

Institute of Architecture of Application Systems

University of Stuttgart  
Universitätsstraße 38  
D-70569 Stuttgart

Bachelorarbeit

# **HTN Planning Under Uncertainty for Sustainable Buildings**

Anna-Maria Halacheva

**Course of Study:** Softwaretechnik  
**Examiner:** Prof. Dr. Marco Aiello  
**Supervisor:** Ebaa Alnazer, M.Sc.

**Commenced:** April 1, 2022  
**Completed:** October 4, 2022



## **Abstract**

Smart buildings are buildings equipped with numerous sensors and actuators, allowing them to evaluate the state of the environment in or around them and react to it or alter it. Accordingly, one could use this available technology in order to optimize the operation of the buildings in terms of sustainability. One option to approach this challenge is through Hierarchical Task Network (HTN) planning, which allows us to analyze the current state of the environment and create a plan to be automatically executed by the actuators, achieving a certain goal. In this study, we systematically analyze the domain of smart buildings and identify strategies with which we can optimize the energy efficiency. We put primary focus on load balancing and HVAC control techniques and develop a domain model for HTN planning concentrating on them. Special attention is paid to usability and ensuring that the model satisfies the requirements of the occupants. Furthermore, we explore the question of how to represent the uncertain nature of the domain in its model. We choose to realize an approach based on cost-variable operators. For this, we use operators whose cost is non-deterministic. We implement our model and evaluate the implementation in terms of quality and quantity. Our solution proves to lead to a significant decrease in electricity bills, making its usage not only beneficial for the environment but also for its users.



## Kurzfassung

Smart Buildings sind Gebäude, die mit zahlreichen Sensoren und Aktuatoren ausgestattet sind, die es ihnen ermöglichen, den Zustand ihrer Umgebung zu bewerten und darauf zu reagieren. Dementsprechend entsteht die Idee, mit Smart Buildings den Wohnungssektor in Sachen Nachhaltigkeit zu optimieren. Eine Möglichkeit, dieser Herausforderung zu begegnen, ist das Hierarchical Task Network (HTN) Planning. Es ermöglicht uns, den aktuellen Zustand der Umgebung zu analysieren und einen Plan zu erstellen, der automatisch von den Aktuatoren ausgeführt wird, um ein bestimmtes Ziel zu erreichen. In dieser Studie analysieren wir systematisch den Bereich Smart Buildings und identifizieren Strategien, mit denen wir deren Energieeffizienz optimieren können. Wir legen den Fokus auf Lastausgleichs-, Heiz- und Kühlsteuerungstechniken und entwickeln ein Domänenmodell, das sich darauf konzentriert. Dabei wird besonderer Wert auf Benutzerfreundlichkeit gelegt und darauf geachtet, dass das Modell den Anforderungen der Bewohner entspricht. Darüber hinaus beschäftigen wir uns mit der Frage, wie die probabilistische Natur der Domäne im Modell abgebildet werden kann. Wir entscheiden uns für einen Ansatz, der auf kostenvariablen Operatoren basiert. Dazu verwenden wir Operatoren, deren Kosten nicht deterministisch sind. Wir setzen unser Modell um und bewerten die Umsetzung qualitativ und quantitativ. Unsere Lösung führt nachweislich zu einer erheblichen Senkung der Stromrechnung, wodurch ihre Nutzung nicht nur für die Umwelt, sondern auch für die Benutzer von Vorteil ist.



# Contents

<b>1</b>	<b>Introduction</b>	<b>17</b>
1.1	Motivation . . . . .	17
1.2	Objective . . . . .	19
1.3	Outline . . . . .	19
<b>2</b>	<b>Background Information</b>	<b>21</b>
2.1	Automated Planning . . . . .	21
2.2	HTN Planning . . . . .	23
2.3	Smart Buildings and Smart Grid . . . . .	26
2.4	Uncertainty in AI Planning and HTN Planning . . . . .	28
<b>3</b>	<b>Sustainable Solutions with HTN Planning</b>	<b>31</b>
3.1	Performing Knowledge Acquisition . . . . .	31
3.2	Addressing Complex Problem Requirements . . . . .	40
3.3	Modelling the Domain . . . . .	44
<b>4</b>	<b>Implementation</b>	<b>55</b>
4.1	Planner . . . . .	55
4.2	Domain Model . . . . .	55
4.3	Input and Output . . . . .	57
<b>5</b>	<b>Evaluation</b>	<b>61</b>
5.1	Evaluation Framework . . . . .	61
5.2	Results . . . . .	61
<b>6</b>	<b>Related Work</b>	<b>67</b>
<b>7</b>	<b>Conclusion and Future Work</b>	<b>69</b>
	<b>Bibliography</b>	<b>71</b>
<b>A</b>	<b>Predicates for Defining Problem Instances</b>	<b>75</b>
<b>B</b>	<b>Structure of Domain Model</b>	<b>77</b>
<b>C</b>	<b>Problem Instances for Qualitative Evaluation and Produced Plans</b>	<b>81</b>
C.1	Highest Automation Level . . . . .	81
C.2	Lowest Automation Level . . . . .	89





## List of Figures

2.1	Approach for solving a problem with automated planning . . . . .	21
2.2	Different types of buildings and their evolution. Adopted from [BMB14] . . . . .	27
3.1	Conceptual framework for knowledge acquisition. Adapted from [GA16] . . . . .	33
3.2	Electricity demand through the day. Adopted from [Hod] . . . . .	36
3.3	The idea behind load shifting. Adopted from [HT] . . . . .	37
3.4	Example for size reduction of task networks decomposing compound tasks . . . . .	48
3.5	Part of the domain model which represents Algorithm 3.2 and Algorithm 3.3 . . . . .	49
4.1	Implementing variable operator costs . . . . .	56
5.1	An overview of the planning time for an initial state with a varying number of providers and varying types of appliances. . . . .	62
5.2	An overview of the planning time for an initial state with a varying number of providers and fixed types of appliances for every planning instance. . . . .	63
5.3	An overview of the planning time for an initial state with a varying number of appliances to schedule. . . . .	64
5.4	An exponential regression over the data for planning time for an initial state with a varying number of appliances to schedule. Extreme values have been removed. . . . .	64



## List of Tables

3.1	Acquired knowledge for the domain of sustainable smart buildings. . . . .	34
3.2	Calculated temperature set points for the three different tolerance levels and the set point interval $[T_{min}, T_{max}] = [68, 74]$ for both pre-cooling and pre-floating using Algorithm 3.2 for pre-cooling and Algorithm 3.3 for pre-floating. . . . .	45



## List of Listings

4.1	CalcCostSolarUsage.java . . . . .	58
-----	-----------------------------------	----



## List of Algorithms

3.1	Electricity Source Determination . . . . .	38
3.2	Set Point Calculation for Cooling [AEA12] . . . . .	43
3.3	Set Point Calculation for Heating . . . . .	44
3.4	Part of an Algorithm with Conditional Statements . . . . .	47





# 1 Introduction

## 1.1 Motivation

According to the European Statistical Office, the housing sector accounts for 28% of the energy usage in the European Union for 2020, making it the second-largest consumer after the transportation sector (with 28.4%) [Offb]. This percentage is expected to grow further in the years to come, as Reyna et al. [RC17] even predict an electricity demand increase of 41 – 87% between 2020 and 2060. Additionally, taking into consideration the effects of the COVID-19 pandemic and how it affected the industry, those estimates might even be too low. The reason for this is the adoption of new workspace practices by many companies, shifting from using commercial buildings and offices, towards allowing and even encouraging remote and hybrid working [FWR; MZH]. It is expected that by 2025, 40.7 million USA citizens will be working remotely, accounting for an increase of 87%, compared to the levels until 2019 [FWR]. Such a change would burden not only the environment and the energy infrastructure, but also the residents. The reason for this is the observed increase in the electricity bill by 167% on average in the cold months when switching from going to work to working remotely in home-office [Che20]. Those projections imply the urgent need to make the housing sector more sustainable in regard to energy management and usage.

Analysis of energy sources in housing shows that the second most used energy source in buildings in the EU is electricity, which in Malta is even the main source, accounting for 70% of energy [Com]. The current importance of electricity and the projected future decline in demand for natural gas [Age], the main energy source in the EU, speak for the need for solutions specifically aimed at optimizing household electricity consumption.

