CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

# Reducing Environmental Impact in Automated Waste Collection through Optimization and User Feedback

## Scheduling the emptying process to reduce energy consumption and changing user behavior

Master's thesis in Computer Science - Algorithms, Languages and Logic

EMMA GUSTAFSSON

ELIN LJUNGGREN

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2017

Master's thesis 2017

# Reducing Environmental Impact in Automated Waste Collection through Optimization and User Feedback

Scheduling the emptying process to reduce energy consumption and changing user behavior

EMMA GUSTAFSSON

ELIN LJUNGGREN



Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2017

Reducing Environmental Impact in Automated Waste Collection through Optimization and User Feedback
Scheduling the emptying process to reduce energy consumption and changing user behavior
EMMA GUSTAFSSON
ELIN LJUNGGREN



Supervisor: Birgit Grohe, Computer Science and Engineering Department
Advisor: Alexander Ask, Sigma IT Consulting
Examiner: Peter Damaschke, Computer Science and Engineering Department

Master's Thesis 2017
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2017

Reducing Environmental Impact in Automated Waste Collection through Optimization and User Feedback
Scheduling the emptying process to reduce energy consumption and changing user behavior
EMMA GUSTAFSSON
ELIN LJUNGGREN
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

# Abstract

The automated vacuum waste collection system is an efficient way of handling waste in modern cities which is more sustainable than conventional systems. However, the environmental impact can be further reduced in the areas of system operation since the system consumes a substantial amount of energy where most of it is needed for the emptying process. One way of reducing the energy needed is to schedule the emptying procedure in a more efficient manner. Another way would be to stimulate reduction of waste production.

The system has a graph structure in the shape of a tree where waste enters through the leaf nodes. During the emptying procedure, waste is transported to the root node, that is the collection station, by air suction. To be able to decide when an emptying sequence should be initiated and which type of waste and individual waste inlets should be included, the tree needs to be traversed and evaluated. Our method can be seen as a scheduling algorithm inspired by ideas from online optimization, since the system is scanned continuously to see if an emptying process should be initiated. Once an emptying process is initiated, the emptying sequence is performed according to dynamic programming principles. The algorithms have been simulated and then compared to the statistics for the current procedure during January 2017. The best result shows an approximate improvement of 37.6% less kWh than current monthly use.

Another way to reduce the energy consumption is by lowering the amount of waste in the system by helping users gain insight in their waste generation. Less waste generated by users will lead to a lower demand on the system and fewer emptyings will be required. The goal was to create a prototype that can be used for creating incentive for reducing waste generation. Our solution to achieve this is a shell for a web application easily scaled to mobile screens.

Keywords: automated vacuum waste collection and scheduling, computational sustainability, data analysis, user feedback and behavioral change

# Acknowledgements

We would first like to thank our supervisor in this work, Birgit Grohe, for your guidance throughout the thesis. Secondly, we would like to thank Sigma ITC, and our advisor Alexander Ask, for their commitment and collaboration in forming the idea behind this thesis. A big thanks to Envac for letting us implement the idea and providing us with information about the system. Also, we would like to thank our friends and families for their support. Last but not least, we want to thank our examiner, Peter Damaschke.

Emma Gustafsson and Elin Ljunggren, Gothenburg, May 2017

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# 1
# Introduction

Urban areas are growing rapidly in population and in the last decade a term for describing a sustainable and living city, *smart city*, has been introduced [2]. There does not exist a formal definition, but the common opinion is that a smart city is a city where technology within Information and Communication Technology (ICT) is combined with traditional city structures and communication.

The *automated vacuum waste collection* (AVWC) system is one example of when a new technology has enabled modernization of a traditional city infrastructure. The AVWC system is a closed network of underground pipes where air suction is used to transport and collect waste in a city. A fully working waste collection system is crucial in a city. Lack of correct waste management has throughout history proven to be devastating for human health by spreading diseases through contaminated drinking water [3]. The system eliminates the need of waste trucks in densely populated areas and thereby contributes to less traffic and emissions. It also removes other side effects from traditional waste collection, such as foul-smelling garbage bins. These are positive effects, but there are possibilities of further improvement. The system requires a lot of energy to operate, and the majority of this energy is used to produce air suction.

## 1.1 Background

The amount of waste produced by humans has increased along with urbanization. According to a report by the World Bank, both the world population and the amount of waste they produce will increase in the coming years [4]. They conclude that as we are moving towards an urban future, the amount of waste is growing even faster than the rate of urbanization. Hence, to adapt to a more sustainable lifestyle, the waste management needs to be more effective and the amount of waste generated needs to be reduced.

The AVWC system was invented by Envac AB [1] in the process of installing a central vacuum system at the hospital in the Swedish city Sollefteå in the late 1950's. Just a few years later the vacuum waste collection system was installed in a residential area [5]. Today, these systems are in use globally and each system is designed to fit the client. Some systems require identification to be able to open the lid to the inlets and in this way limiting waste disposal to the residents only. The collection of this kind of data opens up to possibilities of tracking waste disposal

patterns in certain areas at specific times. More information about inlets and how the user interacts with them can be found in Section 5.1.

The data originating from disposal patterns can be used to increase efficiency in waste management and create incentive for people to reduce their waste volume. Giving users feedback on energy use has proven to be an effective way of changing energy consumption behavior [6–8]. Giving similar feedback on waste generation could possibly give the same effect, resulting in a reduced amount of waste. However, feedback directed to users must be convenient for them and the platform used should be easily accessible to the target group, i.e. Envac's users.

The AVWC system needs a substantial amount of energy to operate where the major part is used for the emptying process. Thus, an optimization of the emptying in the system have the greatest potential of improving energy efficiency. The emptying procedure is done in sequences that describes in what order to empty the inlets and may be planned in a time schedule or started by a trigger in the system, for further details see section 3.1.

## 1.2 Purpose and Objectives

The purpose with the thesis is to use data from AVWC in ambition to further minimize the environmental impact of the system. There are two problems addressed in this thesis. Firstly, the possible optimization of emptying sequences. Secondly, how to create incentive for a change in individual waste generation behavior, by presenting feedback to a user.

The first problem can be modelled as a scheduling problem, where the desired outcome is a schedule for emptying the inlets in order to minimize the energy used to operate the system. The energy use is directly proportional to the running time of the fans producing air suction, which transport waste from an inlet to the waste collection station. Our schedule should perform better than the worst case scenario, which would be to run pre-scheduled emptying sequences that empties the entire system each time, and the schedule currently in use at Envac, described in Section 3.2.

It is a complicated problem to visualize data in such manner that it encourages the user to improve their waste generation behavior. The choice of platform for presenting the data is also decisive for the outcome of the feedback. Previous studies done on user feedback on energy use will act as the foundation of how to present information on the chosen platform.

The objectives of this thesis can be stated as:

- First sub-goal:
  - Model the system and implement one or more scheduling algorithm(s) for emptying the inlets.
  - Develop a program that simulates the possible solutions and compares their result with the performance of the current system.
  - Evaluate the solution. It should result in an energy consumption lower than the worst case scenario, where each sequence is pre-scheduled and empties the whole system.
- Second sub-goal:
  - Gain insight in what way of presenting feedback is most effective and suitable under prevailing conditions.
  - Choose a platform for presenting feedback to the user.
  - Develop a prototype on the chosen platform that follows the concepts found most suitable.

## 1.3 Literature review

Although the system has been in use for quite some time, there has not been much research done regarding its energy consumption. A series of papers were produced [9] that attempts to formally define the system and to model the problem as a Constraint Integer Programming (CIP) problem [10].

### 1.3.1 A Vacuum Waste Collection System in Spain

The research group in Spain chose to use CIP because of its capacity of solving hard computational problems, including real-world operation problems [10]. Their objective is to optimize energy consumption when emptying the system by finding a set of emptying sequences and air speed operations for an operative period of time that minimizes energy cost.

The model is built on constraints, parameters and functions, decision variables and auxiliary variables that contribute to the solution. Many of the constraints are directly or indirectly focused on air speed. The objective function of their CIP model is:

$$min(E_{tr} + E_{st} + P)$$

where $P$ denotes a penalty cost for leaving unloaded inlets above a threshold of their maximum capacity, in other words, to avoid a solution where no inlets should be emptied. The energy consumption is denoted by $E$ and is divided in two phases, transitory, $E_{tr}$, and stationary, $E_{st}$. The transitory phase is the energy consumed when air speed is altered and the stationary phase is the energy needed for transporting waste.

Their study concluded that their encoding is able to find the optimal solution for all possible levels of waste in two of the three examined plants within 5 minutes. For the third plant they were unable find an optimal solution when the waste level was above 20%. This aside, they conclude that the used approach manages to find an optimal emptying sequence within a few minutes and is thereby a feasible solving solution.

### 1.3.2 Feedback and Behavioral Change

Different types of feedback has proven to be able to influence behavioral change with various techniques. In the field of energy consumption, several studies have shown that providing households with feedback on their energy use leads to reduced consumption [11, 12]. Frequent or instant feedback has been found to give a better result than for example monthly feedback, e.g. via the electricity bill [8]. A study by Vassileva et al. on users in both houses and apartments in Sweden indicated that the environmental factor was more important than the price concerning energy saving, more so for the users living in apartments than the ones living in houses [13]. If information is easily accessible and contains some competitive, social or committing elements, it is more interesting to the user and encourages change [7, 14].

Siero et al. have been studying how comparative feedback can be used to change organizational energy consumption behavior [7]. Their studies were performed on two user units in a company, which were given different kinds of feedback on their energy consumption. One group received feedback on both their own and the other unit's performance, while the other only received feedback on their own efforts. The results clearly show that the first group saved more energy than the other group. The first group also proclaimed that they felt a stronger group identity than before and the effects on their energy consumption behavior lasted even long after the study.

Few studies have been made in the area of user feedback in waste generation, but Dahlén et al. researched how a weight-based billing system affects the amount of waste in household waste collection systems in Sweden [15]. The studies show that the amount of waste is reduced when presenting feedback about the waste generated and its economic consequences.

## 1.4 Delimitations

Stationary AVWC systems are in use in various areas, e.g. hospitals, airports and residential areas. The systems are adapted for each purpose and are similar, but not equivalent. The situation of residential areas is applicable on everyday life and is considered more interesting than the other cases. Therefore, only the case of residential areas will be considered in the thesis. Calculations and comparisons will be based on one specific system, further described in Section 3. However, the solution should be applicable on other systems as well.

The user feedback application will be digitally developed and no physical product will take form. Considering that the application is intended to be a prototype and not a full product, no user studies or interviews will be carried out during the development. Standards in user interfaces will be followed. This will, however, lead to an uncertainty whether the intended user incitement has been fulfilled or not. In general, incitement is a factor that is rather difficult to measure. Therefore the previous research in user feedback will act as a foundation of how to create incitement.

# 2
# Theory

In this chapter, the different concepts used in this thesis are described.

## 2.1 Graph Models

A graph is defined as $G = (V, E)$, where $u, v \in V$ are nodes, and $e = (u, v) \in E$ are edges. In a graph, the edges can be directed or undirected. In this thesis we only consider graphs with undirected edges. A graph is a tree if it is connected and does not contain a cycle. The topmost node in the tree is the root node, and the tree consists of nodes linked to it, see Figure 2.1. In the figure the root node is linked to two other nodes, 1 and 4. These are called the child nodes of the root and the root is their parent node. The root node does not have a parent node. The nodes that do not have any child nodes are called leaf nodes, see node 2, 3 and 4 in Figure 2.1. Each edge in the tree can have a value attached that represents its cost. The depth of a node in the tree is the number of edges from the node to the root node. The tree in the figure is a binary tree, which is a tree where each node has at most two children [16].

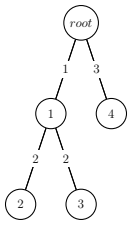


**Figure 2.1:** An example of a tree.

## 2.2 Linear Programming and Integer Linear Programming

Linear Programming (LP) and Integer Linear Programming (ILP) [17] are often used to describe various optimization problems. An LP consists of a linear objective function, linear constraints and continuous variables, commonly stated as:

$$\begin{gathered} \min c^T x \text{ s.t.} \\ Ax \geq b, \\ x \geq 0 \end{gathered}$$

where $x$ is a vector of variables, $c$ and $b$ are vectors of dimensions $n$ and $m$ respectively, and $A$ is a matrix of dimension $m * n$, where $n$ is the number of variables and $m$ is the number of constraints. There are many possible formulations to express an LP, but a common objective is to minimize the cost while considering a number of constraints.

If $x$ is required to be integer, the problem is an ILP. Both LPs and ILPs can be solved with various methods. For example, the simplex method [18] or interior point methods [19] can be used to solve LPs. Methods for solving ILPs often use an LP solver as a subroutine in combination with a tree search, e.g. branch and bound. ILPs belong to the class of NP-hard problems while LPs are polynomially solvable.

