Smart Grid Adaption of Church Heat

Master's thesis in Department of Computer Science and Engineering

Jesper Karlberg
Theodor Åstrand

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

Master's thesis 2017

Smart Grid Adaption of Church Heat

Jesper Karlberg
Theodor Åstrand

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

Smart Grid Adaption of Church Heat
Jesper Karlberg
Theodor Åstrand

Industry supervisor: Marcus Larsson, i3tex AB
Supervisor: Birgit Grohe, Department of Computer Science and Engineering
Supervisor: Stefan Lemurell, Department of Mathematical Sciences
Examiner: Peter Damaschke, Department of Computer Science and Engineering

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

Cover: The church of Segerstad, Karlstad, Sweden.

Typeset in LaTeX
Gothenburg, Sweden 2017

Smart Grid Adaption of Church Heat
Jesper Karlberg
Theodor Åstrand
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

The Internet is growing in a rapid pace and, as a natural result of this development, more aspects of daily life have relocated to the Internet. The Internet of Things, IoT, is a way of connecting devices over the Internet, enabling them to collect and distribute data. Smart grids are one of the primary applications of IoT and, in the past few years, comprehensive research has been conducted on smart grid heat management systems. This thesis proposes a smart grid heat management system based on a model for the temperature change and a graph model, together with a shortest path algorithm in order to achieve financial savings. The proposed model for temperature and the graph model make use of energy prices and weather forecasts, provided by Nord Pool and SMHI, respectively.

The system is evaluated and analyzed in terms of financial savings in relation to the intermittent heating method used today, as well as in terms of performance, as it will be deployed on a microprocessor. The performance is evaluated by comparing memory usage and execution time by using A* search and Dijkstra's algorithm. The findings of the thesis suggests that it is possible to create a smart grid heat management system and achieve a decrease in energy costs, in relation to the intermittent heating method, of approximately 2.5%.

Keywords: Smart grid, Smart heating system, Shortest path algorithm, A* search algorithm, Dijkstra's algorithm, Internet of Things

Acknowledgements

We would like to thank Marcus Larsson, our supervisor at i3tex AB for his assistance throughout the entire project.

In addition, we would like to thank Fredrik Hallberg and Andrejs Bondarevs at i3tex AB for providing us with essential system components as well as insight into the already existing system.

Furthermore, we want to convey our thanks to our supervisors at Chalmers University of Technology, Birgit Grohe and Stefan Lemurell for their guidance and feedback throughout the project.

Finally, we would like to direct a thanks to our examiner Peter Damaschke for his thorough feedback throughout the entire project.

Jesper Karlberg and Theodor Åstrand, Gothenburg, 2017

Contents

List of Figures

List of Tables

1

Introduction

Smart Grid Adaption of Church Heat is a master's thesis performed at Chalmers University of Technology in collaboration with i3tex AB. The purpose of the thesis is to answer the following question: is it feasible to create an algorithm based on a smart grid adaption of church heat that, in terms of financial savings, outperforms the intermittent heating method used today?

1.1 Problem description

The Internet is growing in a rapid pace and, as a natural result of this development, more aspects of daily life have been relocated to the Internet. The Internet of Things, IoT, is a way of connecting devices over the Internet, enabling them to collect and distribute data. IoT has a vast number of possible applications, for instance in the healthcare domain, smart environment, transportation and logistics domain [1].

Smart grid is one of the primary applications of IoT and, in recent years, extensive research on this topic has been published [2]–[4]. Smart grids are used to distribute energy in order to take advantage of excess energy, since areas produce and consume different amounts of energy at various times [2]. One of the more interesting aspects of smart grids is dynamic pricing, the prices of energy commonly vary during the day, and during a cold day or in the afternoon the energy price is at its peak [5]. The intention of this project is to seize the opportunity when the energy price is at its lowest, in order to create a system that has dynamic energy prices in mind.

As of today, churches commonly use intermittent heating - which means that a church is heated for a short period of time in association to events, otherwise the church is either unheated or kept at a low temperature. The churches may contain antique items, such as paintings and sculptures, which are easily harmed by a high level of humidity in the air. A condition that can create high level of humidity and condensation is when the temperature in the church is too low. A simple solution to this problem is to always have the heat on, but this is not sustainable since it is associated with high energy costs. As a result, there is a demand for a system which regulates the temperature as follows: if a church is not in use, the temperature

should be low, but not low enough to damage the interior of the church. Given a date on which the church is to be used, the church should be heated to a given temperature with energy prices in mind.

The idea of the thesis is to create, analyze and evaluate an algorithm that makes use of energy prices and weather forecasts, provided by Nord Pool and the Swedish Meteorological and Hydrological Institute, SMHI, respectively. Nord Pool is Europe's leading power markets and is offering both day-ahead and intraday energy prices [6]. The algorithm is analyzed in terms of its performance, both by using A* search and Dijkstra's algorithm, since it is important that the algorithm is not too computationally heavy, as it will be deployed on a microprocessor. The main part of the evaluation consists of evaluating the financial savings by using the algorithm, in comparison to the intermittent heating method.

1.1.1 System architecture

The system to be developed is a further development of an already existing system owned by i3tex AB [7]. The architecture of the system to be developed is depicted in Figure 1.1. Sensors are strategically spread across a church and are organized in a wireless mesh topology, which means that all sensors in a network collaborate on collecting and exchanging data [8]. The wireless sensor network communicates with a coordinator over a 2.4 GHz ZigBee protocol [9]. The coordinator collects and forwards data measurements received from the sensors to a server over a USB interface.

The server stores the data measurements in both an external and a local database. In addition, the server requests weather forecasts and energy prices from SMHI and Nord Pool respectively, which is further described in Chapter 4.1. In a web interface an end user can view past temperature measurements, initialize values for how the temperature changes for various differences in temperatures inside and outside the church, but more importantly configure the lowest allowed temperature.

Based on energy prices, weather forecasts and temperature measurements, the algorithm determines whether to activate the heating or not. The binary decision is thereafter stored in the local database. The coordinator frequently reads the binary decision from the local database and forwards it to a contactor. The contactor is a control switch, which is used for switching an electrical power circuit, that either closes or opens the circuit to the heating.

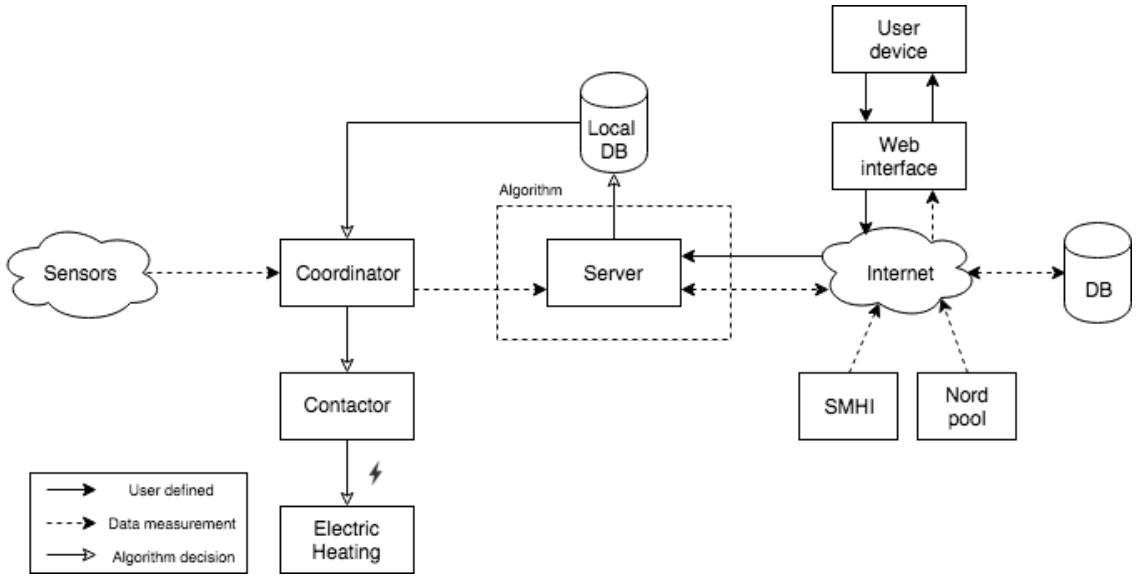

Figure 1.1: System architecture.

1.2 Related work

In the past few years, comprehensive research has been conducted on smart grid heat management systems [5], [10]–[12]. The idea of a smart grid heat management system is therefore not unexplored, but the way that the problem is tackled differ remarkably.

The authors of [5] present a smart Heating, Ventilation and Air-Conditioning, HVAC, control where smart pricing and user comfort are taken into account. This is done by using an energy cost function, specified by the energy provider, that depends on a smart pricing tariff. In [5], there is a distinct focus on preserving a steady temperature and the authors do not consider changing the temperature when the facility is empty.

Another study, where a HVAC system is used, is [10], where a simulation-optimization approach is used for the effective energy management. The authors of [10] achieve immediate energy savings by resetting operating parameters for the air and water sidesystems with respect to dynamic cooling loads and changing weather conditions. In contrast to [10], this thesis aims to achieve financial savings rather than energy efficiency. Additionally, while [10] aims to produce a result by changing parameters of system components, this thesis aims to further develop an already existing system which relies on external data to reach desired result.

The authors of [11] presents a HVAC model that can be applied in the scheduling problem of a Home Energy Management System, HEMS. The HVAC model considers customer convenience and parameters that are related to indoor heat capacity, and heat dissipation is estimated statistically from historical data. However, the HVAC model does not operate when a home is unoccupied in order to minimize the

overall energy cost. In contrast to [11], the system considered in this thesis includes requirements when a facility is not in use which demands that the system operates at all times.

In [12], a two-stage load control strategy for residential HVAC systems is presented. The first stage consists of mappings between electricity price ranges and temperature set points, for all hours of the day, within a predefined thermal comfort interval. The second stage improves the financial and energy efficiency by using pre-cooling and pre-floating procedures. In the model presented by the authors of [12], a discomfort tolerance index paves the way for the trade-off between electricity payment and the thermal discomfort. This index specifies the resident's tolerance to thermal discomfort, where a higher index allows for a higher thermal discomfort. In similarity to [12], this thesis includes a comfort constraint when residents are present at the facility but unlike [12], this constraint is removed when the facility is empty.

1.3 Scope

The area of smart grid heat management systems is well explored and there are a vast number of implementations in various settings, thus a delimitation of this thesis is motivated.

The authors of [5], [10]–[12] are using HVAC systems, in this thesis both ventilation and air-conditioning are out of scope. Instead, the algorithm will strictly focus on regulating the temperature in regard to dynamic pricing. In contrast to [5], the system will not depend upon an energy cost function, instead energy prices are gathered directly from the energy market.

The system has a specific application, namely facilities that are not used in a regular fashion such as: churches and holiday cottages. The system can be applied to an ordinary house, but it will be troublesome since there is a requirement to specify when the house will be used.

The system relies on data gathered from Nord Pool [6] and SMHI [13] and is therefore limited to the availability of such data. New data from Nord Pool is commonly available for the next day about 11 hours in advance, resulting in resources being available for the following 11-35 hours. Data from SMHI is usually available for every hour the following 72 hours.

This thesis strictly focuses on an implementation of a smart grid heat management system based on a shortest path algorithm, using A* search. Another shortest path algorithm, Dijkstra's algorithm, will be used in order to evaluate performance of the system.

1.4 Thesis outline

Chapter 2 begins with an analysis of the underlying theory needed to fully grasp the thesis. First, a short introduction of graphs is displayed in Section 2.1, followed by an explanation of shortest path algorithms in Section 2.2. Thereafter, how the temperature of an object changes is described in Section 2.3 and spot prices and smart grids are explained in Section 2.4. Chapter 2 ends with an analysis of the potential environmental impact of the system in Section 2.5.

Chapter 3 describes how the underlying theory was used in order to create a graph model and a temperature model used in the system.

Chapter 4 proceeds to show how the system was implemented, beginning with Section 4.1 displaying what data, from external sources, was needed to implement the functionality of the system. This is followed by an explanation of how temperature data was collected and manipulated in order to predict how the temperature changes in the facility, shown in Section 4.2. Further, Section 4.3 shows how the gathered data was used to create a graph that includes the potential ways of reaching a target temperature. Thereafter, Section 4.4 introduces how a shortest path algorithm was used to find the most economically beneficial path through this graph. This chapter will, in Section 4.5, end by an explanation of how the system used a so called scheduler to keep the system alive and updated.

The thesis then proceeds to show the findings of the project, in Chapter 5. The chapter includes potential financial savings of using the system as well as the performance of the system.

The result of using the system with regards to financial savings and performance is discussed in Section 6.1 and 6.2, respectively. Additionally, the accuracy of the weather forecasts gathered from SMHI is discussed in Section 6.3 and a review of potential improvements of the system is examined in Section 6.4. Furthermore, the credibility of the result is discussed in Section 6.5.

Finally, the outcome of the project is concluded in Chapter 7.

2

Background

This chapter aims to explain the underlying theory necessary for the project and its implementation. Initially, a brief review of graph theory and two shortest path algorithms are presented. Thereafter, the underlying theory of how an object emits heat is explained, as well as interpolation and extrapolation which are used to generate new data points from a data set of known data points. Then, the chapter will describe how the system is affected by electricity price changes, accuracy in temperature measurements and what impact the system has on the environment.

2.1 Graph definitions

A graph $G = (V, E)$ consists of a collection of vertices V, and a collection of edges E. An edge (u, v) is a connection between two vertices u and v. A graph contains two distinguished vertices that are called source and target. A graph can either be directed or undirected, in a directed graph an edge (u, v) has an unique direction from vertex u to vertex v. In an undirected graph, an edge does not have an unique direction, thus the edge (u, v) is identical to the edge (v, u). In addition, an edge has a weight $w(u, v)$, specifying the cost of traversing from vertex u to vertex v.