The smart grid is a concept for an improved electrical grid, allowing dynamic pricing from competing providers as well as better integration of renewable energy sources. In this way, it also enables the users to purchase electricity at lower prices. The dynamic pricing strategy would also lead to shifting part of the demand from on-peak to off-peak hours due to the cheaper prices (*load shifting*). As an effect, more renewable energy would be used and the load on the grid would be distributed more evenly, leading to a more sustainable operation [Mat19]. However, optimal use of the smart grid would require constant monitoring or day-ahead analysis of electricity prices, active planning of the electricity purchases, and the working times of the household appliances. All of this has to be done in accordance with the analysis results and has to be followed by the execution of the created plans. The whole process demands very strong user involvement, and even then it can still be infeasible for humans due to the large amounts of data. Therefore, utilizing the smart grid in the explained way requires automation.

Automation at such level, including analysis and planning phases, as well as the automatic execution of the computed plan, can be realized in *smart buildings*. A smart building is a building, which is equipped with different sensors and actuators, which allow its management systems to make informed decisions to control the environment, e.g., turning lighting on and off. If the system

incorporates *automated planning* functionality (incorporates a *planner*), it can also plan how to control the environment so that it satisfies some predefined objectives, e.g., energy saving and minimizing the electricity cost. In this case, the planner searches through the search space of possible actions to undertake and constructs a *plan*, a sequence of actions, whose execution should fulfill a predefined goal. The automated planning requires the usage of Artificial Intelligence (AI) in order to be able to reason over the knowledge base [GNT04a].

*Hierarchical Task Network planning (HTN planning)* is a technique for AI planning, which is exceptionally suitable for planning in the building domain due to its allowance for modularity. The housing sector, as most fields from real life, is characterized by various levels of granularity and constant changes. Since HTN planning supports both primitive and compound tasks, we can realistically recreate those different granularity levels into the domain model. Furthermore, the hierarchy ensures high flexibility of the model. We can alter isolated methods or operators without drastically affecting the overall structure [GNN+17].

An additional characteristic of this domain, besides its granular structure, is its dependency on uncertain factors, e.g., on the weather forecast. According to it, a smart home often decides whether to cool/heat a house or not and estimates how much solar energy it will have at its disposal. As a result, a wrong weather forecast may lead to the home having less energy than estimated. This could lead to additional unplanned purchases from the smart grid and differing costs from the estimated ones. Accordingly, every plan, constructed with dependence on such uncertain factors, for example, on the weather forecast, is of probabilistic nature. A “probabilistic plan” can be defined in many ways, e.g., a plan whose real execution cost differs from the estimated one, as in the weather forecast example. Another definition could be a plan for which we are not completely sure whether it will indeed be executable in the real situation. Therefore, factors of uncertainty in a domain can have huge implications on the plans for it. Not identifying the possible implications and, consequently, failing to adapt the domain model to counteract them, will often cause problems. For example, we could assume that the cost of executing a created plan is always deterministic, but when we start using the solution, we would find out that, because of factors of uncertainty, the cost is 10 times higher.

However, HTN planning where the uncertainty is incorporated in the domain model is a concept that has been theorized [AGA22; GA16], but not really implemented in reality. The more typical approach for planning under uncertainty is the usage of probabilistic planning [CC07; Ric17]. Therefore, the representation of the probabilistic nature of the domain in its model is of particular interest, and thus addressed later in this study. Additionally, most scientific works addressing the problem of making smart buildings more sustainable through AI or HTN planning usually focus either on *global* benefits or on *local* ones [AEA12; KJU+13]. “Global” means prioritizing advantages on a larger scale, e.g., reducing electricity usage on a national level, by making some “local” sacrifices [KJU+13], e.g., not using air conditioning in summer. “Local” regards the reversed direction—big advantages for the occupants, which don’t make a big impact on the larger scale. Accordingly, solutions combining both approaches are desirable. Consequently, this study aims at developing a strategy ensuring both types of gains.

## 1.2 Objective

The aim of this bachelor's thesis is to develop an approach to optimizing energy consumption in smart buildings that uses HTN planning. Firstly, we systematically acquire knowledge about the housing sector and smart buildings in order to identify areas of the domain whose improvement will lead to their more sustainable operation. As a next step, we model the smart home domain for HTN planning by mapping the acquired knowledge to HTN planning constructs. Due to the uncertain nature of the domain to be modeled, we also deal with the issue of determining different sources of uncertainty and finding a way to represent their impact in the planning process. Finally, we implement the model and evaluate the developed solution. For this, we test it on various problem instances and evaluate its effectiveness against the defined objectives by creating and applying an evaluation framework.

For this thesis, we are focusing on smart homes, as smart buildings from the residential buildings sector. Even though office buildings account for a large proportion of the energy usage, they often remain a territory of various interests, meaning that an additional strategy for energy management would be necessary. Such strategy would address questions such as: "How is the storage space in a BESS (Battery Energy Storage System) divided among the different owners and tenants? Can they use the unused storage space of others? If yes, according to what rules?", etc. The answers to those questions may require different implementations of the domain model. Since the development of such a strategy is not within the scope of this work, we have decided to focus only on residential housing and not on the entire building sector. However, our domain model, the used algorithms, and the example planning problems can be easily adapted for the commercial sector, to meet the needs of office buildings, once the aforementioned strategy has been determined.

## 1.3 Outline

The remainder of this thesis is organized as follows. In Chapter 2 we introduce the background information for the study while focusing on the core concepts of AI and HTN planning. We then move on to describe our methodical approach for the research in Chapter 3. More specifically, firstly, in Section 3.1, we explain both the conceptual framework based on which we perform knowledge acquisition and the resulting findings. Then we address the more complex requirements of the domain in Section 3.2. Finally, we present and discuss how we model the domain in Section 3.3. Chapter 4 provides all details about the implementation of the developed model, while Chapter 5 offers an evaluation of the solution. Chapter 6 discusses related work. In Chapter 7, we provide a summary of the conducted research and the derived conclusions, and finally point out fields, which need further research.



## 2 Background Information

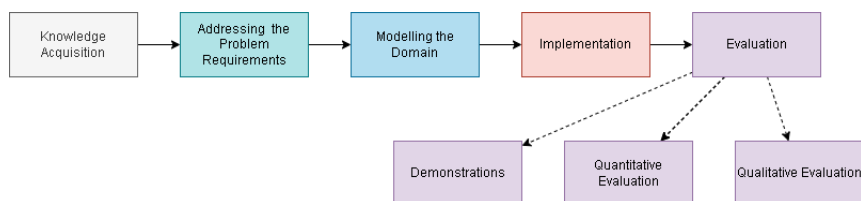
In this chapter, we introduce the preliminaries necessary for understanding this thesis. Firstly, we explain the concept of automated planning and afterwards present HTN planning as a specific planning technique. Then we examine the smart buildings and the smart grid domain and define the related terminology. Finally, we provide some insight into uncertainty in AI planning and HTN planning to be able to address the probabilistic nature of the domain later.

### 2.1 Automated Planning

Automated planning, also called *AI planning*, is a branch of artificial intelligence, dealing with the creation of plans which lead to the satisfaction of some pre-defined goal. In the planning context, a plan is a series of actions whose execution in a specific *initial state* would lead to achieving the chosen *goal*. The initial state is a description of the state of the environment from which the plan should be executed. The goal is defined differently for the different planning techniques. However, it usually gives the state of the environment (or a part of it) that should be observed after the execution of the plan. We must point out that this is not valid for the goal in HTN planning, whose definition is offered in Section 2.2. The plan (the series of actions) can be ordered or not, allowing for very diverse use cases of AI planning.

Solving a problem with automated planning is a process associated with multiple phases, as depicted in Figure 2.1. The first step is always domain knowledge acquisition, which can be followed by a thorough review of all complex domain requirements. Next are the modeling of the domain, its implementation, and evaluation. Finally, if the results of the evaluation are satisfactory, the constructed model and its implementation can be used to create plans for specific domain problem instances.

