Branch and Bound algorithms in greenhouse climate control

Author: Marleen Hermelink
MSc Research report
Universidad de Málaga in co-operation with Wageningen University
Computer Architecture and Operations Research and Logistics

Grant TIN2015-66680-c2-2-R from the Spanish state
in part financed by the European Regional Development Fund (ERDF)
Supervisors: Dr. E.M.T. Hendrix and Dr.R. Haijema

June 28, 2016

Branch and Bound Algorithms in Greenhouse Climate Control

Marleen Hermelink
Student ID: 92092329050
MSc program: Master in Biosystems Engineering
Specialisation: Operations Research and Logistics
Supervisor: Eligius Hendrix

June 28, 2016

Abstract

The horticultural sector has become an increasingly important sector of food production, for which greenhouse climate control plays a vital role in improving its sustainability. One of the methods to control the greenhouse climate is Model Predictive Control, which can be optimized through a branch and bound algorithm. The application of the algorithm in literature is examined and analyzed through small examples, and later extended to greenhouse climate simulation. A comparison is made of various alternative objective functions available in literature. Subsequently, a modified version of the B&B algorithm is presented, which reduces the number of node evaluations required for optimization. Finally, three alternative algorithms are developed and compared to consider the optimization problem from a discrete to a continuous control space.

Keywords: Branch and Bound, Model Predictive Control, Optimization, greenhouse climate, control action, control sequence, state variable.

Contents

1 Introduction

In the past two decades, horticultural production has undergone a technological revolution. Producers around the world have become increasingly competitive with help of the new available technologies, and so also have producers in the Mediterranean area [3]. Where a few years ago 100 tons of tomato per hectare was considered impressive, now a harvest of 300 tons per hectare is quite standard [3]. This exorbitant productivity has been achieved mainly through the implementation of greenhouses. These are considered ideal for growing crops, as they provide an enclosed environment which allows for controlled climate and fertigation [28]. Due to this highly productive nature, the greenhouse industry in the Mediterranean has become increasingly important to ensure global food security. Spain, for example, is often referred to as La Huerta de Europa (the kitchen garden of Europe), producing approximately 50% of all lettuce, 30% of all tomatoes, and 18% of all vegetables consumed in the EU [34]. These figures show the scale and importance of the greenhouse industry, hence clarifying the extensive research which is currently performed on it.

Ongoing research on greenhouse production is very broad, but a large fraction of the issues being investigated are related to increasing the sustainability of the production system. Matters such as improved water use efficiency, reduced risks of pests and diseases and reduction of chemical residues on the crop and in the soil are all becoming increasingly important issues which the industry must address. Simultaneously, they must keep costs low and product quality high [25]. Therefore, one of the main issues requiring research at the moment is energy consumption [3]. Despite the mild climate, greenhouses in the Mediterranean still require some degree of heating in the winter and cooling in the summer [9]. A reduction of energy demand for these seasons could lower the sector's production costs while simultaneously improving its environmental performance. Furthermore, due to the increasing concern on global warming, it is expected that regulations concerning CO_2 emission will become more stringent [3], so that a decrease in energy consumption will likely be necessary for the greenhouse sector.

One of the main methods which has been implemented to lower energy consumption is climate control [28]. This consists of regulating the greenhouse climate so as to avoid extreme conditions which can damage the crop and to achieve suitable temperature integrals to speed up crop development and improve its quality [28]. Accurate and optimal climate control contributes to avoiding unnecessary heating and cooling, hence reducing energy consumption of the greenhouse. However, achieving this is not a simple task. Extensive research has been performed already on the control and optimization of greenhouse climate, yet many of the methods proposed thus far have some computational or practical drawbacks [7]. Therefore, it is interesting to further investigate an alternative method which has emerged during the

past few years, which makes use of the Branch and Bound algorithm to optimize climate regulation. This research focuses on the possibilities of this new approach, and aims to evaluate its potential as a climate control mechanism. In order to do this, the research questions in the following section are investigated.

1.1 Research Questions

Main Research Question:
To what extent can Branch and Bound algorithms be applied to optimize climate control in greenhouses?

Specific Research Questions:

1. How has climate control in Mediterranean greenhouses been achieved thus far?
2. What kind of investigation has already been done on the topic of branch and bound algorithms for climate optimization?
3. How do the Branch and Bound algorithms for greenhouse climate optimization work?
4. What are the advantages and disadvantages of this optimization method?
5. How can the current algorithms be improved?
6. Can the algorithm be applied on a real data set from a greenhouse?

2 Literature Background

2.1 Climate in the Greenhouse

In order to delve into the application of the Branch and Bound algorithm for climate regulation, first some understanding of the greenhouse climate is required. The climatic conditions in the greenhouse are of great importance to the crop production. The climate affects not only the yield of the crop, but also the quality of the products [28]. To maximize the economic benefit of the horticultural farm, a balance must be found between improving production and the costs of obtaining the right climatic conditions [28]. Most of the crops grown in greenhouses are adapted to temperatures between 17-27°C with a lower and upper limit of 10°C and 35°C [3]. They require a humidity within a range of 60-80% [6]. Temperatures outside this range lead to sub-optimal crop production and even to permanent crop damage [1]. Too high humidity levels can lead to the development of fungi on the crop, while a humidity that is too low can cause water stress [3] [28], both of which lead to a decrease in production.

Inside the greenhouse, the climatic conditions which can be controlled are temperature, humidity, Photosynthetically Active Radiation (PAR) and CO_2 concentration [28]. Temperature is the condition that influences crop growth most directly, and is thus traditionally the main focus of climate control inside the greenhouse [3][28]. On the other hand, humidity has an indirect effect on crop growth through its influence on crop transpiration, and should thus also be taken into consideration. However, humidity and temperature are highly inversely correlated [28]. To address this, the general solution is to keep temperature as the main control variable, but to adjust the desired temperature depending on the relative humidity [28].

The climate inside the greenhouse is affected by the outside climatic conditions (such as air temperature, wind speed, humidity, etc.) as well as by dynamic processes inside [25]. These factors are considered disturbances, as they can cause the temperature inside the greenhouse to deviate from the desired set point [28]. A complete overview of the main disturbances is given in Figure 1. To correct the temperature for these disturbances, the greenhouse has some main actuators: heating, ventilation, shading screens and fog system [28],[5]. Heating can be applied through hot air, used to avoid sudden temperature drops, or through hot water, which is applied in a more permanent way [9]. Ventilation can be performed passively, by opening windows, or forced, which is done by bringing in or extracting air with fans [9]. Screens are placed above the crop and can be folded or unfolded to create or remove shade and the fog system can be turned on to increase the humidity and reduce the temperature. Control engineering allows the automatic control of these actuators to keep track of the reference values despite the disturbances acting on the system [27].

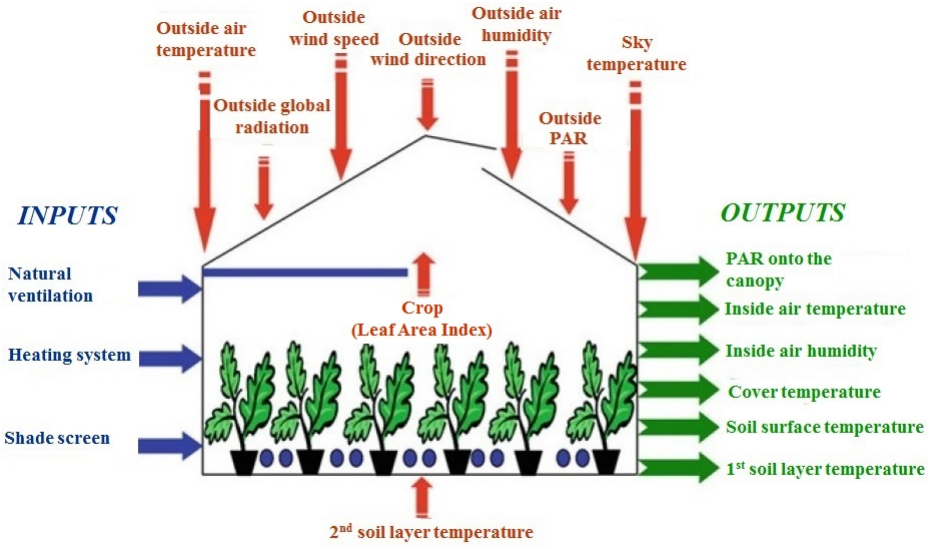


Figure 1: Inputs and outputs of climatic conditions in the greenhouse. Taken from [28].

2.2 Common Control Systems

The main goal of climate control is to keep the temperature inside the greenhouse at the desired set-point. To do this, climate control systems must determine what the best control action is at a certain moment in time. A control action can be, for example, the opening of a vent, or turning on the heating. Due to the complexity of the climate, the changes caused by actuators cannot be fully described by linear models [28]. Despite its complex nature, a number of control systems have been developed which enable climate regulation. In order to understand how climate control is generally achieved, a brief overview is given of the most commonly used systems:

Proportional Integral Derivative (PID) controllers: These have been widely used for greenhouse climate control [28]. This technique is based on simplified transfer function models obtained from reaction curve tests. Three parameters must be determined beforehand (proportional gain, integral time, and derivative time) to tune the controller. In general, the goal of the controller is to quickly correct temperature deviations from the set-point. However, due to the dynamic nature of the greenhouse climate, it is difficult the obtain a good performance using fixed parameter values. Therefore, this technique is often combined with other control schemes by actively adjusting its parameter values or combining it with a feed-forward loop to account for measured disturbances before they alter the climate [28].

Gain Scheduling controllers: This type of controller actively changes the control parameters based on a table or function calculated previously.

A drawback is that the construction of the table or function relating the parameters with the measured variables requires extensive simulation. Furthermore, there are little results on robustness, performance or stability of the system [28].

Feed Forward controllers: This type of controller is based on physical laws and measured data. The system measures the disturbances to the greenhouse climate and tries to compensate their effect before they have caused a deviation from the set-point. However, this system requires a mathematical model of the process, which is difficult to build accurately, and is thus often combined with a feedback system [28].

Besides the previously described methods, there are many other types of more complicated control systems. Some of these are model predictive control, robust control, adaptive control, optimal control and many more, as well as combinations of the different systems [28]. However, the focus of this report is on model predictive control, as this type of control can make use of Branch and Bound algorithms. For a detailed explanation of the other types of control, the book [28] can be consulted.

3 Model Predictive Control (MPC)

3.1 General Overview

MPC is a general methodology for solving control problems in time [33], and can be applied to climate control in the greenhouse [28]. The application of MPC requires the discretization of time. MPC samples the state of the system at certain time instances. At each of these time instances, MPC computes the optimal control action, and implements it for the corresponding interval. Then it moves to the next time interval, and repeats the same process. As the control action is calculated at each sampling instant, MPC is said to be solved *on-line* [18]. Hence, it does not require a pre-computed control law, allowing it to handle control problems which would otherwise require much more difficult computations [18]. MPC differs from other control methods in that it not only looks at the past, but also at the future. This can be described through an analogy with a driver. Regular control mechanisms (such as PIDs) can be seen as drivers looking only in their rear-view mirrors, because they only correct for what has already happened. On the other hand, MPC can be compared with a driver who looks in his mirror but also looks ahead, since MPC not only corrects for errors, but also predicts future effects of disturbances [26].

To include the future in its control strategy, MPC has a certain prediction horizon. This is the number of discrete time intervals ahead that MPC takes into account to choose a control action [33]. The dynamics of the MPC algorithm are shown in Figure 2. At a certain time instant, MPC first samples the state of the system. It then uses a (nonlinear) model of the process as a transition function to predict the state of the system for different control actions over the prediction horizon. An optimization technique is used to compute the best control action sequence over the prediction horizon, based on the minimization of an objective function [20]. From the chosen control action sequence, only the first control action is actually implemented [33]. After the implementation, MPC moves to the next time instant, the prediction horizon also shifts one time unit, and the process starts again. For this reason, MPC is also said to follow the receding (or rolling) horizon principle [33].

3.2 Mathematical Notation

Mathematically, MPC can be described as follows. The control problem is concerned with a total period of time. The time is discretized into intervals of equal length. At each time interval k, an optimal control action u_k must be determined, as shown in Figure 3. In order to accomplish this, at the k^{th} interval a prediction horizon is considered of PH number of time intervals ahead. A temporary set of optimal control actions $\boldsymbol{v}_i$, with $i = 1, \ldots, PH$, is computed for the intervals in the prediction horizon. This is called the

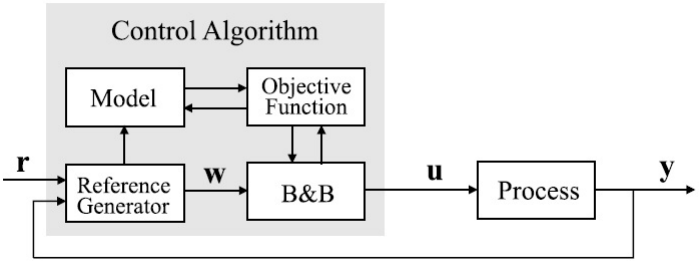


Figure 2: Model Predictive Control scheme using Branch and Bound as optimization technique. Taken from [20].

optimal control sequence, V_k:

$$V_k = (\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{PH}), \tag{1}$$

$$V_k \in \mathbb{V}$$

where $\mathbb{V}$ is the set of all sequences of size PH formed as combinations of control alternatives [7]. It must be noted that the control actions $\boldsymbol{v}$ remain constant from the control horizon CH to the prediction horizon PH. Therefore, all the control actions $\boldsymbol{v}_i$ at intervals $i > CH$ are kept constant at $\boldsymbol{v}_{CH}$ [20]. So

$$\boldsymbol{v}_i = \boldsymbol{v}_{CH} \qquad \forall\ i = CH+1, \ldots, PH \tag{2}$$

From the optimal control sequence V_k, the first control action $\boldsymbol{v}_1$ is implemented as optimal control action u_k [7]:

$$u_k = \boldsymbol{v}_1 \tag{3}$$

After u_k has been implemented, the MPC algorithm moves to the next interval $k+1$, finds V_{k+1}, and implements the new $\boldsymbol{v}_1$ as u_{k+1}. Therefore, each interval k can be considered as a separate optimization problem to find the optimal sequence V_k.

A control sequence V is optimized given a certain objective function $J(V)$. In most cases J is a cost function, describing the control goals of the user [20], such as minimum deviation from a setpoint, minimum control effort, and minimum water and energy use. The possibilities for the objective function are discussed in Section 5.3.

A control action $\boldsymbol{v}$ contains the settings of the multiple actuators, such as heater, window, fog system, etc. The control action $\boldsymbol{v}$ is thus a control vector. Given NA number of actuators, each control vector $\boldsymbol{v}$ contains the

control variables $v_1, \ldots, v_{NA}$:

$$\boldsymbol{v} = \begin{pmatrix} v_1 \\ \vdots \\ v_{NA} \end{pmatrix} \tag{4}$$

Notice that the control action $\boldsymbol{v}$ is a *vector*, and is indicated in bold font in this report, while the control *variable* v is in non-bold, since v is an element of the vector $\boldsymbol{v}$. The control vector $\boldsymbol{v}$ at interval i is indicated as $\boldsymbol{v}_i$, where the index should not be confused as an element indicator. Rather, the a^{th} element of $\boldsymbol{v}$ is expressed as v_a. Therefore, the control variable of actuator a in interval i can be indicated as v_{ia}. The control space of each actuator v_a is the discrete set Ω_a. In total, the problem has $PH \times NA$ number of variables in a control sequence V:

$$V = \begin{pmatrix} v_{11} & v_{21} & \cdots & v_{PH1} \\ v_{12} & v_{22} & \cdots & v_{PH2} \\ \vdots & \vdots & \ddots & \vdots \\ v_{1NA} & v_{2NA} & \cdots & v_{PHNA} \end{pmatrix} \tag{5}$$

$$\text{with } v_{ia} \in \Omega_a \qquad \forall\, a = 1, \ldots, NA \quad \forall\, i = 1, \ldots, PH$$

The state of the system at the beginning of interval i is denoted as $\boldsymbol{s}_i$. In the case of greenhouse climate control, the system at hand is the climate inside the greenhouse. The state vector $\boldsymbol{s}$ contains NS number of state conditions s, such as temperature, humidity, PAR, etc. Again, notice that the state vector $\boldsymbol{s}$ is denoted in bold, while the elements $s_1, \ldots, s_{NS}$ of the vector are indicated in non-bold:

$$\boldsymbol{s} = \begin{pmatrix} s_1 \\ \vdots \\ s_{NS} \end{pmatrix} \tag{6}$$

The MPC algorithm uses a mathematical model to simulate the greenhouse climate in order to predict the changes in state. The transition from the current state $\boldsymbol{s}_i$ to the future state $\boldsymbol{s}_{i+1}$, with $i = 1, \ldots, PH$, is predicted through the transition function T. The future state $\boldsymbol{s}_{i+1}$ depends on the current state $\boldsymbol{s}_i$, and on the current control action $\boldsymbol{v}_i$ [20]. In other words, the transition function T predicts the state $\boldsymbol{s}_{i+1}$ of the system at the beginning of the *next* interval, given that control action $\boldsymbol{v}_i$ is implemented in the *current* interval:

$$\boldsymbol{s}_{i+1} = T(\boldsymbol{v}_i, \boldsymbol{s}_i) \qquad \forall\ i = 1, \ldots, PH \tag{7}$$

3.3 General Problem Statement

As discussed in the previous section, MPC consists of repetitively optimizing control sequence V. One of such optimization problems is formulated below.

$$
\begin{array}{lrcll}
\min\limits_{V} & J(V) & & & \\
\text{with} & V & = & (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{PH}) & \\
 & \boldsymbol{v} & = & (v_1, \ldots, v_{NA}) & \\
 & \boldsymbol{s} & = & (s_1, \ldots, s_{NS}) & \\
\text{s.t.} & \boldsymbol{s}_{i+1} & = & T(\boldsymbol{v}_i, \boldsymbol{s_i}) & \forall\, i = 1, \ldots, PH \\
 & \boldsymbol{v}_i & = & \boldsymbol{v}_{CH} & \forall\, i = CH+1, \ldots, PH \\
 & v_{ia} & \in & \Omega_a \qquad \forall\, i = 1, \ldots, PH & \forall\, a = 1, \ldots, NA
\end{array}
\tag{8}
$$

The optimization technique used to solve V will be the main focus of this report, as this is where a Branch and Bound algorithm can be applied. However, to explore the B&B application, it is necessary to fully understand the MPC optimization problem. Therefore, a stylized example is used in the next section to illustrate the concept.

3.4 Example Problem

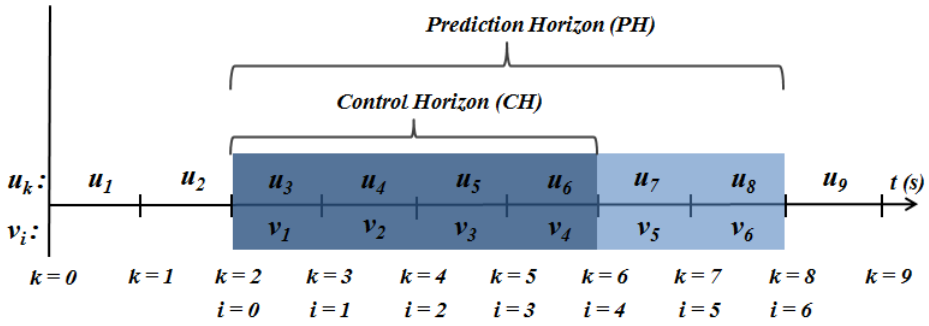


Figure 3: Discretized time scale for a Model Predictive Control algorithm.

A simple example of greenhouse climate control through MPC is depicted in Figure 3. Assume time is discretized into intervals of one minute, so the interval number k is equal to the number of minutes which have passed so far. At this moment in time, two minutes have passed already since the start of the control observations, so we are at $k = 2$. The problem has a prediction horizon $PH = 6$ and a control horizon $CH = 4$. There is only one actuator v_1 in the greenhouse; a heater, which can be turned off or on

with 0 or 1 respectively. Since the problem involves only one actuator, the control vectors $\boldsymbol{v}_i$, with $i = 1, \ldots, PH$ contain only one element:

$$\boldsymbol{v} = (v_1) \qquad\qquad v_1 \in \{0, 1\}$$

The MPC algorithm must determine for each interval i in the prediction horizon whether it would be better to turn the heater on or off; in other words, the control actions $\boldsymbol{v}_i$ comprising the control sequence V. Table 1 illustrates a few possible control sequences. Control sequence c, for example, would imply that the heater v_1 is turned on during the first minute at interval $i = 1$, turned off the next minute in interval $i = 2$, then turned back on again, and left on for the rest of the prediction horizon. Notice that in all control sequences, the last three control actions $\boldsymbol{v}_4$, $\boldsymbol{v}_5$, and $\boldsymbol{v}_6$ are the same. That is because after the control horizon CH, the MPC algorithm does not change the control action anymore, and keeps it constant at $\boldsymbol{v}_{CH}$, as specified in Eq.(2). Since in this problem $CH = 4$, it follows that $\boldsymbol{v}_{CH} = \boldsymbol{v}_4 = \boldsymbol{v}_5 = \boldsymbol{v}_6$.

