to improve the project and to gain different perspectives. I'd also like to thank Riccardo Taormina, who was part of the thesis committee at the start of my thesis, for the feedback he provided.

Furthermore, I'm grateful to the people from the Watertechnology Consultancy team at Sweco who took time to meet with me and brainstorm, especially Patricia Clevering. I'm eager to start working together with them.

I'd like to thank everyone from the team I did my internship in at Sweco as well, team Water Zuidwest, and other people I met during my internship. I saw them on a daily basis in the office in Rotterdam and they were always interested in my project and provided a supporting environment.

In addition, I'd like to thank my fellow students, especially the ones who were working on their thesis at the same time and therefore provided advice on some of the common challenges we encountered.

Lastly, I want to thank my family and friends for supporting me throughout the journey. Especially the care and support I received during the time I was sidelined due to a concussion. I would like to express particular gratitude to my sister and her fiance, whose help had a crucial role in shifting my perspective on the project when I was already accepting the scenario where my code would not work. Fortunately, I was able to resolve the issue due to their help.

Bibliography

Abraham, E. J., Al-Mohannadi, D. M., & Linke, P. (2022). Resource integration of industrial parks over time. *Computers and Chemical Engineering*, *164*. https://doi.org/10.1016/j.compchemeng.2022.107886

Ahmed, R., Shehab, S., Al-Mohannadi, D. M., & Linke, P. (2020). Synthesis of integrated processing clusters. *Chemical Engineering Science*, *227*. https://doi.org/10.1016/j.ces.2020.115922

AID, G., DAVIS, C., ZHU, B., KOREVAAR, G., KING, S., LOWITT, P., & BOBERG, N. (n.d.). *Isdata | industrial symbiosis data repository*. https://isdata.org/

Alnouri, S. Y., Linke, P., & El-Halwagi, M. (2014). Water integration in industrial zones: A spatial representation with direct recycle applications. *Clean Technologies and Environmental Policy*, *16*, 1637–1659. https://doi.org/10.1007/s10098-014-0739-2

Alnouri, S. Y., Linke, P., & El-Halwagi, M. M. (2016). Synthesis of industrial park water reuse networks considering treatment systems and merged connectivity options. *Computers and Chemical Engineering*, *91*, 289–306. https://doi.org/10.1016/j.compchemeng.2016.02.003

Alva-Argáez, A., Kokossis, A. C., & Smith, R. (1998). *Wastewater minimisation of industrial systems using an integrated approach*.

Alva-Argáez, A., Kokossis, A. C., & Smith, R. (2007). A conceptual decomposition of minlp models for the design of water-using systems. *Environment and Pollution*, *29*, 177–205.

Ampl. (n.d.). https://ampl.com/

Aviso, K. B. (2014). Design of robust water exchange networks for eco-industrial symbiosis. *Process Safety and Environmental Protection*, *92*, 160–170. https://doi.org/10.1016/j.psep.2012.12.001

Aviso, K. B., Tan, R. R., & Culaba, A. B. (2010). Designing eco-industrial water exchange networks using fuzzy mathematical programming. *Clean Technologies and Environmental Policy*, *12*, 353–363. https://doi.org/10.1007/s10098-009-0252-1

bedrijvenopdekaart.nl. (n.d.). *Alle sectoren en branches in nederland op de kaart*. https://bedrijvenopdekaart.nl/

Behera, C. R., Al, R., Gernaey, K. V., & Sin, G. (2020). A process synthesis tool for wwtp – an application to design sustainable energy recovery facilities. *Chemical Engineering Research and Design*, *156*, 353–370. https://doi.org/10.1016/j.cherd.2020.02.014

Bernal, D., & Peng, Z. (n.d.). *Mindtpy solver — pyomo 6.7.0 documentation*. https://pyomo.readthedocs.io/en/stable/contributed_packages/mindtpy.html

Bishnu, S. K., Linke, P., Alnouri, S. Y., & El-Halwagi, M. (2014). Multiperiod planning of optimal industrial city direct water reuse networks. *Industrial and Engineering Chemistry Research*, *53*, 8844–8865. https://doi.org/10.1021/ie5008932

Boix, M., Montastruc, L., Azzaro-Pantel, C., & Domenech, S. (2015). Optimization methods applied to the design of eco-industrial parks: A literature review. *Journal of Cleaner Production*, *87*, 303–317. https://doi.org/10.1016/j.jclepro.2014.09.032

Boix, M., Montastruc, L., Pibouleau, L., Azzaro-Pantel, C., & Domenech, S. (2011). A multiobjective optimization framework for multicontaminant industrial water network design. *Journal of Environmental Management*, *92*, 1802–1808. https://doi.org/10.1016/j.jenvman.2011.02.016

Boix, M., Montastruc, L., Pibouleau, L., Azzaro-Pantel, C., & Domenech, S. (2012). Industrial water management by multiobjective optimization: From individual to collective solution through eco-industrial parks. *Journal of Cleaner Production*, *22*, 85–97. https://doi.org/10.1016/j.jclepro.2011.09.011

Boix, M., Négny, S., Montastruc, L., & Mousqué, F. (2023). Flexible networks to promote the development of industrial symbioses: A new optimization procedure. *Computers and Chemical Engineering*, *169*. https://doi.org/10.1016/j.compchemeng.2022.108082

Capelleveen, G. V., Amrit, C., & Yazan, D. M. (2017). *A literature survey of information systems facilitating the identification of industrial symbiosis. in: Otjacques, b., hitzelberger, p., naumann, s.,*

wohlgemuth, v. (eds) from science to society. progress in is. Springer, Cham. https://doi.org/ https://doi.org/10.1007/978-3-319-65687-8_14

Centraal Bureau voor de Statistiek. (2023). *Watergebruik bedrijven en particuliere huishoudens nationale rekeningen*. https://www.cbs.nl/nl-nl/cijfers/detail/82883NED

Chen, C. L., Hung, S. W., & Lee, J. Y. (2010). Design of inter-plant water network with central and decentralized water mains. *Computers and Chemical Engineering*, *34*, 1522–1531. https://doi.org/10.1016/j.compchemeng.2010.02.024

Chertow, M. R. (2007). *"uncovering" industrial symbiosis* (1). https://doi.org/10.1162/jiec.2007.1110

Chew, I. M. L., & Foo, D. C. Y. (2009). Automated targeting for inter-plant water integration. *Chemical Engineering Journal*, *153*, 23–36. https://doi.org/10.1016/j.cej.2009.05.026

Chew, I. M. L., Foo, D. C. Y., Bonhivers, J. C., Stuart, P., Alva-Argaez, A., & Savulescu, L. E. (2013). A model-based approach for simultaneous water and energy reduction in a pulp and paper mill. *Applied Thermal Engineering*, *51*, 393–400. https://doi.org/10.1016/j.applthermaleng.2012.08.070

Chew, I. M. L., Foo, D. C. Y., Ng, D. K. S., & Tan, R. (2010). Flowrate targeting algorithm for interplant resource conservation network. part 1: Unassisted integration scheme. *Industrial and Engineering Chemistry Research*, *49*, 6439–6455. https://doi.org/10.1021/IE901802M

Chew, I. M. L., Foo, D. C. Y., & Tan, R. R. (2010). Flowrate targeting algorithm for interplant resource conservation network. part 2: Assisted integration scheme. *Industrial and Engineering Chemistry Research*, *49*, 6456–6468. https://doi.org/10.1021/ie901804z

Chew, I. M. L., Tan, R. R., Ng, D. K. S., Foo, D. C. Y., Majozi, T., & Gouws, J. (2008). Synthesis of direct and indirect interplant water network. *Industrial and Engineering Chemistry Research*, *47*, 9485–9496. https://doi.org/10.1021/ie800072r

Chew, I. M. L., Thillaivarrna, S. L., Tan, R. R., & Foo, D. C. Y. (2011). Analysis of inter-plant water integration with indirect integration schemes through game theory approach: Pareto optimal solution with interventions. *Clean Technologies and Environmental Policy*, *13*, 49–62. https://doi.org/10.1007/s10098-010-0280-x

Chin, H. H., Liew, P. Y., Varbanov, P., & Klemeš, J. J. (2020). Extension of pinch analysis to targeting and synthesis of multi-contaminant material recycle and reuse networks. https://doi.org/10.22541/au.160315137.74142765/v1

Chin, H. H., Varbanov, P. S., Klemeš, J. J., & Tan, R. R. (2022). Accounting for regional water recyclability or scarcity using machine learning and pinch analysis. *Journal of Cleaner Production*, *368*. https://doi.org/10.1016/j.jclepro.2022.133260

Chin, H. H., Varbanov, P. S., Liew, P. Y., & Klemeš, J. J. (2021). Pinch-based targeting methodology for multi-contaminant material recycle/reuse. *Chemical Engineering Science*, *230*. https://doi.org/10.1016/j.ces.2020.116129

Clevering, P., Mol, P., van der Schraaf, L., & Wesselingh, S. (2022). *Circulaire waterketen m4h: Prefeasibility studie*. Sweco.

Costa, A. R., & de Pinho, M. N. (2006). Performance and cost estimation of nanofiltration for surface water treatment in drinking water production. *Desalination*, *196*, 55–65. https://doi.org/10.1016/j.desal.2005.08.030

cottontail. (2023). *How to add trendline to a scatter plot*. https://stackoverflow.com/questions/26447191/how-to-add-trendline-to-a-scatter-plot

De Waterbank. (2023, December 1). *De waterbank - brengt de waterbalans terug in evenwicht*. https://www.dewaterbank.nl/

de Faria, D. C., de Souza, A. A. U., & de Arruda Guelli Ulson de Souza, S. M. (2009). Optimization of water networks in industrial processes. *Journal of Cleaner Production*, *17*, 857–862. https://doi.org/10.1016/j.jclepro.2008.12.012

De-León Almaraz, S., Boix, M., Montastruc, L., Azzaro-Pantel, C., Liao, Z., & Domenech, S. (2016). Design of a water allocation and energy network for multi-contaminant problems using multi-objective optimization. *Process Safety and Environmental Protection*, *103*, 348–364. https://doi.org/10.1016/j.psep.2016.03.015

De-León Almaraz, S., Boix, M., Azzaro-Pantel, C., Montastruc, L., & Domenech, S. (2015). Design of a multi-contaminant water allocation network using multi-objective optimization. *Computer Aided Chemical Engineering*, *37*, 911–916. https://doi.org/10.1016/B978-0-444-63578-5.50147-X

Dicopt. (n.d.). https://www.gams.com/latest/docs/S_DICOPT.html

Duhbacı, T. B., Özel, S., & Bulkan, S. (2021). Water and energy minimization in industrial processes through mathematical programming: A literature review. *Journal of Cleaner Production*, *284*. https://doi.org/10.1016/j.jclepro.2020.124752

Erkman, S. (1997). *Industrial ecology: An historical view* (2).

European Commission. (2020). *Regulation (eu) 2020/741 of the european parliament and of the council of 25 may 2020 on minimum requirements for water reuse*. Elsevier B.V.

Eurostat. (2006). *Nace rev. 2 introductory guidelines*.

Fadzil, A. F. A., Alwi, S. R. W., Manan, Z. A., & Klemeš, J. J. (2020). Study on impacts of multiple centralised water reuse header from consumer and operator perspectives. *Journal of Sustainable Development of Energy, Water and Environment Systems*, *8*, 754–765. https://doi.org/10.13044/j.sdewes.d7.0299

Fadzil, A. F. A., Alwi, S. R. W., Manan, Z., & Klemeš, J. J. (2018). Industrial site water minimisation via one-way centralised water reuse header. *Journal of Cleaner Production*, *200*, 174–187. https://doi.org/10.1016/j.jclepro.2018.07.193

Fadzil, A. F. A., Alwi, S. R. W., Manan, Z. A., & Klemeš, J. J. (2018). Maximizing total site water reuse via a two-way centralized water header. *ACS Sustainable Chemistry and Engineering*, *6*, 2563–2573. https://doi.org/10.1021/acssuschemeng.7b04050

Feng, X., Bai, J., Wang, H., & Zheng, X. (2008). Grass-roots design of regeneration recycling water networks. *Computers and Chemical Engineering*, *32*, 1892–1907. https://doi.org/10.1016/j.compchemeng.2007.10.006

Foo, D. C. Y. (2015). *Design of waste recovery systems with process integration*. KLM Technology Group.

Foo, D. C. Y. (2009). State-of-the-art review of pinch analysis techniques for water network synthesis. *Industrial and Engineering Chemistry Research*, *48*, 5125–5159. https://doi.org/10.1021/ie801264c

Foo, D. C. Y. (2012). *Process integration for resource conservation*. CRC Press.

Foong, S. Z., & Ng, D. K. (2021). Simultaneous design and integration of multiple processes for eco-industrial park development. *Journal of Cleaner Production*, *298*. https://doi.org/10.1016/j.jclepro.2021.126797

Fouladi, J., AlNouss, A., & Al-Ansari, T. (2021). Optimising the sustainability performance of an industrial park: An energy-water-food nexus. *Computer Aided Chemical Engineering*, *50*, 1505–1510. https://doi.org/10.1016/B978-0-323-88506-5.50232-1

GAMS Software GmbH. (n.d.). *Gams - cutting edge modeling*. https://www.gams.com/

Generous, M. M., Qasem, N. A., Akbar, U. A., & Zubair, S. M. (2021). Techno-economic assessment of electrodialysis and reverse osmosis desalination plants. *Separation and Purification Technology*, *272*. https://doi.org/10.1016/j.seppur.2021.118875

Geng, Y., Côté, R., & Tsuyoshi, F. (2007). A quantitative water resource planning and management model for an industrial park level. *Regional Environmental Change*, *7*, 123–135. https://doi.org/10.1007/s10113-007-0026-4

Getting started with pyomo. (n.d.). https://ndcbe.github.io/CBE60499/01.00-Pyomo-Introduction.html

Gunaratnam, M., Alva-Argáez, A., Kokossis, A., Kim, J. K., & Smith, H. (2005). Automated design of total water systems. *Industrial and Engineering Chemistry Research*, *44*, 588–599. https://doi.org/10.1021/ie040092r

Guo, T., Englehardt, J., & Wu, T. (2014). Review of cost versus scale: Water and wastewater treatment and reuse processes. *Water Science and Technology*, *69*, 223–234. https://doi.org/10.2166/wst.2013.734

Hilbig, J., Boysen, B., Wolfsdorf, P., & Rudolph, K. U. (2020). Economic evaluation of different treatment options for water reuse in industrial parks using modular cost functions. *Journal of Water Reuse and Desalination*, *10*, 419–430. https://doi.org/10.2166/wrd.2020.032

Hipólito-Valencia, B. J., Lira-Barragán, L. F., Ponce-Ortega, J. M., Serna-González, M., & El-Halwagi, M. M. (2014). Multiobjective design of interplant trigeneration systems. *AIChE Journal*, *60*, 213–236. https://doi.org/10.1002/aic.14292

Hong, X., Liao, Z., Sun, J., Jiang, B., Wang, J., & Yang, Y. (2018). Energy and water management for industrial large-scale water networks: A systematic simultaneous optimization approach. *ACS Sustainable Chemistry and Engineering*, *6*, 2269–2282. https://doi.org/10.1021/acssuschemeng.7b03740

Huang, X., Luo, X., Chen, J., Yang, Z., Chen, Y., Ponce-Ortega, J. M., & El-Halwagi, M. M. (2018). Synthesis and dual-objective optimization of industrial combined heat and power plants compromising the water-energy nexus. *Applied Energy*, *224*, 448–468. https://doi.org/10.1016/j.apenergy.2018.04.095

Ibrić, N., Ahmetović, E., Nemet, A., Kravanja, Z., & Grossmann, I. E. (2022). Synthesis of heat-integrated water networks using a modified heat exchanger network superstructure. *Energies*, *15*. https://doi.org/10.3390/en15093158

Industrial Wastewater Treatment Technology Database (IWTT) | US EPA. (2023). https://www.epa.gov/eg/industrial-wastewater-treatment-technology-database-iwtt

Industrial Wastewater Treatment Technology Database (IWTT) | US EPA. (2021). https://watersgeo.epa.gov/iwtt/guided-search

International Syngergies. (n.d.). *Projects archive | international synergies*. https://international-synergies.com/our-projects/

Iris Technology Solutions. (n.d.). *Sharebox – seamless sharing*. https://sharebox-project.eu/

Jeżowski, J. (2010). Review of water network design methods with literature annotations. *Industrial and Engineering Chemistry Research*, *49*, 4475–4516. https://doi.org/10.1021/ie901632w

Kalundborg Symbiosis. (2022, October 13). *50 years of circular production - kalundborg symbiosis*. https://www.symbiosis.dk/en/50-years-of-circular-production/

Karagiannis, I. C., & Soldatos, P. G. (2008). Water desalination cost literature: Review and assessment. *Desalination*, *223*, 448–456. https://doi.org/10.1016/j.desal.2007.02.071

kcoskun. (2021). *Drawing multiple edges between two nodes with networkx*. https://stackoverflow.com/questions/22785849/drawing-multiple-edges-between-two-nodes-with-networkx

Keckler, S. E., & Allen, D. T. (1998). Material reuse modeling: A case study of water reuse in an industrial park. *Journal of Industrial Ecology*, *2*, 79–92. https://doi.org/10.1162/jiec.1998.2.4.79

Khor, C. S., Chachuat, B., & Shah, N. (2014). Optimization of water network synthesis for single-site and continuous processes: Milestones, challenges, and future directions. *Industrial and Engineering Chemistry Research*, *53*, 10257–10275. https://doi.org/10.1021/ie4039482

Klemes, J. J. (2012). Industrial water recycle/reuse. *Current Opinion in Chemical Engineering*, *1*, 238–245. https://doi.org/10.1016/j.coche.2012.03.010

Kuo, W. C., & Smith, R. (1998). Design of water-using systems involving regeneration. *Process Safety and Environmental Protection*, *76*, 94–114. https://doi.org/10.1205/095758298529399

Lawal, M., Alwi, S. R. W., Manan, Z. A., & Ho, W. S. (2020). Industrial symbiosis tools—a review. *Journal of Cleaner Production*, *280*. https://doi.org/10.1016/j.jclepro.2020.124327

Le, T. M., Kujawa-Roeleveld, K., Tran, D. T., & Rijnaarts, H. H. (2022). Data envelopment analysis as a tool to assess the water demand minimization potential in industrial zones in the vietnamese delta. *Water Resources and Industry*, *28*. https://doi.org/10.1016/j.wri.2022.100181

Leong, Y. T., Tan, R. R., & Chew, I. M. L. (2014). Optimization of chilled and cooling water systems in a centralized utility hub. *Energy Procedia*, *61*, 846–849. https://doi.org/10.1016/j.egypro.2014.11.979

Li, A. H., Yang, Y. Z., & Liu, Z. Y. (2015). Analysis of water-using networks with multiple contaminants involving regeneration recycling. *Chemical Engineering Science*, *134*, 44–56. https://doi.org/10.1016/j.ces.2015.04.044

Liao, Z., Rong, G., Wang, J., & Yang, Y. (2011). Systematic optimization of heat-integrated water allocation networks. *Industrial and Engineering Chemistry Research*, *50*, 6713–6727. https://doi.org/10.1021/ie1016392

Liao, Z., Wu, J., Jiang, B., Wang, J., & Yang, Y. (2007). Design methodology for flexible multiple plant water networks. *Industrial and Engineering Chemistry Research*, *46*, 4954–4963. https://doi.org/10.1021/ie061299i

Lim, S. R., & Park, J. M. (2010). Interfactory and intrafactory water network system to remodel a conventional industrial park to a green eco-industrial park. *Industrial and Engineering Chemistry Research*, *49*, 1351–1358. https://doi.org/10.1021/ie9014233

LINDO Systems Inc. (2016). Lingo: The modeling language and optimizer. https://www.lindo.com/

Liu, C., Côté, R. P., & Zhang, K. (2015). Implementing a three-level approach in industrial symbiosis. *Journal of Cleaner Production*, *87*, 318–327. https://doi.org/10.1016/j.jclepro.2014.09.067

Liu, Z. Y., Yang, Y., Wan, L. Z., Wang, X., & Hou, K. H. (2009). A heuristic design procedure for water-using networks with multiple contaminants. *AIChE Journal*, *55*, 374–382. https://doi.org/10.1002/aic.11693

Lovelady, E. M., & El-Halwagi, M. M. (2009). Design and integration of eco-industrial parks for managing water resources. *Environmental Progress and Sustainable Energy*, *28*, 265–272. https://doi.org/10.1002/ep.10326

Martyniuk, J. (2022, July 15). *What is sdlc? phases, models and tools — devox software*. https://devoxsoftware.com/blog/software-development-lifecycle/

Massard, G., & Erkman, S. (2007). *A regional industrial symbiosis methodology and its implementation in geneva, switzerland*.

Mohammadnejad, S., Ataei, A., Bidhendi, G. R. N., Mehrdadi, N., Ebadati, F., & Lotfi, F. (2012). Water pinch analysis for water and wastewater minimization in tehran oil refinery considering three contaminants. *Environmental Monitoring and Assessment*, *184*, 2709–2728. https://doi.org/10.1007/s10661-011-2146-z

Molinos-Senante, M., Garrido-Baserba, M., Reif, R., Hernández-Sancho, F., & Poch, M. (2012). Assessment of wastewater treatment plant design for small communities: Environmental and economic aspects. *Science of the Total Environment*, *427-428*, 11–18. https://doi.org/10.1016/j.scitotenv.2012.04.023

Montastruc, L., Boix, M., Pibouleau, L., Azzaro-Pantel, C., & Domenech, S. (2013). On the flexibility of an eco-industrial park (eip) for managing industrial water. *Journal of Cleaner Production*, *43*, 1–11. https://doi.org/10.1016/j.jclepro.2012.12.039

Mu, D., Xin, C., & Zhou, W. (2019). Life cycle assessment and techno-economic analysis of algal biofuel production, 281–292. https://doi.org/10.1016/B978-0-12-817536-1.00018-7

Nemati-Amirkolaii, K. (2021). *Optimization of industrial water networks. development of generic design tools and application in a real industrial site*. Université Paris-Saclay. https://pastel.archives-ouvertes.fr/tel-03766337

Nemati-Amirkolaii, K., Romdhana, H., & Lameloise, M. L. (2021). A novel user-friendly tool for minimizing water use in processing industry. *Cleaner Engineering and Technology*, *4*. https://doi.org/10.1016/j.clet.2021.100260

Neves, A., Godina, R., Azevedo, S. G., & Matias, J. C. (2020). A comprehensive review of industrial symbiosis. *Journal of Cleaner Production*, *247*. https://doi.org/10.1016/j.jclepro.2019.119113

Ng, D. K. S., Chew, I. M. L., Tan, R. R., Foo, D. C. Y., Ooi, M. B., & El-Halwagi, M. M. (2014). Rcnet: An optimisation software for the synthesis of resource conservation networks. *Process Safety and Environmental Protection*, *92*, 917–928. https://doi.org/10.1016/j.psep.2013.10.006

Ng, D. K. S., Foo, D. C. Y., & Tan, R. (2009). Automated targeting technique for single-impurity resource conservation networks. part 1: Direct reuse/recycle. *Industrial and Engineering Chemistry Research*, *48*, 7637–7646. https://doi.org/10.1021/ie900120y

O'Dwyer, E., Chen, K., Wang, H., Wang, A., Shah, N., & Guo, M. (2020). Optimisation of wastewater treatment strategies in eco-industrial parks: Technology, location and transport. *Chemical Engineering Journal*, *381*. https://doi.org/10.1016/j.cej.2019.122643

Olesen, S., & Polley, G. (1996). Dealing with plant geography and piping constraints in water network design. *Trans IChemE*, *74*.

Oxford University Press. (2023). *Definition of symbiosis noun from the oxford advanced learner's dictionary*. https://www.oxfordlearnersdictionaries.com/definition/english/symbiosis?q=symbiosishttps://www.oxfordlearnersdictionaries.com/definition/english/symbiosis?q=symbiosis

Pan, C., Shi, J., & Liu, Z. Y. (2012). An iterative method for design of water-using networks with regeneration recycling. *AIChE Journal*, *58*, 456–465. https://doi.org/10.1002/aic.12595

Pham, T. T., Mai, T. D., Pham, T. D., Hoang, M. T., Nguyen, M. K., & Pham, T. T. (2016). Industrial water mass balance as a tool for water management in industrial parks. *Water Resources and Industry*, *13*, 14–21. https://doi.org/10.1016/j.wri.2016.04.001

Poplewski, G., Jezowski, J. M., & Jezowska, A. (2011). Water network design with stochastic optimization approach. *Chemical Engineering Research and Design*, *89*, 2085–2101. https://doi.org/10.1016/j.cherd.2010.12.016

Putra, Z. A., & Amminudin, K. A. (2008). Two-step optimization approach for design of a total water system. *Industrial and Engineering Chemistry Research*, *47*, 6045–6057. https://doi.org/10.1021/ie071620c

Relvas, S., Matos, H. A., Fernandes, M. C., Castro, P., & Nunes, C. P. (2008). Aquomin: A software tool for mass-exchange networks targeting and design. *Computers and Chemical Engineering*, *32*, 1085–1105. https://doi.org/10.1016/j.compchemeng.2006.12.002

Ritchie, H., & Roser, M. (2017). Water use and stress [https://ourworldindata.org/water-use-stress]. *Our World in Data*.

Rubio-Castro, E., Ponce-Ortega, J. M., Nápoles-Rivera, F., El-Halwagi, M. M., Serna-González, M., & Jiménez-Gutiérrez, A. (2010). Water integration of eco-industrial parks using a global optimization approach. *Industrial and Engineering Chemistry Research*, *49*, 9945–9960. https://doi.org/10.1021/ie100762u

Rubio-Castro, E., Ponce-Ortega, J. M., Serna-González, M., & El-Halwagi, M. M. (2012). Optimal reconfiguration of multi-plant water networks into an eco-industrial park. *Computers and Chemical Engineering*, *44*, 58–83. https://doi.org/10.1016/j.compchemeng.2012.05.004

Rubio-Castro, E., Ponce-Ortega, J. M., Serna-González, M., El-Halwagi, M. M., & Pham, V. (2013). Global optimization in property-based interplant water integration. *AIChE Journal*, *59*, 813–833. https://doi.org/10.1002/aic.13874

Rubio-Castro, E., Ponce-Ortega, J. M., Serna-González, M., Jiménez-Gutiérrez, A., & El-Halwagi, M. M. (2011). A global optimal formulation for the water integration in eco-industrial parks considering multiple pollutants. *Computers and Chemical Engineering*, *35*, 1558–1574. https://doi.org/10.1016/j.compchemeng.2011.03.010

Scip. (n.d.). https://www.scipopt.org/

Sharma, J. R. (2010). *Development of a preliminary cost estimation method for water treatment plants*. https://www.academia.edu/22909825/DEVELOPMENT_OF_A_PRELIMINARY_COST_ESTIMATION_METHOD_FOR_WATER_TREATMENT_PLANTS

Short, M. (n.d.). *Github - mchlshort/mexnets: Mass exchanger network synthesis using pyomo*. https://github.com/mchlshort/MExNetS

Short, M., Isafiade, A. J., Biegler, L. T., & Kravanja, Z. (2018). Synthesis of mass exchanger networks in a two-step hybrid optimization strategy. *Chemical Engineering Science*, *178*, 118–135. https://doi.org/10.1016/j.ces.2017.12.019

Sipala, S., Mancini, G., & Vagliasindi, F. G. A. (2003). Development of a web-based tool for the calculation of costs of different wastewater treatment and reuse scenarios. http://iwaponline.com/ws/article-pdf/3/4/89/407599/89.pdf

Sotelo-Pichardo, C., Ponce-Ortega, J. M., El-Halwagi, M. M., & Frausto-Hernández, S. (2011). Optimal retrofit of water conservation networks. *Journal of Cleaner Production*, *19*, 1560–1581. https://doi.org/10.1016/j.jclepro.2011.05.011

Spanjers, H. (2022). *Industry water - module m6 water treatment technologies: Introduction*. Delft University of Technology.

Su, W. N., Li, Q. H., Liu, Z. Y., & Pan, C. H. (2012). A new design method for water-using network of multiple contaminants with single internal water main. *Journal of Cleaner Production*, *29-30*, 38–45. https://doi.org/10.1016/j.jclepro.2012.01.041

Symbiosis4growth. (2023, May 25). *Home - symbiosis4growth*. https://www.symbiosis4growth.nl/

Teles, J. P., Castro, P. M., & Novais, A. Q. (2007). Optwatnet □ a software for the optimal design of water-using networks with multi-contaminants. *Computer Aided Chemical Engineering*, *24*, 497–502. https://doi.org/10.1016/S1570-7946(07)80106-4

Tiu, B. T. C., & Cruz, D. E. (2017). An milp model for optimizing water exchanges in eco-industrial parks considering water quality. *Resources, Conservation and Recycling*, *119*, 89–96. https://doi.org/10.1016/j.resconrec.2016.06.005

Tokos, H., & Pintarič, Z. N. (2012). Development of a minlp model for the optimization of a large industrial water system. *Optimization and Engineering*, *13*, 625–662. https://doi.org/10.1007/s11081-011-9162-2

Tsagarakis, K., Mara, D., & Angelakis, A. (2003). Application of cost criteria for selection of municipal wastewater treatment systems. *Water, Air, and Soil Pollution*, *142*, 187–210. https://doi.org/10.1023/A:1022032232487

United Nations. Statistical Division. (2008). *International standard industrial classification of all economic activities (isic)*. United Nations.

Wang, B., Feng, X., & Zhang, Z. (2003). A design methodology for multiple-contaminant water networks with single internal water main. *Computers and Chemical Engineering*, *27*, 903–911. https://doi.org/10.1016/S0098-1354(02)00177-1

Wang & Smith, R. (1994). Wastewater minimisation. *Chemical Engineering Science*, *49*, 981–1006. https://doi.org/10.1016/0009-2509(94)80006-5

Wang, X., Fan, X., & Liu, Z. Y. (2019). Design of interplant water network of multiple contaminants with an interplant water main. *Chemical Engineering Transactions*, *72*, 295–300. https://doi.org/10.3303/CET1972050

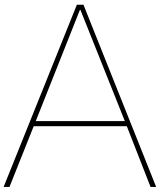
Water Europe Marketplace. (n.d.). *Water europe marketplace*. https://mp.watereurope.eu/

Water Watch - CDP Water Impact Index - CDP. (2022). https://www.cdp.net/en/investor/water-watch-cdp-water-impact-index

Yeo, Z., Masi, D., Low, J. S. C., Ng, Y. T., Tan, P. S., & Barnes, S. (2019). Tools for promoting industrial symbiosis: A systematic review. *Journal of Industrial Ecology*, *23*, 1087–1108. https://doi.org/10.1111/jiec.12846

Yoo, C., Lee, T. Y., Kim, J., Moon, I., Jung, J. H., Han, C., Oh, J.-M., & Lee, I.-B. (2007). Integrated water resource management through water reuse network design for clean production technology: State of the art. *Korean J. Chem. Eng.*, *24*.

Zhou, L., Liao, Z., Wang, J., Jiang, B., Yang, Y., & Yu, H. (2015). Simultaneous optimization of heat-integrated water allocation networks using the mathematical model with equilibrium constraints strategy. *Industrial and Engineering Chemistry Research*, *54*, 3355–3366. https://doi.org/10.1021/ie501960e



Overview of Manufacturing Industries

The United Nations made an international standard industrial classification of all economic activities (ISIC), which is divided in different sections. ISIC is based on the European classification NACE and is the same as ISIC on high classfication levels, but NACE is more detailed at lower levels (Eurostat, 2006). Section C is the manufacturing industry in both categorization methods and consists of 24 divisions that are shown below (United Nations. Statistical Division., 2008).

- Manufacture of food products
- Manufacture of beverages
- Manufacture of tobacco products
- Manufacture of textiles
- Manufacture of wearing apparel
- Manufacture of leather and related products
- Manufacture of wood and of products of wood and cork, except furniture; manufacture of articles of straw and plaiting materials
- Manufacture of paper and paper products
- Printing and reproduction of recorded media
- Manufacture of coke and refined petroleum products
- Manufacture of chemicals and chemical products
- Manufacture of basic pharmaceutical products and pharmaceutical preparations
- Manufacture of rubber and plastic products
- Manufacture of other non-metallic mineral products
- Manufacture of basic metals
- Manufacture of fabricated metal products, except machinery and equipment
- Manufacture of computer, electronic and optical products
- Manufacture of electrical equipment
- Manufacture of machinery and equipment n.e.c.
- Manufacture of motor vehicles, trailers and semi-trailers
- Manufacture of other transport equipment
- Manufacture of furniture
- Other manufacturing
- Repair and installation of machinery and equipment

All these categories are categorized further. The Carbon Disclosure Project (CDP), an international non-profit organisation, quantified the environmental impact of more than 200 different types of industrial activities based on the NACE classification. CDP made a tool that ranks industries according to their potential impact on freshwater resources, both relying on quantity and quality, which is called 'Water Watch - CDP Water Impact Index'. ("Water Watch - CDP Water Impact Index - CDP", 2022)

The ranking is done by ranking the different value chain stages (direct operations, supply chain and product use) with a number from 0 to 3 for both 'the dependence of the activity on high volumes of freshwater withdrawals or consumption' and 'the water pollution or degradation potential of the activity'. The total rank of an industrial activity ranges therefore from 0 (no impact) to 18 (high impact). ("Water Watch - CDP Water Impact Index - CDP", 2022).

The following methodology is used to make a selection for the industrial activities that are relevant for this research. First, CDP Industries and CDP Activity Group that cannot contribute in industrial symbiosis in eco-industrial parks are excluded. The irrelevant CDP Industries are 'fossil fuels, hospitality, infrastructure, international bodies, retail, and transportation services'. The following CDP Activity Groups are excluded in addition: 'commercial & consumer services, crop farming, financial services, fish & animal farming, health care provision, industrial support services, IT & software development, logging & rubber tapping, metallic mineral mining, other mineral mining, other services, print & publishing services, specialized professional services, specialized professional services, and web & marketing services.

After this selection, the list was ordered from highest to lowest overall water impact rate. The following categories are considered to have a critical water impact (water impact rate between 15 and 18):

- Textiles & fabric goods
  - Apparel design & manufacturing
  - Textiles
- Chemicals
  - Inorganic base chemicals
  - Agricultural chemicals
  - Nitrogenous fertilizers
  - Non-nitrogenous fertilizers
  - Other base chemicals
  - Personal care & household products
  - Specialty chemicals
  - Basic plastics
- Electrical & electronic equipment
  - Semiconductors
  - Electronic components
- Transportation equipment: Alternative vehicles
- Metal smelting, refining & reforming
  - Iron & steel
  - Aluminum
  - Copper
  - Metal processing
  - Other non-ferrous metals
  - Precious metals
- Biotech & pharma: pharmaceuticals
- Food & beverage processing: soybean processing
- Metal products manufacturing: fabricated metal components

The following categories were considered to have a very high water impact (water impact rate between 11 and 14):

- Food & beverage processing:
  - Palm oil processing
  - Sugar
  - Animal processing
  - Dairy & egg products
  - Oilseed processing
  - Alcoholic beverages
  - Chocolate confection
  - Coffee
  - Fruit, nut & vegetable processing
  - Non-alcoholic beverages
  - Non-chocolate confection
  - Other food processing
  - Tea
- Tobacco: Tobacco products
- Metal products manufacturing: Metal containers & packaging
- Powered machinery
  - Engines & motors
  - Agriculture, construction & mining machinery
  - Industrial machinery

Other vehicle equipment & systems

- Transportation equipment: Aerospace
- Medical equipment & supplies: Medical equipment
- Electrical & electronic equipment
  - Batteries
  - Household appliances
  - Communications equipment
  - Computer hardware
  - Electrical equipment
  - Electronic equipment
- Plastic product manufacturing: Plastic products
- Transportation equipment
  - Automobiles
  - Heavy vehicles
  - Railroad rolling stock
  - Recreational vehicles
  - Shipbuilding
- Cement & concrete
  - Cement
  - Concrete products
- Textiles & fabric goods: Luggage & bags
- Renewable energy equipment
  - Solar energy equipment
  - Other renewable energy equipment
- Wood & rubber products: Rubber products
- Chemicals: Biofuels
- Wood & paper materials: Pulp & paper mills
- Renewable power generation: Hydro generation
- Thermal power generation
  - Non-CCGT generation
  - CCGT generation
  - Coal generation
- Light manufacturing: Tires
- Paper products & packaging: Paper products
- Nuclear power generation: Nuclear generation
- Biomass generation: Biomass generation

The following activity groups were considered to have a high water impact (between 8 and 10):

- Medical equipment & supplies Health care supplies
- Food & beverage processing
  - Baked goods & cereals
  - Grain & corn milling
  - Seafood processing
- Leisure & home manufacturing: Furniture
- Paper products & packaging: Paper packaging
- Other materials: Ceramics
- Wood & paper materials: Sawmills & wood materials
- Media, telecommunications & data center services: Servers & data centers
- Leisure & home manufacturing
  - Accessories
  - Homeware
  - Toys & games
- Light manufacturing
  - Automotive interior
  - Munitions
- Other materials: Other non-wood building materials
- Renewable power generation: Solar generation

B

Model Variables & Equations

B.1. Model Variables

The variables are taken from Gunaratnam et al. (2005). First, a set of constraints is shown that is the basis of the design and optimization problem. These constraints are used to formulate continuous and integer variables. Improvements are made in the formulation of Gunaratnam et al. (2005), which are shown in the footnotes.