A path $< u_0, ..., u_i, u_{i+1}, ..., u_n >$ consists of vertices, connected by edges $(u_i,\ u_{i+1})$ $i = 0, 1, ..., n-1$, from the source vertex $s = u_0$ to the target vertex $t = u_n$. The depth of a vertex is defined as the number of edges on the path, containing the least number of edges, from the source vertex to the vertex. The source vertex is said to have depth zero. The total weight of all edges on a path is called the weight of the path [14].

2.2 Shortest path algorithms

Given a directed graph $G = (V, E)$, the shortest path from one vertex to another vertex is a path such that there is no other path in the graph that has a lower weight between the vertices. There are numerous algorithms to find the shortest path

between two vertices. In this thesis, two shortest path algorithms are highlighted: Dijkstra's algorithm and the A* search algorithm.

2.2.1 Dijkstra's algorithm

Dijkstra's algorithm can be used to find the shortest path between a source vertex s and every other vertex in a graph [14], [15]. Dijkstra's algorithm begins at the source vertex and explores adjacent vertices until the shortest path to the target vertex t is discovered. A vertex is said to be settled, when the shortest path from the source vertex to the vertex has been discovered. The algorithm is only applicable if all $w(v, u) \geq 0$, thus no edges can have a negative weight.

The authors of [14] explains Dijkstra's algorithm as follows:

1. The algorithm begins by initializing a tentative distance for all vertices in an array D. For all vertices, except the source vertex, a value of ∞ is assigned. The source vertex is assigned the value of zero.

2. Then, the algorithm proceeds by selecting the source vertex s as the current vertex c.

3. The algorithm then considers each adjacent vertex v of the current vertex c and computes a new tentative distance to v, in the following manner: $D[c]+w(c, v)$. Thereafter, the current value of $D[v]$ and the newly computed tentative distance to vertex v is compared, and the lowest value of the two is assigned to $D[v]$.

4. Thereafter, the vertex u with the lowest tentative distance in the array D is chosen as the current vertex c.

5. If the shortest path to the target vertex u has been found, the algorithm terminates. Otherwise the algorithm proceeds at step three.

The shortest path can be found by traversing each vertex's predecessor, if each vertex stores its predecessor, from the target vertex to the source vertex.

An example of a graph that Dijkstra's algorithm can be applied on is shown in Figure 2.1. As mentioned above, the algorithm proceeds by initializing the array D, as show in Table 2.1.

D[s]	0
D[a]	∞
D[b]	∞
D[t]	∞

Table 2.1: Initializing the D array for all vertices.

Then, the source vertex s is considered settled and chosen as the current vertex c. Thereafter, a new distance to all adjacent vertices of vertex s is calculated, as shown in Table 2.2

D[s]	**0**
D[a]	1
D[b]	2
D[t]	∞

Table 2.2: The D array after vertex s is settled.

The vertex a is now considered as the current vertex c. Thus, the distance to all adjacent vertices of vertex a is calculated, as shown in Table 2.3

D[s]	**0**
D[a]	**1**
D[b]	2
D[t]	5

Table 2.3: The D array after vertices s and a are settled.

Lastly, vertex b is considered settled and chosen as the current vertex c. A new distance to the target vertex can now be calculated, as can be seen in Table 2.4.

D[s]	**0**
D[a]	**1**
D[b]	**2**
D[t]	**4**

Table 2.4: The D array after vertices s, a, b and t are settled.

Finally all vertices are settled, and the resulting shortest path is $< s, b, t >$ with a weight of four on the path. Note that the vertices were expanded in the following order $< s, a, b, t >$, which will be used to compare Dijkstra's algorithm and A* search in Section 2.2.2.

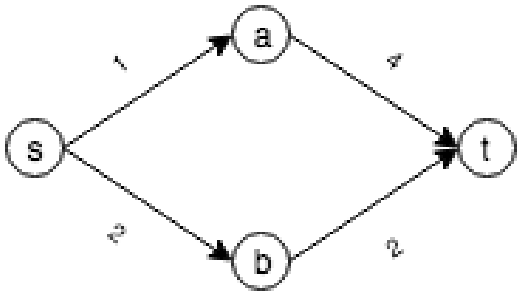

Figure 2.1: Example of a connected and directed graph.

2.2.2 A* search algorithm

One of the most commonly used heuristic search algorithms is the A* search algorithm [16] and much like Dijkstra's algorithm, the A* search algorithm can be used to find the shortest path between two vertices.

In contrast to Dijkstra's algorithm, A* search uses a heuristic function, $h(v)$, in order to consider the vertex that is most likely to be a part of the shortest path [14]. A heuristic function estimates the weight of the shortest path between vertex v and the target vertex t. The heuristic function is unique for each problem, and a requirement is that the heuristic function is admissible in order to guarantee that A* search finds the shortest path. An admissible heuristic function is a non-negative underestimate of the cost to reach the target vertex.

The A* search algorithm and Dijkstra's algorithm are similar, in fact, according to the authors of [14], the A* search algorithm is equivalent to running Dijkstra's algorithm with the edge weights substituted according to Equation 2.1.

$$w'(u, v) \leftarrow w(u, v) - h(u) + h(v) \tag{2.1}$$

In Equation 2.1: $w'(u, v)$ represents the new cost of the edge after applying the heuristic function. Therefore, Dijkstra's algorithm can be seen as a special case of the A* search algorithm with the result of the heuristic function being zero.

The example provided in Section 2.2.1 can now be revisited in order to compare Dijkstra's algorithm and A* search. As mentioned above, A* search relies on a heuristic function in order to expand vertices in a more efficient manner. For simplicity, a heuristic look-up table with optimal values, with regards to Figure 2.1, is assumed, as shown in Table 2.5. The weights in Figure 2.1, can now be substituted according to Equation 2.1 using the heuristic look-up table in Table 2.5 into a new graph, depicted in Figure 2.2. By applying Dijkstra's algorithm on Figure 2.2 the resulting path is $< s, b, t >$ with a weight on the path of four and the vertices were expanded in the following order $< s, b, t >$. Thus, the shortest path was found with A* search even though less vertices were expanded with A* search compared to Dijkstra's algorithm.

h(s)	4
h(a)	4
h(b)	2
h(t)	0

Table 2.5: A heuristic look-up table with optimal values.

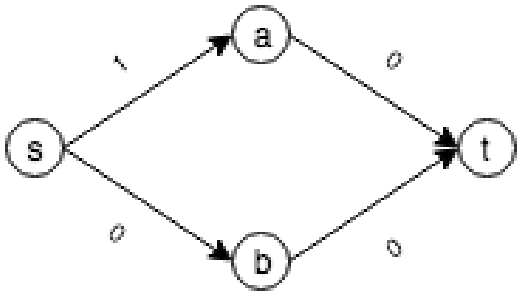

Figure 2.2: Graph 2.1 with substituted weights according to Equation 2.1.

2.3 Transferred quantity of heat

How an object emits heat can be determined by examining attributes of the object such as: area and heat transfer coefficient. In addition, the emitted heat also depends on external influences such as: the temperature difference between the object and its surroundings, as well as the time lapse of the process according to [17], as shown in Equation 2.2.

$$\Delta Q = \alpha \cdot A \cdot (T - T_M) \cdot \Delta t \tag{2.2}$$

In Equation 2.2: ΔQ is the transferred quantity of heat from the object, α is the heat transfer coefficient that specifies the ability of a medium to transfer heat from the object, A is the contact surface of the object, T is the temperature of the object, T_M is the temperature of the medium and Δt is the time interval.

Since α and A are constant for a specific object, the transferred quantity of heat will change linearly in correlation to the difference in temperature between the object and its surroundings per Δt. As a result, less quantity of heat will emit from the object and less energy will be required to increase the temperature of the object at a lower temperature difference.

2.3.1 Interpolation and extrapolation

When a data set contains insufficient data, interpolation can be used to estimate the missing values between already known values. If such values are linear in relation to each other, linear interpolation can be used, as shown in Equation 2.3.

$$y_i = \frac{y_b \cdot (x_i - x_a) - y_a \cdot (x_i - x_b)}{x_b - x_a} \tag{2.3}$$

Values can be estimated using this equation if $x_a < x_i < x_b$ and y_a and y_b are values corresponding to x_a and x_b respectively and x_i is the value for which y_i should be

calculated. If the values of x_a, x_b, y_a and y_b are known, interpolation can be used to calculate an estimation of y_i corresponding to a chosen value for x_i.

In contrast to interpolation, extrapolation is used to estimate values outside the already known interval. Values can be estimated for y_i using Equation 2.3 in the same manner as interpolation if $x_i < x_a < x_b$ or $x_a < x_b < x_i$

2.4 Energy market

Energy distribution has of recent years made it possible to trade energy with surrounding countries. Smart grids takes advantage of these opportunities to create distributed power grids where residual energy is transferred and used where additional energy is needed [18]. This has allowed for international power markets, such as Nord Pool, to form. Figure 2.3 shows how energy is bought and sold across energy zones and country borders.

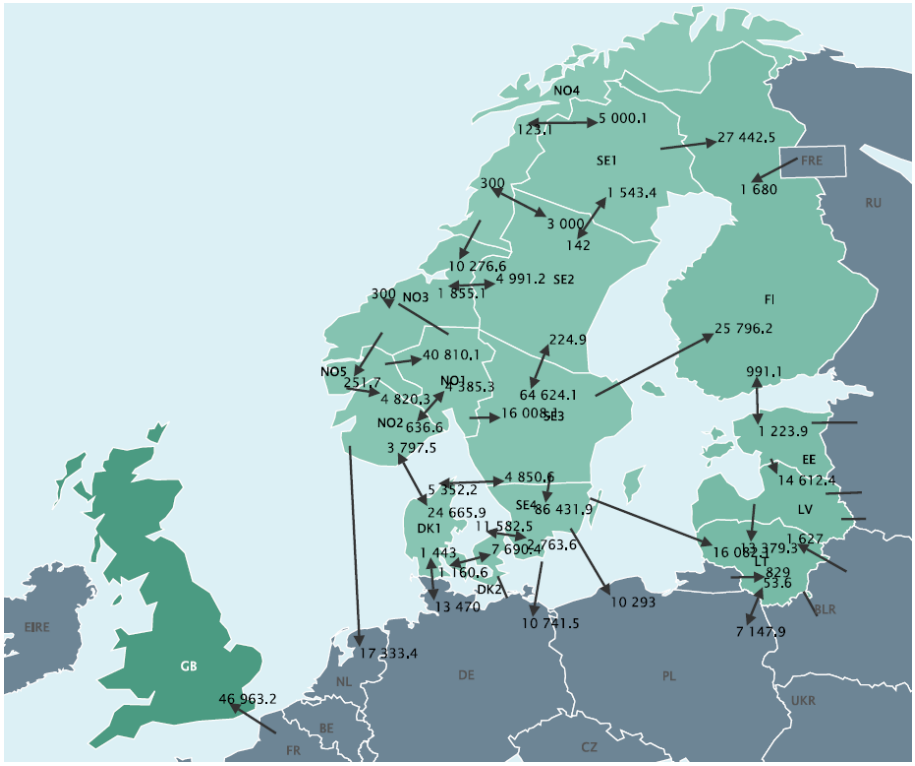

Figure 2.3: A map showing the flow of energy across energy zones. Source: [6]

Spot prices in the energy market are prices of energy for a specific energy zone [19], shown in Figure 2.4. A country can be divided into several energy zones and the energy price for each zone differ depending on supply and demand. Furthermore, the energy prices within a zone differ throughout the day due to the same reason, which results in lower prices of energy when the relation between production of energy and consumption of energy is beneficial. This makes it more profitable to use energy

during certain times of day compared to others.

The price of electricity is, as mentioned above, determined by the relation between supply and demand [6]. As a result, the price of electricity will be high when the electricity consumption is high as well as when the supply of electricity is low. According to [20], the price of electricity commonly differ 100% over an hour. An example provided by [20] shows an 606% price increase from 141 Nkr/MWh to 993 Nkr/MWh between 7 a.m. and 8 a.m..

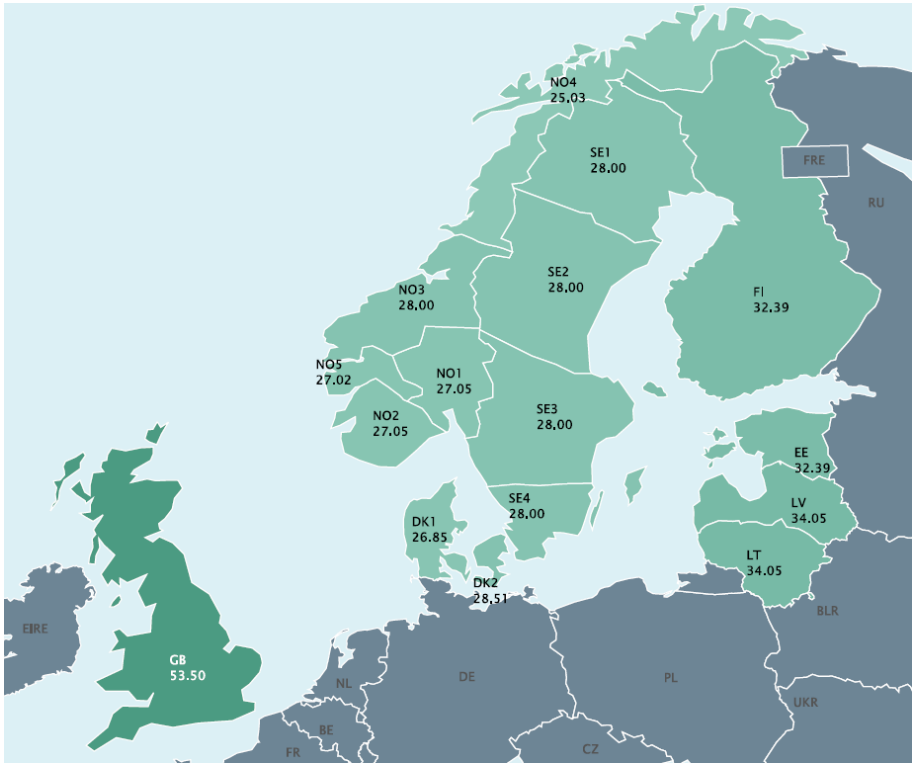

Figure 2.4: A map showing the areas and the energy price in EUR/MWh for each area. Source: [6]

2.5 Environment

The system will be developed with focus on financial savings, rather than energy efficiency. As a result, the system will potentially use more energy than the intermittent heating method currently in use. An argument can be made that the system, therefore, will contribute to an increased negative environmental impact [21]. However, the amount of energy used does not independently determine the environmental impact, instead it depends upon several factors, for instance the origin of the energy.