During the domain knowledge acquisition part, the developer follows a chosen conceptual framework to get acquainted with the specifics of the domain and the way everything in it functions (the domain “physics”). The second phase, addressing the problem requirements, is about identifying and understanding the more complex requirements of the domain. It is important to review them thoroughly since their satisfaction could complicate the model a lot or could require novel solutions.



**Figure 2.1:** Approach for solving a problem with automated planning

The acquired knowledge is then used in the domain modeling part to build the domain model, which includes the different domain elements, their states, and the actions which can change those states. Of course, the model is created at a certain level of abstraction, omitting irrelevant details, which would only increase the complexity. Both the “research” phase and the modeling are critical for the whole process since every plan for every problem instance will be constructed entirely according to the knowledge in the domain model. Therefore, any misrepresentations, inexactness, or errors in the model can lead to wrong plans, whose execution would not only not bring the wanted results, but might even be harmful. This is also the reason the evaluation part of the planning process should never be skipped. A comprehensive evaluation covers various aspects of the testee, therefore we differentiate between multiple types of evaluation. According to Georgievski et al. [GA16], we can distinguish *demonstrations*, as well as *quantitative*, *qualitative*, and *usability evaluation*. For the demonstrations we often use scenarios, meaning that we construct the problem instance to represent a situation, for which we know what a good plan would be. This form of evaluation allows us to examine the credibility and effectiveness of the constructed plans. With the quantitative evaluation, we review the performance of the planner, e.g., we could analyze the scalability of our solution with respect to the size of the initial state. The qualitative evaluation looks at the quality of the created plans, for example, by comparing them to plans created with other planners/approaches. There is also usability evaluation. It measures how easy a solution is to use and whether, and to what extent, it covers the needs and expectations of the user. However, it is used relatively rarely, as pointed out by Georgievski et al. [GA16]. We are not conducting a usability evaluation for this study either, as depicted in Figure 2.1.

As it can be seen, solving problems with automated planning is a computation-intensive process. Therefore, it is not recommended to be used for areas, which require very basic reasoning and can be easily grasped by humans.

The creation of a plan to solve a specific problem in the domain is the task of a so-called planner. A planner’s input consists of a planning domain model and a planning problem. For this input, it outputs a plan which solves the problem instance according to the “physics” of the domain. Those “physics” are described in the given domain model. In order for this to be possible, the problem instance should contain an initial state and a goal. The planner searches in the search space for actions whose performance is possible in the specific state and would also lead to achieving the goal(s) or alter the state of the planning domain, enabling the performance of other actions which facilitate reaching the goal(s). As already mentioned, the planner acts entirely according to the information in the domain model and considers only the actions described there in the way described there.

Since usually multiple plans can be constructed for the same initial state and goals, we can define objectives, which guide the planning process to create an “optimal” plan, according to some criteria. Such objectives can be to decrease the planning time as much as possible, to construct a plan, consisting of a minimal number of actions, or to minimize the cost. This objective is then incorporated into the search algorithm of the planner.

## 2.2 HTN Planning

*Hierarchical Task Network (HTN) planning* is a technique for AI planning, and, accordingly, works with the same elements we described in the previous section – domain model, containing the obtained domain knowledge, and problem instances, consisting of an initial state and one or more goals. The goals in HTN planning are given in a so-called *initial task network* [GA15]. However, since the formal definitions for those constructs show some variations across different scientific works, in this research study we define and use those constructs according to their meaning in [GNT04b]. In the remainder of this section, we explain those constructs and their usage.

A *task-network* is a *hierarchy of tasks*, each of which can be *primitive* or *compound*, where the compound tasks can be broken down in other primitive or compound tasks. The initial task network includes the tasks which we want to accomplish (*goal tasks*). Every sequence of primitive tasks which represent a decomposition of the initial task network, is called a *plan* for the given problem instance. Respectively, by executing the created plan, we execute the tasks of which it comprises.

The decision whether a certain compound task can be decomposed in a specific way depends both on the possible decompositions, described in the domain model, and on the current domain state. The states in HTN planning are conjunctions of *ground predicates*. A predicate consists, as defined by [GA15], of a name of the *predicate* and an ordered sequence of *terms*  $\tau_1, \dots, \tau_n$ , which can be either *variables* or *constants*. The predicate can take only two values – true or false, however, the value can vary in different states. For example, let  $\langle on, (appliance) \rangle$  be a predicate describing whether the appliance the variable *appliance* will be bound to is on. Let the predicate  $\langle on, (tv) \rangle$  be true in state  $S_1$ , meaning that the TV is on. If we execute the task of switching *tv* off in  $S_1$ , then in  $S_2$   $\langle on, (tv) \rangle$  will hold the value false. Additionally, we distinguish between two types of predicates regarding their terms – non-ground and ground. The latter has no variables, but only constants, e.g.,  $\langle on, (tv) \rangle$ , but not  $\langle on, (appliance) \rangle$ , given that we've defined *appliance* as a variable and *tv* as a constant.

The domain model is a set of methods and operators. An operator is a primitive executable task together with the *preconditions* for its execution and its effects on the domain state. A method, on the other hand, refers to a compound task which can be decomposed with this method, and also includes the preconditions that need to be satisfied for the method to be applicable, as well as the task network representing the decomposition. The preconditions of an operator or method are a conjunction of predicates, all of which have to evaluate to true in a certain state  $S$ , so that the corresponding operator/method can be applied in this state  $S$ . An explanation when a predicate evaluates to true or false in a given state is provided later in this section.

As mentioned in Section 2.1, the actual automated planning is a responsibility of a planner. There exist multiple types of AI planners for the different planning techniques [GNN+17], meaning that hierarchical planning requires the usage of an HTN planner. In this work, we use *state-based* planning and, accordingly, state-based HTN planners. Such a planner takes the domain model, initial domain state, and goals, here in the form of an initial task network (itn). Then, it starts performing *task decomposition* on the itn until there are no more compound tasks [GA15]. The planner chooses a task  $t$  from the current task network  $tn$ . If  $t$  is primitive and its corresponding operator is applicable in the current state of the domain, the operator is applied, and the state is updated according to its effects. If  $t$  is a compound task, the planner identifies methods that provide decompositions for  $t$ . One strategy for the planner to decide which of those methods to use can be

to do non-deterministically and then check whether the domain's state satisfies its preconditions. If so, the compound task is broken down into the task network offered by the method. If no complete decomposition can be produced, this means that no plan exists to achieve the tasks from the initial task network, when starting from the given initial state and acting according to the domain model.

In the remainder of this section, we provide formal definitions of different HTN planning constructs and approaches, which are according to [GNT04b].

### Definition 2.2.1 (Predicate)

A predicate is defined as:  $p = \langle \text{name}(p), \text{terms}(p) \rangle$ , where:

- $\text{name}(p)$  is the name of the predicate
- $\text{terms}(p) = \langle \tau_1, \dots, \tau_n \rangle$  is an ordered sequence of terms, for which applies:  $\forall i \in \{1, \dots, n\} : \tau_i \in C \cup V$ .  $C$  represents the set of constants and  $V$  the set of variables.

A predicate can evaluate only to the values true and false.

### Definition 2.2.2 (Ground Predicate)

A ground predicate is predicate  $p = \langle \text{name}(p), \text{terms}(p) \rangle$ , where  $\text{terms}(p) = \langle \tau_1, \dots, \tau_n \rangle$  and  $\forall i \in \{1, \dots, n\} : \tau_i \in C$

### Definition 2.2.3 (Primitive Task)

A primitive task is defined as:  $pt = \langle \text{name}(pt), \text{terms}(pt) \rangle$ , where:

- $\text{name}(pt)$  is the name of the primitive task,
- $\text{terms}(pt) = \langle \tau_1, \dots, \tau_n \rangle$  is an ordered sequence of terms.

### Definition 2.2.4 (Compound Task)

A compound task is defined as:  $ct = \langle \text{name}(ct), \text{terms}(ct) \rangle$ , where the different parts have the same meaning as in primitive tasks.

### Definition 2.2.5 (Task Network)

A task network is defined as:  $tn = \langle T, Co \rangle$ , where:

- $T = \{t_1, \dots, t_n\}$ , where  $\forall i \in [1, n] : t_i$  is a task, primitive or compound.
- $Co$  is a set of constraints, which apply to the tasks in  $T$ .