B.1.1. Sets

C = {c|c is a contaminant present in the water}, c = 1, 2, ..., N_c

S = {s|s is a freshwater source available}, s = 1, 2, ..., N_S

E = {e|e is a wastewater discharge point}, e = 1, 2, ..., N_E

U = {u|u is a process operation within the water network}, u = 1, 2, ..., N_{TU}

WU = {wu|wu is a water-using operation within the water network}, wu = 1, 2, ..., N_{WU}

TU = {tu|tu is a treatment unit within the water network}, tu = 1, 2, ..., N_{TU}

$WU + TU = U$ [1]

B.1.2. Decision Variables

Continuous variables associated with flow rates

$F_{s,u}^{w}$ = freshwater flow from a freshwater source $s \in S$ to operation $u \in U$

F_{u}^{t} = total flow through operation $u \in U$

$F_{u,ua}^{ua}$ = flow from operation $u \in U$ to operation $ua \in U$

$F_{u,e}^{out}$ = flow from operation $u \in U$ to discharge point $e \in E$

Contaminant concentration and mass flow in process operations

$C_{c,u}^{out}$ = concentration $c \in C$ in streams leaving operation $u \in U$

$M_{c,u}^{in}$ = mass flow entering operation $u \in U$

$M_{c,u}^{out}$ = mass flow leaving operation $u \in U$

$M_{c,u}^{loss}$ = mass flow loss from operation $u \in U$

$M_{c,u}^{gain}$ = mass flow gain in operation $u \in U$

[1] This is the correction of the following: $WU >> TU = U$

Cross-sectional area of the pipes connecting different operations

$A_{s,u}^{fw}$ = pipe connections between freshwater source $s \in S$ and operation $u \in U$

$A_{u,ua}^{ua}$ = pipe connections between operation $u \in U$ and operation $ua \in U$

$A_{u,e}^{out}$ = pipe connections between operation $u \in U$ and discharge point $e \in E$

Cost terms in the objective function

$Cost_s^{fw}$ = cost of freshwater supply $s \in S$

$Cost_u^{tu}$ = cost of water treatment $tu \in TU$

$Cost_{s,u}^{fw,pipe}$ = piping cost from freshwater source $s \in S$ to operation $u \in U$

$Cost_{u,ua}^{ua,pipe}$ = piping cost from operation $u \in U$ to operation $ua \in U$

$Cost_{u,e}^{out,pipe}$ = piping cost from operation $u \in U$ to discharge point $e \in E$

O^{cost} = total annualized cost

Binary variables related to existence and/or nonexistence of connections

$B_{s,u}^{fw}$ = stream from freshwater source $s \in S$ to operation $u \in U$

$B_{u,ua}^{ua}$ = stream from operation $u \in U$ to operation $ua \in U$

$B_{u,e}^{out}$ = stream from operation $u \in U$ to operation $e \in E$

$B_{u,ua}^{G1}$ = operation $u \in U$ belongs to group 1 with respect to treatment $tu \in TU$

$B_{u,ua}^{G2}$ = operation $u \in U$ belongs to group 2 with respect to treatment $tu \in TU$

B.1.3. Parameters

Concentration bounds

$C_{c,u}^{in,max}$ = maximum inlet concentration $c \in C$ to operation $u \in U$

$C_{c,u}^{out,max}$ = maximum outlet concentration $c \in C$ from operation $u \in U$

Distances between water sources and sinks

$d_{s,u}^{fw}$ = distance between freshwater source $s \in S$ and operation $u \in U$

$d_{u,ua}^{ua}$ = distance between operation $u \in U$ and operation $ua \in U$

$d_{u,e}^{out}$ = distance between operation $u \in U$ and discharge point $e \in E$

Flow velocities within the pipe connections

$v_{s,u}^{fw}$ = velocity in pipes between freshwater source $s \in S$ and operation $u \in U$

$v_{u,ua}^{ua}$ = velocity in pipes between operation $u \in U$ and operation $ua \in U$

$v_{u,e}^{out}$ = velocity in pipes between operation $u \in U$ and discharge point $e \in E$

Regression parameters for piping costs

$a_{s,u}^{fw}$ = freshwater flow from source $s \in S$ to operation $u \in U$

$b_{s,u}^{fw}$ = freshwater flow from source $s \in S$ to operation $u \in U$

$a_{u,ua}^{ua}$ = flow from operation $u \in U$ to operation $ua \in U$

$b_{u,ua}^{ua}$ = flow from operation $u \in U$ to operation $ua \in U$

$a_{u,e}^{out}$ = flow through operation $u \in U$ to discharge point $e \in E$

$b_{u,e}^{out}$ = flow through operation $u \in U$ to discharge point $e \in E$ [2]

Other parameters

$C_{c,s}^{fw}$ = concentration of contaminant $c \in C$ in freshwater source $s \in S$

C_c^{env} = environmental discharge limit on contaminant $c \in C$

$C_{c,u}^{loss}$ = concentration $c \in C$ at which the loss occurs in operation $u \in U$

F_u^{loss} = water flow-rate loss from operation $u \in U$

$L_{c,u}^{ml}$ = limiting mass load of contaminant $c \in C$ from operation $wu \in WU$

$RR_{c,u}$ = removal ratio of contaminant $c \in C$ in treatment $tu \in TU$ $(0 \leq RR_{c,u} < 1)$

$C_{c,u}^{con,k}$ = fixed concentration c in streams leaving operation $u \in U$

$C_{s,u}^{fw,k}$ = fixed concentration in freshwater source $s \in S$ to operation $u \in U$

$C_{u,ua}^{fua,k}$ = fixed concentration in streams from operation $u \in U$ to operation $ua \in U$

CO_s^{fw} = cost per unit volume of freshwater from source $s \in S$

NS_u^{max} = scalar value for the maximum number of sources into operation $u \in U$

$F_u^{t,max}$ = maximum water flow rate in operation $u \in U$

$F_u^{t,min}$ = minimum water flow rate in operation $u \in U$

B.2. Model Equations

All equations are taken from Gunaratnam et al. (2005), with small corrections in equations B.1, B.12, B.17, B.22, B.28, B.29, B.30, B.33, and B.36, and large corrections in equations B.2 and B.20. An additional equation was added to B.17, to include a limit on the number of leaving streams leaving an operation besides the limit on the number of streams entering.

B.2.1. A: Balances around Operations, Mixers and Splitters

A1: Overall balance around the entire water system

$$\sum_{s \in S} \sum_{u \in U} F_{s,u}^{W} - \sum_{e \in E} \sum_{u \in U} F_{u,e}^{out} = \sum_{u \in U} F_u^{loss} \tag{B.1}$$

$$\sum_{ua \in U} F_{ua,u}^{ua} + \sum_{s \in S} F_{s,u}^{w} - \sum_{ua \in U} F_{u,ua}^{ua} - \sum_{e \in E} F_{u,e}^{out} = F_u^{loss} \tag{B.2}$$

A2: Contaminant mass balance for each operation (and each contaminant)

$$\sum_{ua \in U} (F_{ua,u}^{ua} C_{c,ua}^{out}) + \sum_{s \in S} (F_{s,u}^{w} C_{c,s}^{fw}) - M_{c,u}^{in} = 0 \tag{B.3}$$

$$C_{c,u}^{out} (\sum_{s \in S} F_{s,u}^{w} + \sum_{ua \in U} F_{ua,u}^{ua} - F_u^{loss}) - M_{c,u}^{out} = 0 \tag{B.4}$$

$$M_{c,u}^{in} - M_{c,u}^{out} + L_{c,u}^{ml} - F_u^{loss} C_{c,u}^{loss} = 0 \quad \forall \quad u \in WU \tag{B.5}$$

[2] Small correction

$$(1 - RR_{c,u})M_{c,u}^{in} - F_u^{loss}C_{c,u}^{loss} - M_{c,u}^{out} = 0 \quad \forall \quad u \in TU \tag{B.6}$$

B.2.2. B: Availability and Capacity Constraints

B1: Constraints of water flow rate

$$F_u^{t,min} \leq F_u^t, \text{ where } F_u^t - (\sum_{s \in S} F_{s,u}^w + \sum_{ua \in U} F_{ua,u}^{ua} - F_u^{loss}) = 0 \tag{B.7}$$

$$F_u^{t,max} \geq F_u^t, \text{ where } F_u^t - (\sum_{s \in S} F_{s,u}^w + \sum_{ua \in U} F_{ua,u}^{ua} - F_u^{loss}) = 0 \tag{B.8}$$

B2: Constraints on the quality and quantity of water

$$M_{c,u}^{in} \leq C_{c,u}^{in,max} F_u^t \tag{B.9}$$

B3: Constraint on the environmental discharge limit of contaminants

$$\sum_{u \in U} C_{c,u}^{out} F_{u,e}^{out} \leq C_c^{env} \sum_{u \in U} F_{u,e}^{out} \tag{B.10}$$

B.2.3. C: Logic Constraints

C1: Upper and lower bounds on the flow rates

$$F_{s,u}^w - U_{s,u}^{fw} B_{s,u}^{fw} \leq 0 \tag{B.11}$$

$$F_{s,u}^w - L_{s,u}^{fw} B_{s,u}^{fw} \geq 0 \tag{B.12}$$

$$F_{u,e}^{out} - U_{u,e}^{out} B_{u,e}^{out} \leq 0 \tag{B.13}$$

$$F_{u,e}^{out} - L_{u,e}^{out} B_{u,e}^{out} \geq 0 \tag{B.14}$$

$$F_{u,ua}^{ua} - U_{u,ua}^{ua} B_{u,ua}^{ua} \leq 0 \tag{B.15}$$

$$F_{u,ua}^{ua} - L_{u,ua}^{ua} B_{u,ua}^{ua} \geq 0 \tag{B.16}$$

C2: Maximum number of sources to feed each operation

$$\begin{aligned} \textstyle\sum_{ua \in U} B_{ua,u}^{ua} + \sum_{s \in S} B_{s,u}^{fw} &\leq NS_u^{max} \\ \textstyle\sum_{ua \in U} B_{u,ua}^{ua} + \sum_{e \in E} B_{u,e}^{out} &\leq NS_u^{max} \end{aligned} \tag{B.17}$$

C3: Elimination of regeneration recycling

$$B_{u,tu}^{ua} - B_{u,tu}^{G1} \leq 0 \tag{B.18}$$

$$B_{tu,u}^{ua} - B_{u,tu}^{G2} \leq 0 \tag{B.19}$$

$$B_{u,tu}^{G1} + B_{u,tu}^{G2} \leq 1 \tag{B.20}$$

$$2 - (B_{u,tu}^{G2} + B_{ua,tu}^{G2}) \geq B_{u,ua}^{ua} \tag{B.21}$$

$$\sum_{u \in U} B_{u,tu}^{G1} + \sum_{u \in U} B_{u,tu}^{G2} = N_{OP} - 1 \tag{B.22}$$

C4: Elimination of direct recycling

$$B_{u,ua}^{ua} + B_{ua,u}^{ua} \leq 1 \tag{B.23}$$

B.2.4. D: Objective Function

D1: Freshwater supply cost

$$Cost_s^{fw} = \sum_{u \in U} CO_s^{fw} F_{s,u}^{w} \tag{B.24}$$

D2: Water and wastewater treatment cost

$$Cost_u^{tu} = CO_u^{tu} F_u^{t\beta_u} \tag{B.25}$$

D3: Piping cost

$$F_{s,u}^{w} = A_{s,u}^{fw} v_{s,u}^{fw} \tag{B.26}$$

$$Cost_{s,u}^{fw,pipe} = [(a_{s,u}^{fw} A_{s,u}^{fw}) + (b_{s,u}^{fw} B_{s,u}^{fw})] d_{s,u}^{fw} \tag{B.27}$$

D4: Overall objective function

$$O^{cost} = (\sum_{s \in S} Cost_s^{fw}) + (\sum_{s \in S} \sum_{u \in U} Cost_{s,u}^{fw,pipe} + \sum_{u \in U} \sum_{ua \in U} Cost_{u,ua}^{ua,pipe} + \sum_{e \in E} \sum_{u \in U} Cost_{u,e}^{out,pipe}) + (\sum_{u \in TU} Cost_{tu}^{u}) \tag{B.28}$$

B.2.5. E: MILP Formulation

$$M_{c,u}^{in} - M_{c,u}^{out} + L_{c,u}^{ml} - F_u^{loss} C_{c,u}^{loss} + M_{c,u}^{loss} - M_{c,u}^{gain} = 0 \quad \forall \quad u \in WU \tag{B.29}$$

$$(1 - RR_{c,u}) M_{c,u}^{in} - F_u^{loss} C_{c,u}^{loss} - M_{c,u}^{out} - M_{c,u}^{gain} = 0 \quad \forall \quad u \in TU \tag{B.30}$$

$$\sum_{ua \in U} (F_{ua,u}^{ua} C_{c,ua}^{con,k}) + \sum_{s \in S} (F_{s,u}^{w} C_{c,s}^{fw}) - M_{c,u}^{in} = 0 \tag{B.31}$$

$$C_{c,u}^{con,k}(\sum_{s\in S} F_{s,u}^{w} + \sum_{ua\in U} F_{ua,u}^{ua} - F_{u}^{loss}) - M_{c,u}^{out} = 0 \tag{B.32}$$

$$O_{milp}^{cost} = (\sum_{s\in S} Cost_{s}^{fw}) + (\sum_{s\in S}\sum_{u\in U} Cost_{s,u}^{fw,pipe} + \sum_{u\in U}\sum_{ua\in U} Cost_{u,ua}^{ua,pipe} + \sum_{e\in E}\sum_{u\in U} Cost_{u,e}^{out,pipe}) + (\sum_{u\in TU} Cost_{tu}^{u}) \tag{B.33}$$

B.2.6. F: LP Formulation

$$\sum_{ua\in U} (F_{ua,u}^{ua,k} C_{c,ua}^{out}) + \sum_{s\in S} (F_{s,u}^{w,k} C_{c,s}^{fw}) - M_{c,u}^{in} = 0 \tag{B.34}$$

$$C_{c,u}^{out}(\sum_{s\in S} F_{s,u}^{w,k} + \sum_{ua\in U} F_{ua,u}^{ua,k} - F_{u}^{loss}) - M_{c,u}^{out} = 0 \tag{B.35}$$

$$M^{terms} = \sum_{c\in C}\sum_{u\in U} M_{c,u}^{loss} + \sum_{c\in C}\sum_{u\in U} M_{c,u}^{gain} \tag{B.36}$$

C

Industrial Wastewater Treatment Technology Database (IWTT)

The IWTT database is developed by the Environmental Protection Agency (EPA) of the United States. The database includes data from different sources, namely ”peer-reviewed journals, conference proceedings, industry-specific organization, and government reports” (Industrial Wastewater Treatment Technology Database (IWTT) | US EPA, 2021). In this database, it is possible to select treatment technologies that are used in specific types of industries. The concentrations of different water quality indicators in the influent and effluent in the treatment train are shown, as well as the removal ratio and the reference of the research. The database can be downloaded as a csv file.

The following water treatment technologies are considered in the database:

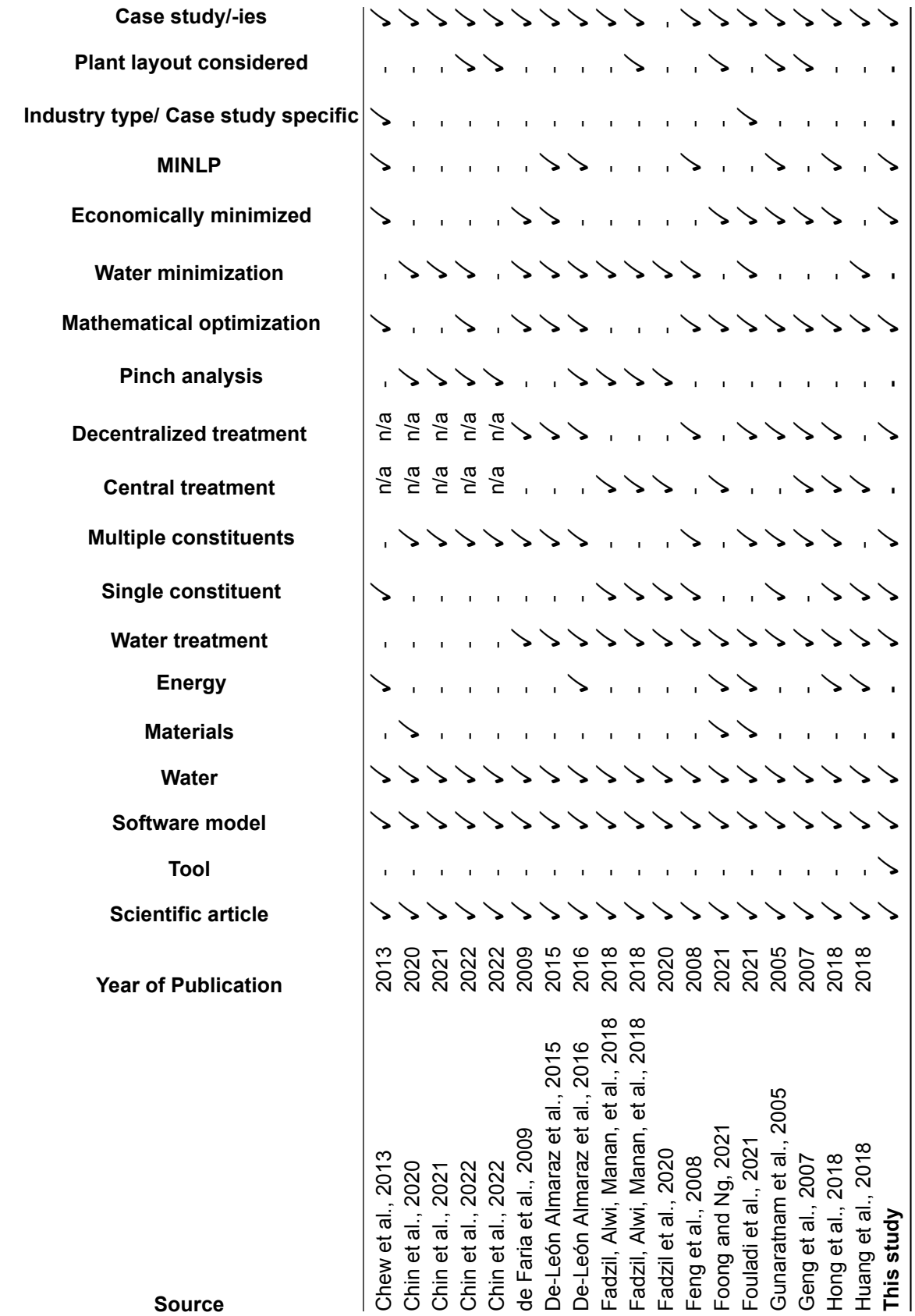
- Biological
  - Aerobic Suspended Growth
  - Aerobic Biological Treatment
  - Anaerobic Biological Treatment
  - Anaerobic Suspended Growth
  - Biological Nutrient Removal
  - Unspecified Biological Treatment
  - Constructed Wetlands
  - Aerobic Fixed Film Biological Treatment
  - Membrane Bioreactor
  - Moving Bed Bioreactor
  - Anaerobic Fixed Film Biological Treatment
  - Enhanced Biological Phosphorus Removal
  - Denitrification Filters
  - Biologically Active Filters
  - Biofilm Airlift Suspension Reactor
  - Integrated Fixed-Film Activated Sludge
  - Anaerobic Membrane Bioreactor
  - Granular Sludge Sequencing Batch Reactor
  - Bioaugmentation
- Chemical
  - Alkaline Chlorination
  - Hydrolysis, Acid or Alkaline
  - Chemical Phosphorous Removal
  - Chemical Precipitation
  - Chemical Disinfection
  - Ion Exchange
  - UV
  - Liquid Extraction
  - Chemical Oxidation
  - Zero Valent Iron
  - Dechlorination
  - Ozonation
  - Advanced Oxidation Processes, NEC
  - Gasification
  - Chemical Nitrogen Removal
  - Wet Air Oxidation

- Filtration
  - Granular-Media Filtration
  - Cloth Filtration
  - Bag and Cartridge Filtration
- Membrane
  - Micro- and Ultra-Membrane Filtration
  - Reverse Osmosis
  - Nanofiltration
  - Forward Osmosis
  - Membrane Distillation
- Physical
  - Clarification
  - Dissolved Air Flotation
  - Dissolved Gas Flotation
  - Ultrasound
  - Aeration
  - Mechanical Pre-Treatment
  - Controlled Hydrodynamic Cavitation
  - Crystallization
  - Distillation
  - Electrocoagulation
  - Capacitive Deionization
  - Evaporation
  - Stripping
  - Flow Equalization
  - Electrodialysis
  - Centrifugal Separator
  - Degasification
  - Oil/Water Separation
  - Ballasted Clarification
  - Surface Impoundment
- Sorption
  - Adsorptive Media
  - Powdered Activated Carbon
  - Granular Activated Carbon Unit

D

Literature Study Table

Source	Year of Publication	Scientific article	Tool	Software model	Water	Materials	Energy	Water treatment	Single constituent	Multiple constituents	Central treatment	Decentralized treatment	Pinch analysis	Mathematical optimization	Water minimization	Economically minimized	MINLP	Industry type/ Case study specific	Plant layout considered	Case study/-ies
Abraham et al., 2022	2022	✓	-	✓	✓	✓	✓	✓	✓	-	-	✓	-	✓	-	✓	-	✓	-	✓
Ahmed et al., 2020	2020	✓	-	✓	✓	✓	✓	✓	✓	-	-	✓	-	✓	-	✓	-	-	-	✓
Alnouri et al., 2014	2014	✓	-	✓	✓	-	-	-	✓	✓	n/a	n/a	✓	✓	✓	✓	-	-	✓	✓
Alnouri et al., 2016	2016	✓	-	✓	✓	-	-	✓	✓	✓	✓	✓	-	✓	-	✓	✓	-	✓	✓
Alva-Argáez et al., 1998	1998	✓	-	-	✓	-	-	✓	-	✓	-	✓	✓	✓	-	✓	✓	-	✓	✓
Alva-Argáez et al., 2007	2007	✓	-	-	✓	-	-	✓	✓	✓	✓	✓	✓	✓	-	✓	✓	-	✓	✓
Aviso et al., 2010	2010	✓	-	✓	✓	-	-	✓	✓	✓	✓	-	-	✓	✓	✓	-	-	-	✓
Aviso, 2014	2014	✓	-	✓	✓	-	-	-	✓	✓	n/a	n/a	-	✓	✓	-	-	-	-	✓
Behera et al., 2020	2020	✓	✓	✓	✓	-	✓	✓	-	✓	✓	-	-	✓	✓	✓	-	n/a	-	✓
Bishnu et al., 2014	2014	✓	-	✓	✓	-	-	-	-	✓	n/a	n/a	-	✓	✓	✓	✓	-	✓	✓
Boix et al., 2011	2011	✓	-	✓	✓	-	-	✓	-	✓	-	✓	-	✓	✓	-	✓	-	-	✓
Boix et al., 2012	2012	✓	-	✓	✓	-	-	✓	✓	-	✓	✓	-	✓	-	-	-	-	-	✓
Boix et al., 2023	2023	✓	-	✓	✓	-	✓	-	-	-	n/a	n/a	-	✓	-	✓	-	-	✓	✓
Chen et al., 2010	2010	✓	-	✓	✓	-	-	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	-	✓	✓
Chew et al., 2008	2008	✓	-	✓	✓	-	-	✓	✓	✓	✓	-	-	✓	-	✓	✓	-	✓	✓
Chew and Foo, 2009	2009	✓	-	✓	✓	-	-	✓	✓	-	✓	-	✓	✓	-	✓	-	-	✓	✓
Chew, Foo, Ng, et al., 2010	2010	✓	-	✓	✓	-	-	✓	✓	-	✓	✓	✓	-	✓	-	-	-	-	✓
Chew, Foo, and Tan, 2010	2010	✓	-	✓	✓	-	-	✓	✓	-	✓	✓	✓	-	✓	-	-	-	-	✓
Chew et al., 2011	2011	✓	-	✓	✓	-	-	✓	✓	✓	✓	-	-	-	n/a	n/a	-	-	-	✓
This study		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	✓	✓

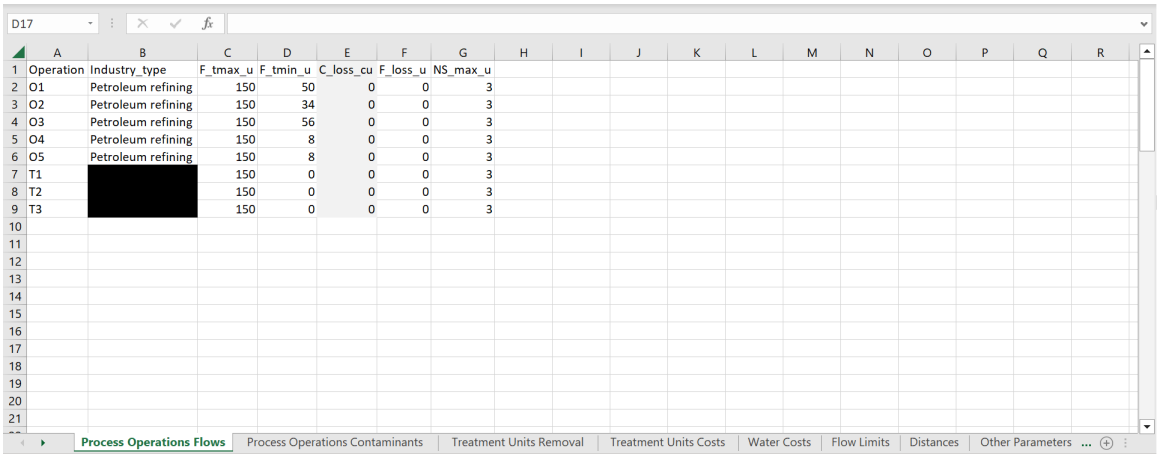

Source
Year of Publication
Scientific article
Tool
Software model
Water
Materials
Energy
Water treatment
Single constituent
Multiple constituents
Central treatment
Decentralized treatment
Pinch analysis
Mathematical optimization
Water minimization
Economically minimized
MINLP
Industry type/ Case study specific
Plant layout considered
Case study/-ies
Chew et al., 2013
Chin et al., 2020
Chin et al., 2021
Chin et al., 2022
Chin et al., 2022
de Faria et al., 2009
De-León Almaraz et al., 2015
De-León Almaraz et al., 2016
Fadzil, Alwi, Manan, et al., 2018
Fadzil, Alwi, Manan, et al., 2018
Fadzil et al., 2020
Feng et al., 2008
Foong and Ng, 2021
Fouladi et al., 2021
Gunaratnam et al., 2005
Geng et al., 2007
Hong et al., 2018
Huang et al., 2018
This study

Source	Year of Publication	Scientific article	Tool	Software model	Water	Materials	Energy	Water treatment	Single constituent	Multiple constituents	Central treatment	Decentralized treatment	Pinch analysis	Mathematical optimization	Water minimization	Economically minimized	MINLP	Industry type/ Case study specific	Plant layout considered	Case study/-ies
Keckler and Allen, 1998	1998	✓	-	✓	✓	-	-	✓	-	✓	✓	✓	✓	✓	✓	✓	✓	-	-	✓
Kuo and Smith, 1998	1998	✓	-	-	✓	-	-	✓	✓	-	✓	✓	✓	✓	✓	✓	✓	-	-	✓
Le et al., 2022	2022	✓	-	-	✓	-	-	-	-	✓	n/a	n/a	-	✓	✓	✓	✓	-	-	✓
Leong et al., 2014	2014	✓	-	-	✓	-	✓	-	-	-	✓	-	-	✓	-	-	✓	✓	-	✓
Li et al., 2015	2015	✓	-	✓	✓	-	-	✓	✓	✓	-	✓	✓	-	✓	-	-	-	-	✓
Liao et al., 2007	2007	✓	-	✓	✓	-	-	-	✓	-	n/a	n/a	✓	✓	✓	✓	✓	-	-	✓
Liao et al., 2011	2011	✓	-	✓	✓	-	✓	✓	✓	-	-	✓	-	✓	-	✓	-	-	-	✓
Lim and Park, 2010	2010	✓	-	✓	✓	-	-	-	✓	✓	n/a	n/a	-	✓	✓	-	-	-	-	✓
Z. Y. Liu et al., 2009	2009	✓	-	-	✓	-	-	-	-	✓	n/a	n/a	-	-	-	-	-	-	-	✓
C. Liu et al., 2015	2015	✓	-	-	✓	✓	✓	-	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	-	✓	✓	✓
Lovelady and El-Halwagi, 2009	2009	✓	-	✓	✓	-	-	✓	✓	-	-	✓	-	✓	-	✓	✓	-	-	✓
Mohammadnejad et al., 2012	2012	✓	-	-	✓	-	-	-	-	✓	n/a	n/a	✓	-	✓	-	-	✓	-	✓
Nemati-Amirkolaii et al., 2021	2021	✓	✓	✓	✓	-	-	-	✓	✓	n/a	n/a	-	✓	✓	-	✓	-	-	✓
Ng et al., 2009	2009	✓	-	✓	✓	✓	-	-	✓	-	n/a	n/a	✓	✓	✓	✓	-	-	-	✓
Ng et al., 2014	2014	✓	✓	✓	✓	✓	-	-	✓	-	n/a	n/a	✓	✓	✓	-	-	-	-	✓
O'Dwyer et al., 2020	2020	✓	-	✓	✓	✓	-	✓	-	✓	✓	✓	-	✓	-	✓	-	-	✓	✓
Olesen and Polley, 1996	1996	✓	-	-	✓	-	-	✓	✓	✓	-	✓	✓	-	✓	-	-	-	✓	✓
Pan et al., 2012	2012	✓	-	✓	✓	-	-	✓	-	✓	-	✓	-	✓	✓	-	-	-	-	✓
This study		✓	✓	✓	✓	-	-	✓	✓	✓	-	✓	-	✓	-	✓	✓	-	-	✓

Source	Year of Publication	Scientific article	Tool	Software model	Water	Materials	Energy	Water treatment	Single constituent	Multiple constituents	Central treatment	Decentralized treatment	Pinch analysis	Mathematical optimization	Water minimization	Economically minimized	MINLP	Industry type/ Case study specific	Plant layout considered	Case study/-ies
Pham et al., 2016	2016	✓	✓	✓	✓	-	-	✓	-	-	✓	-	-	-	-	-	-	✓	✓	✓
Poplewski et al., 2011	2011	✓	-	✓	✓	-	-	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	-	✓	✓
Putra and Amminudin, 2008	2008	✓	-	✓	✓	-	-	✓	-	✓	-	✓	-	✓	✓	✓	-	-	-	✓
Relvas et al., 2008	2008	✓	✓	✓	✓	-	-	✓	✓	-	-	✓	✓	✓	✓	✓	✓	-	-	✓
Rubio-Castro et al., 2010	2010	✓	-	✓	✓	-	-	✓	✓	-	-	✓	-	✓	-	✓	-	-	-	✓
Rubio-Castro et al., 2011	2011	✓	-	✓	✓	-	-	✓	-	✓	✓	✓	-	✓	-	✓	-	-	-	✓
Rubio-Castro et al., 2012	2012	✓	-	✓	✓	-	-	✓	✓	✓	✓	✓	-	✓	-	✓	-	-	-	✓
Rubio-Castro et al., 2013	2013	✓	-	✓	✓	-	-	✓	-	✓	-	✓	-	✓	-	✓	-	-	-	✓
Short et al., 2018	2018	✓	✓	✓	✓	-	-	✓	✓	-	n/a	n/a	-	✓	-	✓	-	-	-	✓
Sotelo-Pichardo et al., 2011	2011	✓	-	✓	✓	-	-	✓	✓	-	-	✓	-	✓	-	✓	-	-	-	✓
Su et al., 2012	2012	✓	-	✓	✓	-	-	-	-	✓	n/a	n/a	-	✓	✓	-	-	-	-	✓
Teles et al., 2007	2007	✓	✓	✓	✓	-	-	-	-	✓	n/a	n/a	-	✓	✓	-	-	-	-	✓
Tiu and Cruz, 2017	2017	✓	-	✓	✓	-	✓	✓	✓	-	-	✓	-	✓	✓	✓	-	-	-	✓
Tokos and Pintarič, 2012	2012	✓	-	✓	✓	-	-	✓	✓	-	✓	✓	-	✓	-	✓	✓	✓	✓	✓
Wang and Smith, 1994	1994	✓	-	-	✓	-	-	✓	✓	✓	-	✓	✓	-	✓	-	-	-	-	✓
B. Wang et al., 2003	2003	✓	-	✓	✓	-	-	-	-	✓	✓	n/a	-	✓	✓	-	-	-	-	✓
X. Wang et al., 2019	2019	✓	-	✓	✓	-	-	-	-	✓	n/a	n/a	-	✓	✓	-	-	-	-	✓
Zhou et al., 2015	2015	✓	-	✓	✓	-	✓	✓	✓	-	-	✓	-	✓	-	✓	✓	-	-	✓
This study		✓	✓	✓	✓	-	-	✓	✓	✓	-	✓	-	✓	-	✓	✓	-	-	✓

E

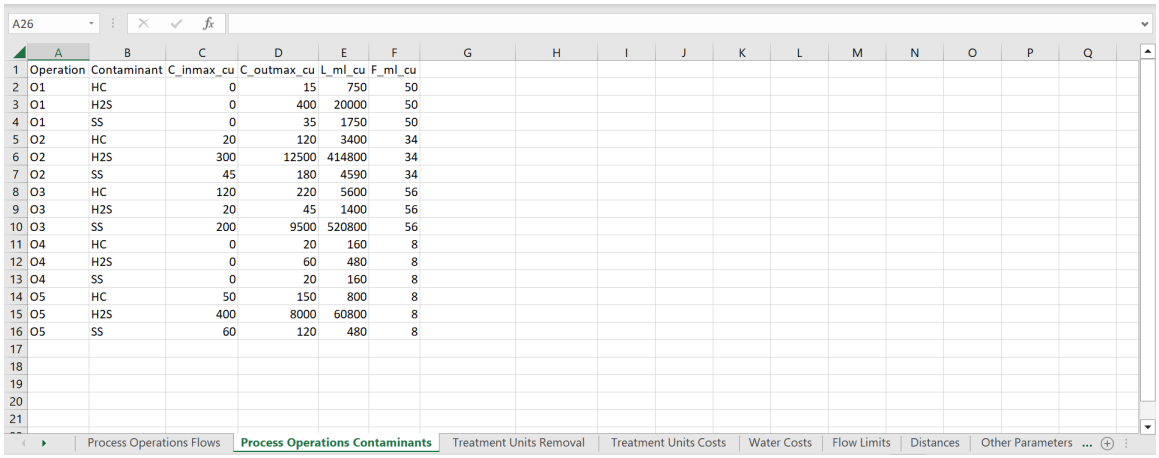
Excel File



	A	B	C	D	E	F	G
1	Operation	Industry_type	F_tmax_u	F_tmin_u	C_loss_cu	F_loss_u	NS_max_u
2	O1	Petroleum refining	150	50	0	0	3
3	O2	Petroleum refining	150	34	0	0	3
4	O3	Petroleum refining	150	56	0	0	3
5	O4	Petroleum refining	150	8	0	0	3
6	O5	Petroleum refining	150	8	0	0	3
7	T1		150	0	0	0	3
8	T2		150	0	0	0	3
9	T3		150	0	0	0	3

Process Operations Flows | Process Operations Contaminants | Treatment Units Removal | Treatment Units Costs | Water Costs | Flow Limits | Distances | Other Parameters

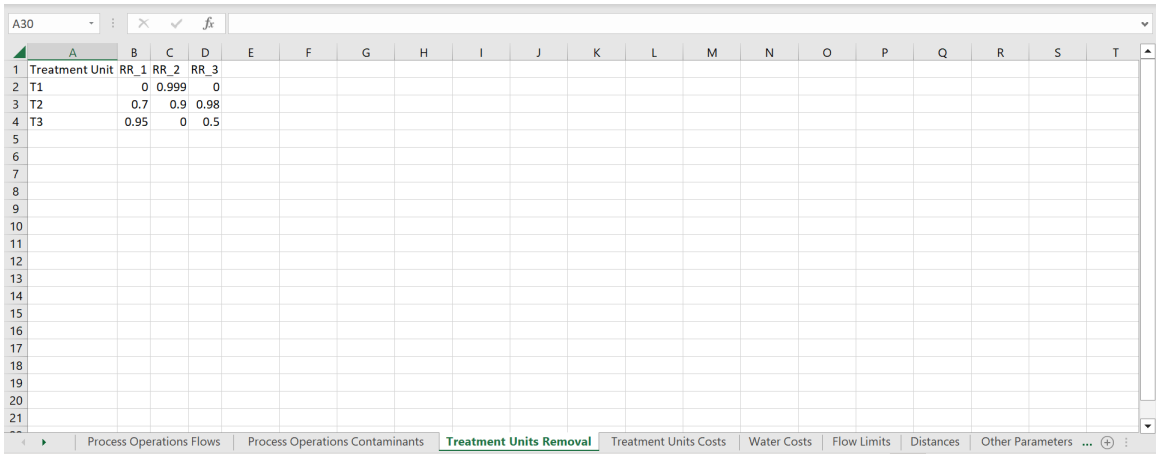
Figure E.1: Excel Input File, 'Process Operations Contaminants' tab



	A	B	C	D	E	F
1	Operation	Contaminant	C_inmax_cu	C_outmax_cu	L_ml_cu	F_ml_cu
2	O1	HC	0	15	750	50
3	O1	H2S	0	400	20000	50
4	O1	SS	0	35	1750	50
5	O2	HC	20	120	3400	34
6	O2	H2S	300	12500	414800	34
7	O2	SS	45	180	4590	34
8	O3	HC	120	220	5600	56
9	O3	H2S	20	45	1400	56
10	O3	SS	200	9500	520800	56
11	O4	HC	0	20	160	8
12	O4	H2S	0	60	480	8
13	O4	SS	0	20	160	8
14	O5	HC	50	150	800	8
15	O5	H2S	400	8000	60800	8
16	O5	SS	60	120	480	8

Process Operations Flows | Process Operations Contaminants | Treatment Units Removal | Treatment Units Costs | Water Costs | Flow Limits | Distances | Other Parameters

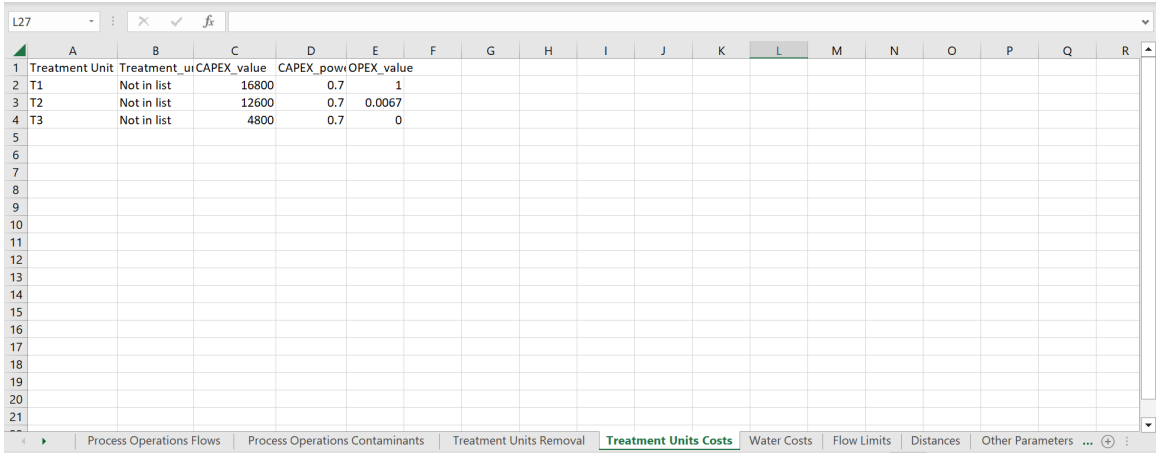
Figure E.2: Excel Input File, 'Process Operations Flows' tab



	A	B	C	D
1	Treatment Unit	RR_1	RR_2	RR_3
2	T1	0	0.999	0
3	T2	0.7	0.9	0.98
4	T3	0.95	0	0.5

Process Operations Flows | Process Operations Contaminants | **Treatment Units Removal** | Treatment Units Costs | Water Costs | Flow Limits | Distances | Other Parameters