## 2.3 Constraint Programming and Constraint Integer Programming

Constraint Programming (CP) is a technique for solving Constraint Satisfaction Problems (CSP) [20]. As the name indicates, CSP aims to find a solution that satisfies a number of constraints. CSP can be described as:

$$\begin{gathered}\text{A set of variables: } X = \{X_1, ..., X_n\},\\ \text{A set of domains: } D = \{D_1, ..., D_n\},\\ \text{A set of constraints: } C = \{C_1, ..., C_n\}\end{gathered}$$

Each variable $X_i$ has a non-empty domain $D_i$. A constraint $C_j$ has a subset $x_j \in X$ for which the constraint applies and it indicates which combinations of the domain that are allowed. The ambition is to find a selection of domain values to each variable where all constraints are fulfilled, or determine that no such selection exists. The constraints can be of arbitrary shape, e.g. for discrete variable domains they can be expressed as truth tables for all combinations of variables and values.

When modelling a large range of constraints or using predefined global constraints, CSPs allows many possibilities for the user. Note, that CSP does not use an objective function in the same way LP and ILP does.

When solving CSPs the procedure is branched into two parts. Firstly, the constraints are studied one at a time. Values that are not consistent with the domain and the other values in the current variable's domains are removed. This method is called propagation. However, propagation on its own does not solve all problems, hence a second type of method is added: tree search. In this second type, the solver backtracks to a previous state when a dead end is reached in the propagation. A common example of CSP is the map coloring problem [20].

Constraint Integer Programming [21] is a combination of both Mixed Integer Programming (MIP) and CP. MIP is a combined variant of LP and ILP, where a subset

of the variables are restricted to be integers while the rest is continuous. CIP aims to combine benefits of both methods, i.e. the freedom to express any constraints with the slight limitation to only allow integer variables. The model also allows a linear objective function.

## 2.4 Scheduling Algorithms

Scheduling algorithms aim to optimally schedule a number of tasks over a number of resources [16]. That is, decide the order in which the tasks should be done within a given time frame. In some cases, it has to be decided which tasks to include or not. Some scheduling problems have precedence constraints, which means that some tasks must be completed before another can begin. It can also be a desire to minimize the total sum of time, or minimize the cost of each job.

Algorithms for scheduling problems can be greedy [22], i.e. they always choose the best action in each stage of the algorithm. Another way of solving scheduling problems is by the concept of Dynamic Programming, which breaks a problem down to a series of subproblems [16]. Correct solutions are then built up to larger and larger subproblems, and in the end the original problem will be solved.

In an offline algorithm, all information is given before the search for an optimal solution is initiated. In contrast, an online algorithm make decisions based on data as that is received along the way, with no knowledge of the future. A common way to decide the effectiveness of an online algorithm is by its competitive ratio, which is the relation between itself and its hypothetical offline counterpart [23]. An example of online algorithm is in the elevator problem. An elevator receives a request to travel between floor X and Y, and during its travel new requests are added. The question is how the elevator should handle the new requests in order to receive the best outcome.

# 3
# System Overview

This chapter describes the construction of the system and the emptying procedure step by step. The data required to meet the thesis' objectives is described. Furthermore, the data acquired is specified and interpreted.

## 3.1 System Description

The system can be described as a tree where the waste collection station is the root of the tree, see Figure 3.1. The nodes in the graph are depicted as circles in the figure and represent inlet clusters or pipe junctions. An inlet cluster is where one or more inlets are located. The edges in the network represent the pipes between these nodes. In Figure 3.1, an inlet cluster is visualized by a node with one or several rectangles attached, one rectangle per inlet. Each inlet handles waste from a specific fraction, such as paper, plastic, organic or residual waste. At each inlet cluster there can be several inlets for different fractions, but also multiple inlets for the same fraction. For example, residual waste is very common and might need two or three inlets at one node. The system can only empty one fraction at a time. Depending on which fraction is being emptied, the pipe leading into the terminal directs the incoming waste automatically to the correct container.

The network is segmented into sub-parts by placing sector valves strategically along the tree. Instead of emptying the entire tree at the same time, the vacuum can be concentrated on a certain branch of the tree. There are also air valves at some inlets that are opened to ensure that the air flow is strong enough to transport the waste. Each node needs a corresponding air valve that is opened when inlets at that node are emptied. Each system is unique in the choice of fractions, location and number of inlets.

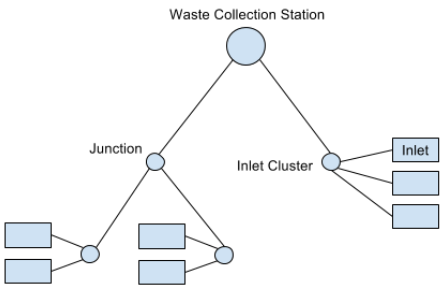


**Figure 3.1:** A simple AVWC system.

The system considered in this thesis will be referred to as *Plant 1*, and its actual whereabouts are censored for the sake of anonymity of the inhabitants in the area. The plant is located in a rather newly built area (2012) in Sweden which is

still expanding. A map of Plant 1 is available in Appendix A. The longest branch in the tree network is 1,34 km and there are currently over 140 inlets connected to the system at 38 inlet clusters. The system handles three fractions of waste: paper, plastic and residual waste. A graph representation of the system can be seen in Appendix B, where the cost on each edge represents the length of the pipes in centimeters in scale 1:2000.

## 3.2 Emptying Procedure

When initiating an emptying process in the system, it has to be decided which inlets to empty and in what order. This is referred to as an emptying sequence. Today, the most common way to decide when to start an emptying sequence is to schedule fixed hours based on simple statistics on system use. The order in which inlets are emptied is decided beforehand. Either the sequence starts with the inlet closest to the collection terminal, working its way out, or the other way around. Starting with the inlet farthest away from the terminal is more energy efficient and the most practiced way of modelling the sequence.

In figure 3.2 a more extended example of the system is presented. Air valves are represented by triangles, and their denotation, $AV_i$, correlate to the one of inlet clusters, $IC_i$, e.g. $AV_1$ is the air valve needed to empty $IC_1$. Every inlet cluster is connected to an air valve. One air valve can be associated with multiple inlet clusters. All inlets connected to the same air valve are not always located at the same node, as with inlet clusters $IC_{2\_1}$ and $IC_{2\_2}$. Fractions are denoted by $f_i$. Figures 3.3, 3.4, 3.5, 3.6, 3.7 and 3.8 show a step by step emptying of fraction 2 in the system. In Figure 3.3 the content of $IC_1$, $f_2$, is released into the system and moved towards the waste collection station. Only one inlet of the same fraction can be emptied at a time and the next inlet cluster, $IC_{2\_2}$, can be emptied as soon as the waste from $IC_1$ has passed the junction where their paths connect, as can be seen in 3.4. However, this does not apply when emptying several inlets associated with the same air valve. In this case it is possible to empty an inlet immediately after the discharge valve of the one before is closed, but only if they are emptied in the opposite order, i.e. starting with the inlet with shortest length to the root.

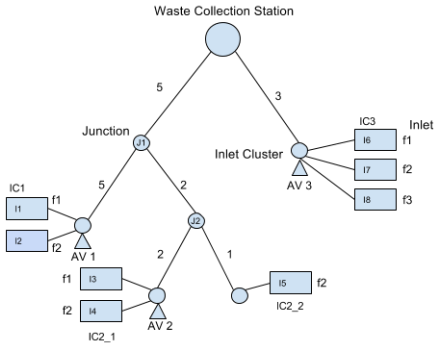


**Figure 3.2:** Extended example of the system.

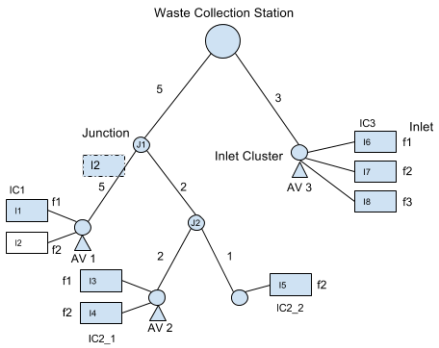


**Figure 3.3:** Emptying example, part 1.

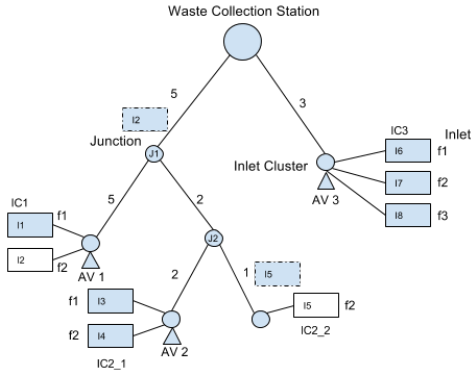


**Figure 3.4:** Emptying example, part 2. Here, contents from $IC_1$ has passed J1 and $IC_{2\_2}$ $f_2$ is released into the network.

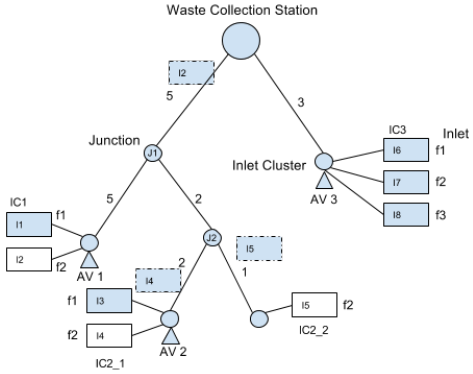


**Figure 3.5:** Emptying example, part 3. Here, $IC_{2\_1}$ has closed its discharge valve and contents from $IC_{2\_2}$ has been released into the network.

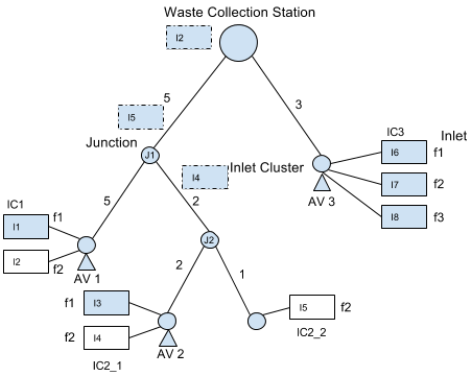


**Figure 3.6:** Emptying example, part 4. Here, contents from $IC_{2\_1}$ has passed $J_1$ and contents of $IC_{2\_2}$ has passed $J2$.

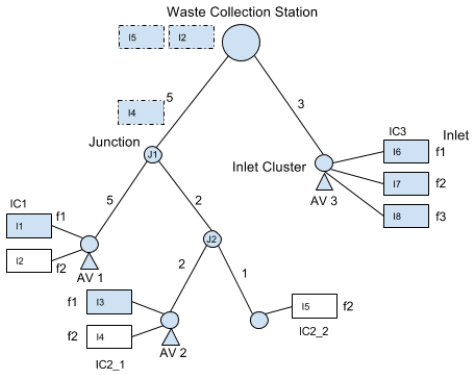


**Figure 3.7:** Emptying example, part 5. Here, contents from $IC_{2\_2}$ has passed $J_2$.

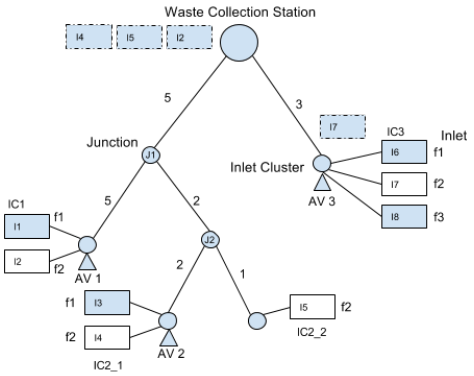


**Figure 3.8:** Emptying example, part 6. Here, contents from $IC_3$ has been released into the network.

Most systems measure one or two levels by sensors in each inlet to anticipate the amount of waste, *low level* and *high level*. In Plant 1, the most common procedure is to measure only one level that is placed approximately at 50-75% capacity. In Plant 1, there are two kinds of emptying procedures. The first is to empty the entire system of a fraction in one sequence. In the second procedure, the system decides which inlets should be emptied by checking their level. If an inlet has not reached any level yet it does not need to be emptied. All inlets that have reached at least level low will be emptied. If the level is high in an inlet the system is triggered to start the emptying sequence immediately.

An inlet is emptied by starting fans at the terminal that creates vacuum in the pipes and the waste is then transported by air from the inlet to the waste collection station. In Plant 1, the system has four available fans, each with a capacity of 90kW and at most two are used simultaneously. It is possible to set frequency for the fans for a desired air speed. Air speed is the most important parameter and it is calculated to be as low as possible but still be able to transport the heaviest type of waste of a certain fraction. The air speed is constant for each fraction. Around 70 to 90% of all energy used by the system during an emptying sequence is for creating and maintaining air flow. This means that the difference in energy used when transporting waste through the pipes compared to only transporting the air is not significant.