Control Sequence V	$\boldsymbol{v}_1$	$\boldsymbol{v}_2$	$\boldsymbol{v}_3$	$\boldsymbol{v}_4$	$\boldsymbol{v}_5$	$\boldsymbol{v}_6$
a	1	1	1	1	1	1
b	1	1	1	0	0	0
c	1	0	1	1	1	1
d	0	1	0	1	1	1
e	0	0	1	0	0	0
f	0	0	0	0	0	0

Table 1: Example of possible control sequences for a MPC problem with one actuator $v_1 \in \{0, 1\}$, prediction horizon $PH = 6$, and control horizon $CH = 4$.

The MPC algorithm must now proceed by choosing which control sequence V is optimal. To do this, first the effect of each sequence on the climate is predicted with a model. In this example, the only state that will be considered is the air temperature inside the greenhouse. Therefore, the state vector $\boldsymbol{s}$ contains a single element s_1 for the temperature:

$$\boldsymbol{s} = (s_1)$$

The climate is modeled in a rough manner, by assuming that turning on the heater ($v_1 = 1$) would increase the inside temperature by 1°C per interval, while leaving it off ($v_1 = 0$) will cause no change. The transition function T of the problem would thus be:

$$T(\boldsymbol{v}, \boldsymbol{s}) = \boldsymbol{s} + \boldsymbol{v}$$

Therefore, sequence a, for example, would cause a total temperature increase of 6°C. Knowing the transition function, it can be used to evaluate a sequence V with respect to the desired state of the system, such as a setpoint or range. For example, assume that at the moment the inside temperature is 23°C, while the setpoint is 25°C. It would thus make sense to choose a sequence which turns the heater on during the first two intervals, and then leaves it off. This is the path that would follow the set point most accurately. But there might be more objectives other than just accurately following the setpoint. For example, we might also want to minimize energy use. In that case, the heater should always be left off, so sequence f would be optimal.

This leads to the use of an objective function J, with which the sequences can be evaluated based on the objectives of the user. However, to avoid having to evaluate each possible sequence to determine which one is optimal, an optimization technique can be used. The range of possible optimization methods are explored in the next section. For now, it will be assumed that after optimization, sequence b is found to be optimal. Therefore, the optimal control sequence V at $k = 2$ is:

$$\begin{aligned} V_k &= [\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_{PH}] \\ V_2 &= [\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4, \boldsymbol{v}_5, \boldsymbol{v}_6] \\ V_2 &= [1, 1, 1, 0, 0, 0] \end{aligned}$$

Recall from Eq.(3) that the first control action $\boldsymbol{v}_1$ in the optimal control sequence V_k is applied as the control action u_k, and is the only control action of the sequence that is actually implemented in the greenhouse. Therefore, in this case at interval $k = 2$ the control action u_k is:

$$u_2 = \boldsymbol{v}_1 = 1$$

At $k = 2$, the heater will thus be turned on. Then, the prediction horizon shifts one interval, and the whole process starts over again at $k = 3$.

3.5 Solving MPC

The performance of MPC is highly dependent on the process model used to simulate the system [20]. If a linear time-invariant model is used, a solution can be obtained analytically. If the optimization problem is quadratic and the non-linear optimization problem is convex, the problem can be solved using fast gradient-descent methods, guaranteeing a global solution [20]. However, in most cases both non-linear models and constraints are used, resulting in a non-convex problem [20]. In that case, the most relevant solving techniques are *Sequential Quadratic Programming* [10] and the *simplex method* [23]. However, as these methods rely on iterative optimization, in the presence of non-linear constraints they hamper the application of MPC to fast systems due to their high computational costs. This makes them

unsuitable for systems with short sampling times, as is the case in a greenhouse. Additionally, the convergence can lead to local minima, causing poor performance of the MPC algorithm [20].

These issues have led to the interest in using alternative methods of optimization for non-convex optimization problems [20]. These can be employed when the control space is discretized, so the problem is transformed into a discrete optimization problem. In that case, techniques such as dynamic programming [2], genetic algorithms [5, 24], and branch and bound (B&B) [4, 7, 15, 20, 30, 33] can be used. When the B&B method is applied, the discretized control space is structurally searched through a tree structure and bounds are applied to restrict the branching and avoid an enumerative search [7, 20]. This method has proven to give better results than iterative optimization techniques [33]. This is due to some of the algorithm's intrinsic properties, which lend it advantages over other techniques when applied to MPC. Previous studies point out the following advantages:

- The global optimum is always found, hence guaranteeing the optimality of the controller (within the discrete decision space)[33].
- The method deals with constraints implicitly, and is not negatively affected by them. Constraints may even improve the bounding efficiency by eliminating more branches [7].
- The algorithm does not require an initial guess of the optimal solution. Hence, as opposed to iterative optimization, its performance cannot be negatively affected by poor initialization [33].

In order to test these properties and to investigate the dynamics and possible improvements, the implementation of B&B to MPC will be explored further in the next chapters. The purpose of this report is to examine the possibilities of a solution method to one time interval, regardless of where that interval is in the total time line. Therefore, from now on it is assumed that the algorithm is at a certain time interval k, but the value of k will not be specified anymore, as it is not relevant to the optimization process of a sequence V.

4 Solving MPC by B&B

In this chapter, first the general concept of the Branch and Bound algorithm is described in Section 4.1. Subsequently, the optimization problem is specified in more detail in Section 4.2, and Section 4.3 is an analysis of how the general concept of B&B can be applied to solve the optimization problem. Finally, the application of the algorithm is illustrated with a small example problem in the last subsection.

4.1 The Branch and Bound Algorithm

The branch and bound method is characterized by the application of four rules: *Branching, Bounding, Elimination* and *Selection* [22]. In doing this, the method avoids visiting branches which are known not to be optimal [12]. The algorithm for a minimization problem starts with a set C_1 containing all feasible solutions. The *Branching* rule consists of splitting the set C_1 into multiple subsets. For every subset j a lower bound J_j^L of the minimum objective function value is determined. The lower bound is the best possible objective function value that this subset could potentially offer [12]. At every stage, there exists also a global upper bound J^U of the minimum objective function value over the total feasible set. The upper bound is defined by the objective value of the best (lowest) feasible solution found thus far [12].

The *Bounding* and *Elimination* rules consist of discarding all subsets for which $J_j^L > J^U$, as they can never contain a better solution than the one already found for J^U. In the case that $J_j^L < J^U$, then for some type of problems a feasible solution x_m in the subset j can be evaluated to determine J_j. If $J_j < J^U$, then J_j will replace J^U as the upper bound during further branching [12]. Finally, the *Selection* rule is to choose one of the subsets to further branch into. This influences the performance of the algorithm, as it defines how the decision tree will be searched [12]. For example, choosing the subset with the lowest lower bound for further branching yields a depth first search, while choosing the largest subset yields a breadth first search. The main target is to quickly find a sharp upper bound J^U, so that large parts of the tree can be pruned [12].

4.2 General Problem Statement

Although the application of the B&B algorithm to MPC does follow the same rules of *Branching, Bounding, Elimination* and *Selection* normally used in B&B optimization, it's search structure differs from the traditional algorithm described in the previous section. In order to solve control sequence V with branch and bound, it is necessary to define the objective function J of the control problem cumulatively over the prediction horizon [5, 7, 20]. Each control action $\boldsymbol{v}_i$ of a control sequence V contributes a certain transi-

tion cost c to its total objective cost $J(V)$:

$$J(V) = \sum_{i=1}^{PH} c(\boldsymbol{v}_i, \boldsymbol{s}_i) \qquad \text{with} \quad V = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{PH}) \tag{9}$$

The dependence of the objective value on V is henceforth considered implicit in the simplified notation

$$J = J(V)$$

Up to the i^{th} control action of V, the partial cost thus far can be calculated by adding the transition costs up to i:

$$J_i = \sum_{\ell=1}^{i} c(\boldsymbol{v}_\ell, \boldsymbol{s}_\ell) \qquad \forall\ i = 1, \ldots, PH \tag{10}$$

Therefore, the cost J_{PH} up to control action $\boldsymbol{v}_{PH}$ is equal to the total costs J of the whole control sequence:

$$\begin{aligned} J_{PH} &= \sum_{\ell=1}^{PH} c(\boldsymbol{v}_\ell, \boldsymbol{s}_\ell) \\ J_{PH} &= J \end{aligned} \tag{11}$$

The transition cost $c(\boldsymbol{v}, \boldsymbol{s})$ is a function dependent on the control action $\boldsymbol{v}$ and on the state $\boldsymbol{s}$ at that interval. It computes the cost of implementing a certain control action $\boldsymbol{v}$, as the actuator settings. The state vector $\boldsymbol{s}_i$ contains the state of the system at the beginning of interval i. It contains NS elements, consisting of two different types of state variables. The first elements are *climatic* state variables, which are states of the climate that must be controlled or followed. Examples are temperature, humidity, ground temperature, or the PAR radiation in the greenhouse. The number of climatic state variables is denoted NSC. The last elements of $\boldsymbol{s}_i$ are *actuator* state variables, as they contain the control action $\boldsymbol{v}_{i-1}$ that was implemented in the previous interval. This works as the memory of the algorithm, in order to be able to recall the previous control action. Since $\boldsymbol{v}_{i-1}$ contains NA elements, the last NA elements of $\boldsymbol{s}_i$ together comprise $\boldsymbol{v}_{i-1}$. The total number of elements NS in the state vector $\boldsymbol{s}_i$ is thus

$$NS = NSC + NA$$

The state vector $\boldsymbol{s}_i$ is hence defined as:

$$\boldsymbol{s}_i = \begin{pmatrix} s_1 \\ \vdots \\ s_{(NS-NA)} \\ s_{(NS-NA+1)} \\ \vdots \\ s_{NS} \end{pmatrix} = \begin{pmatrix} s_1 \\ \vdots \\ s_{(NSC)} \\ s_{(NSC+1)} \\ \vdots \\ s_{NS} \end{pmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \text{Climatic state variables} \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \boldsymbol{v}_{i-1} \end{matrix} \tag{12}$$

$$\forall\ i = 1, \ldots, PH$$

The transition cost function $c(\boldsymbol{v}, \boldsymbol{s})$ is a topic on which the available literature differs, so this will be expanded on in depth in Section 5.3. For now, a conventional MPC objective function will be used, which uses sum-quadratic functions to minimize the overshoot (error) and the control effort [32]. The error e must be minimized to maintain the necessary conditions to insure the quality of the products [28]. On the other hand, the control effort, denoted $\Delta\boldsymbol{v}$, is minimized, because every change in the control action requires energy [5]. The transition cost is calculated as follows:

$$c(\boldsymbol{v}, \boldsymbol{s}) = ||e||^2_{\lambda_1} + ||\Delta\boldsymbol{v}||^2_{\lambda_2} \tag{13}$$

The error e is defined as the difference between the setpoint $\boldsymbol{SP}$ and the predicted state $\boldsymbol{s}_{i+1} = T(s, v)$ at the beginning of the next interval [20], where T is the transition function of the state:

$$e = T(s, v) - \boldsymbol{SP} \tag{14}$$

The setpoint vector $\boldsymbol{SP}$ contains the desired values of the climatic state variables of the greenhouse. Its last elements have a value of zero because $\boldsymbol{SP}$ must have the same dimensions as the state vector $\boldsymbol{s}$ to compute Eq.(14). Since $s_{(NSC+1)}$ to s_{NS} are not related to the climatic state of the greenhouse, they do not have a setpoint, hence leading to zero elements in $\boldsymbol{SP}$:

$$\boldsymbol{SP} = \begin{pmatrix} SP_1 \\ \vdots \\ SP_{(NSC)} \\ SP_{(NSC+1)} \\ \vdots \\ SP_{NS} \end{pmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \text{Climatic state setpoints} \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} = 0 \end{matrix} \tag{15}$$

The control effort $\Delta\boldsymbol{v}$ at interval i is the change in control action from the previous interval $i-1$ to i, with $i = 1, \ldots, PH$. Recall from Eq.(12)

that the control action at $i-1$ are stored as the last elements $s_{(NSC+1)}$ to s_{NS} in the state vector $\boldsymbol{s}$. This functions as the "memory" of the algorithm to recall the previous control action $\boldsymbol{v}_{i-1}$. Therefore, the control effort $\Delta\boldsymbol{v}$ is defined as:

$$\Delta\boldsymbol{v}_i = \boldsymbol{v}_i - \boldsymbol{v}_{i-1} \qquad \forall\ i = 1, \ldots, PH \tag{16}$$

$$\Delta\boldsymbol{v} = \boldsymbol{v} - (s_{(NSC+1)}, \ldots, s_{NS})^T \tag{17}$$

Finally, λ_1 and λ_2 are matrices which simultaneously normalize and weigh the error and control effort [20]. For the normalization, the matrices must map the terms linearly in the range [0,1] by dividing each element by its user-estimated maximum. The weights W_1 and W_2 of the terms should be a value between 0 and 1, and together should sum to 1 [29]. The weights are used-defined, and are meant to reflect the priorities or goals of the user. The matrices look as follows:

$$\lambda_1 = W_1 \begin{pmatrix} \overline{e_1}^{-1} & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & 0 & \cdots & \vdots \\ 0 & \cdots & \overline{e}_{NSC}{}^{-1} & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 \end{pmatrix} \tag{18}$$

$$\lambda_2 = W_2 \begin{pmatrix} \overline{\Delta v_1}^{-1} & 0 & \cdots & 0 \\ 0 & \overline{\Delta v_2}^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \overline{\Delta v_{NA}}^{-1} \end{pmatrix} \tag{19}$$

where $\overline{e_1}, \ldots, \overline{e}_{NSC}$ are the maximum errors of climatic state variables, and $\overline{\Delta v}_a$ the maximum change of control variable v_a, with $a = 1, \ldots, NA$. Note that the last rows of λ_1 contain only zeros because the last elements of $\boldsymbol{s}$ comprise the control action at the previous interval i. Since these are climatic state variables, they are not relevant to the deviation term e.

The general optimization problem statement for a control sequence V can thus be formulated as follows:

$$\min_{V} \quad J(V) = \sum_{i=1}^{PH} c(\boldsymbol{v}_i, \boldsymbol{s}_i)$$

$$\begin{aligned}
\text{with} \qquad V &= (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{PH}) \\
\boldsymbol{v} &= (v_1, \ldots, v_{NA}) \\
\boldsymbol{s} &= (s_1, \ldots, s_{NS}) \\
c(\boldsymbol{v}, \boldsymbol{s}) &= ||T(\boldsymbol{v}, \boldsymbol{s}) - \boldsymbol{SP}||^2_{\lambda_1} + ||\boldsymbol{v} - s_{(NSC+1,\ldots,NS)}||^2_{\lambda_2}
\end{aligned}$$

$$\begin{aligned}
\text{s.t.} \qquad \boldsymbol{s}_{i+1} &= T(\boldsymbol{v}_i, \boldsymbol{s}_i) && \forall\, i = 1, \ldots, PH \\
\boldsymbol{v}_i &= \boldsymbol{v}_{CH} && \forall\, i = CH+1, \ldots, PH \\
v_{ia} &\in \Omega_a \qquad \forall\, i = 1, \ldots, PH \quad \forall\, a = 1, \ldots, NA
\end{aligned}$$

4.3 Solving by Branch and Bound

The optimization problem outlined in the previous section can be solved using a branch and bound search method. The search method is analyzed in the following subsections.

4.3.1 Defining the Search Tree

To implement the B&B to solve the optimization problem, first the search tree must be defined. Each time interval i, with $i = 1, \ldots, PH$, represents a level in the search tree ($i = 0$ at the initial node) [20]. At each level i within the control horizon CH, $i = 1, \ldots, CH$, the algorithm must decide the control action $\boldsymbol{v}_i$. The system has B alternatives for $\boldsymbol{v}_i$. Therefore, each node will have B branches [20]. The j^{th} branch, with $j = 1, \ldots, B$, is represented by ω_j. For example, consider a greenhouse with a heater as single actuator v_1, which can be turned off or on, so $v_1 \in \{0, 1\}$. Since the problem has only two control alternatives, at each level i of the tree each node will branch in two. So $\boldsymbol{v}_i$ can be either 0 or 1. Therefore, the algorithm can choose between ω_1 and ω_2 for $\boldsymbol{v}_i$, where control alternative $\omega_1 = (0)$ is to turn the heater off and control alternative $\omega_2 = (1)$ is to turn the heater on [20]. The transition cost c of implementing branch ω_j as control action $\boldsymbol{v}_i$ is $c(\omega_j, \boldsymbol{s}_i)$, calculated by Eq.(13). Although for this simple problem this notation might seem overly complicated or redundant, more complex problems later on will justify its necessity. Figure 4 gives a graphical representation of a search tree with two control alternatives.

Figure 4 also shows that no branching takes place beyond the control horizon ($i > CH$). Control action $\boldsymbol{v}_{CH}$ is applied successively until PH. In other words, when level CH is reached, the control action at the last interval within the control horizon is applied [20].

So far the tree has been illustrated in the case of only one actuator v_1. However, in reality a greenhouse has more than one actuator. In that case,

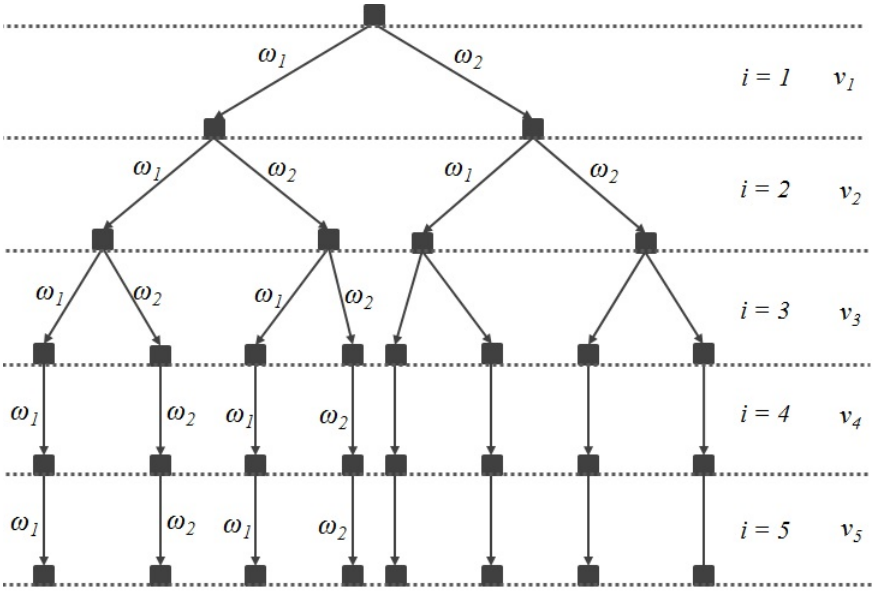


Figure 4: Example of a branch and bound tree applied to a MPC optimization problem with one actuator $v_1 \in \{0, 1\}$, prediction horizon $PH = 5$, and control horizon $CH = 3$.

the tree will look largely the same as for the case of one actuator, except that the number of control alternatives B will be much larger. Recall from Eq.(4) that the control action $\boldsymbol{v}_i$ contains NA control variables, one for each actuator. Without loss of generality, it is assumed that each actuator v_a with $a = 1, \ldots, NA$, has M discrete alternative control actions. Therefore, a discrete control action b of actuator v_a is represented by ω_{ab} [20]. The set of all possible discrete control actions for the actuator v_a is represented by Ω_a [20]:

$$\Omega_a = \{\omega_{ab} | b = 1, \ldots, M\} \tag{20}$$

The discrete set containing all the possible control actions is hence given by [20]:

$$\Omega = \Omega_1 \times \Omega_2 \times \cdots \times \Omega_{NA} \tag{21}$$

The number of the total possible discrete control alternatives B is given by [20]:

$$B = M^{NA} \tag{22}$$

Each vector of the matrix Ω can be represented by ω_j, with $j = 1, \ldots, B$ [20]:

$$\Omega = \{\omega_1, \omega_2, \cdots, \omega_B\} \tag{23}$$

Each vector ω_j is therefore a control alternative of the actuators, and corresponds to the j^{th} branch at each tree node. Hence, at each time interval,

B control alternatives can be implemented for $\boldsymbol{v}_i$, resulting in a maximum of B branches per node [20]. The general problem is depicted in Figure 5.