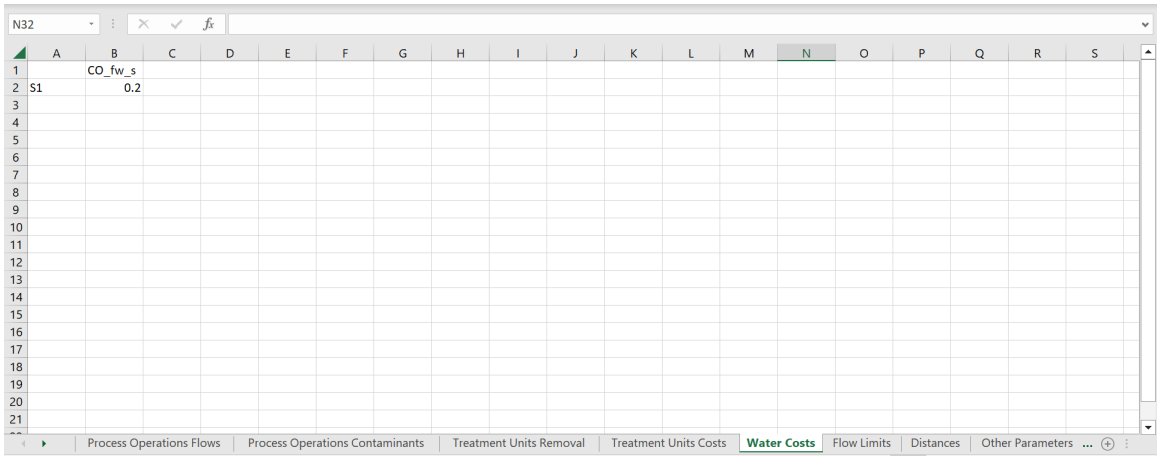
Figure E.3: Excel Input File, 'Treatment Units Removal' tab



	A	B	C	D	E
1	Treatment Unit	Treatment_u	CAPEX_value	CAPEX_pow	OPEX_value
2	T1	Not in list	16800	0.7	1
3	T2	Not in list	12600	0.7	0.0067
4	T3	Not in list	4800	0.7	0

Process Operations Flows | Process Operations Contaminants | Treatment Units Removal | **Treatment Units Costs** | Water Costs | Flow Limits | Distances | Other Parameters

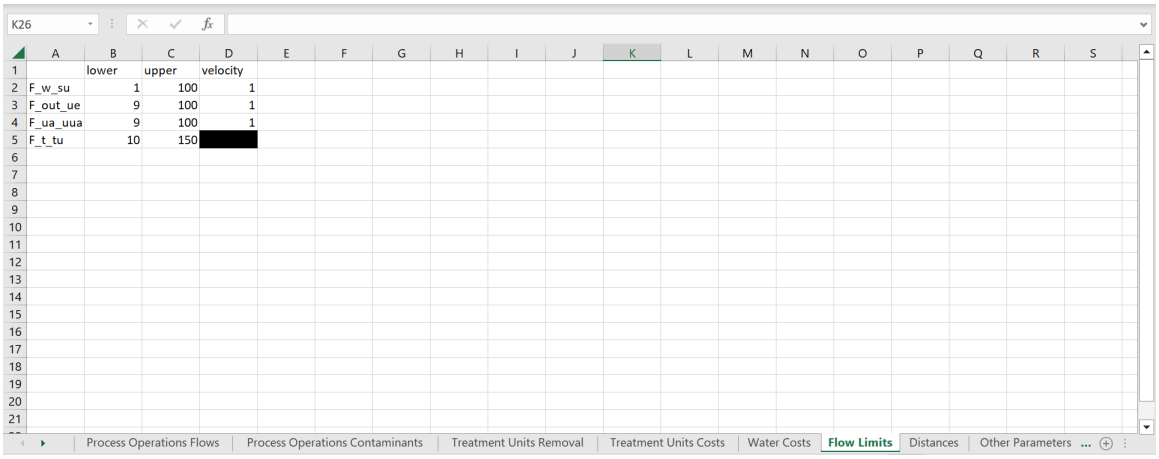
Figure E.4: Excel Input File, 'Treatment Units Costs' tab



	A	B
1		CO_fw_s
2	S1	0.2

Process Operations Flows | Process Operations Contaminants | Treatment Units Removal | Treatment Units Costs | **Water Costs** | Flow Limits | Distances | Other Parameters

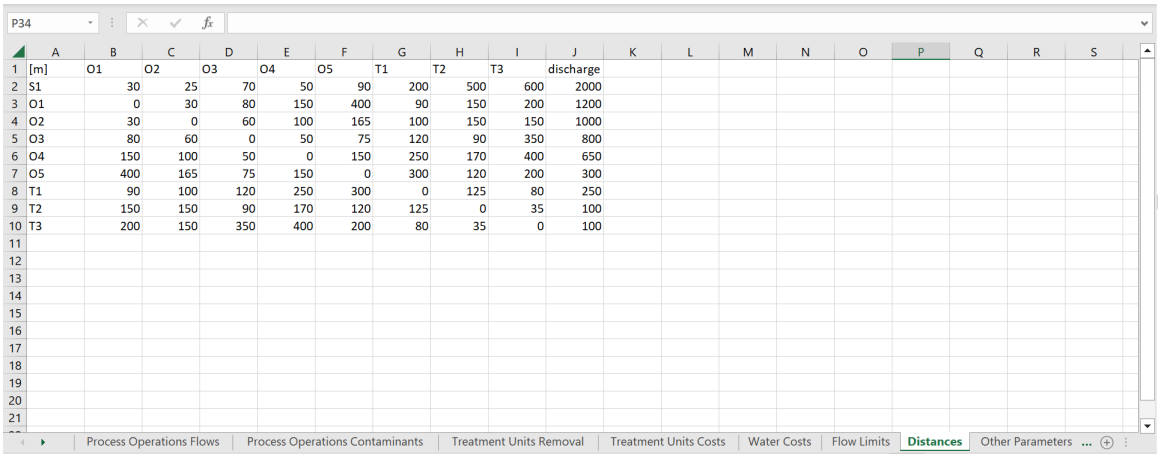
Figure E.5: Excel Input File, 'Water Costs' tab



	A	B	C	D
1		lower	upper	velocity
2	F_w_su	1	100	1
3	F_out_ue	9	100	1
4	F_ua_uua	9	100	1
5	F_t_tu	10	150	

Process Operations Flows | Process Operations Contaminants | Treatment Units Removal | Treatment Units Costs | Water Costs | **Flow Limits** | Distances | Other Parameters

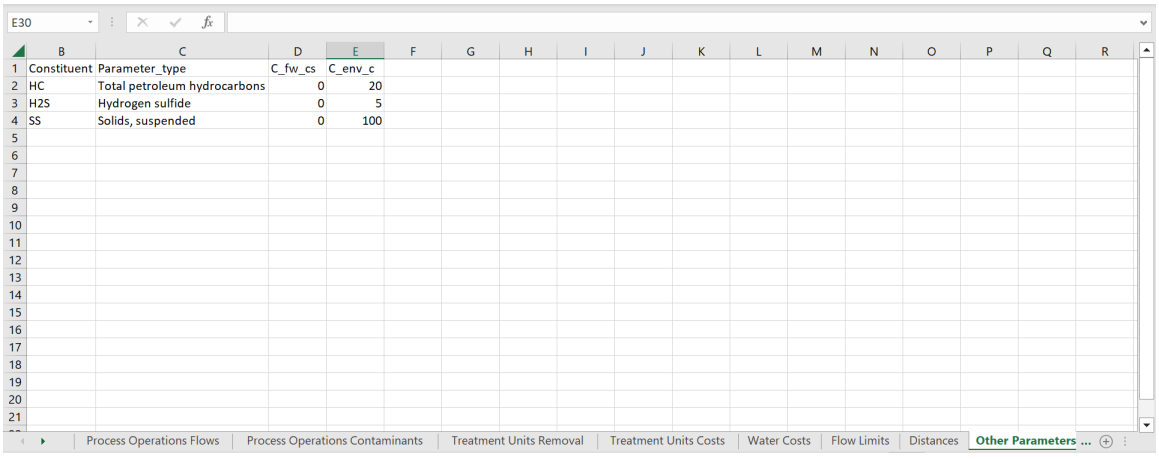
Figure E.6: Excel Input File, 'Flow Limits' tab



	A	B	C	D	E	F	G	H	I	J
1	[m]	O1	O2	O3	O4	O5	T1	T2	T3	discharge
2	S1	30	25	70	50	90	200	500	600	2000
3	O1	0	30	80	150	400	90	150	200	1200
4	O2	30	0	60	100	165	100	150	150	1000
5	O3	80	60	0	50	75	120	90	350	800
6	O4	150	100	50	0	150	250	170	400	650
7	O5	400	165	75	150	0	300	120	200	300
8	T1	90	100	120	250	300	0	125	80	250
9	T2	150	150	90	170	120	125	0	35	100
10	T3	200	150	350	400	200	80	35	0	100

Process Operations Flows | Process Operations Contaminants | Treatment Units Removal | Treatment Units Costs | Water Costs | Flow Limits | **Distances** | Other Parameters

Figure E.7: Excel Input File, 'Distances' tab



	B	C	D	E
1	Constituent	Parameter_type	C_fw_cs	C_env_c
2	HC	Total petroleum hydrocarbons	0	20
3	H2S	Hydrogen sulfide	0	5
4	SS	Solids, suspended	0	100

Process Operations Flows | Process Operations Contaminants | Treatment Units Removal | Treatment Units Costs | Water Costs | Flow Limits | Distances | **Other Parameters**