An example of a terminal can be seen in Figure 3.9. At the terminal, the waste is separated from the air stream and placed into containers, where it is compacted. In Plant 1, if a container is full, it is replaced automatically with a new container. However, in Envac's most common systems the switch of containers is done manually. The system handles about one to two tons of waste a day.

**Figure 3.9:** Inside of the terminal of Plant 1. The blue pipes carry air and the green carry waste.

In Plant 1, all disposals are registered by the id of the disposer. At some inlets there are scales that can weigh the waste for each disposal. However, Plant 1 only has scales at about 6% of all inlets and therefore it is not possible to calculate the weight of the waste for each household in the system. Furthermore, since the level of waste in the inlets is not measured in detail it is difficult to decide if they should be emptied or not. Some inlets fill up very fast during some periods while others may not. For the same reason it is not possible to bill households based on their waste. Instead, the housing association is billed by the number of apartments connected to the system.

The total running time of the system, i.e. for how long the fans are on, is the main parameter to be considered in order to lower the energy consumption. The data described in 3.3 is intended to be used to find a substantial improvement compared to the system currently in use at Envac.

When calculating the time used in an emptying sequence, Envac considers the following steps:

1. The air valve (AV) is opened.
2. The time it takes for the pressure in the pipes to go from atmospheric pressure to stabilized vacuum.
3. The discharge valve (DV) of an inlet is opened.
4. The time it takes for all waste to be emptied from the inlet.
5. The discharge valve is closed.
6. The time required to transport the waste from the inlet to the collection terminal.
7. Repeat all steps for each new air valve. If the next inlet to be emptied use the same air valve, then only repeat steps 3-6.

The time for step 1, 3, 4 and 5 are constant and equal for each inlet, usually a few seconds only, while the time for step 2 and 6 is bounded by the length from the root to the current inlet.

## 3.3 Data Sets

The data required to approach this thesis' objectives can be divided into two parts. Firstly, data regarding which user has disposed waste at what time is needed to be able to give the users feedback on their waste habits. Secondly, data regarding emptying sequences of the system is needed for evaluation purposes, meaning which inlets are being emptied, in what order they are being emptied, and at what waste level an emptying process is initiated.

The first part of the data originates from logs of each disposal for a given time. A disposal is logged each time the lid or hatch to an inlet is opened and does not depend on the amount of waste disposed. A disposal has parameters such as id for the logged disposal, id of the used inlet, time stamp of when the disposal occurred, id of the customer, information of which tag that was used, weight of the disposed waste bag (if a weight is registered), address of the inlet and also some parameters that indicates whether the disposal succeeded without errors or not. Each household, represented by the customer id, owns a number of specific tags.

The second part of the received data is not as exhaustive as desired. It consists of statistics over a month regarding emptying procedure of Plant 1, meaning the average number of inlets emptied per minute, how long all emptying sequences took in total and how the details of the emptying sequence look like. For example, there can be emptying sequences including all inlets, but also sequences including only the inlets that has a certain level. More details about emptying sequences is accounted for in Section 3.2. Even though the data set acquired is not what was anticipated, it is possible to estimate the rate in which the system fills up by using the first part of the data in combination with approximations regarding waste bag sizes and inlet capacity. An inlet has an average capacity of 188,5 liter and a waste bag has an approximate volume of 8 liters. This makes an inlet able to handle 23 waste bags before being full.

Figure 3.10 shows statistics of the total number of disposals over time in the given area. The figure shows how disposals differentiate over months and weekdays. Figure 3.11 shows the number of disposals per unique inlet in 2016. This data can be used as statistics for the probability of an inlet to fill up in a time span. As Figure 3.10 shows, there is a slightly larger probability of disposals to occur on weekends then on a weekday. Some inlets are more frequently reoccurring than others, which might mean that they are used by commercial users such as restaurants, etc.

The disposal data contains some anomalies that will be considered inaccurate. For instance, one user tag has been registered to open an inlet hatch more than twenty times during two minutes. It is not likely that it is twenty separate bags, therefore an average is calculated. If more than one disposal is registered within four seconds at the same inlet, only one disposal will be counted.

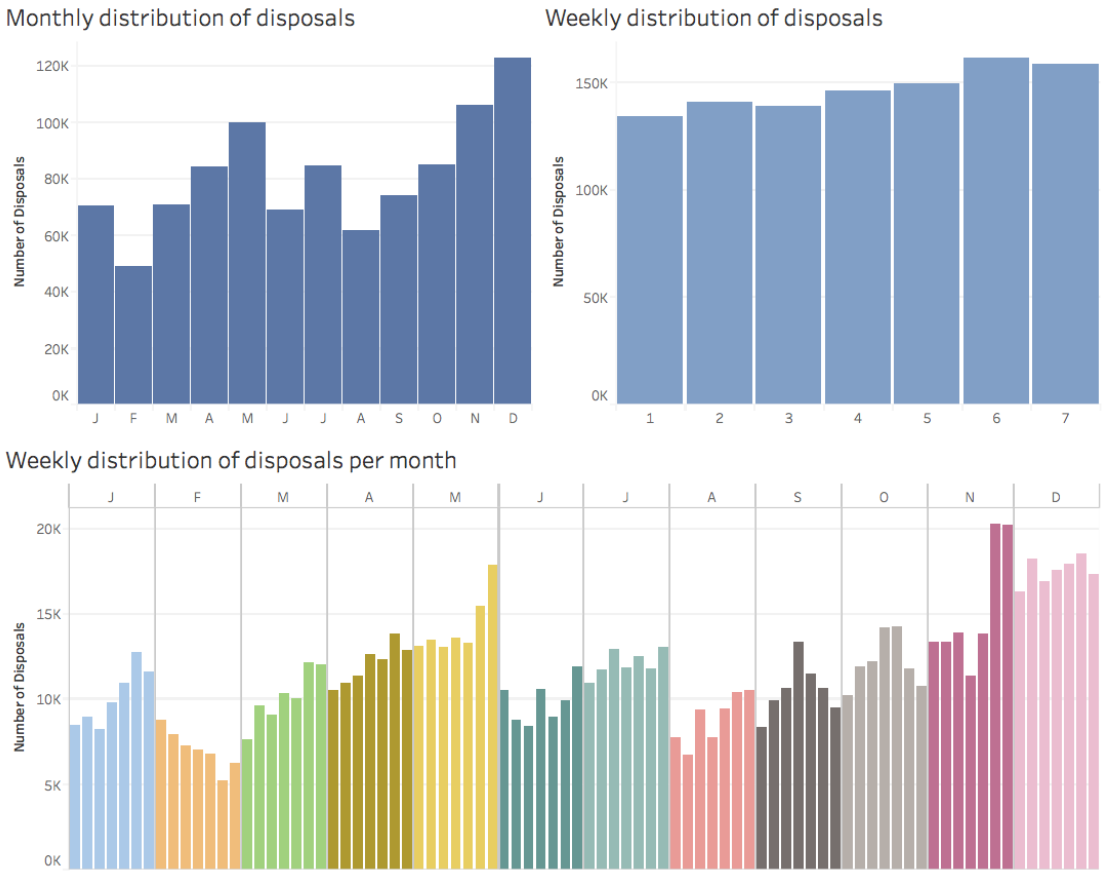


**Figure 3.10:** Distribution of a total number of 974902 disposals during the years 2013-2016.

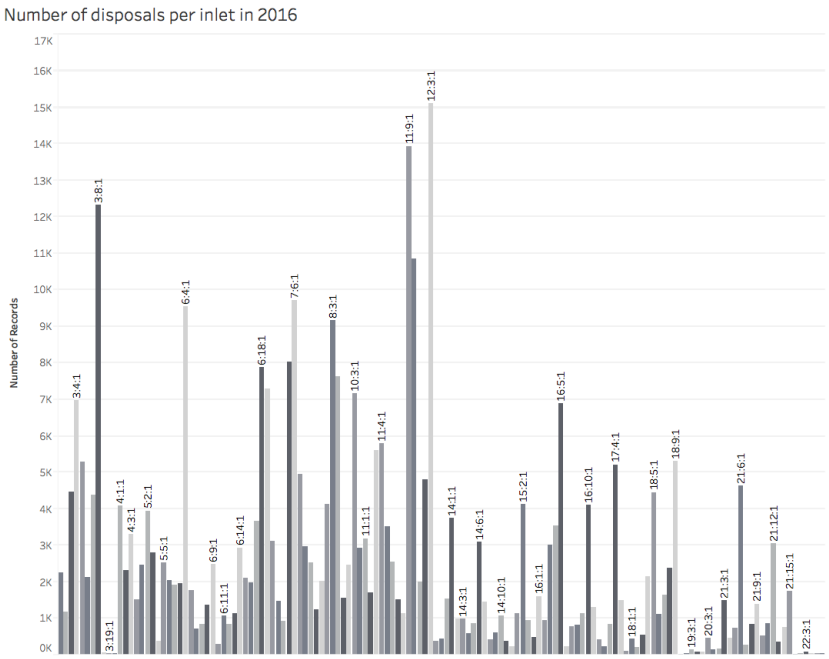


**Figure 3.11:** Statistics for the amount of disposals made at each inlet in 2016.

# 4
# System Optimization

Several methods have been tested when searching for a suitable model for the scheduling problem. In this chapter the methods, formulations and decisions regarding choice of method are described. All algorithms are described in detail. These descriptions act as an aid for the reader to grasp the algorithms, step by step.

## 4.1 Using Constraint Integer Programming

Fernàndez et al. [10] suggest an approach using CIP to solve the scheduling problem. When comparing their problem to the one presented in this thesis, the main difference is the inclusion of sectors and air speed in their model. In the system studied in this work, the air speed is a constant, based on the fraction that is to be transported, and does not need to be included in any constraints. Sectors do not have a significant impact on normal functionality in the system and do not need to be included in the model. Another difference is that Fernàndez et al. calculate the transitory and stationary energy when describing the total energy consumption. We, on the other hand, only consider the system run-time as the important factor for calculating the energy consumed.

Without the parameters described, the CIP approach appeared to be unnecessarily complex for solving this scheduling problem. Thus, the method was discarded and other possibilities were explored.

## 4.2 Using Integer Linear Programming

ILP is a common approach when modelling scheduling problems and it could be suitable for modelling our problem. First variables needs to be stated:

- $F$ is the set of all fractions in the system.
- $I$ is the set of all inlets in the system.
- $A$ is the set of air valves in the system.
- $J$ is the set of junctions in the system.
- $x_i^f \in \{0,1\}, f = 1, ..., |F|, i = 1, ..., |I|$, indicates whether an inlet should be emptied in this sequence or not. If $x_i^f$ is 0, the inlet should not be emptied.
- $y_k \in \{0,1\}, k = 1, ..., |A|$, indicates whether an air valve should be opened in this sequence or not. If $y_k$ is 0, the air valve should not be opened.

- $z_i \in \{0,1\}, i = 1, ..., |J|$, indicates whether a junction should be passed in this sequence or not. If $z_i$ is 0, the junction should not be passed.
- $\lambda_{ik} \in \{0,1\}, i = 1, ..., |I|, k = 1, ..., |A|$, indicates if inlet $i$ belongs to air valve $k$. If $\lambda_{ik}$ is 0, they do not belong together.

Constraints needs to be added to these variables to represent the system. Let $I^k$ be the index set of all inlets that belong to $AV_k$. Then

$$\forall i \in I^k \; \forall k, \quad x_i \leq y_k \tag{4.1}$$

or the following cumulative constraint could be used:

$$\forall k, \quad \sum_{i \in I^k} x_i \leq M y_k \tag{4.2}$$

where M is the maximum number of inlets that can be connected to an air valve. The objective function could be modelled as

$$min \sum_k c_k y_k \tag{4.3}$$

where $c_k$ denotes the cost associated with opening the air valve $k$.

The current model is very general and does not contain any details of in which order the air valves are opened and at what point in time. To describe a whole sequence, one would have to introduce a new set of variables for each point in time and also define connecting constraints between variables for e.g. the same inlet but for time $t$ and $t-1$. The number of variables and constraints will increase substantially and it is unsure if anything reasonable will come out of such a model. Thus, it was decided not to investigate this model further.

## 4.3 Scheduling Algorithms

We consider different scenarios for how an emptying sequence can be executed. The first scenario is the simple case of the worst-case scenario, where all inlets are emptied, and then more advanced emptying sequences are considered. The different scenarios are:

1. Empty all inlets (worst-case scenario).
2. Empty triggered inlet.
3. Empty all inlets with at least a low waste level.
4. Empty based on statistics.
5. Empty using statistics, minimum and maximum level.