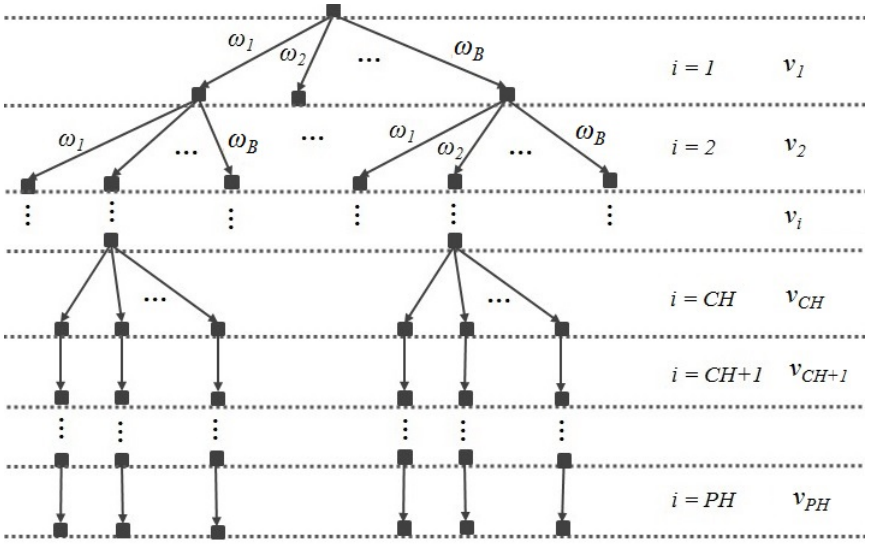


Figure 5: General case of a branch and bound tree applied to MPC, with a prediction horizon PH, control horizon CH, and B control alternatives for control actions $\boldsymbol{v}_i$ with $i = 1, \ldots, PH$. Taken and modified from [20].

As an example, consider a greenhouse with actuators v_1 and v_2, both of which have $M = 3$ control alternatives. Assume that $v_1 \in \{0, 50, 100\}$ and $v_2 \in \{0, 30, 70\}$. The maximum number of branches will be $B = M \times M = 3 \times 3 = 9$. The problem will thus have $B = 9$ branches or control alternatives per node. The matrix Ω of control alternatives is given as an illustrative example in Table 2.

Table 2: Control alternatives $\omega_1, \ldots, \omega_9$ for a problem with actuators $v_1 \in \{0, 50, 100\}$ and $v_2 \in \{0, 30, 70\}$.

	ω_1	ω_2	ω_3	$\cdots$	ω_9
v_1	0	0	50	$\cdots$	100
v_2	0	30	70	$\cdots$	70

4.3.2 Bounding the Search

The characteristic of the branch and bound algorithm that distinguishes it from an enumerative search is the *Bounding* rule. If all the possible branches of the tree were searched, it would result in B^{CH} different control sequences, which even for a small number of actuators and discretizations

can be too large [20]. Hence, the bounding is essential for the applicability of the algorithm.

To avoid extensive data storage, the algorithm performs a depth first search. Therefore, a branch ω_j at level i is evaluated to determine if it can be followed to the next level $i+1$ of the tree. The *Bounding* rule states that a particular branch is only followed if the previous cumulative cost J_{i-1} plus a lower bound on the cost from the level i to PH, denoted $J_L^{(i)}$, is lower than an upper bound of the total cost, denoted J^U [20]. In other words, at each branch, a lower bound $J_L^{(i)}$ must be computed of the estimated costs of the current and the remaining intervals, see Figure 6. The lower bound is assumed to consist of the transition cost $c(\omega_j, \boldsymbol{s}_i)$ at interval i plus the cost of the remaining intervals from $i+1$ to PH, see Figure 6. However, the cost of the remaining intervals is very difficult to estimate. As the lower bound may only be an under-estimate, the remaining cost is set to 0, to avoid bounding branches which could contain a better solution [7, 20]. Therefore, the bounding condition is the following [20]:

$$\begin{aligned} J_{i-1} + J_L^{(i)} &< J^U \\ J_{i-1} + c(\omega_j, \boldsymbol{s}_i) + 0 &< J^U \\ J_i &< J^U \end{aligned} \tag{24}$$

When the terminal level of the tree $i = PH$ is reached, the cumulative cost J_i is the terminal cost J_{PH}. At the bottom of the tree, the path followed down the tree is a complete control sequence V. The objective value $J(V)$ of the sequence is $J(V) = J_{PH} = J$ (see Eq.(11)). If this terminal cost J of the sequence is lower than the current upper bound J^U, then J^U is replaced by J and the control sequence V is saved as the best so far. After the whole tree has been evaluated, the control sequence stored at that moment as best so far can be concluded to be the overall optimal sequence.

The initial value of J^U can be set arbitrarily. In theory, it could be set to infinity, as it will be replaced by a better value by the first branch reaching the terminal level of the tree. However, to decrease the number of computations, the upper bound should always be as low as possible. Therefore, finding an initial upper bound close to the optimum can be convenient. In most of the literature, a greedy algorithm is used for this [7, 20]. The first path followed through the search tree chooses the smallest transition cost $c(\omega_j, \boldsymbol{s}_i)$ at each level i. This strategy yields a terminal cost J_{PH} close to the optimum and can hence reduce the necessary iterations to find the optimal solution. The logical steps that the algorithm takes to evaluate the search tree are shown in Algorithms 1 and 2.

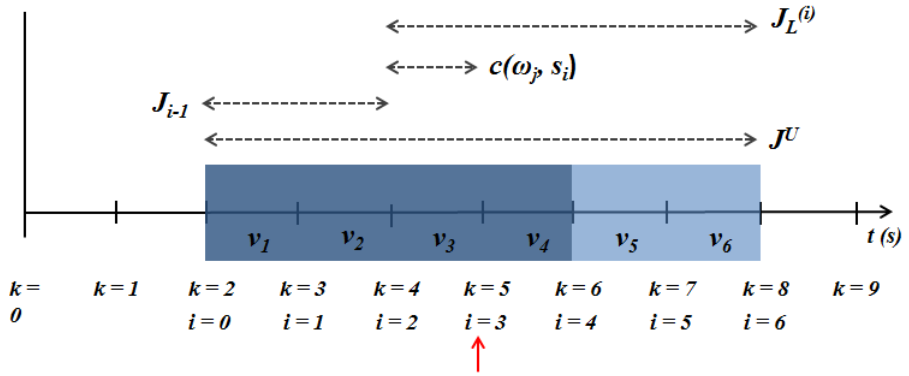


Figure 6: Graphical representation of upper bound J^U, cumulative cost J_{i-1} and transition cost $c(\omega_j, \boldsymbol{s}_i)$ for a problem at $i = 3$.

Algorithm 1 ControlSequence$(PH, CH, B, \boldsymbol{s}_1)$

Require: PH, CH, B, and $\boldsymbol{s}_1$
  Compute alternative settings $\Omega := \{\omega_1, \ldots, \omega_B\}$
  $\{V, J^U\} := UpperBound(PH, CH, B, \boldsymbol{s}_1, \Omega)$
  Let $\Omega_a := \Omega$ for $a = 1, \ldots, CH$
  $i := 1$ and $J_0 := 0$
  while ($\Omega_a \neq \varnothing$ for $a = 1, \ldots, PH$)
    remove ω from Ω_i
    calculate transition cost $c(\omega, \boldsymbol{s}_i)$ and new state $\boldsymbol{s}_{i+1}$
    calculate total cost up to i: $J_i := J_{i-1} + c(\omega, \boldsymbol{s}_i)$
    if ($J_i < J^U$)
      $\boldsymbol{v}_i := \omega$
      if ($i < PH$), $i := i + 1$
      else
        update upper bound $J^U := J_i$ and $V := (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{PH})$
        $i := i - 1$
    while ($\Omega_i = \varnothing$)
      $\Omega_i := \Omega$
      $i := i - 1$
    endwhile
    if ($i > CH$), $\Omega_a = \{\boldsymbol{v}_{CH}\}$ for $a = i, \ldots, PH$
  endwhile
  return: V, J^U

Algorithm 2 UpperBound($PH, CH, B, \boldsymbol{s}_1, \Omega$)

Require: PH, CH, $B, \boldsymbol{s}_1$, and Ω
  $i := 1$
  for $i = 1 : PH$
    if $i \leqslant CH$
      $c(\omega, \boldsymbol{s}_i) = \min\{c(\omega_1, \boldsymbol{s}_i), \ldots, c(\omega_B, \boldsymbol{s}_i)\}$ with $\Omega = \{\omega_1, \ldots, \omega_B\}$
    else, $c(\omega, \boldsymbol{s}_i) = c(\boldsymbol{v}_{CH}, \boldsymbol{s}_i)$
    $\boldsymbol{v}_i = \omega$
  endfor
  $V^U = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{PH})$
  $J^U = \sum_{i=1}^{PH} c(\boldsymbol{v}_i, \boldsymbol{s}_i)$
  return: V, J^U

4.4 Example Problem

In order to illustrate the application of the B&B algorithm, a hypothetical simplified example is used. Consider the previously mentioned greenhouse with as only actuator v_1 a heater. Since the problem involves only one actuator, the control vectors $\boldsymbol{v}_i$, with $i = 1, \ldots, PH$ are vectors of only one element. To further simplify the system, the only climatic state variable s_1 to consider is the temperature inside the greenhouse. For the initial state $\boldsymbol{s}_1$ of the problem, it is assumed that the initial temperature of the greenhouse is $24°C$, and that the heater is off. The setpoint temperature the problem is $SP_1 = 27°C$. The heater can be turned off or on, so $v_1 \in \{0, 1\}$, causing a $0°C$ or a $1°C$ change on the air temperature respectively. The effect of the heater on the temperature is described in transition function T. Finally, the prediction and control horizon were defined to be $PH = 4$ and $CH = 2$ respectively. Therefore, the problem has the following conditions:

$$\min_{V} \quad J(V) = \sum_{i=1}^{PH} c(\boldsymbol{v}_i, \boldsymbol{s}_i)$$

$$\begin{aligned}
\text{with} \qquad V &= (\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4) \\
\boldsymbol{v} &= (v_1) \\
\boldsymbol{s} &= (s_1, s_2)^T \\
c(\boldsymbol{v}, \boldsymbol{s}) &= ||T(\boldsymbol{v}, \boldsymbol{s}) - \boldsymbol{SP}||^2_{\lambda_1} + ||\boldsymbol{v} - s_2||^2_{\lambda_2} \\
T(\boldsymbol{v}, \boldsymbol{s}) &= (s_1 + v_1, \boldsymbol{v})^T \\
\text{s.t.} \qquad \boldsymbol{s}_{i+1} &= T(\boldsymbol{v}_i, \boldsymbol{s}_i) && \forall\, i = 1, \ldots, 4 \\
\boldsymbol{v}_i &= \boldsymbol{v}_2 && \forall\, i = 3, 4 \\
v_{i1} &\in \{0, 1\} && \forall\, i = 1, \ldots, 4
\end{aligned}$$

To define the normalization and weighing matrices λ_1 and λ_2, the maximum temperature deviation $\overline{e_1}$ was set to $10°C$ and the maximum control effort $\overline{\Delta v_1}$ is 1. For the sake of simplicity, both were assigned equal importance, so each objective received a weight of $W_1 = W_2 = 0.5$. Applying Eq.(31), the matrices look as follows:

$$\lambda_1 = 0.5 \begin{pmatrix} 0.1 & 0 \\ 0 & 0 \end{pmatrix} \qquad \lambda_2 = 0.5 \begin{pmatrix} 1 \end{pmatrix}$$

Hence, the transition cost function c of the problem will look as follows:

$$c(\boldsymbol{v}, \boldsymbol{s}) = ||e||^2_{\lambda_1} + ||\Delta \boldsymbol{v}||^2_{\lambda_2}$$

$$c(\boldsymbol{v}, \boldsymbol{s}) = ||T(\boldsymbol{s}, \boldsymbol{v}) - \boldsymbol{SP}||^2_{\lambda_1} + ||v - s_2||^2_{\lambda_2}$$

$$c(\boldsymbol{v}_i, \boldsymbol{s}_i) = \left|\left| \begin{pmatrix} s_1 + v_1 \\ \boldsymbol{v}_i \end{pmatrix} - \begin{pmatrix} 27 \\ 0 \end{pmatrix} \right|\right|^2_{\lambda_1} + ||(\boldsymbol{v}_i) - (\boldsymbol{v}_{i-1})||^2_{\lambda_2}$$

$$c(\boldsymbol{v}_i, \boldsymbol{s}_i) = \begin{pmatrix} s_1 + v_1 - 27 \\ \boldsymbol{v}_i \end{pmatrix}^T \begin{pmatrix} 0.05 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} s_1 + v_1 - 27 \\ \boldsymbol{v}_i \end{pmatrix} + \\ + 0.5(\boldsymbol{v}_i - \boldsymbol{v}_{i-1})^T(\boldsymbol{v}_i - \boldsymbol{v}_{i-1})$$

$$\forall \ i = 1, \ldots, 4$$

To start solving the problem, an initial upper bound J^U is determined with Algorithm 2. A graphical representation of the tree is shown in Figure 7. The problem starts at the initial node, where $i = 0$. As the problem has only two control alternatives, each node branches in two ($B = 2$). The set

$$\Omega = \{\omega_1, \omega_2\}$$

contains the two alternative control actions at each node, where $\omega_1 = (0)$ (heater off) and $\omega_2 = (1)$ (heater on). The j^{th} branch of each node at level i is thus the cost $c(\omega_j, \boldsymbol{s}_i)$ of implementing control vector ω_j as $\boldsymbol{v}_i$. The first branching occurs at the level $i = 1$. First the left branch ω_1 is examined.

The calculation of the upper bound J^U is only concerned with the transition cost c at each node, and not with the cumulative cost J_i. At branch ω_1, the heater is left off so the temperature at the next interval will remain constant and the control effort is $\Delta \boldsymbol{v} = 0$. The transition cost is hence

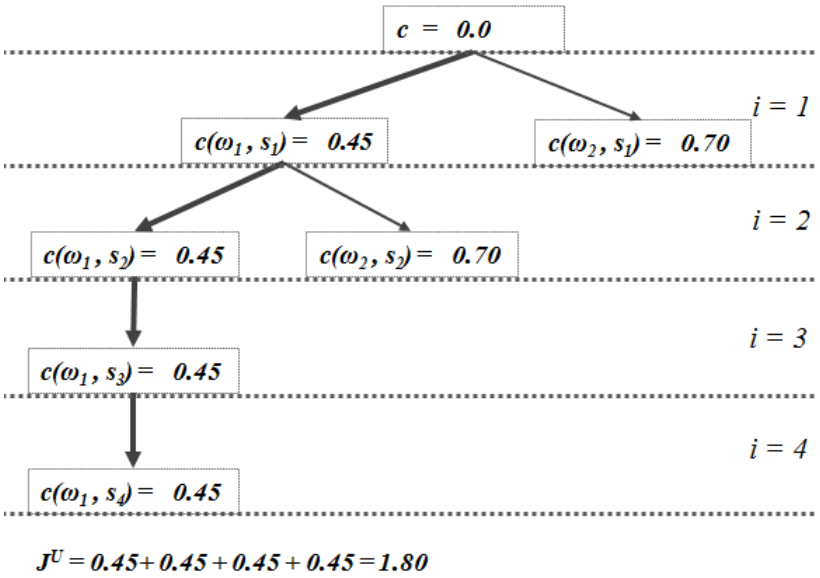


Figure 7: Example tree of how the initial upper bound J^U is calculated by Algorithm 2.

calculated:

$$c(\omega_1, \boldsymbol{s}_1) = \left|\left| \begin{pmatrix} 24 \\ 0 \end{pmatrix} - \begin{pmatrix} 27 \\ 0 \end{pmatrix} \right|\right|^2_{\lambda_1} + ||(0) - (0)||^2_{\lambda_2}$$

$$c(\omega_1, \boldsymbol{s}_1) = \begin{pmatrix} -3 \\ 0 \end{pmatrix}^T \begin{pmatrix} 0.05 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} -3 \\ 0 \end{pmatrix} + 0$$

$$c(\omega_1, \boldsymbol{s}_1) = 0.45$$

This value of $c(\omega_1, \boldsymbol{s}_1)$ is stored and then the algorithm moves on to the next branch, ω_2. At this control alternative the heater is turned on, so the temperature increases by 1°C, and the control effort is $\Delta \boldsymbol{v} = 1$. The transition cost is calculated and found to be $c(\omega_2, \boldsymbol{s}_1) = 0.70$. The algorithm chooses the lowest of the two transition costs to continue branching, which in this case is $c(\omega_1, \boldsymbol{s}_1)$. This process is repeated until the control horizon $i = CH = 2$ is reached. After the control horizon, the transition cost is only calculated for the branch ω_j chosen at $i = CH$. In this case, at $i = CH = 2$ branch ω_1 had the lowest transition cost, so c is only calculated for ω_1 at $i = 3$ and $i = 4$. When the prediction horizon $i = PH$ is reached, the transition costs of the "greedy" path are added up to find the cumulative cost of this control sequence. The result is the control sequence V and its objective value J^U that the search will initially use as its upper bound.

To solve the optimal control sequence V, the decision tree is considered from the top level again. Figure 8 shows the exploration of the tree. At

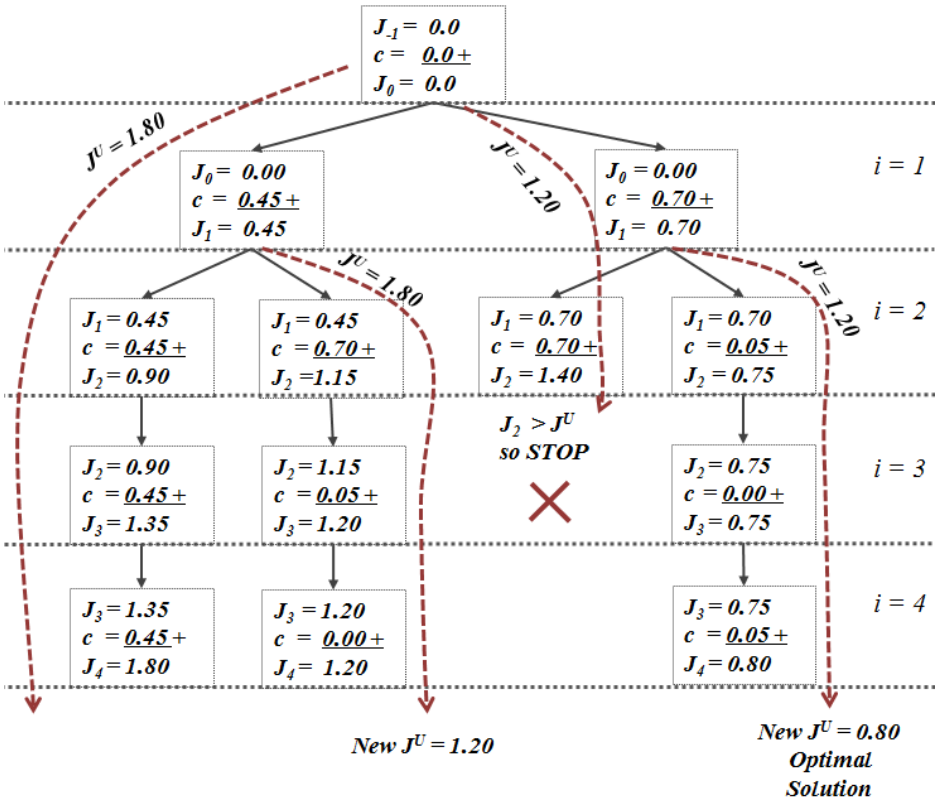


Figure 8: Example of Branch and Bound tree for a MPC problem with $PH = 4$ and $CH = 2$. The initial upper bound J^U was calculated beforehand with a greedy algorithm (see Figure 7).

$i = 1$, first branch ω_1 is considered. The transition cost is computed at each interval and added to the previous cumulative cost J_{i-1}. If the sum is lower than the upper bound J^U, then the algorithm stores control action ω_1 at level $i = 1$ as $\boldsymbol{v}_1$ and moves to $i = 2$ (see Eq.(24)). In Figure 8 we can see that this is the case. The algorithm goes down as far as it can in a certain branch. When it reaches PH or is bounded by J^U it moves up again to the last level i at which the J^U was not exceeded, and goes to the next branch. After searching the whole tree, the algorithm has found that the last path is the optimal solution. Notice that the order in which the branches are evaluated influences the number of nodes that must be evaluated before the optimal solution is found. In Figure 8 the search is performed from left to right. However, if the search had been performed from right to left, the lowest J would have been found much sooner, hence bounding the left side of the tree and resulting in a much shorter search.

The solution of the control problem is shown in Table 3. The temperature

at each interval i is the temperature at the beginning of the time interval, caused by the implementation of the control action at the previous interval. Furthermore, the table shows that the temperature exceeds the setpoint of $SP_1 = 27\ ^\circ\text{C}$ in the last interval. This happens because at $i = 4$ the system has passed the control horizon CH and thus cannot change its control action anymore, but keeps it constant at $\boldsymbol{v}_{CH} = \boldsymbol{v}_2 = 1$. The optimal control sequence V is:

$$V = (\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4)$$
$$V = (1, 1, 1, 1)$$

Table 3: Optimal solution and temperature state of the example problem with $PH = 4$, $CH = 2$ and temperature setpoint of 27 °C.

$\boldsymbol{v}_i$	i	ω_j	v_1	*Temp.* (°C)
$\boldsymbol{v}_1$	1	ω_2	1	24
$\boldsymbol{v}_2$	2	ω_2	1	25
$\boldsymbol{v}_3$	3	ω_2	1	26
$\boldsymbol{v}_4$	4	ω_2	1	27
-	-	-	-	28

5 Extension of the B&B Implementation

In order to further extend the application of the B&B algorithm, a model was built in Matlab R2015b that uses the B&B algorithm discussed before to solve a more realistic optimization problem. The model was built to solve a control sequence V at a certain moment in time. The next sections provide an overview of the extensions that were implemented in the model.