Figure E.8: Excel Input File, 'Other Parameters' tab

F

Python Code

```
## IMPORT PACKAGES
import numpy as np
import pandas as pd
import networkx as nx  # Visualisation of flows
import graphviz as gv  # label curved edges in figure
from amplpy import AMPL, modules  # Installation of solvers
import matplotlib.pyplot as plt  # Visualisation of flows
from pyomo.environ import (
    Var,
    NonNegativeReals,
    ConcreteModel,
    Binary,
    Constraint,
    Reals,
    Objective,
    minimize,
    SolverFactory,
    value,
)
from sys import exit

## FUNCTIONS (PARTLY) EXTERNAL
def my_draw_networkx_edge_labels(
    G,
    pos,
    edge_labels=None,
    label_pos=0.5,
    font_size=10,
    font_color="k",
    font_family="sans-serif",
    font_weight="normal",
    alpha=None,
    bbox=None,
    horizontalalignment="center",
    verticalalignment="center",
    ax=None,
    rotate=True,
    clip_on=True,
    rad=0
):
    """Draw edge labels.
```

```
    Parameters
    ----------
    G : graph
        A networkx graph

    pos : dictionary
        A dictionary with nodes as keys and positions as values.
        Positions should be sequences of length 2.

    edge_labels : dictionary (default={})
        Edge labels in a dictionary of labels keyed by edge two-tuple.
        Only labels for the keys in the dictionary are drawn.

    label_pos : float (default=0.5)
        Position of edge label along edge (0=head, 0.5=center, 1=tail)

    font_size : int (default=10)
        Font size for text labels

    font_color : string (default='k' black)
        Font color string

    font_weight : string (default='normal')
        Font weight

    font_family : string (default='sans-serif')
        Font family

    alpha : float or None (default=None)
        The text transparency

    bbox : Matplotlib bbox, optional
        Specify text box properties (e.g. shape, color etc.) for edge labels.
        Default is {boxstyle='round', ec=(1.0, 1.0, 1.0), fc=(1.0, 1.0, 1.0)}.

    horizontalalignment : string (default='center')
        Horizontal alignment {'center', 'right', 'left'}

    verticalalignment : string (default='center')
        Vertical alignment {'center', 'top', 'bottom', 'baseline', 'center_baseline'}

    ax : Matplotlib Axes object, optional
        Draw the graph in the specified Matplotlib axes.

    rotate : bool (deafult=True)
        Rotate edge labels to lie parallel to edges

    clip_on : bool (default=True)
        Turn on clipping of edge labels at axis boundaries

    Returns
    -------
    dict
        `dict` of labels keyed by edge

    Examples
    --------
    >>> G = nx.dodecahedral_graph()
    >>> edge_labels = nx.draw_networkx_edge_labels(G, pos=nx.spring_layout(G))

    Also see the NetworkX drawing examples at
```

```
    https://networkx.org/documentation/latest/auto_examples/index.html

    See Also
    --------
    draw
    draw_networkx
    draw_networkx_nodes
    draw_networkx_edges
    draw_networkx_labels
    """
    import matplotlib.pyplot as plt
    import numpy as np

    if ax is None:
        ax = plt.gca()
    if edge_labels is None:
        labels = {(u, v): d for u, v, d in G.edges(data=True)}
    else:
        labels = edge_labels
    text_items = {}
    for (n1, n2), label in labels.items():
        (x1, y1) = pos[n1]
        (x2, y2) = pos[n2]
        (x, y) = (
            x1 * label_pos + x2 * (1.0 - label_pos),
            y1 * label_pos + y2 * (1.0 - label_pos),
        )
        pos_1 = ax.transData.transform(np.array(pos[n1]))
        pos_2 = ax.transData.transform(np.array(pos[n2]))
        linear_mid = 0.5*pos_1 + 0.5*pos_2
        d_pos = pos_2 - pos_1
        rotation_matrix = np.array([(0,1), (-1,0)])
        ctrl_1 = linear_mid + rad*rotation_matrix@d_pos
        ctrl_mid_1 = 0.5*pos_1 + 0.5*ctrl_1
        ctrl_mid_2 = 0.5*pos_2 + 0.5*ctrl_1
        bezier_mid = 0.5*ctrl_mid_1 + 0.5*ctrl_mid_2
        (x, y) = ax.transData.inverted().transform(bezier_mid)

        if rotate:
            # in degrees
            angle = np.arctan2(y2 - y1, x2 - x1) / (2.0 * np.pi) * 360
            # make label orientation "right-side-up"
            if angle > 90:
                angle -= 180
            if angle < -90:
                angle += 180
            # transform data coordinate angle to screen coordinate angle
            xy = np.array((x, y))
            trans_angle = ax.transData.transform_angles(
                np.array((angle,)), xy.reshape((1, 2))
            )[0]
        else:
            trans_angle = 0.0
        # use default box of white with white border
        if bbox is None:
            bbox = dict(boxstyle="round", ec=(1.0, 1.0, 1.0), fc=(1.0, 1.0, 1.0))
        if not isinstance(label, str):
            label = str(label)  # this makes "1" and 1 labeled the same

        t = ax.text(
            x,
```

```
            y,
            label,
            size=font_size,
            color=font_color,
            family=font_family,
            weight=font_weight,
            alpha=alpha,
            horizontalalignment=horizontalalignment,
            verticalalignment=verticalalignment,
            rotation=trans_angle,
            transform=ax.transData,
            bbox=bbox,
            zorder=1,
            clip_on=clip_on,
        )
        text_items[(n1, n2)] = t

    ax.tick_params(
        axis="both",
        which="both",
        bottom=False,
        left=False,
        labelbottom=False,
        labelleft=False,
    )

    return text_items

def my_linearization_procedure(
    F_t_tu_min,
    F_t_tu_max,
    steps,
    CAPEX_value,
    CAPEX_power,
    OPEX_value
):
    import numpy as np
    F_t_tu_list = np.linspace(F_t_tu_min, F_t_tu_max, steps)

    Costs_list = []
    for i in range(len(F_t_tu_list)):
        CAPEX = CAPEX_value * (F_t_tu_list[i] ** 0.7)
        OPEX = OPEX_value * F_t_tu_list[i]
        Costs_tu = CAPEX * Annualization_factor + OPEX * Operation_time
        Costs_list.append(Costs_tu)

    x_mean = sum(F_t_tu_list) / len(F_t_tu_list)
    y_mean = sum(Costs_list) / len(Costs_list)
    covar = sum((xi - x_mean) * (yi - y_mean) for xi, yi in zip(F_t_tu_list, Costs_list))
    x_var = sum((xi - x_mean)**2 for xi in F_t_tu_list)
    beta = covar / x_var
    alpha = y_mean - beta * x_mean
    y_hat = [alpha + beta * xi for xi in F_t_tu_list]

    return beta, alpha

## INPUT CASE STUDY 1
# input_file = 'Gunaratnam2005-casestudy1-IWTT.xlsx'

# # SETS SIZES
# amount_of_constituents = 3
```

```
# amount_of_water_sources = 1
# amount_of_discharge_points = 1
# amount_of_water_using_operations = 5
# amount_of_treatment_units = 3

# #COST EQUATION FACTORS
# Operation_time = 8600 # h/year
# Annualization_factor = 0.1

# # SOLVER
# MILP_solver = 'scip'
# MILP_solver_location = r'C:\Users\NL1A60\AMPL\scip.exe'
# LP_solver = 'scip'
# LP_solver_location = r'C:\Users\NL1A60\AMPL\scip.exe'
# MINLP_solver = 'scip'
# MINLP_solver_location = r'C:\Users\NL1A60\AMPL\scip.exe'

# # ITERATION VALUES
# Maximum_amount_of_iterations = 10
# convergence_criterion = 0.01

# ## SWITCHES (0 = no, 1 = yes)
# #Optional functions
# water_reuse_allowed_switch = 1  # reuse after regeneration and between water operations
# regeneration_recycling_allowed_switch = 1  # is 1 in scenario 1, is 0 in scenarios 2 and 3
# include_environmental_limit_switch = 1
# pipe_connection_costs_switch = 0  # is 0 in scenarios 1 and 2, is 1 in scenario 3

# #Solve
# initation_activated_switch = 1
# iteration_activated_switch = 1
# final_MINLP_switch = 1

# #Visualisation & printing
# show_excel_data_switch = 0
# create_MINLP_visualization_switch = 1
# MINLP_print_results_switch = 1
# MILP_print_results_switch = 0
# LP_print_results_switch = 0

# #IWTT data switches
# Use_IWTT_switch = 0
# IWTT_show_data_switch = 0
# IWTT_make_excel_files_switch = 0
# only_check_for_iwtt_data_switch = 0

# ## INPUT CASE STUDY 2
# input_file = 'Gunaratnam2005-casestudy2.xlsx'

# # SETS SIZES
# amount_of_constituents = 1
# amount_of_water_sources = 1
# amount_of_discharge_points = 1
# amount_of_water_using_operations = 4
# amount_of_treatment_units = 0

# #COST EQUATION FACTORS
# Operation_time = 8600 # h/year
# Annualization_factor = 0.1

# # SOLVER
```

```
# MILP_solver = 'scip'
# MILP_solver_location = r'C:\Users\NL1A6O\AMPL\scip.exe'
# LP_solver = 'scip'
# LP_solver_location = r'C:\Users\NL1A6O\AMPL\scip.exe'
# MINLP_solver = 'scip'
# MINLP_solver_location = r'C:\Users\NL1A6O\AMPL\scip.exe'

# # ITERATION VALUES
# Maximum_amount_of_iterations = 10
# convergence_criterion = 0.01

# ## SWITCHES (0 = no, 1 = yes)
# #Optional functions
# water_reuse_allowed_switch = 0  # reuse after regeneration and between water operations
# regeneration_recycling_allowed_switch = 0
# include_environmental_limit_switch = 0
# pipe_connection_costs_switch = 0

# #Solve
# initation_activated_switch = 1
# iteration_activated_switch = 0
# final_MINLP_switch = 1

# #Visualisation & printing
# show_excel_data_switch = 0
# create_MINLP_visualization_switch = 1
# MINLP_print_results_switch = 1
# MILP_print_results_switch = 0
# LP_print_results_switch = 0

# #IWTT data switches
# Use_IWTT_switch = 0
# IWTT_show_data_switch = 0
# IWTT_make_excel_files_switch = 0
# only_check_for_iwtt_data_switch = 0

# ## INPUT CASE STUDY 3
# input_file = 'Gunaratnam2005-casestudy3.xlsx'

# # SETS SIZES
# amount_of_constituents = 3
# amount_of_water_sources = 1
# amount_of_discharge_points = 1
# amount_of_water_using_operations = 3
# amount_of_treatment_units = 3

# #COST EQUATION FACTORS
# Operation_time = 8600 # h/year
# Annualization_factor = 0.4

# # SOLVER
# MILP_solver = 'scip'
# MILP_solver_location = r'C:\Users\NL1A6O\AMPL\scip.exe'
# LP_solver = 'scip'
# LP_solver_location = r'C:\Users\NL1A6O\AMPL\scip.exe'
# MINLP_solver = 'scip'
# MINLP_solver_location = r'C:\Users\NL1A6O\AMPL\scip.exe'

# # ITERATION VALUES
# Maximum_amount_of_iterations = 10
# convergence_criterion = 0.01
```

```
# ## SWITCHES (0 = no, 1 = yes)
# #Optional functions
# water_reuse_allowed_switch = 0  # reuse after regeneration and between water operations
# regeneration_recycling_allowed_switch = 0
# include_environmental_limit_switch = 0
# pipe_connection_costs_switch = 0

# #Solve
# initation_activated_switch = 1
# iteration_activated_switch = 1
# final_MINLP_switch = 1

# #Visualisation & printing
# show_excel_data_switch = 0
# create_MINLP_visualization_switch = 1
# MINLP_print_results_switch = 1
# MILP_print_results_switch = 0
# LP_print_results_switch = 0

# #IWTT data switches
# Use_IWTT_switch = 0
# IWTT_show_data_switch = 0
# IWTT_make_excel_files_switch = 0
# only_check_for_iwtt_data_switch = 0

## IMPORT OF DATA AND DATA VISUALISATION
process_operations_flows_data = pd.read_excel(
    input_file,
    sheet_name='Process Operations Flows',
    index_col=[0],
    header=[0])
process_operations_contaminants_data = pd.read_excel(
    input_file,
    sheet_name='Process Operations Contaminants',
    index_col=[0,1],
    header=[0])
treatment_units_data = pd.read_excel(
    input_file,
    sheet_name='Treatment Units Removal',
    index_col=[0],
    header=[0])
treatment_units_data2 = pd.read_excel(
    input_file,
    sheet_name='Treatment Units Costs',
    index_col=[0],
    header=[0])
water_cost_data = pd.read_excel(
    input_file,
    sheet_name='Water Costs',
    index_col=[0],
    header=[0])
flow_limits_data = pd.read_excel(
    input_file,
    sheet_name='Flow Limits',
    index_col=[0],
    header=[0])
distances_data = pd.read_excel(
    input_file,
    sheet_name='Distances',
    index_col=[0],
```

```
    header=[0])
other_parameters_data = pd.read_excel(
    input_file,
    sheet_name='Other Parameters',
    index_col=[0],
    header=[0])

if show_excel_data_switch == 1:
    display(process_operations_flows_data)
    display(process_operations_contaminants_data)
    display(treatment_units_data)
    display(treatment_units_data2)
    display(distances_data)
    display(other_parameters_data)

## INDUSTRIAL WASTEWATER TREATMENT TECHNIQUES (IWTT)
if Use_IWTT_switch == 1:
    # Import IWTT database
    IWTT_parameters_codes = pd.read_csv(
        'KEY_PARAMETER_CODE.csv',
        header=[0])
    IWTT_treatment_techniques_codes = pd.read_csv(
        'KEY_TREATMENT_TECH_CODES.csv',
        header=[0], nrows=3)
    IWTT_treatment_techniques_data = pd.read_csv(
        'PARAMETER.csv',
        header=[0],
        encoding='unicode_escape',
        on_bad_lines='skip')
    IWTT_treatment_techniques_overview = pd.read_csv(
        'TREATMENT_SYSTEM.csv',
        encoding='unicode_escape',
        on_bad_lines='skip')
    IWTT_treatment_techniques_coupling = pd.read_csv(
        'TREATMENT_UNITS.csv',
        header=[0])

    # Remove unimportant columns
    IWTT_parameters_codes = IWTT_parameters_codes.drop(
        columns=['CAS_NMBR',
                 'CATEGORY',
                 'CATEGORY_2',])

    IWTT_treatment_techniques_codes = IWTT_treatment_techniques_codes.drop(
        columns=['TT_CATEGORY',
                 'TT_VARIATION'])

    IWTT_treatment_techniques_data = IWTT_treatment_techniques_data.drop(
        columns=['REPORTEDREMOVAL',
                 'CALCULATEDREMOVAL',
                 'ANALYTICAL_METHOD',
                 'INFLUENTDL',
                 'INFLUENTDLUNITS',
                 'EFFLUENTDL',
                 'EFFLUENTDLUNITS',
                 'ELDESCRIPTION',
                 'EFFLUENTLIMIT',
                 'ELUNITS',
                 'ELDESCRIPTION2',
                 'EL2',
                 'ELUNITS2',
```

```
            'ELDESCRIPTION3',
            'EL3',
            'ELUNITS3',
            'PERFORMSTATID',
            'ADDTDATA',
            'INFLUENTFLAG',
            'INFLUENTCONCENTRATION',
            'INFLUENTUNITS',
            'EFFLUENTFLAG',
            'EFFLUENTCONCENTRATION',
            'EFFLUENTUNITS',
            'REPORTEDREMOVALFLAG',
            'SYSTEMQUALIFIED',
            'UPDATEREMOVALFLAG',])

IWTT_treatment_techniques_overview = IWTT_treatment_techniques_overview.drop(
    columns=['NAICS_CODE',
            'PSC_CODE',
            'SIC_CODE',
            'TREATMENT_TECH_DESCRIPTION',
            'DISCHARGE_DESIGNATION',
            'WW_USE',
            'MEDIA_TYPE',
            'LOW_PH_RANGE',
            'HIGH_PH_RANGE',
            'PH_DESCRIPTOR',
            'CHEMICAL_ADDITION',
            'SCALE',
            'ADDT_SYSTEM_PARAMETERS',
            'MANUFACTURER',
            'HASPERFDATA',
            'CAPITAL_COST',
            'OANDM_COST',])

#Make one overview of data
df1 = IWTT_treatment_techniques_overview
df2 = IWTT_treatment_techniques_coupling
df_trains = IWTT_treatment_techniques_coupling
df3 = IWTT_treatment_techniques_data
df4 = df1.merge(df2, on=('SYSTEM_NAME','REF_ID'))
df5 = df4.merge(df3, on=('SYSTEM_NAME','REF_ID'))
df_trains = df_trains.groupby(df_trains['SYSTEM_NAME'])['TT_CODE'].apply(
    (', ').join).reset_index()
df_trains = df_trains.rename(columns={'TT_CODE': 'TT_CODE_TRAIN'})
df_final = df5.merge(df_trains, on=('SYSTEM_NAME'))
df_final = df_final.dropna(subset='UPDATEREMOVAL')
df_final = df_final.drop(columns=['TT_CODE',
                                  'TT_CODE_ORDER',
                                  'REF_ID',
                                  'PARAMID',
                                  'SYSTEM_NAME'])
df_final = df_final.drop_duplicates()

#Put industry list from Excel in list
industry = process_operations_flows_data.Industry_type.dropna().drop_duplicates()
Industry_types_list = []
for i in range(len(industry)):
    Industry_types_list.append(industry[i])

#Show relevant IWTT data
if IWTT_show_data_switch == 1:
```

```
        for i in range(len(Industry_types_list)):
            industry = Industry_types_list[i]
            display(df_final.loc[df_final.INDUSTRY == industry])
            if IWTT_make_excel_files_switch == 1:
                df_final.loc[df_final.INDUSTRY == industry].to_excel(
                    f"{industry} data overview.xlsx")

    if only_check_for_iwtt_data_switch == 1:
        print(f'The only_check_for_iwtt_data_switch is on',
              'so the code stopped running from this point.')
        exit()

## DEFINE PARAMETERS BASED ON INPUT

#  Sets
c = amount_of_constituents
s = amount_of_water_sources
e = amount_of_discharge_points
u = ua = amount_of_water_using_operations + amount_of_treatment_units
wu = amount_of_water_using_operations
tu = amount_of_treatment_units

# (i) Concentration bounds
C_inmax_cu = process_operations_contaminants_data.C_inmax_cu  # ppm
C_outmax_cu = process_operations_contaminants_data.C_outmax_cu  # ppm

if pipe_connection_costs_switch == 1:
    # (ii) Distances between water sources and sinks
    d_fw_su = distances_data  # m
    d_ua_uua = distances_data  # m
    d_out_ue = distances_data.discharge  # m

    # (iii) Flow velocities within the pipe connections
    u_fw_su = flow_limits_data.velocity[0]  # m/s
    u_out_ue = flow_limits_data.velocity[1]  # m/s
    u_ua_uua = flow_limits_data.velocity[2]  # m/s

    # (iv) Regression parameters for piping costs
    a_fw_su = 3603.4  # -
    b_fw_su = 124.6  # -
    a_ua_uua = 3603.4  # -
    b_ua_uua = 124.6  # -
    a_out_ue = 3603.4  # -
    b_out_ue = 124.6  # -

# (v) Other parameters
C_fw_cs = other_parameters_data.C_fw_cs  # ppm
C_env_c = other_parameters_data.C_env_c  # ppm
C_loss_cu = process_operations_flows_data.C_loss_cu  # ppm
F_loss_u = process_operations_flows_data.F_loss_u  # t/h
L_ml_cu = process_operations_contaminants_data.L_ml_cu  # g/h
RR_cu = treatment_units_data  # -
CO_fw_s = water_cost_data.CO_fw_s # $/ton
NS_max_u = process_operations_flows_data.NS_max_u  # -
F_tmax_u = process_operations_flows_data.F_tmax_u  # t/h
F_tmin_u = process_operations_flows_data.F_tmin_u  # t/h
N__OP = u  # -  # equation 22

#  Defined for cost functions
CAPEX_value = treatment_units_data2.CAPEX_value
CAPEX_power = treatment_units_data2.CAPEX_power
```

```
OPEX_value = treatment_units_data2.OPEX_value
F_t_tu_min = flow_limits_data.lower[3]
F_t_tu_max = flow_limits_data.upper[3]
steps = 10

# Parameters later introduced in equations
# U_fw_su =  # constraint 11  # t/h
U_fw = np.zeros((s, wu))
for i in range(s):
    for j in range(wu):
        U_fw[i][j] = flow_limits_data.upper[0]

# L_fw_su =  # constraint 12  # t/h
L_fw = np.zeros((s, wu))
for i in range(s):
    for j in range(wu):
        L_fw[i][j] = flow_limits_data.lower[0]

# U_out_ue =  # constraint 13  # t/h
U_out = np.zeros((u, e))
for i in range(u):
    for j in range(e):
        U_out[i][j] = flow_limits_data.upper[1]

# L_out_ue =  # constraint 14  # t/h
L_out = np.zeros((u, e))
for i in range(u):
    for j in range(e):
        L_out[i][j] = flow_limits_data.lower[1]

# U_ua_u,ua = U_ua_ua,u # constraint 15  # t/h
U_ua = np.zeros((u, ua))
for i in range(u):
    for j in range(ua):
        U_ua[i][j] = flow_limits_data.upper[2]

# L_ua_u,ua = L_ua_ua,u # constraint 16  # t/h
L_ua = np.zeros((u, ua))
for i in range(u):
    for j in range(ua):
        L_ua[i][j] = flow_limits_data.lower[2]

## DEFINE LISTS TO STORE RESULTS

Cost_results_list = []
M_terms_results_list = []

## MILP FUNCTION

MILP_model = ConcreteModel()

def MILP_function(
    MILP_input_list
):
    #DEFINE LIST FOR FLOW RESULTS
    MILP_flow_results_list = []
    MILP_binary_variables_list = []
    MILP_no_treatment_flows = []

    ## DEFINE MILP_MODEL DECISION VARIABLES
    #(i) Continuous variables associated with flow rates
```

```
#F_w_su
m = 0
for i in range(s):
    for j in range(wu):
        m += 1
        MILP_model.add_component(f'F_w_{i+1}{j+1}',
                                 Var(within=NonNegativeReals,
                                     bounds=(0, F_tmax_u[m-1])))

#F_out_ue
for i in range(u):
    for j in range(e):
        MILP_model.add_component(f'F_out_{i+1}{j+1}',
                                 Var(within=NonNegativeReals,
                                     bounds=(0, F_tmax_u[m-1])))

#F_ua_u,ua = F_ua_ua,u
for i in range(u):
    for j in range(ua):
        MILP_model.add_component(f'F_ua_{i+1}{j+1}',
                                 Var(within=NonNegativeReals,
                                     bounds=(0, F_tmax_u[m-1])))

#F_t_u
for i in range(u):
    MILP_model.add_component(f'F_t_{i+1}',
                             Var(within=NonNegativeReals,
                                 bounds=(F_tmin_u[i], F_tmax_u[i]),
                                 initialize=F_tmin_u[i]))

#(ii) Contaminant concentration and mass flow in process operations
#C _conk_cu
m = 0
for j in range(u):
    for i in range(c):
        m += 1
        MILP_model.add_component(f'C_conk_{i+1}{j+1}',
                                 Var(within=NonNegativeReals))
        MILP_model.add_component(f'eq_fix_concentrations_{m}', Constraint(expr=(
            getattr(MILP_model, f'C_conk_{i+1}{j+1}')
        ) == MILP_input_list[m-1]))

#M_in_cu & M_out_cu
m=0
for i in range(c):
    for j in range(u):
        m+=1
        MILP_model.add_component(f'M_in_{i+1}{j+1}',
                                 Var(within=NonNegativeReals))
        MILP_model.add_component(f'M_out_{i+1}{j+1}',
                                 Var(within=NonNegativeReals))

#M_loss_cu
for i in range(c):
    for j in range(wu):
        MILP_model.add_component(f'M_loss_{i+1}{j+1}',
                                 Var(within=NonNegativeReals,
                                     initialize=0))

#M_gain_cu
for i in range(c):
```

```
    for j in range(u):
        MILP_model.add_component(f'M_gain_{i+1}{j+1}',
                                 Var(within=NonNegativeReals,
                                     initialize=0))

#(iii) Cross-sectional area of the pipes connecting different operations
if pipe_connection_costs_switch == 1:
    #A_fw_su
    for i in range(s):
        for j in range(wu):
            MILP_model.add_component(f'A_fw_{i+1}{j+1}',
                                     Var(within=NonNegativeReals))

    #A_ua_uua
    for i in range(u):
        for j in range(ua):
            MILP_model.add_component(f'A_ua_{i+1}{j+1}',
                                     Var(within=NonNegativeReals))

    #A_out_ue
    for i in range(u):
        for j in range(e):
            MILP_model.add_component(f'A_out_{i+1}{j+1}',
                                     Var(within=NonNegativeReals))

#(iv) Cost terms in the objective function

#Cost_fw_s
for i in range(s):
    MILP_model.add_component(f'Cost_fw_{i+1}',
                             Var(within=NonNegativeReals))

#Cost_tu_u
for i in range(tu):
    MILP_model.add_component(f'Cost_tu_{i+1+wu}',
                             Var(within=NonNegativeReals))

if pipe_connection_costs_switch == 1:
    #Cost_fwpipe_su
    for i in range(s):
        for j in range(wu):
            MILP_model.add_component(f'Cost_fwpipe_{i+1}{j+1}',
                                     Var(within=NonNegativeReals))

    #Cost_uapipe_u,ua
    for i in range(u):
        for j in range(ua):
            MILP_model.add_component(f'Cost_uapipe_{i+1}{j+1}',
                                     Var(within=NonNegativeReals))

    #Cost_outpipe_ue
    for i in range(u):
        for j in range(e):
            MILP_model.add_component(f'Cost_outpipe_{i+1}{j+1}',
                                     Var(within=NonNegativeReals))

#O_cost
MILP_model.O_cost = Var(within=NonNegativeReals)

#M_terms
MILP_model.M_terms = Var(within=Reals)
```

```
#(v) Binary variables related to existence and/or nonexistence of connections

#B_fw_su
for i in range(s):
    for j in range(wu):
        MILP_model.add_component(f'B_fw_{i+1}{j+1}', Var(within=Binary))

#B_ua_u,ua = B_ua_ua,u
for i in range(u):
    for j in range(ua):
        MILP_model.add_component(f'B_ua_{i+1}{j+1}', Var(within=Binary))

#Remove water reuse:
if water_reuse_allowed_switch == 0:
    m = 0
    for i in range(tu):
        for j in range(wu):
            m += 1
            MILP_model.add_component(f'eqWR_{m}', Constraint(expr=(
                getattr(MILP_model, f'B_ua_{i+wu+1}{j+1}')
            ) == 0))

    for i in range(wu):
        for j in range(wu):
            m += 1
            MILP_model.add_component(f'eqWR_{m}', Constraint(expr=(
                getattr(MILP_model, f'B_ua_{i+1}{j+1}')
            ) == 0))

#B_out_ue
for i in range(u):
    for j in range(e):
        MILP_model.add_component(f'B_out_{i+1}{j+1}', Var(within=Binary))

#B_G1_u,ua
for i in range(u):
    for j in range(ua):
        MILP_model.add_component(f'B_G1_{i+1}{j+1}', Var(within=Binary))

#B_G2_u,ua
for i in range(u):
    for j in range(ua):
        MILP_model.add_component(f'B_G2_{i+1}{j+1}', Var(within=Binary))

#Summation of costs
MILP_model.Cost_fw_s_sum = Var(within=NonNegativeReals)
if pipe_connection_costs_switch == 1:
    MILP_model.Cost_fwpipe_su_sum = Var(within=NonNegativeReals)
    MILP_model.Cost_uapipe_ua_sum = Var(within=NonNegativeReals)
    MILP_model.Cost_outpipe_ue_sum = Var(within=NonNegativeReals)
MILP_model.Cost_tu_u_sum = Var(within=NonNegativeReals)

## DEFINE MILP_MODEL EQUATIONS

# (A) Balances around Operations, Mixers and Splitters
# (i) Overall balance around the entire water system:
# Equation 1
MILP_model.eq1 = Constraint(expr=(
    sum(getattr(MILP_model,
                f'F_w_{i+1}{j+1}') for i in range(s) for j in range(wu)) -
```

```
        sum(getattr(MILP_model,
                    f'F_out_{i+1}{j+1}') for i in range(u) for j in range(e)) -
        sum(F_loss_u[i] for i in range(u))
    ) == 0)

    # Equation 2
    #for water using operations
    for i in range(wu):
        MILP_model.add_component(f'eq2_{i+1}', Constraint(expr=(
            sum(getattr(MILP_model, f'F_ua_{j+1}{i+1}') for j in range(ua)) +
            sum(getattr(MILP_model, f'F_w_{j+1}{i+1}') for j in range(s)) -
            sum(getattr(MILP_model, f'F_ua_{i+1}{j+1}') for j in range(ua)) -
            sum(getattr(MILP_model, f'F_out_{i+1}{j+1}') for j in range(e))
        ) == F_loss_u[i]))

    #for treatment units
    for i in range(tu):
        MILP_model.add_component(f'eq2_{i+wu+1}', Constraint(expr=(
            sum(getattr(MILP_model, f'F_ua_{j+1}{i+wu+1}') for j in range(u)) -
            sum(getattr(MILP_model, f'F_ua_{i+wu+1}{j+1}') for j in range(u)) -
            sum(getattr(MILP_model, f'F_out_{i+wu+1}{j+1}') for j in range(e))
        ) == F_loss_u[i+wu]))

    # (ii) Contaminant mass balance for each operation (& each contaminant):
    # Equation 31
    # for water using operation
    m = 0
    for i in range(wu):
        for j in range(c):
            m += 1
            MILP_model.add_component(f'eq31_{m}', Constraint(expr=(
                sum(getattr(MILP_model, f'F_ua_{k+1}{i+1}') *
                    getattr(MILP_model, f'C_conk_{j+1}{k+1}') for k in range(ua)) +
                sum(getattr(MILP_model, f'F_w_{l+1}{i+1}') *
                    C_fw_cs[j] for l in range(s))
            ) == getattr(MILP_model, f'M_in_{j+1}{i+1}')))

    # for treatment units
    for i in range(tu):
        for j in range(c):
            m += 1
            MILP_model.add_component(f'eq31_{m}', Constraint(expr=(
                sum(getattr(MILP_model, f'F_ua_{k+1}{i+wu+1}') *
                    getattr(MILP_model, f'C_conk_{j+1}{k+1}') for k in range(ua))
            ) == getattr(MILP_model, f'M_in_{j+1}{i+wu+1}')))

    # Equation 32
    # for water using operation
    m = 0
    for i in range(wu):
        for j in range(c):
            m += 1
            MILP_model.add_component(f'eq32_{m}', Constraint(expr=(
                getattr(MILP_model, f'C_conk_{j+1}{i+1}') *
                (sum(getattr(MILP_model, f'F_w_{k+1}{i+1}') for k in range(s)) +
                sum(getattr(MILP_model, f'F_ua_{l+1}{i+1}') for l in range(ua)) -
                F_loss_u[j])
            ) == getattr(MILP_model, f'M_out_{j+1}{i+1}')))

    # for treatment units
    for i in range(tu):
```

```
        for j in range(c):
            m += 1
            MILP_model.add_component(f'eq32_{m}', Constraint(expr=(
                getattr(MILP_model, f'C_conk_{j+1}{i+wu+1}') *
                (sum(getattr(MILP_model, f'F_ua_{l+1}{i+wu+1}') for l in range(ua)) -
                 F_loss_u[j])
            ) == getattr(MILP_model, f'M_out_{j+1}{i+wu+1}')))

    # Equation 29
    m = 0
    for j in range(wu):
        for i in range(c):
            m += 1
            MILP_model.add_component(f'eq29_{m}', Constraint(expr=(
                getattr(MILP_model, f'M_in_{i+1}{j+1}') -
                getattr(MILP_model, f'M_out_{i+1}{j+1}') +
                L_ml_cu[m-1] -
                (F_loss_u[j] * C_loss_cu[j]) +
                getattr(MILP_model, f'M_loss_{i+1}{j+1}') -
                getattr(MILP_model, f'M_gain_{i+1}{j+1}')
            ) == 0))

    #Equation 30
    m = 0
    for j in range(tu):
        for i in range(c):
            m += 1
            MILP_model.add_component(f'eq30_{m}', Constraint(expr=(
                (1 - RR_cu.iloc[j][i]) *
                getattr(MILP_model, f'M_in_{i+1}{j+wu+1}') -
                (F_loss_u[j+wu] * C_loss_cu[j+wu]) -
                getattr(MILP_model, f'M_out_{i+1}{j+wu+1}') -
                getattr(MILP_model, f'M_gain_{i+1}{j+wu+1}')
            ) == 0))

    ## DEFINE MILP_MODEL CONSTRAINTS
    #(B) Availability and Capacity Constraints

    #(i) Constraints of water flow rate
    #Constraint 7 & Constraint 8
    for i in range(u):
        MILP_model.add_component(f'c7_{i+1}', Constraint(expr=(
            F_tmin_u[i]) <= getattr(MILP_model, f'F_t_{i+1}')))
        MILP_model.add_component(f'c8_{i+1}', Constraint(expr=(
            F_tmax_u[i]) >= getattr(MILP_model, f'F_t_{i+1}')))

    #for water using operations
    for i in range(wu):
        MILP_model.add_component(f'eq8_{i+1}', Constraint(expr=(
             sum(getattr(MILP_model, f'F_w_{j+1}{i+1}') for j in range(s)) +
             sum(getattr(MILP_model, f'F_ua_{k+1}{i+1}') for k in range(ua)) -
             F_loss_u[i]
        ) == getattr(MILP_model, f'F_t_{i+1}')))

    #for treatment units
    for i in range(tu):
        MILP_model.add_component(f'eq8_{i+wu+1}', Constraint(expr=(
             sum(getattr(MILP_model, f'F_ua_{k+1}{i+wu+1}') for k in range(ua)) -
             F_loss_u[wu+i]
        ) == getattr(MILP_model, f'F_t_{i+wu+1}')))
```

```
#(ii) Constraints on the quality and quantity of water
# Constraint 9
m = 0
for i in range(wu):
    for j in range(c):
        m += 1
        MILP_model.add_component(f'c9_{m}', Constraint(expr=(
            getattr(MILP_model, f'M_in_{j+1}{i+1}')
        ) <= C_inmax_cu[m-1] * getattr(MILP_model, f'F_t_{i+1}')))

#(C) Logic Constraints

#(i) Upper and lower bounds on the flow rates
# Constraint 11 & Constraint 12
m = 0
for i in range(wu):
    for j in range(s):
        m += 1
        MILP_model.add_component(f'c11_{m}', Constraint(expr=(
            getattr(MILP_model, f'F_w_{j+1}{i+1}') -
            U_fw[j][i] * getattr(MILP_model, f'B_fw_{j+1}{i+1}')
        ) <= 0))
        MILP_model.add_component(f'c12_{m}', Constraint(expr=(
            getattr(MILP_model, f'F_w_{j+1}{i+1}') -
            L_fw[j][i] * getattr(MILP_model, f'B_fw_{j+1}{i+1}')
        ) >= 0))

# Constraint 13 & Constraint 14
m = 0
for i in range(e):
    for j in range(u):
        m += 1
        MILP_model.add_component(f'c13_{m}', Constraint(expr=(
            getattr(MILP_model, f'F_out_{j+1}{i+1}') -
            U_out[j][i] * getattr(MILP_model, f'B_out_{j+1}{i+1}')
        ) <= 0))
        MILP_model.add_component(f'c14_{m}', Constraint(expr=(
            getattr(MILP_model, f'F_out_{j+1}{i+1}') -
            L_out[j][i] * getattr(MILP_model, f'B_out_{j+1}{i+1}')
        ) >= 0))

# Constraint 15 & Constraint 16
m = 0
for i in range(ua):
    for j in range(u):
        m += 1
        MILP_model.add_component(f'c15_{m}', Constraint(expr=(
            getattr(MILP_model, f'F_ua_{j+1}{i+1}') -
            U_ua[j][i] * getattr(MILP_model, f'B_ua_{j+1}{i+1}')
        ) <= 0))
        MILP_model.add_component(f'c16_{m}', Constraint(expr=(
            getattr(MILP_model, f'F_ua_{j+1}{i+1}') -
            L_ua[j][i] * getattr(MILP_model, f'B_ua_{j+1}{i+1}')
        ) >= 0))

# (ii) Maximum number of sources to feed each operation
#Constraint 17
#for water-using operations
for i in range(wu):
    MILP_model.add_component(f'c17_{i+1}', Constraint(expr=(
        sum(getattr(MILP_model, f'B_ua_{j+1}{i+1}') for j in range(ua)) +
```

```
            sum(getattr(MILP_model, f'B_fw_{k+1}{i+1}') for k in range(s))
        ) <= NS_max_u[i]))

    #for treatment units
    for i in range(tu):
        MILP_model.add_component(f'c17_{i+wu+1}', Constraint(expr=(
            sum(getattr(MILP_model, f'B_ua_{j+1}{i+wu+1}') for j in range(ua))
        ) <= NS_max_u[i+wu]))

    #Constraint 17a: self added --> Maximum number of sources to leave each operation
    for i in range(u):
        MILP_model.add_component(f'c17a_{i+1}', Constraint(expr=(
            sum(getattr(MILP_model, f'B_ua_{i+1}{j+1}') for j in range(ua)) +
            sum(getattr(MILP_model, f'B_out_{i+1}{k+1}') for k in range(e))
        ) <= NS_max_u[i]))

    # (iii) Elimination of regeneration recycling
    if regeneration_recycling_allowed_switch == 0:
        # Equation 18, Equation 19 & Equation 20
        m = 0
        for i in range(u):
            for j in range(tu):
                m += 1
                MILP_model.add_component(f'c18_{m}', Constraint(expr=(
                    getattr(MILP_model, f'B_ua_{i+1}{j+wu+1}') -
                    getattr(MILP_model, f'B_G1_{i+1}{j+wu+1}')
                ) <= 0))
                MILP_model.add_component(f'c19_{m}', Constraint(expr=(
                    getattr(MILP_model, f'B_ua_{j+wu+1}{i+1}') -
                    getattr(MILP_model, f'B_G2_{i+1}{j+wu+1}')
                ) <= 0))
                # MILP_model.add_component(f'c20_{m}', Constraint(expr=(
                #     getattr(MILP_model, f'B_G2_{i+1}{j+wu+1}')
                # ) ==  (1 - getattr(MILP_model, f'B_G1_{i+1}{j+wu+1}'))))
                # Equation 20, In case it can also be both 0:
                MILP_model.add_component(f'c20_{m}', Constraint(expr=(
                    getattr(MILP_model, f'B_G2_{i+1}{j+wu+1}') +
                    getattr(MILP_model, f'B_G1_{i+1}{j+wu+1}')
                ) <= 1))

        # Equation 21
        m = 0
        for i in range(u):
            for j in range(tu):
                for k in range(ua):
                    m += 1
                    MILP_model.add_component(f'c21_{m}', Constraint(expr=(
                        2 - (getattr(MILP_model, f'B_G2_{i+1}{j+wu+1}') +
                             getattr(MILP_model, f'B_G1_{k+1}{j+wu+1}'))
                    ) >= getattr(MILP_model, f'B_ua_{i+1}{k+1}')))

        # Equation 22
        for i in range(tu):
            MILP_model.add_component(f'c22_{i+1}', Constraint(expr=(
                sum(getattr(MILP_model, f'B_G1_{j+1}{i+wu+1}') for j in range(u)) +
                sum(getattr(MILP_model, f'B_G2_{j+1}{i+wu+1}') for j in range(u))
            ) == N__OP - 1))

    #(iv) Elimination of direct recycling
    # Equation 23
    m = 0
```

```
    for i in range(u):
        for j in range(ua):
            m += 1
            MILP_model.add_component(f'c23_{m}', Constraint(expr=(
                getattr(MILP_model, f'B_ua_{j+1}{i+1}') +
                getattr(MILP_model, f'B_ua_{i+1}{j+1}')
            ) <= 1))

    #(D) Objective Function

    #(i) Freshwater supply cost
    #Equation 24
    for i in range(s):
        MILP_model.add_component(f'eq24{i}', Constraint(expr=(
            getattr(MILP_model, f'Cost_fw_{i+1}') -
            CO_fw_s[i] * Operation_time *
            sum(getattr(MILP_model, f'F_w_{i+1}{j+1}') for j in range(wu))
        ) == 0))

    #(ii) Water and wastewater treatment cost  # Specific for each treatment system!
    for i in range(tu):
        if value(getattr(MILP_model, f'F_t_{i+wu+1}')) > 0:
            MILP_model.add_component(f'eq25{i}', Constraint(expr=(
                my_linearization_procedure(F_t_tu_min,
                                           F_t_tu_max,
                                           steps,
                                           CAPEX_value[i],
                                           CAPEX_power[i],
                                           OPEX_value[i])[0] *
                getattr(MILP_model, f'F_t_{i+wu+1}') +
                my_linearization_procedure(F_t_tu_min,
                                           F_t_tu_max,
                                           steps,
                                           CAPEX_value[i],
                                           CAPEX_power[i],
                                           OPEX_value[i])[1]
            ) == getattr(MILP_model, f'Cost_tu_{i+wu+1}')))
        elif value(getattr(MILP_model, f'F_t_{i+wu+1}')) == 0:
            MILP_model.add_component(f'eq25{i}', Constraint(expr=(
                getattr(MILP_model, f'Cost_tu_{i+wu+1}')) == 0))

    #(iii) Piping and sewer cost
    if pipe_connection_costs_switch == 1:
#         #Equation 26
#         m = 0
#         for i in range(s):
#             for j in range(wu):
#                 m += 1
#                 MILP_model.add_component(f'eq26_{m}', Constraint(expr=(
#                     getattr(MILP_model, f'F_w_{i+1}{j+1}')
#                 ) == getattr(MILP_model, f'A_fw_{i+1}{j+1}') * u_fw_su))

#         for i in range(u):
#             for j in range(ua):
#                 m += 1
#                 MILP_model.add_component(f'eq26_{m}', Constraint(expr=(
#                     getattr(MILP_model, f'F_ua_{i+1}{j+1}')
#                 ) == getattr(MILP_model, f'A_ua_{i+1}{j+1}') * u_ua_uua))

#         for i in range(u):
#             for j in range(e):
```

```
#                 m += 1
#                 MILP_model.add_component(f'eq26_{m}', Constraint(expr=(
#                     getattr(MILP_model, f'F_out_{i+1}{j+1}')
#                 ) == getattr(MILP_model, f'A_out_{i+1}{j+1}') * u_out_ue))

#         #Equation 26
#         m = 0
#         for i in range(s):
#             for j in range(wu):
#                 m += 1
#                 MILP_model.add_component(f'eq26_{m}', Constraint(expr=(
#                     getattr(MILP_model, f'A_fw_{i+1}{j+1}')
#                 ) == u_fw_su / getattr(MILP_model, f'F_w_{i+1}{j+1}')))

#         for i in range(u):
#             for j in range(ua):
#                 m += 1
#                 MILP_model.add_component(f'eq26_{m}', Constraint(expr=(
#                     getattr(MILP_model, f'A_ua_{i+1}{j+1}')
#                 ) == u_ua_uua / getattr(MILP_model, f'F_ua_{i+1}{j+1}')))

#         for i in range(u):
#             for j in range(e):
#                 m += 1
#                 MILP_model.add_component(f'eq26_{m}', Constraint(expr=(
#                     getattr(MILP_model, f'A_out_{i+1}{j+1}')
#                 ) == u_out_ue / getattr(MILP_model, f'F_out_{i+1}{j+1}')))

        #Equation 27
        m = 0
        for i in range(s):
            for j in range(wu):
                m += 1
                MILP_model.add_component(f'eq27_{m}', Constraint(expr=(
                    getattr(MILP_model, f'Cost_fwpipe_{i+1}{j+1}')
                ) == (d_fw_su.iloc[i][j] * Annualization_factor *
                      ((a_fw_su * getattr(MILP_model, f'A_fw_{i+1}{j+1}')) +
                        (b_fw_su * getattr(MILP_model, f'B_fw_{i+1}{j+1}'))))))

        for i in range(u):
            for j in range(ua):
                m += 1
                MILP_model.add_component(f'eq27_{m}', Constraint(expr=(
                    getattr(MILP_model, f'Cost_uapipe_{i+1}{j+1}')
                ) == (d_ua_uua.iloc[i+s][j] * Annualization_factor *
                      ((a_ua_uua * getattr(MILP_model, f'A_ua_{i+1}{j+1}')) +
                       (b_ua_uua * getattr(MILP_model, f'B_ua_{i+1}{j+1}'))))))

        for i in range(u):
            for j in range(e):
                m += 1
                MILP_model.add_component(f'eq27_{m}', Constraint(expr=(
                    getattr(MILP_model, f'Cost_outpipe_{i+1}{j+1}')
                ) == (d_out_ue.iloc[i+s] * Annualization_factor *
                      ((a_out_ue * getattr(MILP_model, f'A_out_{i+1}{j+1}')) +
                       (b_out_ue * getattr(MILP_model, f'B_out_{i+1}{j+1}'))))))

    #(iv) Overall objective function

    #Total costs per category
    MILP_model.eq28_1 = Constraint(expr=(
```

```
    sum(getattr(MILP_model, f'Cost_fw_{i+1}') for i in range(s))
) == MILP_model.Cost_fw_s_sum)
if pipe_connection_costs_switch == 1:
    MILP_model.eq28_2 = Constraint(expr=(
        MILP_model.Cost_fwpipe_su_sum -
        sum(getattr(
            MILP_model, f'Cost_fwpipe_{i+1}{j+1}') for i in range(s) for j in range(wu))
    ) == 0)
    MILP_model.eq28_3 = Constraint(expr=(
        MILP_model.Cost_uapipe_ua_sum -
        sum(getattr(
            MILP_model, f'Cost_uapipe_{i+1}{j+1}') for i in range(u) for j in range(ua))
    ) == 0)
    MILP_model.eq28_4 = Constraint(expr=(
        MILP_model.Cost_outpipe_ue_sum -
        sum(getattr(
            MILP_model, f'Cost_outpipe_{i+1}{j+1}') for i in range(u) for j in range(e))
    ) == 0)
MILP_model.eq28_5 = Constraint(expr=(
    MILP_model.Cost_tu_u_sum -
    sum(getattr(MILP_model, f'Cost_tu_{i+wu+1}') for i in range(tu))
) == 0)

if pipe_connection_costs_switch == 1:
    MILP_model.eq28_tot = Constraint(expr=(
        MILP_model.Cost_fw_s_sum +
        MILP_model.Cost_fwpipe_su_sum +
        MILP_model.Cost_uapipe_ua_sum +
        MILP_model.Cost_outpipe_ue_sum +
        MILP_model.Cost_tu_u_sum
    ) == MILP_model.O_cost)
else:
    MILP_model.eq28_tot = Constraint(expr=(
        MILP_model.Cost_fw_s_sum +
        MILP_model.Cost_tu_u_sum
    ) == MILP_model.O_cost)

# M_terms equation
MILP_model.eq36 = Constraint(expr=(
    sum(getattr(MILP_model, f'M_loss_{i+1}{j+1}') for i in range(c) for j in range(wu)) +
    (sum(getattr(MILP_model, f'M_gain_{i+1}{j+1}') for i in range(c) for j in range(u)))
) == MILP_model.M_terms)

## MILP_MODEL SOLVER
MILP_model.obj = Objective(expr=MILP_model.O_cost, sense=minimize)
SolverFactory(modules.find(MILP_solver),
              executable=MILP_solver_location).solve(MILP_model)#.write()

## PRINT MILP RESULTS
## Print results
if MILP_print_results_switch == 1:
    print("============================================================")
    #F_w_su
    for i in range(s):
        for j in range(wu):
            print(f'F_w_{i+1}{j+1}=',
                  value(getattr(MILP_model, f'F_w_{i+1}{j+1}')))

    #F_t_u
    for i in range(u):
        print(f'F_t_{i+1}=',
```

```
            value(getattr(MILP_model, f'F_t_{i+1}')))

#F_out_ue
for i in range(u):
    for j in range(e):
        print(f'F_out_{i+1}{j+1}=',
              value(getattr(MILP_model, f'F_out_{i+1}{j+1}')))

#F_ua_u,ua = F_ua_ua,u
for i in range(u):
    for j in range(ua):
        print(f'F_ua_{i+1}{j+1}=',
              value(getattr(MILP_model, f'F_ua_{i+1}{j+1}')))

#M_in_cu
for i in range(c):
    for j in range(u):
        print(f'M_in_{i+1}{j+1}=',
              value(getattr(MILP_model, f'M_in_{i+1}{j+1}')))

#M_out_cu
for i in range(c):
    for j in range(u):
        print(f'M_out_{i+1}{j+1}=',
              value(getattr(MILP_model, f'M_out_{i+1}{j+1}')))

#M_loss_cu
for i in range(c):
    for j in range(wu):
        print(f'M_loss_{i+1}{j+1}=',
              value(getattr(MILP_model, f'M_loss_{i+1}{j+1}')))

#M_gain_cu
for i in range(c):
    for j in range(u):
        print(f'M_gain_{i+1}{j+1}=',
              value(getattr(MILP_model, f'M_gain_{i+1}{j+1}')))

#C_conk_cu
for j in range(u):
    for i in range(c):
        print(f'C_conk_{i+1}{j+1} = ',
              value(getattr(MILP_model, f'C_conk_{i+1}{j+1}')))

#Cost_fw_s
for i in range(s):
        print(f'Cost_fw_{i+1}=',
              value(getattr(MILP_model, f'Cost_fw_{i+1}')))

#Cost_tu
for i in range(tu):
        print(f'Cost_tu_{i+wu+1}=',
              value(getattr(MILP_model, f'Cost_tu_{i+wu+1}')))

print('Cost_fw_s_sum=', MILP_model.Cost_fw_s_sum.value)
if pipe_connection_costs_switch == 1:
    print('Cost_fwpipe_su_sum=', MILP_model.Cost_fwpipe_su_sum.value)
    print('Cost_uapipe_ua_sum=', MILP_model.Cost_uapipe_ua_sum.value)
    print('Cost_outpipe_ue_sum=', MILP_model.Cost_outpipe_ue_sum.value)
print('Cost_tu_u_sum=', MILP_model.Cost_tu_u_sum.value)
```

```
    #Final costs
    print('O_cost.value=', MILP_model.O_cost.value)
    print("============================================================")

## SAVE MILP_MODEL RESULTS
Cost_results_list.append(MILP_model.O_cost.value)
M_terms_results_list.append(MILP_model.M_terms.value)

#F_w_su
for i in range(s):
    for j in range(wu):
        MILP_flow_results_list.append(value(getattr(MILP_model, f'F_w_{i+1}{j+1}')))
#F_out_ue
for i in range(u):
    for j in range(e):
        MILP_flow_results_list.append(value(getattr(MILP_model, f'F_out_{i+1}{j+1}')))
#F_ua_u,ua = F_ua_ua,u
for i in range(u):
    for j in range(ua):
        MILP_flow_results_list.append(value(getattr(MILP_model, f'F_ua_{i+1}{j+1}')))

#F_t_u
for i in range(u):
    MILP_no_treatment_flows.append(value(getattr(MILP_model, f'F_t_{i+1}')))

#B_ua_u,ua --> only for between WU
for i in range(wu):
    for j in range(wu):
        MILP_binary_variables_list.append(value(getattr(MILP_model, f'B_ua_{i+1}{j+1}')))

## DELETE ALL VARIABLES, CONSTRAINTS, AND EQUATIONS

# Delete variables
MILP_model.del_component(f'O_cost')
MILP_model.del_component(f'M_terms')
MILP_model.del_component(f'Cost_fw_s_sum')
MILP_model.del_component(f'Cost_fwpipe_su_sum')
MILP_model.del_component(f'Cost_uapipe_ua_sum')
MILP_model.del_component(f'Cost_outpipe_ue_sum')
MILP_model.del_component(f'Cost_tu_u_sum')
MILP_model.del_component(f'obj')

#F_w_su
for i in range(s):
    for j in range(wu):
        MILP_model.del_component(f'F_w_{i+1}{j+1}')

#F_out_ue
for i in range(u):
    for j in range(e):
        MILP_model.del_component(f'F_out_{i+1}{j+1}')

#F_ua_u,ua = F_ua_ua,u
for i in range(u):
    for j in range(ua):
        MILP_model.del_component(f'F_ua_{i+1}{j+1}')

#F_t_u
for i in range(u):
    MILP_model.del_component(f'F_t_{i+1}')
```

```
    #(ii) Contaminant concentration and mass flow in process operations

    #C_conk_cu, M_in_cu, M_out_cu & M_gain_cu
    for i in range(c):
        for j in range(u):
            MILP_model.del_component(f'C_conk_{i+1}{j+1}')
            MILP_model.del_component(f'M_in_{i+1}{j+1}')
            MILP_model.del_component(f'M_out_{i+1}{j+1}')
            MILP_model.del_component(f'M_gain_{i+1}{j+1}')

    #M_loss_cu
    for i in range(c):
        for j in range(wu):
            MILP_model.del_component(f'M_loss_{i+1}{j+1}')

    #(iii) Cross-sectional area of the pipes connecting different operations
    if pipe_connection_costs_switch == 1:
        #A_fw_su
        for i in range(s):
            for j in range(wu):
                MILP_model.del_component(f'A_fw_{i+1}{j+1}')

        #A_ua_uua
        for i in range(u):
            for j in range(ua):
                MILP_model.del_component(f'A_ua_{i+1}{j+1}')

        #A_out_ue
        for i in range(u):
            for j in range(e):
                MILP_model.del_component(f'A_out_{i+1}{j+1}')

    #(iv) Cost terms in the objective function

    #Cost_fw_s
    for i in range(s):
        MILP_model.del_component(f'Cost_fw_{i+1}')

    #Cost_tu_u
    for i in range(tu):
        MILP_model.del_component(f'Cost_tu_{i+1+wu}')

    if pipe_connection_costs_switch == 1:
        #Cost_fwpipe_su
        for i in range(s):
            for j in range(wu):
                MILP_model.del_component(f'Cost_fwpipe_{i+1}{j+1}')

        #Cost_uapipe_u,ua
        for i in range(u):
            for j in range(ua):
                MILP_model.del_component(f'Cost_uapipe_{i+1}{j+1}')

        #Cost_outpipe_ue
        for i in range(u):
            for j in range(e):
                MILP_model.del_component(f'Cost_outpipe_{i+1}{j+1}')


    #(v) Binary variables related to existence and/or nonexistence of connections
```

```
#B_fw_su
for i in range(s):
    for j in range(wu):
        MILP_model.del_component(f'B_fw_{i+1}{j+1}')

#B_out_ue
for i in range(u):
    for j in range(e):
        MILP_model.del_component(f'B_out_{i+1}{j+1}')

#B_ua_u,ua, B_G1_u,ua, B_G2_u,ua
for i in range(u):
    for j in range(ua):
        MILP_model.del_component(f'B_ua_{i+1}{j+1}')
        MILP_model.del_component(f'B_G1_{i+1}{j+1}')
        MILP_model.del_component(f'B_G2_{i+1}{j+1}')

# Delete constraints & Equations
# Equation 1
MILP_model.del_component(f'eq1')

# Equation 2
for i in range(u):
        MILP_model.del_component(f'eq2_{i+1}')

# Equation 31
# for water using operation
m = 0
for i in range(wu):
    for j in range(c):
        m += 1
        MILP_model.del_component(f'eq31_{m}')

# for treatment units
for i in range(tu):
    for j in range(c):
        m += 1
        MILP_model.del_component(f'eq31_{m}')

# Equation 32
# for water using operation
m = 0
for i in range(wu):
    for j in range(c):
        m += 1
        MILP_model.del_component(f'eq32_{m}')

# for treatment units
for i in range(tu):
    for j in range(c):
        m += 1
        MILP_model.del_component(f'eq32_{m}')

# Equation 29
m = 0
for j in range(wu):
    for i in range(c):
        m += 1
        MILP_model.del_component(f'eq29_{m}')

#Equation 30
```

```
m = 0
for i in range(c):
    for j in range(tu):
        m += 1
        MILP_model.del_component(f'eq30_{m}')

## DELETE MILP_MODEL CONSTRAINTS
#Constraint 7, Constraint 8 & Equation 8
for i in range(u):
    MILP_model.del_component(f'c7_{i+1}')
    MILP_model.del_component(f'c8_{i+1}')

for i in range(u):
    MILP_model.del_component(f'eq8_{i+1}')

# Constraint 9
m = 0
for i in range(wu):
    for j in range(c):
        m += 1
        MILP_model.del_component(f'c9_{m}')
        MILP_model.del_component(f'c9a_{m}')

if include_environmental_limit_switch == 1:
    # Constraint 10
    for i in range(c):
        MILP_model.del_component(f'c10_{i+1}')

# Constraint 11 & Constraint 12
m = 0
for i in range(wu):
    for j in range(s):
        m += 1
        MILP_model.del_component(f'c11_{m}')
        MILP_model.del_component(f'c12_{m}')

# Constraint 13 & Constraint 14
m = 0
for i in range(e):
    for j in range(u):
        m += 1
        MILP_model.del_component(f'c13_{m}')
        MILP_model.del_component(f'c14_{m}')

# Constraint 15 & Constraint 16
m = 0
for i in range(ua):
    for j in range(u):
        m += 1
        MILP_model.del_component(f'c15_{m}')
        MILP_model.del_component(f'c16_{m}')

#Constraint 17
for i in range(u):
    MILP_model.del_component(f'c17_{i+1}')
    MILP_model.del_component(f'c17a_{i+1}')

if regeneration_recycling_allowed_switch == 0:
    # Equation 18, Equation 19 & Equation 20
    m = 0
    for i in range(u):
```

```
            for j in range(tu):
                m += 1
                MILP_model.del_component(f'c18_{m}')
                MILP_model.del_component(f'c19_{m}')
                MILP_model.del_component(f'c20_{m}')

        # Equation 21
        m = 0
        for i in range(u):
            for j in range(tu):
                for k in range(ua):
                    m += 1
                    MILP_model.del_component(f'c21_{m}')

        # Equation 22
        for i in range(tu):
            MILP_model.del_component(f'c22_{i+1}')

    #(iv) Elimination of direct recycling
    # Equation 23
    m = 0
    for i in range(u):
        for j in range(ua):
            m += 1
            MILP_model.del_component(f'c23_{m}')

    # Equation 24
    for i in range(s):
        MILP_model.del_component(f'eq24{i}')

    # Equation 25
    for i in range(tu):
        MILP_model.del_component(f'eq25{i}')

    #(iii) Piping cost
    if pipe_connection_costs_switch == 1:
        #Equations 26 & 27
        m = 0
        for i in range(s):
            for j in range(wu):
                m += 1
                # MILP_model.del_component(f'eq26_{m}')
                MILP_model.del_component(f'eq27_{m}')

        for i in range(u):
            for j in range(ua):
                m += 1
                # MILP_model.del_component(f'eq26_{m}')
                MILP_model.del_component(f'eq27_{m}')

        for i in range(u):
            for j in range(e):
                m += 1
                # MILP_model.del_component(f'eq26_{m}')
                MILP_model.del_component(f'eq27_{m}')

    m = 0
    for j in range(u):
        for i in range(c):
            m += 1
            MILP_model.del_component(f'eq_fix_concentrations_{m}')
```

```
    #Remove water reuse:
    if water_reuse_allowed_switch == 0:
        m = 0
        for i in range(tu):
            for j in range(wu):
                m += 1
                MILP_model.del_component(f'eqWR_{m}')

        for i in range(wu):
            for j in range(wu):
                m += 1
                MILP_model.del_component(f'eqWR_{m}')

    MILP_model.del_component(f'eq28_1')
    if pipe_connection_costs_switch == 1:
            MILP_model.del_component(f'eq28_2')
            MILP_model.del_component(f'eq28_3')
            MILP_model.del_component(f'eq28_4')
    MILP_model.del_component(f'eq28_5')
    MILP_model.del_component(f'eq28_tot')
    MILP_model.del_component(f'eq36')
    return MILP_flow_results_list, MILP_binary_variables_list, MILP_no_treatment_flows

## LP FUNCTION
LP_model = ConcreteModel()
def LP_function(
    LP_input_list
):

    #DEFINE LIST FOR FLOW RESULTS
    LP_concentrations_results_list = []

    ## DEFINE LP_MODEL DECISION VARIABLES
    #(i) Continuous variables associated with flow rates
    #F_w_su
    m = 0
    for i in range(s):
        for j in range(wu):
            m += 1
            LP_model.add_component(f'F_w_{i+1}{j+1}',
                                   Var(within=NonNegativeReals))
            LP_model.add_component(f'eq_fix_flows_{m}', Constraint(expr=(
                getattr(LP_model, f'F_w_{i+1}{j+1}')
            ) == LP_input_list[m-1]))

    #F_out_ue
    for i in range(u):
        for j in range(e):
            m += 1
            LP_model.add_component(f'F_out_{i+1}{j+1}',
                                   Var(within=NonNegativeReals))
            LP_model.add_component(f'eq_fix_flows_{m}', Constraint(expr=(
                getattr(LP_model, f'F_out_{i+1}{j+1}')
            ) == LP_input_list[m-1]))

    #F_ua_u,ua = F_ua_ua,u
    for i in range(u):
        for j in range(ua):
            m += 1
            LP_model.add_component(f'F_ua_{i+1}{j+1}',
```

```
                                Var(within=NonNegativeReals, initialize=0))
        LP_model.add_component(f'eq_fix_flows_{m}', Constraint(expr=(
            getattr(LP_model, f'F_ua_{i+1}{j+1}')
        ) == LP_input_list[m-1]))

#F_t_u
for i in range(u):
    LP_model.add_component(f'F_t_{i+1}',
                           Var(within=NonNegativeReals,
                               bounds=(F_tmin_u[i], F_tmax_u[i])))

#(ii) Contaminant concentration and mass flow in process operations
m = 0
for j in range(wu):
    for i in range(c):
        m += 1
        LP_model.add_component(f'C_out_{i+1}{j+1}',
                               Var(within=NonNegativeReals,
                                   bounds=(0, C_outmax_cu[m-1])))

for j in range(tu):
    for i in range(c):
        m += 1
        LP_model.add_component(f'C_out_{i+1}{j+wu+1}',
                               Var(within=NonNegativeReals))

#M_in_cu & M_out_cu
m=0
for i in range(c):
    for j in range(u):
        m+=1
        LP_model.add_component(f'M_in_{i+1}{j+1}',
                               Var(within=NonNegativeReals))
        LP_model.add_component(f'M_out_{i+1}{j+1}',
                               Var(within=NonNegativeReals))

#M_loss_cu
for i in range(c):
    for j in range(wu):
        LP_model.add_component(f'M_loss_{i+1}{j+1}',
                               Var(within=NonNegativeReals,
                                   initialize=0))

#M_gain_cu
for i in range(c):
    for j in range(u):
        LP_model.add_component(f'M_gain_{i+1}{j+1}',
                               Var(within=NonNegativeReals,
                                   initialize=0))

#M_terms
LP_model.M_terms = Var(within=Reals, initialize=0) #, bounds=(-5,5))

# (ii) Contaminant mass balance for each operation (& each contaminant):
# Equation 34
# for water using operation
m = 0
for i in range(wu):
    for j in range(c):
        m += 1
        LP_model.add_component(f'eq34_{m}', Constraint(expr=(
```

```
                sum(getattr(LP_model, f'F_ua_{k+1}{i+1}') *
                    getattr(LP_model, f'C_out_{j+1}{k+1}') for k in range(ua)) +
                sum(getattr(LP_model, f'F_w_{k+1}{i+1}') *
                    C_fw_cs[j] for k in range(s))
            ) == getattr(LP_model, f'M_in_{j+1}{i+1}')))

    # for treatment units
    for i in range(tu):
        for j in range(c):
            m += 1
            LP_model.add_component(f'eq34_{m}', Constraint(expr=(
                sum((getattr(LP_model, f'F_ua_{k+1}{i+wu+1}') *
                     getattr(LP_model, f'C_out_{j+1}{k+1}')) for k in range(ua))
            ) == getattr(LP_model, f'M_in_{j+1}{i+wu+1}')))

    # Equation 35
    # for water using operation
    m = 0
    for i in range(wu):
        for j in range(c):
            m += 1
            LP_model.add_component(f'eq35_{m}', Constraint(expr=(
                getattr(LP_model, f'C_out_{j+1}{i+1}') *
                (sum(getattr(LP_model, f'F_w_{k+1}{i+1}') for k in range(s)) +
                sum(getattr(LP_model, f'F_ua_{l+1}{i+1}') for l in range(ua)) -
                F_loss_u[j])
            ) == getattr(LP_model, f'M_out_{j+1}{i+1}')))

    # for treatment units
    for i in range(tu):
        for j in range(c):
            m += 1
            LP_model.add_component(f'eq35_{m}', Constraint(expr=(
                (getattr(LP_model, f'C_out_{j+1}{i+wu+1}') *
                (sum(getattr(LP_model, f'F_ua_{l+1}{i+wu+1}') for l in range(ua)) -
                 F_loss_u[j]))
            ) == getattr(LP_model, f'M_out_{j+1}{i+wu+1}')))

    # Equation 29
    m = 0
    for j in range(wu):
        for i in range(c):
            m += 1
            LP_model.add_component(f'eq29_{m}', Constraint(expr=(
                getattr(LP_model, f'M_in_{i+1}{j+1}') -
                getattr(LP_model, f'M_out_{i+1}{j+1}') +
                L_ml_cu[m-1] -
                (F_loss_u[j]) * (C_loss_cu[j]) +
                getattr(LP_model, f'M_loss_{i+1}{j+1}') -
                getattr(LP_model, f'M_gain_{i+1}{j+1}')
            ) == 0))

    #Equation 30
    m = 0
    for i in range(c):
        for j in range(tu):
            m += 1
            LP_model.add_component(f'eq30_{m}', Constraint(expr=(
                (1 - RR_cu.iloc[j][i]) * getattr(LP_model, f'M_in_{i+1}{j+wu+1}') -
                (F_loss_u[j+wu] * C_loss_cu[j+wu]) -
                getattr(LP_model, f'M_out_{i+1}{j+wu+1}') -
```

```
            getattr(LP_model, f'M_gain_{i+1}{j+wu+1}')
        ) == 0))

# M_terms equation
LP_model.eq36 = Constraint(expr=(
    sum(getattr(LP_model, f'M_loss_{i+1}{j+1}') for i in range(c) for j in range(wu)) +
    sum(getattr(LP_model, f'M_gain_{i+1}{j+1}') for i in range(c) for j in range(u))
) == LP_model.M_terms)

## DEFINE LP_MODEL CONSTRAINTS
#(B) Availability and Capacity Constraints

# #(i) Constraints of water flow rate
#for water using operations
for i in range(wu):
    LP_model.add_component(f'eq8_{i+1}', Constraint(expr=(
        sum(getattr(LP_model, f'F_w_{j+1}{i+1}') for j in range(s)) +
        sum(getattr(LP_model, f'F_ua_{k+1}{i+1}') for k in range(ua)) -
        F_loss_u[i]
    ) == getattr(LP_model, f'F_t_{i+1}')))

#for treatment units
for i in range(tu):
    LP_model.add_component(f'eq8_{i+wu+1}', Constraint(expr=(
        sum(getattr(LP_model, f'F_ua_{k+1}{i+wu+1}') for k in range(ua)) -
        F_loss_u[wu+i]
    ) == getattr(LP_model, f'F_t_{i+wu+1}')))

## MODEL SOLVER
LP_model.obj = Objective(expr=LP_model.M_terms, sense=minimize)
# SolverFactory('gurobi').solve(LP_model) #, tee=True).write()
SolverFactory(modules.find(LP_solver), executable=LP_solver_location).solve(LP_model)

## SAVE LP_MODEL RESULTS
# Cost_results_list.append(LP_model.O_cost.value)
Cost_results_list.append('nope')
M_terms_results_list.append(LP_model.M_terms.value)
m = 0
for j in range(wu):
    for i in range(c):
        m += 1
        LP_concentrations_results_list.append(value(getattr(LP_model,
                                                            f'C_out_{i+1}{j+1}')))

for j in range(tu):
    for i in range(c):
        m += 1
        LP_concentrations_results_list.append(value(getattr(LP_model,
                                                            f'C_out_{i+1}{j+wu+1}')))

## Print results
if LP_print_results_switch == 1:
    print("============================================================")
    #F_w_su
    for i in range(s):
        for j in range(wu):
            print(f'F_w_{i+1}{j+1}=',
                  value(getattr(LP_model, f'F_w_{i+1}{j+1}')))

    #F_t_u
    for i in range(u):
```

```
            print(f'F_t_{i+1}=',
                  value(getattr(LP_model, f'F_t_{i+1}')))

        #F_out_ue
        for i in range(u):
            for j in range(e):
                print(f'F_out_{i+1}{j+1}=',
                      value(getattr(LP_model, f'F_out_{i+1}{j+1}')))

        #F_ua_u,ua = F_ua_ua,u
        for i in range(u):
            for j in range(ua):
                print(f'F_ua_{i+1}{j+1}=',
                      value(getattr(LP_model, f'F_ua_{i+1}{j+1}')))

        #M_in_cu
        for i in range(c):
            for j in range(u):
                print(f'M_in_{i+1}{j+1}=',
                      value(getattr(LP_model, f'M_in_{i+1}{j+1}')))

        #M_out_cu
        for i in range(c):
            for j in range(u):
                print(f'M_out_{i+1}{j+1}=',
                      value(getattr(LP_model, f'M_out_{i+1}{j+1}')))

        #M_loss_cu
        for i in range(c):
            for j in range(wu):
                print(f'M_loss_{i+1}{j+1}=',
                      value(getattr(LP_model, f'M_loss_{i+1}{j+1}')))

        #M_gain_cu
        for i in range(c):
            for j in range(u):
                print(f'M_gain_{i+1}{j+1}=',
                      value(getattr(LP_model, f'M_gain_{i+1}{j+1}')))

        #C_out_cu
        for j in range(u):
            for i in range(c):
                print(f'C_out_{i+1}{j+1} = ',
                      value(getattr(LP_model, f'C_out_{i+1}{j+1}')))

        #M_terms
        print('M_terms.value=', LP_model.M_terms.value)
        print("============================================================")

    ## DELETE ALL VARIABLES, CONSTRAINTS, AND EQUATIONS

    # Delete variables
    LP_model.del_component(f'M_terms')
    LP_model.del_component(f'obj')

    #F_w_su
    for i in range(s):
        for j in range(wu):
            LP_model.del_component(f'F_w_{i+1}{j+1}')

    #F_out_ue
```

```
for i in range(u):
    for j in range(e):
        LP_model.del_component(f'F_out_{i+1}{j+1}')

#F_ua_u,ua = F_ua_ua,u
for i in range(u):
    for j in range(ua):
        LP_model.del_component(f'F_ua_{i+1}{j+1}')

#F_t_u
for i in range(u):
    LP_model.del_component(f'F_t_{i+1}')

#(ii) Contaminant concentration and mass flow in process operations

#C_out_cu, M_in_cu & M_out_cu
for i in range(c):
    for j in range(u):
        LP_model.del_component(f'C_out_{i+1}{j+1}')
        LP_model.del_component(f'M_in_{i+1}{j+1}')
        LP_model.del_component(f'M_out_{i+1}{j+1}')

#M_loss_cu
for i in range(c):
    for j in range(wu):
        LP_model.del_component(f'M_loss_{i+1}{j+1}')

#M_gain_cu
for i in range(c):
    for j in range(u):
        LP_model.del_component(f'M_gain_{i+1}{j+1}')

#Equations
m = 0
for i in range(u):
    for j in range(c):
        m += 1
        LP_model.del_component(f'eq34_{m}')
        LP_model.del_component(f'eq35_{m}')

m = 0
for j in range(wu):
    for i in range(c):
        m += 1
        LP_model.del_component(f'eq29_{m}')

m = 0
for j in range(tu):
    for i in range(c):
        m += 1
        LP_model.del_component(f'eq30_{m}')

for i in range(u):
    LP_model.del_component(f'eq8_{i+1}')

for i in range(s):
    LP_model.del_component(f'eq24_{i}')

for i in range(tu):
    LP_model.del_component(f'eq25_{i}')
```

```
    LP_model.del_component(f'eq36')

    #Delete fix_flows function
    m = 0
    for i in range(s):
        for j in range(wu):
            m += 1
            LP_model.del_component(f'eq_fix_flows_{m}')

    for i in range(u):
        for j in range(e):
            m += 1
            LP_model.del_component(f'eq_fix_flows_{m}')

    for i in range(u):
        for j in range(ua):
            m += 1
            LP_model.del_component(f'eq_fix_flows_{m}')

    return LP_concentrations_results_list

## MINLP FUNCTION
MINLP_model = ConcreteModel()
def MINLP_function(
    MILP_output_list,
    MILP_binary_list,
    LP_output_list,
    MILP_F_t_list
):

    ## DEFINE MINLP_MODEL DECISION VARIABLES
    #F_w_su
    m = 0
    for i in range(s):
        for j in range(wu):
            m += 1
            if iteration_activated_switch == 1:
                MINLP_model.add_component(f'F_w_{i+1}{j+1}',
                                          Var(within=NonNegativeReals,
                                              initialize=MILP_output_list[m-1],
                                              bounds=(MILP_output_list[m-1]-2,
                                                      MILP_output_list[m-1]+2)))
            elif iteration_activated_switch == 0:
                MINLP_model.add_component(f'F_w_{i+1}{j+1}',
                                          Var(within=NonNegativeReals))

    #F_out_ue
    for i in range(u):
        for j in range(e):
            m += 1
            MINLP_model.add_component(f'F_out_{i+1}{j+1}',
                                      Var(within=NonNegativeReals,
                                          initialize=MILP_output_list[m-1]))

    #F_ua_u,ua = F_ua_ua,u
    for i in range(u):
        for j in range(ua):
            m += 1
            MINLP_model.add_component(f'F_ua_{i+1}{j+1}',
                                      Var(within=NonNegativeReals,
                                          initialize=MILP_output_list[m-1]))
```

```
#F_t_u
m = 0
for i in range(u):
    if iteration_activated_switch == 1:
        MINLP_model.add_component(f'F_t_{i+1}',
                                  Var(within=NonNegativeReals,
                                      bounds=(F_tmin_u[i], F_tmax_u[i])))
    elif iteration_activated_switch == 0:
        m += 1
        MINLP_model.add_component(f'F_t_{i+1}',
                                  Var(within=NonNegativeReals,
                                      bounds=(F_tmin_u[i], MILP_F_t_list[m-1])))

#(ii) Contaminant concentration and mass flow in process operations
m = 0
for j in range(wu):
    for i in range(c):
        m += 1
        MINLP_model.add_component(f'C_out_{i+1}{j+1}',
                                  Var(within=NonNegativeReals,
                                      bounds=(0, C_outmax_cu[m-1]),
                                      initialize=LP_output_list[m-1]))

for j in range(tu):
    for i in range(c):
        m += 1
        MINLP_model.add_component(f'C_out_{i+1}{j+wu+1}',
                                  Var(within=NonNegativeReals,
                                      initialize=LP_output_list[m-1]))

#M_in_cu & M_out_cu
m=0
for i in range(c):
    for j in range(u):
        m+=1
        MINLP_model.add_component(f'M_in_{i+1}{j+1}',
                                  Var(within=NonNegativeReals))
        MINLP_model.add_component(f'M_out_{i+1}{j+1}',
                                  Var(within=NonNegativeReals))

# (iii) Cross-sectional area of the pipes connecting different operations
if pipe_connection_costs_switch == 1:
    #A_fw_su
    for i in range(s):
        for j in range(wu):
            MINLP_model.add_component(f'A_fw_{i+1}{j+1}',
                                      Var(within=NonNegativeReals))

    #A_ua_uua
    for i in range(u):
        for j in range(ua):
            MINLP_model.add_component(f'A_ua_{i+1}{j+1}',
                                      Var(within=NonNegativeReals))

    #A_out_ue
    for i in range(u):
        for j in range(e):
            MINLP_model.add_component(f'A_out_{i+1}{j+1}',
                                      Var(within=NonNegativeReals))
```

```
#(iv) Cost terms in the objective function

#Cost_fw_s
for i in range(s):
    MINLP_model.add_component(f'Cost_fw_{i+1}',
                              Var(within=NonNegativeReals))

#Cost_tu_u
for i in range(tu):
    MINLP_model.add_component(f'Cost_tu_{i+1+wu}',
                              Var(within=NonNegativeReals))

if pipe_connection_costs_switch == 1:
    #Cost_fwpipe_su
    for i in range(s):
        for j in range(wu):
            MINLP_model.add_component(f'Cost_fwpipe_{i+1}{j+1}',
                                      Var(within=NonNegativeReals))

    #Cost_uapipe_u,ua
    for i in range(u):
        for j in range(ua):
            MINLP_model.add_component(f'Cost_uapipe_{i+1}{j+1}',
                                      Var(within=NonNegativeReals))

    #Cost_outpipe_ue
    for i in range(u):
        for j in range(e):
            MINLP_model.add_component(f'Cost_outpipe_{i+1}{j+1}',
                                      Var(within=NonNegativeReals))

#O_cost
MINLP_model.O_cost = Var(within=NonNegativeReals)

#(v) Binary variables related to existence and/or nonexistence of connections

#B_fw_su
for i in range(s):
    for j in range(wu):
        MINLP_model.add_component(f'B_fw_{i+1}{j+1}', Var(within=Binary))

#B_ua_u,ua = B_ua_ua,u
for i in range(u):
    for j in range(ua):
        MINLP_model.add_component(f'B_ua_{i+1}{j+1}', Var(within=Binary))

#Fix binary variables reuse flows:
if iteration_activated_switch == 1:
    m = 0
    for i in range(wu):
        for j in range(wu):
            m += 1
            MINLP_model.add_component(f'eqfixbin_{m}', Constraint(expr=(
                getattr(MINLP_model, f'B_ua_{i+1}{j+1}')
            ) == MILP_binary_list[m-1]))

#Remove water reuse:
if water_reuse_allowed_switch == 0:
    m = 0
    for i in range(tu):
        for j in range(wu):
```

```
            m += 1
            MINLP_model.add_component(f'eqWR_{m}', Constraint(expr=(
                getattr(MINLP_model, f'B_ua_{i+wu+1}{j+1}')
            ) == 0))

    for i in range(wu):
        for j in range(wu):
            m += 1
            MINLP_model.add_component(f'eqWR_{m}', Constraint(expr=(
                getattr(MINLP_model, f'B_ua_{i+1}{j+1}')
            ) == 0))

#B_out_ue
for i in range(u):
    for j in range(e):
        MINLP_model.add_component(f'B_out_{i+1}{j+1}', Var(within=Binary))

if regeneration_recycling_allowed_switch == 0:
    #B_G1_u,ua & B_G2_u,ua
    for i in range(u):
        for j in range(ua):
            MINLP_model.add_component(f'B_G1_{i+1}{j+1}', Var(within=Binary))
            MINLP_model.add_component(f'B_G2_{i+1}{j+1}', Var(within=Binary))

#Summation of costs
MINLP_model.Cost_fw_s_sum = Var(within=NonNegativeReals)
if pipe_connection_costs_switch == 1:
    MINLP_model.Cost_fwpipe_su_sum = Var(within=NonNegativeReals)
    MINLP_model.Cost_uapipe_ua_sum = Var(within=NonNegativeReals)
    MINLP_model.Cost_outpipe_ue_sum = Var(within=NonNegativeReals)
MINLP_model.Cost_tu_u_sum = Var(within=NonNegativeReals)

## DEFINE MINLP_MODEL EQUATIONS

# (A) Balances around Operations, Mixers and Splitters
# (i) Overall balance around the entire water system:
# Equation 1
MINLP_model.eq1 = Constraint(expr=(
    (sum(getattr(
        MINLP_model, f'F_w_{i+1}{j+1}') for i in range(s) for j in range(wu))) -
    (sum(getattr(
        MINLP_model, f'F_out_{i+1}{j+1}') for i in range(u) for j in range(e))) -
    (sum(F_loss_u[i] for i in range(u)))
== 0))

# Equation 2
#for water using operations
for i in range(wu):
    MINLP_model.add_component(f'eq2_{i+1}', Constraint(expr=(
        (sum(getattr(MINLP_model, f'F_ua_{j+1}{i+1}') for j in range(ua))) +
        (sum(getattr(MINLP_model, f'F_w_{j+1}{i+1}') for j in range(s))) -
        (sum(getattr(MINLP_model, f'F_ua_{i+1}{j+1}') for j in range(ua))) -
        (sum(getattr(MINLP_model, f'F_out_{i+1}{j+1}') for j in range(e))) -
        F_loss_u[i]
    ) == 0))

#for treatment units
for i in range(tu):
    MINLP_model.add_component(f'eq2_{i+wu+1}', Constraint(expr=(
        (sum(getattr(MINLP_model, f'F_ua_{j+1}{i+wu+1}') for j in range(ua))) -
        (sum(getattr(MINLP_model, f'F_ua_{i+wu+1}{j+1}') for j in range(ua))) -
```

```
            (sum(getattr(MINLP_model, f'F_out_{i+wu+1}{j+1}') for j in range(e))) -
            F_loss_u[i+wu]
        ) == 0))

    # (ii) Contaminant mass balance for each operation (& each contaminant):
    # Equation 3
    # for water using operation
    m = 0
    for i in range(wu):
        for j in range(c):
            m += 1
            MINLP_model.add_component(f'eq3_{m}', Constraint(expr=(
                (sum(getattr(MINLP_model, f'F_ua_{k+1}{i+1}') *
                     getattr(MINLP_model, f'C_out_{j+1}{k+1}') for k in range(ua))) +
                (sum(getattr(MINLP_model, f'F_w_{k+1}{i+1}') *
                     C_fw_cs[j] for k in range(s)))
            ) == getattr(MINLP_model, f'M_in_{j+1}{i+1}')))

    # for treatment units
    for i in range(tu):
        for j in range(c):
            m += 1
            MINLP_model.add_component(f'eq3_{m}', Constraint(expr=(
                (sum(getattr(MINLP_model, f'F_ua_{k+1}{i+wu+1}') *
                     getattr(MINLP_model, f'C_out_{j+1}{k+1}') for k in range(ua)))
            ) == getattr(MINLP_model, f'M_in_{j+1}{i+wu+1}')))

    # Equation 4
    # for water using operation
    m = 0
    for i in range(wu):
        for j in range(c):
            m += 1
            MINLP_model.add_component(f'eq4_{m}', Constraint(expr=(
                getattr(MINLP_model, f'C_out_{j+1}{i+1}') *
                (sum(getattr(MINLP_model, f'F_w_{k+1}{i+1}') for k in range(s)) +
                sum(getattr(MINLP_model, f'F_ua_{l+1}{i+1}') for l in range(ua)) -
                F_loss_u[j])
            ) == getattr(MINLP_model, f'M_out_{j+1}{i+1}')))

    # for treatment units
    for i in range(tu):
        for j in range(c):
            m += 1
            MINLP_model.add_component(f'eq4_{m}', Constraint(expr=(
                (getattr(MINLP_model, f'C_out_{j+1}{i+wu+1}') *
                (sum(getattr(MINLP_model, f'F_ua_{l+1}{i+wu+1}') for l in range(ua)) -
                 F_loss_u[j]))
            ) == getattr(MINLP_model, f'M_out_{j+1}{i+wu+1}')))

    # Equation 5
    m = 0
    for j in range(wu):
        for i in range(c):
            m += 1
            MINLP_model.add_component(f'eq5_{m}', Constraint(expr=(
                getattr(MINLP_model, f'M_in_{i+1}{j+1}') -
                getattr(MINLP_model, f'M_out_{i+1}{j+1}') +
                L_ml_cu[m-1] -
                (F_loss_u[j]) * (C_loss_cu[j])
            ) == 0))
```

```
#Equation 6
m = 0
for j in range(tu):
    for i in range(c):
        m += 1
        MINLP_model.add_component(f'eq6_{m}', Constraint(expr=(
            (1 - RR_cu.iloc[j][i]) *
            getattr(MINLP_model, f'M_in_{i+1}{j+wu+1}') -
            (F_loss_u[j+wu] * C_loss_cu[j+wu])
        ) == getattr(MINLP_model, f'M_out_{i+1}{j+wu+1}')))

## DEFINE MINLP_MODEL CONSTRAINTS
#(B) Availability and Capacity Constraints

#(i) Constraints of water flow rate

#Constraint 7 & Constraint 8
for i in range(u):
    MINLP_model.add_component(f'c7_{i+1}', Constraint(expr=(
        F_tmin_u[i]
    ) <= getattr(MINLP_model, f'F_t_{i+1}')))
    MINLP_model.add_component(f'c8_{i+1}', Constraint(expr=(
        F_tmax_u[i]
    ) >= getattr(MINLP_model, f'F_t_{i+1}')))

#Equation 8
#for water using operations
for i in range(wu):
    MINLP_model.add_component(f'eq8_{i+1}', Constraint(expr=(
        sum(getattr(MINLP_model, f'F_w_{j+1}{i+1}') for j in range(s)) +
        sum(getattr(MINLP_model, f'F_ua_{k+1}{i+1}') for k in range(ua)) -
        F_loss_u[i]
    ) == getattr(MINLP_model, f'F_t_{i+1}')))

#for treatment units
for i in range(tu):
    MINLP_model.add_component(f'eq8_{i+wu+1}', Constraint(expr=(
        sum(getattr(MINLP_model, f'F_ua_{k+1}{i+wu+1}') for k in range(ua)) -
        F_loss_u[wu+i]
    ) == getattr(MINLP_model, f'F_t_{i+wu+1}')))

#(ii) Constraints on the quality and quantity of water
# Constraint 9
m = 0
for i in range(wu):
    for j in range(c):
        m += 1
        MINLP_model.add_component(f'c9_{m}', Constraint(expr=(
            getattr(MINLP_model, f'M_in_{j+1}{i+1}')
        ) <= C_inmax_cu[m-1] * getattr(MINLP_model, f'F_t_{i+1}')))

#(iii) Constraint on the environmental discharge limit of contaminants
if include_environmental_limit_switch == 1:
    # Constraint 10
    for i in range(c):
        MINLP_model.add_component(f'c10_{i+1}', Constraint(expr=(
            sum(getattr(
                MINLP_model, f'C_out_{i+1}{j+1}') *
                getattr(
                MINLP_model, f'F_out_{j+1}{k+1}') for j in range(u) for k in range(e))
```

```
        ) <= C_env_c[i] * sum(getattr(
            MINLP_model, f'F_out_{j+1}{k+1}') for j in range(u) for k in range(e))))

#(C) Logic Constraints

#(i) Upper and lower bounds on the flow rates
# Constraint 11 & 12
m = 0
for i in range(wu):
    for j in range(s):
        m += 1
        MINLP_model.add_component(f'c11_{m}', Constraint(expr=(
            getattr(MINLP_model, f'F_w_{j+1}{i+1}') -
            U_fw[j][i] * getattr(MINLP_model, f'B_fw_{j+1}{i+1}')
        ) <= 0))
        MINLP_model.add_component(f'c12_{m}', Constraint(expr=(
            getattr(MINLP_model, f'F_w_{j+1}{i+1}') -
            L_fw[j][i] * getattr(MINLP_model, f'B_fw_{j+1}{i+1}')
        ) >= 0))

# Constraint 13 & 14
m = 0
for i in range(e):
    for j in range(u):
        m += 1
        MINLP_model.add_component(f'c13_{m}', Constraint(expr=(
            getattr(MINLP_model, f'F_out_{j+1}{i+1}') -
            U_out[j][i] * getattr(MINLP_model, f'B_out_{j+1}{i+1}')
        ) <= 0))
        MINLP_model.add_component(f'c14_{m}', Constraint(expr=(
            getattr(MINLP_model, f'F_out_{j+1}{i+1}') -
            L_out[j][i] * getattr(MINLP_model, f'B_out_{j+1}{i+1}')
        ) >= 0))

# Constraint 15 & 16
m = 0
for i in range(ua):
    for j in range(u):
        m += 1
        MINLP_model.add_component(f'c15_{m}', Constraint(expr=(
            getattr(MINLP_model, f'F_ua_{j+1}{i+1}') -
            U_ua[j][i] * getattr(MINLP_model, f'B_ua_{j+1}{i+1}')
        ) <= 0))
        MINLP_model.add_component(f'c16_{m}', Constraint(expr=(
            getattr(MINLP_model, f'F_ua_{j+1}{i+1}') -
            L_ua[j][i] * getattr(MINLP_model, f'B_ua_{j+1}{i+1}')
        ) >= 0))

# (ii) Maximum number of sources to feed each operation
#Constraint 17
#for water-using operations
for i in range(wu):
    MINLP_model.add_component(f'c17_{i+1}', Constraint(expr=(
        sum(getattr(MINLP_model, f'B_ua_{j+1}{i+1}') for j in range(ua)) +
        sum(getattr(MINLP_model, f'B_fw_{k+1}{i+1}') for k in range(s))
    ) <= NS_max_u[i]))

#for treatment units
for i in range(tu):
    MINLP_model.add_component(f'c17_{i+wu+1}', Constraint(expr=(
        sum(getattr(MINLP_model, f'B_ua_{j+1}{i+wu+1}') for j in range(ua))
```

```
    ) <= NS_max_u[i+wu]))

#Constraint 17a --> self added, maximum number of sources to leave each operation
for i in range(u):
    MINLP_model.add_component(f'c17a_{i+1}', Constraint(expr=(
        sum(getattr(MINLP_model, f'B_ua_{i+1}{j+1}') for j in range(ua)) +
        sum(getattr(MINLP_model, f'B_out_{i+1}{k+1}') for k in range(e))
    ) <= NS_max_u[i]))

# (iii) Elimination of regeneration recycling
if regeneration_recycling_allowed_switch == 0:
    # Equation 18, Equation 19 & Equation 20
    m = 0
    for i in range(u):
        for j in range(tu):
            m += 1
            MINLP_model.add_component(f'c18_{m}', Constraint(expr=(
                getattr(MINLP_model, f'B_ua_{i+1}{j+wu+1}') -
                getattr(MINLP_model, f'B_G1_{i+1}{j+wu+1}')
            ) <= 0))
            MINLP_model.add_component(f'c19_{m}', Constraint(expr=(
                getattr(MINLP_model, f'B_ua_{j+wu+1}{i+1}') -
                getattr(MINLP_model, f'B_G2_{i+1}{j+wu+1}')
            ) <= 0))
            # MINLP_model.add_component(f'c20_{m}', Constraint(
            #     expr=(getattr(MILP_model, f'B_G2_{i+1}{j+wu+1}'))
            # ) ==  (1 - getattr(MILP_model, f'B_G1_{i+1}{j+wu+1}')))
            # Equation 20, In case G1 and G2 can also both be 0:
            MINLP_model.add_component(f'c20_{m}', Constraint(expr=(
                getattr(MINLP_model, f'B_G2_{i+1}{j+wu+1}') +
                getattr(MINLP_model, f'B_G1_{i+1}{j+wu+1}')
            ) <= 1))

    # Equation 21
    m = 0
    for i in range(u):
        for j in range(tu):
            for k in range(ua):
                m += 1
                MINLP_model.add_component(f'c21_{m}', Constraint(expr=(
                    2 - (getattr(MINLP_model, f'B_G2_{i+1}{j+wu+1}') +
                         getattr(MINLP_model, f'B_G1_{k+1}{j+wu+1}'))
                ) >= getattr(MINLP_model, f'B_ua_{i+1}{k+1}')))

    # Equation 22
    for i in range(tu):
        MINLP_model.add_component(f'c22_{i+1}', Constraint(expr=(
            sum(getattr(MINLP_model, f'B_G1_{j+1}{i+wu+1}') for j in range(u)) +
            sum(getattr(MINLP_model, f'B_G2_{j+1}{i+wu+1}') for j in range(u))
        ) == N__OP - 1))

#(iv) Elimination of direct recycling
# Equation 23
m = 0
for i in range(u):
    for j in range(ua):
        m += 1
        MINLP_model.add_component(f'c23_{m}', Constraint(expr=(
            getattr(MINLP_model, f'B_ua_{j+1}{i+1}') +
            getattr(MINLP_model, f'B_ua_{i+1}{j+1}')
        ) <= 1))
```

```
    #(D) Objective Function

    #(i) Freshwater supply cost
    #Equation 24
    for i in range(s):
        MINLP_model.add_component(f'eq24_{i}', Constraint(expr=(
            CO_fw_s[i] * Operation_time *
            sum(getattr(MINLP_model, f'F_w_{i+1}{j+1}') for j in range(wu))
        ) == getattr(MINLP_model, f'Cost_fw_{i+1}')))

    #(ii) Water and wastewater treatment cost  # Specific for each treatment system!
    for i in range(tu):
        MINLP_model.add_component(f'eq25_{i}', Constraint(expr=(
            CAPEX_value[i] * Annualization_factor *
            (getattr(MINLP_model, f'F_t_{i+wu+1}') ** CAPEX_power[i]) +
            OPEX_value[i] * Operation_time * getattr(MINLP_model, f'F_t_{i+wu+1}')
        ) == getattr(MINLP_model, f'Cost_tu_{i+wu+1}')))

    #(iii) Piping cost
    if pipe_connection_costs_switch == 1:
        # Equation 26
#         m = 0
#         for i in range(s):
#             for j in range(wu):
#                 m += 1
#                 MINLP_model.add_component(f'eq26_{m}', Constraint(expr=(
#                     getattr(MINLP_model, f'F_w_{i+1}{j+1}')
#                 ) == getattr(MINLP_model, f'A_fw_{i+1}{j+1}') * u_fw_su))

#         for i in range(u):
#             for j in range(ua):
#                 m += 1
#                 MINLP_model.add_component(f'eq26_{m}', Constraint(expr=(
#                     getattr(MINLP_model, f'F_ua_{i+1}{j+1}')
#                 ) == getattr(MINLP_model, f'A_ua_{i+1}{j+1}') * u_ua_uua))

#         for i in range(u):
#             for j in range(e):
#                 m += 1
#                 MINLP_model.add_component(f'eq26_{m}', Constraint(expr=(
#                     getattr(MINLP_model, f'F_out_{i+1}{j+1}')
#                 ) == getattr(MINLP_model, f'A_out_{i+1}{j+1}') * u_out_ue))

#         #Equation 26
#         m = 0
#         for i in range(s):
#             for j in range(wu):
#                 m += 1
#                 MINLP_model.add_component(f'eq26_{m}', Constraint(expr=(
#                     getattr(MINLP_model, f'A_fw_{i+1}{j+1}')
#                 ) == u_fw_su / getattr(MINLP_model, f'F_w_{i+1}{j+1}')))

#         for i in range(u):
#             for j in range(ua):
#                 m += 1
#                 MINLP_model.add_component(f'eq26_{m}', Constraint(expr=(
#                     getattr(MINLP_model, f'A_ua_{i+1}{j+1}')
#                 ) == u_ua_uua / getattr(MINLP_model, f'F_ua_{i+1}{j+1}')))

#         for i in range(u):
```

```
#             for j in range(e):
#                 m += 1
#                 MINLP_model.add_component(f'eq26_{m}', Constraint(expr=(
#                     getattr(MINLP_model, f'A_out_{i+1}{j+1}')
#                 ) == u_out_ue / getattr(MINLP_model, f'F_out_{i+1}{j+1}')))

        # Equation 27
        m = 0
        for i in range(s):
            for j in range(wu):
                m += 1
                MINLP_model.add_component(f'eq27_{m}', Constraint(expr=(
                    getattr(MINLP_model, f'Cost_fwpipe_{i+1}{j+1}')
                ) == (d_fw_su.iloc[i][j] * Annualization_factor *
                      ((a_fw_su * getattr(MINLP_model, f'A_fw_{i+1}{j+1}')) +
                       (b_fw_su * getattr(MINLP_model, f'B_fw_{i+1}{j+1}'))))))

        for i in range(u):
            for j in range(ua):
                m += 1
                MINLP_model.add_component(f'eq27_{m}', Constraint(expr=(
                    getattr(MINLP_model, f'Cost_uapipe_{i+1}{j+1}')
                ) == (d_ua_uua.iloc[i+s][j] * Annualization_factor *
                      ((a_ua_uua * getattr(MINLP_model, f'A_ua_{i+1}{j+1}')) +
                       (b_ua_uua * getattr(MINLP_model, f'B_ua_{i+1}{j+1}'))))))

        for i in range(u):
            for j in range(e):
                m += 1
                MINLP_model.add_component(f'eq27_{m}', Constraint(expr=(
                    getattr(MINLP_model, f'Cost_outpipe_{i+1}{j+1}')
                ) == (d_out_ue.iloc[i+s] * Annualization_factor *
                      ((a_out_ue * getattr(MINLP_model, f'A_out_{i+1}{j+1}')) +
                       (b_out_ue * getattr(MINLP_model, f'B_out_{i+1}{j+1}'))))))

    #(iv) Overall objective function

    #Total costs per category
    MINLP_model.eq28_1 = Constraint(expr=(
        sum(getattr(MINLP_model, f'Cost_fw_{i+1}') for i in range(s))
    ) == MINLP_model.Cost_fw_s_sum)

    if pipe_connection_costs_switch == 1:
        MINLP_model.eq28_2 = Constraint(expr=(
            MINLP_model.Cost_fwpipe_su_sum -
            sum(getattr(
                MINLP_model, f'Cost_fwpipe_{i+1}{j+1}') for i in range(s) for j in range(wu))
        ) == 0)
        MINLP_model.eq28_3 = Constraint(expr=(
            MINLP_model.Cost_uapipe_ua_sum -
            sum(getattr(
                MINLP_model, f'Cost_uapipe_{i+1}{j+1}') for i in range(u) for j in range(ua))
        ) == 0)
        MINLP_model.eq28_4 = Constraint(expr=(
            MINLP_model.Cost_outpipe_ue_sum -
            sum(getattr(
                MINLP_model, f'Cost_outpipe_{i+1}{j+1}') for i in range(u) for j in range(e))
        ) == 0)

    MINLP_model.eq28_5 = Constraint(expr=(
        sum(getattr(MINLP_model, f'Cost_tu_{i+wu+1}') for i in range(tu))
```

```
    ) == MINLP_model.Cost_tu_u_sum)

    if pipe_connection_costs_switch == 1:
        MINLP_model.eq28_tot = Constraint(expr=(
            MINLP_model.O_cost - (MINLP_model.Cost_fw_s_sum +
                                  MINLP_model.Cost_fwpipe_su_sum +
                                  MINLP_model.Cost_uapipe_ua_sum +
                                  MINLP_model.Cost_outpipe_ue_sum +
                                  MINLP_model.Cost_tu_u_sum)
        ) == 0)
    else:
        MINLP_model.eq28_tot = Constraint(expr=(
            MINLP_model.Cost_fw_s_sum +
            MINLP_model.Cost_tu_u_sum
        ) == MINLP_model.O_cost)

    ## MINLP_MODEL SOLVER
    MINLP_model.obj = Objective(expr=MINLP_model.O_cost, sense=minimize)
    SolverFactory(modules.find(MINLP_solver),
                  executable=MINLP_solver_location).solve(MINLP_model).write()

    ## Print MINLP results
    if MINLP_print_results_switch == 1:
        #F_w_su
        for i in range(s):
            for j in range(wu):
                print(f'F_w_{i+1}{j+1}=',
                      value(getattr(MINLP_model, f'F_w_{i+1}{j+1}')))

        #F_out_ue
        for i in range(u):
            for j in range(e):
                print(f'F_out_{i+1}{j+1}=',
                      value(getattr(MINLP_model, f'F_out_{i+1}{j+1}')))

        #F_ua_u,ua = F_ua_ua,u
        for i in range(u):
            for j in range(ua):
                print(f'F_ua_{i+1}{j+1}=',
                      value(getattr(MINLP_model, f'F_ua_{i+1}{j+1}')))

        #F_t_u
        for i in range(u):
            print(f'F_t_{i+1}=',
                  value(getattr(MINLP_model, f'F_t_{i+1}')))

        #M_in_cu
        for j in range(u):
            for i in range(c):
                print(f'M_in_{i+1}{j+1}=',
                      value(getattr(MINLP_model, f'M_in_{i+1}{j+1}')))

        #M_out_cu
        for j in range(u):
            for i in range(c):
                print(f'M_out_{i+1}{j+1}=',
                      value(getattr(MINLP_model, f'M_out_{i+1}{j+1}')))

        #C_out_cu
        for j in range(u):
            for i in range(c):
```

```
                print(f'C_out_{i+1}{j+1} = ',
                      value(getattr(MINLP_model, f'C_out_{i+1}{j+1}')))

        #B_w_su
        for i in range(s):
            for j in range(wu):
                print(f'B_fw_{i+1}{j+1}=',
                      value(getattr(MINLP_model, f'B_fw_{i+1}{j+1}')))

        #B_ua_u,ua = B_ua_ua,u
        for i in range(u):
            for j in range(ua):
                print(f'B_ua_{i+1}{j+1}=',
                      value(getattr(MINLP_model, f'B_ua_{i+1}{j+1}')))

        #B_out_ue
        for i in range(u):
            for j in range(e):
                print(f'B_out_{i+1}{j+1}=',
                      value(getattr(MINLP_model, f'B_out_{i+1}{j+1}')))

        if regeneration_recycling_allowed_switch == 0:
            #B_G1_u,tu = B_G1_tu,u
            for i in range(u):
                for j in range(tu):
                    print(f'B_G1_{i+1}{j+wu+1}=',
                          value(getattr(MINLP_model, f'B_G1_{i+1}{j+wu+1}')))

            #B_G2_u,tu = B_G2_tu,u
            for i in range(u):
                for j in range(tu):
                    print(f'B_G2_{i+1}{j+wu+1}=',
                          value(getattr(MINLP_model, f'B_G2_{i+1}{j+wu+1}')))

        #Cost_fw_s
        for i in range(s):
                print(f'Cost_fw_{i+1}=',
                      value(getattr(MINLP_model, f'Cost_fw_{i+1}')))

        #Cost_tu
        for i in range(tu):
                print(f'Cost_tu_{i+wu+1}=',
                      value(getattr(MINLP_model, f'Cost_tu_{i+wu+1}')))

        print('Cost_fw_s_sum=', MINLP_model.Cost_fw_s_sum.value)
        if pipe_connection_costs_switch == 1:
            print('Cost_fwpipe_su_sum=', MINLP_model.Cost_fwpipe_su_sum.value)
            print('Cost_uapipe_ua_sum=', MINLP_model.Cost_uapipe_ua_sum.value)
            print('Cost_outpipe_ue_sum=', MINLP_model.Cost_outpipe_ue_sum.value)
        print('Cost_tu_u_sum=', MINLP_model.Cost_tu_u_sum.value)

        #Final costs
        print('O_cost.value=', MINLP_model.O_cost.value)

    return MINLP_model.O_cost.value

## INITIATION
if initation_activated_switch == 1:
    MILP_initiation = []
    m = 0
    for i in range(wu):
```

```
        for j in range(c):
            m += 1
            MILP_initiation.append(C_outmax_cu[m-1])
    for i in range(tu):
        for j in range(c):
            MILP_initiation.append(0)
            # MILP_initiation.append(C_env_c[j])
    MILP_result = MILP_function(MILP_initiation)
    LP_result = LP_function(MILP_result[0])
    print("============================================================")
    print(f'Initiation:')
    print(f'MILP_M_terms_result_initiation: {M_terms_results_list[0]}')
    print(f'MILP_Cost_result_initiation: {Cost_results_list[0]}')
    print(f'MILP_result_initiation: {MILP_result}')
    print(f'LP_M_terms_result_initiation: {M_terms_results_list[1]}')
    print(f'LP_Cost_result_initiation: {Cost_results_list[1]}')
    print(f'LP_result_initiation: {LP_result}')
    print("============================================================")

## ITERATION
if iteration_activated_switch == 1:
    l = 2
    for k in range(Maximum_amount_of_iterations):
        print("============================================================")
        print(f'Iteration_{k+1}:')
        MILP_result = MILP_function(LP_result)
        print(f'MILP_M_terms_result_iteration_{k+1}: {M_terms_results_list[l]}')
        print(f'MILP_Cost_result_iteration_{k+1}: {Cost_results_list[l]}')
        print(f'MILP_result_iteration_{k+1}:{MILP_result}')
        l += 1
        LP_result = LP_function(MILP_result[0])
        print(f'LP_M_terms_result_iteration_{k+1}: {M_terms_results_list[l]}')
        print(f'LP_Cost_result_iteration_{k+1}: {Cost_results_list[l]}')
        print(f'LP_result_iteration_{k+1}:{LP_result}')
        # print(f'LP_result_iteration_{k+1}:{LP_result}')
        l += 1
        print("============================================================")
        if M_terms_results_list[l-1] <= convergence_criterion:
            break

## FINAL MINLP
if final_MINLP_switch == 1:
    print(f'MINLP function result:')
    MINLP_function(MILP_result[0], MILP_result[1], LP_result, MILP_result[2])

## WATER SAVINGS
#Freshwater use
F_w_total = 0
for i in range(s):
    for j in range(wu):
        F_w_total += value(getattr(MINLP_model, f'F_w_{i+1}{j+1}'))

#Freshwater saving
print(f'Freshwater use compared to linear system:',
      (F_w_total - sum(F_tmin_u)) / sum(F_tmin_u) * 100,'%')

#Wastewater production
F_out_total = 0
for i in range(u):
    for j in range(e):
        F_out_total += value(getattr(MINLP_model, f'F_out_{i+1}{j+1}'))
```

```
#Wastewater saving
print(f'Wastewater produced compared to linear system:',
      (F_out_total - sum(F_tmin_u)) / sum(F_tmin_u) * 100,'%')

## MINLP RESULTS VISUALISATION

if create_MINLP_visualization_switch == 1:
    #Store results in lists
    Edge_dictionary = {}

    #FW to O
    FW_O_edges = []
    m = 0
    for i in range(s):
        for j in range(wu):
            if value(getattr(MINLP_model, f'F_w_{i+1}{j+1}')) > 0.1:
                FW_O_edges.append((i+1,j+s+1))
                Edge_dictionary[FW_O_edges[m]] = round(
                    value(getattr(MINLP_model, f'F_w_{i+1}{j+1}')), 1)
                m += 1

    # O to O
    O_O_edges = []
    m = 0
    for i in range(wu):
        for j in range(wu):
            if value(getattr(MINLP_model, f'F_ua_{i+1}{j+1}')) > 0.1:
                O_O_edges.append((i+s+1, j+s+1))
                Edge_dictionary[O_O_edges[m]] = round(
                    value(getattr(MINLP_model, f'F_ua_{i+1}{j+1}')), 1)
                m += 1

    # T to T
    T_T_edges = []
    m = 0
    for i in range(tu):
        for j in range(tu):
            if value(getattr(MINLP_model, f'F_ua_{i+wu+1}{j+wu+1}')) > 0.1:
                T_T_edges.append((i+s+wu+1, j+s+wu+1))
                Edge_dictionary[T_T_edges[m]] = round(
                    value(getattr(MINLP_model, f'F_ua_{i+wu+1}{j+wu+1}')), 1)
                m += 1

    # O to T
    O_T_edges = []
    m = 0
    for i in range(wu):
        for j in range(tu):
            if value(getattr(MINLP_model, f'F_ua_{i+1}{j+wu+1}')) > 0.1:
                O_T_edges.append((i+s+1, j+s+wu+1))
                Edge_dictionary[O_T_edges[m]] = round(
                    value(getattr(MINLP_model, f'F_ua_{i+1}{j+wu+1}')), 1)
                m += 1

    # O to DP
    O_DP_edges = []
    m = 0
    for i in range(wu):
        for j in range(e):
            if value(getattr(MINLP_model, f'F_out_{i+1}{j+1}')) > 0.1:
```

```
            O_DP_edges.append((i+s+1, j+s+u+1))
            Edge_dictionary[O_DP_edges[m]] = round(
                value(getattr(MINLP_model, f'F_out_{i+1}{j+1}')), 1)
            m += 1

# T to O
T_O_edges = []
m = 0
for i in range(tu):
    for j in range(wu):
        if value(getattr(MINLP_model, f'F_ua_{i+wu+1}{j+1}')) > 0.1:
            T_O_edges.append((i+s+wu+1, j+s+1))
            Edge_dictionary[T_O_edges[m]] = round(
                value(getattr(MINLP_model, f'F_ua_{i+wu+1}{j+1}')), 1)
            m += 1

# T to DP
T_DP_edges = []
m = 0
for i in range(tu):
    for j in range(e):
        if value(getattr(MINLP_model, f'F_out_{i+wu+1}{j+1}')) > 0.1:
            T_DP_edges.append((i+s+wu+1, j+s+u+1))
            Edge_dictionary[T_DP_edges[m]] = round(
                value(getattr(MINLP_model, f'F_out_{i+wu+1}{j+1}')), 1)
            m += 1

# Create an empty DiGraph
G = nx.DiGraph()

# Create nodelist for plot
Node_dictionary = {}

Node_names_dictionary = {}
node_count = 0

# Create nodes
m = 0
key = 0
for i in range(s):
    node_count = node_count + 1
    Node_names_dictionary[node_count] = f'FW{i+1}'
    key = i + 1
    pos = (-5, m)
    Node_dictionary[key] = pos
    m = m + 3

m = 0
key = 0
for i in range(wu):
    node_count = node_count + 1
    Node_names_dictionary[node_count] = f'O{i+1}'
    key = s + i + 1
    pos = (0, m)
    Node_dictionary[key] = pos
    m = m + 3

m = wu * 3
key = 0
for i in range(tu):
    node_count = node_count + 1
```

```
        Node_names_dictionary[node_count] = f'T{i+1}'
        key = s + i + wu + 1
        pos = (0, m)
        Node_dictionary[key] = pos
        m = m + 3

    m = 0
    key = 0
    for i in range(e):
        node_count = node_count + 1
        Node_names_dictionary[node_count] = f'DP{i+1}'
        key = s + u + i + 1
        pos = (5, m)
        Node_dictionary[key] = pos
        m = m + 3

    # Create edges
    for i in range(len(FW_O_edges)):
        G.add_edge(FW_O_edges[i][0], FW_O_edges[i][1])

    for i in range(len(O_O_edges)):
        G.add_edge(O_O_edges[i][0], O_O_edges[i][1])

    for i in range(len(T_T_edges)):
        G.add_edge(T_T_edges[i][0], T_T_edges[i][1])

    for i in range(len(O_T_edges)):
        G.add_edge(O_T_edges[i][0], O_T_edges[i][1])

    for i in range(len(O_DP_edges)):
        G.add_edge(O_DP_edges[i][0], O_DP_edges[i][1])

    for i in range(len(T_O_edges)):
        G.add_edge(T_O_edges[i][0], T_O_edges[i][1])

    for i in range(len(T_DP_edges)):
        G.add_edge(T_DP_edges[i][0], T_DP_edges[i][1])


    # Draw figure
    plt.figure(figsize=(15,10))
    nx.draw_networkx(G,
                     pos=Node_dictionary,
                     with_labels=False,
                     node_size=500,
                     alpha=0.5,
                     connectionstyle="arc3, rad=-0.5")
    nx.draw_networkx_labels(G,
                            pos=Node_dictionary,
                            labels=Node_names_dictionary,
                            font_color="white",
                            font_size=9)
    my_draw_networkx_edge_labels(G,
                                 pos=Node_dictionary,
                                 edge_labels=Edge_dictionary,
                                 label_pos=0.5,
                                 rotate=False,
                                 rad=-0.5,
                                 font_size=7,
                                 bbox=dict(boxstyle='round',
                                           ec=(0.5, 0.5, 0.5),
```

```
                                        fc=(1.0, 1.0, 1.0)))
    plt.show()
```