In a society where an increased portion of the electricity used comes from renewable sources, the energy supply will be less trivial to manage, since energy from such sources can only be generated when conditions are sufficient. For instance, a solar power plant can not generate electricity during the night and it generates substantially less electricity when the sun is blocked by clouds. As a result, the price of electricity increase or decrease when the conditions to accommodate the renewable energy plants are beneficial or non-beneficial, respectively. The authors of [22] show that the price of energy decreased by 0.1% when the power generated from wind increased by 1% in MWh per day. In general, energy is difficult to preserve and is instead produced to satisfy the current energy consumption. However, energy from renewable sources such as wind, solar and wave are in general not produced by demand, instead it is produced when the respective energy source is available. Therefore, it is important to use renewable energy when the supply is high. When renewable energy makes up a comprehensive part of the total electricity produced, it will be difficult to produce energy in correlation to the demand, which will lead to more excess electricity [23]. In fact, negative electricity prices appeared on the European Energy Exchange in October 2008, because a lot of residual energy was generated from wind power [24]. Since the developed system will use electricity when the energy price is at its lowest, which is also when the most excess energy is available, a more substantial part of the energy used will come from renewable energy sources which will result in a less negative impact on the environment [25].

The temperature at a specific location is influenced by local conditions such as: soil surface, if the location is in a slope and on surrounding objects such as: building, trees and mountains [26]. It is therefore inaccurate to assume that weather forecasts from a meteorological institute are accurate, even for the current time, since the location where the temperature was measured and the location of a church can be significantly different. Figure 2.5 shows how the temperature measured outside the Gothenburg office of i3tex AB differed in comparison to the temperature collected from the closest weather station. As displayed, the main part of the sampled data is located between 0 °C and -2 °C, indicating that the temperatures collected from the weather station are slightly higher. This supports the claims made by [26] that local conditions influences the actual temperature at a location. The data used in Figure 2.5 can be found in Appendix A.

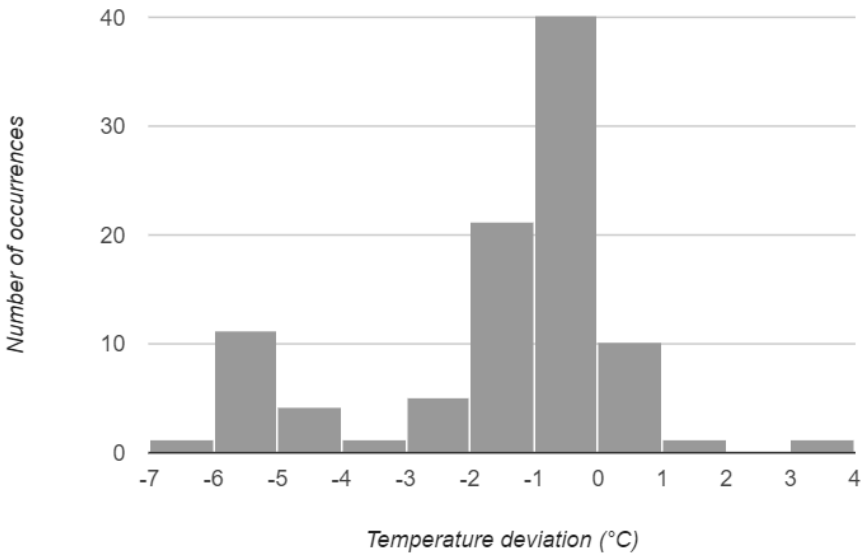

Figure 2.5: Histogram showing the difference in temperature between SMHI temperatures and temperatures measured at the office of i3tex AB.

3

Models

This chapter begins with an explanation of how the transferred quantity of heat, mentioned in Section 2.3, is used to create a model for how the temperature changes in a church over time. Thereafter, an explanation of how a graph model was built using the temperature model in combination with energy prices is presented.

3.1 Temperature model

According to [5], the transferred quantity of heat from an object is linear to the difference in temperature between the object and its surroundings, at least for shorter periods of time. As shown in Section 2.3, it is therefore relevant to know how the temperature in a church changes in relation to the temperature inside and outside the church. How the temperature changes for such temperature differences can then be used when creating vertices in a graph in accordance to Equation 3.1.

$$T_v = T_u + \Delta T \tag{3.1}$$

In Equation 3.1: T_v is the temperature of vertex v, T_u is the temperature of vertex u, a predecessor of vertex v, and ΔT is the temperature change for the difference in temperature inside and outside the church for the concerned time.

The temperature inside a church will change depending on the size of the church and the church's capacity to preserve heat. It is therefore not possible to create an accurate model for all churches without first introducing building-specific parameters. Since no such parameters are known, changes in temperature inside a church will be measured continuously while the system is running. When a number of values have been measured, interpolation and extrapolation, described in Section 2.3.1, are used to obtain a higher frequency and a broader interval of values for how the temperature changes in correlation to differences in temperature inside and outside the church.

As previously stated, the change in temperature is dependent on the temperature

difference inside and outside the church. Consequently, to estimate the temperature of the next vertex, in addition to the temperature in the current vertex, the outside temperature is needed. This temperature is gathered from SMHI's Open Data API. As mentioned in Section 2.5, the temperature surrounding the church is dependent on local conditions and may not be the same as those collected at a meteorological institute. These collected values will therefore be adjusted according to previous errors. The correction is based on the mean error, which is an average of the difference in temperature between the values measured at the church and those collected from the meteorological institute.

3.2 Graph model

A highly simplified example of a graph is demonstrated in Figure 3.1, where each vertex represents a specific temperature in some point of time. The edge $E_{u,v}$ from vertex u to vertex v represents the price of increasing or decreasing the temperature from vertex u to vertex v. As can be seen in Figure 3.1, the time interval between each vertex is equal and will depend on the update frequency of the energy prices. Consequently, the system would either have the heating on or off for one full hour, potentially resulting in vast temperature changes over said time. To avoid such temperature changes and to allow for a higher number of possible solutions to reach the targeted temperature, the time between vertices can be set to a shorter time interval, in the future referred to as graph resolution. The resolution can be set to any number of minutes, less than 60 and which divides 60. The reason why the resolution must divide 60 evenly is to ensure that every edge contains only one energy price. Such a resolution allows the graph to decrease the time between vertices to, for instance, 30 minutes, as shown in Figure 3.2.

The temperature of the source vertex, s, will be measured from sensors that are placed inside a church. In order to calculate the temperatures that can be reached from vertex s, the temperature model described in Section 3.1 will be used. As can be seen in Figure 3.1, there are multiple paths from vertex s to vertex t, an example of such a path is thickened. When the church is not in use, the shortest path will be recalculated within a predefined time interval to take advantage of recently updated energy prices and weather forecasts.

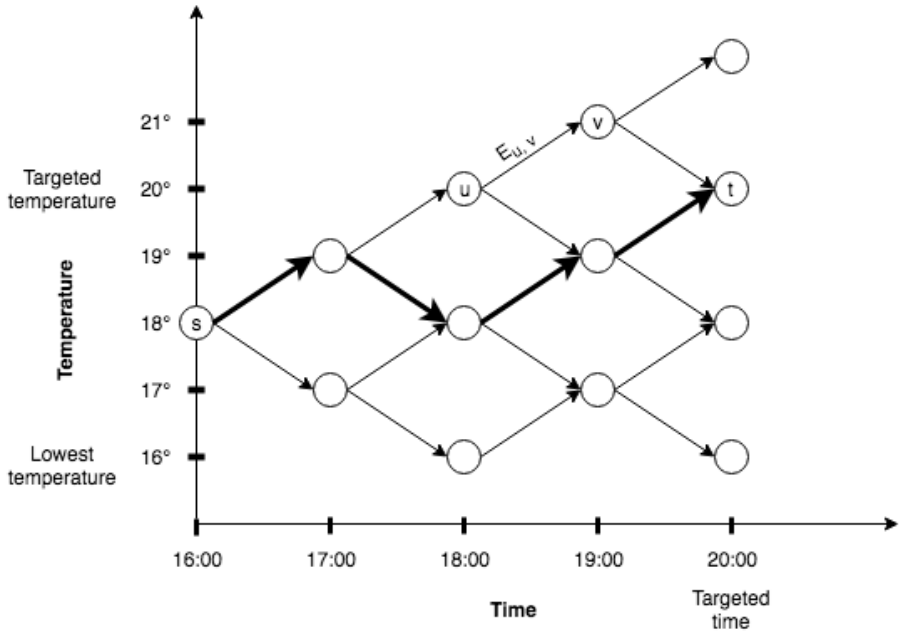

Figure 3.1: A graph that demonstrates the relationship of vertices and edges in the context of the system.

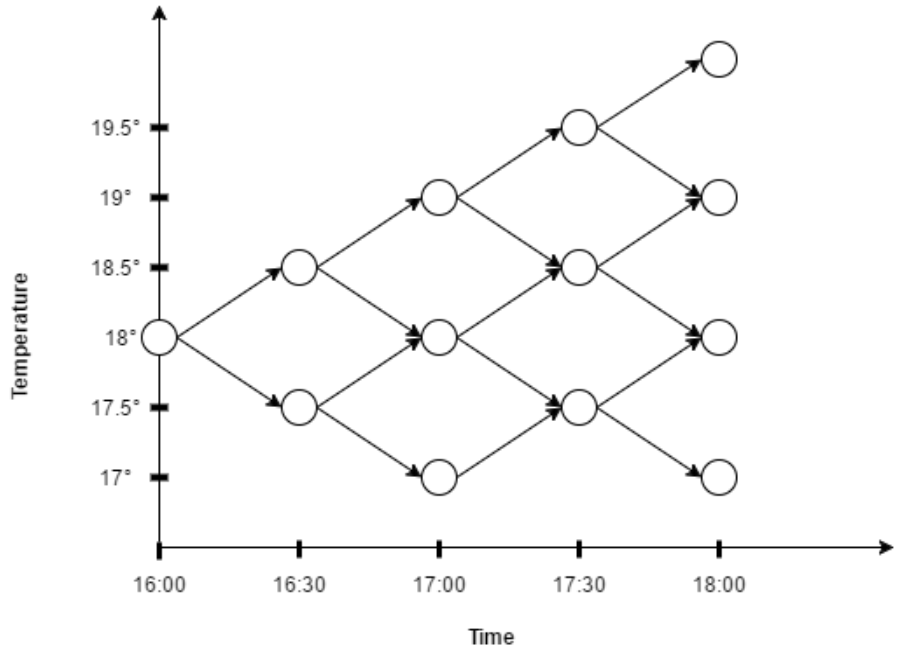

Figure 3.2: A graph that demonstrates the relationship of vertices and edges in the context of the system when the graph resolution has been set to 30 minutes.

Unlike what is shown in Figure 3.1, the temperature for longer periods of time does not increase and decrease linearly but can instead change irregularly depending on external influences. According to [17], the only parameter, not related to the church, influencing the released quantity of heat is the difference in temperature between two substances, in this case the temperature difference inside and outside the church. If

the temperature outside the church were to remain constant, and the temperature inside the church increased or decreased over time, the released quantity of heat would increase or decrease respectively, in relation to the difference in temperature [17]. Therefore, the assumption that the gain and decay in temperature is equivalent for different temperatures, as can be seen in Figure 3.1, is not valid.

Accordingly, the graph to be used in the shortest path algorithm will look more similar to the graph shown in Figure 3.3, where the change in temperature will depend on the difference in temperature inside and outside the church. In addition, over time some paths in the graph could potentially reach unwanted temperatures, for example temperatures that are too low and could be harmful for the interior of the church. In order to resolve this issue and to avoid creating such vertices, the graph contains a lower boundary. Furthermore, the model includes a variable for user comfort, meaning that the targeted temperature, when the facility is used, no longer is a fixed temperature but rather a temperature range. As can be seen in Figure 3.3, each vertex has two outgoing edges, thus the number of vertices can be calculated as follows, where d is the depth of the graph:

$$|V_d| = \begin{cases} 0, & \text{if } d < 0. \\ 2^d - 1, & \text{otherwise.} \end{cases} \tag{3.2}$$

In correlation with Equation 3.2, the number of edges can be calculated as follows:

$$|E_d| = \begin{cases} 0, & \text{if } d \leq 0. \\ |V_d| - 1, & \text{otherwise.} \end{cases} \tag{3.3}$$

According to Equation 3.2 and Equation 3.3 the number of vertices and edges grow exponentially. Consider a graph, G_1, with $d = 36$, then $|V_d| = 68719476735$. This is, by all means, not sustainable on a small memory space. Instead, the temperatures are rounded to one decimal and therefore multiple vertices with negligible differences in temperature coincide. However, if the graph resolution is increased or if the church's capacity to increase the temperature is inadequate, then the number of decimals needs to be increased in order to avoid rounding errors. Multiple examples of coinciding vertices can be seen in Figure 3.3 and are highlighted by a dashed circle, these vertices are merged to create a representative vertex.