By developing separate scheduling algorithms for the different scenarios it is possible to simulate the system in a good way and compare the different outcomes. More detailed descriptions of the different scenarios will be given in their respective subsection in this chapter.

The system is modelled as a graph $G = (V, E)$, where $E$ is the set of edges and $V$ the set of nodes. Each edge represents a pipe between two nodes and has a cost in terms of its length in meters. A node is either a junction $J_i$ in the set $J$ of all junctions or an inlet cluster $IC_i$ in the set $IC$ of all inlet clusters, thus $V = J \cup IC$. Each node has a number of parameters, depending on if it is an inlet cluster or a junction. However, all nodes have three parameters in common: $id$, $L_i$ and $l_i$. The id of the node is denoted by $id$, $L_i$ is the total length in meters from the node to the root node and $l_i$ is the length in meters from the node to its parent node. A junction is associated with four additional parameters:

$$J_i = (id, L_i, l_i, child^l, child^r, d_i^l, d_i^r)$$

where $child^l$ and $child^r$ are the left and right child of the node and $d_i^l$ and $d_i^r$ denote the depth, in terms of length in meters, of the left and right subtree. An inlet cluster is associated with two additional parameters:

$$IC_i = (id, L_i, l_i, av_i, inlets_i)$$

where $av_i$ is the air valve to which the inlet cluster belongs and $inlets_i$ is a list of all inlets belonging to this cluster. Each air valve is denoted $AV_i$ in the set of all air valves $AV$ and each inlet is denoted $I_i$ in the set of all inlets $I$. Air valves and inlets are associated with the following parameters:

$$AV_i = (id, L_i, clusters_i),$$
$$I_i = (id, fraction_i)$$

Both $id$ and $L_i$ are interpreted the same way in the context of $AV_i$ and $I_i$ as for the nodes, $clusters$ is a list of all inlet clusters belonging to that AV and $fraction_i$ is the fraction of waste in that inlet. All presented parameter lists will later be extended in the process to fit the current scenario.

### 4.3.1 Scenario 1: Empty All Inlets

The sets of parameters for $IC_i$, $J_i$, $AV_i$ and $I_i$ described above are sufficient when developing an algorithm for the worst-case scenario of an emptying sequence, where all inlets in the system are emptied. Algorithm 1 describes the process of emptying a whole subtree and could be used to represent the worst case scenario by giving the root node as argument to $processSubtree()$.

In Algorithm 1 the goal is to create an emptying sequence, $emptySeq$, that could be executed by the system. The variable $emptySeq$ is a global list where information is stored about which AVs and DVs that are opened in the sequence and for how long.

Also the variable $emptiedAVs$ is a global list containing all AVs that have been added to $emptySeq$. Given a node $V_i$, $processSubtree()$ empties all inlet clusters belonging to the tree with root $V_i$. When the argument node is a junction, the two children are compared and then the function is called recursively. The first call is made for the child with the deepest subtree. The return values from each call are saved. Null will be returned in the case where no air valves have been opened in a subtree. Otherwise, the air valve opened last will stay open until the next AV to be emptied is reached. When the recursion is done, all the transported waste is in the root node and the sequence is finished.

For every inlet cluster that is reached, the function $emptyAV()$ is called with the AV that should be emptied and the fraction it belongs to as arguments. It adds all inlets of the correct fraction which belongs to that AV to the emptying sequence. Also the time for stabilizing air speed, and opening and closing the AVs is added, these are constants denoted by $STABILIZE\_TIME$, $OPEN\_TIME$ and $CLOSE\_TIME$ respectively. These constants are typically a few seconds, more details can be found in Chapter 6.

With Figure 3.2 as an example for the outcome of running Algorithm 1, the procedure of emptying all inlets with fraction 2 is to begin with the air valve furthest away from the root and then continue with inlet clusters in subtrees in the path to the root node. This results in the following emptying sequence: $AV1_{:}\{IC_1\}$, $AV_2 : \{IC_{2\_2}, IC_{2\_1}\}$, $AV3 : \{IC_3\}$, where fraction notation has been omitted.

**Algorithm 1:** Empty all inlets in subtree $V_i$

```
1  IC_i = (id, L_i, l_i, av_i, inlets);
2  J_i = (id, L_i, l_i, child^l, child^r, d_i^l, d_i^r);
3  A_i = (id, L_i, clusters_i);
4  I_i = (id, fraction);
5  List emptySeq;
6  List emptiedAVs;
7  Function processSubtree(V_i, fraction)
8      if V_i is IC_i then
9          if av_i is not in emptiedAVs then
10             emptyAV(av_i, fraction);
11             return av_i;
12         else
13             return null ;                          // No new AV is opened
14     else
           // V_i is a Junction
15         compare V_i.d_i^l and V_i.d_i^r ;           // Assume V_i.d_i^r is larger than V_i.d_i^l
16         av1 = processSubtree(child^r, fraction);
17         av2 = processSubtree(child^l, fraction);
18         if av1 == null and av2 == null then
19             return null;
20         else if V_i.L_i != 0 then
21             av = av2;
22             if av2 == null then
23                 av = av1;
24             add av and V_i.l_i time to emptySeq;
25             return av;
26         else
               // Root reached
27             add emptySeq.lastAV and CLOSE_TIME time to emptySeq;
28             return av ;                            // Sequence is finished
29 Function emptyAV(av, fraction)
30     if emptySeq is not empty then
31         add emptySeq.lastAV and CLOSE_TIME time to emptySeq;
32     add av and OPEN_TIME time to emptySeq;
33     add av and STABILIZE_TIME time to emptySeq;
34     foreach IC ic in av.inletClusters do
35         foreach I i in ic.inlets do
36             if i.fraction == fraction then
37                 add i.id and emptyTime time to emptySeq;
38                 set lvl of inlet i to 0;
39     add av and av.inletClusters.lastCluster.l time to emptySeq;
40     add av to emptiedAVs;
```

### 4.3.2 Scenario 2: Empty Triggered Inlet

Algorithm 2 describes the case where one inlet fills up and triggers the system. Not all inlets in Plant 1 have a sensor for high level that can trigger the system. Instead, the data set in combination with approximations regarding volume of waste bags and inlet capacity, as described in Section 3.3, is used to calculate the level in an inlet. Thereby, an approximate high level can be used. The parameter list has to be extended by adding another parameter, $lvl_i$, to $I_i$, to enable a level check. The new parameter indicates the level of waste, in percent, in the inlet. This can be checked to decide when an inlet is triggered. Also, the list is extended to let the AVs have a list, $List_i$, containing all nodes on the path from the AV to the root node. In this case we want to empty the AV containing the triggered inlet first and then add all AVs with an inlet cluster connected to the path. An inlet cluster $IC$ is connected to the path if there is only one edge between it and a node in $List_i$, i.e. $List_i$ contains the parent of $IC$.

In contrast to Algorithm 1, Algorithm 2 does not have a recursive function. Instead it has a function, *emptyTriggered*(), which given the triggered inlet cluster loops over the path from the inlet cluster to the root node. First the AV containing the triggered inlet is emptied, using the subroutine *emptyAV*() from Algorithm 1, and then it checks for leaf nodes (inlet clusters) connected to the path. All AVs with connected clusters are emptied and the system is turned off when the root node is reached.

**Algorithm 2:** Empty a triggered inlet and all connecting inlets

```
1  IC_i = (id, L_i, l_i, parent, av_i, inlets);
2  J_i = (id, L_i, l_i, parent, child^l, child^r, d_i^l, d_i^r);
3  A_i = (id, L_i, clusters_i, List_i);
4  I_i = (id, fraction);
5  List emptySeq;
6  List emptiedAVs;
7  Function emptyTriggered (IC_i)
8      add av_i and l_i time to emptySeq;
9      av = av_i;
10     last = I_i;
11     foreach v in List_i do
12         if other child than last is leaf and child.av is not in emptiedAVs then
13             emptyAV(child.av);
14             av = child.av;
15         add av and v.l time to emptySeq;
16         last = v;
17     turn off;
```

As an example of the procedure in Algorithm 2, assume the system depicted in Figure 3.2. If the triggered inlet was in the inlet cluster $IC_{2\_1}$, Algorithm 2 would create the following emptying sequence: $AV_2 : \{IC_{2\_1}, IC_{2\_2}\}$, $AV_1 : \{IC_1\}$. This example is very small, but in a slightly larger system such as in Figure 4.1, if $IC_{2\_1}$ would trigger on fraction 2 the emptying sequence would be: $AV_2 : \{IC_{2\_1}, IC_{2\_2}\}$, $AV_1 : \{IC_1\}$. However, if $IC_4$ would trigger afterwards, then it would include the same inlets on the shared path that was just recently emptied. Its emptying sequence would be: $AV_4 : \{IC_4\}$, $AV_2 : \{IC_{2\_1}, IC_{2\_2}\}$, $AV_1 : \{IC_1\}$. This would be both unnecessary and inefficient.

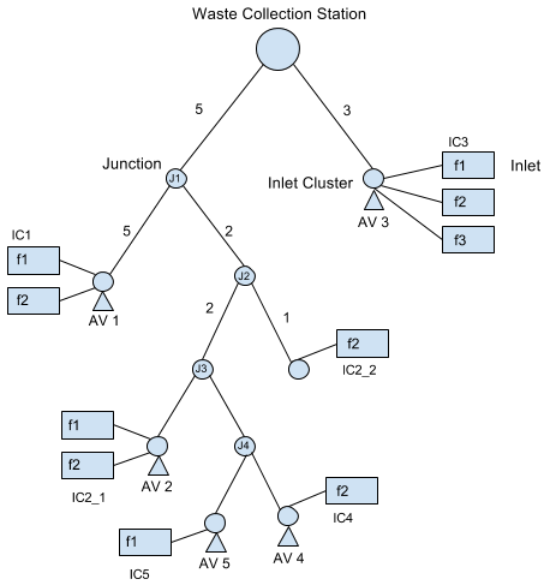


**Figure 4.1:** Example of a deeper system structure.

This function could be used when the system trigger on a high level in an inlet in the system. However, it can not be considered an adequate solution by itself for emptying the entire system. As described in the example, this kind of emptying sequences would be of very low efficiency. In many cases it would also generate long start up and transport times compared to the number of inlets emptied. However, the new parameter $lvl_i$ can be of use in several other scenarios for how the emptying sequence can be executed.

### 4.3.3 Scenario 3: Empty All Inlets With at Least a Low Waste Level

Adding a level check on all inlets would enable emptying sequences which empty only those inlets that needs to be emptied. There are several ways in which the algorithms can be developed to empty those inlets with at least low level, some more efficient than others. Two attempts are described in this section.

#### 4.3.3.1 An Extension of Algorithm 1

For this scenario, a simple extension of Algorithm 1 could be sufficient. In this case, on Line 9 in Algorithm 1 the if-statement should be updated to include the new level constraint. The new line would look as follows:

**if** $av_i$ *is not in emptiedAVs and* $hasInd(av_i, fraction)$ **then**,

where the first check, for controlling if the AV has been emptied before, is followed by a second check for controlling if there is indication that the AV should be emptied for the given fraction, i.e. a low level has been reached in one of the inlets belonging to the AV. The *hasInd*() function is presented in Algorithm 3, where $MAX\_LEVEL$ is a constant for the maximum level an inlet can have before it needs emptying. For the scenario of emptying all inlets with at least low level, the $MAX\_LEVEL$ is set to low level, i.e. somewhere between 50-75 %. The map $AVIndicator$ maps every AV, that has already been checked, to an indication (true or false) if they should be emptied or not. The purpose of this map is to avoid looping over all inlets for the same AV more than one time.

**Algorithm 3:** Check if an AV has level by checking the level of the associated inlets

```
1  IC_i = (id, L_i, l_i, parent_i, av_i, inlets_i);
2  J_i = (id, L_i, l_i, parent_i, child_i^l, child_i^r, d_i^l, d_i^r);
3  A_i = (id, L_i, clusters_i, List_i);
4  I_i = (id, fraction_i, lvl_i);
5  Map AVIndicator;
6  Function hasInd (AV_i, fraction)
7      if AVIndicator contains AV_i then
8          return AVIndicator.getValueFor(AV_i);
9      foreach ic in clusters_i do
10         foreach i in ic.inlets do
11             if i.fraction == fraction and i.lvl > MAX_LEVEL then
12                 add true to AV_i in AVIndicator;
13                 return true;
14     add false to AV_i in AVIndicator;
15     return false;
```

The change does not work as intended. The algorithm will not always empty the inlets furthest away first and thereby not choose the order which is most energy efficient. As an example, assume a tree with two subtrees where the left subtree is deeper than the right. One shallow inlet in the left subtree has level that should be emptied, but the right subtree may have several inlets with level. The algorithm would still be forced to start in the deeper of the two subtrees, even if all inlets in it do not need to be emptied. This could be avoided by always running Algorithm 1 on a tree where all inlets should be emptied. Therefore, we introduce an algorithm for building a new tree.