G

Python Results

G.1. MINLP Example Output

```
# ==========================================================
# = Solver Results                                         =
# ==========================================================
# ----------------------------------------------------------
#   Problem Information
# ----------------------------------------------------------
Problem:
- Name: unknown
  Lower bound: 2.43844718719117
  Upper bound: 2.4384471872707754
  Number of objectives: 1
  Number of constraints: 2
  Number of variables: 2
  Number of binary variables: 1
  Number of integer variables: 0
  Number of continuous variables: 1
  Number of nonzeros: None
  Sense: minimize
  Number of disjunctions: 0
# ----------------------------------------------------------
#   Solver Information
# ----------------------------------------------------------
Solver:
- Name: MindtPyOA
  Status: ok
  Message: None
  User time: 0.5563923000008799
  Wallclock time: 0.5563923000008799
  Termination condition: optimal
  Termination message: None
  Timing: Call after main solve: 3.300000389572233e-05
Call after subproblem solve: 2.4500011932104826e-05
OA cut generation: 0.0023806999961379915
feasibility subproblem: 0.06961859999864828
fixed subproblem: 0.1395637999958126
initialization: 0.07902530000137631
main loop: 0.4606383999926038
main: 0.2430266000010306
main_timer_start_time: 106845.8335593
total: 0.5563923000008799
```

```
  Iterations: 3
  Num infeasible nlp subproblem: 1
  Best solution found time: 0.48048350001045037
  Primal integral: 0.0
  Dual integral: 0.5773586097562803
  Primal dual gap integral: 0.5773586097562803
```