If we consider another graph, G_2, with $d = 36$, that makes use of coinciding vertices and has a lower temperature boundary of 15 °C and an upper temperature boundary of 25 °C, then at most 100 vertices are created at each level of depth since $\frac{25-15}{0.1} = 100$. Then the graph can at most consist of 3600 vertices, since $100 \times 36 = 3600$, which implies that G_2 is considerably more memory space efficient than G_1.

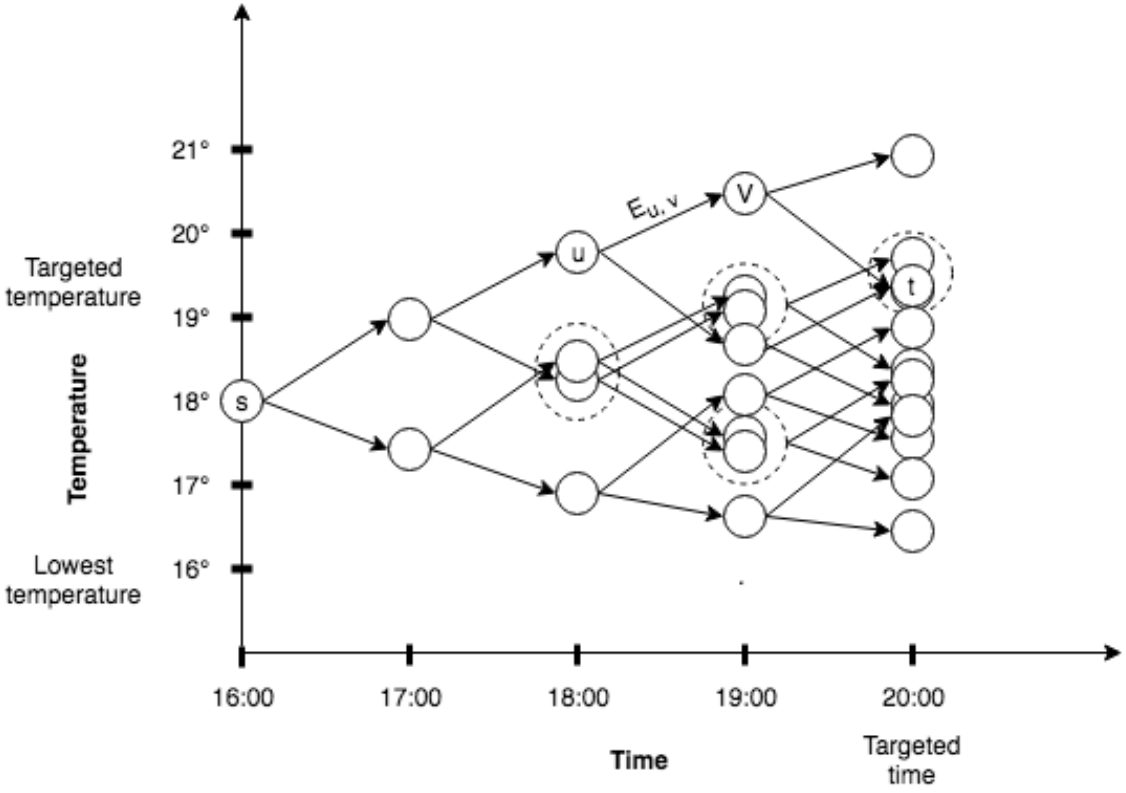

Figure 3.3: A realistic graph that demonstrates the relationship of vertices and edges in the context of the system.

4

Implementation

This chapter begins by explaining how weather forecasts and energy prices were collected and used when implementing the models, described in Section 3.1 and 3.2. Thereafter, the chapter provides a review of how the graph and the shortest path algorithms were implemented, followed by an explanation of how different components of the system were used to provide a lasting system functionality.

4.1 Gathering third party data

In order for the system to function in accordance with requirements requested by the main stakeholder, i3tex AB, data sets from several third party sources were required. These data sets are collected by two parsers and are used in the algorithm to decide on when to heat the church.

4.1.1 SMHI Open Data API

As mentioned in Chapter 3.2, weather forecasts, gathered from the SMHI Open Data API, are used to collect the temperature in close proximity to a church. In particular, the SMHI Open Data Meteorological Forecasts API is used and it contains weather forecasts, for every whole hour, for the following three days. The weather forecasts are generally updated six times a day and are based on a number of forecast models, statistical adjustments and manual edits [13].

To request the SMHI Open Data Meteorological Forecasts API, the longitude and latitude of the church are required. The weather forecasts are returned in the data format JavaScript Object Notation, JSON, which is used to structure data [27]. The JSON file contain several parameters, for every whole hour, such as: wind speed, air temperature, relative humidity and thunder probability.

4.1.2 Nord Pool

The system, more specifically the graph model, uses energy prices to make decisions on whether to heat the church. This information has, in accordance with the request of the main stakeholder, been gathered from Europe's leading power market Nord Pool. Nord Pool provides intraday and day-ahead energy prices divided over sixteen energy zones across: Sweden, Norway, Denmark, Finland, Estonia, Latvia, Lithuania and Great Britain as shown in Figure 2.4. The energy prices are updated every 24 hours.

To collect necessary information from Nord Pool their data API is used. Using the API, with information such as currency, date and a number specifying the resolution of the information set as hourly, daily, weekly, monthly or yearly, a JSON-file is composed containing the energy prices for each energy zone.

4.2 Calibration of temperature data

Since the system is built to accommodate any church and relies on building specific parameters such as the church's capacity to preserve heat, an initial calibration is performed when the system launches. The calibration is performed by first heating the church for a specified number of hours, by default two hours, then turning the heat off for the same amount of time. The purpose of the calibration is to populate the database with information of how the temperature changes in relation to the difference in temperature inside and outside the church, as shown in Table 4.1. In addition to the calibration, the system is recalibrated during run time to take advantage of newly measured values.

4.2.1 Interpolation

The calibration provides an initial data set of how the temperature changes in correlation to specific differences in temperature inside and outside the church. However, the data set does not contain data for all differences in temperature required to build a graph. For the missing data, between the known values, interpolation as shown in Section 2.3.1 is used. Using interpolation on the "temperature changes" found in Table 4.1, a new data set containing calibrated and estimated values is created for every difference in temperature, from the first to the last known value, with an interval of 0.1 °C as shown in Table 4.2.

Temp. Difference	Temp. change up	Temp. change down
15 °C	+0.33 °C	-0.73 °C
15.3 °C	+0.321 °C	-0.742 °C
15.4 °C	+0.319 °C	-0.745 °C

Table 4.1: Measured temperature change values, showing how the temperature increases, Temp. change up, and decreases, Temp. change down, in relation to the difference in temperature inside and outside the church.

Temp. Difference	Temp. change up	Temp. change down
15 °C	+0.33 °C	-0.73 °C
15.1 °C	+0.327 °C	-0.734 °C
15.2 °C	+0.324 °C	-0.738 °C
15.3 °C	+0.321 °C	-0.742 °C
15.4 °C	+0.319 °C	-0.745 °C

Table 4.2: Interpolated and measured temperature change values, showing how the temperature increases, Temp. change up, and decreases, Temp. change down, in relation to the difference in temperature in side and outside the church.

Algorithm 1 shows how ”temperature changes” within the known interval of values were estimated. A *TemperatureChange* contains ”temperature changes” for specific differences in temperature inside and outside the church. *tempC1* and *tempC2* are already known *TempeartureChanges*, such as those collected during calibration, for which new values shall be estimated between. The algorithm then proceeds to use Equation 2.3 to find new ”temperature changes” for every temperature difference between *tempC1* and *tempC2* with an interval of 0.1 °C.

Algorithm 1 Interpolating values for how the temperature changes in correlation to the difference in temperature in a known interval.

```
1:  function INTERPOLATION(TemperatureChange tempC1, temperatureChange tempC2)
2:      polatedList ← new List<TemperatureChange>
3:      y_1 ← tempC1.getTemperatureChange
4:      y_2 ← tempC2.getTemperatureChange
5:      x_1 ← tempC1.getTemperatureDifference
6:      x_2 ← tempC2.getTemperatureDifference
7:      for i = x_1; i <= x_2; i+ = 0.1 do
8:          Calculate y_0 for x_0 = i using Equation 2.3
9:          polatedList.add(new TemperatureChange(i, y_0))
10:     return polatedList
```

4.2.2 Extrapolation

Interpolation estimates values within a known interval, values outside this interval can be estimated using linear extrapolation, as mentioned in section 2.3.1. Commonly, Equation 2.3 is used when extrapolating a data set. Since this equation depends solely on two values for the estimation, very inaccurate data could be generated if the two values are not a good representation of the data set. Therefore, a mean of the known "temperature changes" is used, further shown in Algorithm 2.

Algorithm 2 Extrapolating temperature changes in correlation to the difference in temperature outside a known interval.

```
1:  function EXTRAPOLATION(TemperatureChangeList tempCList)
2:      polatedList ← new List<TemperatureChange>
3:      y1 ← tempCList.getFirst.getTemperatureChange
4:      y2 ← tempCList.getLast.getTemperatureChange
5:      x1 ← tempCList.getLast.getTemperatureDifference
6:      x2 ← tempCList.getLast.getTemperatureDifference
7:      tChange ← 0.0
8:      for i = 0; i < tempClist.size() − 1; i++ do
9:          tChange ← tChange + change per 0.1 degree between
10:             tempClist.get(i) and tempClist.get(i + 1)
11:     tChange ← tChange/tempCList.size() − 1
12:     for i = x1 − 0.1; i >= −100; i− = 0.1 do
13:         y1 ← y1 − tChange
14:         polatedList.add(new TemperatureChange(i, y1))
15:     for i = x2 + 0.1; i <= 100; i+ = 0.1 do
16:         y2 ← y2 + tChange
17:         polatedList.add(new TemperatureChange(i, y0))
18:     return polatedList
```

A mean value for how the temperature changed per 0.1 °C was calculated based on the values gathered in the calibration. For all differences in temperature, outside the interval of known values, within reasonable boundaries, new values for how the temperature changed were estimated using the mean value, as shown in Table 4.4, using the values found in Table 4.3.

Temp. Difference	Temp. change up	Temp. change down
10 °C	+0.43 °C	-0.83 °C
10.3 °C	+0.421 °C	-0.842 °C
10.4 °C	+0.419 °C	-0.845 °C

Table 4.3: Measured temperature change values, showing how the temperature increases, Temp. change up, and decreases, Temp. change down, in relation to the difference in temperature inside and outside the church.

Temp. Difference	Temp. change up	Temp. change down
⋮	⋮	⋮
9.9 °C	+0.4325 °C	-0.8265 °C
10 °C	+0.43 °C	-0.83 °C
10.3 °C	+0.421 °C	-0.842 °C
10.4 °C	+0.419 °C	-0.845 °C
10.5 °C	+0.4165 °C	-0.8485 °C
⋮	⋮	⋮

Table 4.4: Extrapolated and measured temperature change values, showing how the temperature increases, Temp. change up, and decreases, Temp. change down, in relation to the difference in temperature inside and outside the church.

4.3 Graph

The weather forecasts, energy prices and interpolated and extrapolated temperatures are used to create a connected and directed graph using Algorithm 3. The algorithm consists of three functions: *createGraph*, *createVertices* and *buildSubGraph*.

Initially, in the *createGraph* function, a vertex, $sourceVertex$, is created using the temperature inside and outside the church with the current date and time. In addition, a vertex, $targetVertex$, is created with the targeted temperature, date and time. Thereafter, if the depth of the graph is larger than one, the adjacent vertices of the $sourceVertex$ is created in order to build two subgraphs in parallel.

The function *createVertices* is used to create succeeding vertices given a vertex, $prevVertex$. Based on $prevVertex$'s temperature, the temperature of the adjacent vertices can be calculated using the $prevVertex$'s inside and outside temperature, in combination with the interpolated and extrapolated temperatures, as described in Section 3.1. As described in Section 1.1, the temperature inside the church should not be too low, therefore a new vertex is only created if the newly calculated temperature is above a specified lower boundary. If a new vertex is created, an edge is added from $prevVertex$ to the newly created vertex.

The function *buildSubGraph* is a recursive function that, given a $prevVertex$ terminates when the date of the $prevVertex$ is equal to the targeted date of the $targetVertex$. If the termination condition is met, the temperature of the $prevVertex$ is checked, if it is within the accuracy of the $targetVertex$ an edge from $prevVertex$ to the $targetVertex$ is added. The accuracy of the $targetVertex$ means that the temperature of the $prevVertex$ does not need to be exactly equal to the temperature of the $targetVertex$, instead it is acceptable that the temperature of the $prevVertex$ is a few tens of degrees above or below the $targetVertex$. Otherwise, if the date of the $prevVertex$ is before the date of the $targetVertex$, the succeeding vertices of $prevVertex$ is created using the function *createVertices* and the function *buildSubGraph* is recursively called with the newly created vertices.