5.1 Climate Model

In previous examples of the B&B application, a very simple transition function was used to quantify the effect of the control actions on the state of the climate. Yet in reality, the greenhouse climate is a much more complicated system, so a dynamic model is necessary to simulate the climate dynamics. Models for this purpose can be either black box models or first principles models [16]. In much of the available literature, black box models are used by applying artificial neural networks [8, 11, 21]. On the other hand, a first principles model provides a better understanding of the processes involved in the dynamics of a system [5]. This allows a better traceability of causal relationships, and can hence be useful to identify and untangle possible problem sources. Therefore, a first principles climate model from literature was implemented to simulate climate dynamics.

The model uses three actuators, which are the window opening, the fog system and the heating. The model hence has $NA = 3$ number of actuators. All three control variables are given in percentages, so their range is [0,100]. Control action $\boldsymbol{v}$ of the model is given by:

$$\boldsymbol{v} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} \qquad \begin{aligned} &\rightarrow \text{Window Opening} \\ &\rightarrow \text{Fog System} \\ &\rightarrow \text{Heating} \end{aligned} \tag{25}$$

The climatic state variables are the air temperature, air humidity, and ground temperature. The number of climatic state variables is thus $NSC = 3$. Of these three, only the air temperature and air humidity are climatic state *control* variables, as these are the states that the user seeks to control. The ground temperature is only included in the state vector because of its influence on the air temperature and humidity. The last three elements in are the actuator state variables. The model thus has $NS = NSC + NA = 3 + 3 = 6$ state variables in total. The state vector $\boldsymbol{s}$ of the model is defined as:

$$\boldsymbol{s}_i = \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \end{pmatrix} \begin{matrix} \\ \rightarrow \text{Air Temperature} \\ \rightarrow \text{Air Relative Humidity} \\ \rightarrow \text{Ground Temperature} \\ \Big\} \boldsymbol{v}_{i-1} \end{matrix} \tag{26}$$

The climate model was taken from articles [5] and [13]. In state space form, the model is defined as a mass and energy balance [5]. The state space equations of the climate model are defined in Eq.(28) to (29). The symbolic representation of the equations was adapted to show clearly which are the dependent and independent variables. All capital letters are dependent variables, all α symbols represent constant parameters, and non-capital letters are the independent variables. Since all the parameter values of the equations are fixed, the model can be said to be deterministic. The actuator and state variables, as well as the disturbances which the model takes into account are given in Table 4.

$$\begin{aligned} \frac{\mathrm{d}s_1}{\mathrm{d}t} = \Big(& Q_2(d_3) - Q_3(s_1, d_1) + Q_7(s_1, s_3) \\ & - F_7(s_1, s_2)(Q_4(s_1, s_2) + Q_6(v_2, s_1)) \\ & - Q_5(v_1, s_1, d_1) + Q_1(v_3) \Big) / (\alpha_{18}\alpha_{24}\alpha_5) \end{aligned} \tag{27}$$

$$\frac{\mathrm{d}s_2}{\mathrm{d}t} = f_3(s_1, \frac{G_2(v_1, s_1, s_2, d_2) + F_7(s_1, s_2)(G_1(v_2) + G_3(s_1, s_2))}{\alpha_{24}\alpha_{18}}) \tag{28}$$

$$\frac{\mathrm{d}s_3}{\mathrm{d}t} = \frac{Q_8(d_3) - Q_7(s_1, s_3) - Q_9(s_3)}{\alpha_2\alpha_4} \tag{29}$$

The data of the model has a sample time of 15 seconds [5]. Consequently, for the optimization problem, time was split into time intervals of 15 seconds. Within that time interval, it is assumed that the change in state is linear. Therefore, the changes in state over one interval i are:

$$\Delta s_1 = 15\frac{\mathrm{d}s_1}{\mathrm{d}t} \qquad \Delta s_2 = 15\frac{\mathrm{d}s_2}{\mathrm{d}t} \qquad \Delta s_3 = 15\frac{\mathrm{d}s_3}{\mathrm{d}t}$$

The transition function T of the problem is therefore defined as:

$$T(\boldsymbol{v}, \boldsymbol{s}) = \begin{pmatrix} s_1 + \Delta s_1 \\ s_2 + \Delta s_2 \\ s_3 + \Delta s_3 \\ v_1 \\ v_2 \\ v_3 \end{pmatrix} \tag{30}$$

Table 4: Climatic state variables, disturbances and control variables (actuators) taken into account in the climate model of [5] and [13]

Climatic State Variables		
s_1	°C	Inside temperature
s_2	%	Inside relative humidity
s_3	°C	Ground temperature

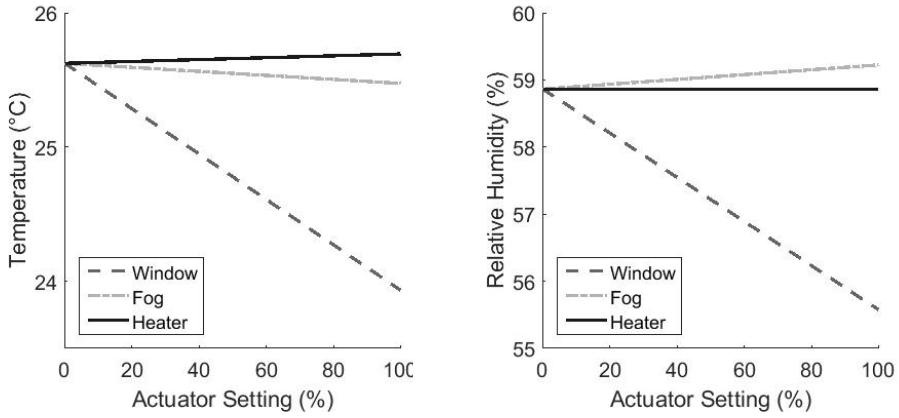
Disturbances		
d_1	°C	Outside temperature
d_2	%	Outside relative humidity
d_3	Wm^{-2}	Outside Solar radiation
d_4	ms^{-1}	Outside Wind speed

Control Variables		
v_1	%	Window opening
v_2	%	Fog system
v_3	%	Heating

The climate model was developed with data from a rose hydroponic crop in a plastic greenhouse with arch shaped roofs located in Moncada (Valencia, Spain) in the summer [5]. Thus, the model parameters are specific for these greenhouse conditions. A detailed account of the used parameters as well as the complementary equations can be found in Appendix A. For information on how the parameters were obtained, article [5] can be consulted. The graphs in Figure 9 illustrate the effect of each actuator on the climate state. These are meant as a visual aid to grasp the effect of the transition function. Additionally, the graphs were computed to check the quality of the climate model, as the results they show should not be counter-intuitive (for example, declining temperature with increasing heating).

5.2 Discretization of the Control Space

As discussed in Section 3.5, the application of branch and bound to a control sequence optimization problem requires the discretization of the control space. Hence, an actuator v_1 such as a heater for example, which has a con-



(a) Temperature (b) Humidity

Figure 9: Temperature and humidity after implementation of the range of control actions during one time interval, assuming parameters and outside weather conditions given in Table 11 in the Appendix

tinuous range between 0 and 100%, could be discretized to $v_1 \in \{0, 50, 100\}$, implying that the heater can only be turned off, half power, or full power [5]. The drawback of the discretization is that it can cause the output to oscillate around a reference trajectory, instead of following it exactly [33]. However, literature shows that crops are more responsive to the average daily temperature than to accurate temperature evolution during the day [14, 35]. This led to the concept of Temperature Integration (TI) [17], where the greenhouse temperature is allowed to fluctuate within certain boundaries, as long as the average over a certain period is maintained. Hence, the application of TI to a greenhouse allows temperature oscillations caused by the control space discretization.

Recall from the previous section that the actuators that were implemented in the programmed model are the window opening v_1, the fog system v_2, and the heating v_2. For the discretization, each actuator's physical limitations of saturation (maximum) and output resolution (minimum step size) were also considered. For example, a window can only be opened between 0 and 100% (saturation) and the teeth of the window rack allow a minimal movement of 5% (output resolution)[28]. Therefore, all the elements in the control space of the window opening should be within 0 and 100% and must be represented by multiples of five.

Another restriction on the discretization was the computation time, since a slight increase in the number of alternative control actions causes a drastic increase in search branches and thus also in computational time. Therefore, each actuator was restricted to a maximum of four discrete control settings. The settings were spread evenly across the total range of 0 to 100%. Finally,

as Section 5.1 indicates, the climate model has parameters for the summer period, so it is assumed the heater is turned off. Actuator v_3 was thus restricted to 0. The actuators of the model were discretized as follows:

$$v_1 \in \{0, 33, 66, 100\}$$
$$v_2 \in \{0, 33, 66, 100\}$$
$$v_3 \in \{0\}$$

5.3 Objective Function

In order for the optimization problem to be relevant, its objective function J should accurately reflect the goals of the user. Since the literature on this topic differs, the issue was further investigated by implementing three different objective functions chosen from relevant articles. All three functions work according to the same cumulative principle outlined in Eq.(9), in Section 4.2, where the transition cost c at each interval i, with $i = \ldots, PH$, is added to form the total cost J of a control sequence V. Moreover, all three functions compute the transition costs c as a summation of normalized and weighed terms, each term representing an objective of the user. However, the content of the terms differs per objective function, depending on the overall approach of the function. The different objective functions are described in the next sub-sections.

5.3.1 Objective Function A: Sum of Squares

The first objective function is the same that was used for the short example in the previous section. It is considered the classic MPC objective function and was taken from article [20]. The objective function seeks to minimize the deviation, or error, e from the setpoint $\boldsymbol{SP}$ and the control effort $\Delta \boldsymbol{v}$. This function was described in Section 4.2, so this chapter can be referenced for details. As a brief refresher, the equation of the transition cost is described below:

$$c(\boldsymbol{v}, \boldsymbol{s}) = ||e||^2_{\lambda_1} + ||\Delta \boldsymbol{v}||^2_{\lambda_2}$$

with

$$e = T(\boldsymbol{v}, \boldsymbol{s}) - \boldsymbol{SP} \qquad \text{and} \qquad \Delta \boldsymbol{v} = \boldsymbol{v} - (s_4, s_5, s_6)^T$$

The normalization and weight matrices λ_1 and λ_2 are defined according to the following equation, with $\overline{e_1}$ and $\overline{e_2}$ the maximum temperature and humidity deviation from the setpoint, and $\overline{\Delta v_1}$, $\overline{\Delta v_2}$ and $\overline{\Delta v_3}$ the maximum control efforts of the window, fog system and heater respectively. It is

assumed that the minimum change of the control action is always 0.

$$\lambda_1 = W_1 \begin{pmatrix} \overline{e_1}^{-1} & 0 & 0 & \cdots & 0 \\ 0 & \overline{e_2}^{-1} & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix} \lambda_2 = W_2 \begin{pmatrix} \overline{\Delta v_1}^{-1} & 0 & 0 \\ 0 & \overline{\Delta v_2}^{-1} & 0 \\ 0 & 0 & \overline{\Delta v_3}^{-1} \end{pmatrix} \tag{31}$$

Notice that the last rows of λ_1 contain only zero elements. The third row is empty because the third state variable s_3 is the ground temperature, which is not a state *control* variable. It is therefore not relevant to the deviation from the setpoint, since the user does not aim to control it. All other rows are empty because the state variable elements s_4 to s_6 are the previous control action $\boldsymbol{v}_{i-1}$, which is also irrelevant to the deviation. Recall that all three actuators have the same range of 0 to 100% (see Section 5.2). Therefore,

$$\overline{\Delta v_1} = \overline{\Delta v_2} = \overline{\Delta v_3} = 100 \tag{32}$$

Consequently, λ_2 can be simplified to:

$$\lambda_2 = \frac{W_2}{100}$$

An important characteristic of this objective function is that its deviation term e includes both the temperature *and* the humidity. The main advantage of this feature is that it allows the user direct control over both the temperature and the humidity inside the greenhouse, whereas normally only the temperature is controlled. However, a disadvantage of the function is that it is not concerned with objectives such as water or energy use minimization, even though these are also important to achieve a more sustainable production.

5.3.2 Objective Function B: Sum of Costs

The second objective function was taken and adapted from [7]. Similarly to function A, it strives to minimize the deviation from the setpoint $\boldsymbol{SP}$ as well as the control effort $\Delta \boldsymbol{v}$. The function also minimizes the energy and water use. Furthermore, besides having a setpoint, the function also has a range within which the states are allowed to fluctuate. Any solution going outside the range receives a penalty. The transition cost c for each level of

the search tree is calculated as follows:

$$
\begin{aligned}
c(\boldsymbol{v}, \boldsymbol{s}) = \quad & \lambda_1 e + && \Big\} \text{Error} \\
& + \lambda_2 \sum_{a=1}^{NA} \Delta v_a + && \Big\} \text{Control Effort} \\
& + \lambda_3 \sum_{a=1}^{NA} v_a w_a + && \Big\} \text{Water and Energy Use} \\
& + \lambda_4 P(\boldsymbol{v}, \boldsymbol{s}) && \Big\} \text{Penalty}
\end{aligned}
\tag{33}
$$

The first term of the equation is the cost of the deviation of the states with respect to the setpoint. However, this objective function considers only the temperature as a climate state *control* variable. The states of the humidity and ground temperature have no setpoint, and hence do not contribute to the objective value. As a consequence, their corresponding rows in the normalization and weight matrix λ_1 only contain zero valued elements. The matrix is defined as:

$$
\lambda_1 = W_1 \begin{pmatrix} \overline{e_1}^{-1} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}
$$

The second term of the objective function is the costs of the control effort, similarly to function A. However, in this case the changes in actuator settings are all added and then normalized and weighed by λ_2, which is a single value:

$$
\lambda_2 = W_2 \left(\overline{\Delta v}_1 + \overline{\Delta v}_2 + \overline{\Delta v}_3 \right)^{-1}
$$

Making use of Eq.(32), λ_2 is simplified to:

$$
\lambda_2 = \frac{W_2}{300} \tag{34}
$$

The third term is meant to minimize water and energy consumption by attributing a cost to the use of each actuator. The importance attributed by the user to each actuator is indicated by the weights in vector $\boldsymbol{w}$. For example, as opening a window requires much less energy and water than the heater and fog system, it is expected that the lowest weight is attributed to this actuator. All the elements in $\boldsymbol{w}$ should be in the range [0,1], and together should sum to 1. Weight vector $\boldsymbol{w}$ is defined as:

$$
\boldsymbol{w} = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} \tag{35}
$$

where w_1, w_2 and w_3 are the weights of the window, fog system and heating respectively. Since all three actuators have a maximum value of 100, and the weights must sum to one, the weighted sum can never be greater than 100. Therefore, the weight and normalization value λ_3 is:

$$\lambda_3 = \frac{W_3}{100} \tag{36}$$

Finally, the last term is a penalty for the violation of the temperature and humidity ranges in which the states are allowed to fluctuate (see Section 5.2). The penalty P is given by:

$$P(\boldsymbol{v}, \boldsymbol{s}) = \max\{0,\ \underline{R} - T(\boldsymbol{v}, \boldsymbol{s}),\ T(\boldsymbol{v}, \boldsymbol{s}) - \overline{R}\} \tag{37}$$

The vectors $\underline{R}$ and $\overline{R}$ are the lower and upper bound respectively of the allowed range of the temperature and the humidity. The vectors are defined as follows:

$$\underline{R} = \begin{pmatrix} \underline{T} \\ \underline{H} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \qquad \overline{R} = \begin{pmatrix} \overline{T} \\ \overline{H} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{38}$$

Recall that elements s_3 to s_6 of the state vector $\boldsymbol{s}$ are the ground temperature and the previous control action. As both are not relevant to the penalty term, the corresponding elements of the vectors $\underline{R}$ and $\overline{R}$ are irrelevant, and are thus set to zero. Similarly, the same rows of the matrix λ_4 consist only of zero-elements as well. The matrix is defined as:

$$\lambda_4 = W_4 \begin{pmatrix} \overline{P_1}^{-1} & 0 & 0 & \cdots & 0 \\ 0 & \overline{P_2}^{-1} & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix} \tag{39}$$

where $\overline{P_1}$ and $\overline{P_2}$ are the maximum temperature and humidity violations of the range, and are defined by the user.

5.3.3 Objective Function C: Water and Energy Reduction

The third objective function's main goal is to reduce the water and energy consumption as much as possible, given that the temperature and humidity stay within a certain range. The function was taken and adapted from article [5]. The transition cost is calculated as follows:

$$
\begin{aligned}
c(\boldsymbol{v}, \boldsymbol{s}) = \quad & \lambda_1 \Delta v_1 + && \Big\} \text{Control Effort for Windows} \\
& + \lambda_2 v_2 + && \Big\} \text{Water Use (fog)} \\
& + \lambda_3 v_3 + && \Big\} \text{Energy Use (heating)} \\
& + \lambda_4 P(\boldsymbol{v}, \boldsymbol{s}) + && \Big\} \text{Penalty} \\
& + \text{offset}(\boldsymbol{v}, \boldsymbol{s})
\end{aligned}
\tag{40}
$$

The first term of the equation evaluates the control effort. However, only the control effort of the windows is considered, as this is the only actuator which requires a significant energy consumption to physically change its settings [5]. The second term evaluates the water use by the fog system, and the third one considers the energy consumption by the heating system. As all three terms consider only one actuator, their weight and normalization matrices λ_1, λ_2, and λ_3 are single values:

$$
\lambda_1 = \frac{W_1}{100} \qquad \lambda_2 = \frac{W_2}{100} \qquad \lambda_3 = \frac{W_3}{100} \tag{41}
$$

The fourth term is a penalty for the deviation from the established temperature and humidity range. The penalty P is calculated using Eq.(37), defined in the previous objective function. However, objective function B allows the user to determine the normalization of the penalties, whereas [5] defines the normalization and weight with λ_4:

$$
\lambda_4 = \begin{pmatrix}
15^{-1} & 0 & 0 & \cdots & 0 \\
0 & 20^{-1} & 0 & \cdots & 0 \\
0 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 0
\end{pmatrix} \tag{42}
$$

Note that a higher importance is given to the temperature (1/15) than to the humidity (1/20). Finally, the last term is an offset. This term is added to the equation to ensure that every solution going outside the allowed range has a greater objective value than those solutions inside the range. In other words, by using the offset, solutions with a penalty will always be less favorable than solutions without a penalty. The term is calculated as follows:

$$
\text{offset}(\boldsymbol{v}, \boldsymbol{s}) = \begin{cases}
3, & P(\boldsymbol{v}, \boldsymbol{s}) \neq 0 \\
0, & P(\boldsymbol{v}, \boldsymbol{s}) = 0
\end{cases} \tag{43}
$$

, where penatly P is given by (37). The offset has a value of 3 because the weights W_1, W_2 and W_3 are all in the range [0,1], so the sum of the first three terms can never be greater than 3. Therefore, the offset ensures that solutions with any violation of the ranges always have a higher value than those solutions with no violation.

The main advantage of this objective function is that its offset term prevents it from exploring solutions that are not promising, thus cutting off more branches of the search tree from the beginning. This should drastically reduce the computational load of the algorithm. However, as it does not have a term to follow a setpoint, it can prove to be less accurate than the other two objective functions.

6 Implementation of the Extensions

6.1 General Problem Statement

The extensions of Section 5 were incorporated in the optimization problem modeled in Matlab. This led to an optimization problem with a more realistic state transition function, a clearly defined discrete control space, and three alternative objective functions. The general problem formulation is given below.

$$\min_{V} \quad J(V) = \sum_{i=1}^{PH} c(\boldsymbol{v}_i, \boldsymbol{s}_i)$$

$$\begin{aligned}
\text{with} \qquad V &= (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{PH}) \\
\boldsymbol{v} &= (v_1, v_2, v_3) \\
\boldsymbol{s} &= (s_1, \ldots, s_6) \\
c(\boldsymbol{v}, \boldsymbol{s}) &= \text{Eq.(13), (33) or (40)}
\end{aligned}$$

$$\begin{aligned}
\text{s.t.} \qquad \boldsymbol{s}_{i+1} &= T(\boldsymbol{v}_i, \boldsymbol{s_i}) && \forall\, i = 1, \ldots, PH \\
\boldsymbol{v}_i &= \boldsymbol{v}_{CH} && \forall\, i = CH+1, \ldots, PH \\
v_{i1}, v_{i2} &\in \{0, 33, 66, 100\},\ v_{i3} \in \{0\} && \forall\, i = 1, \ldots, PH
\end{aligned}$$

6.2 Example Problem

The model built in Matlab R2015b solves the optimization problem outlined above by applying the branch and bound search structure from Section 4. The three alternative objective functions were built as separate functions, so that the model could optimize either of the three, depending on which one is called on. Since the transition function of the problem is deterministic, the search tree of a given set of weights and parameters will always result in the same solution. Indeed, because of the absence of stochasticity in the problem, it can even be said that the search tree is known beforehand. However, applying the branch and bound method avoids the need for a full enumeration of the whole tree.