G.2. Petroleum Refinery Case Study, Scenario 1

In this run, the initial value of $C_{c,tu}^{out}$ was set to 0.

```
============================================================
Initiation:
MILP_M_terms_result_initiation: 2849150.65
MILP_Cost_result_initiation: 0.0
MILP_result_initiation: ([0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, -0.0, 25.00000000000001, 0.0, -0.0, 0.0, 0.0, 0.0, 24.99999999999999,
-0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 34.00000000000001, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0,
0.0, 56.0, 0.0, -0.0, 0.0, 0.0, -0.0, 40.00000000000001, 0.0, 0.0, 0.0, -0.0, 0.0,
0.0, -0.0, 0.0, 0.0, 0.0, 40.00000000000001, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
50.0, 0.0, 0.0, 40.00000000000001, 0.0, 0.0, 0.0, 0.0, 0.0, 9.0, 56.0, 0.0, 0.0,
0.0, 0.0, 0.0], [0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0], [50.0,
34.00000000000001, 56.0, 40.00000000000001, 40.00000000000001, 0.0,
90.00000000000001, 65.0])
LP_M_terms_result_initiation: 124820.9750000001
LP_Cost_result_initiation: nope
LP_result_initiation: [15.0, 400.0, 35.0, 111.4315610859729, 12500.0,
165.1233031674208, 101.5192307692308, 45.0, 9316.576923076922, 20.0, 0.0, 20.0,
40.0, 0.0, 32.0, -0.0, -0.0, -0.0, 31.57916666666667, 475.0222222222234,
117.1872222222222, 1.519230769230769, 153.8461538461539, 16.57692307692308]
============================================================
============================================================
Iteration_1:
MILP_M_terms_result_iteration_1: 2250017.821990776
MILP_Cost_result_iteration_1: 0.0
MILP_result_iteration_1:([0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 50.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0,
35.82142857142799, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 59.0, 0.0, -0.0, 0.0, 0.0,
-0.0, 9.0, 0.0, 0.0, 0.0, -0.0, 0.0, 59.0, -0.0, 0.0, 0.0, 0.0, 0.0, 50.0,
35.82142857142799, 0.0, 9.0, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0,
94.82142857142799, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0], [0.0, 0.0,
0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
1.0, 0.0, 0.0, 1.0, 0.0, 0.0], [50.0, 35.82142857142799, 59.0, 9.0, 59.0,
94.82142857142799, 94.82142857142799, 0.0])
LP_M_terms_result_iteration_1: 91336.275959092
LP_Cost_result_iteration_1: nope
LP_result_iteration_1:[15.0, 400.0, 35.0, 120.0, 11580.10128744242, 180.0,
124.2372881355932, 45.0, 8867.966101694916, 20.0, 0.0, 20.0, 29.32203389830508,
1369.491525423729, 40.84745762711864, 36.79096045197742, 0.4402704930811538,
111.7169114877596, 36.79096045197741, 440.2704930811538, 111.7169114877596, -0.0,
-0.0, -0.0]
============================================================
============================================================
Iteration_2:
MILP_M_terms_result_iteration_2: 4478545.021024181
MILP_Cost_result_iteration_2: 0.0
MILP_result_iteration_2:([0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, -0.0, 82.18857397355289, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0,
-0.0, 0.0, 0.0, 9.608885200559358, 100.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0,
```

```
15.82639915386273, 40.17360084613727, -0.0, 10.98502687258438, 47.0, -0.0,
9.82639915386273, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 0.0,
9.82639915386273, -0.0, 16.43528435442208, 9.0, -0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0,
0.0, -0.0, 0.0, 25.43528435442208, 0.0, 0.0, 82.18857397355289, 0.0, 0.0,
67.81142602644711, 0.0, 0.0, 0.0, 0.0], [0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0,
0.0], [82.18857397355289, 109.6088852005594, 56.0, 67.81142602644711,
9.82639915386273, 25.43528435442208, 25.43528435442209, 150.0])
LP_M_terms_result_iteration_2: 647387.4837986166
LP_Cost_result_iteration_2: nope
LP_result_iteration_2:[12.35508248756179, 400.0, 35.0, 44.64688608035609,
4090.335523827174, 87.64406255058725, 108.7671710661538, 45.0, 9335.563265750325,
5.589210819393833, 60.00000000001474, 20.00000000001475, 87.0025533341361,
6247.414031120426, 68.84800550886011, 25.36318568812248, 0.1573237864556917,
116.8380980019426, 25.36318568812247, 157.3237864556917, 116.8380980019425,
3.229726446692418, 3148.206322466837, 1281.613755167505]
============================================================
============================================================
Iteration_3:
MILP_M_terms_result_iteration_3: 2104044.218893624
MILP_Cost_result_iteration_3: 102274.1502828029
MILP_result_iteration_3:([50.0, 0.0, -0.0, 9.461715280699336, -0.0, 0.0,
22.50615972514411, 0.0, -0.0, 0.0, 9.0, 0.0, 27.95555555555523, -0.0, 41.0, 0.0,
-0.0, 9.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 27.95555555555524, 9.0,
-0.0, 0.0, 0.0, -0.0, 0.0, -0.0, 46.04444444444468, 9.955555555555247, -0.0,
9.461715280699336, 0.0, -0.0, 0.0, -0.0, -0.0, -0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0,
0.0, 9.0, -0.0, 9.0, 55.99999999999993, -0.0, 2.578347878195721e-18, 0.0, 0.0, 0.0,
-0.0, 0.0, 0.0, -0.0, 0.0, 73.99999999999993, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0,
0.0, 1.953992523340276e-14, 0.0], [0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0], [50.0,
59.46171528069934, 55.99999999999993, 9.461715280699336, 9.0, 73.99999999999993,
73.99999999999994, 27.95555555555525])
LP_M_terms_result_iteration_3: -6.166374078020453e-10
LP_Cost_result_iteration_3: nope
LP_result_iteration_3:[15.0, 400.0, 35.0, 75.2746394525629, 7259.839048737683,
121.8955555968902, 133.4399087761769, 25.27583330480239, 9418.124313431661,
16.91025308343183, 50.73075925029549, 16.91025308343183, 103.8888888888889,
7155.555555555555, 88.33333333333333, 33.43990877617689, 0.2758333048023859,
118.12431343166, 33.43990877617688, 275.833304802386, 118.12431343166,
5.260028144277219, 4649.887430889807, 1710.835609075533]
============================================================
MINLP function result:
# ============================================================
# = Solver Results                                           =
# ============================================================
# ------------------------------------------------------------
#   Problem Information
# ------------------------------------------------------------
Problem:
- Lower bound: -inf
  Upper bound: inf
  Number of objectives: 1
  Number of constraints: 389
  Number of variables: 241
  Sense: unknown
# ------------------------------------------------------------
#   Solver Information
# ------------------------------------------------------------
Solver:
- Status: ok
  Message: SCIP 8.0.3\x3a optimal solution; objective 591926.224426302; 542205
```

```
  simplex iterations; 7646 branching nodes
  Termination condition: optimal
  Id: 0
  Error rc: 0
  Time: 49.29707741737366
# ----------------------------------------------------------
#   Solution Information
# ----------------------------------------------------------
Solution:
- number of solutions: 0
  number of solutions displayed: 0
F_w_11= 50.0
F_w_12= 0.0
F_w_13= 0.0
F_w_14= 9.699999997951979
F_w_15= 0.0
F_out_11= 0.0
F_out_21= 0.0
F_out_31= 0.0
F_out_41= 0.0
F_out_51= 0.0
F_out_61= 0.0
F_out_71= 59.69999999795198
F_out_81= 0.0
F_ua_11= 0.0
F_ua_12= 24.30000000204802
F_ua_13= 0.0
F_ua_14= 0.0
F_ua_15= 12.80605605396715
F_ua_16= 0.0
F_ua_17= 12.89394394398483
F_ua_18= 0.0
F_ua_21= 0.0
F_ua_22= 0.0
F_ua_23= 0.0
F_ua_24= 0.0
F_ua_25= 0.0
F_ua_26= 33.99999999999999
F_ua_27= 0.0
F_ua_28= 0.0
F_ua_31= 0.0
F_ua_32= 0.0
F_ua_33= 0.0
F_ua_34= 0.0
F_ua_35= 0.0
F_ua_36= 0.0
F_ua_37= 0.0
F_ua_38= 90.30000000204802
F_ua_41= 0.0
F_ua_42= 9.69999999795175
F_ua_43= 0.0
F_ua_44= 0.0
F_ua_45= 0.0
F_ua_46= 0.0
F_ua_47= 0.0
F_ua_48= 0.0
F_ua_51= 0.0
F_ua_52= 0.0
F_ua_53= 0.0
F_ua_54= 0.0
F_ua_55= 0.0
```

F_ua_56= 12.80605605396715
F_ua_57= 0.0
F_ua_58= 0.0
F_ua_61= 0.0
F_ua_62= 0.0
F_ua_63= 0.0
F_ua_64= 0.0
F_ua_65= 0.0
F_ua_66= 0.0
F_ua_67= 46.80605605396715
F_ua_68= 0.0
F_ua_71= 0.0
F_ua_72= 0.0
F_ua_73= 90.30000000204802
F_ua_74= 0.0
F_ua_75= 0.0
F_ua_76= 0.0
F_ua_77= 0.0
F_ua_78= 0.0
F_ua_81= 0.0
F_ua_82= 0.0
F_ua_83= 0.0
F_ua_84= 0.0
F_ua_85= 0.0
F_ua_86= 0.0
F_ua_87= 90.30000000204802
F_ua_88= 0.0
F_t_1= 50.0
F_t_2= 33.99999999999955
F_t_3= 90.30000000204802
F_t_4= 9.699999997951979
F_t_5= 12.80605605396715
F_t_6= 46.80605605396714
F_t_7= 150.0
F_t_8= 90.30000000204802
M_in_11= 0.0
M_in_21= 0.0
M_in_31= 0.0
M_in_12= 524.5000000307168
M_in_22= 10199.99999999988
M_in_32= 1010.499999798615
M_in_13= 982.30420709173
M_in_23= 451.5000000102391
M_in_33= 3238.752490110579
M_in_14= 0.0
M_in_24= 0.0
M_in_34= 0.0
M_in_15= 192.0908408095072
M_in_25= 5122.42242158686
M_in_35= 448.2119618888503
M_in_16= 4916.590841593264
M_in_26= 490922.4224215868
M_in_36= 6528.711961687465
M_in_17= 5439.115210849754
M_in_27= 7500.000000000001
M_in_37= 268999.3762445113
M_in_18= 6582.30420709173
M_in_28= 1851.500000010239
M_in_38= 524038.7524901106
M_out_11= 750.0
M_out_21= 20000.0

```
M_out_31= 1750.0
M_out_12= 3924.500000030717
M_out_22= 424999.9999999999
M_out_32= 5600.499999798615
M_out_13= 6582.30420709173
M_out_23= 1851.500000010239
M_out_33= 524038.7524901106
M_out_14= 160.0
M_out_24= 480.0
M_out_34= 160.0
M_out_15= 992.0908408095072
M_out_25= 65922.42242158686
M_out_35= 928.2119618888503
M_out_16= 4916.590841593264
M_out_26= 490.9224224215872
M_out_36= 6528.711961687465
M_out_17= 1631.734563254926
M_out_27= 750.0
M_out_37= 5379.98752489023
M_out_18= 329.1152103545868
M_out_28= 1851.500000010239
M_out_38= 262019.3762450553
C_out_11 =  15.0
C_out_21 =  400.0
C_out_31 =  35.0
C_out_12 =  115.426470611287
C_out_22 =  12500.0
C_out_32 =  164.720588229371
C_out_13 =  72.89373429580645
C_out_23 =  20.50387596518777
C_out_33 =  5803.308443834147
C_out_14 =  16.49484535875426
C_out_24 =  49.48453607626277
C_out_34 =  16.49484535875426
C_out_15 =  77.47044341430066
C_out_25 =  5147.753698987673
C_out_35 =  72.48226604256593
C_out_16 =  105.0417671660795
C_out_26 =  10.48843811669218
C_out_36 =  139.4843426648287
C_out_17 =  10.87823042169951
C_out_27 =  5.0
C_out_37 =  35.86658349985049
C_out_18 =  3.644686712086775
C_out_28 =  20.50387596553261
C_out_38 =  2901.654221907626
B_fw_11= 1.0
B_fw_12= 0.0
B_fw_13= 0.0
B_fw_14= 1.0
B_fw_15= 0.0
B_ua_11= 0.0
B_ua_12= 1.0
B_ua_13= 0.0
B_ua_14= 0.0
B_ua_15= 1.0
B_ua_16= 0.0
B_ua_17= 1.0
B_ua_18= 0.0
B_ua_21= 0.0
B_ua_22= 0.0
```

```
B_ua_23= 0.0
B_ua_24= 0.0
B_ua_25= 0.0
B_ua_26= 1.0
B_ua_27= 0.0
B_ua_28= 0.0
B_ua_31= 0.0
B_ua_32= 0.0
B_ua_33= 0.0
B_ua_34= 0.0
B_ua_35= 0.0
B_ua_36= 0.0
B_ua_37= 0.0
B_ua_38= 1.0
B_ua_41= 0.0
B_ua_42= 1.0
B_ua_43= 0.0
B_ua_44= 0.0
B_ua_45= 0.0
B_ua_46= 0.0
B_ua_47= 0.0
B_ua_48= 0.0
B_ua_51= 0.0
B_ua_52= 0.0
B_ua_53= 0.0
B_ua_54= 0.0
B_ua_55= 0.0
B_ua_56= 1.0
B_ua_57= 0.0
B_ua_58= 0.0
B_ua_61= 0.0
B_ua_62= 0.0
B_ua_63= 0.0
B_ua_64= 0.0
B_ua_65= 0.0
B_ua_66= 0.0
B_ua_67= 1.0
B_ua_68= 0.0
B_ua_71= 0.0
B_ua_72= 0.0
B_ua_73= 1.0
B_ua_74= 0.0
B_ua_75= 0.0
B_ua_76= 0.0
B_ua_77= 0.0
B_ua_78= 0.0
B_ua_81= 0.0
B_ua_82= 0.0
B_ua_83= 0.0
B_ua_84= 0.0
B_ua_85= 0.0
B_ua_86= 0.0
B_ua_87= 1.0
B_ua_88= 0.0
B_out_11= 0.0
B_out_21= 0.0
B_out_31= 0.0
B_out_41= 0.0
B_out_51= 0.0
B_out_61= 0.0
B_out_71= 1.0
```

```
B_out_81= 0.0
Cost_fw_1= 102683.9999964774
Cost_tu_6= 427336.0237609891
Cost_tu_7= 50680.26247889105
Cost_tu_8= 11225.93818994399
Cost_fw_s_sum= 102683.9999964774
Cost_tu_u_sum= 489242.2244298242
O_cost.value= 591926.2244263015
Freshwater use compared to linear system: -61.73076923208206 %
Wastewater produced compared to linear system: -61.73076923208206 %
```

In this run, the initial value of $C_{c,tu}^{out}$ was set to C_c^{env}.

```
============================================================
Initiation:
MILP_M_terms_result_initiation: 2650124.830769232
MILP_Cost_result_initiation: 101480.0
MILP_result_initiation: ([50.0, 3.552713678800501e-15, 0.0, 9.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 59.0, 0.0, -0.0, 28.69230769230768, 0.0, -0.0, 21.30769230769232,
0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 47.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0,
0.0, 56.0, 0.0, -0.0, 9.0, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0,
9.0, 12.30769230769232, 0.0, -0.0, 0.0, 56.0, -0.0, 0.0, 0.0, 0.0, 0.0, -0.0,
9.307692307692314, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0,
0.0, 0.0], [0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0], [50.0, 47.0, 56.0, 9.0,
21.30769230769232, 56.0, 68.30769230769232, 0.0])
LP_M_terms_result_initiation: -1.233729562954977e-09
LP_Cost_result_initiation: nope
LP_result_initiation: [15.00000000002, 400.0, 35.0, 94.613865558585,
9092.081737701033, 153.1166083873241, 187.8528181863891, 33.15372693032038,
9437.75400493776, 17.77777777777778, 53.33333333333334, 17.77777777777778,
52.54512635381059, 3253.429602888085, 57.52707581227435, 87.85281818638911,
8.153726930320381, 137.7540049377625, 49.04191616766924, 61.33836144902785,
154.9524605343391, -0.0, -0.0, -0.0]
============================================================
============================================================
Iteration_1:
MILP_M_terms_result_iteration_1: 2874669.320608941
MILP_Cost_result_iteration_1: 0.0
MILP_result_iteration_1:([0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, -0.0, 9.0, 0.0, -0.0, 41.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0,
56.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 56.0, -0.0, 47.0, 0.0, -0.0,
0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 41.0, -0.0, 0.0, 56.0,
-0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, 50.0, 0.0, 0.0,
47.0, 0.0, 0.0, 0.0, 0.0], [0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0], [50.0, 56.0,
56.0, 47.0, 41.0, 56.0, 0.0, 97.0])
LP_M_terms_result_iteration_1: 340464.104310618
LP_Cost_result_iteration_1: nope
LP_result_iteration_1:[15.0, 400.0, 35.0, 70.73055807108896, 7521.785714285714,
104.375, 170.730558071089, 0.0, 9404.375, 9.061941531510266, 60.0, 20.0,
34.51219512195122, 1882.926829268293, 46.70731707317074, 70.73055807108896,
7.521785714285714, 104.375, -0.0, -0.0, -0.0, 5.65768621236133, 795.8762886597938,
2724.536082474227]
============================================================
============================================================
Iteration_2:
MILP_M_terms_result_iteration_2: 2140937.20546839
MILP_Cost_result_iteration_2: 0.0
MILP_result_iteration_2:([0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, -0.0, 24.0, 0.0, -0.0, 26.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0,
```

```
0.0, 18.83815448898108, 15.16184551101891, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 0.0,
56.0, -0.0, 9.999999999999998, 0.0, -0.0, -3.858025010572419e-15, 0.0, 0.0, 0.0,
-0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 26.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0,
50.0, 0.0, 56.0, 9.999999999999995, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0,
0.0, 97.16184551101891, 0.0], [0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, -5.551115123125783e-17, 0.0, 0.0, 0.0,
0.0, 0.0], [50.0, 34.0, 56.0, 9.999999999999995, 26.0, 0.0, 116.0,
97.16184551101891])
LP_M_terms_result_iteration_2: 51721.37221421315
LP_Cost_result_iteration_2: nope
LP_result_iteration_2:[15.0, 400.0, 35.0, 116.4705882352941, 12500.0,
165.5882352941177, 106.8302007325523, 45.0, 9345.992669504334, 20.0, 60.0, 20.0,
45.76923076923077, 2738.461538461539, 53.46153846153847, -0.0, -0.0, -0.0,
6.830200732552168, 429.9310344827586, 45.99266950433371, 4.599748110697562,
2709.325533116625, 2713.391300674797]
============================================================
============================================================
Iteration_3:
MILP_M_terms_result_iteration_3: 3098503.119248109
MILP_Cost_result_iteration_3: 0.0
MILP_result_iteration_3:([0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, -0.0, 36.40503102073323, 0.0, -0.0, 13.59496897926677, 0.0, 0.0, 0.0,
-0.0, 0.0, 0.0, -0.0, 0.0, 9.0, 9.000000000000014, 33.57379394603872, -0.0, 0.0,
0.0, -0.0, 0.0, 0.0, 9.000000000000002, 57.42620605396128, -0.0, 15.16876292530551,
22.14206868465376, -0.0, 9.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0,
31.59496897926677, 0.0, 0.0, 50.0, 0.0, 44.28413736930752, 46.31083160995926, 0.0,
0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 9.0, 0.0, 0.0, 9.0, -0.0, 0.0, 0.0, -0.0, 0.0,
100.0, 0.0, 0.0], [0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0], [50.0,
51.57379394603874, 66.4262060539613, 46.31083160995926, 31.59496897926677,
140.5949689792668, 18.00000000000001, 100.0])
LP_M_terms_result_iteration_3: 183421.1617560006
LP_Cost_result_iteration_3: nope
LP_result_iteration_3:[15.0, 400.0, 35.0, 82.3955423613643, 8329.145479070057,
119.5869233643124, 103.4136652936329, 27.58984606153442, 9113.698945245747, 20.0,
13.42365734428337, 20.0, 45.41124778330277, 2219.319520463152, 62.25103888343449,
18.66438221443198, 3.058911488655615, 1900.12965534901, 27.8713811482496,
417.8367662565802, 92.3328586861006, 4.477913921797171, 2849.85925142114,
2641.055679976505]
============================================================
============================================================
Iteration_4:
MILP_M_terms_result_iteration_4: 2160440.426734039
MILP_Cost_result_iteration_4: 196940.0
MILP_result_iteration_4:([50.0, 8.499999999999169, 46.99999999999997,
9.000000000000858, -0.0, 15.50000000000004, 0.0, 25.44225424926354, -0.0, 9.0,
30.6281842656544, 18.0, 15.92956148508202, -0.0, 25.49999999999997, 0.0, -0.0, 9.0,
0.0, -1.421085471520202e-14, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 9.0, 9.00000000000001,
15.99999999999999, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 8.999999999999996,
21.55774575073647, -0.0, 8.580395407869086e-13, 9.0, -0.0, -0.0, -0.0, -0.0, -0.0,
-0.0, -0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 3.052130716050979e-14, -0.0,
0.0, 0.0, 0.0, 0.0, -0.0, -0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0,
0.0, 21.62818426565443, 0.0, 0.0], [0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 9.533772675410095e-14, 1.0, 0.0, -0.0, 0.0, -0.0,
0.0, 0.0, 0.0], [50.0, 34.0, 56.0, 9.000000000000858, 9.0, 30.62818426565443,
17.99999999999999, 37.55774575073646])
LP_M_terms_result_iteration_4: 8.178925980928398e-09
LP_Cost_result_iteration_4: nope
LP_result_iteration_4:[15.00000000001999, 400.0000000000209, 35.00000000001998,
111.2500000000178, 12500.0, 161.2500000000019, 102.8571428571426, 33.57142857142777,
9302.857142857149, 17.77777777777608, 53.33333333332825, 17.77777777777608,
```

```
103.8888888889089, 7155.555555555576, 88.33333333335332, 36.4483631529768,
7.447053608040032, 1956.974156746323, 32.11607142857407, 626.6785714285714,
94.64107142857168, 5.321629472774418, 5344.402873745316, 2704.217531974138]
============================================================
MINLP function result:
WARNING (W1001): Setting Var 'F_ua_17' to a value `-1.421085471520202e-14`
(float) not in domain NonNegativeReals.
    See also https://pyomo.readthedocs.io/en/stable/errors.html#w1001
WARNING (W1002): Setting Var 'C_out_11' to a numeric value `15.000000000001999`
outside the bounds (0, 15).
    See also https://pyomo.readthedocs.io/en/stable/errors.html#w1002
WARNING (W1002): Setting Var 'C_out_21' to a numeric value `400.0000000000209`
outside the bounds (0, 400).
    See also https://pyomo.readthedocs.io/en/stable/errors.html#w1002
WARNING (W1002): Setting Var 'C_out_31' to a numeric value `35.000000000001998`
outside the bounds (0, 35).
    See also https://pyomo.readthedocs.io/en/stable/errors.html#w1002
# ==========================================================
# = Solver Results                                         =
# ==========================================================
# ----------------------------------------------------------
#   Problem Information
# ----------------------------------------------------------
Problem:
- Lower bound: -inf
  Upper bound: inf
  Number of objectives: 1
  Number of constraints: 389
  Number of variables: 241
  Sense: unknown
# ----------------------------------------------------------
#   Solver Information
# ----------------------------------------------------------
Solver:
- Status: ok
  Message: SCIP 8.0.3\x3a optimal solution; objective 673010.997895607; 1.24565e+06
  simplex iterations; 14425 branching nodes
  Termination condition: optimal
  Id: 0
  Error rc: 0
  Time: 101.14606547355652
# ----------------------------------------------------------
#   Solution Information
# ----------------------------------------------------------
Solution:
- number of solutions: 0
  number of solutions displayed: 0
F_w_11= 50.0
F_w_12= 8.500000000000174
F_w_13= 44.99999999999997
F_w_14= 9.000000000000002
F_w_15= 0.0
F_out_11= -0.0
F_out_21= 0.0
F_out_31= -0.0
F_out_41= 0.0
F_out_51= -0.0
F_out_61= 0.0
F_out_71= 100.0
F_out_81= 12.50000000000013
F_ua_11= 0.0
```

```
F_ua_12= 25.49999999999983
F_ua_13= 0.0
F_ua_14= 0.0
F_ua_15= 12.14539538264042
F_ua_16= 0.0
F_ua_17= 12.35460461735975
F_ua_18= 0.0
F_ua_21= 0.0
F_ua_22= 0.0
F_ua_23= 0.0
F_ua_24= 0.0
F_ua_25= 0.0
F_ua_26= 34.0
F_ua_27= 0.0
F_ua_28= 0.0
F_ua_31= 0.0
F_ua_32= 0.0
F_ua_33= 0.0
F_ua_34= 0.0
F_ua_35= 0.0
F_ua_36= 0.0
F_ua_37= 91.49999999999984
F_ua_38= 0.0
F_ua_41= 0.0
F_ua_42= 0.0
F_ua_43= 9.000000000000002
F_ua_44= 0.0
F_ua_45= 0.0
F_ua_46= 0.0
F_ua_47= 0.0
F_ua_48= 0.0
F_ua_51= 0.0
F_ua_52= 0.0
F_ua_53= 0.0
F_ua_54= 0.0
F_ua_55= 0.0
F_ua_56= 12.14539538264042
F_ua_57= 0.0
F_ua_58= 0.0
F_ua_61= 0.0
F_ua_62= 0.0
F_ua_63= 0.0
F_ua_64= 0.0
F_ua_65= 0.0
F_ua_66= 0.0
F_ua_67= 46.14539538264042
F_ua_68= 0.0
F_ua_71= 0.0
F_ua_72= 0.0
F_ua_73= 0.0
F_ua_74= 0.0
F_ua_75= 0.0
F_ua_76= 0.0
F_ua_77= 0.0
F_ua_78= 50.0
F_ua_81= 0.0
F_ua_82= 0.0
F_ua_83= 37.49999999999987
F_ua_84= 0.0
F_ua_85= 0.0
F_ua_86= 0.0
```

```
F_ua_87= 0.0
F_ua_88= 0.0
F_t_1= 50.0
F_t_2= 34.0
F_t_3= 91.49999999999984
F_t_4= 9.000000000000002
F_t_5= 12.14539538264043
F_t_6= 46.14539538264042
F_t_7= 150.0
F_t_8= 50.0
M_in_11= 0.0
M_in_21= 0.0
M_in_31= 0.0
M_in_12= 382.4999999999973
M_in_22= 10200.0
M_in_32= 892.4999999999936
M_in_13= 200.3136758225855
M_in_23= 667.5000004661647
M_in_33= 1482.756891928148
M_in_14= 0.0
M_in_24= 0.0
M_in_34= 0.0
M_in_15= 182.1809307396062
M_in_25= 4858.158153056167
M_in_35= 425.0888383924148
M_in_16= 4764.680930739603
M_in_26= 490658.1581530562
M_in_36= 6387.588838392408
M_in_17= 10750.31367582257
M_in_27= 7500.000004661904
M_in_37= 529102.7568910959
M_in_18= 1075.031367580481
M_in_28= 250.0000003836518
M_in_38= 3527.351712631169
M_out_11= 750.0
M_out_21= 20000.0
M_out_31= 1750.0
M_out_12= 3782.499999999997
M_out_22= 425000.0
M_out_32= 5482.499999999994
M_out_13= 5800.313675822586
M_out_23= 2067.500000466165
M_out_33= 522282.7568919281
M_out_14= 160.0
M_out_24= 480.0
M_out_34= 160.0
M_out_15= 982.1809307396062
M_out_25= 65658.15815305617
M_out_35= 905.0888383924148
M_out_16= 4764.680930739603
M_out_26= 490.6581581530566
M_out_36= 6387.588838392408
M_out_17= 3225.094102746772
M_out_27= 750.0000004661903
M_out_37= 10582.05513782193
M_out_18= 53.75156837902412
M_out_28= 250.0000003836518
M_out_38= 1763.675856315584
C_out_11 =  14.99999999999999
C_out_21 =  400.0
C_out_31 =  34.99999999999999
```

```
C_out_12 =  111.2499999999999
C_out_22 =  12500.0
C_out_32 =  161.2499999999998
C_out_13 =  63.39140629314313
C_out_23 =  22.59562842039528
C_out_33 =  5708.00827204294
C_out_14 =  17.77777777777777
C_out_24 =  53.33333333333331
C_out_34 =  17.77777777777778
C_out_15 =  80.86858427908705
C_out_25 =  5406.012409188616
C_out_35 =  74.5211506188818
C_out_16 =  103.2536592487848
C_out_26 =  10.63287362993187
C_out_36 =  138.4231034232961
C_out_17 =  21.50062735166276
C_out_27 =  5.000000000825175
C_out_37 =  70.54703425261287
C_out_18 =  1.075031355268952
C_out_28 =  5.000000012431074
C_out_38 =  35.27351711808408
B_fw_11= 1.0
B_fw_12= 1.0
B_fw_13= 1.0
B_fw_14= 1.0
B_fw_15= 0.0
B_ua_11= 0.0
B_ua_12= 1.0
B_ua_13= 0.0
B_ua_14= 0.0
B_ua_15= 1.0
B_ua_16= 0.0
B_ua_17= 1.0
B_ua_18= 0.0
B_ua_21= 0.0
B_ua_22= 0.0
B_ua_23= 0.0
B_ua_24= 0.0
B_ua_25= 0.0
B_ua_26= 1.0
B_ua_27= 0.0
B_ua_28= 0.0
B_ua_31= 0.0
B_ua_32= 0.0
B_ua_33= 0.0
B_ua_34= 0.0
B_ua_35= 0.0
B_ua_36= 0.0
B_ua_37= 1.0
B_ua_38= 0.0
B_ua_41= 0.0
B_ua_42= 0.0
B_ua_43= 1.0
B_ua_44= 0.0
B_ua_45= 0.0
B_ua_46= 0.0
B_ua_47= 0.0
B_ua_48= 0.0
B_ua_51= 0.0
B_ua_52= 0.0
B_ua_53= 0.0
```

```
B_ua_54= 0.0
B_ua_55= 0.0
B_ua_56= 1.0
B_ua_57= 0.0
B_ua_58= 0.0
B_ua_61= 0.0
B_ua_62= 0.0
B_ua_63= 0.0
B_ua_64= 0.0
B_ua_65= 0.0
B_ua_66= 0.0
B_ua_67= 1.0
B_ua_68= 0.0
B_ua_71= 0.0
B_ua_72= 0.0
B_ua_73= 0.0
B_ua_74= 0.0
B_ua_75= 0.0
B_ua_76= 0.0
B_ua_77= 0.0
B_ua_78= 1.0
B_ua_81= 0.0
B_ua_82= 0.0
B_ua_83= 1.0
B_ua_84= 0.0
B_ua_85= 0.0
B_ua_86= 0.0
B_ua_87= 0.0
B_ua_88= 0.0
B_out_11= -0.0
B_out_21= 0.0
B_out_31= -0.0
B_out_41= 0.0
B_out_51= -0.0
B_out_61= 0.0
B_out_71= 1.0
B_out_81= 1.0
Cost_fw_1= 193500.0000000003
Cost_tu_6= 421408.7475662808
Cost_tu_7= 50680.26247889105
Cost_tu_8= 7421.987850435127
Cost_fw_s_sum= 193500.0000000003
Cost_tu_u_sum= 479510.997895607
O_cost.value= 673010.9978956071
Freshwater use compared to linear system: -27.88461538461529 %
Wastewater produced compared to linear system: -27.8846153846153 %
```