[GNT04b] describe different types of constraints, but we will only use the *precedence constraint*  $t_i < t_j$ , which defines the order of the tasks in the task network. The example given means that  $t_i$  precedes  $t_j$ .

### Definition 2.2.6 (State)

A state is defined as a set of ground predicates  $p_i$ :  $s = \{p_1, p_2, \dots, p_n\}$ . The order of the predicates can be random, since it is of no importance.

### Definition 2.2.7 (Predicate Evaluation)

We define the predicate evaluation function  $\text{Evaluate}(p, s) \in \{\text{true}, \text{false}\}$  where  $p$  is a predicate, and  $s$  a state.  $\text{Evaluate}(p, s) = \text{true}$ , iff  $p \in s$ , in this case we say that the predicate  $p$  applies in the state  $s$ .



**Definition 2.2.8 (Operator)**

An operator is defined as:  $o = \langle pt(o), pre(o), eff(o), cost(o) \rangle$ , where:

- $pt(o)$  is the primitive task the operator solves
- $pre(o)$  are the preconditions - a set of predicates, all of which should evaluate to true in the current state of the environment, in order to apply the operator  $o$ . Formally, we can formulate this as  $\forall p_i \in pre(o) : Evaluate(p_i, s) = true$ , where  $s$  is the current state of the environment,
- $eff(o) = \{eff^-(o), eff^+(o)\}$  are the effects of the operator. Let  $s$  be the current state of the environment, and  $s_1$  the state after the application of the operator. Then it applies:  $s_1 = (s \setminus eff^-(o)) \cup eff^+(o)$
- $cost(o)$  is the cost of applying the operator

**Definition 2.2.9 (Method)**

We denote methods as  $m = \langle ct(m), pre(m), tn(m) \rangle$ , where:

- $ct(m)$  is the compound task the method decomposes,
- $pre(m)$  are the preconditions, as defined for operators,
- $tn(m)$  is the task network resulting from the decomposition of the compound task  $ct(m)$  by the method  $m$ .

**Definition 2.2.10 (Planning Domain)**

A planning domain is defined as  $d = \langle O, M \rangle$ , where

- $O$  is a set of operators,
- $M$  is a set of methods.

**Definition 2.2.11 (Planning Problem)**

A planning problem is denoted as:  $P = \langle s_0, tn_0, d \rangle$ , where:

- $s_0$  is the initial state of the environment,
- $tn_0$  is the initial task network,
- $d$  is the domain model, according to which the solution is to be constructed.

**Definition 2.2.12 (Solution)**

A solution of a planning problem  $P = \langle s_0, tn_0, \langle O, M \rangle \rangle$  is a plan  $\pi = \langle o_1, \dots, o_n \rangle$ , which satisfies the following requirements:

- $\forall i \in [1, n] : o_i \in O$
- If  $tn_0$  is a primitive task network (includes only primitive tasks):

Let  $tn_0 = \langle T, Co \rangle, T = \{t_1, \dots, t_k\}$ . Let  $\langle t_1, \dots, t_k \rangle$  be the total ordering of the tasks in the initial task network according to the constraints in  $Co$ .  $\pi$  is the solution to  $P = \langle s_0, tn_0, \langle O, M \rangle \rangle$  iff  $\pi' = \langle o_2, \dots, o_n \rangle$  is the solution to  $P' = \langle s_0[o_1], tn_0 \setminus \{t_1\}, \langle O, M \rangle \rangle$ ,  $pt(o_1) = t_1$  and  $pre(o_1) \subseteq s_0$ .

- If  $tn_0$  is a compound task network (includes at least one compound task):

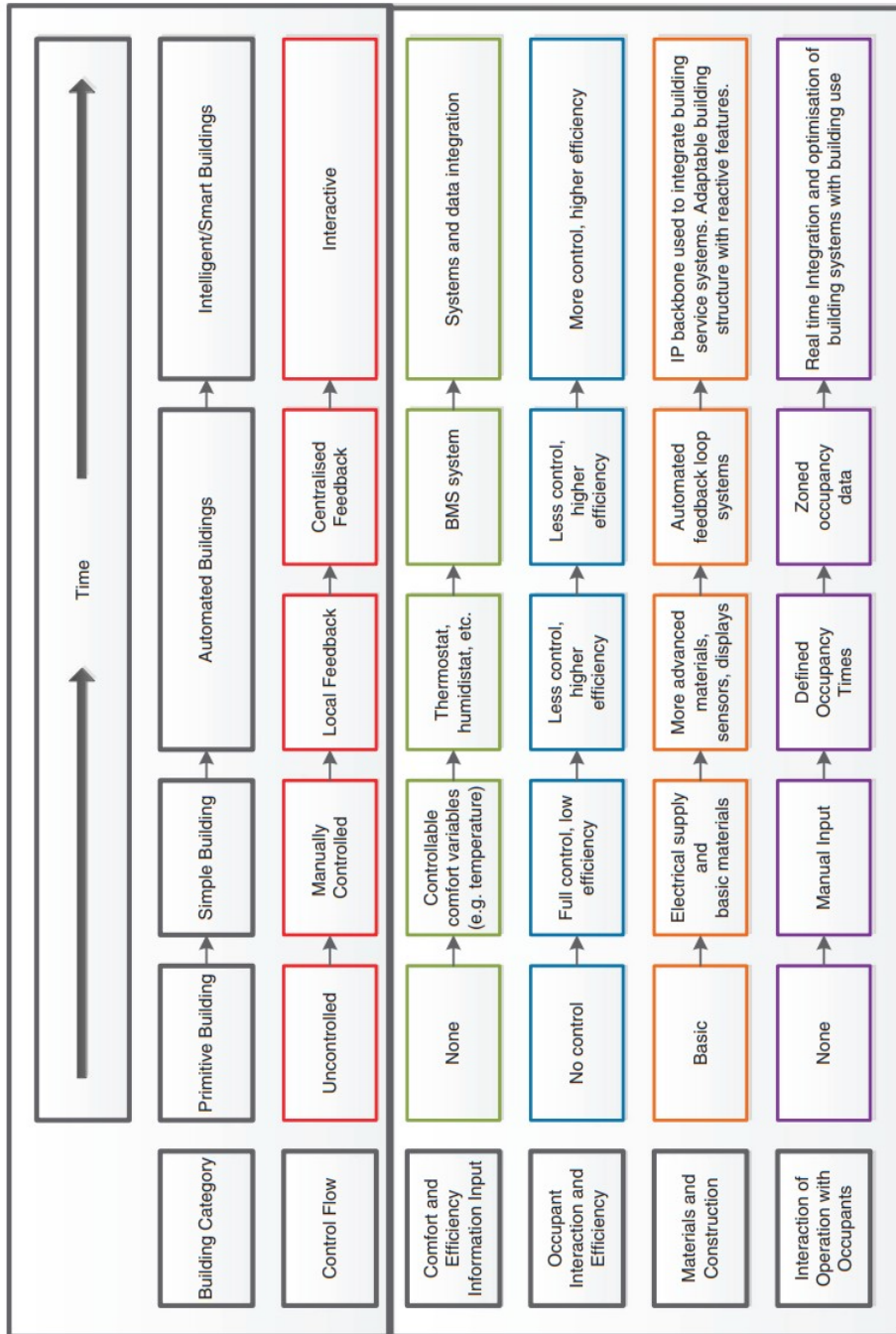
*There is a sequence of task decompositions that can be applied to  $tn_0$ , transforming it into a primitive task network  $tn_{0,p}$  such that  $\pi$  is a solution for  $P' = \langle s_0, tn_{0,p}, \langle O, M \rangle \rangle$ .*