The model was used to solve an example problem with a prediction and control horizon of $PH = 9$ and $CH = 5$ respectively. The parameters and weights of the objective functions established for the problem are given in Tables 5, 6, 7, and 8. The parameters in Table 5 regarding the state of the climate inside the greenhouse were partially taken from literature and partially determined from personal judgement. The outside climatic conditions given in Table 6 were taken from climatic data recorded in Almería. Notice from the tables that for this particular problem, the initial temperature is outside of the permitted range, while the humidity is inside. Additionally, the outside temperature is lower than the desired setpoint, while the outside humidity is higher.

The weights in Table 8 were determined by personal judgement. Both objective functions A and B received the same weight of 0.7 for the deviation term (W_1), and the remaining 0.3 was distributed over the other terms of the function. Both objective functions A and B received the same weight of 0.7 for the deviation term (W_1), and the remaining 0.3 was distributed over the other terms of the function. Function C does not need a weight for the states, as it works with a penalty and an offset term. Therefore, the weights were distributed over the sustainability terms. The water consumption term received the highest weight (W_3) because of the water scarcity that plays a large role in mediterranean greenhouse production [3]. The same applies for the weights assigned to the actuators in Table 7.

Table 5: Conditions and parameter values regarding the temperature and humidity, with initial state $\boldsymbol{s}_1$, setpoint $\boldsymbol{SP}$, lower and upper vectors $\underline{R}$ and $\overline{R}$ of the state range, maximum deviation $\overline{e}$ from the setpoint, and maximum deviation $\overline{P}$ from the ranges.

	$\boldsymbol{s}_1$	$\boldsymbol{SP}$	$\underline{R}$	$\overline{R}$	$\overline{e}$	$\overline{P}$
s_1 (°C)	26	22	18	25	4	4
s_2 (%)	58	60	50	70	10	10
s_3 (°C)	26	-	-	-	-	-

Table 6: Outside climate conditions for the example problem, with temperature d_1, humidity d_2, solar radiation d_3 and wind velocity d_4.

	Value	Unit	
d_1	Temperature	20	°C
d_2	Rel. Humidity	64	%
d_3	PAR	300	Wm^{-2}
d_4	Wind Speed	6	ms^{-1}

Table 7: Initial settings $\boldsymbol{v}_0$ and weights w of the actuators.

Actuator		$\boldsymbol{v}_0$ (%)	$\boldsymbol{w}$
v_1	Window opening	0	0.1
v_2	Fog system	0	0.6
v_3	Heating system	0	0.3

Table 8: Weight of each term of objective function A, B and C.

Objective Function	W_1	W_2	W_3	W_4
A	0.70	0.30	-	-
B	0.70	0.05	0.15	0.1
C	0.10	0.60	0.30	-

6.3 Results

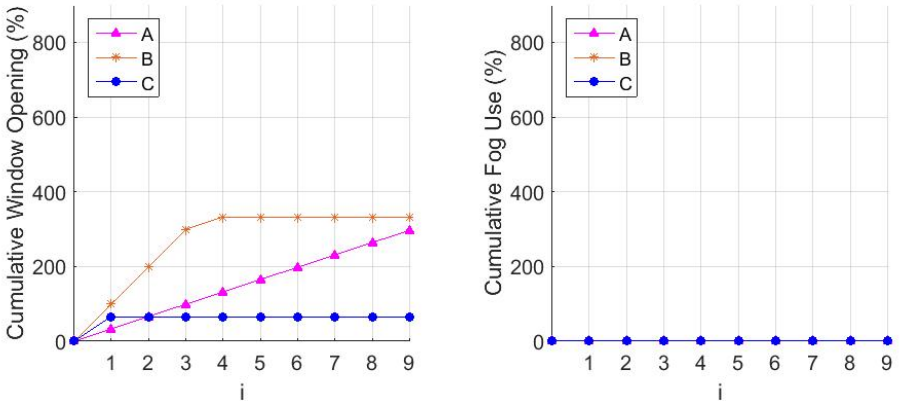
After running the program, each objective function yields a different optimal control sequence. In order to avoid extensive data tables, the resulting solutions are shown graphically in Figures 10a and 10b. The corresponding trajectories of the temperature and humidity are indicated in Figure 11. Additionally, the number of node evaluations required by the branch and bound algorithm for each objective function to find the optimal solution is shown in Figure 12.

The temperature trajectories show that all three objective functions can correct the temperature if it starts outside the permitted range. All three functions also show the inverse relation between temperature and humidity, as the decrease in temperature in Figure 11a corresponds to an increase in humidity in Figure 11b. Furthermore, the graph shows that objective function B follows the temperature setpoint most accurately and quickly. However, function A is the best at following the temperature *and* humidity setpoint. This is not unexpected since A is the only objective function to include the humidity state in its deviation term (see Section 5.3.2). Finally, function C only stays within the range, but does not attempt to follow the setpoint, as its priority is to minimize water and energy use.

Figure 10 shows the cumulative actuator settings of the sequences. The setting for the heater v_3 is not shown, as this was restricted to 0 (see Section 5.2). The figure shows that all three objective functions only use the window opening to reduce the temperature and do not use the fog system at all. This is because in functions B and C the water use has a cost, and function A refrains from using the fog due to the humidity increase, which would cause a deviation from the humidity setpoint.

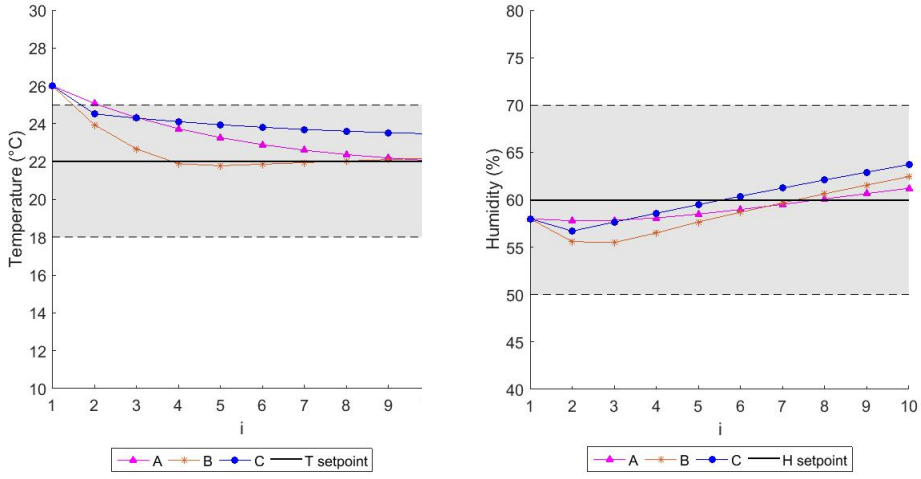
Finally, Figure 12 shows that function B requires many more node evaluations to find its optimal solution than the other two functions. The high number of node evaluations occurs when the algorithm needs to go deep into the tree before it can discard a branch. This can happen because the difference between the upper bound (best solution so far) and the other branches is small, which is more likely to occur in a function with many small terms, as is the case in function B. On the other hand, function C is by far the best in computational time. The 'offset' term ensures that branches violating the allowed range are quickly discarded, hence leaving very little of the tree to

explore.



(a) Window opening (b) Fog use

Figure 10: Cumulative use of actuators in optimal control sequences rendered by objective functions A, B and C.



(a) Temperature (b) Humidity

Figure 11: State trajectory of the greenhouse climate applying the optimal control sequences rendered by objective functions A, B and C, (see Figure 10).

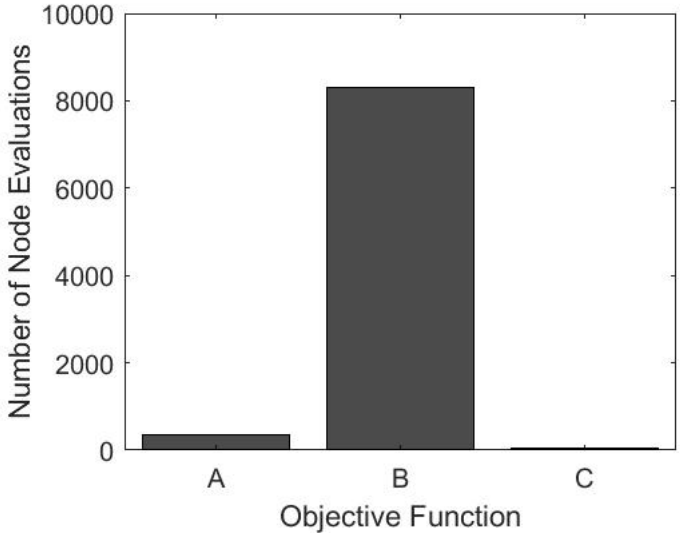


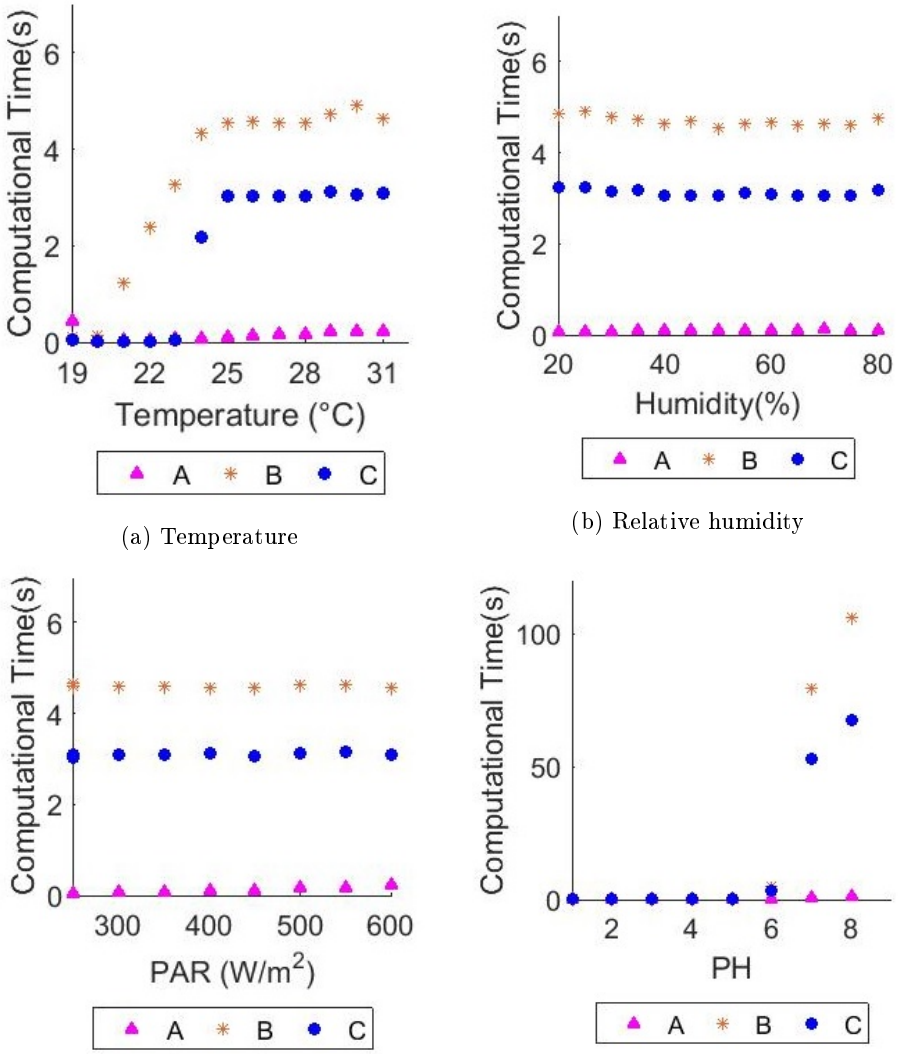
Figure 12: Number of node evaluations required to find the optimal solution of the example problem.

6.4 Design of Experiments

In order to examine the the model more thoroughly, the effect of outside climatic conditions on the computational time was examined. These are considered as "disturbances" in the climatic model. The disturbances that were analyzed are the outside temperature d_1, relative humidity d_2 and PAR d_3, shown in Figures 13a, 13b and 13c respectively. Furthermore, the effect of the length of the prediction horizon was examined as well, shown in Figure 13d. The parameters were tested while keeping all other climatic conditions at average values. The averages were calculated from the data of the summer month June in Almería, given in Table 9. All other parameters and weights were kept at the values specified in the example problem of Section 6.2.

Table 9: Average outside climate conditions from climatic data in Almería in June, with temperature d_1, humidity d_2, Photosynthetically Active Radiatio (PAR) d_3 and wind velocity d_4.

		Value	Unit
d_1	Temperature	25.5	°C
d_2	Rel. Humidity	58	%
d_3	PAR	476	Wm^{-2}
d_4	Wind Speed	4.5	ms^{-1}



(a) Temperature

(b) Relative humidity

(c) Photosynthetically Active Radiation

(d) Prediction Horizon

Figure 13: Computational time required by objective functions A, B and C at increasing temperatures (a), relative humidity (b), PAR (c) and PH (d), with all other conditions at the average values given in Table 9. All other parameters and weights were kept at the values specified in the example problem of Section 6.2. The prediction and control horizon of all runs in (a), (b), and (c) were $PH = 6$ and $CH = 4$ respectively.

The figures show that the computational time of objective function A is unaffected by any of the tested parameters. This suggests that the function is likely convex. For objective functions B and C, the only disturbance with a clear effect on the computational time is the temperature. Both the humid-

ity and the PAR do not seem to have any strong effect on the computational time. However, to confirm this as a certainty, the model should be tested at all combinations of climatic conditions instead of only at average conditions. Figure 13a shows that at outside temperatures lower than the allowed range of 18 to 25 °C, the computational time is low, while it increases at temperatures inside the range. Since the example problem of Section 6.2 had an outside temperature of 20 °C, it is a relatively simple case, hence clarifying the low number of node evaluations and computational time of the case. At temperatures higher than the allowed range, the computational time is the highest and appears to be constant. Therefore, it can be concluded that high outside temperatures can be problematic for the model.

Figure 13d shows the effect of the prediction horizon on the computational time. As was expected, the increase in complexity caused by the larger prediction horizon PH causes an apparently exponential increase in computational time. Recall from Section 5.1 that the time interval of the model is 15 s. Since MPC performs one optimization at every time interval, the maximumm computational time may thus be 15s for this model. From Figure 13d it is thus clear that, in this implementation medium and platform, the B&B optimization can only be performed within the 15s time limit for $PH \leqslant 6$, regardless of the objective function used.

Consequently, besides the temperature, the prediction horizon can also be identified as a possible problem source for the algorithm at high values. Moreover, the graphs show that the difference in computational time between functions B and C in the example problem is a trend in all the tested cases. In general, it seems that objective function C is the most suitable of the three functions for the branch and bound algorithm application. Function C requires the smallest number of node evaluations, and performs the best on sustainability without violating the temperature and humidity ranges. In the case that the user would prefer to steer the climate states more closely to the setpoints, the ranges can be made more narrow.

On a more general level, it is also necessary to evaluate the performance of the discrete branch and bound search as optimization method. The main drawback of the algorithm is that it requires a trade off between discretization and performance [20]. In order to keep the computational time low, the number of discrete control alternatives should be kept low. However, this coarse discretization results in relatively poor control performance, as the control space is greatly reduced [20]. Hence, the user must find a balance between discretization and control performance.

7 Modification for Node Reduction

7.1 General Approach

The computational time required to solve the MPC optimization problem with the B&B search algorithm is caused by the large number of nodes that must be evaluated. The nodes after the control horizon are relatively simple. Those nodes have only one branch (see Figure 5), since the control action remains constant from there until the end of the prediction horizon. Therefore, this portion of the search tree could be potentially interesting for computational time reduction. This possibility was investigated creating a modified algorithm which searches the final nodes slightly differently. Since all the nodes of a vertical branch after the control horizon have the same control action $\boldsymbol{v}_{CH}$, the cost and state at each node can be simulated until the prediction horizon in one single step. Consequently, after the control horizon, the modified algorithm does not create a new node at each level i, but rather evaluates all intervals after control horizon CH in one single node. A graphical representation is given in Figure 14. The general idea behind the modification is that a reduction in number of node evaluations will lead to a reduction in computational time.

To avoid confusion, the new algorithm uses a new index m for the levels of the tree instead of the index i, which has been used thus far. Up to the control horizon, both indices can be considered as alias indices, since $m = i$. However, in the modified algorithm, the nodes $i = CH + 1, \ldots, PH$ are comprised into one node $m = CH + 1$. Therefore, m is defined as $m = 1, \ldots, CH + 1$. In the case that $CH = PH$, there are no nodes after CH, so $m = 1, \ldots, CH$. In general form, m can thus be defined as:

$$m = 1, 2, \ldots, CH + h$$
$$h = \begin{cases} 0, & \text{if } CH = PH \\ 1, & \text{if } CH < PH \end{cases}$$

The transition cost at each node is re-defined as C. At the nodes where $m \leqslant CH$ the transition cost is calculated in the same manner as has been done thus far: $C = c(\omega_j, \boldsymbol{s}_i)$, using Eq.(40). At $m = CH + 1$, the transition cost C is the sum of all the individual transition costs $c(\boldsymbol{v}_{CH}, \boldsymbol{s}_i)$ for $i = CH + 1, \ldots, PH$ (recall that for $i > CH$ the control action $\boldsymbol{v}$ remains constant at $\boldsymbol{v}_i = \boldsymbol{v}_{CH}$). Therefore, the transition cost C at each level m is calculated as follows:

$$C(\omega, \boldsymbol{s}_m) = \begin{cases} c(\omega, \boldsymbol{s}_m) & m \leqslant CH \\ \sum_{i=CH+1}^{PH} c(\boldsymbol{v}_{CH}, \boldsymbol{s}_i) & m = CH+1 \end{cases} \tag{44}$$

with

$$\boldsymbol{s}_{i+1} = T(\boldsymbol{v}_{CH}, \boldsymbol{s}_i) \qquad \forall\, i = CH, \ldots, PH$$

The logical steps of the modified algorithm are outlined in Algorithm 3.

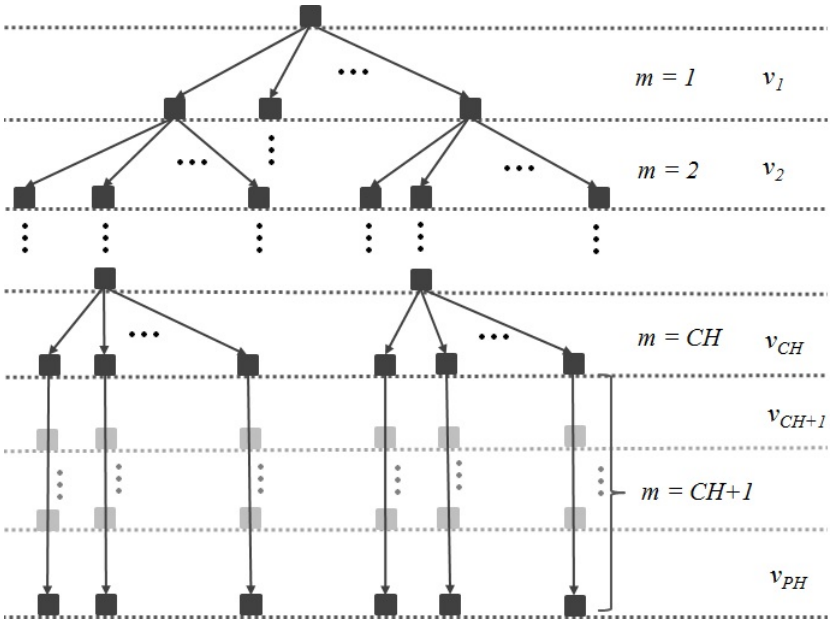


Figure 14: Graphical representation of the modified algorithm, in which the search tree simulates the nodes after the control horizon in one single node.