#### 4.3.3.2 Building an Emptying Tree

When the emptying of the system is supposed to be restricted to only include the inlets that needs to be emptied, the procedure of choosing inlets and emptying them are separated. Therefore, a function $buildTree()$ is implemented. Similar to $processSubtree()$ in Algorithm 1, $buildTree()$ is called with the original root node and a fraction as arguments, to recursively decide which inlets to include in the new tree. The new tree is a modified version of the original tree, including only desired inlets.

Modifications of the original tree is done by removing junctions when one of its two child nodes does not need to be emptied. As an example, in a scenario where fraction 1 is emptied in Figure 3.2, $IC_{2\_2}$ is not needed, and therefore, junction $J2$ is removed and $IC_{2\_1}$ is made the right child of $J_1$. The original length in the path from $J_2$ to $IC_{2\_1}$ is retained in the new edge between these nodes.

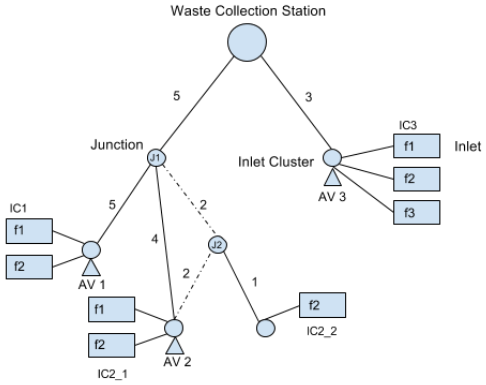


**Figure 4.2:** Example of function $buildTree()$.

In Algorithm 4 the recursive function $buildTree()$ is described. For every inlet cluster that reached, $hasInd()$ from Algorithm 3 is called to check if the cluster should be included in the new tree, in which case it is returned. Otherwise, null is returned to indicate that the node should not be included. For every junction $V_i$, the function is called for both child nodes and the returned values are saved in $v1$ and $v2$, which represents the new child nodes of $V_i$. If the values of both child nodes are null, neither them nor the current junction should be included in the tree, hence null is returned. If only one of the child nodes should be included the junction is not needed and its parent and child should be re-assigned as described in the example above. However, the depths, $d_i^l$ and $d_i^r$, first have to be updated to match the new subtree of the junction. This is done through the function $updateJunctionDepth()$, that takes the start junction to update as a first argument and a node to recurse over as a second. This function is also described in Algorithm 4.

**Algorithm 4:** Build a tree of all inlets with indicator

```
1  IC_i = (id, L_i, l_i, parent_i, av_i, inlets_i);
2  J_i = (id, L_i, l_i, parent_i, child_i^l, child_i^r, d_i^l, d_i^r);
3  A_i = (id, L_i, clusters_i, List_i);
4  I_i = (id, fraction_i, lvl_i);
5  Function buildTree (V_i, fraction)
6      if V_i is I_i then
7          if hasInd(av_i, fraction) then
8              return V_i;
9          else
10             return null;
11     else
12         v1 = buildTree(child_i^r, fraction);
13         v2 = buildTree(child_i^l, fraction);
14         if v1 and v2 is null then
15             return null;
16         else if v2 is null then
17             if V_i is root then
18                 set child_i^l = null;
19                 return V_i;
20             parent_i.child^{V_i} = child_i^r;
21             child_i^r.l_i+ = l_i;
22             return child_i^r;
23         else if v1 is null then
24             if V_i is root then
25                 set child_i^r = null;
26                 return V_i;
27             parent_i.child^{V_i} = child_i^l;
28             child_i^l.l_i+ = l_i;
29             return child_i^l;
30         else
31             updateJunctionDepth(V_i, V_i);
32             return V_i;
33 Function updateJunctionDepths(start, V_i)
34     if V_i is I_i then
35         return l_i;
36     else
37         dLeft = updateJunctionDepths(child^l);
38         dRight = updateJunctionDepths(child^r);
39         deepest;
40         if dRight ≥ dLeft then
41             deepest = dRight;
42         else
43             deepest = dLeft;
44         if V_i is start then
               // Finished - Set new depths
45             d_i^l = dLeft;
46             d_i^r = dRight;
47             return deepest;
48         else
49             return l_i + deepest;
```

### 4.3.4 Scenario 4: Empty Based on Statistics

By using the same data set, originating from disposal logs, as for the $lvl_i$ parameter in $I_i$, a probable level of waste for an inlet within a certain time span can be estimated. The statistics can be used to decide which inlets should be added to the emptying sequence. If an inlet is probable to fill up in the coming time span, it could be a better choice to add it to the sequence now, even if its current level is not greater than $MAX\_LEVEL$. This scenario is easily added by making small changes in already stated algorithms. In Algorithm 3 at Line 11, a check on probable level could be added if statistics are available. The new line looks as follows:

**if** $i.fraction == fraction$ and ($i.lvl > MAX\_LEVEL$ or $i.probableLvl > MAX\_LEVEL$) **then**,

where *probableLvl* is calculated by fetching the average number of disposals for an inlet in a given time interval. Since the distribution of disposals differs not only by time of day, but also by day of week, the probable level should be based on statistics for the day of week. The distribution also differs by month. However, since the system has been growing a lot, with more inlets connecting each month, the data used should be from no more than one or two years back to give a realistic result. Thus, there is not enough data from separate months to take that into consideration when calculating an average.

### 4.3.5 Scenario 5: Empty Using Statistics, Minimum and Maximum Level

The statistics described are a good foundation for choosing which inlets should be included in an emptying sequence. By also having more than one level which can induce different ways of choosing inlets, it is possible to make the sequence more effective and possibly minimize the risk of inlets getting filled up. The constant $MAX\_LEVEL$ has already been introduced and now a new constant is introduced, $MIN\_EMPTY\_LEVEL$. The new constant can be used to determine which other inlets should be emptied than the triggered one(s), e.g. if one inlet of fraction 1 has a greater level than $MAX\_LEVEL$, then all inlets of fraction 1 which have reached $MIN\_EMPTY\_LEVEL$ will also be emptied.

Two functions are described in Algorithm 5, one to calculate the level in each inlet and one to select which inlets to empty. The first function, *updateLevels*(), is called with the argument *disposalsPerInlet*, which is a map containing lists of all disposals made in a time interval mapped to their respective inlet. The function loops over all inlets where disposals have been made, and for each inlet calculates the new level. The level added is computed through the number of disposals made times the volume of a waste bag, which is then divided by the capacity of the inlet. If an inlet has exceeded maximum level it is added to $inletsWithMaxLevel$ and if the inlet is overfull, it is added to $inletsOverLimit$. If $inletsOverLimit$ contains any elements, the emptying has failed and the solution is not feasible.

The second function, $setInletsToEmpty()$, computes which inlets and fractions that should be emptied. The function is called with the argument $inletsWithMaxLevel$, which is the list returned from $updateLevels()$ containing inlets that have reached maximum level in this time interval. It loops over all inlets in the system and checks their current level and probable level. All inlets that have a level or a probable level that indicates it might need emptying are considered for the emptying sequence. If they also belong to a fraction that is set to be emptied, they are added to the list $inletsToEmpty$. If they do not belong to a fraction that is already set to be emptied, there are two cases available. Either they have a probable level above maximum level, or they only have reached the minimum empty level. In the first case this is an inlet with a risk of filling up before the next emptying sequence and in that case the inlet is added to $inletsToEmpty$. Furthermore, the new fraction needs to be added to $fractionsToEmpty$ and change $fractionAdded$ to $true$ for the fraction to declare that it has been added. In the second case the inlet is added to the list $overMinF^{i.fraction}$, where all inlets that have reached minimum level are stored in case the fraction will be added to the emptying sequence.

After looping through and sorting all inlets, we loop over all fractions, to add those inlets with minimum level to $inletsToEmpty$ in the case where their fraction has been added.

**Algorithm 5:** Find and save all inlets which should be emptied

```
1  IC_i = (id, L_i, l_i, parent_i, av_i, inlets_i);
2  J_i = (id, L_i, l_i, parent_i, child_i^l, child_i^r, d_i^l, d_i^r);
3  A_i = (id, L_i, clusters_i, List_i);
4  I_i = (id, fraction_i, lvl_i);
5  List inletsToEmpty;
6  List inletsOverLimit;
7  Function updateLevels(disposalsPerInlet
8      addedVolume, newLevel, oldLevel;
9      fraction;
10     List inletsWithMaxLevel;
11     foreach i in disposalsPerInlet do
12         oldLevel = i.lvl;
13         addedLevel = (i.nbrOfDisposals * BAG_VOLUME) /
             MAX_VOLUME; newLevel = addedLevel + oldLevel;
14         if newLevel > MAX_LEVEL then
15             if newLevel ≥ 1 then
16                 add i to inletsOverLimit;
17             add i to inletsWithMaxLevel;
18         i.lvl = newLevel;
19     return inletsWithMaxLevel;
20 Function setInletsToEmpty(inletsWithMaxLevel)
21     List fractionsToEmpty;
22     List overMinF^1, overMinF^2, overMinF^3;
23     Boolean[] fractionAdded = {false, false, false};
24     add all fractions of inlets in inletsWithMaxLevel to fractionsToEmpty;
25     foreach i in I do
26         if i.lvl ≥ MIN_EMPTY_LEVEL or probableLevel > MAX_LEVEL
            then
27             if fractionsToEmpty contains i.fraction then
28                 add i to inletsToEmpty;
29             else if probableLevel > MAX_LEVEL then
30                 add i to inletsToEmpty;
31                 add i.fraction to fractionsToEmpty;
32                 fractionAdded[i.fraction] = true;
33             else
                   // The inlet level is over minimum, but the probable level is not
                      over maximum
34                 add i to overMinF^{i.fraction}
35     foreach fraction in F do
36         if fractionAdded[fraction] then
               // If the fraction should be emptied, add all inlets with level ≥
                  MIN_EMPTY_LEVEL to inletsToEmpty
37             add all elements from overMinF^{fraction} to inletsToEmpty;
38     return fractionsToEmpty;
```

The function *hasInd*() in Algorithm 3 also needs to be adjusted to this change and the final version of the if-statement on Line 11 will look the following way:

**if** $i.fraction == fraction$ and *inletsToEmpty* contains $i$) **then**,

where the level check is now replaced by a control for if the inlet is in the list of all inlets that should be emptied.

The procedure in scenario 5 can be explained in a number of steps. First, the system checks its levels at given times by calling the method *updateLevels*(). Then, if any inlets have been found with a high level, the function *setInletsToEmpty*() will be called with these as an argument and assign which fraction(s) and the list of the inlets that should be a part of the sequence. For each fraction, a new tree with this list will be built with the function *buildTree*(). Lastly, *processSubtree*() will be called with the new tree as an argument, and an emptying sequence will be built.

### 4.3.6 Time Complexity

Since the structure of the system is a tree where all nodes are visited, the time complexity of visiting all nodes and all edges in the tree is $O(n_1 + n_2)$, where $n_1$ is the number of nodes in the tree and $n_2$ is the number of inlets in the system. This complexity holds for the worst-case scenario, function *processSubtree*(), which iterates over the entire tree and empties it. The complexity is the same for all the other algorithms.

### 4.3.7 Simulation

The scenarios are simulated in a Java program on a model of Plant 1. A map in scale 1:2000 over the area where Plant 1 is situated has been used to measure the distance from the root node to each cluster of inlets. The accuracy of the calculated lengths are therefore not exact but should give close enough values for the simulator. The exact times for each step in the procedure are not available, but the average time needed has been consulted with Envac to make the simulations as close to reality as possible. A discharge valve needs 4 seconds for opening and closing, it is opened for 10 seconds to make sure all waste is released from the inlet. An air valve also needs 4 seconds for opening and closing. The speed for transporting residual and plastic waste is $10m/s$, and $8m/s$ for paper. Paper is more prone to get stuck in the pipes and is therefore emptied at a slower speed. When running the algorithms, an emptying sequence prints which inlets are emptied, total time for the process and how much time was needed for each step.

# 5

# User Feedback Prototype

A digital prototype has been developed to act as a tool to communicate feedback to the users of the AVWC system. In the modern society, digital communication is easily accessible. Therefore, the feedback is intended to be communicated via software in a manner that fits the entire spectra of users. This chapter describes how user feedback is communicated in the Envac system and how the prototype was implemented and which frameworks were used.