Algorithm 3 Build a graph given an initial temperature inside the church and temperatures outside the church.

```
1: function CREATEGRAPH(insideTemp, outsideTemp, startDate, targetedDate,
   resolution)
2:     sourceVertex ← new Vertex (insideTemp, outsideTemp, startDate)
3:     targetVertex ← new Vertex (targetedTemp, outsideTemp(targetedDate),
4:         targetedDate)
5:     depth ← number of hours between sourceVertex and targetVertex × 60/resolution
6:     if depth > 1 then
7:         neighborVertices ← CREATEVERTICES(sourceVertex, startDate
8:             + resolution)
9:         for all vertex ∈ neighborVertices do
10:            new Thread(BUILDSUBGRAPH(vertex, startDate + resolution))
11:    return sourceVertex
12:
13: function CREATEVERTICES(prevVertex, date)
14:    meanOutsideTemp ← mean(prevVertex.outsideTemp + outsideTemp(date))
15:    tempDiff ← prevVertex.insideTemp - meanOutsideTemp
16:    temperatureList ← prevVertex.insideTemp ± temperatureChange(tempDiff)
17:    neighborVertices ← new List<Vertex>
18:    for all temperature ∈ temperatureList do
19:        if temperature is within boundaries then
20:            newVertex ← new Vertex(temperature, outsideTemp(date), date)
21:            prevVertex.addEdge(newVertex, energyPrice(date))
22:            neighborVertices.add(newVertex)
23:    return neighborVertices
24:
25: function BUILDSUBGRAPH(prevVertex, date)
26:    if date ≤ targetVertex.targetedDate then
27:        neighborVertices ← CREATEVERTICES(prevVertex, date + resolution)
28:        for all vertex ∈ neighborVertices do
29:            BUILDSUBGRAPH(vertex, date + resolution)
30:    else
31:        if prevVertex is within accuracy of target then
32:            prevVertex.addEdge(targetVertex, 0)
```

4.4 Finding the shortest path with A* search

Based on a graph that is created according to Section 4.3, an A* search algorithm implementation, described in Algorithm 4, is used to find a path that heats the church in the most economically beneficial way. The A* search implementation initially queues the *sourceVertex*, with a cost of zero, of the graph into a priority queue. The algorithm proceeds until the priority queue is empty or the graph's

targetVertex has been reached. In each iteration, the vertex with highest priority is dequeued and for each of its successors a new cost of reaching that successor is calculated. If the successor is not yet discovered, or the successor is already discovered but the newly calculated cost is less than the already existing cost, then the successor is added to the priority queue.

When the algorithm has terminated, and a path from the *sourceVertex* to the *targetVertex* has been discovered, the path can be recovered by traversing from the *targetVertex* until the *sourceVertex*.

Algorithm 4 A* algorithm.

```
1:  function aStar(graph)
2:      queue ← new PriorityQueue<Vertex>(compareVertices)
3:      costs ← new Map<Vertex, Double>
4:      paths ← new Map<Vertex, Vertex>
5:      queue.add(graph.sourceVertex)
6:      costs.put(graph.sourceVertex, 0.0)
7:      while queue is not empty do
8:          c ← queue.pop
9:          if c is graph.targetVertex then break
10:         for all successor ∈ c do
11:             newCost ← costs(c) + successor.price
12:             if successor ∉ costs or newCost < costs(successor) then
13:                 paths.put(successor, c)
14:                 costs.put(successor, newCost)
15:                 queue.add(successor)
16:     shortestPath ← new List<Vertex>
17:     while c is not graph.sourceVertex do
18:         add c first in shortestPath
19:         c ← paths(c)
20:     return shortestPath
21:
22: function compareVertices(vertex1, vertex2)
23:     vertex1Cost ← costs(vertex1) + heuristic(vertex1)
24:     vertex2Cost ← costs(vertex2) + heuristic(vertex2)
25:     return vertex1Cost - vertex2Cost
```

4.4.1 Heuristic function

A heuristic function, shown in Algorithm 5, was developed to assist the A* search. Since a graph, built according to Section 4.3, potentially could have paths that can not reach the targeted vertex, the heuristic function was developed to guide the A* search away from such paths.

As mentioned in Section 2.2.2, it is of importance that the heuristic function is admissible in order to guarantee that A* search finds the shortest path. The heuristic function, explained in Algorithm 5, is based on five conditional statements. The first conditional statement considers vertices that have temperatures above the targeted temperature and can not reach the targeted temperature in time of the targeted date, meaning that a path does not exist from these vertices to the targeted vertex. The second conditional statement considers vertices that are below the targeted temperature, therefore, in order to reach the targeted temperature, the system must heat at least once. In order for the heuristic function to be admissible, only the cheapest energy price within the depth of these vertices and the targeted vertex can be returned as a heuristic value. The third and the forth conditional statements considers vertices above or below the targeted temperature at the targeted date, thus these vertices are irrelevant.

Algorithm 5 Heuristic function.

```
1:  function HEURISTIC(v)
2:      if v.depth < graph.depth && v.insideTemp >
3:              target.insideTemp + accuracy of target  then
4:          if v can not cool to target.insideTemp then
5:              return ∞
6:      else if v.depth < graph.depth && v.insideTemp <
7:              target.insideTemp - accuracy of target then
8:          return cheapest weight within v.depth to graph.depth
9:      else if v.depth == graph.depth && v.insideTemp <
10:             target.insideTemp - accuracy of target then
11:         return ∞
12:     else if v.depth == graph.depth && v.insideTemp >
13:             target.insideTemp + accuracy of target then
14:         return ∞
15:     else
16:         return 0
```

4.5 Scheduler

Since the system consists of many separate components that perform different tasks, a scheduler was introduced to manage these components. The scheduler ensures that the SMHI parser and Nord Pool parser fetch data every hour. In addition, the scheduler ensures that the change in temperature is measured continuously.

As described in Section 1.1.1, a user can interact with a web interface, where the user can add an event when the church is to be heated. If an event is added, the scheduler creates a graph for the event, if energy prices and weather forecasts are available for the date and time of the event. Thereafter, the scheduler performs

the A* algorithm on the graph, and based on the result from the A* algorithm the church is heated.

If there are no events created by a user, the scheduler automatically creates a graph that maintains the temperature above the lowest allowed temperature while still incorporating energy prices to reach the most financially beneficial heat management.

5

Results

In this chapter, results, primarily in terms of relative financial savings, between the price of using the intermittent heating method compared to the price of using the developed system, are presented. In addition, performance was measured, both in terms of building a graph using various graph resolutions as well as in terms of using A* search in comparison to using Dijkstra's algorithm.

5.1 Financial savings

Five simulations were performed in order to evaluate the financial benefits of using the A* heating method compared to the intermittent heating method. In the first simulation, described in Section 5.1.1, both heating methods had the assignment to maintain a temperature of 15 °C, with a lower boundary of 14 °C, over a fixed time interval using a graph resolution of 60 minutes. In the second simulation, described in Section 5.1.2, the lower boundary was, in contrast to the first simulation, set to 14.5 °C. Simulation three and four, described in Section 5.1.3, show how different graph resolutions affect the A* heating method. In the fifth simulation, described in Section 5.1.4, both heating methods had the task to increase the temperature from 15 °C to 20 °C over a fixed time interval.

In order to simulate a credible environment for all simulations, data of how the temperature changes inside a church was gathered from Tunabergs kyrka. This data, presented in Table 5.1, is used in the temperature model in order to build a credible graph. In addition, a power consumption of 25 kW is assumed, for the church's heating system.

Temperature difference	**Temp. change up**	**Temp. change down**
10 °C	+0.32 °C	-0.55 °C
11 °C	+0.30 °C	-0.60 °C

Table 5.1: Temperature data, showing how the temperature increases, Temp. change up, and decreases, Temp. change down, in relation to the difference in temperature inside and outside the church, gathered from Tunabergs kyrka.

5.1.1 Maintaining a fixed temperature with a lower boundary of 14 °C

The result of the simulation to maintain a temperature of 15 °C over a fixed time interval, with a lower boundary of 14 °C and a graph resolution of 60 minutes, using the intermittent heating method and the A* heating method, is shown in Figure 5.1.

In total of the 48 hour simulation, the A* heating method heats for a total of 22 hours while the intermittent heating method heats for 24 hours. This yields a total price of 152.81 SEK for the A* heating method and 170.87 SEK for the intermittent heating method.

The relative financial save can thus be calculated according to Equation 5.1.

$$1 - \frac{152.81}{170.87} = 10.57\% \tag{5.1}$$

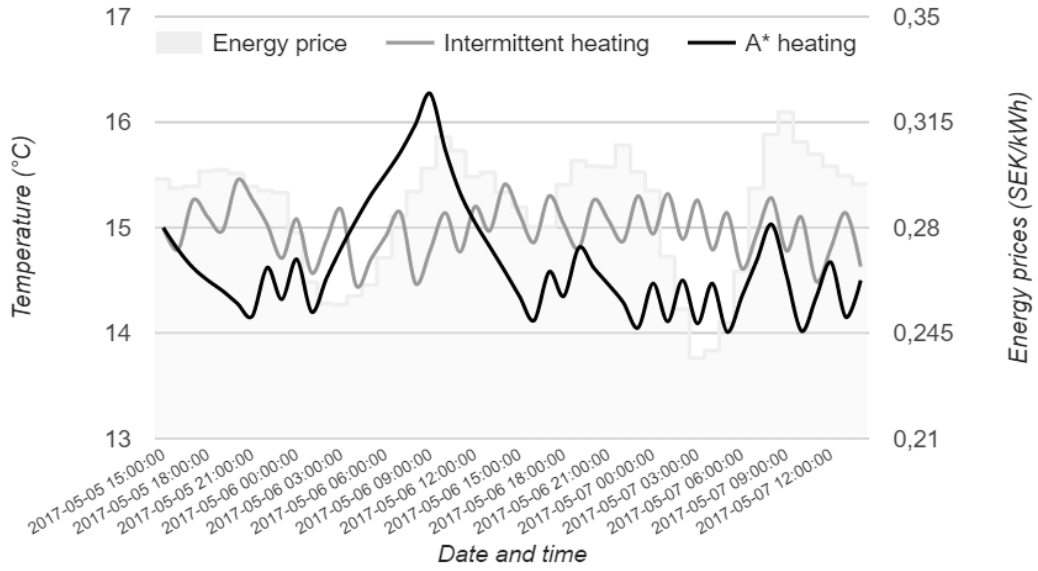

Figure 5.1: Comparison of heating strategies, maintaining a temperature of 15 °C with a lower boundary of 14 °C and a graph resolution of 60 minutes.

Figures 5.2 and 5.3 show when the intermittent heating method and the A* heating method heat in correlation to the energy price. The intermittent heating method spends on average 0.28478 SEK/kWh while the A* heating method spends on average 0.27783 SEK/kWh, which, according to Equation 5.2, results in an average decrease in price of 2.44% when the A* heating method is used. This is due to that the A* heating method primarily heats when the energy price is at its lowest,

while the intermittent heating method does not incorporate the price of energy. Instead the intermittent heating method adjust the heating solely based on the current temperature.

$$1 - \frac{0.27783}{0.28478} = 2.44\% \tag{5.2}$$

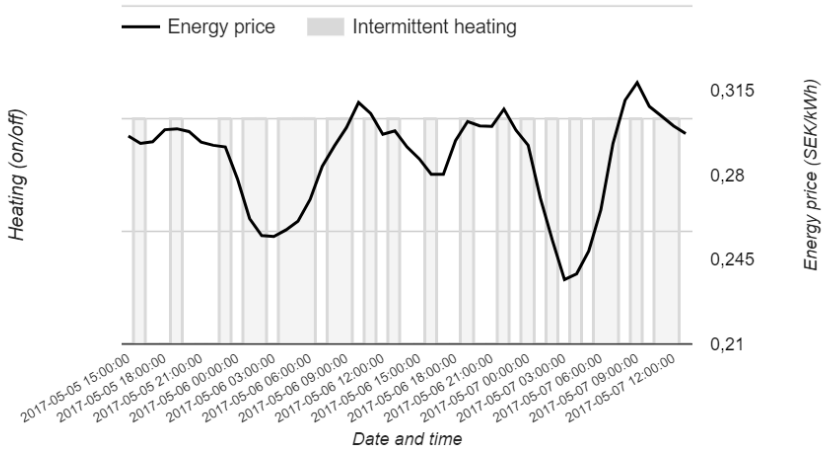

Figure 5.2: Showing when the heating is on or off using the intermittent heating method, maintaining a temperature of 15 °C, with a lower boundary of 14 °C.

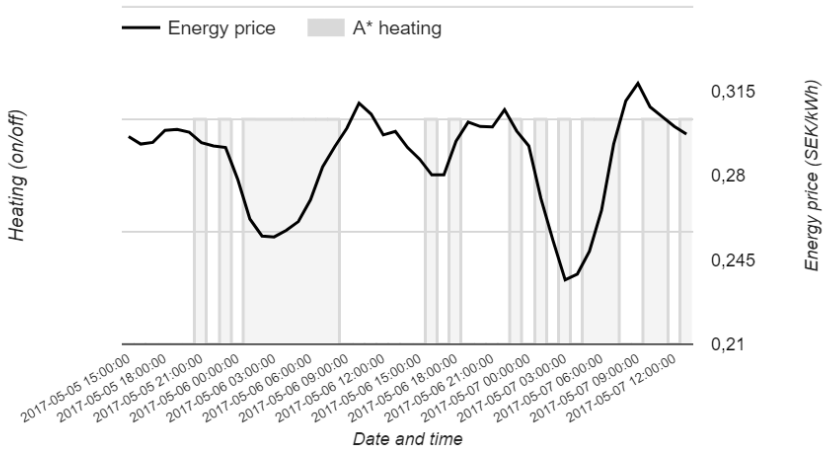

Figure 5.3: Showing when the heating is on or off using the A* heating method, maintaining a temperature of 15 °C, with a lower boundary of 14 °C and a graph resolution of 60 minutes.

5.1.2 Maintaining a fixed temperature with a lower boundary of 14.5 °C

In contrast to Figure 5.1, Figure 5.4 shows the result of a simulation to maintain a temperature of 15 °C over a fixed time interval using a lower boundary of 14.5 °C. As a result, the A* heating method never subsides below 14.5 °C as opposed to the previous simulation where temperatures of 14.0 °C were allowed.