Algorithm 3 ControlSequenceModified($PH, CH, B, \boldsymbol{s}_1$))

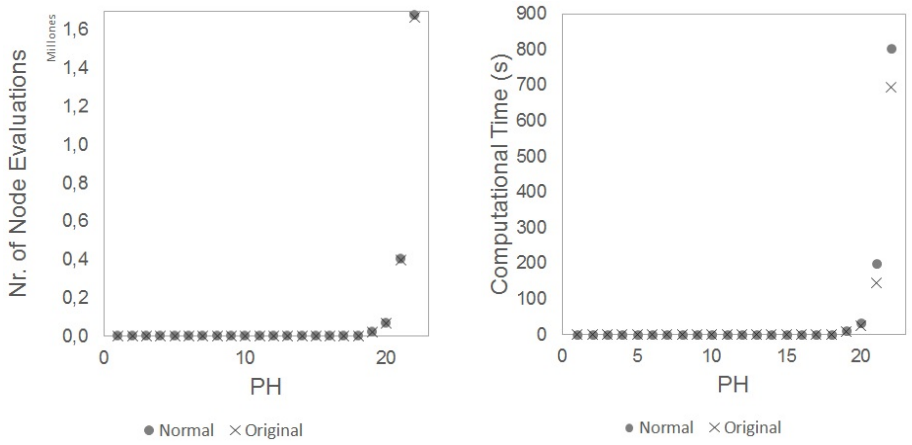
Require: PH, CH, B, and $\boldsymbol{s}_1$
  Compute alternative settings $\Omega := \{\omega_1, \ldots, \omega_B\}$
  $\{V, J^U\} := UpperBound(PH, CH, B, \boldsymbol{s}_1, \Omega)$
  Let $\Omega_a := \Omega$ for $a = 1, \ldots, CH$
  $m := 1$ and $J_0 := 0$
  while ($\Omega_a \neq \varnothing$ for $a = 1, \ldots, CH + 1$)
    remove ω from Ω_m
    calculate transition cost $C(\omega, \boldsymbol{s}_m)$ with Eq.(44)
    calculate total cost up to m: $J_m := J_{m-1} + C(\omega, \boldsymbol{s}_m)$
    if ($J_m < J^U$)
      if ($m \leqslant CH$)
        $\boldsymbol{v}_m := \omega$
        $m := m + 1$
      else
        $\boldsymbol{v}_i := \omega$, for $i = CH + 1, \ldots, PH$
        update upper bound $J^U := J_m$ and $V := (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{PH})$
        $m := CH$
    while ($\Omega_m = \varnothing$)
      $\Omega_m := \Omega$
      $m := m - 1$
    endwhile
    if ($m > CH$), $\Omega_m := \{\boldsymbol{v}_{CH}\}$
  endwhile
  return: V, J^U

7.2 Results

The modified algorithm was created under the premise that a reduction in node evaluations will also lead to a reduction in computational time. To test this, both the original and the modified algorithms were used to solve the same optimization problem, and the number of node evaluations and computational time of both were compared. The control problem on which the models were tested is formulated in Section 6.2. Due to the practical issue of time availability for this report, the modified algorithm was implemented with objective function C, as this was deemed the easiest objective function in Section 6.4.

To obtain a more comprehensive overview of the effect of the modification in the algorithm, the problem was also solved at different prediction horizon lengths. At all the tested PH values, the $CH : PH$ ratio was kept constant at 2:3. These results are plotted in Figure 15. Additionally, the models were tested on the same problem with a constant prediction horizon of $PH = 20$

but with different ratios of control to prediction horizon. The results are given in Figure 16.



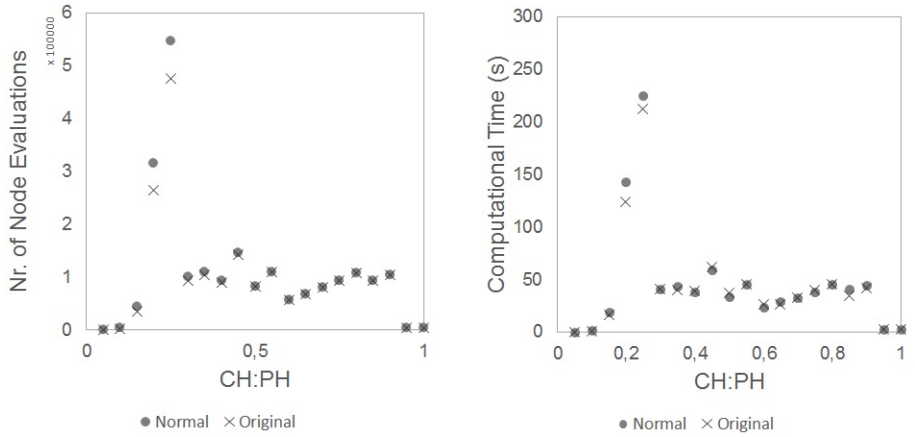
(a) Number of node evaluations

(b) Computational time

Figure 15: Node evaluations and computational time of the original and the modified algorithm at increasing PH values under a constant $CH : PH$ ratio of 2:3.

Figure 15a shows that the problem evaluates a constant number of nodes at prediction horizon values up to $PH = 19$. This is likely caused by the fact that at these prediction horizon lengths, the algorithms can stay within the admitted temperature and humidity range without implementing any actuator change (see example in Figures 11 and 12). For a larger prediction horizon PH, the problem becomes more complex and both the number of node evaluations and the computational time increases exponentially. The graphs show that by implementing the modified model, the number of node evaluations is not reduced. Although the data did show a slight reduction, it was so small that it does not show in the graph. On the other hand, Figure 15b shows that at PH values where the number of node evaluations was high ($PH \geqslant 19$), there is a substantial reduction in computational time.

Figure 16 shows the result of investigating the length of the control horizon with respect to the prediction horizon. It was expected that for a longer control horizon, the complexity of the problem would be higher and thus the number of node evaluations and computational time would increase. However, both graphs show that there is no such pattern in the observed data. There is a clear peak at $CH : PH = 5 : 20 = 0.25$ and otherwise the nodes and the time stay within a relatively small range from each other. It was also unexpected that at the highest $CH : PH$ ratios of 0.9 and 1, the computational effort was the lowest. The data shows that implementing the modified model in most cases did not lead to a great



(a) Number of node evaluations

(b) Computational time

Figure 16: Node evaluations and computational time of the original and the modified algorithms at increasing $CH : PH$ ratio under a constant prediction horizon of $PH = 20$

reduction of computational effort, except at the peak around $CH : PH = 0.2$. At those points, the data shows a reasonable reduction in number of node evaluations as well as computational time. Therefore, the general overall conclusion from Figures 15 and 16 is that the adjustment to the model *does* lead to a computational time reduction, but only in the cases where the computational effort is high.

8 From Discrete to Continuous Control Space

Thus far the issue of greenhouse model predictive control has been explored with a Branch and Bound algorithm applied to problems with discrete control space. However, as in reality the control space of greenhouse actuators is continuous, it is interesting to examine the possibility of applying the branch and bound algorithm to a continuous control space. This would eliminate the previously discussed need for a trade off between discretization and control performance (Section 6.4). This chapter outlines how the transition from discrete to continuous control space was approached.

8.1 Multi-modality of the Objective Functions on the Control Space

In order to apply the branch and bound algorithm to a problem with continuous search space, firstly the multi-modality of each of the three objective functions of Section 5.3 must be examined. This is necessary to determine if there are multiple local minima; or in other words, if the functions are non-convex. However, the control problem has a very high dimensionality, since the setting of each actuator at each time interval is a variable. A simple problem with, for example, a control horizon $CH = 4$ and three actuators, already has $n = 4 \times 3 = 12$ dimensions. Since such a high-dimensional control space cannot be examined graphically, an alternative method was applied to determine the presence of multiple optima.

The control space was explored by applying Multistart. This is an algorithm where local searches are performed from randomly generated starting points [12]. Each starting point will reach one of the local optimum points in the search space. The local minimum that is reached depends on the region of attraction in which the starting point is situated, and on the local optimizer that is used [12]. For all three objective functions, 100 starting points were generated and solved locally with the FMINCON function in Matlab R2015b. This was done repeatedly for increasing prediction horizons, always given a ratio of control to prediction horizon of $CH : PH = 2 : 3$. The resulting solutions, i.e. local minima, were rounded off to integer numbers. This implies that all control settings within a distance of 1 from each other are considered to have the same value. Although this is a relatively rough method of examination, it is sufficient for the purpose of determining the presence of multiple minima. The number of local minima that resulted for each run is shown in Figure 17.

The graph shows that objective function A always has only one solution, regardless of the problem's prediction horizon. The minimum point found by the local solver is thus the global optimum. Admittedly, there is a possibility that if the number of random starting points were higher, the local solver would find more than one minimum. However, for simplification purposes

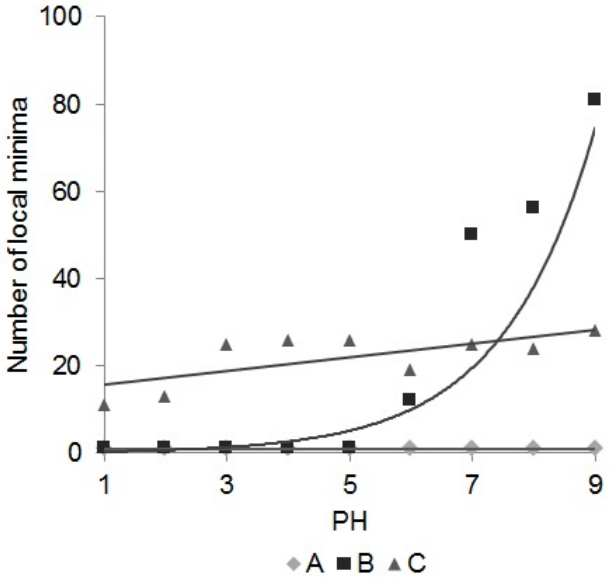


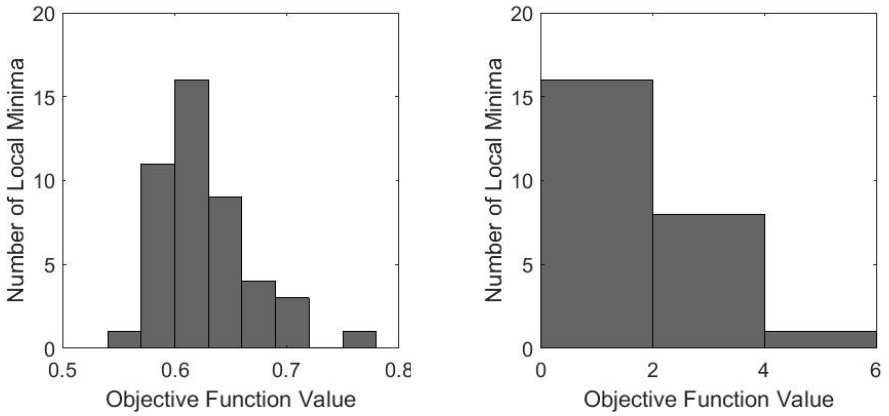
Figure 17: Number of local minima for objective functions A, B and C at increasing prediction horizon length using 100 random starting points and a ratio of $PH : CH = 2 : 3$.

it will be assumed that the function has a convex shape, and thus does not require any non-convex optimization method. Function A is thus dismissed, as it is of no added value to apply the B&B algorithm to its search space.

On the other hand, objective functions B and C show multiple local minima. Function B seems to be convex for problems with $PH < 6$, but as the dimensionality of the problem becomes higher, the number of local minima increases drastically. Conversely, function C has a large number of minima regardless of the length of the prediction horizon. With increasing prediction horizon PH, the number of minima increases slightly, yet not as steeply as for function B. The histograms in Figure 18 show that for both functions B and C, the local minima are distributed over different objective function values, hence proving that they are truly different from each other. The main conclusion that can be derived from this is that functions B and C are indeed non-convex, and thus require, as expected, a non-convex optimization method such as branch and bound to identify their global minimum.

8.2 Computing the Lower Bound

In order to perform a search in a continuous search space, it is necessary to evaluate intervals of the control settings, instead of discrete values. In order to avoid confusion between control space intervals and time intervals, the



(a) Objective Function B (b) Objective Function C

Figure 18: Histograms of objective function values of the local minima found for functions B and C with 100 random starting points, for a problem with $PH = 7$.

time intervals i, with $i = 1, \ldots, PH$, will be referred to in this chapter as *stages*. The control actions $\boldsymbol{v}$ that have thus far been comprised of discrete values, are re-defined as a constrained control space box of NA dimensions:

$$\boldsymbol{v} = [\underline{v_1}, \overline{v_1}] \times [\underline{v_2}, \overline{v_2}] \times \cdots \times [\underline{v_{NA}}, \overline{v_{NA}}] \tag{45}$$

Therefore, it is impossible to calculate the exact transition cost c of $\boldsymbol{v}$. However, it is possible to determine the lowest possible value of c, given the intervals $\boldsymbol{v}$ of the actuators. This value can be used as the lower bound of the transition cost, denoted c^L. Since the optimization problem makes use of an analytic model to calculate the state transition, as opposed to a black box model, it is well suited to apply the natural inclusion function to calculate its lower bounds [19]. This consists of exchanging the usual elementary operators by their interval extensions [19]:

$$\begin{cases} [a,b] + [c,d] & = [a+c, b+d] \\ [a,b] - [c,d] & = [a-d, b-c] \\ [a,b] \times [c,d] & = [\min\{ac, ad, bc, bd\}, \max\{ac, ad, bc, bd\}] \\ [a,b] \div [c,d] & = [a,b] \times \left[\frac{1}{d}, \frac{1}{c}\right] \quad \text{if } 0 \notin [c,d] \end{cases}$$

The above given interval extensions were applied to each operation in the transition cost function c, in Eq.(40). It was hence also applied to the state transition function equations of the climate model. By implementing the extensions consistently throughout the equations, the natural inclusion function of the transition cost was obtained, denoted by F. The inclusion

function yields the maximum and minimum values of the transition cost c, given the actuator intervals $\boldsymbol{v}$ and state intervals $\boldsymbol{s}$.

$$F(\boldsymbol{v}, \boldsymbol{s}) = [\underline{c(\boldsymbol{v}, \boldsymbol{s})}, \overline{c(\boldsymbol{v}, \boldsymbol{s})}] \tag{46}$$

The lowest value of the range was taken as the lower bound of the transition cost, c^L:

$$c^L(\boldsymbol{v}, \boldsymbol{s}) = \min(F(\boldsymbol{v}, \boldsymbol{s})) \tag{47}$$

The sum of c^L up to i is a lower bound of the cost thus far:

$$J_i^L = \sum_{\ell=1}^{i} c^L(\boldsymbol{v}_\ell, \boldsymbol{s}_\ell) \qquad \forall\ i = 1, \ldots, PH \tag{48}$$

It follows that the lower bound of the cost over a complete control sequence V with PH stages is:

$$J^L = J_{PH}^L = \sum_{i=1}^{PH} c^L(\boldsymbol{v}_i, \boldsymbol{s}_i) \tag{49}$$

8.3 Method 1: Multi-Stage Bisection

This section describes the first algorithm that was developed to search for a solution in a continuous search space. The aim of the developed algorithm is to stay close to the B&B search method developed thus far. To achieve this, the total control space of $NA \times PH$ dimensions, was broken up into PH separate control space boxes of dimensions NA. In other words, each stage i has its own control space $\boldsymbol{v}_i$. At the start of the search, the box of each stage i is denoted $\boldsymbol{v}_{i0}$. The continuous set S_0 comprises the initial search boxes of all stages $i = 1, \ldots, PH$, and therefore encompasses the whole search space.

$$S_0 = \boldsymbol{v}_{10} \times \boldsymbol{v}_{20} \times \ldots \times \boldsymbol{v}_{PH0}$$

Let the sub-set S_t be a series of PH control boxes denoted $\boldsymbol{v}_{it}$. Therefore, S_t is defined as:

$$S_t = \boldsymbol{v}_{1t} \times \boldsymbol{v}_{2t} \times \ldots \times \boldsymbol{v}_{PHt} \tag{50}$$

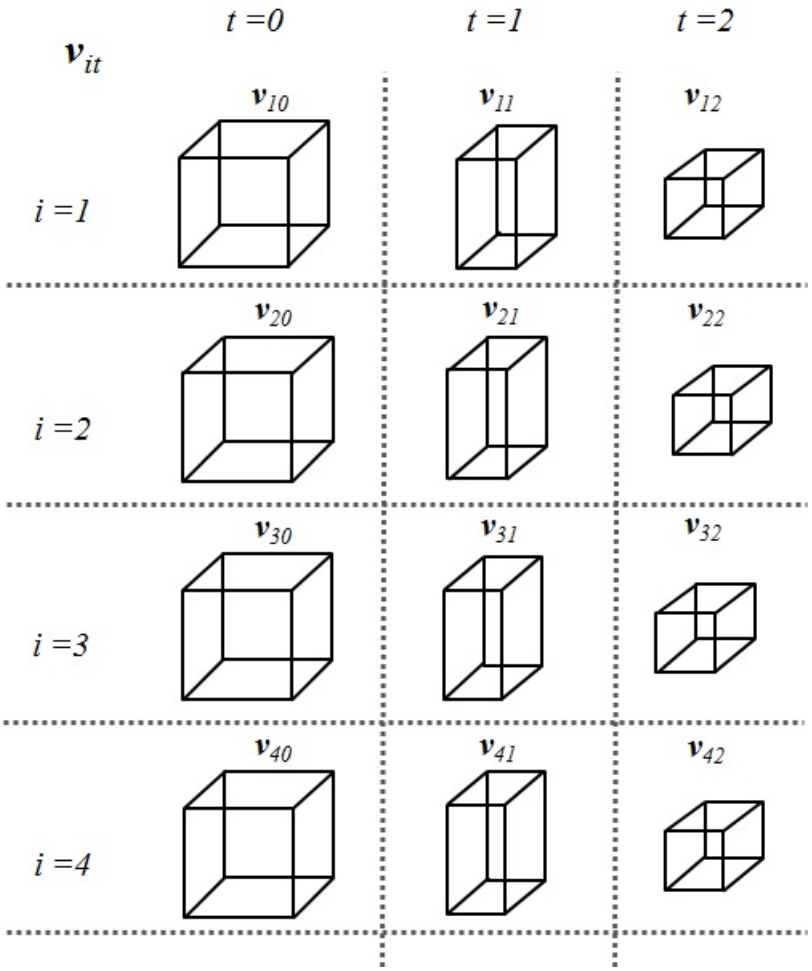


Figure 19: Graphical representation of individual control space for each stage i, narrowed down in each trajectory t with a bisection.

To minimize the data storage, the algorithm performs a depth first search. Therefore, instead of refining each control box $\boldsymbol{v}_i$ in depth separately, the algorithm makes use of multiple trajectories, each going through all the control stages 1 to PH. Let a trajectory, indexed t, be a search through all stages i in the set S_{t-1}, yielding the subset S_t. When the algorithm reaches the prediction horizon, it starts a new trajectory $t+1$, which searches the set S_t by going through all the stages again and yields the set S_{t+1}. This process is performed repeatedly until the desired size of a subset is reached.

The optimization problem outlined in Section 6 has three actuators (windows, fog system and heating), each with a control range of [0,100]. The set S_0 of control space boxes before the start of the search, at $t = 0$, is thus:

$$S_0 = \boldsymbol{v}_{10} \times \boldsymbol{v}_{20} \times \ldots \times \boldsymbol{v}_{PH0}$$
$$\text{with} \quad \boldsymbol{v}_{i0} = [0, 100] \times [0, 100] \times [0, 100] \qquad \forall\ i = 1, \ldots, PH$$

A trajectory t starts at the the first stage $i = 1$. At each stage i, the algorithm retrieves the control space box in S_{t-1} belonging to stage i, denoted $\boldsymbol{v}_{i(t-1)}$. The longest edge of $\boldsymbol{v}_{i(t-1)}$ is bisected in the middle, resulting in two alternative control spaces for stage i, denoted ω_1 and ω_2. Both are stored in Ω_{it}. Each control alternative ω is a branch in trajectory t at stage i.

For a branch ω, the algorithm calculates the lower bound of its cumulative cost up to i, denoted J_i^L, with Eq.(48). If $J_i^L > J^U$, then the branch is pruned, and the algorithm continues to search the next branch. If the branch is not bounded, then ω is stored as $\boldsymbol{v}_{it}$ in S_t. Then the algorithm continues to the next stage $i + 1$, and bisects the next control box.

If the algorithm reaches the prediction horizon at trajectory t, the control space narrowed down is denoted as the set S_t. The algorithm then determines a feasible control sequence $V_f \in S_t$. For the specific control problem at hand, V_f was chosen to be the left side of the control intervals in S_t. The objective value of V_f is $J(V_f) = J$, calculated through Eq.(9). If $J < J^U$, then the upper bound is updated as $J^U = J$ and $V = V_f$. Subsequently, the algorithm starts a new trajectory $t + 1$, starting again at $i = 1$.

The algorithm starts a new trajectory every time the prediction horizon is reached. Every trajectory narrows the boxes further down bisecting the control spaces again. This process is continued until the maximum number of trajectories NT is reached. The value of NT depends on the accuracy ϵ to which the user wishes to narrow down the search. The following equation gives the relation for the number of trajectories NT necessary to narrow down the control space of stage i to a maximum width ϵ, given NA actuators:

$$\epsilon \geqslant \frac{Size(\boldsymbol{v}_{i0})}{2^{\frac{NT}{NA}}}$$

$$\text{with} \quad Size(\boldsymbol{v}) = \max_a \{\overline{v_a} - \underline{v_a}\}$$

The initial control space $\boldsymbol{v}_{i0}$ is the same for all stages i. Consequently, NT does not depend on stage i. In the next equation, $i = 1$ was chosen. The number of trajectories NT must be an integer number, so it is rounded up to the next integer value:

$$NT(\epsilon, \boldsymbol{v}_{10}) = \left\lceil NA \times \log_2 \left(\frac{Size(\boldsymbol{v}_{10})}{\epsilon} \right) \right\rceil \tag{51}$$

When all trajectories have been completely evaluated, the current best sequence V can be concluded to be the optimum control sequence. The

logical steps that the algorithm takes to search the whole space are outlined in Algorithm 4.

Note that at every new stage i, the algorithm must retrieve $\boldsymbol{v}_{i(t-1)}$ and compute the new branches in Ω_{it} from it. Therefore, one of the main differences from the discrete algorithm is that the continuous algorithm has a different set of branches to evaluate at each stage i, while the discrete algorithm has a fixed set Ω of alternative control actions computed before starting the tree evaluation.