G.3. Petroleum Refinery Case Study, Scenario 2

In this run, the initial value of $C_{c,tu}^{out}$ was set to 0.

```
============================================================
Initiation:
MILP_M_terms_result_initiation: 2907257.0
MILP_Cost_result_initiation: 0.0
MILP_result_initiation: ([0.0, 0.0, 5.551115123125783e-17, -5.551115123125783e-17,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 0.0,
50.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 34.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0,
59.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 100.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0,
0.0, 0.0, 16.0, 41.0, 0.0, 59.0, 0.0, 0.0, 0.0, 0.0, 0.0, 9.0, 34.0, 0.0, 0.0,
16.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0], [50.0, 34.0, 59.0, 100.0, 16.0, 100.0, 59.0, 100.0])
LP_M_terms_result_initiation: 502362.6695540038
LP_Cost_result_initiation: nope
LP_result_initiation: [15.0, 400.0, 35.0, 120.0, 0.0, 180.0, 99.56922036616712,
23.78881355939205, 8847.118644067796, 4.653966128878801, 60.0, 20.0,
79.87076610985014, 3802.378881355939, 120.0, 4.653966128878984, 0.06000000007001006,
20.0, 29.87076610985014, 2.378881355939205, 176.9423728813559, 3.053966128878801,
808.3806210169502, 48.95000000000501]
============================================================
============================================================
Iteration_1:
MILP_M_terms_result_iteration_1: 2124180.410958158
MILP_Cost_result_iteration_1: 101480.0
MILP_result_iteration_1:([50.0, 0.0, 0.0, 9.0, 0.0, 0.0, 0.0, 59.0, 0.0, 0.0, 0.0,
0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 50.0, 0.0, 0.0, 0.0, -0.0, 0.0, 34.0, -0.0, 0.0,
0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 9.0, -0.0, 0.0,
0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 9.0, 0.0, 41.0, -0.0, 21.38376860017696,
16.0, -0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, -0.0,
12.61623139982304, 0.0, -0.0, 0.0, 28.38376860017696, 0.0, 0.0], [0.0, 0.0, 0.0,
0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0], [50.0, 34.0, 59.0, 9.0, 50.0, 37.38376860017696,
0.0, 41.0])
LP_M_terms_result_iteration_1: 434473.2421121721
LP_Cost_result_iteration_1: nope
LP_result_iteration_1:[15.0, 400.0, 35.0, 106.0091292201879, 12500.0,
160.676522295762, 161.0601694915254, 45.0, 8929.92711864407, 17.77777777777778,
53.33333333333334, 17.77777777777778, 31.0, 1616.0, 44.6, 8.639975407100753,
1.616, 27.66864012150583, -0.0, -0.0, -0.0, 1.55, 1616.0, 22.3]
============================================================
============================================================
Iteration_2:
MILP_M_terms_result_iteration_2: 2462937.678980644
MILP_Cost_result_iteration_2: 0.0
MILP_result_iteration_2:([0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 50.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0,
34.00000000000006, -6.394884621840902e-14, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0,
56.0, 0.0, -0.0, 25.00000000000006, 0.0, -0.0, 0.0, 65.99999999999994, 0.0, 0.0,
-0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 85.00000000000006, 9.0, -0.0,
-5.684341886080801e-14, 56.0, -0.0, 94.00000000000006, 0.0, 0.0, 0.0, 50.0,
8.999999999999993, 0.0, 91.0, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0,
0.0, 9.0, 0.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 1.0, 0.0, 0.0, -7.105427357601002e-15, 0.0, 0.0, 0.0, 0.0, 0.0],
[50.0, 34.0, 56.0, 91.0, 94.00000000000006, 150.0, 150.0, 9.0])
LP_M_terms_result_iteration_2: 12186.01831458354
LP_Cost_result_iteration_2: nope
LP_result_iteration_2:[15.0, 400.0, 35.0, 120.0, 12245.63811177586,
```

```
168.4043139606241, 141.0, 27.92970642487277, 9358.638311164408, 20.0,
47.03436258387922, 20.0, 49.51063829787234, 649.7382170631706, 63.74469414313155,
41.0, 2.929706424872769, 58.63831116440814, 24.25336808510638, 41.75963730915394,
70.63851940680227, 2.475531914893617, 649.7382170631706, 31.87234707156577]
============================================================
============================================================
Iteration_3:
MILP_M_terms_result_iteration_3: 2417384.328987328
MILP_Cost_result_iteration_3: 101480.0
MILP_result_iteration_3:([50.0, 0.0, 0.0, 9.0, 0.0, 0.0, 0.0, 48.22113250070878,
0.0, 0.0, 0.0, 10.77886749929122, 0.0, -0.0, 11.67172464214836, 0.0, -0.0, 0.0,
38.32827535785164, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 19.62456934948715, 0.0,
14.37543065051285, -0.0, 0.0, 0.0, -0.0, 0.0, -0.0, 9.0, 0.0, -0.0, 9.0, 0.0,
-0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 15.10714285714285, 0.0,
-0.0, 0.0, 57.22113250070878, -0.0, 15.10714285714285, 0.0, 0.0, 0.0, -0.0,
13.32827535785164, 0.0, -0.0, 0.0, -0.0, 0.0, 0.0, -0.0, -0.0, 0.0, -0.0, 0.0,
14.37543065051286, 1.633167995622868e-15, 0.0], [0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0], [50.0, 34.0, 57.22113250070878, 9.0, 15.10714285714285, 72.32827535785164,
24.10714285714285, 14.37543065051285])
LP_M_terms_result_iteration_3: 100.5959996823558
LP_Cost_result_iteration_3: nope
LP_result_iteration_3:[15.0, 400.0, 35.0, 120.0, 12450.89521504224,
179.1833148962227, 139.5664384914084, 30.53135756262992, 9186.503519530337,
17.77777777777778, 53.33333333333334, 17.77777777777778, 94.6555798364515,
4030.651160629389, 116.7440192292707, 41.70049709413469, 6.064872213311494,
84.97096958388059, 33.42669012029064, 253.7273100817798, 70.05575132016673,
6.0, 12450.89521504224, 89.59165744811138]
============================================================
============================================================
Iteration_4:
MILP_M_terms_result_iteration_4: 2481183.13088508
MILP_Cost_result_iteration_4: 102702.2846146816
MILP_result_iteration_4:([50.71063058993118, 0.0, -0.0, 9.000000000000004, -0.0,
-0.0, -0.0, 47.0, 0.0, -0.0, -0.0, 12.71063058993118, 0.0, -0.0, 26.60348773278834,
0.0, -0.0, 24.10714285714284, 0.0, -0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 47.0, 0.0,
0.0, -0.0, 0.0, 0.0, -0.0, 0.0, -0.0, 9.0, 0.0, -0.0, 9.000000000000004, 0.0, -0.0,
0.0, 0.0, 0.0, 0.0, -0.0, -0.0, 0.0, -0.0, 0.0, 9.0, 15.10714285714284, 0.0, -0.0,
0.0, 56.0, -0.0, 0.0, 0.0, -0.0, 0.0, -0.0, 11.39651226721166, 0.0, -0.0, 0.0, -0.0,
0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0], [0.0, 1.0, 0.0, 0.0, 1.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, -0.0, 0.0, -0.0,
0.0, 0.0, 0.0], [50.71063058993118, 47.0, 56.0, 9.000000000000004,
24.10714285714284,
56.0, 24.10714285714284, 0.0])
LP_M_terms_result_iteration_4: -1.159605744760484e-10
LP_Cost_result_iteration_4: nope
LP_result_iteration_4:[14.78979833764709, 394.3946223372558, 34.50952945450988,
91.30954274624749, 9103.601005931727, 137.8257106503365, 184.3450600139129,
33.10924045618737, 9424.42132438672, 17.77777777777777, 53.33333333333331,
17.77777777777777, 47.9749835228323, 2916.468696411332, 54.420640565621,
84.3450600139129, 8.109240456187377, 124.4213243867215, 29.66594362385073,
184.001449952141, 71.05108458384342, -0.0, -0.0, -0.0]
============================================================
MINLP function result:
```

No result came from the MINLP, so the run was manually stopped after 1000s.

In this run, the initial value of $C_{c,tu}^{out}$ was set to C_c^{env}.

```
============================================================
Initiation:
MILP_M_terms_result_initiation: 2534801.000000001
MILP_Cost_result_initiation: 101480.0
MILP_result_initiation: ([50.0, 0.0, 0.0, 9.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
59.0, 0.0, -0.0, 27.0, 0.0, -0.0, 11.0, 0.0, 12.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0,
45.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 56.0, 0.0, -0.0, 9.0, 0.0, -0.0,
0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 11.0, 0.0, 0.0, -0.0, 0.0, 56.0,
-0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 9.0, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0,
-0.0, 0.0, 0.0, 0.0, 0.0], [0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0], [50.0, 45.0,
56.0, 9.0, 11.0, 56.0, 68.0, 0.0])
LP_M_terms_result_initiation: -9.609948392608203e-10
LP_Cost_result_initiation: nope
LP_result_initiation: [15.0, 400.0, 35.0, 97.95184618002382, 9470.412494021573,
157.6838362986992, 195.9434478232334, 33.77443861126734, 9442.156654168597,
17.77777777777778, 53.33333333333334, 17.77777777777778, 87.72727272727273,
5927.272727272727, 78.63636363636364, 95.94344782323341, 8.774438611267335,
142.1566541685976, 49.20367534456355, 9.840247885647436, 155.6414037157181, -0.0,
-0.0, -0.0]
============================================================
============================================================
Iteration_1:
MILP_M_terms_result_iteration_1: 2481303.208460547
MILP_Cost_result_iteration_1: 15480.0
MILP_result_iteration_1:([0.0, 0.0, 0.0, 8.999999999999993, 7.105427357601002e-15,
9.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 41.0, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0,
-0.0, 0.0, 0.0, -0.0, 0.0, 56.00000000000001, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0,
0.0, 55.99999999999999, 0.0, -0.0, 0.0, -7.105427357601002e-15, -0.0, 9.0, 0.0, 0.0,
0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 9.000000000000007, 0.0, -0.0, 0.0,
56.00000000000001, -0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 0.0,
65.0, 50.0, 15.00000000000001, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0], [0.0, 1.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
-7.894919286223335e-16, 0.0, 1.000000000000007, 0.0, 0.0, 0.0, 0.0, 0.0], [50.0,
56.00000000000001, 56.0, 8.999999999999993, 9.000000000000007, 56.00000000000001,
65.0, 65.0])
LP_M_terms_result_iteration_1: 9044.070817285074
LP_Cost_result_iteration_1: nope
LP_result_iteration_1:[15.0, 400.0, 35.0, 72.35237360092628, 7726.00796282112,
129.3757572419854, 172.3523736009263, 32.72600796280173, 9429.375757241985,
17.77777777777779, 53.33333333333338, 17.77777777777779, 106.6666666666667,
6808.888888888889, 71.11111111111111, 72.35237360092628, 7.72600796282112,
129.3757572419854, 48.97722886916249, 97.09639453217984, 162.6723207401696,
2.448861443458124, 97.09639453217984, 81.33616037008481]
============================================================
============================================================
Iteration_2:
MILP_M_terms_result_iteration_2: 2403365.597179777
MILP_Cost_result_iteration_2: 101480.0
MILP_result_iteration_2:([50.0, 0.0, 0.0, 9.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
59.0, 0.0, -0.0, 40.57673946045774, 0.0, -0.0, 9.423260539542262, 0.0, 0.0, 0.0,
-0.0, 0.0, 0.0, -0.0, 0.0, 58.63362113492962, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0,
0.0, 58.63362113492962, 0.0, -0.0, 9.0, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0,
0.0, -0.0, 0.0, 0.0, 9.42326053954226, 0.0, -0.0, 0.0, 58.63362113492962, -0.0, 0.0,
0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 9.056881674471882, -0.0,
9.056881674471882, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0], [0.0, 1.0, 0.0, 0.0, 1.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0], [50.0, 58.63362113492963, 58.63362113492962, 9.0, 9.423260539542262,
58.63362113492962, 68.05688167447188, 9.056881674471882])
```

```
LP_M_terms_result_iteration_2: 1.818989403545856e-12
LP_Cost_result_iteration_2: nope
LP_result_iteration_2:[15.0, 400.0, 35.0, 71.46194916572017, 7374.512245994759,
117.2276480200116, 166.9702921881151, 31.25159800159348, 8999.503549102736,
17.77777777777778, 53.33333333333334, 17.77777777777778, 99.89630490879543,
6852.119173068452, 85.93778294527726, 71.46194916572017, 7.374512245994759,
117.2276480200116, 47.30493741560124, 97.56794160877743, 155.3063503988494,
2.365246870780063, 97.56794160877742, 77.65317519942472]
============================================================
MINLP function result:
# ============================================================
# = Solver Results                                           =
# ============================================================
# ------------------------------------------------------------
#   Problem Information
# ------------------------------------------------------------
Problem:
- Lower bound: -inf
  Upper bound: inf
  Number of objectives: 1
  Number of constraints: 656
  Number of variables: 289
  Sense: unknown
# ------------------------------------------------------------
#   Solver Information
# ------------------------------------------------------------
Solver:
- Status: ok
  Message: SCIP 8.0.3\x3a optimal solution; objective 679413.75649779; 743808
  simplex iterations; 62848 branching nodes
  Termination condition: optimal
  Id: 0
  Error rc: 0
  Time: 293.77429270744324
# ------------------------------------------------------------
#   Solution Information
# ------------------------------------------------------------
Solution:
- number of solutions: 0
  number of solutions displayed: 0
Freshwater use compared to linear system: -61.73076923214699%
Wastewater produced compared to linear system: -61.730769232146976%
F_w_11= 50.0
F_w_12= 0.0
F_w_13= 0.0
F_w_14= 9.699999997850698
F_w_15= 0.0
F_out_11= 0.0
F_out_21= 0.0
F_out_31= 0.0
F_out_41= 0.0
F_out_51= 0.0
F_out_61= 0.0
F_out_71= 59.6999999978507
F_out_81= 0.0
F_ua_11= 0.0
F_ua_12= 24.3000000021493
F_ua_13= 0.0
F_ua_14= 0.0
F_ua_15= 10.6197954016861
F_ua_16= 15.08020459616459
```

```
F_ua_17= 0.0
F_ua_18= 0.0
F_ua_21= 0.0
F_ua_22= 0.0
F_ua_23= 0.0
F_ua_24= 0.0
F_ua_25= 0.0
F_ua_26= 34.00000000000001
F_ua_27= 0.0
F_ua_28= 0.0
F_ua_31= 0.0
F_ua_32= 0.0
F_ua_33= 0.0
F_ua_34= 0.0
F_ua_35= 0.0
F_ua_36= 0.0
F_ua_37= 0.0
F_ua_38= 59.6999999978507
F_ua_41= 0.0
F_ua_42= 9.699999997850698
F_ua_43= 0.0
F_ua_44= 0.0
F_ua_45= 0.0
F_ua_46= 0.0
F_ua_47= 0.0
F_ua_48= 0.0
F_ua_51= 0.0
F_ua_52= 0.0
F_ua_53= 0.0
F_ua_54= 0.0
F_ua_55= 0.0
F_ua_56= 10.6197954016861
F_ua_57= 0.0
F_ua_58= 0.0
F_ua_61= 0.0
F_ua_62= 0.0
F_ua_63= 59.6999999978507
F_ua_64= 0.0
F_ua_65= 0.0
F_ua_66= 0.0
F_ua_67= 0.0
F_ua_68= 0.0
F_ua_71= 0.0
F_ua_72= 0.0
F_ua_73= 0.0
F_ua_74= 0.0
F_ua_75= 0.0
F_ua_76= 0.0
F_ua_77= 0.0
F_ua_78= 0.0
F_ua_81= 0.0
F_ua_82= 0.0
F_ua_83= 0.0
F_ua_84= 0.0
F_ua_85= 0.0
F_ua_86= 0.0
F_ua_87= 59.6999999978507
F_ua_88= 0.0
F_t_1= 50.0
F_t_2= 34.0
F_t_3= 59.6999999978507
```

```
F_t_4= 9.699999997850698
F_t_5= 10.6197954016861
F_t_6= 59.6999999978507
F_t_7= 59.6999999978507
F_t_8= 59.6999999978507
M_in_11= 0.0
M_in_21= 0.0
M_in_31= 0.0
M_in_12= 524.5000000322395
M_in_22= 10200.0
M_in_32= 1010.499999098703
M_in_13= 5110.000000403091
M_in_23= 496.0799999008713
M_in_33= 6979.999998981392
M_in_14= 0.0
M_in_24= 0.0
M_in_34= 0.0
M_in_15= 159.2969310252916
M_in_25= 4247.918160674442
M_in_35= 371.6928390590136
M_in_16= 5110.0
M_in_26= 496079.9999991404
M_in_36= 6979.999999023478
M_in_17= 535.4999992901983
M_in_27= 1896.079999302937
M_in_37= 263889.9999994907
M_in_18= 10710.00000040309
M_in_28= 1896.079999302937
M_in_38= 527779.9999989814
M_out_11= 750.0
M_out_21= 20000.0
M_out_31= 1750.0
M_out_12= 3924.50000003224
M_out_22= 425000.0
M_out_32= 5600.499999098703
M_out_13= 10710.00000040309
M_out_23= 1896.079999900871
M_out_33= 527779.9999989814
M_out_14= 160.0
M_out_24= 480.0
M_out_34= 160.0
M_out_15= 959.2969310252915
M_out_25= 65047.91816067444
M_out_35= 851.6928390590136
M_out_16= 5110.0
M_out_26= 496.0799999991409
M_out_36= 6979.999999023478
M_out_17= 160.6499997870595
M_out_27= 189.6079999302937
M_out_37= 5277.799999989818
M_out_18= 535.5000000201551
M_out_28= 1896.079999302937
M_out_38= 263889.9999994907
C_out_11 =  15.0
C_out_21 =  400.0
C_out_31 =  35.0
C_out_12 =  115.4264705891835
C_out_22 =  12500.0
C_out_32 =  164.7205882087854
C_out_13 =  179.3969849361848
C_out_23 =  31.76013399318099
```

```
C_out_33 =  8840.536013700448
C_out_14 =  16.4948453661361
C_out_24 =  49.48453600977616
C_out_34 =  16.49484526546181
C_out_15 =  90.33101812382625
C_out_25 =  6125.157378328309
C_out_35 =  80.19861086540433
C_out_16 =  85.59463987630775
C_out_26 =  8.309547737497843
C_out_36 =  116.9179229356238
C_out_17 =  2.690954770282885
C_out_27 =  3.176013399441079
C_out_37 =  88.40536013555295
C_out_18 =  8.969849246656313
C_out_28 =  31.76013399357563
C_out_38 =  4420.268006852244
B_fw_11= 1.0
B_fw_12= 0.0
B_fw_13= 0.0
B_fw_14= 1.0
B_fw_15= 0.0
B_ua_11= 0.0
B_ua_12= 1.0
B_ua_13= 0.0
B_ua_14= 0.0
B_ua_15= 1.0
B_ua_16= 1.0
B_ua_17= 0.0
B_ua_18= 0.0
B_ua_21= 0.0
B_ua_22= 0.0
B_ua_23= 0.0
B_ua_24= 0.0
B_ua_25= 0.0
B_ua_26= 1.0
B_ua_27= 0.0
B_ua_28= 0.0
B_ua_31= 0.0
B_ua_32= 0.0
B_ua_33= 0.0
B_ua_34= 0.0
B_ua_35= 0.0
B_ua_36= 0.0
B_ua_37= 0.0
B_ua_38= 1.0
B_ua_41= 0.0
B_ua_42= 1.0
B_ua_43= 0.0
B_ua_44= 0.0
B_ua_45= 0.0
B_ua_46= 0.0
B_ua_47= 0.0
B_ua_48= 0.0
B_ua_51= 0.0
B_ua_52= 0.0
B_ua_53= 0.0
B_ua_54= 0.0
B_ua_55= 0.0
B_ua_56= 1.0
B_ua_57= 0.0
B_ua_58= 0.0
```

```
B_ua_61= 0.0
B_ua_62= 0.0
B_ua_63= 1.0
B_ua_64= 0.0
B_ua_65= 0.0
B_ua_66= 0.0
B_ua_67= 0.0
B_ua_68= 0.0
B_ua_71= 0.0
B_ua_72= 0.0
B_ua_73= 0.0
B_ua_74= 0.0
B_ua_75= 0.0
B_ua_76= 0.0
B_ua_77= 0.0
B_ua_78= 0.0
B_ua_81= 0.0
B_ua_82= 0.0
B_ua_83= 0.0
B_ua_84= 0.0
B_ua_85= 0.0
B_ua_86= 0.0
B_ua_87= 1.0
B_ua_88= 0.0
B_out_11= 0.0
B_out_21= 0.0
B_out_31= 0.0
B_out_41= 0.0
B_out_51= 0.0
B_out_61= 0.0
B_out_71= 1.0
B_out_81= 0.0
B_G1_16= 1.0
B_G1_17= 1.0
B_G1_18= 1.0
B_G1_26= 1.0
B_G1_27= 0.0
B_G1_28= 0.0
B_G1_36= 0.0
B_G1_37= 1.0
B_G1_38= 1.0
B_G1_46= 0.0
B_G1_47= 0.0
B_G1_48= 0.0
B_G1_56= 1.0
B_G1_57= 0.0
B_G1_58= 0.0
B_G1_66= 0.0
B_G1_67= 0.0
B_G1_68= 0.0
B_G1_76= 0.0
B_G1_77= 0.0
B_G1_78= 0.0
B_G1_86= 0.0
B_G1_87= 1.0
B_G1_88= 0.0
B_G2_16= 0.0
B_G2_17= 0.0
B_G2_18= 0.0
B_G2_26= 0.0
B_G2_27= 1.0
```

```
B_G2_28= 1.0
B_G2_36= 1.0
B_G2_37= 0.0
B_G2_38= 0.0
B_G2_46= 0.0
B_G2_47= 1.0
B_G2_48= 1.0
B_G2_56= 0.0
B_G2_57= 1.0
B_G2_58= 1.0
B_G2_66= 1.0
B_G2_67= 0.0
B_G2_68= 0.0
B_G2_76= 1.0
B_G2_77= 1.0
B_G2_78= 1.0
B_G2_86= 1.0
B_G2_87= 0.0
B_G2_88= 1.0
Cost_fw_1= 102683.9999963032
Cost_tu_6= 542829.7471843696
Cost_tu_7= 25497.22440201635
Cost_tu_8= 8402.784915101025
Cost_fw_s_sum= 102683.9999963032
Cost_tu_u_sum= 576729.756501487
O_cost.value= 679413.7564977901
```

G.4. Petroleum Refinery Case Study, Scenario 3

In this run, the initial value of $C_{c,tu}^{out}$ was set to 0.

```
============================================================
Initiation:
MILP_M_terms_result_initiation: 4962695.586666674
MILP_Cost_result_initiation: 13332.19999999999
MILP_result_initiation: ([0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, -0.0, 56.00000000000016, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0,
-0.0, 0.0, 0.0, 0.0, 79.33333333333354, -0.0, 0.0, 0.0, -0.0, 0.0, 56.0, 0.0, 0.0,
-0.0, 23.33333333333354, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0,
0.0, 79.33333333333354, 0.0, 56.00000000000016, -1.63424829224823e-13, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 56.0, 23.33333333333354, 2.997602166487923e-15, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 79.33333333333354, 0.0, 0.0, 0.0], [0.0, 1.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0], [56.00000000000016, 79.3333333333336, 56.0,
23.33333333333354, 79.33333333333354, 56.0, 79.33333333333354, 79.33333333333354])
LP_M_terms_result_initiation: 574031.5100489089
LP_Cost_result_initiation: nope
LP_result_initiation: [15.0, 357.187857142919, 35.0, 56.60166577709346,
5498.351092436992, 84.86502565627981, 103.87423507069, 45.0, 9300.969658659924,
10.73137792783288, 60.0, 7.826801517066953, 12.91411690230003, 6264.737647058851,
48.48293299620721, 103.87423507069, 0.04500000006287506, 9300.969658659924,
3.874235070690009, 626.4737647058851, 0.9696586599241441, 2.830083288854679,
5498.351092437004, 42.43251282813999]
============================================================
============================================================
Iteration_1:
MILP_M_terms_result_iteration_1: 1967452.03936749
MILP_Cost_result_iteration_1: 215668.4887434267
MILP_result_iteration_1:([50.0, 5.443598106643506, 37.33333333333328,
18.66666666666664, -6.600575815789547e-17, 0.0, 0.0, 55.99999999999991, 0.0, 0.0,
0.0, 21.44359810664351, 34.0, -0.0, 28.55640189335649, 0.0, -0.0, 21.44359810664351,
0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 34.0, -0.0, 0.0, 0.0, -0.0, 0.0,
0.0, 0.0, 0.0, -0.0, 0.0, 18.66666666666664, -0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0,
0.0, -0.0, 0.0, 0.0, 21.44359810664351, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 0.0,
0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0,
-5.940518234210592e-16, 0.0, 0.0, 0.0], [0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0], [50.0, 34.0, 55.99999999999991, 18.66666666666664, 21.44359810664351, 0.0,
21.44359810664351, 34.0])
LP_M_terms_result_iteration_1: 1222.560757341683
LP_Cost_result_iteration_1: nope
LP_result_iteration_1:[15.0, 400.0, 35.00000000002001, 112.5984126000102, 12500.0,
164.3962960667073, 102.857142857143, 33.57142857142862, 9302.857142857158,
8.571428571428584, 25.71428571428575, 8.571428571428584, 52.30717186646719,
3235.345061851507, 57.38430311990032, -0.0, -0.0, -0.0, 15.69215155994016,
323.5345061851507, 1.147686062398006, 5.629920630000511, 12500.0, 82.19814803335365]
============================================================
============================================================
Iteration_2:
MILP_M_terms_result_iteration_2: 2572632.290550611
MILP_Cost_result_iteration_2: 42516.22748091602
MILP_result_iteration_2:([6.000000000000007, 0.0, 0.0, 9.083969465648853, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 15.08396946564886, 0.0, -0.0, 24.91603053435115, 0.0, -0.0,
0.0, 0.0, 25.08396946564885, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 34.0, 0.0, -0.0,
0.0, 0.0, -0.0, 0.0, 0.0, 56.00000000000001, 0.0, -0.0, 9.083969465648853, 0.0,
-0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 100.0, 0.0, 0.0,
43.99999999999999, 0.0, 56.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 100.0,
0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, -4.297240178680159e-15, 4.297240178680159e-
```

```
15, 0.0], [0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0], [50.0, 34.0, 56.0,
9.083969465648853, 100.0, 100.0, 115.0839694656489, 0.0])
LP_M_terms_result_iteration_2: 6317.929107125052
LP_Cost_result_iteration_2: nope
LP_result_iteration_2:[15.0, 400.0, 35.0, 115.6982487651549, 12500.0,
165.3547373147732, 139.6165496458307, 25.98727860623412, 9397.385192052127,
17.61344537815127, 52.84033613445379, 17.61344537815127, 39.61654964583074,
987.2786062341161, 97.3851920521284, 39.61654964583074, 0.9872786062341161,
97.38519205212651, 31.61654964583075, 379.278606234116, 92.5851920521284, -0.0,
-0.0, -0.0]
============================================================
============================================================
Iteration_3:
MILP_M_terms_result_iteration_3: 3013654.331966674
MILP_Cost_result_iteration_3: 41023.00000000003
MILP_result_iteration_3:([0.0, 0.0, 0.0, 0.0, 9.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
9.0, 0.0, -0.0, 49.99999999999978, 0.0, -0.0, 0.0, 1.110223024625157e-13,
1.110223024625157e-13, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 57.02253296582184,
13.20730519547253, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 12.36463948222401,
44.65789348359804, -0.0, 20.22983816129448, 0.0, -0.0, 0.0, 1.110223024625157e-13,
1.110223024625157e-13, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 25.57194467769688,
-0.0, 0.0, 57.02253296582206, -0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0,
16.57194467769688, 0.0, 0.0, 0.0, 50.0, 1.110223024625157e-13, 0.0,
20.2298381612947, 0.0, 0.0, 1.110223024625157e-13, 0.0], [0.0, 1.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0], [50.0, 70.22983816129438, 57.02253296582206, 20.2298381612947,
25.57194467769688, 57.02253296582207, 25.57194467769688, 70.22983816129492])
LP_M_terms_result_iteration_3: 273905.8101626547
LP_Cost_result_iteration_3: nope
LP_result_iteration_3:[15.0, 400.0, 35.0, 63.12428894890108, 6208.383810999264,
96.03605731990757, 161.3310803476012, 30.76008166067432, 9229.267657399037,
13.99954832411651, 59.99999999999999, 20.0, 52.78850843132859, 2586.366248238369,
77.25295980053286, 63.12428894890109, 6.208383810999264, 96.03605731990758,
33.18287792525349, 322.1356768948571, 90.24338683801403, 6.090439080419031,
961.302159226254, 2948.427415130997]
============================================================
============================================================
Iteration_4:
MILP_M_terms_result_iteration_4: 2306127.251878498
MILP_Cost_result_iteration_4: 153505.5199664673
MILP_result_iteration_4:([50.0, 9.466666666666654, 0.0, 9.0, 9.516019360349187, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 77.98268602701584, 0.0, 50.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 58.71617309547347, 9.750493571193186, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 9.12838281520707, 49.5877902802664, 0.0, 9.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 28.39489574674944, 0.0, 0.0,
58.71617309547347, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 18.87887638640025,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0], [0.0, 1.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0], [50.0, 68.46666666666665, 58.71617309547347, 9.0, 28.39489574674944,
58.71617309547347, 18.87887638640026, 77.98268602701584])
LP_M_terms_result_iteration_4: 4.672173758990539e-11
LP_Cost_result_iteration_4: nope
LP_result_iteration_4:[15.0, 400.0, 35.0, 62.95034079844207, 6357.546251217138,
94.93670886075951, 158.3244039359409, 30.20106203559186, 8964.724580648157,
17.77777777777778, 53.33333333333334, 17.77777777777778, 49.92843823260763,
2360.511926808087, 75.19599193224121, 62.95034079844208, 6.357546251217139,
94.93670886075951, 32.71978616576425, 329.8125345616247, 87.67377451985135,
5.942774357519282, 878.7090762351718, 2863.943828648552]
============================================================
MINLP function result:
```

```
# ==========================================================
# = Solver Results                                         =
# ==========================================================
# ----------------------------------------------------------
#   Problem Information
# ----------------------------------------------------------
Problem:
- Lower bound: -inf
  Upper bound: inf
  Number of objectives: 1
  Number of constraints: 736
  Number of variables: 438
  Sense: unknown
# ----------------------------------------------------------
#   Solver Information
# ----------------------------------------------------------
Solver:
- Status: ok
  Message: SCIP 8.0.3\x3a optimal solution; objective 862984.538918406; 226751
  simplex iterations; 7409 branching nodes
  Termination condition: optimal
  Id: 0
  Error rc: 0
  Time: 41.03666162490845
# ----------------------------------------------------------
#   Solution Information
# ----------------------------------------------------------
Solution:
- number of solutions: 0
  number of solutions displayed: 0
F_w_11= 50.0
F_w_12= 7.466666666666654
F_w_13= 0.0
F_w_14= 9.0
F_w_15= 8.999999999999998
F_out_11= 0.0
F_out_21= 0.0
F_out_31= 0.0
F_out_41= 0.0
F_out_51= 0.0
F_out_61= 0.0
F_out_71= 14.76311486825916
F_out_81= 60.70355179840744
F_ua_11= 0.0
F_ua_12= 18.54186910123076
F_ua_13= 0.0
F_ua_14= 0.0
F_ua_15= 0.0
F_ua_16= 31.45813089876924
F_ua_17= 0.0
F_ua_18= 0.0
F_ua_21= 0.0
F_ua_22= 0.0
F_ua_23= 0.0
F_ua_24= 0.0
F_ua_25= 0.0
F_ua_26= 35.00853576789747
F_ua_27= 0.0
F_ua_28= 0.0
F_ua_31= 0.0
F_ua_32= 0.0
```

```
F_ua_33= 0.0
F_ua_34= 0.0
F_ua_35= 0.0
F_ua_36= 0.0
F_ua_37= 55.99999994399175
F_ua_38= 0.0
F_ua_41= 0.0
F_ua_42= 9.0
F_ua_43= 0.0
F_ua_44= 0.0
F_ua_45= 0.0
F_ua_46= 0.0
F_ua_47= 0.0
F_ua_48= 0.0
F_ua_51= 0.0
F_ua_52= 0.0
F_ua_53= 0.0
F_ua_54= 0.0
F_ua_55= 0.0
F_ua_56= 8.999999999999998
F_ua_57= 0.0
F_ua_58= 0.0
F_ua_61= 0.0
F_ua_62= 0.0
F_ua_63= 55.99999994399175
F_ua_64= 0.0
F_ua_65= 0.0
F_ua_66= 0.0
F_ua_67= 0.0
F_ua_68= 19.46666672267497
F_ua_71= 0.0
F_ua_72= 0.0
F_ua_73= 0.0
F_ua_74= 0.0
F_ua_75= 0.0
F_ua_76= 0.0
F_ua_77= 0.0
F_ua_78= 41.23688507573259
F_ua_81= 0.0
F_ua_82= 0.0
F_ua_83= 0.0
F_ua_84= 0.0
F_ua_85= 0.0
F_ua_86= 0.0
F_ua_87= 0.0
F_ua_88= 0.0
F_t_1= 50.0
F_t_2= 35.00853576789741
F_t_3= 55.99999994399175
F_t_4= 9.0
F_t_5= 8.999999999999998
F_t_6= 75.46666666666671
F_t_7= 55.99999994399175
F_t_8= 60.70355179840755
M_in_11= 0.0
M_in_21= 0.0
M_in_31= 0.0
M_in_12= 438.1280365184612
M_in_22= 7896.747640492336
M_in_32= 808.9654185430763
M_in_13= 3791.872787213078
```

```
M_in_23= 368.115900687822
M_in_33= 5179.505295015057
M_in_14= 0.0
M_in_24= 0.0
M_in_34= 0.0
M_in_15= 0.0
M_in_25= 0.0
M_in_35= 0.0
M_in_16= 5110.0
M_in_26= 496079.9999999996
M_in_36= 6980.0
M_in_17= 9391.872787213078
M_in_27= 1768.115900687822
M_in_37= 525979.505295015
M_in_18= 3392.903529315861
M_in_28= 258.1633712731377
M_in_38= 9546.836288437175
M_out_11= 750.0
M_out_21= 20000.0
M_out_31= 1750.0
M_out_12= 3838.128036518461
M_out_22= 422696.7476404923
M_out_32= 5398.965418543076
M_out_13= 9391.872787213078
M_out_23= 1768.115900687822
M_out_33= 525979.505295015
M_out_14= 160.0
M_out_24= 480.0
M_out_34= 160.0
M_out_15= 800.0
M_out_25= 60800.0
M_out_35= 480.0
M_out_16= 5110.0
M_out_26= 496.08
M_out_36= 6980.0
M_out_17= 2817.561836163924
M_out_27= 176.8115900687822
M_out_37= 10519.59010590031
M_out_18= 169.6451764657932
M_out_28= 258.1633712731377
M_out_38= 4773.418144218587
C_out_11 =  15.0
C_out_21 =  400.0
C_out_31 =  35.0
C_out_12 =  109.6340635836035
C_out_22 =  12074.10531085684
C_out_32 =  154.2185441271606
C_out_13 =  167.7120142198092
C_out_23 =  31.57349825676108
C_out_33 =  9392.491175387318
C_out_14 =  17.77777777777778
C_out_24 =  53.33333333333333
C_out_34 =  17.77777777777778
C_out_15 =  88.88888888888889
C_out_25 =  6755.555555555555
C_out_35 =  53.33333333333333
C_out_16 =  67.71201413427816
C_out_26 =  6.573498233394419
C_out_36 =  92.49116608584589
C_out_17 =  50.3136042703624
C_out_27 =  3.157349825673157
```

```
C_out_37 =  187.8498235068732
C_out_18 =  2.794649932362296
C_out_28 =  4.252854479251065
C_out_38 =  78.63490689822592
B_fw_11= 1.0
B_fw_12= 1.0
B_fw_13= 0.0
B_fw_14= 1.0
B_fw_15= 1.0
B_ua_11= 0.0
B_ua_12= 1.0
B_ua_13= 0.0
B_ua_14= 0.0
B_ua_15= 0.0
B_ua_16= 1.0
B_ua_17= 0.0
B_ua_18= 0.0
B_ua_21= 0.0
B_ua_22= 0.0
B_ua_23= 0.0
B_ua_24= 0.0
B_ua_25= 0.0
B_ua_26= 1.0
B_ua_27= 0.0
B_ua_28= 0.0
B_ua_31= 0.0
B_ua_32= 0.0
B_ua_33= 0.0
B_ua_34= 0.0
B_ua_35= 0.0
B_ua_36= 0.0
B_ua_37= 1.0
B_ua_38= 0.0
B_ua_41= 0.0
B_ua_42= 1.0
B_ua_43= 0.0
B_ua_44= 0.0
B_ua_45= 0.0
B_ua_46= 0.0
B_ua_47= 0.0
B_ua_48= 0.0
B_ua_51= 0.0
B_ua_52= 0.0
B_ua_53= 0.0
B_ua_54= 0.0
B_ua_55= 0.0
B_ua_56= 1.0
B_ua_57= 0.0
B_ua_58= 0.0
B_ua_61= 0.0
B_ua_62= 0.0
B_ua_63= 1.0
B_ua_64= 0.0
B_ua_65= 0.0
B_ua_66= 0.0
B_ua_67= 0.0
B_ua_68= 1.0
B_ua_71= 0.0
B_ua_72= 0.0
B_ua_73= 0.0
B_ua_74= 0.0
```

```
B_ua_75= 0.0
B_ua_76= 0.0
B_ua_77= 0.0
B_ua_78= 1.0
B_ua_81= 0.0
B_ua_82= 0.0
B_ua_83= 0.0
B_ua_84= 0.0
B_ua_85= 0.0
B_ua_86= 0.0
B_ua_87= 0.0
B_ua_88= 0.0
B_out_11= 0.0
B_out_21= 0.0
B_out_31= 0.0
B_out_41= 0.0
B_out_51= 0.0
B_out_61= 0.0
B_out_71= 1.0
B_out_81= 1.0
B_G1_16= 1.0
B_G1_17= 1.0
B_G1_18= 1.0
B_G1_26= 1.0
B_G1_27= 1.0
B_G1_28= 1.0
B_G1_36= 0.0
B_G1_37= 1.0
B_G1_38= 0.0
B_G1_46= 1.0
B_G1_47= 1.0
B_G1_48= 1.0
B_G1_56= 1.0
B_G1_57= 0.0
B_G1_58= 1.0
B_G1_66= 0.0
B_G1_67= 0.0
B_G1_68= 1.0
B_G1_76= 0.0
B_G1_77= 0.0
B_G1_78= 1.0
B_G1_86= 0.0
B_G1_87= 0.0
B_G1_88= 0.0
B_G2_16= 0.0
B_G2_17= 0.0
B_G2_18= 0.0
B_G2_26= 0.0
B_G2_27= 0.0
B_G2_28= 0.0
B_G2_36= 1.0
B_G2_37= 0.0
B_G2_38= 0.0
B_G2_46= 0.0
B_G2_47= 0.0
B_G2_48= 0.0
B_G2_56= 0.0
B_G2_57= 1.0
B_G2_58= 0.0
B_G2_66= 1.0
B_G2_67= 0.0
```

```
B_G2_68= 0.0
B_G2_76= 0.0
B_G2_77= 1.0
B_G2_78= 0.0
B_G2_86= 1.0
B_G2_87= 1.0
B_G2_88= 1.0
Cost_fw_1= 129802.6666666666
Cost_tu_6= 683666.0971716935
Cost_tu_7= 24317.96243023719
Cost_tu_8= 8501.412649809148
Cost_fw_s_sum= 129802.6666666666
Cost_fwpipe_su_sum= 2429.7
Cost_uapipe_ua_sum= 11774.7
Cost_outpipe_ue_sum= 2492.0
Cost_tu_u_sum= 716485.4722517398
O_cost.value= 862984.5389184065
Freshwater use compared to linear system: -51.62393162393163 %
Wastewater produced compared to linear system: -51.623931623931675 %
```