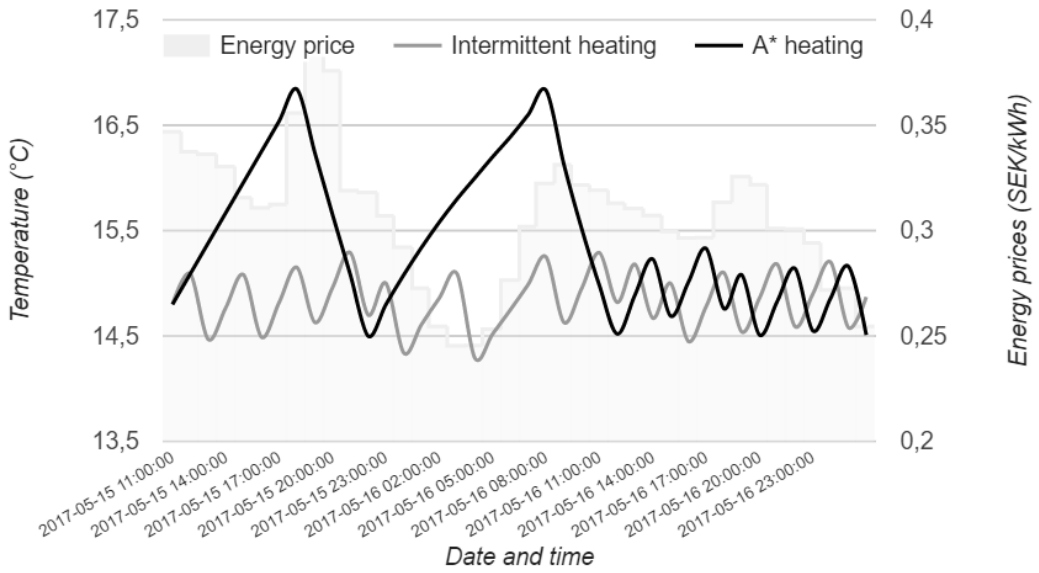

Figure 5.4: Comparison of heating strategies, maintaining a temperature of 15 °C with a lower boundary of 14.5 °C and a graph resolution of 60 minutes.

Out of the 40 hours of simulation, both the A* heating method and the intermittent heating method heat for 27 hours. During the simulation the A* heating method yields a total cost of 200.83 SEK compared to 206.26 for the intermittent heating method.

The relative financial save can thus be calculated according to Equation 5.3.

$$1 - \frac{200.83}{206.26} = 2.63\% \tag{5.3}$$

Figures 5.5 and 5.6 show when the intermittent heating method and the A* heating method heat in correlation to the energy price. The intermittent heating method spends on average 0.30557 SEK/kWh while the A* heating method spends on average 0.29753 SEK/kWh, which, according to Equation 5.4, results in an average decrease in price of 2.63% when the A* heating method is used.

$$1 - \frac{0.29753}{0.30557} = 2.63\% \tag{5.4}$$

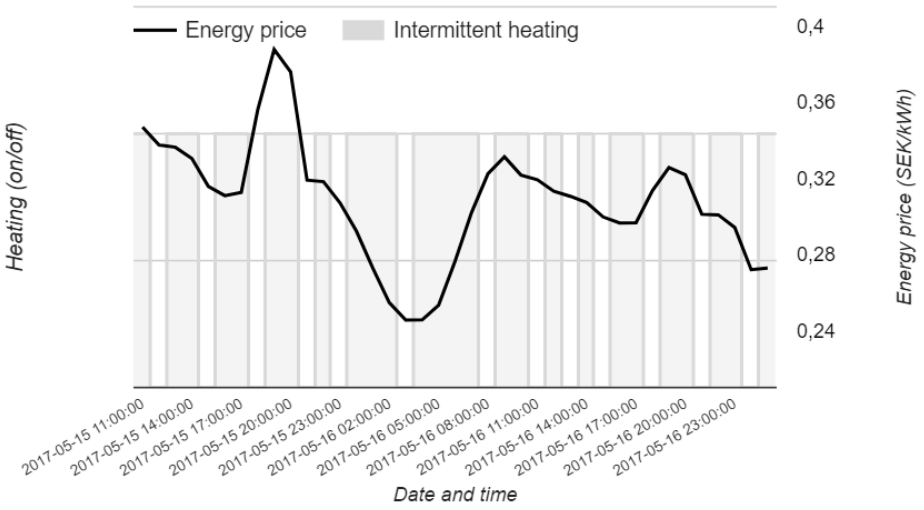

Figure 5.5: Showing when the heating is on or off using the intermittent heating method, maintaining a temperature of 15 °C, with a lower boundary of 14.5 °C.

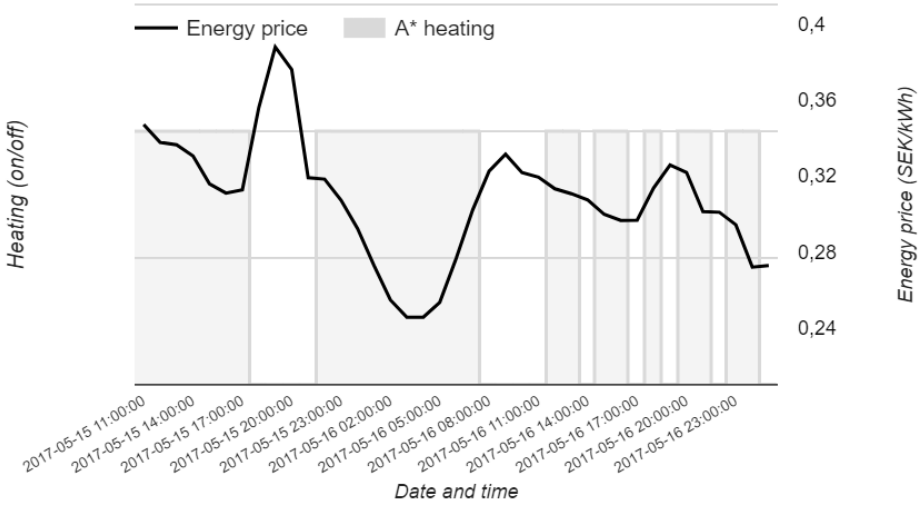

Figure 5.6: Showing when the heating is on or off using the A* heating method, maintaining a temperature of 15 °C, with a lower boundary of 14.5 °C and a graph resolution of 60 minutes.

5.1.3 Maintaining a fixed temperature with various graph resolutions

Figures 5.7 and 5.8 show a simulation with the task to maintain a temperature of 15 °C using the A* heating method, over a fixed time interval, using graph resolutions of 60 minutes, 30 minutes and 20 minutes.

In total out of the 51 hour simulations, both when using graph resolutions of 60 and 30 minutes the A* heating method heats for 32 hours. When a graph resolution of 20 minutes is used, the A* heating method heats for 31 hours and 40 minutes. This yields a total price of 235.00 SEK, 231.81 SEK and 230.47 SEK for using a graph resolution of 60, 30 and 20 minutes, respectively.

The relative financial save of using the A* heating method with a graph resolution of 30 minutes compared to using the A* heating method with a graph resolution of 60 minutes can thus be calculated according to Equation 5.5.

$$1 - \frac{231.81}{235.00} = 1.36\% \tag{5.5}$$

In the same manner, the relative financial save of using the A* heating method with a graph resolution of 20 minutes compared to using the A* heating method with a graph resolution of 30 minutes can be calculated according to Equation 5.6.

$$1 - \frac{230.47}{231.81} = 0.58\% \tag{5.6}$$

The A* heating method spends on average 0.29375 SEK/kWh, 0.28976 SEK/kWh and 0.29112 SEK/kWh when using a graph resolution of 60, 30 and 20 minutes, respectively. The average price of using a graph resolution of 20 minutes is larger than using a graph resolution of 30 minutes. As mentioned above, the A* heating method heats for a shorter period of time when using a graph resolution of 20 minutes compared to a graph resolution of 30 minutes, which results in a lower total price.

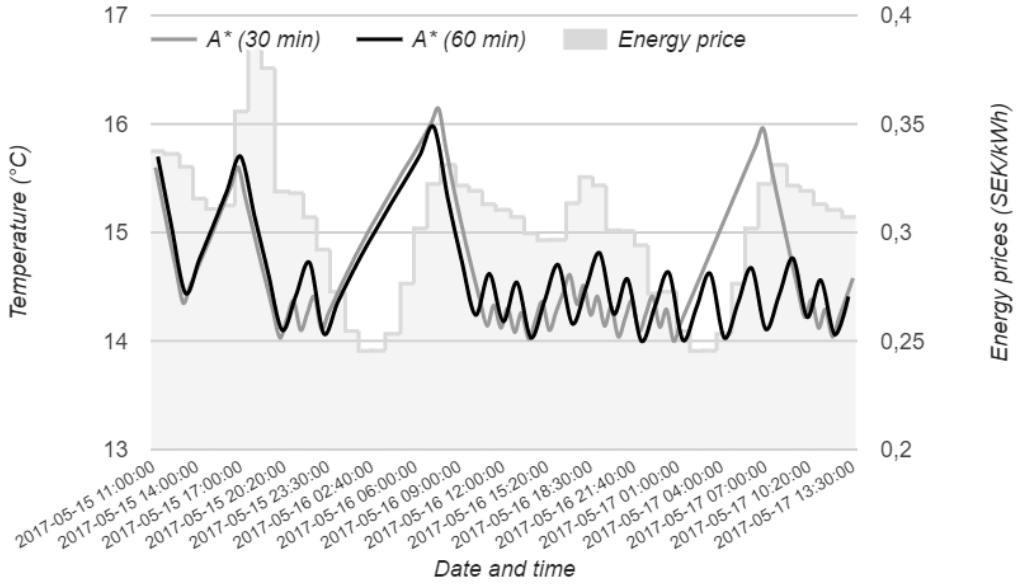

Figure 5.7: Comparison of heating with the A* heating method using a graph resolution of 30 minutes versus a graph resolution of 60 minutes.

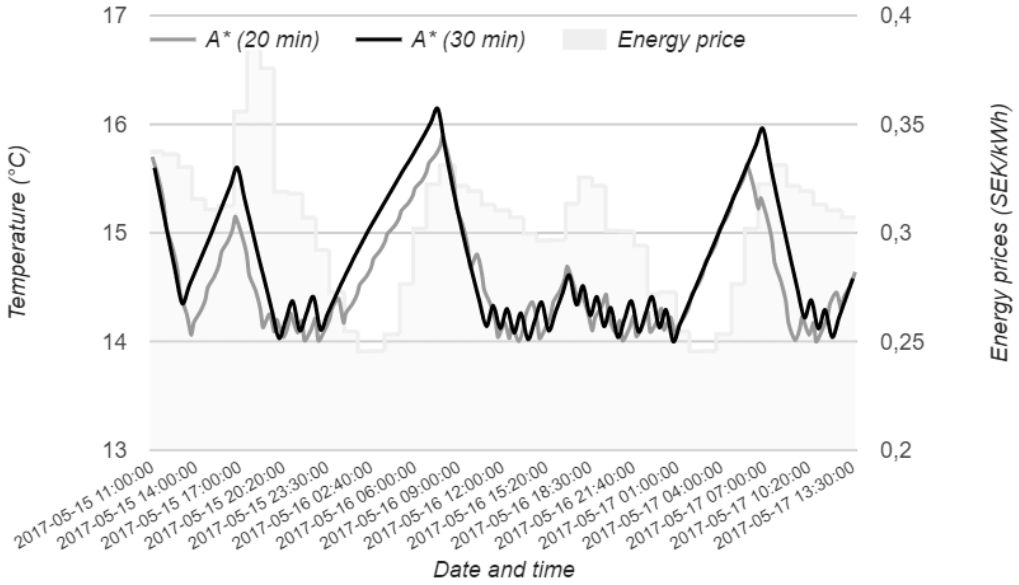

Figure 5.8: Comparison of heating with the A* heating method using a graph resolution of 20 minutes versus a graph resolution of 30 minutes.

5.1.4 Heating to a fixed temperature

The result of the simulation to increase the temperature from 15 °C to a set point of 20 °C using the intermittent heating method and the A* heating method is shown in Figure 5.9.

Out of the 52 hours of simulation, both the intermittent heating method and the A* heating method heats for a total of 26 hours. This yields a total of 178.39 SEK for the A* heating method and 182.40 SEK for the intermittent heating method.

The relative financial saving can thus be calculated according to Equation 5.7.

$$1 - \frac{178.39}{182.40} = 2.20\% \tag{5.7}$$

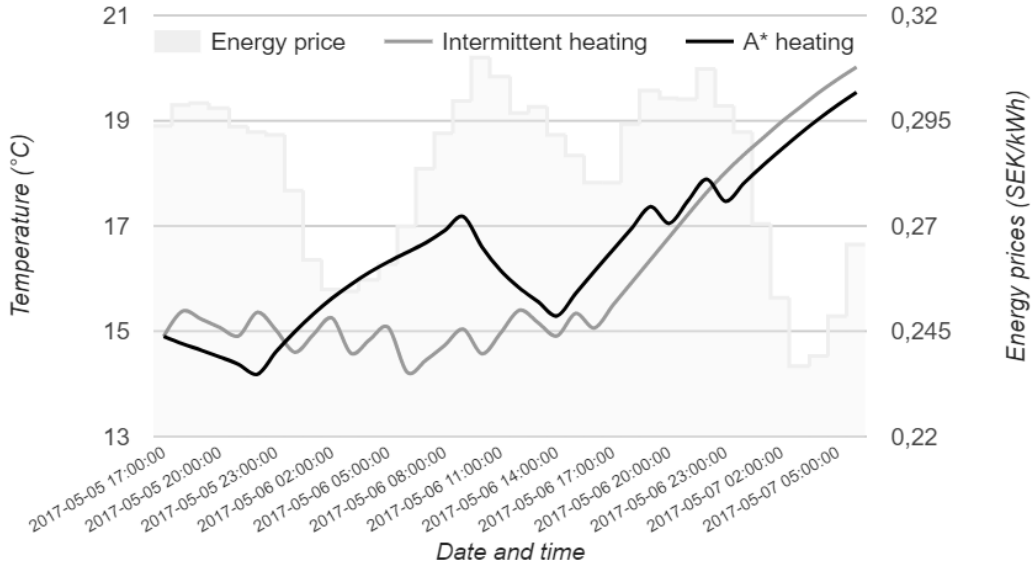

Figure 5.9: Comparison of heating strategies, when increasing the temperature from 15 °C to 20 °C, with a graph resolution of 60 minutes.