## 5.1 User Feedback in the Envac System

In a traditional waste collection system, there are garbage cans for each fraction available, where the user opens the lid or hatch to dispose of their waste in that container. The interaction between users and the Envac system in residential areas is very similar to the ordinary waste collection system, as can be seen in Figure 5.1. The garbage cans in the traditional system and the inlets in Envac's system are each others counterparts. In areas where each household manages its own garbage can(s), like many house owners do in Sweden, the feedback is straightforward. They receive visual feedback while filling the garbage can and the service cost of their waste collection is based on the weight of the waste upon emptying [15]. In contrast, areas with shared waste collection, for instance apartment blocks in the city, there is limited feedback on the individual waste generation. Waste is collected without any involvement of the households and the service fee is included in the rent rather than charged based on individual use.

**Figure 5.1:** Inlets in an Envac System [1]. The colors of the hatches represent different fractions.

## 5.2 Application Development

According to Statistics Sweden 73% of the Swedish population have access to a smartphone [24]. Developing an app for smartphones is deemed to be effective to reach a broad user group. However, to avoid the need to develop for both Android and iOS, the choice of platform is a web application since it is easily adapted to mobile devices. Furthermore, a web application makes it possible for users who do not own a smartphone to use the application on a computer.

### 5.2.1 Requirements and Design

As a first step, a list of milestones was established for the work process for the application. These milestones were given a time span and a deadline for completion. All milestones are listed in table 5.1.

| Milestone | Description |
|---|---|
| 1 | Requirements should be specified |
| 2 | A mock-up of the application UI should be done |
| 3 | The data should be processed and added to a database |
| 4a | The backend should be implemented (supporting at least minimum requirements) |
| 4b | The frontend should be implemented (supporting at least minimum requirements) |
| 5 | The prototype should be finished |

**Table 5.1:** Table of Milestones

A list of requirement was specified and then prioritized after what usage was considered to be of most importance. The main functionality was decided to be the visual representation of the users interaction with the system, i.e. what fraction of waste they have disposed and when. These requirements are viewed as minimum for the prototype to be finished. When the time devoted for the prototype was over, any unfulfilled requirement that was not considered minimum was discharged. All requirements are listed in table 5.2.

| Prio | Requirement |
|---|---|
| 1 | A user should be able to view:<br>- previous disposals<br>- the total amount of waste and for a specific time span<br>- the number of disposals of different fractions |
| 2 | A user should be able to view graphical statistics over their disposals and choose between layouts and visualizations of their liking |
| | A user should get positive feedback when improving |
| 3 | A user should be able to see their position in contrast to anonymous neighbours |
| | A user should get encouragement to improve (without tons of notifications) |
| | A user should be able to receive information about the system that they are a part of e.g. the fractions available, a map and so on |
| | A user should be able to change their settings |
| | A user should be able to contact the company<br>- regarding the application<br>- regarding their waste |
| 4 | A user should be able to get information about recycling and its benefits |
| | A user should be able to add a plan for reducing their amount of waste/start recycling more |
| | A user should be able to see their environmental impact |
| 5 | A user should be able to calculate their invoice |
| | A user should be able to see if an inlet is closed for emptying |
| | A user should be able to get information of which inlet in their neighbourhood is most optimal (if an inlet is scheduled to soon be emptied or is more full than another) for disposal at a given time |

**Table 5.2:** Table of Requirements

Regarding design of the application, it was chosen to implement according to mobile application convention to give a sense of familiarity for the user. The first view displays the statistics of the users monthly waste generation and contains some trivia on benefits of recycling or similarly encouraging information. An early paper prototype can be seen in figure 5.2. The navigation is done in a left side drawer which emerges when the user presses a button with three bars. The overall information that is presented to the user in the application is their previous history, their personal account, a top list for adding competitive feedback and information about the area and what inlets that are close by. The top list is intended to vary its information depending on the user's habits. If the user is, in fact, amongst the top performing household's in the area, the information will encourage to perform even better. If the user is in the lower half in performance on the other hand, the encouragement will be in comparison of the user's history, e.g. *"You recycled 5% better this month than the last. Good job!"*.

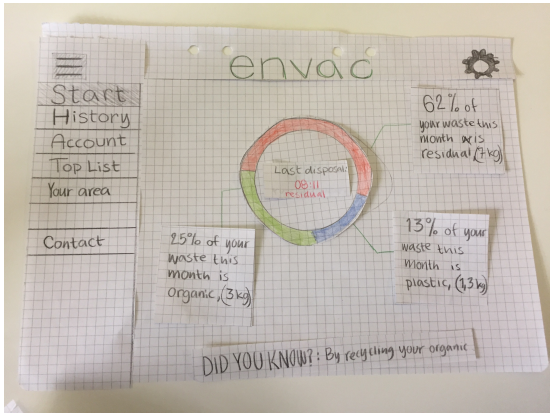


**Figure 5.2:** A sketch of the user feedback prototype.

### 5.2.2 Implementation

The framework used to develop the frontend of the web application is Angular 2, the backend is written in JavaScript using Node.js and the database is built in MySQL. The database contains data of all users connected to Plant 1 and all disposals done during 2016. The backend handles simple requests to the database from the frontend such as listing all disposals made by a certain household for a given time span.

# 6
# Results and Discussion

The first goal of this thesis has been to explore the possibility of optimizing the emptying process in the Envac AVWC system, and thereby reduce the total energy use in the system. The second goal has been to investigate the possibility of creating incentive for individual users to change their waste generation behavior. In this chapter, the optimization results of the simulation program and the results of the feedback study are presented and discussed.

## 6.1 Simulation Outcome

All simulations presented in this chapter are conducted over the total number of disposals in Plant 1 during January 2017. For the simulations, each fraction has been assigned a number. Residual waste is denoted by 1, plastic by 2 and paper by 3. The desired outcome of the simulations is a lower total run time than the current in Plant 1. The results are compared both to the current run time and that of the worst-case scenario, which is to empty the entire system in pre-scheduled sequences in a given time interval. Figure 6.1 shows all disposals made during 2016 per fraction. The figure suggests that residual waste is more than twice as frequent as plastic, and more than five times as frequent as paper. However, as paper is more prone to get stuck in the pipes, it needs to be emptied more often. Therefore, the worst case scenario will be modelled to empty residual waste twice as often as plastic and paper.

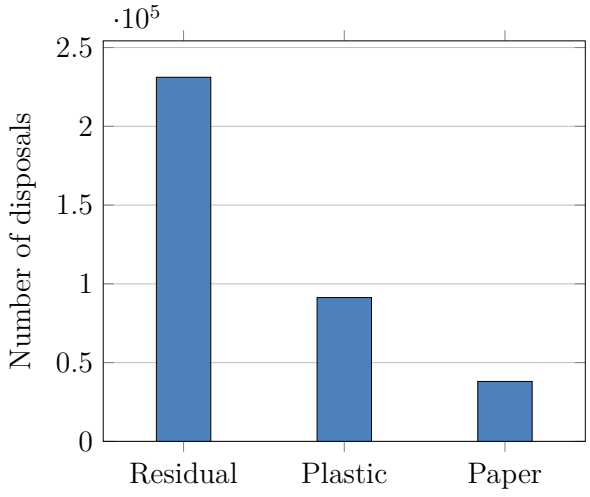


**Figure 6.1:** Number of disposals per fraction in 2016

Table 6.1 presents actual data of Plant 1 run time in January 2017. "DV emptyings per minute" implies how many discharge valves in inlets have been opened on average per minute. It is a measurement of efficiency, a higher value means more inlets have been emptied per minute. However, if the amount of waste in an inlet is low, it is not necessarily more efficient to include it in a sequence; but it will result in a higher value on DV Emptyings per minute.

| **Fraction:** | **Residual (1)** | **Plastic (2)** | **Paper (3)** | **Total** |
|---|---|---|---|---|
| Time (%) | 59.8 | 18.1 | 22.1 | 100 |
| Accumulated Time (s) | 262509 | 79369 | 97130 | 439008 |
| Accumulated Time (h) | 72,92 | 22,05 | 26,98 | 121,95 |
| Energy (%) | 63.2 | 16.8 | 20 | 100 |
| Accumulated Energy (kWh) | 3760 | 1003 | 1191 | 5954 |
| DV Emptyings | 3305 | 749 | 738 | 4792 |
| DV Emptyings per minute | 0.755 | 0.566 | 0.456 | 0.654 |

**Table 6.1:** Table of system run time and energy use in January 2017, from Envac.

### 6.1.1 Simulator Setup

Table 6.2 presents statistics for the emptyings of each fraction in Plant 1 during January 2017. No sequence to empty all inlets of fraction 2 has been executed, only sequences where inlets with at least low level are emptied are executed on this fraction. Figure 6.1 indicates, residual waste are by far the greatest contributor of volume in the system, hence most emptying processes transport residual waste. Notable is that the number of inlets may have increased during the month, thus the total number of inlets emptied is not divisible by the number of inlets in Table 6.3.

| **Scenario** | **Fraction** | **#Inlets** | **Time (s)** | **Inlets/min** |
|---|---|---|---|---|
| Empty all | 1 | 1453 | 100221 | 0.870 |
| | 2 | 0 | 0 | 0 |
| | 3 | 77 | 5390 | 0.857 |

**Table 6.2:** Comparative statistics for one month in Plant 1, from Envac.

There are some factors in Plant 1 that affect the total emptying time. These factors are not consistent, nor are they all known to us. For example, there are different types of inlets and discharge valves in the system and they are of various sizes and do not require the same time for emptying. There are also some emptying sequences where an extra fan needs to be started or the emptying might be paused for some reason, which adds more time to the sequence. To compensate for the different factors and be able to compare with the data in Table 6.1, extra time needs to be added in the simulation for each inlet that is emptied. The extra time added is called correction factor and several different time values have been tested. The result is presented in Table 6.3. Notable is that while Table 6.2 is over an entire month, Table 6.3 is of a single run only. The scenario of emptying the entire system is

comparable to the procedure currently in use at Envac, therefore obtaining a similar value to theirs regarding inlets/min, is considered desirable. Residual waste is by far the most common fraction in the system and the approximations are mainly based on it, therefore the value for residual waste will be used as benchmark. As Table 6.3 suggests, a correction factor of 19 seconds yields a value of 0.876 inlets/min for fraction 1, compared to 0.870 inlets/min, as seen in Table 6.2. This is regarded satisfactorily similar. All simulations henceforward will use a correction factor of 19 seconds.

| Scenario | Fraction | Correction Factor (s) | #Inlets | Time (s) | Inlets/min |
|---|---|---|---|---|---|
| Empty all | 1 | 10 | 62 | 3222 | 1.154 |
| | 1 | 15 | 62 | 3792 | 0.981 |
| | 1 | 19 | 62 | 4248 | 0.876 |
| | 1 | 20 | 62 | 4362 | 0.853 |
| | 2 | 10 | 47 | 2802 | 1.006 |
| | 2 | 15 | 47 | 3297 | 0.855 |
| | 2 | 19 | 47 | 3693 | 0.764 |
| | 2 | 20 | 47 | 3792 | 0.744 |
| | 3 | 10 | 39 | 2678 | 0.874 |
| | 3 | 15 | 39 | 3133 | 0.747 |
| | 3 | 19 | 39 | 3497 | 0.669 |
| | 3 | 20 | 39 | 3588 | 0.652 |

**Table 6.3:** Table of simulating an emptying of entire system of a fraction with various correction factors.

When simulating the scenarios from Section 4.3, the system is scanned in specified time intervals. In most scenarios, various time intervals are used between scans to generate comparable outcome. All disposals since the previous scan are converted into volume. Depending on the current scenario, the simulator makes decisions based on the scenario specific constraints on levels and statistics.

### 6.1.2 Scenario 1: Empty All Inlets

In Table 6.4 the results of various time intervals attempted for scenario 1 are presented. This is also considered to be the worst-case scenario for the system. The column "Comparison (%)" shows the difference in percentage between this running time and the total time presented in Table 6.1.

In this scenario the residual waste is emptied in every time interval and the other two fractions alternately in every other interval. It is likely that an emptying sequence requires a long run time when all inlets are emptied. In the shorter intervals, an emptying sequence might need longer time than the actual time interval. If an emptying sequence has been initiated in an interval, the next emptying sequence can

not run until the sequence before has finished, even if it is longer than the time interval between emptyings. In these cases, the simulator postpones the next interval to when the previous sequence has finished.

| Scenario | Interval (min) | #Inlets | Time (s) | Comparison (%) | Inlets/min | #Overfull Inlets |
|---|---|---|---|---|---|---|
| Worst case | 120 | 35801 | 2633649 | 599.91 | 0.816 | 0 |
| | 180 | 25620 | 1884558 | 429.28 | 0.816 | 5 |
| | 400 | 11550 | 735964 | 167.64 | 0.816 | 53 |
| | 520 | 8921 | 656407 | 149.52 | 0.815 | 96 |

**Table 6.4:** Table of worst-case scenario modelled over various time intervals, for all fractions.

The worst-case scenario can result in a never-ending emptying sequence, as for the interval of 120 minutes where the total run-time over a month is 30.5 days. The first entries in the data set are from 12:30, 1 January 2017, meaning that the shortest intervals in Table 6.4 would in fact be running constantly. Since 120 minutes is the approximate time for emptying all inlets of fraction 1 and fraction 3, which happens in every other time interval, an interval of 120 minutes is the same as running a new emptying sequence as soon as the previous one is over. Thus, the outcome is expected.