## 2.3 Smart Buildings and Smart Grid

### 2.3.1 Smart Buildings

Even though various research studies and industry projects concentrate on smart buildings, there is no standardized definition of them [KFS+21]. However, the meaning in which smart building is used in every study determines the scope for which its findings are applicable. Some authors explain them as “automated buildings” [KFS+21], others as “intelligent” [MOG+10] and for some “smart” is its own category [BMB14; Hoy16]. Here we will be using the definitions provided by Hoy and Buckman et al. [BMB14; Hoy16]. They argue that smart buildings are developed from intelligent buildings, which, on their side, enhanced the concept of automated buildings. The key differences between a simple building, an automated one, and the smart one are shown in Figure 2.2 from Buckman et al. [BMB14]. While a primitive building is just a living space without any adjustment options, a simple building’s environment can be changed manually by a human. E.g., by pressing the switch for the lights to turn them on or changing the temperature by adjusting a radiator valve or opening a window. An automated building automates the activities that require to be executed mechanically by humans. For example, an automated building would have sensors controlling the lights and/or thermostats, which would turn the heating/cooling on and off **according to a schedule which is predefined by a human**. The ability to change the environment automatically is also a characteristic of intelligent buildings, however, they do not need exact directions about performing those instructions. They can observe the state of the environment, recognize when changes are necessary so that certain predefined goals remain satisfied, and then perform automatically the needed adjustments. This feature is characterized as supporting real-time reactions. For example, if a sensor for humidity estimates that the humidity in a room is above a certain threshold, it will turn on a dehumidifier and then turn it off, once the humidity level is normalized. The given example shows the *reactive* nature of the intelligent buildings – they can observe the state of the environment and, only by need, react to it. The smart building, on the other hand, is of *adaptive* nature. For example, the smart building can recognize that someone is taking a shower, which, according to the *reasoning logic* in the building’s system, could mean that the humidity level would later be increasing in the whole house. Even though no sensor has already detected too high humidity, the building would turn on a dehumidifier near the bathroom, so that the humidity level never gets too high. This adaptive nature also incorporates energy efficiency, as the building adapts its energy usage to the electrical grid, e.g., by scheduling devices for off-peak hours or buying electricity in advance and storing it.

HTN planning can be easily performed for the domain of smart buildings since they are equipped with numerous sensors and actuators. The sensors provide detailed information about the environment, which would allow the planner to receive a very realistic initial state of the domain, whereas the actuators would ensure the execution of the plan. The corresponding domain could include the different appliances and the grid as objects and the tasks would be, e.g., buying or selling electricity



**Figure 2.2:** Different types of buildings and their evolution.  
Adopted from [BMB14]

from/to the grid, storing this electricity, turning a device on or off, adjusting the thermostat, etc. Accordingly, we could plan the purchases and usages of electricity, and the operation of the devices a day ahead, so that we save as much energy as possible and also benefit financially.

### 2.3.2 Smart Grid

The smart buildings are connected to the concept of a smart electrical grid, which will be powering them. The current electric infrastructure is aging and needs modernization, in order to make it more sustainable and answer today's needs, e.g., in this highly electricity-dependent age outages can be disastrous. Additionally, in order for it to be more sustainable, it should, for example, provide the possibility to produce electricity mainly from renewable sources, but also should give people more opportunities to act more eco-friendly[LCL16]. With the traditional grid, its users can only decrease the amount of energy they use. However, they cannot demand to use energy from renewable sources or even decide to support providers of such energy by, e.g., paying a higher price for it. Additionally, currently, there is no way for the users to know in which hours they should restrain themselves from consuming electricity from the grid. Such a feature is very necessary since the demand in some hours is so high that it requires powering additional generators, which are usually using non-renewable resources. The smart grid addresses this issue by ensuring bidirectional communication between the grid and the users. The smart grid operates with dynamic pricing, offering different prices for electricity for different hours. In this way, buying energy at on-peak hours will cost more than buying at off-peak hours when the demand is low. Accordingly, the users would be motivated to decrease their electricity load during the peaks and shift it towards the off-peak hours (*load shifting*) [KJU+13].

Additionally, in the smart grid, we have multiple providers, meaning that the consumers can, e.g., choose to buy from a provider of electricity from renewable sources. It also allows the consumers to be providers and sell their electricity to the grid, if, e.g., they have solar panels which supply them with more energy than they need [KJU+13].

## 2.4 Uncertainty in AI Planning and HTN Planning

AI planning can be used for solving problems in various domains, however, in the real world, a domain is rarely completely deterministic. This means that in order to be able to model domains realistically, we have to be able to integrate uncertainty in them. During our research, we get acquainted with different strategies addressing this issue, e.g., using Partially Observable Markov Decision Processes [Ric17], variable probability of success or cost-variable operators [AGA22].

For our work, we decide to use the idea of cost-variable operators, as defined by Alnazer et al. [AGA22]. The reason for this decision is the intuitiveness of their suggestion. Their research suggests that the uncertainties in a domain affect the costs of actions, e.g., fuel or time, which become non-deterministic. Since every action has a cost, not being sure whether and up to what extent we would be able to execute an action, means we cannot be sure about the costs we will have at the end. For example, if we want to automatize farming with HTN planning we could have primitive tasks such as  $\langle \textit{plant}, (\textit{plant}, h) \rangle$ ,  $\langle \textit{harvest}, (\textit{plant}, h) \rangle$ ,  $\langle \textit{water}, (\textit{plant}, h) \rangle$ . When we define the corresponding operators, we would have to define their costs. However, the farming

domain is highly dependent on the weather, which is inherently uncertain. Accordingly, if we were to define the cost of an action as the financial expenses related to performing it, our actions would have costs that would be very wide-ranging. For example, if we decide to water plants when no rain is expected, the cost could be  $c_{normal}$ . However, if it starts raining after we have watered the plants, the cost for us will be way higher, since we have lost money without gaining any advantage. Thus, the question arises what cost should our operator for watering have, e.g., when the weather suggests that rain is possible but not very likely. This decision is of utmost relevance since some planners construct plans with the objective of cost-minimization. Accordingly, deterministic costs are likely to lead to suboptimal plans.

The proposal of Alnazer et al. [AGA22] is to make the costs of the operators variable. Precisely, this means modeling an operator cost as a probability distribution over all possible costs. Afterwards, we can take the expected value over the probability distribution as the cost for the operator. Of course, the construction of this probability distribution is far from trivial for most domains and should be planned carefully.



## 3 Sustainable Solutions with HTN Planning

In Section 2.1 we described the different phases in the process of developing a solution to a problem using automated planning. Those phases, as depicted in Figure 2.1, are also valid when using HTN planning as a technique for automated planning. Accordingly, we are following the same steps in our research study which addresses the issue of making smart buildings more sustainable.

The core of this work consists in identifying the areas of the domain which need optimization and are susceptible to it, and in constructing an optimization strategy. It also includes mapping this strategy to HTN constructs in order to create a domain model. Those tasks cover the first three phases of our method from Figure 2.1 - structured knowledge acquisition, addressing complex problem requirements, and the modelling of the domain. In the remainder of this chapter, we explain our approach to each of those steps and present the results from performing them. The implementation of the domain model, the fourth phase, is described in Chapter 4, and the evaluation of it - in Chapter 5.

### 3.1 Performing Knowledge Acquisition

The knowledge acquisition phase in our approach is about obtaining knowledge about the domain we want to model. As pointed out in Section 2.1, this step is of high importance for the quality of the solution we will develop. Therefore, it is required to perform this phase in a structured way, so as not to omit or misunderstand any relevant information [Lio92]. For our study, we chose to follow the conceptual framework for knowledge acquisition described by Georgievski and Aiello [GA16].