Figures 5.10 and 5.11 show when the intermittent heating method and the A* heating method, respectively, heat in correlation to the price of energy. The intermittent heating method spends an average of 0.28060 SEK/kWh while the A* heating spends an average of 0.27440 SEK/kWh resulting in an average decrease in price of 2.20%, when using the A* heating method, as shown in Equation 5.8. Similarly to what is mentioned in Section 5.1.1, this is due to that the A* heating method primarily heats when the energy price is at its lowest, while the intermittent heating method does not incorporate the price of energy. Instead the intermittent heating method adjusts the heating solely based on the current temperature.

$$1 - \frac{0.27440}{0.28060} = 2.20\% \tag{5.8}$$

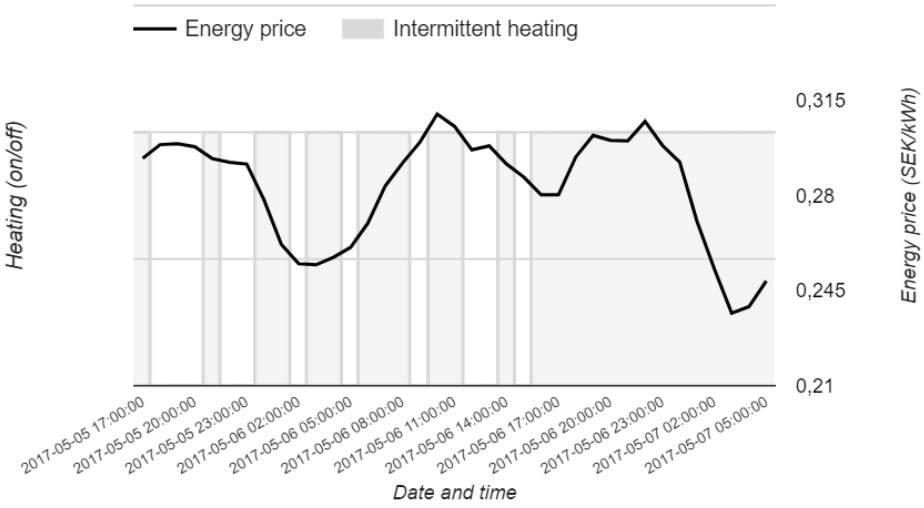

Figure 5.10: Showing when the heating is on or off using the intermittent heating method, increasing the temperature from 15 °C to 20 °C.

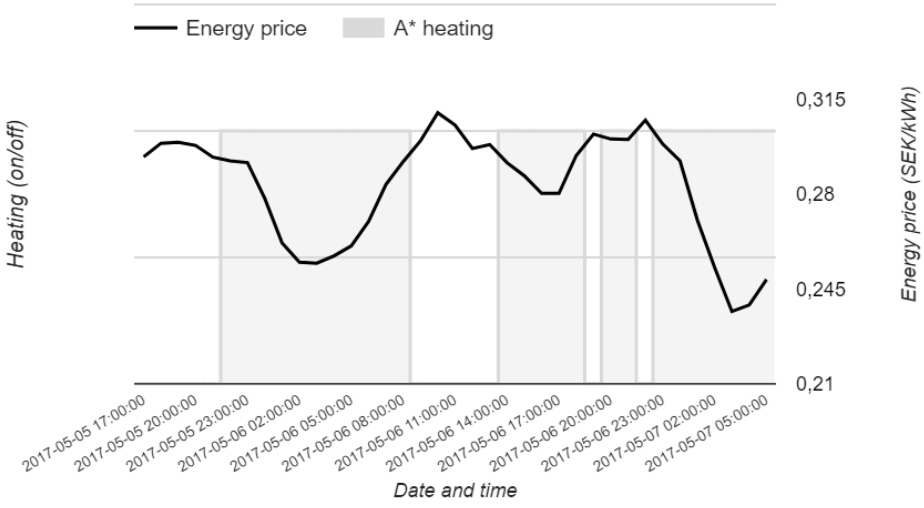

Figure 5.11: Showing when the heating is on or off using the A* heating method, increasing the temperature from 15 °C to 20 °C and a graph resolution of 60 minutes.

5.2 Performance

The performance of the system has been measured in terms of execution time and memory utilization to perform a task. The most time consuming tasks to perform on the system has been identified as:

- Building a graph
- Finding the shortest path in a graph

Therefore, performance was measured on these tasks, using various graph resolutions and by comparing A* search and Dijkstra's Algorithm.

5.2.1 Building a graph with various resolutions

When building a graph, performance was measured in terms of time and memory utilization, using various graph resolutions, on a graph lasting for 52 hours. The result of the performance test is displayed in Table 5.2. The performance test using graph resolutions of 60, 30 and 20 minutes, was performed with one decimal place accuracy. On a graph with a resolution of 5 minutes, the performance test was conducted with an accuracy of two decimal places in order to avoid rounding errors.

Resolution	**No. Vertices**	**No. Edges**	**Execution time**	**Memory usage**
60 minutes	984	1786	0.8693 seconds	116 mb
30 minutes	4441	8506	2.1212 seconds	116 mb
20 minutes	6840	13334	2.5720 seconds	164 mb
10 minutes	13722	27097	22.3697 seconds	168.5 mb
5 minutes	27442	54605	237 seconds	172 mb

Table 5.2: The performance of building a graph using various resolutions.

5.2.2 Dijkstra's Algorithm compared to A* search

In Table 5.3, the performance test on the shortest path algorithms is presented. The test was performed by measuring the run time of Dijkstra's algorithm and A* search, a thousand times, on the same graph.

Depth	No. Vertices	No. Edges	Dijkstra's algorithm	A* search
41	1411	2692	0.00278 seconds	0.00284 seconds
82	5051	9916	0.00825 seconds	0.00551 seconds
123	7792	15397	0.01225 seconds	0.01061 seconds
246	15792	31414	0.01608 seconds	0.01543 seconds
492	31242	62350	0.03569 seconds	0.03047 seconds

Table 5.3: The performance of finding the shortest path on a graph using Dijkstra's algorithm and A* search.

6

Discussion

In this chapter, a discussion of the result is presented. Initially the result is discussed in terms of financial savings, performance and the accuracy of SMHI weather forecasts. Thereafter, potential improvements of the system are discussed as well as the credibility of the result.

6.1 Financial savings

The result, presented in Section 5.1, displays the potential financial savings of using the system, but as the energy prices and outdoor temperature alters by the hour - so will the financial savings.

The results, presented in Section 5.1, show that the smart grid heat management system based on A* search can result in approximately 2.5% in relative financial savings compared to the intermittent heating method. While 2.5% might not sound significant, over time the accumulated savings would result in a substantial economical benefit. For instance, in Section 5.1.2, the intermittent heating method costs 206.36 SEK compared to 200.83 SEK for the A* heating method over a 40 hour period. Assuming these energy prices, over a year this results in 1130 SEK in financial savings, according to Equation 6.1.

$$\frac{365 \cdot 24}{40} \cdot 206.36 \cdot 0.025 = 1130 \text{ SEK} \tag{6.1}$$

Additionally, as mentioned in Section 2.4, the price of energy sometimes increases by several hundred percent and these spikes will be avoided using the algorithm, which will result in a more substantial financial saving.

One interesting aspect of Figures 5.1 and 5.9 is that the A* heating method always ends up with a lower temperature than the intermittent heating method. As mentioned in Section 3.2, the accuracy of the target allows the targeted temperature to be slightly above or below the targeted temperature. The system will always try to find the most economically beneficial path to the targeted temperature and in

general this path will end up at the lowest possible temperature.

An interesting aspect of Figures 5.3 and 5.11 is that the A* heating method tends to heat when the energy price is at its lowest, but not always. The reason for this could be that the difference in temperature inside and outside the church has increased or that it is not necessary to heat at that time in order to reach the targeted temperature.

6.1.1 Maintaining a fixed temperature

As can be seen in Figure 5.1, the A* heating method and the intermittent heating method have two vastly different approaches in maintaining the temperature at 15 °C. The A* heating method heats at first when the price of energy is low and then it maintains a low temperature. When the temperature inside the church is low, it is in general easier to increase the temperature inside the church since the difference in temperature is lower. Therefore, it is both beneficial to have a low temperature and heat when the energy price is low. By staying at a lower temperature, the A* heating method only needs to heat the church for 22 hours while the intermittent heating method needs to heat the church for 24 hours. Naturally, heating for a shorter period of time is likely to result in lower total energy cost.

Figure 5.4 shows a simulation where a lower boundary of 14.5 °C is used. In this simulation, the A* heating method never subsides below said temperature and can therefore not, to the same extent, take advantage of the fact that it is more beneficial to heat when the temperature inside the church is low. In this simulation, the lowest temperature of both heating methods are approximately the same, but the A* heating method still achieves financial saving compared to the intermittent heating method.

As can be seen in Figures 5.7 and 5.8, the A* heating method chooses similar paths for graph resolutions of 60 minutes, 30 minutes and 20 minutes. When using an increased graph resolution, the A* heating method can more proficiently stay above the lower boundary. As a result, the A* heating method heats for a shorter period of time, and still ends up at a temperature closer to the desired temperature, when using a graph resolution of 20 minutes compared to using graph resolutions of 60 minutes and 30 minutes.

6.1.2 Heating to a fixed temperature

Similarly, in Figure 5.9, the A* heating method and the intermittent heating method choose different paths. In this simulation both approaches heat for the same amount of hours and therefore the average energy price is the deciding factor for the financial savings. As can be seen in Figure 5.11 compared to Figure 5.10, the A* heating

method utilizes the times when the energy price is low, in order to heat more efficiently than the intermittent heating method.

6.2 Performance

The performance test presented in Section 5.2.1, shows that the execution time and memory usage increase as the graph resolution increases. As can be seen in Table 5.2, the execution time and memory usage is fairly similar for graph resolutions of 60, 30 and 20 minutes. Thereafter, for graph resolutions of 10 and 5 minutes the execution time increased significantly, mainly due to that more vertices and edges needs to be iterated as the graph resolution increases. In Table 6.1, factors between the depth of a graph and the graph resolution are presented. As mentioned in Section 3.2, more specificly in Equations 3.2 and 3.3, the number of edges and vertices increase exponentially as the depth increases. As stated in the problem description, presented in Section 1.1, the algorithm should not be too computationally heavy, therefore the graph resolution needs to be determined based on the computational power of the microprocessor that the algorithm is deployed on.

Graph resolution	Factor of depth
60 minutes	x1
30 minutes	x2
20 minutes	x3
10 minutes	x6
5 minutes	x12

Table 6.1: Factors of the depth of a graph as the graph resolution increases.

As it turns out, the choice of shortest path algorithm barely affects the execution time of the system, as can be seen in Table 5.3. Although, there are a few interesting aspects worth mentioning. A* search, more precise the heuristic function, entails worse performance in terms of execution time on a graph with a small depth compared to Dijkstra's Algorithm. A reason for the loss in performance, is that the heuristic function takes time to compute, and since the graph is small, the computational time for the heuristic function is not worth it. For a graph with a larger depth, the A* search outperforms Dijkstra's algorithm, solely because of the heuristic function. However, as mentioned above, finding the shortest path in a graph is fast compared to creating the graph, thus any shortest path algorithm is suitable for this application.

6.3 Accuracy of SMHI weather forecasts

As mentioned in section 3.1, the temperatures collected from the SMHI Open Data API are corrected based on how they differ from the temperatures measured outside the facility. This correction will not improve the result for all vertices in a graph but it provides an adjustment so that a majority of them receives more accurate outside temperatures.

Potentially, the difference between the temperature collected from the SMHI Open Data API and the temperature measured outside the facility have a correlation to the time of the day, day of the week and time of the year when the temperature was measured. It would therefore, potentially, be a better solution to not just calculate the mean temperature difference for a couple of days of known data but to rather base it on such parameters. However, such correlations was not examined during the project.

6.4 Future work

The system includes functionality reasonable to develop within the time constraint. However, there are several improvements that could potentially result in a more substantial financial saving for the customer, further described in this section.

6.4.1 Power resolution

The developed system only supports heating that is either on or off. A potential improvement could be to allow the heating to be on with a percentage of the total possible output. This would allow the system to create edges with, for instance, a 50% power output when needed. Such an improvement could result in more potential ways of reaching a wanted temperature, more economical benefit and a better accuracy surrounding the desired temperature. To introduce such functionality would, however, result in higher requirements on performance for both building the graph and applying the shortest path algorithm and, as mentioned in Section 5.2, these are the two most demanding components of the system. The reason for the additional performance requirements is mainly due to the increased number of edges that is added when introducing such functionality. Table 6.2 shows how many additional edges would have been created if such functionality was implemented. Since one of the main requests from the stakeholder was to create a lightweight system the functionality of power resolution was not implemented.

Allowed power output	Factor of Edges
0, 100%	x1
0, 50, 100%	x1.5
0, 25, 50, 75, 100%	x2.5
0, 10, 20, 30, ..., 100%	x5.5
0, 1, 2, 3, ..., 100%	x50.5

Table 6.2: The factor of edges created due to various power resolutions.

6.4.2 A path could not be found with A* search

When a path could not be found using A* search, the system will increase the allowed fault margin for what is considered reaching the targeted temperature. For example, if the system initially has an accuracy of 0.5 °C it would find a path if a vertex exists within a 0.5 °C boundary from the targeted temperature on the targeted date and time. If such a path is not found, the boundary will increase resulting in a higher possibility of a path being found.

Using this method, the system will eventually find a path to the targeted temperature, but the boundary can at the same time increase to an unrealistic magnitude if the targeted temperature is unreachable.

The system will, using this method, always provide a path, but perhaps it would be of interest to in some cases for example provide the customer with an alternative to change the targeted temperature or at least make the customer aware of the error.

6.5 Credibility of result

To avoid a result based on inaccurate values, representative values, as mentioned in Section 5.1, were gathered from Tunabergs kyrka and used in the simulations. These values, in accordance to Equation 2.2, change linearly depending on the difference in temperature inside and outside the church.