### 6.1.3 Scenario 2: Empty Triggered Inlet

Scenario 2 was not fully simulated as it was deemed to be of low efficiency, as described in Subsection 4.3.2. The concepts of the scenario are however integrated in the following scenarios.

### 6.1.4 Scenario 3: Empty All Inlets With at Least a Low Waste Level

The simulations of scenario 3 were implemented with several time intervals for different values to act as low level. The results are presented in Table 6.5. If any inlet has a level according to the scenario, an emptying sequence will be initiated. For example, if the levels are checked in 30 minutes intervals, with a triggering level of 50%, a total number of 3861 inlets will have been emptied. In this particular example, 0 inlets have become overfull between two level scans. In the same manner as in the worst-case scenarios, an emptying sequence will postpone the next system check if it is longer than the current interval. However, in the simulations in Table 6.5, encountered postponements are no longer than a few minutes.

The results in Table 6.5 shows that when the total run time is lower than Envac's, $439008s$, there are multiple occurrences of overfull inlets. There are only two scenarios where no inlets are overfull and the run time is lower than Envac's. The first is in the scenario when the level is 50% and the time interval is 60 minutes, where the total run time is $35min$ lower than Envac's. The second case is when the level is 60% and the time interval is 30 min, where the total run time is $39min$.

Both of the cases results is an approximate improvement of 0.5% in total kWh per month, which is a very small gain that might be due to margin of error.

| Scenario | Interval (min) | #Inlets | Time (s) | Comparison (%) | Inlets/min | #Overfull inlets |
|---|---|---|---|---|---|---|
| Empty if level ≥50% | 10 | 3942 | 619087 | 141.02 | 0.382 | 0 |
| | 30 | 3861 | 528659 | 120.42 | 0.438 | 0 |
| | 60 | 3694 | 436924 | 99.53 | 0.507 | 0 |
| | 90 | 3538 | 384980 | 87.69 | 0.551 | 3 |
| | 120 | 3430 | 354040 | 80.65 | 0.581 | 6 |
| Empty if level ≥60% | 10 | 3150 | 506733 | 115.43 | 0.373 | 0 |
| | 30 | 3064 | 436627 | 99.46 | 0.421 | 0 |
| | 60 | 2986 | 377903 | 86.08 | 0.474 | 2 |
| | 90 | 2905 | 334157 | 76.12 | 0.522 | 10 |
| | 120 | 2823 | 305822 | 69.66 | 0.554 | 21 |
| Empty if level ≥70% | 10 | 2757 | 449368 | 102.36 | 0.368 | 0 |
| | 30 | 2709 | 396172 | 90.24 | 0.410 | 4 |
| | 60 | 2622 | 340644 | 77.59 | 0.462 | 7 |
| | 90 | 2572 | 307626 | 70.07 | 0.502 | 22 |
| | 120 | 2482 | 278567 | 63.45 | 0.535 | 38 |
| Empty if level ≥80% | 10 | 2452 | 400371 | 91.2 | 0.367 | 2 |
| | 30 | 2414 | 353677 | 80.56 | 0.410 | 5 |
| | 60 | 2359 | 311463 | 70.95 | 0.454 | 19 |
| | 90 | 2306 | 277558 | 63.22 | 0.498 | 53 |
| | 120 | 2259 | 259772 | 59.17 | 0.522 | 77 |

**Table 6.5:** Table of simulation over January 2017 with level constraints, for all fractions.

### 6.1.5 Scenario 4: Empty Based on Statistics

Table 6.6 shows the results of simulating the scenario where an inlet is emptied if its level or probable level is greater than or equal to maximum level. The probable level is based on statistics over disposals made in 2016 on a specific weekday and hour. This means the probable level is the probability of a specific inlet, on the current weekday in the coming hour, to gain a volume over the maximum level. A year of data is considered to be enough to calculate the probable level and 2016 is the most similar to 2017 in number of inlets and connected households.

The results in Table 6.6 show some scenarios where no overfull inlets occur and the total time is lower than Envacs run time 439008$s$. The case with best run time is when the inlet level is set to 70% and time interval to 30 minutes. This results in 15.3$h$ less running time than Envac's current procedure, which is an approximate improvement of 12,6% in total kWh per month. This is significantly better than in Subsection 6.1.4, and is considered to be an improvement of the current procedure. However, it is possibly affected by margin of error and might not be a large enough improvement in total to be considered a satisfactory solution.

| Scenario | Interval (min) | #Inlets | Time (s) | Comparison (%) | Inlets/min | #Overfull inlets |
|---|---|---|---|---|---|---|
| Empty if level ≥50% | 10 | 3958 | 617220 | 140.59 | 0.385 | 0 |
| or if probable | 30 | 3934 | 520489 | 118.56 | 0.453 | 0 |
| level is ≥50% | 60 | 3938 | 458375 | 104.41 | 0.515 | 0 |
| | 90 | 3962 | 409490 | 93.28 | 0.581 | 1 |
| | 120 | 4128 | 422677 | 96.28 | 0.586 | 1 |
| Empty if level ≥60% | 10 | 3199 | 453788 | 103.37 | 0.423 | 0 |
| or if probable | 30 | 3214 | 407009 | 92.71 | 0.474 | 1 |
| level is ≥60% | 60 | 3213 | 363042 | 82.70 | 0.531 | 1 |
| | 90 | 3258 | 337770 | 76.94 | 0.579 | 4 |
| | 120 | 3303 | 328338 | 74.79 | 0.604 | 7 |
| Empty if level ≥70% | 10 | 2761 | 442794 | 100.86 | 0.374 | 0 |
| or if probable | 30 | 2778 | 383749 | 87.41 | 0.434 | 0 |
| level is ≥70% | 60 | 2771 | 332193 | 75.67 | 0.500 | 2 |
| | 90 | 2797 | 303033 | 69.03 | 0.554 | 9 |
| | 120 | 2838 | 288413 | 65.70 | 0.590 | 14 |
| Empty if level ≥80% | 10 | 2450 | 403975 | 92.02 | 0.364 | 2 |
| or if probable | 30 | 2439 | 353915 | 80.62 | 0.413 | 8 |
| level is ≥80% | 60 | 2457 | 304151 | 69.28 | 0.485 | 11 |
| | 90 | 2440 | 277501 | 63.21 | 0.528 | 14 |
| | 120 | 2484 | 260569 | 59.35 | 0.572 | 32 |

**Table 6.6:** Table of simulation over January 2017 with level constraints and statistics for coming time interval.

### 6.1.6 Scenario 5: Empty Using Statistics, Minimum and Maximum Level

Table 6.7 presents the results of scanning the system every minute and initiating an emptying sequence when triggered by one or several inlets with maximum level. The sequence includes all inlets of the same fraction as the ones triggered, which have at least minimum level or have a probability of reaching maximum level within an hour. If an inlet of another fraction has a probability of reaching maximum level within an hour, it will also be included in the emptying sequence. Also all inlets of that fraction which have at least minimum level will be included. In this scenario, the simulator checks inlet levels every minute and bases its further decisions on that data.

The outcome of Scenario 5 shows a lower total time than Envac's 439008$s$ for all cases. The best time of 273736$s$ with no overfull inlets is achieved when maximum level is 80% and minimum level is 50%. This is 45.91$h$ less than Envac's, and results in an approximate improvement of 37.6% kWh per month. This outcome is, even with margin of error, considered as a rather beneficial improvement.

| Scenario | Max Level | Min Level | #Inlets | Time (s) | Comparison (%) | Inlets/min | #Overfull inlets |
|---|---|---|---|---|---|---|---|
| Empty all with min level | 80 | 40 | 3445 | 278523 | 63.44 | 0.742 | 0 |
| if triggered by max level | 80 | 50 | 3131 | 273736 | 62.35 | 0.686 | 0 |
| | 80 | 60 | 2769 | 280280 | 63.84 | 0.593 | 0 |
| | 80 | 70 | 2616 | 307800 | 70.11 | 0.510 | 0 |
| | 70 | 40 | 3669 | 313293 | 71.36 | 0.703 | 0 |
| | 70 | 50 | 3324 | 307331 | 70.01 | 0.649 | 0 |
| | 70 | 60 | 2963 | 343195 | 78.18 | 0.518 | 0 |
| | 60 | 40 | 3909 | 348943 | 79.48 | 0.672 | 0 |
| | 60 | 50 | 3575 | 360769 | 82.18 | 0.595 | 0 |

**Table 6.7:** Table of simulation of emptying entire system of minimum level when triggered by an inlet at maximum level.

This case has also been tested on several months of data from 2016 with a very similar to the results in Table 6.7. However, no comparisons to Envacs are possible since there is no data available on the actual running times for the system in those months. Hence, no conclusions can be drawn from these tests except that it is promising that the occurrences of overfull inlets were almost non-existing. Overfull inlets are unlikely to happen when the system is scanned every minute and these cases might occur due to anomalies in the data. One other reason that could lead to overfull inlets is a lower minimum level. The emptying sequence will then include more inlets and thereby need longer time to finish. It is therefore possible for inlets to fill up while the sequence is running.

### 6.1.7 Analysis

There are many circumstances that might affect the accuracy of the outcome described in the sections above. The values used in the simulations are in many cases approximated, averages or based on the lengths from the map of the area. The last being, even if meticulously, measured by hand could result in an error of 2 meters in reality even if the mistake in measurement only is off by 1 millimeter. The time correction factor used is not exact and only the emptying data for residual waste was used as benchmark. Thus, the value might not be accurate for the other fractions. Since the value of inlets/min for paper was lower than the benchmark and plastic was not known at all, the correction factor is likely to have resulted in longer running times for these fractions. Despite that, the results are in many cases faster than the current run time. Also, the data used to calculate levels in the system is based on the disposals during a month, and not actual levels in the inlets. It is possible that there are logged disposals that did not increase the waste level, or that the volume of the disposed waste is lower or higher than the calculations of an approximated waste bag. There is no data available that can confirm these calculations. Exact data has been the most limiting factor during this work.

Despite the possible sources of error, the result found in Scenario 5 is a large enough improvement to be considered successful, compared to the current procedure. By scanning the system frequently more insight in the actual waste levels are gained. Compared to the current procedure, using pre-scheduled emptying sequences, the on-

line programming approach used in this solution enables the emptying to be adapted to current need. The simulation indicates that a dynamic procedure results in the system being emptied only when needed, and thereby generating shorter emptying sequences which include less inlets with no or low level.

Scanning every minute almost simulates an online scheduling algorithm. However, it is not an online algorithm in the strong sense, since once an emptying sequence is initiated, the details of the sequence are based on the information at the start of the sequence and no changes are made during the execution of the sequence.

Each emptying sequence is done in a Dynamic Programming fashion, that is, the sequence of the entire tree builds on sequences for its subtrees.

In terms of time complexity, the emptying sequences for an entire tree runs in $O(n_1 + n_2)$ where $n_1$ is the number of nodes in the subtrees and $n_2$ is the number of inlets. Thus the algorithms can be considered optimal in terms of time complexity since obviously, every node and every inlet has to be visited.

## 6.2 Feedback Prototype

The prototype resulted in a web application with a simple user interface that focuses on presenting information to the user. The result may be described as a shell rather than a fully working product. The stated minimum requirements are however fulfilled. The minimum requirements were:

- A user is able to view:
  - previous disposals
  - the total amount of waste and for a specific time span
  - the number of disposals of different fractions

The resulting interface can be viewed in Figure 6.2. The first view is a visual representation of the users waste generation of the previous month. The various colors in the chart represent different fractions. In the menu on the left-hand side, the user can navigate through the application.