The used conceptual framework, as depicted in Figure 3.1, divides the knowledge about the domain into four categories, each of which has some sub-categories. Those are defined as follows:

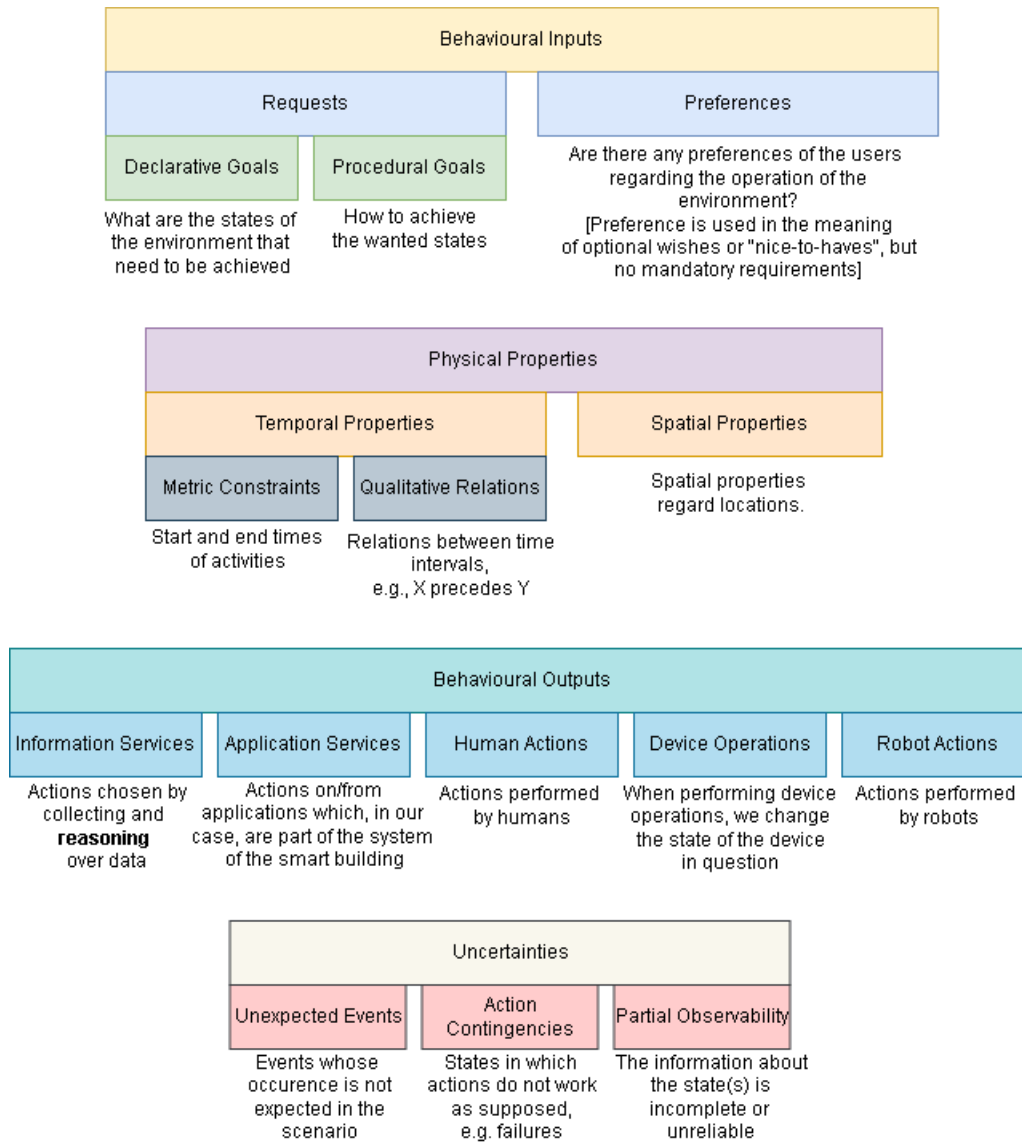
- *Behavioural inputs* include the expectations of the users from the environment. This is the category which gives us the information about the states of the environment that should be achieved through the execution of the plans created with our solution. The following sub-categories are included here:
  - *Requests* describe what could be wanted from the solution, what are the goals. There are two types of goals - *declarative goals* which are the answers to the question “What should apply?”, and *procedural goals* which describe how can we achieve the declarative goals.
  - *Preferences* are the wishes of the users about the way the environment operates. Fulfilling them is nice, but not vital, it is considered optional.

- *Physical properties* define the physics according to which our domain functions, accordingly, the space and time dimensions. The space dimension is addressed by the *spatial properties*, which describe static locations and location changes (movements). In the *temporal properties* we explain how the time dimension looks in our domain. We answer questions such as whether we work full hours or are we allowed to have, e.g., 1.5 hours.
- *Behavioural outputs* consider the actions that can be performed in the environment of our domain, and which would result in a change of its state. The sub-categories of the behavioural outputs define the different groups of actuators to take into consideration - information and application services, humans, devices, robots.
- The knowledge in the category of the *uncertainties* is about different situations in which there is something of unpredictable nature. The following sub-categories are included here:
  - In *unexpected events* we include the information about what events occur in different situations, which are atypical for those situations. An example could be a situation in which someone is having a big party in the smart building, leading to a rapid and unanticipated decline in the air quality.
  - *Action contingencies* regard possible unsuccessful executions of actions, meaning that the execution doesn't proceed as expected. Here we have to think about all failures, timeouts, etc., which can occur with our behavioural outputs.
  - In the sub-category *partial observability* we include details about states in which we work with information which is incomplete or unreliable. An example would be anything weather-forecast-related.

For our work with the conceptual framework, we start with the identification of the declarative goals from the behavioral inputs. However, we decide to leave the formulation of the procedural goals and the preferences as the last step of our knowledge acquisition phase. The reason for this is that those two aspects provide the core of the planning domain since they answer the question “How do we achieve our goals?”. Accordingly, in order to be able to construct the best possible solution, we need information about the sensors and actuators which we have at our disposal in a smart building. The availability or the absence of a sensor or actuator can make a certain action and, therefore, a solution possible or impossible. The preferences are also left as last because they regard the question “What can be the preferences of the occupants regarding our solution?”.

Directly after the declarative goals, we identify all behavioral outputs. For this, we first have to determine all objects of interest in the environment, which will be our sensors and actuators. We consider all objects, services or other sources which deliver information that we need for the initial state as sensors. Among those are the weather forecast service, the service giving us the grid electricity prices, the one estimating the amount of solar power we will generate locally, and even the occupants stating wanted operation times or temperatures. The actuators, on the other hand, are all factors (services, objects, people, etc.) which can change something in the environment. Examples here are the system turning the air conditioning/heating on and off, and adjusting the thermostat settings, or the services and objects with which we purchase electricity, store solar power or use electricity from the battery. Having identified the sensors and actuators, we later pinpoint the behavioral outputs, giving us information about the actions our actuators can perform. We also





**Figure 3.1:** Conceptual framework for knowledge acquisition. Adapted from [GA16]

establish the physical properties of our domain by determining the spatial and temporal rules. Next, we analyze the various uncertainties in our domain, in order to understand their effect on our goals. As already mentioned, we finish with research into the procedural goals and the preferences.

In Table 3.1 we present the results of the performed knowledge acquisition phase. The categories, which are not considered relevant for our domain, are omitted. These include qualitative relations because we use concrete time points (hours), all types of uncertainty except partial observability, and robot actions.

In order to ensure more sustainable operation of the smart homes, we find three main goals to achieve, our declarative goals. The first ones are reducing the overall electricity usage in the home, and supporting the more sustainable operation of the smart grid. Additionally, in order to make the

Category	Acquired knowledge
Declarative goals	Reduced electricity usage, lower electricity bills, more sustainable operation of the grid
Procedural goals	Load shifting, Heating, Ventilation and Air Conditioning (HVAC) control
Preferences	Different automation levels
Temporal Properties (Metric Constraints)	Device scheduling and electricity purchases require specific time points, hours
Spatial Properties	Different cooling/heating settings possible for different rooms
Information Services	Estimation of amount of locally produced solar data through previously collected data
Application Services	Buying and selling electricity from/to the smart grid, calculating expected solar power (not taking into consideration locally stored data for learning), weather-forecasts, estimating indoor temperatures from weather-forecast
Human Actions	Giving information about operation times of the different appliances, choosing indoor temperatures
Device Operations	Setting thermostats, charging battery (BESS), using electricity from battery, scheduling devices
Uncertainties (Partial Observability)	Amount of locally produced solar power

**Table 3.1:** Acquired knowledge for the domain of sustainable smart buildings.

solution more attractive to potential users, it needs to offer support for decreasing electricity bills. The procedural goals for the identified declarative goals (load shifting and Heating, Ventilation and Air Conditioning (HVAC) control) are explained later in this section, accordingly in Section 3.1.1 and Section 3.1.2. Special attention deserve the preferences. Our solution targets many types of users, and, as we know, different groups allow different percentages of automation. Many people endorse technology taking control of some rudimentary activities, while others prefer to keep as much control as possible and are not very open to automation. Therefore, it is preferential for the domain to support different automation levels. For example, a high automation level would be an HVAC system, which determines completely alone the cooling and heating schedules. Accordingly, low automation would mean that the occupants give the exact HVAC settings they want. Then the smart home will only plan the electricity sources and make sure the occupants' wishes are perfectly met.