Algorithm 4 MultiStageBisection(PH, CH, ϵ, $\boldsymbol{s}_1$, S_0))

Require: PH, CH, ϵ, $\boldsymbol{s}_1$, and S_0
Determine maximum number of trajectories $NT(\epsilon, \boldsymbol{v}_{10})$ with Eq.(51)
Let $\Omega_{a1} :=$ Bisection$(\boldsymbol{v}_{a0})$ for $a = 1, \ldots, PH$
$i := 1$, $t := 1$ and $J_0 := 0$
while $(\Omega_{ab} \neq \varnothing)$ for $a = 1, \ldots, PH$ and $b = 1, \ldots, NT$
remove ω from Ω_{it}
calculate transition cost lower bound $c^L(\omega, \boldsymbol{s}_i)$ with Eq.(47)
calculate new state $\boldsymbol{s}_{i+1}$
calculate total cost lower bound up to i: $J_i^L := J_{i-1}^L + c^L(\omega, \boldsymbol{s}_i)$
if$(J_i^L < J^U)$
$\boldsymbol{v}_{it} := \omega$
if $(i < PH)$, $i := i + 1$
else
define set $S_t = \boldsymbol{v}_{1t} \times \ldots \times \boldsymbol{v}_{PHt}$
choose feasible sequence $V_f \in S_t$, with $J(V_f) = J$
if $(J < J^U)$, $J = J^U$, $V = V_f$
if $(t < NT)$, start new tree $t := t + 1$, $i = 1$
$\Omega_{it} :=$ Bisect$(\boldsymbol{v}_{i(t-1)})$
while $(\Omega_{it} \neq \varnothing)$
if $(\Omega_{at} = \varnothing)$, for $a = 1, \ldots, PH$
$t := t - 1$ and $i := CH$
else
$\Omega_{it} :=$ Bisect$(\boldsymbol{v}_{i(t-1)})$
$i := i - 1$
endwhile
endwhile
return: V, J^U

Algorithm 5 Bisect($\boldsymbol{v}$)

Require: $\boldsymbol{v}$
Find a largest edge $LE \in \text{argmax}_a\{\overline{v}_a - \underline{v}_a\}$
Bisect $\boldsymbol{v}$ by largest edge LE into ω_1 and ω_2
return: $\{\omega_1, \omega_2\}$

8.4 Method 2: Multi-stage Multisection

The second method that has been developed to solve the problem in a continuous search space is largely the same as Method 1. In this method the search space is also divided into separate boxes for each stage i. However, instead of bisecting the boxes, this methods attempts to follow the concept of the discrete algorithm by evaluating "all possible combinations" of actuators at each stage i. Therefore, instead of bisecting only the largest edge, this approach bisects all edges in each branching operation. A graphical representation of the multisection is shown in Figure 20.

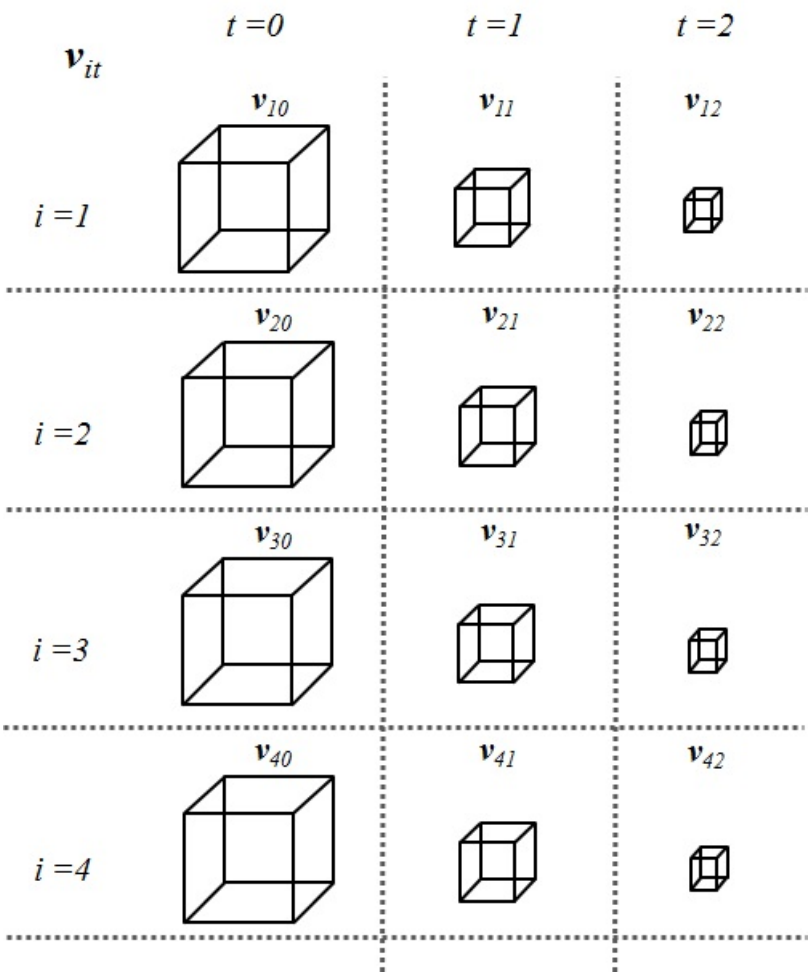


Figure 20: Graphical representation of individual control space for each stage i, narrowed down in each search trajectory t by an octosection.

All control intervals v_a, with $a = 1, \ldots, NA$, are bisected into intervals v_{a1} and v_{a2} simultaneously. All the permutations of the resulting control intervals are denoted ω_j, with $j = 1, \ldots, B$, and stored in the matrix Ω_{it}. An example of the permutations is shown in Table 10.

Since the control intervals are split in two, the number of permutations B, given NA actuators, is:

$$B = 2^{NA}$$

In the specific case of $NA = 3$ (window opening, fog system and heater), the number of permutations is thus

$$B = 2^3 = 8$$

hence leading to an octosection. Each permutation ω_j, with $j = 1, \ldots, B$, is

a portion of the control space of stage i. Each permutation ω_j is therefore a branch that must be evaluated at stage i in trajectory t. Since this method bisects all actuator intervals at each trajectory t, the number of trajectories NT is not dependent on the number of actuators NA. NT is defined as:

$$NT(\epsilon, \boldsymbol{v}_{10}) = \left\lceil \log_2 \left(\frac{\text{Size}(\boldsymbol{v}_{10})}{\epsilon} \right) \right\rceil \tag{52}$$

An example of the branching in the first two trajectories is shown in Figure 21. The logical step taken by the whole algorithm are the same as that for Method 1, outlined in Algorithm 4. The only differences are that the number of trees NT is calculated by Eq.(52), and that the call on the Bisect algorithm is replaced by a call on Multisect, defined in Algorithm 6.

Table 10: Matrix Ω_{it} of permutations $\omega_1, \ldots, \omega_8$ for actuators $v_1, v_2, v_3 \in \{[0, 50], [50, 100]\}$.

	ω_1	ω_2	ω_3	$\cdots$	ω_8
v_1	$[0, 50]$	$[0, 50]$	$[0, 50]$	$\cdots$	$[50, 100]$
v_2	$[0, 50]$	$[0, 50]$	$[50, 100]$	$\cdots$	$[50, 100]$
v_3	$[0, 50]$	$[50, 100]$	$[0, 50]$	$\cdots$	$[50, 100]$

Algorithm 6 Multisect($\boldsymbol{v}$)

Require: $\boldsymbol{v}$
  for $a = 1$ to NA
    bisect v_a at middle point into intervals v_{a1} and v_{a2}
  define all possible permutations $\{\omega_1, \ldots, \omega_B\}$ of $\begin{pmatrix} v_1 \\ \vdots \\ v_{NA} \end{pmatrix}$
    with $v_a \in \{v_{a1}, v_{a2}\}$ for $a = 1, \ldots, NA$
  return: $\{\omega_1, \ldots, \omega_B\}$

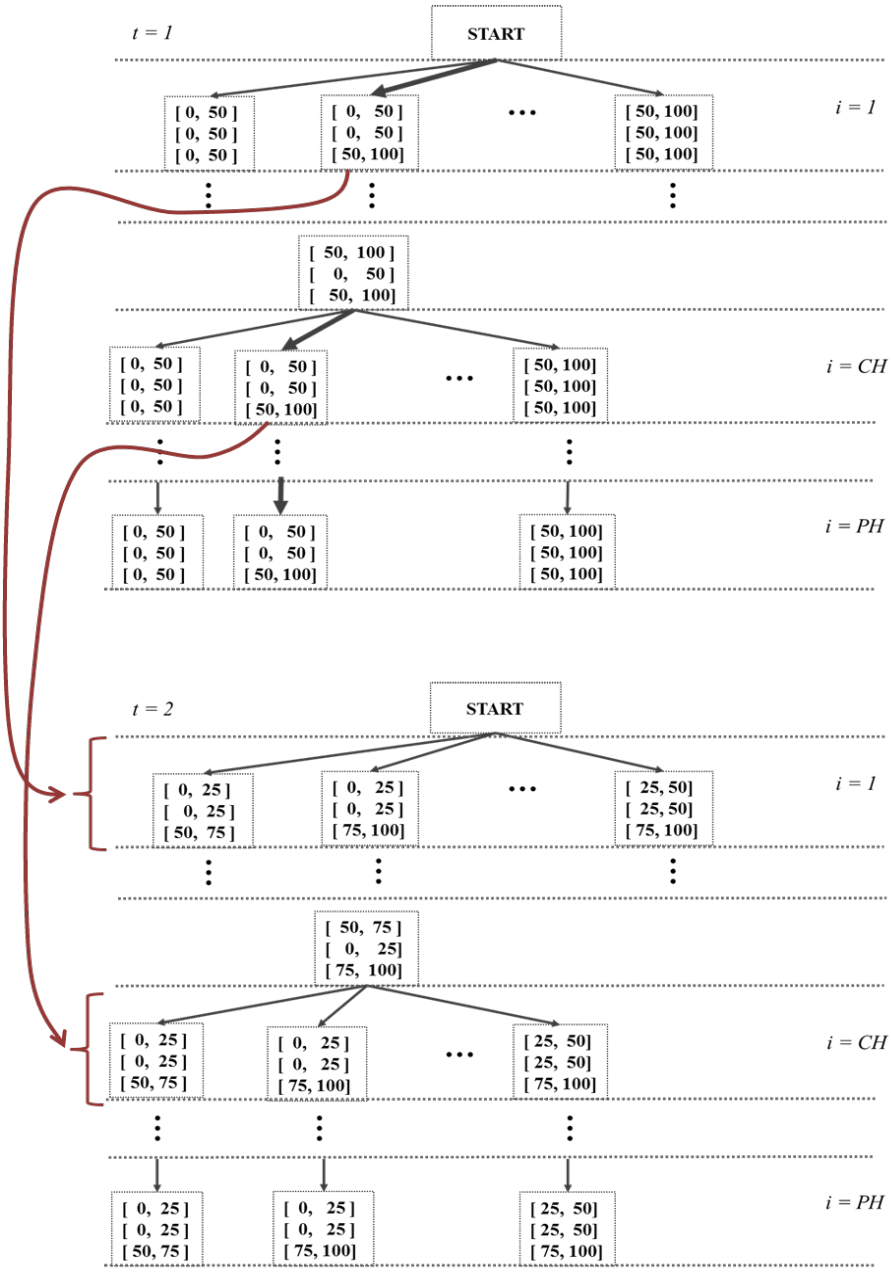


Figure 21: Branching process of the multi-stage multisection.

8.5 Method 3: Single-stage Bisection

The second algorithm implemented to solve the optimization problem in a continuous control space is the conventional application of branch and

bound, disregarding the stage aspect of the problem, outlined in Section 4.1. This method was applied in order to have a benchmark algorithm for the other two continuous algorithms that were developed. The single-stage bisection searches the whole control space over all time stages at once. A problem with NA actuators and prediction horizon PH will have $PH \times NA$ variables in its control sequence V. Each variable is denoted by v_{ai}, with $a = 1, \ldots, NA$ and $i = 1, \ldots, PH$. Recall Eq.(5):

$$V = \begin{pmatrix} v_{11} & v_{21} & \cdots & v_{PH1} \\ v_{12} & v_{22} & \cdots & v_{PH2} \\ \vdots & \vdots & \ddots & \vdots \\ v_{1NA} & v_{2NA} & \cdots & v_{PHNA} \end{pmatrix}$$

The search starts with a set S_0 enclosing the whole feasible area, stored in Λ. The set is removed from Λ and bisected into two subsets $S1$ and S_2 by the longest edge. The algorithm calculates the lower bound J^L for both subsets, according to Eq.(49). If $J^L > J^U$, the subset is discarded. In the case that $J^L < J^U$, the subset is stored in Λ. Furthermore, the left sides of the subset intervals are taken as a feasible control sequence V_f. If the objective cost $J(V_f)$ is lower than J^U, then the upper bound is updated by $J^U = J(V_f)$ and the optimal solution $V = V_f$. Any subsets in Λ with J^L lower than the new upper bound are removed.

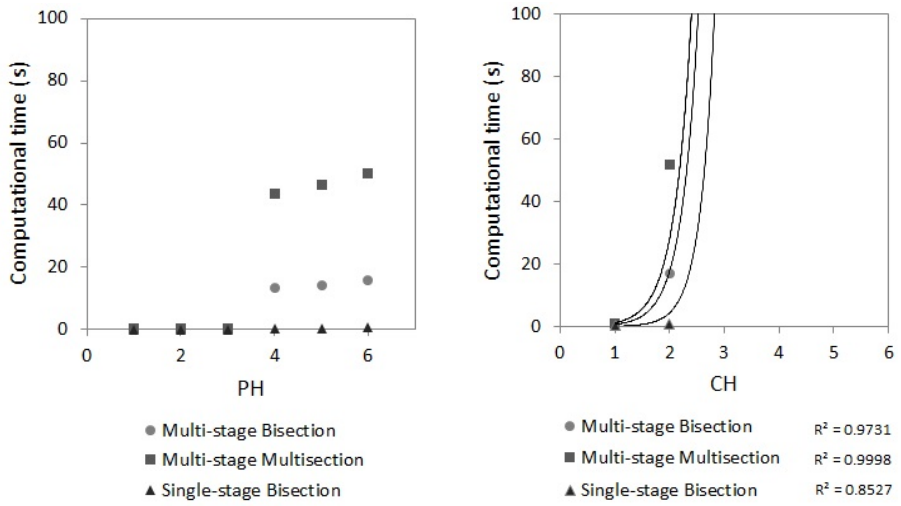
After both subsets are either discarded or stored, the algorithm takes a new subset from Λ and repeats the same steps as outlined for S_1. As this process continues, the width of the sets diminishes. Sets that have a maximum width of ϵ are not bisected anymore, only evaluated and removed from Λ. The branching and bounding process continues until Λ is empty. The best solution thus far, V, hence becomes the best overall solution. The logical steps taken by the algorithm are outlined in Algorithm 7.

Algorithm 7 SingleStageBisection($S_0, \epsilon, \boldsymbol{s}_1$)

Require: S_0, ϵ and $\boldsymbol{s}_1$
  Determine upper bound J^U
  Store S_1 in Λ; $r := 1$
  while $\Lambda \neq \varnothing$
    Remove a subset S from Λ and split
      into 2 new subsets S_{r+1} and S_{r+2}
    Determine lower bounds J^L_{r+1} and J^L_{r+2}
    for $p = r + 1$ to $r + 2$ do
      if $J^L_p < J^U$
        determine a feasible point $V_f \in S_p$ and $J(V_f) = J$
        if $J < J^U$
          $J^U := J$
          remove all sets S_k from Λ with $J^L_k > J^U$
        elseif $Size(S_p) \geqslant \epsilon$, Store S_p in Λ
    $r := r + 2$
  endwhile
  return: V, J^U

8.6 Comparison of Methods

The alternative optimization methods outlined in Sections 8.3, 8.4, and 8.5 were programmed in Matlab 2015b in order to compare them. The accuracy of the methods was checked by comparing their solutions. Due to the determininstic nature of the optimization problem, all three methods should yield exactly the same control sequence V, down to the last decimal. Since this is the case, it is assumed that all three methods are correct. To inspect the quality of the methods further, their computational time was compared. The example problem of Section 6.2 was solved using all three alternative methods, at increasing prediction and control horizon lengths. Runs that took longer than 500 seconds were stopped and recorded at this value. The results are shown in Figure 22.



(a) Prediction Horizon

(b) Control Horizon

Figure 22: Computational time required by the three alternative solution methods Multi-stage Bisection, Multi-stage Multisection, and Single-stage Bisection, to solve the optimization problem described as example problem in Section 6.2 at increasing PH and CH values respectively. All runs were performed with $\epsilon = 1$. All runs in (a) were performed with a $CH : PH$ ratio of 1:3, and all runs in (b) had a prediction horizon of $PH = 6$.

Figure 22a shows in increasing computational time with increasing prediction horizon PH length. This was expected, as the same pattern was found for the discrete algorithm in Figures 15 and 13d. Notice that for both the discrete and the continuous algorithm the computational time drastically increases at $PH > 6$. In the continuous control space, at runs with $PH < 4$ neither of the three methods shows an advantage over the others. For runs with $4 \leqslant PH \leqslant 6$, the single-stage bisection has the lowest computational time, followed by the multi-stage bisection, while the multi-stage octosection is the slowest.

The same order of computational speed occurs in Figure 22b. The graph also shows that the maximum control horizon that the algorithms can handle within the 500s time span, given $PH = 6$, is $CH = 2$. The computational time appears to increase exponentially with CH, since the R^2 values of the exponential trendlines are all close to 1. The reason for this pattern is that, since the branching only occurs before the control horizon, for a longer control horizon CH the complexity of the problem increases.

Overall, all three methods can be concluded to be able to handle only relatively low complexity problems. For the multi-stage multisection, the reason for the high computational time is the large number of node evaluations that must be performed. Since the control space at each stage i is split

in eight, the search tree grows very wide and only small sections of the tree can be pruned. The multi-stage bisection splits the control space in two. Therefore, it can bound larger tree sections and achieve a lower computational time than the octosection. However, since the search space is merely bisected, three times as many splittings are needed to achieve the same interval width ϵ. This method therefore requires a much deeper search. In the single-stage bisection method, the high computational time occurs because the state and control intervals of the inclusion function widen at every stage i, hence making the lower bounds less accurate. Consequently, for a larger prediction horizon PH, the range of the inclusion function becomes too wide and the lower bounds are not high enough to prune large sub-trees.

Regarding the applicability of the methods, it is necessary to recall from Section 5.1 that time was discretized into intervals of 15 s, each of which is one stage. In MPC, one optimization problem must be solved at every stage. Therefore, the computational time of one optimization problem may not be higher than 15 s. However, the computational time is dependent on the implementation medium and platform. As a consequence, conclusions can only be made about the computational time of the methods with respect to each other. The absolute values of the computational time could be reduced by running them on other programs, so it is not possible to conclude whether the algorithms would be applicable in reality.

In both graphs, the single-stage bisection is faster than the multi-stage searches. Consequently, it can be concluded that the branch and bound application from literature does not have any advantages over a regular branch and bound search when applied in continuous control space.

9 Discussion and Conclusion

9.1 Discussion

This thesis investigated the branch and bound algorithm applied to MPC with variants for a discrete and continuous control space. The results of the experiments were already discussed in previous chapters; therefore, this chapter focuses on a discussion of the assumptions that were made throughout the report.

To evaluate results of the experiments, the computational time of the algorithms was compared. However, it must be kept in mind that computational time can also be influenced by possible programming inefficiencies, which are caused by the programmer and are hence unrelated to the algorithm itself. On the other hand, algorithms can also be compared using the number of node evaluations. Yet if the algorithms that are being compared do not perform the same computation in a node, the number of node evaluations is also not a valid method of comparison. Therefore, this leads to the discussion point of the validity of comparisons. In order to make a precise comparison, it would be necessary to revise the code of the models to ensure that the results are not influenced by the programmer.

Another important issue is the validity of the objective functions that were implemented. Although the functions were taken from literature, they are a simplistic approach to describing the objectives of the user. In reality, climate control is a multi-objective problem, and should hence be tackled as such. This puts the results of the report in question, as it raises the doubt whether the branch and bound algorithm would still be applicable and useful in the case of a multi-objective optimization problem.

Finally, the nature of the optimization problem leads to the notion that it might also be possible to solve it using dynamic programming. This technique also uses a transition function to optimize variables in time. However, dynamic programming searches from the last to the first time stage, as opposed to the branch and bound method, which searches the stages in their chronological order. To further evaluate the quality of the branch and bound method, it would be of interest to compare it to dynamic programming to determine which of the two is more suitable.

9.2 Conclusion

The main purpose of this study was to determine the applicability of branch and bound to climate control in greenhouses. Initially, it was found that climate control is usually performed using Proportional Integral Derivative (PID) controllers, which correct for past disturbances of the climate based on simplified transfer function models [28]. An alternative control method is Model Predicitive Control (MPC), which controls the actuators based on a prediction of the future climate state. MPC discretizes time into intervals

and repeatedly optimizes a control sequence V over a prediction horizon of PH time intervals. Literature shows that branch and bound can be used as an optimization method for the control sequence.