In this run, the initial value of $C_{c,tu}^{out}$ was set to C_c^{env}.

```
============================================================
Initiation:
MILP_M_terms_result_initiation: 2703483.26923077
MILP_Cost_result_initiation: 117553.4
MILP_result_initiation: ([50.0, 0.0, -0.0, 9.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
59.0, 0.0, -0.0, 36.30769230769231, 0.0, -0.0, 13.69230769230769, 0.0, -0.0, 0.0,
-0.0, 0.0, 0.0, -0.0, 0.0, 56.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, -0.0, 56.0,
0.0, -0.0, 9.0, -0.0, -0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0,
13.69230769230769, 0.0, -0.0, 0.0, 56.0, -0.0, -0.0, 0.0, 0.0, 0.0, -0.0,
10.69230769230769, 0.0, -0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0,
0.0, 0.0], [0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 1.0, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0], [50.0, 56.0, 56.0, 9.0,
13.69230769230769, 56.0, 69.6923076923077, 0.0])
LP_M_terms_result_initiation: -5.32054400537163e-11
LP_Cost_result_initiation: nope
LP_result_initiation: [15.0, 400.0, 35.0, 82.523968135301, 7693.714240583447,
136.521627079769, 182.523968135301, 32.69371424058345, 9436.521627079768,
17.77777777777778, 53.33333333333334, 17.77777777777778, 73.42696629213484,
4840.449438202248, 70.0561797752809, 82.523968135301, 7.693714240583447,
136.521627079769, 48.32696980215203, 97.72638233632945, 151.926219525697, -0.0,
-0.0, -0.0]
============================================================
============================================================
Iteration_1:
MILP_M_terms_result_iteration_1: 2604064.891594192
MILP_Cost_result_iteration_1: 46809.22355212356
MILP_result_iteration_1:([0.0, 0.0, 0.0, 0.0, 17.18146718146718, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 17.18146718146718, 0.0, -0.0, 50.0, 0.0, -0.0, 0.0, 0.0, 0.0, -0.0, -0.0,
0.0, 0.0, -0.0, 0.0, 70.27027027027026, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0,
70.27027027027026, 0.0, -0.0, 20.27027027027027, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0,
-0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 17.18146718146718, 0.0, -0.0, 0.0,
70.27027027027026, -0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 0.0,
70.27027027027026, 50.0, 0.0, 0.0, 20.27027027027027, 0.0, 0.0, -0.0, 0.0], [0.0,
1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0], [50.0, 70.27027027027026, 70.27027027027026,
20.27027027027027, 17.18146718146718, 70.27027027027026, 87.45173745173744,
70.27027027027026])
LP_M_terms_result_iteration_1: 8386.407043869527
LP_Cost_result_iteration_1: nope
LP_result_iteration_1:[15.0, 400.0, 35.0, 61.86637187368206, 6204.846153846154,
95.99230769230769, 141.5586795659898, 26.12792307692308, 7507.376923076925,
9.736755828764467, 60.0, 20.0, 46.56179775280899, 3538.696629213483,
27.9370786516854, 61.86637187368207, 6.204846153846154, 95.9923076923077,
36.86844990862271, 71.62352406181016, 120.7581280353201, 1.843422495431135,
71.62352406181017, 60.37906401766006]
============================================================
============================================================
Iteration_2:
MILP_M_terms_result_iteration_2: 2516958.933026464
MILP_Cost_result_iteration_2: 120419.2
MILP_result_iteration_2:([50.0, 0.0, -0.0, 9.0, -0.0, -0.0, -0.0, 0.0, -0.0, -0.0,
0.0, 59.00000000000003, -2.842170943040401e-14, -0.0, 50.00000000000001, 0.0, -0.0,
0.0, 0.0, -0.0, -7.105427357601002e-15, -0.0, 0.0, 0.0, -0.0, -0.0,
70.81367974756984, 11.69805174830278, -0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 0.0,
47.30194825169726, 23.51173149587257, -0.0, -1.998401444325282e-15, -0.0, -0.0,
9.000000000000002, 0.0, 0.0, 0.0, -0.0, -0.0, 0.0, -0.0, 0.0, 0.0, 1.77635683940025e-
15, 9.0, -0.0, 0.0, 70.81367974756984, -0.0, 0.0, 0.0, -0.0, -0.0, -0.0,
7.305018013713388e-15, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 32.5117314958726, 0.0,
-0.0, -8.881784197001252e-16, 0.0, -2.111259171634428e-15, 0.0], [0.0, 1.0, 0.0,
```

```
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
-2.220446049250313e-16, -0.0, 0.0, 0.9999999999999998, 0.0, -0.0, 0.0, 0.0, 0.0],
[50.0, 82.51173149587261, 70.81367974756984, 9.0, 9.000000000000002,
70.81367974756984, 59.00000000000004, 32.51173149587257])
LP_M_terms_result_iteration_2: 80382.59033868022
LP_Cost_result_iteration_2: nope
LP_result_iteration_2:[15.0, 400.0, 35.0, 52.75595850247652, 6019.584770399864,
180.0, 131.8367239698604, 25.78977613724583, 7534.511188466711, 17.77777777777778,
53.33333333333334, 17.77777777777778, 106.6666666666667, 6808.888888888889,
71.11111111111111, 52.75595850247653, 6.019584770399864, 180.0000000000141,
34.84716522451989, 121.419187802881, 121.5263415774173, 6.243453468158431,
1903.508655013783, 2734.234626021807]
============================================================
============================================================
Iteration_3:
MILP_M_terms_result_iteration_3: 2211883.5507941
MILP_Cost_result_iteration_3: 141121.4625332633
MILP_result_iteration_3:([50.0, 0.0, 0.0, 17.99364100771126, 0.0, 0.0, -0.0, -0.0,
-0.0, -0.0, -0.0, 35.48190951183844, 32.51173149587282, -0.0, 32.0, 0.0, -0.0, 18.0,
0.0, -0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, 60.57596962039619, 0.0, -0.0, -0.0, 0.0,
0.0, -0.0, 0.0, -0.0, 37.06423812452337, 23.51173149587282, -0.0, 17.99364100771126,
0.0, -0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, -0.0, 9.0, 9.0, -0.0,
-0.0, 60.57596962039619, -0.0, -0.0, 0.0, -0.0, 0.0, -0.0, 10.58232861268493, 0.0,
-0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, -0.0, 0.0, -0.0, -0.0, 0.0], [0.0, 1.0,
0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0], [50.0, 60.57596962039619, 60.57596962039619,
17.99364100771126, 18.0, 60.57596962039619, 46.06423812452337, 32.51173149587282])
LP_M_terms_result_iteration_3: -7.73070496506989e-11
LP_Cost_result_iteration_3: nope
LP_result_iteration_3:[15.0, 400.0, 35.0, 74.33477161681057, 7080.14731716175,
121.4564322557995, 166.7806711227173, 30.19162219363841, 8718.925086305118,
8.892030241763255, 26.67609072528977, 8.892030241763255, 59.44444444444444,
3777.777777777777, 61.66666666666666, 74.33477161681058, 7.08014731716175,
121.4564322557995, 43.74281730810478, 76.23924958753247, 140.5495146641125,
6.853375924786564, 1067.610235365283, 3161.197760075149]
============================================================
MINLP function result:
# ============================================================
# = Solver Results                                           =
# ============================================================
# ------------------------------------------------------------
#   Problem Information
# ------------------------------------------------------------
Problem:
- Lower bound: -inf
  Upper bound: inf
  Number of objectives: 1
  Number of constraints: 736
  Number of variables: 438
  Sense: unknown
# ------------------------------------------------------------
#   Solver Information
# ------------------------------------------------------------
Solver:
- Status: ok
  Message: SCIP 8.0.3\x3a optimal solution; objective 710237.448523624; 2.47683e+07
  simplex iterations; 3.51496e+06 branching nodes
  Termination condition: optimal
  Id: 0
  Error rc: 0
  Time: 4290.286500692368
```

```
# ------------------------------------------------------------
#   Solution Information
# ------------------------------------------------------------
Solution:
- number of solutions: 0
  number of solutions displayed: 0
F_w_11= 50.0
F_w_12= 0.0
F_w_13= 0.0
F_w_14= 18.1034279100078
F_w_15= 0.0
F_out_11= 0.0
F_out_21= 0.0
F_out_31= 0.0
F_out_41= 0.0
F_out_51= 0.0
F_out_61= 0.0
F_out_71= 19.7755537451885
F_out_81= 48.32787416481929
F_ua_11= 0.0
F_ua_12= 24.89657212888442
F_ua_13= 0.0
F_ua_14= 0.0
F_ua_15= 25.10342787111558
F_ua_16= 0.0
F_ua_17= 0.0
F_ua_18= 0.0
F_ua_21= 0.0
F_ua_22= 0.0
F_ua_23= 0.0
F_ua_24= 0.0
F_ua_25= 0.0
F_ua_26= 34.00000003889221
F_ua_27= 0.0
F_ua_28= 0.0
F_ua_31= 0.0
F_ua_32= 0.0
F_ua_33= 0.0
F_ua_34= 0.0
F_ua_35= 0.0
F_ua_36= 0.0
F_ua_37= 59.1034279100078
F_ua_38= 0.0
F_ua_41= 0.0
F_ua_42= 9.103427910007795
F_ua_43= 0.0
F_ua_44= 0.0
F_ua_45= 0.0
F_ua_46= 0.0
F_ua_47= 9.0
F_ua_48= 0.0
F_ua_51= 0.0
F_ua_52= 0.0
F_ua_53= 0.0
F_ua_54= 0.0
F_ua_55= 0.0
F_ua_56= 25.10342787111558
F_ua_57= 0.0
F_ua_58= 0.0
F_ua_61= 0.0
F_ua_62= 0.0
```

F_ua_63= 59.1034279100078
F_ua_64= 0.0
F_ua_65= 0.0
F_ua_66= 0.0
F_ua_67= 0.0
F_ua_68= 0.0
F_ua_71= 0.0
F_ua_72= 0.0
F_ua_73= 0.0
F_ua_74= 0.0
F_ua_75= 0.0
F_ua_76= 0.0
F_ua_77= 0.0
F_ua_78= 48.32787416481929
F_ua_81= 0.0
F_ua_82= 0.0
F_ua_83= 0.0
F_ua_84= 0.0
F_ua_85= 0.0
F_ua_86= 0.0
F_ua_87= 0.0
F_ua_88= 0.0
F_t_1= 50.0
F_t_2= 34.00000003889221
F_t_3= 59.1034279100078
F_t_4= 18.1034279100078
F_t_5= 25.10342787111558
F_t_6= 59.1034279100078
F_t_7= 68.1034279100078
F_t_8= 48.32787416481929
M_in_11= 0.0
M_in_21= 0.0
M_in_31= 0.0
M_in_12= 453.9056353045642
M_in_22= 10200.00001166766
M_in_32= 951.8370778822525
M_in_13= 5030.45705336668
M_in_23= 495.8413700580709
M_in_33= 6900.45705331536
M_in_14= 0.0
M_in_24= 0.0
M_in_34= 0.0
M_in_15= 376.5514180667337
M_in_25= 10041.37114844623
M_in_35= 878.6199754890454
M_in_16= 5030.45705336764
M_in_26= 495841.3711601192
M_in_36= 6900.457053317297
M_in_17= 10709.99999999267
M_in_27= 2134.470209113629
M_in_37= 527779.9999999393
M_in_18= 2280.02414058468
M_in_28= 151.4672759404811
M_in_38= 7490.514415558168
M_out_11= 750.0
M_out_21= 20000.0
M_out_31= 1750.0
M_out_12= 3853.905635304564
M_out_22= 425000.0000116677
M_out_32= 5541.837077882253
M_out_13= 10630.45705336668

```
M_out_23= 1895.841370058071
M_out_33= 527700.4570533154
M_out_14= 160.0
M_out_24= 480.0
M_out_34= 160.0
M_out_15= 1176.551418066734
M_out_25= 70841.37114844623
M_out_35= 1358.619975489045
M_out_16= 5030.45705336764
M_out_26= 495.8413711601197
M_out_36= 6900.457053317297
M_out_17= 3212.999999997802
M_out_27= 213.4470209113628
M_out_37= 10555.5999999988
M_out_18= 114.0012070292341
M_out_28= 151.4672759404811
M_out_38= 3745.257207779084
C_out_11 =  15.0
C_out_21 =  400.0
C_out_31 =  35.0
C_out_12 =  113.350165614486
C_out_22 =  12499.99998604519
C_out_32 =  162.9952079849959
C_out_13 =  179.8619374421528
C_out_23 =  32.07667365962596
C_out_33 =  8928.423878506735
C_out_14 =  8.838105180707084
C_out_24 =  26.51431554212124
C_out_34 =  8.838105180707084
C_out_15 =  46.8681577714131
C_out_25 =  2821.979990627395
C_out_35 =  54.12089466284786
C_out_16 =  85.11277993937063
C_out_26 =  8.389384307440977
C_out_36 =  116.7522307474713
C_out_17 =  47.17824195638792
C_out_27 =  3.134159719466811
C_out_37 =  154.9936666029044
C_out_18 =  2.358912082685502
C_out_28 =  3.134159688656333
C_out_38 =  77.49683329730513
B_fw_11= 1.0
B_fw_12= 0.0
B_fw_13= 0.0
B_fw_14= 1.0
B_fw_15= 0.0
B_ua_11= 0.0
B_ua_12= 1.0
B_ua_13= 0.0
B_ua_14= 0.0
B_ua_15= 1.0
B_ua_16= 0.0
B_ua_17= 0.0
B_ua_18= 0.0
B_ua_21= 0.0
B_ua_22= 0.0
B_ua_23= 0.0
B_ua_24= 0.0
B_ua_25= 0.0
B_ua_26= 1.0
B_ua_27= 0.0
```

```
B_ua_28= 0.0
B_ua_31= 0.0
B_ua_32= 0.0
B_ua_33= 0.0
B_ua_34= 0.0
B_ua_35= 0.0
B_ua_36= 0.0
B_ua_37= 1.0
B_ua_38= 0.0
B_ua_41= 0.0
B_ua_42= 1.0
B_ua_43= 0.0
B_ua_44= 0.0
B_ua_45= 0.0
B_ua_46= 0.0
B_ua_47= 1.0
B_ua_48= 0.0
B_ua_51= 0.0
B_ua_52= 0.0
B_ua_53= 0.0
B_ua_54= 0.0
B_ua_55= 0.0
B_ua_56= 1.0
B_ua_57= 0.0
B_ua_58= 0.0
B_ua_61= 0.0
B_ua_62= 0.0
B_ua_63= 1.0
B_ua_64= 0.0
B_ua_65= 0.0
B_ua_66= 0.0
B_ua_67= 0.0
B_ua_68= 0.0
B_ua_71= 0.0
B_ua_72= 0.0
B_ua_73= 0.0
B_ua_74= 0.0
B_ua_75= 0.0
B_ua_76= 0.0
B_ua_77= 0.0
B_ua_78= 1.0
B_ua_81= 0.0
B_ua_82= 0.0
B_ua_83= 0.0
B_ua_84= 0.0
B_ua_85= 0.0
B_ua_86= 0.0
B_ua_87= 0.0
B_ua_88= 0.0
B_out_11= 0.0
B_out_21= 0.0
B_out_31= 0.0
B_out_41= 0.0
B_out_51= 0.0
B_out_61= 0.0
B_out_71= 1.0
B_out_81= 1.0
B_G1_16= 0.0
B_G1_17= 1.0
B_G1_18= 0.0
B_G1_26= 1.0
```

```
B_G1_27= 1.0
B_G1_28= 1.0
B_G1_36= 0.0
B_G1_37= 1.0
B_G1_38= 1.0
B_G1_46= 1.0
B_G1_47= 1.0
B_G1_48= 1.0
B_G1_56= 1.0
B_G1_57= 0.0
B_G1_58= 1.0
B_G1_66= 0.0
B_G1_67= 1.0
B_G1_68= 1.0
B_G1_76= 0.0
B_G1_77= 0.0
B_G1_78= 1.0
B_G1_86= 0.0
B_G1_87= 0.0
B_G1_88= 1.0
B_G2_16= 0.0
B_G2_17= 0.0
B_G2_18= 0.0
B_G2_26= 0.0
B_G2_27= 0.0
B_G2_28= 0.0
B_G2_36= 1.0
B_G2_37= 0.0
B_G2_38= 0.0
B_G2_46= 0.0
B_G2_47= 0.0
B_G2_48= 0.0
B_G2_56= 0.0
B_G2_57= 0.0
B_G2_58= 0.0
B_G2_66= 1.0
B_G2_67= 0.0
B_G2_68= 0.0
B_G2_76= 1.0
B_G2_77= 1.0
B_G2_78= 0.0
B_G2_86= 1.0
B_G2_87= 1.0
B_G2_88= 0.0
Cost_fw_1= 117137.8960052134
Cost_tu_6= 537493.1968883786
Cost_tu_7= 28111.49920851669
Cost_tu_8= 7247.356421515748
Cost_fw_s_sum= 117137.8960052134
Cost_fwpipe_su_sum= 996.8
Cost_uapipe_ua_sum= 16758.70000000002
Cost_outpipe_ue_sum= 2492.0
Cost_tu_u_sum= 572852.052518411
O_cost.value= 710237.4485236243
Freshwater use compared to linear system: -56.34395646794373 %
Wastewater produced compared to linear system: -56.34395646794373 %
```

G.5. Water Reuse Case Study

In this run, the initial value of $C_{c,tu}^{out}$ was set to 0.

```
============================================================
Initiation:
MILP_M_terms_result_initiation: 192000.0
MILP_Cost_result_initiation: 451500.0
MILP_result_initiation: ([20.0, 30.0, 20.0, 0.0, 0.0, 0.0, 40.0, 30.0, -0.0, 20.0,
0.0, 0.0, -0.0, 0.0, 20.0, 30.0, -0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 0.0], [0.0,
1.0, 0.0, 0.0, 0.0, 0.0, 0.9999999999994998, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0], [20.0, 50.0, 40.0, 30.0])
LP_M_terms_result_initiation: 1999.999999999998
LP_Cost_result_initiation: nope
LP_result_initiation: [-0.0, 100.0, 800.0, 233.3333333333333]
============================================================
MINLP function result:
# ============================================================
# = Solver Results                                           =
# ============================================================
# ------------------------------------------------------------
#   Problem Information
# ------------------------------------------------------------
Problem:
- Lower bound: -inf
  Upper bound: inf
  Number of objectives: 1
  Number of constraints: 109
  Number of variables: 68
  Sense: unknown
# ------------------------------------------------------------
#   Solver Information
# ------------------------------------------------------------
Solver:
- Status: ok
  Message: SCIP 8.0.3\x3a optimal solution; objective 580500; 403 simplex
  iterations; 7 branching nodes
  Termination condition: optimal
  Id: 0
  Error rc: 0
  Time: 0.16827082633972168
# ------------------------------------------------------------
#   Solution Information
# ------------------------------------------------------------
Solution:
- number of solutions: 0
  number of solutions displayed: 0
Freshwater use compared to linear system: -25.0%
Wastewater produced compared to linear system: -25.0%
F_w_11= 20.0
F_w_12= 50.0
F_w_13= 20.0
F_w_14= 0.0
F_out_11= 0.0
F_out_21= 30.0
F_out_31= 40.0
F_out_41= 20.0
F_ua_11= -0.0
F_ua_12= 0.0
F_ua_13= 0.0
F_ua_14= 20.0
```

```
F_ua_21= -0.0
F_ua_22= 0.0
F_ua_23= 20.0
F_ua_24= 0.0
F_ua_31= -0.0
F_ua_32= 0.0
F_ua_33= 0.0
F_ua_34= 0.0
F_ua_41= -0.0
F_ua_42= 0.0
F_ua_43= 0.0
F_ua_44= 0.0
F_t_1= 20.0
F_t_2= 50.0
F_t_3= 40.0
F_t_4= 20.0
M_in_11= 0.0
M_in_12= 0.0
M_in_13= 2000.0
M_in_14= 2000.0
M_out_11= 2000.0
M_out_12= 5000.0
M_out_13= 32000.0
M_out_14= 6000.0
C_out_11 =  100.0
C_out_12 =  100.0
C_out_13 =  800.0
C_out_14 =  300.0
B_fw_11= 1.0
B_fw_12= 1.0
B_fw_13= 1.0
B_fw_14= 0.0
B_ua_11= 0.0
B_ua_12= 0.0
B_ua_13= 0.0
B_ua_14= 1.0
B_ua_21= 0.0
B_ua_22= 0.0
B_ua_23= 1.0
B_ua_24= 0.0
B_ua_31= 0.0
B_ua_32= 0.0
B_ua_33= 0.0
B_ua_34= 0.0
B_ua_41= 0.0
B_ua_42= 0.0
B_ua_43= 0.0
B_ua_44= 0.0
B_out_11= 0.0
B_out_21= 1.0
B_out_31= 1.0
B_out_41= 1.0
Cost_fw_1= 580500.0
Cost_fw_s_sum= 580500.0
Cost_tu_u_sum= 0.0
O_cost.value= 580500.0
```

G.6. Water Treatment Case Study

In this run, the initial value of $C_{c,tu}^{out}$ was set to 0.

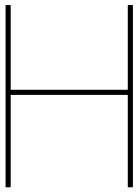
```
============================================================
Initiation:
MILP_M_terms_result_initiation: 558024.7715
MILP_Cost_result_initiation: 183182.8
MILP_result_initiation: ([13.09999999999999, 32.7, 56.5, 0.0, 0.0, 0.0, 0.0, 0.0,
102.3, 0.0, 0.0, 0.0, 13.09999999999999, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 32.7,
0.0, 0.0, 0.0, 56.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 69.6, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 69.6, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0], [13.09999999999999, 32.7, 56.5, 69.6, 69.6, 102.3])
LP_M_terms_result_initiation: -2.310179825215641e-10
LP_Cost_result_initiation: nope
LP_result_initiation: [390.0, 10.0, 25.0, 16780.0, 110.0, 40.0, 25.0, 100.0, 35.0,
0.09369971264369259, 83.06034482758622, 33.11781609195403, 0.009369971264369259,
8.306034482758623, 0.9935344827586214, 5363.701389540568, 2.040615835777126,
10.76950146627566]
============================================================
============================================================
Iteration_1:
MILP_M_terms_result_iteration_1: -5.557421189905653e-12
MILP_Cost_result_iteration_1: 183182.8
MILP_result_iteration_1:([13.09999999999999, 32.7, 56.5, 0.0, 0.0, 0.0,
2.131628207280301e-13, 0.0, 102.2999999999998, 0.0, 0.0, 0.0, 13.09999999999999,
0.0, 0.0, 0.0, 0.0, 0.0, 6.394884621840902e-14, 0.0, 32.69999999999994, 0.0, 0.0,
0.0, 56.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 69.59999999999984, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 69.59999999999984, 0.0, 0.0, 0.0, -1.154631945610163e-14, 0.0, 0.0], [0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0], [13.09999999999999, 32.7, 56.5,
69.60000000000004, 69.59999999999984, 102.2999999999998])
LP_M_terms_result_iteration_1: -1.282537520630456e-09
LP_Cost_result_iteration_1: nope
LP_result_iteration_1:[390.0, 10.0, 25.0, 16780.0, 110.0, 40.0, 25.0, 100.0, 35.0,
0.09369971264369259, 83.06034482758622, 33.11781609195403, 0.009369971264369928,
8.30603448275864, 0.9935344827586237, 5363.701389540579, 2.040615835777713,
10.76950146627568]
============================================================
MINLP function result:
WARNING (W1001): Setting Var 'F_ua_64' to a value `-1.154631945610163e-14`
(float) not in domain NonNegativeReals.
    See also https://pyomo.readthedocs.io/en/stable/errors.html#w1001
# ==========================================================
# = Solver Results                                         =
# ==========================================================
# ----------------------------------------------------------
#   Problem Information
# ----------------------------------------------------------
Problem:
- Lower bound: -inf
  Upper bound: inf
  Number of objectives: 1
  Number of constraints: 476
  Number of variables: 274
  Sense: unknown
# ----------------------------------------------------------
#   Solver Information
# ----------------------------------------------------------
Solver:
- Status: ok
  Message: SCIP 8.0.3\x3a optimal solution; objective 857792.33591139; 2127 simplex
```

```
  iterations; 40 branching nodes
  Termination condition: optimal
  Id: 0
  Error rc: 0
  Time: 0.6965007781982422
# ----------------------------------------------------------
#   Solution Information
# ----------------------------------------------------------
Solution:
- number of solutions: 0
  number of solutions displayed: 0
F_w_11= 13.1
F_w_12= 32.69999999999993
F_w_13= 56.49999999995999
F_out_11= 0.0
F_out_21= 0.0
F_out_31= 0.0
F_out_41= 0.0
F_out_51= 18.10927366404819
F_out_61= 84.19072633591172
F_ua_11= 0.0
F_ua_12= 0.0
F_ua_13= 0.0
F_ua_14= 13.1
F_ua_15= -2.332037696853662e-15
F_ua_16= 0.0
F_ua_21= 0.0
F_ua_22= 0.0
F_ua_23= 0.0
F_ua_24= 32.70000000000001
F_ua_25= -7.631516925484404e-14
F_ua_26= 0.0
F_ua_31= 0.0
F_ua_32= 0.0
F_ua_33= 0.0
F_ua_34= 0.0
F_ua_35= 56.50000000000143
F_ua_36= -4.143657861277461e-11
F_ua_41= 0.0
F_ua_42= 0.0
F_ua_43= 0.0
F_ua_44= 0.0
F_ua_45= 45.80000000000001
F_ua_46= 0.0
F_ua_51= 0.0
F_ua_52= 0.0
F_ua_53= 0.0
F_ua_54= 0.0
F_ua_55= 0.0
F_ua_56= 84.19072633595316
F_ua_61= 0.0
F_ua_62= 0.0
F_ua_63= 0.0
F_ua_64= 0.0
F_ua_65= 0.0
F_ua_66= 0.0
F_t_1= 13.1
F_t_2= 32.69999999999993
F_t_3= 56.49999999995999
F_t_4= 45.80000000000001
F_t_5= 102.3000000000014
```

```
F_t_6= 84.19072633591172
M_in_11= 0.0
M_in_21= 0.0
M_in_31= 0.0
M_in_12= 0.0
M_in_22= 0.0
M_in_32= 0.0
M_in_13= 0.0
M_in_23= 0.0
M_in_33= 0.0
M_in_14= 553815.0000000014
M_in_24= 3728.000000000009
M_in_34= 1635.5
M_in_15= 1966.315000000038
M_in_25= 9378.000000004296
M_in_35= 3613.000000001452
M_in_16= 161.8235464861896
M_in_26= 771.7894736847556
M_in_36= 89.20266693556846
M_out_11= 5109.0
M_out_21= 131.0
M_out_31= 327.5
M_out_12= 548706.0
M_out_22= 3597.0
M_out_32= 1308.0
M_out_13= 1412.5
M_out_23= 5650.0
M_out_33= 1977.5
M_out_14= 553.8150000000019
M_out_24= 3728.000000000009
M_out_34= 1635.5
M_out_15= 196.6315000000037
M_out_25= 937.8000000004295
M_out_35= 108.3900000000436
M_out_16= 161.8235464861896
M_out_26= 38.58947368423782
M_out_36= 71.36213354845476
C_out_11 =  390.0000000000001
C_out_21 =  10.0
C_out_31 =  25.0
C_out_12 =  16780.00000000004
C_out_22 =  110.0000000000002
C_out_32 =  40.0
C_out_13 =  25.0
C_out_23 =  100.0000000000708
C_out_33 =  35.00000000002479
C_out_14 =  12.09203056768563
C_out_24 =  81.39737991266394
C_out_34 =  35.70960698689986
C_out_15 =  1.922106549364624
C_out_25 =  9.16715542522388
C_out_35 =  1.059530791789249
C_out_16 =  1.922106549396267
C_out_26 =  0.4583577712557989
C_out_36 =  0.8476246334173136
B_fw_11= 1.0
B_fw_12= 1.0
B_fw_13= 1.0
B_ua_11= 0.0
B_ua_12= 0.0
B_ua_13= 0.0
```

```
B_ua_14= 1.0
B_ua_15= 0.0
B_ua_16= 0.0
B_ua_21= 0.0
B_ua_22= 0.0
B_ua_23= 0.0
B_ua_24= 1.0
B_ua_25= -2.199284416566111e-15
B_ua_26= 0.0
B_ua_31= 0.0
B_ua_32= 0.0
B_ua_33= 0.0
B_ua_34= 0.0
B_ua_35= 1.0
B_ua_36= 0.0
B_ua_41= 0.0
B_ua_42= 0.0
B_ua_43= 0.0
B_ua_44= 0.0
B_ua_45= 1.0
B_ua_46= 0.0
B_ua_51= 0.0
B_ua_52= 0.0
B_ua_53= 0.0
B_ua_54= 0.0
B_ua_55= 0.0
B_ua_56= 1.0
B_ua_61= 0.0
B_ua_62= 0.0
B_ua_63= 0.0
B_ua_64= 0.0
B_ua_65= 0.0
B_ua_66= 0.0
B_out_11= 0.0
B_out_21= 0.0
B_out_31= 0.0
B_out_41= 0.0
B_out_51= 1.0
B_out_61= 1.0
B_G1_14= 1.0
B_G1_15= 1.0
B_G1_16= 1.0
B_G1_24= 1.0
B_G1_25= 1.0
B_G1_26= 1.0
B_G1_34= 1.0
B_G1_35= 1.0
B_G1_36= 1.0
B_G1_44= 0.0
B_G1_45= 1.0
B_G1_46= 1.0
B_G1_54= 0.0
B_G1_55= 0.0
B_G1_56= 1.0
B_G1_64= 0.0
B_G1_65= 0.0
B_G1_66= 0.0
B_G2_14= 0.0
B_G2_15= 0.0
B_G2_16= 0.0
B_G2_24= 0.0
```

```
B_G2_25= 0.0
B_G2_26= 0.0
B_G2_34= 0.0
B_G2_35= 0.0
B_G2_36= 0.0
B_G2_44= 1.0
B_G2_45= 0.0
B_G2_46= 0.0
B_G2_54= 1.0
B_G2_55= 0.0
B_G2_56= 0.0
B_G2_64= 0.0
B_G2_65= 1.0
B_G2_66= 0.0
Cost_fw_1= 175955.9999999311
Cost_tu_4= 491598.1209826199
Cost_tu_5= 134524.8844103294
Cost_tu_6= 42754.93051850951
Cost_fw_s_sum= 175955.9999999311
Cost_fwpipe_su_sum= 0.0
Cost_uapipe_ua_sum= 7725.199999999986
Cost_outpipe_ue_sum= 5233.2
Cost_tu_u_sum= 668877.9359114589
O_cost.value= 857792.3359113899
Freshwater use compared to linear system: -3.918750845707219e-11 %
Wastewater produced compared to linear system: -3.920139981065498e-11 %
```



Cost Equations

Table H.1

Treatment technology	Unit(s)	Cost equation(s)	Applicable range of x		Reference
			Min	Max	
Biological					
Conventional activated sludge system	$x = p.e.$, $y = 10^6\$10^{-3}p.e$	$CAPEX: y = 0.116x^{0.954}$, $OPEX: y = 0.022x^{0.672}$	40000	180000	(Tsagarakis et al., 2003)
Secondary treatment	$x = p.e.$, $y = €/m^3$	$CAPEX\&OPEX: y = 0.474–7 \cdot 10^{-6}x$	0	30000	(Sipala et al., 2003)
Biological treatment unit	$x = ton/h$, $y = \$(CAPEX)$, $y = \$/h(OPEX)$	$CAPEX: y = 12600x^{0.7}$, $OPEX: y = 0.0067x$			(Gunaratnam et al., 2005)
Wetlands	$x = p.e.$, $y = €/p.e.$	$CAPEX: y = 3897.7x^{-0.407}$, $OPEX: y = 5.543x + 3127.5$			(Molinos-Senante et al., 2012)
Intermittent sand filter	$x = p.e.$, $y = €/p.e.$	$CAPEX: y = 2115.5x^{-0.399}$, $OPEX: y = 12.026x + 3518.9$			(Molinos-Senante et al., 2012)
Membrane bioreactor	$x = p.e.$, $y = €/p.e.$	$CAPEX: y = 5635.3x^{-}0.352$, $OPEX: y = 30.150x + 13542.0$			(Molinos-Senante et al., 2012)
Moving bed biofilm reactor	$x = p.e.$, $y = €/p.e.$	$CAPEX: y = 1187.0x^{-0.165}$, $OPEX: y = 12.794x + 6031.0$			(Molinos-Senante et al., 2012)
Trickling filter	$x = p.e.$, $y = €/p.e.$	$CAPEX: y = 12237.0x^{-0.487}$, $OPEX: y = 13.504x + 6030.0$			(Molinos-Senante et al., 2012)
Nitrification/ denitrification + filtration	$x = p.e.$, $y = €/m^3$	$CAPEX\&OPEX: y = 0.559–8 \cdot 10^{-6}x$	0	30000	(Sipala et al., 2003)
Filtration	$x = p.e.$, $y = €/m^3$	$CAPEX\&OPEX: y = 0.507–7 \cdot 10^{-6}x$	0	30000	(Sipala et al., 2003)
Activated sludge	$x = m^3$, $y = €/m^3$	$CAPEX\&OPEX: y = 1682.7x^{-0.148}$	100	70000	(Hilbig et al., 2020)

Table H.1

Treatment technology	Unit(s)	Cost equation(s)	Applicable range of x		Reference
			Min	Max	
Membrane bioreactor	$x = m^3/d$, $y = \$$	$CAPEX: log(y) = 0.569 * (log(x))^{1.135} + 4.605$, $OPEX: log(y) = 0.639 * (log(x))^{1.143} + 2.633$	37.85	378500	(Guo et al., 2014)
Digestion/stabilisation	$x = m^3$, $y = €/m^3$	$CAPEX\&OPEX: y = 8192.8x^{-0.322}$	400	50000	(Hilbig et al., 2020)
Sludge thickening	$x = m^3$, $y = €/m^3$	$CAPEX\&OPEX: y = 2220.8x^{-0.226}$	10	10000	(Hilbig et al., 2020)
Mechanical sludge dewatering	$x = m^3/d$, $y = €/m^3/d$	$CAPEX\&OPEX: y = 39.745x^{-0.155}$	30	14000	(Hilbig et al., 2020)
Chemical					
Chlorination	$x = m^3/d$, $y = €/m^3/d$	$CAPEX\&OPEX: y = 3.416x^{-0.422}$	3000	100000	(Hilbig et al., 2020)
Phosphate precipitation /flocculation	$x = m^3$, $y = €/m^3$	$CAPEX\&OPEX: y = 11.950x^{-0.574}$	800	20000	(Hilbig et al., 2020)
Coagulation-flocculation	$x = p.e.$, $y = €/m^3$	$CAPEX\&OPEX: y = 0.939–2 \cdot 10^{-5}x$	0	30000	(Sipala et al., 2003)
Gravity ion exchange softening	$x = plantcapacity$, $mgd(CAPEX)$, $x = plantflowrate$, $mgd(OPEX)$ $y =$	$CAPEX: y = –123.73x^2 + 165412x + 447664$, $OPEX: y = 15935x + 108481$	1.5	150	(Sharma, 2010)
UV disinfection	$x = m^3/d$, $y = €/m^3/d$	$CAPEX\&OPEX: y = 1209.2x^{-0.328}$	100	100000	(Hilbig et al., 2020)
Ozonation	$x = m^3/d$, $y = €/m^3/d$	$CAPEX\&OPEX: y = 348.02x^{-0.069}$	100	100000	(Hilbig et al., 2020)
Filtration					

Table H.1

Treatment technology	Unit(s)	Cost equation(s)	Applicable range of x		Reference
			Min	Max	
Sandfiltration	$x = m^3/d$, $y = €/m^3/d$	$CAPEX\&OPEX: y = 2045.3x^{-0.273}$	3000	100000	(Hilbig et al., 2020)
Filtration	$x = p.e.$, $y = €/m^3$	$CAPEX\&OPEX: y = 0.507–7 \times 10^{-6}x$	0	30000	(Sipala et al., 2003)
Membrane					
Ultrafiltration & microfiltration	$x = m^3/d$, $y = \$$	$CAPEX: log(y) = 1.003 * (log(x))^{0.830} + 3.832$, $OPEX: log(y) = 1.828 * (log(x))^{0.598} + 1.876$	37.85	378500	(Guo et al., 2014)
Reverse osmosis	$x = p.e.$, $y = €/m^3$	$CAPEX\&OPEX: y = 1.503–2 \times 10^{-5}x$	0	30000	(Sipala et al., 2003)
Nanofiltration	$€/m^3$	0.214			(Costa & de Pinho, 2006)
Physical					
Pre-/secondary treatment	$x = m3$, $y = €/m3$	$CAPEX\&OPEX: y = 1791.5x^{-0.185}$	100	100000	(Hilbig et al., 2020)
Coagulation-flocculation	$x = p.e.$, $y = €/m^3$	$CAPEX\&OPEX: y = 0.939–2 \times 10^{-5}x$	0	30000	(Sipala et al., 2003)
Aeration towers	$x = aerationtower\ volume, 1000ft^3$	$CAPEX: y = -19.857x^2 + 25002x + 175725$, $OPEX: y = 1525.2x + 4343$	0.68	256	(Sharma, 2010)
Screen	$x = m^3/d$, $y = €/m^3/d$	$CAPEX\&OPEX: y = 680.78x^{-0.297}$	100	50000	(Hilbig et al., 2020)
Grit chamber	$x = m^3/d$, $y = €/m^3/d$	$CAPEX\&OPEX: y = 2215.1x^{-0.392}$	100	20000	(Hilbig et al., 2020)
Primary treatment	$x = p.e.$, $y = €/m^3$	$CAPEX\&OPEX: y = 0.317–9 \times 10^{-6}x$	0	30000	(Sipala et al., 2003)
Multi-effect distillation	$€/m^3$	0.98			(Karagiannis & Soldatos, 2008)

Table H.1

Treatment technology	Unit(s)	Cost equation(s)	Applicable range of x		Reference
			Min	Max	
Steam-stripping column	$x = ton/h$, $y = \$(CAPEX)$, $y = \$/h(OPEX)$	$CAPEX: y = 16800x^{0.7}$, $OPEX: y = x$			(Gunaratnam et al., 2005)
Electrodialysis	$\$/m^3$	0.46			(Generous et al., 2021)
Vacuum filters	$x = totalfilterarea$, ft^2	$CAPEX: y = -0.1664x^2 + 1863x + 465811$, $OPEX: y = 0.0007x^3 - 1.4542x^2 + 1441x + 48536$	9.4	1320	(Sharma, 2010)
API separator	$x = ton/h$, $y = \$(CAPEX)$, $y = \$/h(OPEX)$	$CAPEX: y = 4800x^{0.7}$, $OPEX: y = 0$			(Gunaratnam et al., 2005)
Basket centrifuges	$x = totalmachine\ capacity, 1000gpd\ (CAPEX)$, $x = sludgeflowrate$, $1000gpd(OPEX)$	$CAPEX: y = 0.0001x^4 - 0.1326x^3 + 53.09x^2 - 710.06x + 522542$, $OPEX: y = 1056.4x + 26656$	3.6	720	(Sharma, 2010)
Sorption					
Carbon adsorption	$x = p.e.$, $y = €/m^3$	$CAPEX\&OPEX: y = 1.132 - 1 \times 10 - 5 \cdot x$	0	30000	(Sipala et al., 2003)
Powdered activated carbon feed systems	$x = feedcapacity$, lb/hr	$CAPEX: y = -0.0577x^2 + 778.86x + 175653$, $OPEX: y = -0.0204x^2 + 262.07x + 54144$	3.5	7000	(Sharma, 2010)
Granular activated carbon	$x = m^3/d$, $y = \$$	$CAPEX: log(y) = 0.722 * (log(x))^{1.023} + 3.443$, $OPEX: log(y) = 1.669 * (log(x))^{0.559} + 2.371$	37.85	378500	(Guo et al., 2014)