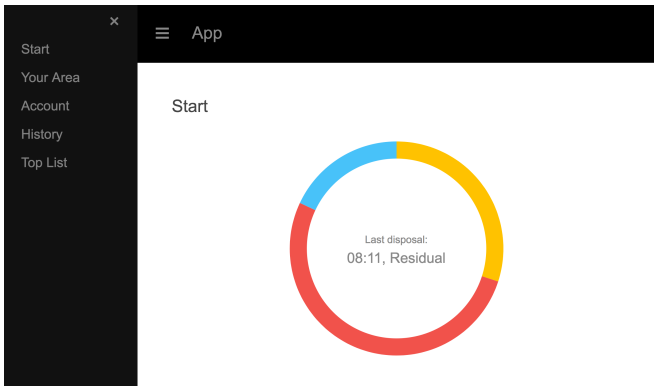


**Figure 6.2:** Start view of the prototype, with emerged side menu.

The feedback given should be displayed differently depending on the user's possible receptivity to the information. A user that already performs in the higher percentage is likely to feel a need to improve even more, while users that performs in the lower percentage is likely to feel like their efforts are pointless if they do not improve [7]. For example, a high performing user could receive competitive phrases such as: *You recycle better than 75% of your neighborhood.* Users that perform worse would get encouraging feedback based on their previous performance, such as: *You recycled 5% better this month than the last. Good job!*

For the results of the prototype to be evaluated, it would need user testing to prove whether the design is beneficial or not. This was deemed to be outside of the scope, hence no such testing has been done. The applications ability to fulfill its purpose is therefore not determinable at this state.

## 6.3 Future Work

The identified possible improvements are depending on a more detailed indication of level in each inlet. The possibility of exact level measurement could increase the effectiveness even more. With detailed level data available the next step would be to test the methods presented in this thesis on the actual system.

The statistics used in this thesis could be used to interpret the system further. Especially in a few years, when Plant 1 is completed, the statistical data will be more reliable and it will be possible to apply machine learning techniques on the data to optimize the efficiency even more.

The emptying sequences that are described in Section 6.1 are all pre-defined at start. That is, they are not online solutions where inlets can be added during runtime. An emptying sequence can be about one hour long, during which time inlets not included in the sequence can fill up. Therefore, the online paradigm could be beneficial to investigate for scheduling the sequences for more dynamic emptyings and an even more efficient system.

One more thing that could be taken into consideration when deciding at what time the emptyings should be executed is the time of day. At certain hours the energy consumption in the whole community is very high and at some points very low, e.g. in the night [25]. When the consumption is low it is more likely that the energy comes from renewable resources and when it is high it is more likely that non-renewable resources have to be used to meet the demand. Furthermore, the price can be regulated depending on when the demand is high as well. Thus, it would be more economical and sustainable to empty at times when the overall energy consumption is low.

The prototype could be further implemented with all requirements. Furthermore, such an application would require user testing before launch. The purpose of the user feedback application was to encourage users to be more thoughtful in their waste

generation behavior. An application giving feedback of their own performance is one way of creating incitement for improvement, but there might be surrounding factors that can help influence a behavior that is beneficial for the system. A more even distribution of waste among the inlets in a cluster is one example of how the system could be emptied more efficiently based on how the users have interacted with it. Inlets could give visual feedback to users such as colored lights indicating which inlet to preferably dispose of their waste in to balance the levels. An even distribution would result in more waste emptied per emptying sequence and a lower risk of having one specific inlet frequently reaching a critical level.

The possibility of handling your waste cost by weighing your disposals is a great chance for households to manage their expenses. However, this would need reliable equipment that cannot be interfered with. Given that the measurements is done automatically, the reliability must be high such that there are no inaccuracies.

## 6.4 Ethical Views

This section will discuss possible rebound effects and the ethical dilemmas regarding user integrity when collecting data on users.

### 6.4.1 The Rebound Effect and Negative Side-Effects

Developing for a reduced environmental impact can be complex. The discussion of rebound effects [26] is common when dealing with environmental issues, changes and its effects. The rebound effect needs to be taken into consideration to know if the goal is actually achieved. If the system is optimized to use less energy by scheduling the emptying sequence there might be some other part of the system that is affected and starts to consume more energy. Another scenario is that the system might be used more if it is more effective and the outcome of saved energy will be zero.

The rebound effect could also be applied to the effect of the user feedback, especially if the feedback involves economic drawbacks for the individual. If people have to pay for the garbage they dispose of or start to see it as a competition against their neighbours, they might choose to dispose of their garbage elsewhere instead of reducing the amount of waste they produce. If the waste from households is placed in the municipal waste containers around town, out in the streets or in nature there are several negative effects which need to be tended to. An increase in waste in the municipal waste containers and in the streets would lead to more work and transports. Even though these rebound effects are serious they are only hypothetical. The model of weighing and charging for the amount of garbage produced is already implemented for house owners and does not show that this is an issue [15], but there is one more ethical factor which can be discussed. The amount of waste produced by an individual could be a question of social class if payment is required. While high income households can throw away as much as they like without affecting their economy significantly, a low income household might not be able to do the same. This could possibly lead to the rebound effects discussed before.

### 6.4.2 User Integrity

For the prototype to serve as an incentive for less generation of waste, the ambition is to gather data of individual waste disposal. This would connect each disposed garbage bag to a specific person or household and present the user with a summary of the waste they have generated, in total and for each fraction. The information could serve as a basis for waste collection fees, chart trends in waste and perhaps contribute to encourage reflection on the packaging of everyday products. A household's waste generation could be in competition with neighbors for competitive feedback. If all Envac's systems world wide gathered this information, a fairly large amount of data would be available. Collecting information about users is however not without touching the matter of big data. The subject is especially present in the field of computer science but comes with a wide range of ethical considerations regarding integrity. Should people's every action be surveyed?

Integrity is an important issue, and one could argue that the data collection could still be useful even with certain restrictions and limitations. The information regarding the amount of waste in each inlet should not be connected to any one individual and therefore might be considered as a negligible violation of integrity. The data used in the prototype could possibly be made private for the individual user, whilst an anonymous rating is provided for the competitive aspect. A user knows the position they have, but not the identity of other competitors. In the end, it is about the individual's trust for the company behind the system.

# 7
# Conclusion

Our method can be seen as a scheduling algorithm inspired by ideas from online optimization, since the system is scanned every minute to see if an emptying process should be initiated. Once an emptying process is initiated, the emptying sequence is performed according to dynamic programming principles. The results presented in Chapter 6 show that there is a great possibility of improvement of kWh consumed per month in Plant 1. The best result shows that the monthly kWh usage potentially can be reduced by 37.6%. The solution presented is based on simulations where waste levels are calculated from the number of disposals during January 2017. Therefore, the accuracy of levels is not exact and a margin of error must be considered.

The solution found in this thesis could be tested on the system. However, it is not certain whether the use of disposal data is enough to estimate the inlet levels. Thus, this improvement might rely on more detailed level measurement in each inlet. This may need an exchange of current level sensors, which could be costly. However, the savings from lowered energy use might compensate for such a cost.

For the second objective of this thesis, a simple prototype has been implemented. The prototype is a web application developed in Angular2, JavaScript, Node.js and MySQL. The minimum requirements have been implemented, thus the objectives are considered fulfilled.

The application is intended to alter its feedback content depending on the user's previous behavior. If a user is already well-performing, they are more likely to improve even further when subjected to comparative feedback in relation to others. Users that do not perform very well might on the other hand stop bothering with the same type of feedback. Therefore, the application must vary its feedback content based on the users previous behavior.

Both parts of this thesis are considered important when aiming to lower the total environmental impact in the AVWC system. An improved scheduling of the emptying procedure in the system can affect the efficiency immediately, and the solutions identified in this thesis are one step closer to an optimized system. When giving user feedback, on the other hand, it might take some time before any effect or behavioral change is noticeable and a lot more work needs to be put into this. The work carried out in this thesis regarding user feedback can act as a foundation for future development.

# Bibliography


[1] (2017, Jan.) Envac - sustainable vacuum waste collection systems. [Online]. Available: http://www.envacgroup.com/

[2] H. Chourabi, T. Nam, S. Walker, J. R. Gil-Garcia, S. Mellouli, K. Nahon, T. A. Pardo, and H. J. Scholl, "Understanding smart cities: An integrative framework," in *2012 45th Hawaii International Conference on System Sciences*, Jan 2012, pp. 2289–2297.

[3] L. Giusti, "A review of waste management practices and their impact on human health," *Waste Management*, vol. 29, no. 8, pp. 2227 – 2239, 2009.

[4] D. Hoornweg and P. Bhada-Tata, "What a waste : A global review of solid waste management," in *Urban development series; knowledge papers no. 15.* World Bank, march 2012.

[5] Envac, *Vakuumsystemets historia*, Accessed 2017-01-25. [Online]. Available: http://www.envac.se/om-envac/om-vakuumtekniken/sopsug_vakuumsystemets_historia

[6] C. J. Midden, J. F. Meter, M. H. Weenig, and H. J. Zieverink, "Using feedback, reinforcement and information to reduce energy consumption in households: A field-experiment," *Journal of Economic Psychology*, vol. 3, no. 1, pp. 65 – 86, 1983.

[7] F. W. Siero, A. B. Bakker, G. B. Dekker, and M. T. V. D. Burg, "Changing organizational energy consumption behaviour through comparative feedback," *Journal of Environmental Psychology*, vol. 16, no. 3, pp. 235 – 246, 1996.

[8] C. Fischer, "Feedback on household electricity consumption: a tool for saving energy?" *Energy Efficiency*, vol. 1, no. 1, pp. 79–104, 2008.

[9] R. Béjar, C. Fernàndez, C. Mateu, F. Manyà, F. Sole-Mauri, and D. Vidal, "The automated vacuum waste collection optimization problem," in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, ser. AAAI'12. AAAI Press, 2012, pp. 264–266.

[10] C. Fernàndez, F. Manyà, C. Mateu, and F. Sole-Mauri, "Modelling energy consumption in automated vacuum waste collection systems," *Environmental Modelling & Software 56*, pp. 63–73, 2014, dOI: 10.1016/j.envsoft.2013.11.013.

[11] A. Faruqui, S. Sergici, and A. Sharif, "The impact of informational feedback on energy consumption—a survey of the experimental evidence," *Energy*, vol. 35, no. 4, pp. 1598 – 1608, 2010, demand Response Resources: the {US} and International ExperienceDemand Response Resources: the {US} and International Experience.

[12] L. McCalley and C. J. Midden, "Energy conservation through product-integrated feedback: The roles of goal-setting and social orientation," *Journal of Economic Psychology*, vol. 23, no. 5, pp. 589 – 603, 2002.

[13] I. Vassileva, F. Wallin, and E. Dahlquist, "Understanding energy consumption behavior for future demand response strategy development," *Energy*, vol. 46, no. 1, pp. 94 – 100, 2012, energy and Exergy Modelling of Advance Energy Systems.

[14] J. Froehlich, L. Findlater, and J. Landay, "The design of eco-feedback technology," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '10. New York, NY, USA: ACM, 2010, pp. 1999–2008.

[15] L. Dahlén and A. Lagerkvist, "Pay as you throw: Strengths and weaknesses of weight-based billing in household waste collection systems in sweden," *Waste Management*, vol. 30, no. 1, pp. 23 – 31, 2010.

[16] J. Kleinberg and E. Tardos, *Algorithm Design*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.

[17] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*. New York, NY, USA: Wiley-Interscience, 1988.

[18] G. B. Dantzig, "Origins of the simplex method," in *A History of Scientific Computing*, S. G. Nash, Ed. New York, NY, USA: ACM, 1990, pp. 141–151. [Online]. Available: http://doi.acm.org/10.1145/87252.88081

[19] F. A. Potra and S. J. Wright, "Interior-point methods," *Journal of Computational and Applied Mathematics*, vol. 124, no. 1–2, pp. 281 – 302, 2000, numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377042700004337

[20] R. Dechter, *Constraint Processing*, ser. The Morgan Kaufmann Series in Artificial Intelligence. San Francisco: Morgan Kaufmann, 2003.

[21] T. Achterberg, T. Berthold, T. Koch, and K. Wolter, *Constraint Integer Programming: A New Approach to Integrate CP and MIP*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 6–20.

[22] J. Leung, L. Kelly, and J. H. Anderson, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. Boca Raton, FL, USA: CRC Press, Inc., 2004.

[23] R. M. Karp, "On-line algorithms versus off-line algorithms: How much is it worth to know the future?" in *Proceedings of the IFIP 12th World Computer Congress on Algorithms, Software, Architecture - Information Processing '92, Volume 1 - Volume I*. Amsterdam, The Netherlands, The Netherlands: North-Holland Publishing Co., 1992, pp. 416–429. [Online]. Available: http://dl.acm.org/citation.cfm?id=645569.659725

[24] Statistics Sweden, "Privatpersoners användning av datorer och internet 2016," 2016.

[25] (2017, May) Nord pool - elspot prices. [Online]. Available: http://www.nordpoolspot.com/Market-data1/Elspot/Area-Prices/ALL1/Hourly/?view=table

[26] C. Sanne, "Rekyleffekten och effektivitetsfällan – att jaga sin egen svans i miljöpolitiken," 2008.

# A

# Map of Plant 1

# B

# Tree Representation of Plant 1