Since we work with electricity prices which vary through the day, and appliances' schedules, the time in our domain is measured in hours. The space, on the other hand, consists of rooms. The motivation behind this inclusion of locations in the domain is to allow different heating/cooling schedules in one home. For example, a baby's room should be kept warm throughout the whole winter period, the master bedroom might be okay with cooler temperatures, whereas an empty quest room should be heated only occasionally. Accordingly, we need to differentiate between the HVAC in the baby's room, the one in the master bedroom, and the one in the guest room.

As explained earlier in this section, the behavioral outputs contain the different sensors and actuators. However, since we have already explained our definition of sensors and actuators, and given examples, we just refer to Table 3.1, where they are grouped in the different categories.

Looking once again at the declarative goals, we remember that the solution should fulfill the goal of decreasing the electricity bills for the smart home. Accordingly, we have to make sure that the costs we estimate are close to the real ones. A factor of uncertainty, which complicates this, is the uncertain amount of locally generated solar energy. The estimate for it is strongly dependent on the correctness of the weather forecast, as well as on the service which will calculate it. The amount of solar energy is one of the factors determining the purchases of electricity from the grid. Therefore, not being sure about it, means that we cannot be sure about the real cost the plan (the price to pay for electricity). Thus, the information about the amount of solar energy becomes an important factor of uncertainty of the type of partial observability in our domain.

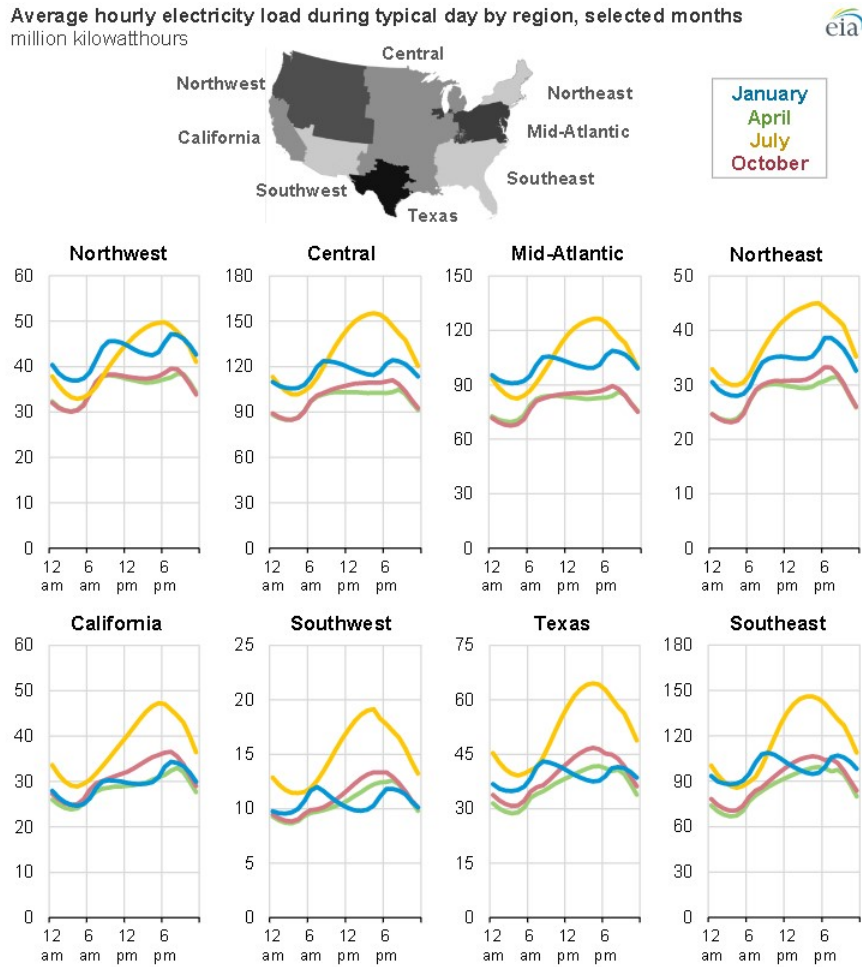
The remainder of this section gives more details about the procedural goals in the domain.

### 3.1.1 Load Shifting

The uneven demand for electricity in the traditional electricity grid is perhaps the factor the most responsible for its unsustainable operation. When analyzing data representing this demand throughout the day, we observe very strong amplitudes, accounting for the so-called on-peak and off-peak periods, as shown in Figure 3.2 [Hod]. When looking at the diagram for the average electricity load for Texas in Figure 3.2, we see that the curve for July has two very distinct extreme values. The maximal average electricity load is reached around 4:00 pm, therefore the hours around 4:00 pm (12:00 pm - 9:00 pm), are called on-peak hours. During those hours, as can be seen from the graph, more electricity is used than during the off-peak hours (2:00 am - 7:00 am). The presence of such extremes is found in every graph in Figure 3.2, meaning that the electricity demand throughout the day is indeed uneven by nature.

This uneven load distribution means that during the on-peak hours, the electric power generated from renewable sources or from the base power plants is not sufficient to cover the demand. This leads to the powering of peaking power plants, which, apart from being very unsustainable, produce electricity that is more expensive [KJU+13]. Therefore, one of the focus points of our research study is making the load shifting process, as described in Section 2.3.2 and shown in Figure 3.3, for smart buildings easier and profitable for the occupants.

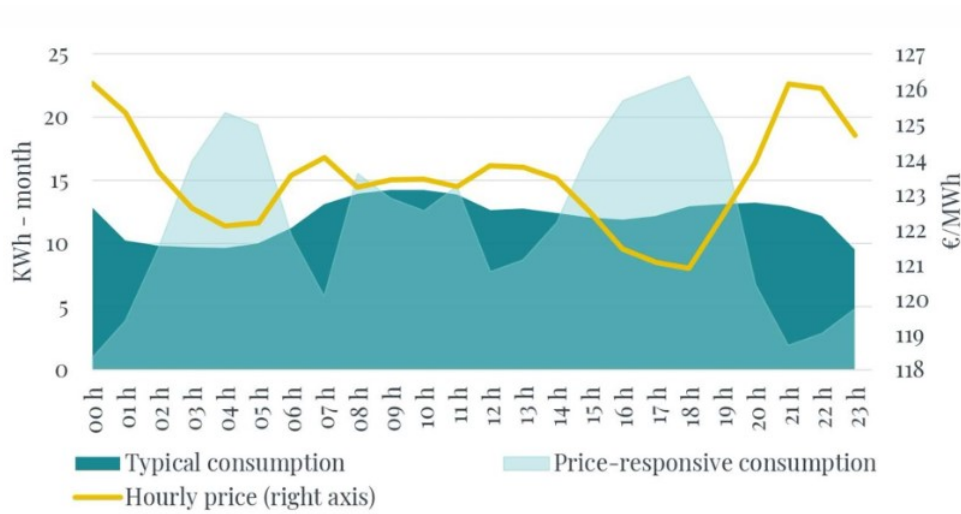
A characteristic of smart homes, which complicates the implementation of load shifting for them, is the lack of large-scale patterns in the occupants' activity and the "urgent" and spontaneous nature of their usage of appliances. Unlike in commercial buildings, for which the electricity usage patterns are well known and easily identifiable, this is not the case for residential buildings. In an office, we would expect usage of  $n$  computers for the duration of the working day, operation of kitchen appliances during the known time slot of the lunch break, etc. Accordingly, pre-purchasing and storing electricity to use during peak hours becomes easy and activities such as scheduling when to charge the notebooks in the buildings are possible. On the other hand, in a smart home, the electricity-requiring processes can hardly be rescheduled, e.g., TV watching, cooking lunch/dinner, heating water to take a shower. Accordingly, we cannot expect and require the occupants to postpone their own or their children's shower for 11:00 pm, in order to assist the load shifting, which as a strategy gives more global than individual advantages [Mai18].