To reach a more reliable result, it would be beneficial to deploy the system in one or several churches. This would provide more realistic values of how the temperature changes, how long it takes to heat the church and how much the outside temperature influences the inside temperature. By doing so, values, of how the temperature changes, gathered from the system could be used in the evaluation of financial savings.

7

Conclusion

In this thesis a smart grid heat management system based on A* search has been developed and evaluated in terms of performance and financial savings. As stated in Section 1, the research question of this thesis is: is it feasible to create an algorithm based on a smart grid adaption of church heat that, in terms of financial savings, outperforms the intermittent heating method used today?

A justified answer to the question stated above, based on the result presented in Section 5.1, is that it is, without a doubt, feasible to create an algorithm based on a smart grid adaption of church heat that outperforms the intermittent heating method used today. The result concludes that, by using a smart grid heat management system based on A* search in favor of the intermittent heating method, can result in approximately 2.5% in relative financial savings.

Bibliography

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey", *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] X. Fang, S. Misra, G. Xue, and D. Yang, "Smart grid - the new and improved power grid: A survey", *IEEE Communications Surveys & Tutorials*, vol. 14, no. 4, pp. 944–980, 2012.

[3] L. T. Berger and K. Iniewski, *Smart grid: Applications, communications, and security*. Wiley, 2012.

[4] J. Ekanayake, N. Jenkins, K. Liyanage, J. Wu, and A. Yokoyama, *Smart grid: Technology and applications*, 2. Aufl.;1; John Wiley & Sons, Ltd, 2012.

[5] J. Serra, D. Pubill, A. Antonopoulos, and C. Verikoukis, "Smart hvac control in iot: Energy consumption minimization with user comfort constraints", *SCIENTIFIC WORLD JOURNAL*, vol. 2014, pp. 1–11, 2014.

[6] *Nord pool*, `http://www.nordpoolspot.com/`, [Accessed: 2016-11-27].

[7] *I3tex – ett ovanligt stort litet teknikkonsultföretag*, `http://www.i3tex.com/`, [Accessed: 2017-05-22].

[8] G. Held, *Wireless mesh networks*. Auerbach Publications, 2005, ISBN: 9781420031263.

[9] C. Wang, T. Jiang, and Q. Zhang, *ZigBee® network protocols and applications*, 1st ed. CRC Press, 2014, ISBN: 1439816018.

[10] K. F. Fong, V. I. Hanby, and T. T. Chow, "Hvac system optimization for energy management by evolutionary programming", *Energy and Buildings*, vol. 38, no. 3, pp. 220–231, 2006.

[11] H.-C. Jo, S. Kim, and S.-K. Joo, "Smart heating and air conditioning scheduling method incorporating customer convenience for home energy management system", *IEEE Transactions on Consumer Electronics*, vol. 59, no. 2, pp. 316–322, 2013.

[12] M. Avci, M. Erkoc, and S. S. Asfour, "Residential hvac load control strategy in real-time electricity pricing environment", IEEE, 2012, pp. 1–6, ISBN: 9781467318365.

[13] *Smhi open data api docs - meteorological forecasts*, `http://opendata.smhi.se/apidocs/metfcst/demo_point.html`, [Accessed: 2017-01-18].

[14] H. Ortega-Arranz, D. R. Llanos, and A. Gonzalez-Escribano, *The shortest-path problem: Analysis and comparison of methods*. Morgan & Claypool Publishers, 2015, vol. Lecture Number 1, ISBN: 1627055401.

[15] E. Dijkstra, "A note on two problems in connexion with graphs", *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[16] K. M. Passino and P. J. Antsaklis, "A metric space approach to the specification of the heuristic function for the a algorithm", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 24, no. 1, pp. 159–166, 1994.

[17] W. Benenson, *Handbook of physics*. Springer, 2002, ISBN: 9780387952697.

[18] V. C. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, and G. P. Hancke, "Smart grid technologies: Communication technologies and standards", *IEEE Transactions on Industrial Informatics*, vol. 7, no. 4, pp. 529–539, 2011.

[19] T. L. Sigrid Colnerud Granström and J. Lundgren, "Elområden i sverige - analys av utvecklingen och konsekvenserna på marknaden", *Energimarknadsinspektionen EI R2012:06*, 2012.

[20] H. N. E. Byström, "Extreme value theory and extremely large electricity price changes", *International Review of Economics and Finance*, vol. 14, no. 1, pp. 41–55, 2005.

[21] A. Bejan, P. Vadász, and D. G. Kröger, *Energy and the Environment*. Springer Netherlands, 1999, vol. 15, ISBN: 9401059438.

[22] J. C. Ketterer, "The impact of wind power generation on the electricity price in germany", *Energy Economics*, vol. 44, pp. 270–280, 2014.

[23] H. Lund, "Excess electricity diagrams and the integration of renewable energy", *International Journal of Sustainable Energy*, vol. 23, no. 4, pp. 149–156, 2003.

[24] M. Nicolosi, "Wind power integration and power system flexibility–an empirical analysis of extreme events in germany under the new negative price regime", *Energy Policy*, vol. 38, no. 11, pp. 7257–7268, 2010.

[25] K. Kaygusuz, "Renewable energy: Power for a sustainable future", *Energy Exploration & Exploitation*, vol. 19, no. 6, 2001.

[26] *Hur mäts lufttemperatur? smhi*, `http://www.smhi.se/kunskapsbanken/meteorologi/hur-mats-lufttemperatur-1.3839`, Accessed: 2017-02-24.

[27] W. Jackson, *JSON quick syntax reference*. Apress, 2016, ISBN: 1484218639.

A

Appendix 1

Table A.1 shows the difference between the temperatures measured outside the offices of i3tex AB and the temperatures collected from the closest weather station.

Date and time	SMHI temperature	Measured temperature	Difference
2017-02-09 10:08	-2.19 °C	-1.00 °C	1.19 °C
2017-02-09 11:08	-1.44 °C	-0.90 °C	0.54 °C
2017-02-09 12:08	-1.19 °C	-1.40 °C	-0.21 °C
2017-02-09 13:08	-1.19 °C	-0.80 °C	0.39 °C
2017-02-09 14:08	-1.13 °C	-0.70 °C	0.43 °C
2017-02-09 15:08	-1.00 °C	-1.00 °C	0.00 °C
2017-02-09 16:08	-1.19 °C	-0.70 °C	0.49 °C
2017-02-09 17:08	-1.25 °C	-0.80 °C	0.45 °C
2017-02-09 18:08	-1.38 °C	-1.00 °C	0.38 °C
2017-02-09 19:08	-1.50 °C	-2.20 °C	-0.70 °C
2017-02-09 20:08	-1.50 °C	-2.30 °C	-0.80 °C
2017-02-09 21:08	-1.50 °C	-2.50 °C	-1.00 °C
2017-02-09 22:08	-1.56 °C	-2.60 °C	-1.04 °C
2017-02-09 23:08	-1.56 °C	-3.00 °C	-1.44 °C
2017-02-10 00:08	-1.56 °C	-2.70 °C	-1.14 °C
2017-02-10 01:08	-1.69 °C	-2.40 °C	-0.71 °C
2017-02-10 02:08	-1.56 °C	-2.20 °C	-0.64 °C
2017-02-10 03:08	-1.56 °C	-2.30 °C	-0.74 °C
2017-02-10 04:08	-1.50 °C	-2.50 °C	-1.00 °C
2017-02-10 05:08	-1.44 °C	-2.40 °C	-0.96 °C
2017-02-10 06:08	-1.44 °C	-2.40 °C	-0.96 °C
2017-02-10 07:08	-1.31 °C	-2.30 °C	-0.99 °C
2017-02-10 08:08	-1.13 °C	-2.20 °C	-1.07 °C
2017-02-10 09:08	-1.00 °C	-1.90 °C	-0.90 °C
2017-02-10 10:08	-0.81 °C	-1.50 °C	-0.69 °C
2017-02-10 11:08	-0.50 °C	-1.10 °C	-0.60 °C
2017-02-10 12:08	-0.25 °C	-0.80 °C	-0.55 °C
2017-02-10 13:08	0.13 °C	-0.40 °C	-0.53 °C
2017-02-10 14:08	0.38 °C	0.00 °C	-0.38 °C
2017-02-10 15:08	0.50 °C	0.30 °C	-0.20 °C

2017-02-10 16:08	0.75 °C	0.30 °C	-0.45 °C
2017-02-10 17:08	0.56 °C	0.40 °C	-0.16 °C
2017-02-10 18:08	0.81 °C	0.40 °C	-0.41 °C
2017-02-10 19:08	1.06 °C	0.50 °C	-0.56 °C
2017-02-10 20:08	1.19 °C	0.80 °C	-0.39 °C
2017-02-10 21:08	1.19 °C	0.80 °C	-0.39 °C
2017-02-10 22:08	1.25 °C	0.30 °C	-0.95 °C
2017-02-10 23:08	1.13 °C	-0.10 °C	-1.23 °C
2017-02-11 00:08	1.13 °C	-0.60 °C	-1.73 °C
2017-02-11 01:08	0.63 °C	-1.00 °C	-1.63 °C
2017-02-11 02:08	0.56 °C	-0.90 °C	-1.46 °C
2017-02-11 03:08	0.00 °C	-0.80 °C	-0.80 °C
2017-02-11 04:08	-0.19 °C	-0.70 °C	-0.51 °C
2017-02-11 05:08	-0.63 °C	-1.30 °C	-0.67 °C
2017-02-11 06:08	-1.38 °C	-1.90 °C	-0.52 °C
2017-02-11 07:08	-1.81 °C	-2.60 °C	-0.79 °C
2017-02-11 08:08	-2.00 °C	-3.10 °C	-1.10 °C
2017-02-11 09:09	1.63 °C	-3.50 °C	-5.13 °C
2017-02-11 10:09	2.13 °C	-3.20 °C	-5.33 °C
2017-02-11 11:09	-0.13 °C	-2.20 °C	-2.07 °C
2017-02-11 12:09	0.25 °C	-1.20 °C	-1.45 °C
2017-02-11 13:09	0.06 °C	-0.40 °C	-0.46 °C
2017-02-11 14:08	0.50 °C	0.00 °C	-0.50 °C
2017-02-11 15:09	0.50 °C	0.00 °C	-0.50 °C
2017-02-11 16:09	-0.13 °C	-0.60 °C	-0.47 °C
2017-02-11 17:09	-0.19 °C	-1.80 °C	-1.61 °C
2017-02-11 18:09	-0.13 °C	-2.80 °C	-2.67 °C
2017-02-11 19:09	0.00 °C	-3.60 °C	-3.60 °C
2017-02-11 20:09	0.13 °C	-4.10 °C	-4.23 °C
2017-02-11 21:09	0.00 °C	-4.70 °C	-4.70 °C
2017-02-11 22:09	0.00 °C	-5.10 °C	-5.10 °C
2017-02-11 23:09	-0.13 °C	-5.40 °C	-5.27 °C
2017-02-12 00:09	-0.25 °C	-5.60 °C	-5.35 °C
2017-02-12 01:09	-0.25 °C	-5.90 °C	-5.65 °C
2017-02-12 02:09	-0.19 °C	-5.70 °C	-5.51 °C
2017-02-12 03:09	-0.13 °C	-5.50 °C	-5.37 °C
2017-02-12 04:09	-0.06 °C	-5.60 °C	-5.54 °C
2017-02-12 05:09	0.00 °C	-5.70 °C	-5.70 °C
2017-02-12 06:09	-0.31 °C	-5.90 °C	-5.59 °C
2017-02-12 07:09	-0.25 °C	-6.30 °C	-6.05 °C
2017-02-12 08:09	-0.25 °C	-4.40 °C	-4.15 °C
2017-02-12 09:09	-0.06 °C	-2.90 °C	-2.84 °C
2017-02-12 10:09	0.31 °C	-1.60 °C	-1.91 °C
2017-02-12 11:09	0.50 °C	-0.80 °C	-1.30 °C

2017-02-12 12:09	0.56 °C	-0.20 °C	-0.76 °C
2017-02-12 13:09	0.75 °C	0.10 °C	-0.65 °C
2017-02-12 16:09	0.31 °C	-0.10 °C	-0.41 °C
2017-02-12 19:09	-0.94 °C	-3.40 °C	-2.46 °C
2017-02-12 22:09	-2.56 °C	-4.90 °C	-2.34 °C
2017-02-13 01:09	-4.06 °C	-4.30 °C	-0.24 °C
2017-02-13 07:09	-5.31 °C	-1.90 °C	3.41 °C
2017-02-13 10:09	2.25 °C	-2.10 °C	-4.35 °C
2017-02-13 11:09	0.50 °C	-1.00 °C	-1.50 °C
2017-02-13 12:09	0.63 °C	-0.30 °C	-0.93 °C
2017-02-13 13:09	0.50 °C	1.20 °C	0.70 °C
2017-02-13 14:09	0.50 °C	1.10 °C	0.60 °C
2017-02-13 15:09	0.31 °C	0.80 °C	0.49 °C
2017-02-13 16:09	0.13 °C	0.00 °C	-0.13 °C
2017-02-13 17:09	-0.44 °C	-1.70 °C	-1.26 °C
2017-02-13 18:09	-0.81 °C	-2.30 °C	-1.49 °C
2017-02-13 19:09	-1.13 °C	-2.70 °C	-1.57 °C
2017-02-13 20:09	-1.56 °C	-2.70 °C	-1.14 °C
2017-02-13 21:09	-1.81 °C	-3.10 °C	-1.29 °C
2017-02-13 22:09	-2.13 °C	-3.70 °C	-1.57 °C
2017-02-13 23:09	-2.50 °C	-3.90 °C	-1.40 °C

Table A.1: A sample of temperatures collected from the SMHI Open Data API and temperatures measured at the office of i3tex AB as well as the difference between them.