The B&B optimization of control sequence V was examined in the available literature. It was found that the algorithm calculates the transition cost of a discrete control action $\boldsymbol{v}$ at every time interval of the prediction horizon, and bounds the search if the sum of the transition costs up to that interval is higher than an upper bound. To test the applicability of the search method, it was extended to a model with climate simulation. The literature was reviewed regarding objective functions for climate control. A comparison was made of three alternative functions by implementing them in the previously mentioned model. It was found that the most suitable objective function is one that does not aim to follow a specific setpoint, but instead keeps the climate state variables within a certain range. By implementing a penalty and an offset for solutions violating the allowed range, the function reduces the computational time while adhering to the desired climatic conditions in the greenhouse.

From the implementation of the programmed model it was concluded that the B&B search could prove a useful optimization method. Its main advantages are that it always finds the global optimum, it deals with constraints implicitly, and it requires no initial guess of the solution. The computational time of the method was found to be mostly dependent on the prediction horizon and the outside temperature an input parameter of the climate model. At increasing prediction horizon PH lengths the computational time increases exponentially, while increasing outside temperatures within the permitted range leads to a linear computational time increase. Computational time for outside temperatures below and above the permitted range were found to be constant, yet higher for temperatures above the range. Overall, the B&B search was deemed applicable, given the specified parameter values, at prediction horizon of lengths lower than 6.

The main disadvantage of the B&B optimization is that it requires a discretization of the control space. Consequently, it requires a trade-off between discretization and performance [20]. A possible computational time reduction was investigated through a modification of the algorithm. The modified algorithm evaluates all intervals after the control horizon CH in one single node by simulating all the states until the end in one step. This was possible because after the control horizon all control actions stay constant. The modification was tested in a programmed model, and found to largely reduce computational time of optimization problems with high computational effort. It was therefore concluded that the modification could be a promising addition to the algorithm.

Finally, the possible use of a continuous control space was examined. The control space was analyzed and concluded to have multiple minima, hence justifying the implementation of a B&B search. The natural extension

function of the objective function was computed using interval arithmetics, to calculate lower bounds of a subset. Three alternative search methods were developed. Multi-stage bisection adheres to the B&B implementation used thus far, by maintaining the principle of a cumulative objective value throughout the time stages. Each time stage has its own control space, which is bisected repeatedly until a certain accuracy degree ϵ. Multi-stage multisection applies the same principle as multi-stage bisection, but bisects all edges of a control space, rather than just the longest edge. Finally, the single-stage bisection performs a regular B&B search, by bisecting the whole control space at once. After implementing the methods in a programmed model, all three were concluded to have an exponentially increasing computational time at increasing prediction as well as control horizon. The single-stage bisection was deemed the most suitable method, since it could handle the same PH and CH values with lower computational times. This leads to the conclusion the B&B algorithm from literature does not show any advantages when applied in continuous control space.

Overall, the main conclusions of the investigation is that B&B as an optimization method for MPC in a greenhouse is a promising search method if applied in discrete control space, its computational time can be reduced by simulating the last nodes of the control space in one step, and it does not show advantages in continuous control space.

9.3 Further Research

The foremost topic requiring further research in this topic is the selection rule that was applied when branching. In the models programmed for this report, the B&B algorithms were programmed to choose branches from left to right. However, it could prove beneficial to use a more elaborate selection method, for example, based on the transition cost, to choose the most promising branch. This could help to reach a lower upper bound faster, and hence prune many more sections of the tree. Another topic for further research is the application of the modification for node reduction to the continuous problem. Since the modification led to a reduction in computational time in the discrete control space, the node reduction could have the same effect on the computational time of the continuous search algorithms.

In the continuous search space it would also be of interest to investigate the bisection technique that is applied to the control spaces. In this report, the bisection was always performed on the largest edge of a control set, as this is a common technique. However, literature has shown that largest edge bisection does not always yield the smallest search tree [31]. It would thus be of interest which bisection technique would be optimal for this particular search algorithm, and whether a different technique could improve its applicability in continuous search space.

Finally, in order to determine the applicability of the B&B algorithm

in MPC with more certainty, it would be necessary to compare its performance to that of other optimization techniques. In particular dynamic programming would be of interest to compare with, as this method also optimizes problems over multiple stages or intervals. Finally, the B&B algorithm should be put in a broader context, to investigate its applicability to control problems in other fields.

References

[1] J. M. Aaslyng, J. B. Lund, N. Ehler, and E. Rosenqvist. Intelligrow: a greenhouse component-based climate control system. *Environmental Modelling & Software*, 18(7):657–666, 2003.

[2] J. Baldwin and B. Pilsworth. Dynamic programming for fuzzy systems with fuzzy environment. *Journal of mathematical analysis and applications*, 85(1):1–23, 1982.

[3] W. Baudoin, R. Nono-Womdim, N. Lutaladio, A. Hodder, N. Castilla, C. Leonardi, S. De Pascale, and M. Qaryouti. *Good agricultural practices for greenhouse vegetable crops: principles for mediterranean climate areas*. Number 217. Food and Agriculture Organization of the United Nations, 2013.

[4] M. Berenguel, F. Rodrıguez, F. Acién, and J. Garcıa. Model predictive control of ph in tubular photobioreactors. *Journal of Process Control*, 14(4):377–387, 2004.

[5] X. Blasco, M. Martínez, J. Herrero, C. Ramos, and J. Sanchis. Model-based predictive control of greenhouse climate for reducing energy and water consumption. *Computers and Electronics in Agriculture*, 55(1):49–70, 2007.

[6] N. Castilla and F. Nuez. Manejo del cultivo intensivo con suelo. *El cultivo de tomate. Madrid: Mundi-prensa*, pages 212–217, 1995.

[7] P. Ferreira and A. Ruano. Discrete model based greenhouse environmental control using the branch & bound algorithm. In *Proceedings of the 17th IFAC World Congress, Seoul, Korea*, volume 611, 2008.

[8] P. M. Ferreira and A. E. Ruano. Application of computational intelligence methods to greenhouse environmental modelling. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 3582–3589. IEEE, 2008.

[9] M. C. García-Martínez, S. Balasch, F. Alcón, and M. Fernández-Zamudio. Characterization of technological levels in mediterranean horticultural greenhouses. *Spanish Journal of Agricultural Research*, 8(3):509–525, 2010.

[10] P. E. Gill, W. Murray, and M. H. Wright. *Practical optimization*. Academic press, 1981.

[11] F. He, C. Ma, J. Zhang, and Y. Chen. Greenhouse air temperature and humidity prediction based on improved bp neural network and genetic algorithm. In *Advances in Neural Networks–ISNN 2007*, pages 973–980. Springer, 2007.

[12] E. M. T. Hendrix and B. G.-. Tóth. *Introduction to Nonlinear and Global Optimization*. Springer, New York, 2010.

[13] J. Herrero, X. Blasco, M. Martínez, C. Ramos, and J. Sanchis. Nonlinear robust identification of a greenhouse model using multi-objective evolutionary algorithms. *Biosystems Engineering*, 98(3):335–346, 2007.

[14] E. Heuvelink. Influence of day and night temperature on the growth of young tomato plants. *Scientia Horticulturae*, 38(1-2):11–22, 1989.

[15] E. Horowitz and S. Sahni. *Fundamentals of computer algorithms*. Computer Science Press, 1978.

[16] R. Johansson. System modelling and identification. *Automatica*, 3(33):477–478, 1997.

[17] H.-P. Liebig. Temperature integration by kohlrabi growth. In *Symposium on High Technology in Protected Cultivation 230*, pages 371–380, 1988.

[18] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[19] I. Mazhoud, K. Hadj-Hamou, J. Bigeon, and G. Remy. Interval-based global optimization in engineering using model reformulation and constraint propagation. *Engineering Applications of Artificial Intelligence*, 25(2):404–417, 2012.

[20] L. F. Mendonça, J. Sousa, and J. S. da Costa. Optimization problems in multivariable fuzzy predictive control. *International Journal of Approximate Reasoning*, 36(3):199–221, 2004.

[21] L. Miranda, I. Schuch, D. Dannehl, T. Rocksch, R. Salazar, and U. Schmidt. Using artificial neural networks to predict the climate in a greenhouse: First simulation results on a semi-closed system. In *II International Symposium on Horticulture in Europe 1099*, pages 137–144, 2012.

[22] L. Mitten. Branch-and-bound methods: General formulation and properties. *Operations Research*, 18(1):24–34, 1970.

[23] J. A. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[24] C. Onnen, R. Babuška, U. Kaymak, J. Sousa, H. Verbruggen, and R. Isermann. Genetic algorithms for optimization in predictive control. *Control Engineering Practice*, 5(10):1363–1372, 1997.

[25] A. Ramírez-Arias, F. Rodríguez, J. L. Guzmán, and M. Berenguel. Multiobjective hierarchical control architecture for greenhouse crop growth. *Automatica*, 48(3):490–498, 2012.

[26] F. Rodríguez. Interview with Francísco Rodríguez. Works at Universidad de Almería, Department of Computer Science, 1 2016.

[27] F. Rodríguez and M. Berenguel. Sistemas de control climático de invernaderos (ii): Sus efectos en el crecimiento del cultivo. *Riegos y drenajes XXI*, (116):28–37, 2001.

[28] F. Rodríguez, M. Berenguel, J. L. Guzmán, and A. Ramírez-Arias. *Modeling and Control of Greenhouse Crop Growth*. Springer, 2015.

[29] B. M. Roth and J. D. Mullen. *Decision making: Its logic and practice*. Rowman & Littlefield, 2002.

[30] J. A. Roubos, S. Mollov, R. Babuška, and H. B. Verbruggen. Fuzzy model-based predictive control using takagi–sugeno models. *International Journal of Approximate Reasoning*, 22(1):3–30, 1999.

[31] J. M. G. Salmerón, L. G. Casado, and E. M. T. Hendrix. On bisecting the unit simplex using various distance norms. *Informatica*, In press, 2016.

[32] R. Soeterboek. *Predictive Control: A Unified Approach*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1992.

[33] J. Sousa, R. Babuška, and H. Verbruggen. Fuzzy predictive control applied to an air-conditioning system. *Control Engineering Practice*, 5(10):1395–1406, 1997.

[34] M. Teruel and A. Cabrera. El sector de las frutas y hortalizas en España. Technical Report AL-294-2007, Cajamar Caja Rural, 2010.

[35] J. Vogelezang, J. De Hoog Jr, and N. Marissen. Effects of diurnal temperature strategies on carbohydrate content and flower quality of greenhouse roses. In *XXV International Horticultural Congress, Part 5: Culture Techniques with Special Emphasis on Environmental Implications 515*, pages 111–118, 1998.

Appendix A Appendix: Detailed Climatic Model

A.1 Extended Notation

Table 11: Variables and parameters of the climate model used in the B&B model.

Notation	Value	Unit	Definition
Actuators			
v_1		%	Windows opening actuator variable
v_2		%	Fog system actuator variable
v_3		%	Heating system actuator variable
State Variables			
s_1		°C	Temperature inside
s_2		%	Inside relative humidity
s_3		°C	Ground temperature
Disturbances			
d_1		°C	Outside temperature
d_2		%	Outside relative humidity
d_3		Wm^{-2}	Solar radiation outside
d_4		ms^{-1}	Wind speed
Coefficients			
α_1	0.0017		Constant for renewal volumetric flow

Continued on next page

Table 11 – continued from previous page

Notation	Value	Unit	Definition
α_2	240	m^2	Greenhouse surface area
α_3	130	m^2	Windows area
α_4	126594	$J°C^{-1}m^{-2}$	Thermal mass heat capacity
α_5	1003	$Jkg^{-1}°C^{-1}$	Air heat capacity
α_6	0.00435	$kg_{H_2O}s^{-1}$	Maximum water rate of fog system
α_7	0.0005	m^3s^{-1}	Losses of renewal air flow
α_8	0.0368	ms^{-1}	Boundary layer conductance
α_9	0.011	ms^{-1}	Maximum stomatal conductance
α_{10}	0.00435	ms^{-1}	Minimum stomatal conductance
α_{11}	8.4	$Wm^{-1}K^{-1}$	Conductivity coeff. ofair & thermal mass
α_{12}	0.52		Extinguishing coeff. of radiation
α_{13}	7.8685	$Wm^{-1}K^{-1}$	Conductivity coeff. of thermal mass & ground
α_{14}	0.796	$m^2_{leaves}m^{-2}_{ground}$	Leaves area index
α_{15}	17.907		Loss coeff. of conduction & convection
α_{16}	98.1	kPa	Atmospheric pressure
α_{17}	18.833	°C	Ground temperature at reference depth
α_{18}	850	m^3	Greenhouse volume
α_{19}	5000	W	Maximum power of heating system
α_{20}	6	m	Reference depth
α_{21}	0.0463		Rate of absorbed heat by thermal mass
α_{22}	12	°	Maximum window angle
α_{23}	0.066	$kPa°C^{-1}$	Psychometric constant
α_{24}	1.25	$kg_{air}m^{-3}$	Air density

Continued on next page

Table 11 – continued from previous page

Notation	Value	Unit	Definition
α_{25}	0.418		Transmission coeff. of greenhouse
α_{26}		Wm^{-2}	Solar radiation absorbed by crop
Functions			
$f_1(Temp)$		kPa	Saturation pressure
$f_2(Temp, Rel.H)$		$\mathrm{kg_{H_2O}kg_{air}^{-1}}$	Relative to absolute humidity
$f_3(Temp, Abs.H)$		$\mathrm{kg_{H_2O}kg_{air}^{-1}}$	Absolute to relative humidity
Dependent Variables			
Energy Flows			
$Q_1(v_3)$		W	Energy from heating system
$Q_2(d_3)$		W	Solar energy supplied to air volume
$Q_3(s_1, d_1)$		W	Energy exchange by conduction & convection
$Q_4(s_1, s_2)$		W	Energy loss of crop evapotranspiration
$Q_5(v_1, s_1, d_1)$		W	Energy exchange by window ventilation
$Q_6(v_2, s_1)$		W	Energy loss by nebulization
$Q_7(s_1, s_3)$		W	Energy exchange with thermal mass
$Q_8(d_3)$		W	Energy stored by thermal mass
$Q_9(s_3)$		W	Energy loss through the ground
Vapour Gas Flows			
$G_1(v_2)$		$\mathrm{kg_{H_2O}s^{-1}}$	Water rate of fog system
$G_2(v_1, s_1, s_2, d_2)$		$\mathrm{kg_{H_2O}s^{-1}}$	Water rate in the air renewal flow
$G_3(s_1, s_2)$		$\mathrm{kg_{H_2O}s^{-1}}$	Crop evapotranspiration

Continued on next page

Table 11 – continued from previous page

Notation	Value	Unit	Definition
Intermediate Values			
$F_1(v_1)$		$^\circ$	Opening window angle
$F_2(v_1)$		$\mathrm{m^3 s^{-1}}$	Renewal air flow
$F_3(s_1)$		kPa	Saturation pressure inside the greenhouse
$F_4(s_1, s_2)$		$\mathrm{kg_{H_2O} kg_{air}^{-1}}$	Inside absolute humidity
$F_5(s_1, d_2)$		$\mathrm{kg_{H_2O} kg_{air}^{-1}}$	Absolute humidity outside
$F_6(s_1)$		$\mathrm{kg_{H_2O} kg_{air}^{-1}}$	Absolute humidity at saturation
$F_7(s_1, s_2)$			Air saturation coefficient
$F_8(s_1)$		$\mathrm{kPa^\circ C^{-1}}$	Slope of water vapour saturation
$F_9(s_1, s_2)$		kPa	Air water vapour deficit
$F_{10}(s_1)$		$\mathrm{J kg^{-1}}$	Latent heat of vaporization
$F_{11}(s_1, s_2)$		$\mathrm{kPa^{-1}}$	
$F_{12}(s_1, s_2)$		$\mathrm{m s^{-1}}$	Stomatal conductance

A.2 Complementary Climatic Model Equations

Opening window angle:

$$F_1(v_1) = \frac{v_1}{100}\alpha_{22} \tag{A.1}$$

Water rate of fog system:

$$G_1(v_2) = \frac{v_2}{100}\alpha_6 \tag{A.2}$$

Energy from heating system:

$$Q_1(v_3) = \frac{v_3}{100}\alpha_{19} \tag{A.3}$$

Water rate in the air renewal flow:

$$G_2(v_1, s_1, s_2, d_2) = \alpha_{24} F_2(v_1)(F_5(s_1, d_2) - F_4(s_1, s_2)) \tag{A.4}$$

Renewal air flow:

$$F_2(v_1) = \alpha_3 d_4(\alpha_1 F_1(v_1) + \alpha_7) \tag{A.5}$$

Saturation pressure inside the greenhouse:

$$F_3(s_1) = f_1(s_1) \tag{A.6}$$

Absolute humidity in the greenhouse:

$$F_4(s_1, s_2) = f_2(s_1, s_2) \tag{A.7}$$

Absolute humidity outside:

$$F_5(s_1, d_2) = f_2(s_1, d_2) \tag{A.8}$$

Absolute humidity at saturation:

$$F_6(s_1) = f_2(s_1, 100) \tag{A.9}$$

Air saturation coefficient:

$$F_7(s_1, s_2) = \begin{cases} 1, & F_4(s_1, s_2) < F_6(s_1) \\ 0, & F_4(s_1, s_2) = F_6(s_1) \end{cases} \tag{A.10}$$

Crop evapotranspiration:

$$G_3(s_1, s_2) = \frac{\alpha_2(F_8(s_1)\alpha_{26} + 2\alpha_{14}\alpha_{24}\alpha_5 F_9(s_1, s_2)\alpha_8)}{(F_8(s_1) + \alpha_{23}(1 + (\alpha_8/F_{12}(s_1, s_2))))F_{10}(s_1)} \tag{A.11}$$

$$F_8(s_1) = f_1(s_1 + 0.5) - f_1(s_1 - 0.5) \tag{A.12}$$

$$\alpha_{26} = (1 - e^{\alpha_{12}\alpha_{14}})\alpha_{25}d_3 \tag{A.13}$$

$$F_9(s_1, s_2) = F_3(s_1)\left(1 - \frac{s_2}{100}\right) \tag{A.14}$$

$$F_{10}(s_1) = (3.1468 - 0.002365(s_1 + 273)) \times 10^6 \tag{A.15}$$

$$F_{11}(s_1, s_2) = \begin{cases} \frac{0.39}{0.029 + F_9(s_1, s_2)}, & F_9(s_1, s_2) \geq 0.361 \\ 1, & F_9(s_1, s_2) < 0.361 \end{cases} \tag{A.16}$$

$$F_{12}(s_1, s_2) = \alpha_{10} + (\alpha_9 - \alpha_{10})\left[1 - exp\left(-\frac{\alpha_{25}d_3}{160}\right)\right] F_{11}(s_1, s_2) \tag{A.17}$$

$$\tag{A.18}$$

Solar energy supplied to air volume:

$$Q_2(d_3) = \alpha_2\alpha_{25}d_3 \tag{A.19}$$

Energy exchange by conduction and convection phenomena:

$$Q_3(s_1, d_1) = \alpha_2\alpha_{15}(s_1 - d_1) \tag{A.20}$$

Energy loss due to crop evapotranspiration:

$$Q_4(s_1, s_2) = F_{10}(s_1)G_3(s_1, s_2) \tag{A.21}$$

Energy exchange due to window ventilation:

$$Q_5(v_1, s_1, d_1) = \alpha_{24}\alpha_5 F_2(v_1)(s_1 - d_1) \tag{A.22}$$

Energy loss by nebulization:

$$Q_6(v_2, s_1) = F_{10}(s_1)G_1(v_2) \tag{A.23}$$

Energy exchange between thermal mass and inside air:

$$Q_7(s_1, s_3) = \alpha_2\alpha_{11}(s_3 - s_1) \tag{A.24}$$

Energy stored by the thermal mass during the day:

$$Q_8(d_3) = \alpha_{21}Q_2(d_3) \tag{A.25}$$

Energy loss through ground:

$$Q_9(s_3) = \alpha_2\alpha_{13}\left(\frac{s_3 - \alpha_{17}}{\alpha_{20}}\right) \tag{A.26}$$

Functions Saturation pressure:

$$f_1(Temp) = 0.61(1 + 1.414sin(5.82e^{-3}s_1emp))^{8.827} \tag{A.27}$$

Relative to absolute humidity:

$$f_2(Temp, Rel.H) = \frac{Rel.H0.611F_3(s_1)}{100\alpha_{16}} \tag{A.28}$$

Absolute to relative humidity:

$$f_3(Temp, Abs.H) = \begin{cases} 100, & f_4(Temp, Abs.H) > 100 \\ f_4(Temp, Abs.H), & f_4(Temp, Abs.H) \leq 100 \end{cases} \tag{A.29}$$

$$f_4(Temp, Abs.H) = \frac{(Abs.H)100\alpha_{16}}{0.611f_1(Temp)} \tag{A.30}$$