**Figure 3.2:** Electricity demand through the day.  
Adopted from [Hod]

Therefore, we implement load shifting using two different approaches. The first approach regards devices, whose working time does not have a big effect on the occupants' comfort, e.g., a dishwasher. Those devices are scheduled for off-peak hours. The second approach concerns the appliances, whose working times are fixed by the occupants and should not be rescheduled for a random time of the day. For example, if we want to take an evening shower at 9:00 pm, we cannot set up the water heater for 2:00 am in the morning. For such devices, we plan the electricity sources which will be powering them during their working time. The inhabitants can give the system of the smart home their plan for the next day. For example, "cooking dinner at 6:00 pm", or, given that an occupant has a sports training at 2:00 pm, "taking a shower at 4:00 pm". Schedules of children are usually also relatively constant as they have specific bedtimes, daily activities, eating times, etc. Such patterns can often be recognized by the systems of smart homes, and, accordingly, the planned appliance usage can be given to the planner as part of the initial state.

We have developed a strategy to determine the best possible electricity supply: Firstly, we try to use energy from the local solar panel, which is being stored in the BESS. If this is not possible or the stored energy covers only a part of the need, we plan an electricity purchase from the grid.



**Figure 3.3:** The idea behind load shifting.  
Adopted from [HT]

For the purchase, we first determine the cheapest provider for the interval between midnight and the hour in which we will be using the electricity. Let the hour of this “cheapest offer” be  $h$ . If  $h$  is the same as the hour of usage, we buy this electricity day-ahead. However, if it is earlier, we have to check whether there will be enough capacity in the BESS to store the electricity from  $h$  to the hour of usage. If there is indeed enough space, we buy the wanted amount, store it in the BESS, and at the hour of usage power the device. However, if the space is insufficient, the process is more complicated. Firstly, we purchase as much electricity from this provider, as the available space in the BESS allows. Then, immediately after booking it, we trigger a new search for an electricity purchase from the grid. For this new search, the amount of necessary electricity is  $necessary_{new} = necessary_{old} - purchased$ . We specify this strategy, which we created, in pseudocode in Algorithm 3.1.

In this way, specifically by using locally produced electricity from renewable resources, or using electricity that we have purchased at an off-peak hour and stored in the battery, we still follow the concept of load shifting, while ensuring the comfort of the inhabitants.

### 3.1.2 Cooling and Heating Strategies

Even though load shifting reduces the bills of the homeowners to a certain point, the savings are not very considerable and the discomfort level is pretty high. Load shifting is a strategy that brings big advantages on a global scale, but causes multiple difficulties for the people individually. If a large portion of the occupants would implement load shifting, the demand on the grid would indeed be without the typical big amplitudes, which, as explained, would make it more sustainable. However, for the individual, this would mean having to conform to certain schedules, which do not offer optimal comfort levels. At the same time, the financial savings on the bill for electricity would barely pay out the investments for having a smart home and not a simple one [Mai18]. This implies that smart homes should support additional strategies, through which the occupants would profit

**Algorithm 3.1** Electricity Source Determination

---

```
1: procedure PLAN_ENERGY_SOURCE(amount  $a$ , hour of usage  $h_u$ )
2:    $capacity \leftarrow$  expected free space in BESS at  $h_u$ 
3:   if  $capacity \geq a$  then
4:     Use energy from BESS
5:   else
6:      $h_{cheap} \leftarrow$  The hour between 00 and  $h_u$  with the cheapest offer for electricity
7:     if  $h_{cheap} == h_u$  then
8:       Buy  $a$  from the cheapest provider at  $h_u$  and use it directly
9:     else
10:      Buy  $capacity$  from the cheapest provider at  $h_u$  and store it in the BESS.
11:      Use  $capacity$  amount of electricity from the BESS at  $h_u$ 
12:      call Plan_energy_source( $a - capacity, h_u$ )
13:     end if
14:   end if
15: end procedure
```

---

more. Accordingly, the need emerges to decrease the level of occupants' discomfort or the amount of energy used generally, not just shift the electricity usage. Since, according to various studies, e.g. [Offa], the HVAC sector is the biggest consumer of energy in a home, our study also concentrates on developing a strategy for a reduction of the energy demand associated with it.

Finding a strategy for optimization of this area proves extremely difficult. The trivial, and most popular, approach appears to be letting the HVAC systems operate for shorter time intervals or for temperatures, which are nearer to the outside ones. However, this idea means a negative impact on both the comfort and satisfaction of the occupants. Furthermore, such strategies do not require AI planning. They also neglect the wishes of the user, since we do not comply to the temperature desired by them. To the best of our knowledge, this study is the first to address the problem of making the operation of residential HVAC systems more sustainable using automated planning. However, achieving sustainability using AI planning is beneficial due to multiple reasons. Firstly, AI planning offers automation and so eases the whole process. Furthermore, the knowledge which is collected and modeled can easily be reused, adapted, or expanded for a wide range of use cases [Geo15].

Although other works also explore the question of addressing the temperature control in smart buildings from an AI planning perspective, their attention is solely on improving the comfort of the occupants and not optimizations in regard to sustainability [Hei20]. Stanullo [Sta21] provides a study that develops an approach for HVAC control, which should also lead to more eco-friendly operation of the building. He concentrates only on commercial buildings and further restricts the domain to concern only HVAC control in one room, which has to be cooled/heated to the wanted temperature as fast as possible. Respectively, he argues for achieving electricity savings by shortening the working time. However, no data regarding the electricity usage by the baseline and the new approach is provided to support this assumption. On the other hand, decreasing the working time of an AC unit in which it achieves some temperature, implies the thermostat is set on a lower temperature, and the AC uses more power. A result of such settings is an increased electricity consumption of the AC, which might even be higher than the one of the baseline[JEG]. Additionally, the research work does not observe the pattern in which the temperature increases/drops after the

end of the cooling phase. For example, if the walls of the considered room are not cooled and the temperature difference between the outdoor and the indoor spaces is high, the effect of the cooling would be offset very fast and a new cooling phase would be initiated. The results of such scenario imply an ineffective cooling strategy, which, though ensuring comfort for the occupants through improved indoor temperature reached in a very short time, is likely to be very unsustainable.

The approach we adopt to ensure more eco-friendly HVAC operation is implementing a *pre-cooling* technique for cooling and *pre-floating* for heating. As mentioned in the last paragraph, one problem with rapid cooling strategies is that they ignore the influence of *building thermal mass*, which is the capacity of the materials in buildings to absorb, store and release heat [Gre]. This means that if a building has been heating throughout the day, during and after a cooling session, the building materials (e.g., walls, floors, and carpets) will be releasing energy and respectively heating the living space. However, this does not apply if the materials have also been cooled. The same observations apply to the heating process. Pre-cooling and pre-heating utilize the effects of building thermal mass, while also ensuring load-shifting [ZZS+21]. Accordingly, applying those techniques not only relieves the on-peak demand on the grid, but also reduces the amount of used electricity and the corresponding bills [AEA12].

In the following, we explain the techniques of pre-cooling and pre-floating and why they are effective.

### 3.1.2.1 Pre-cooling and Pre-floating

According to Zeng et al. [ZZS+21], *pre-cooling* is “the method with the greatest potential of on-peak electricity demand reduction”. Additionally, Avci et al. [AEA12] show that it leads to decreased electricity usage and in this way to financial savings. The principle of this strategy is to lower the *temperature set point*, the temperature point at which the thermostat is set, as much as possible during the off-peak hours. This results in cooling for a very low price. It is preferable that this setting is kept during the whole off-peak period, allowing for the walls, floors, etc. to cool too [ZZS+21]. A factor that facilitates this process is that the off-peak hours are the night hours, when the temperatures naturally drop.

When a peak period begins, the set point is raised, so that the AC unit is running at lower power or is not running at all. At this point, the air inside the living space is warmer than the walls, resulting in the building thermal mass starting to “absorb” some heat. This continues until the two temperatures even out, keeping the indoor air temperature relatively low [ZZS+21]. The equivalent of this technique for heating is pre-floating, where we heat during the night and lower the set points during the day. In this case, the building thermal mass stores energy during the off-peak heating process and “gives” energy to the air during the on-peak, and by doing so warms it. The efficiency of these strategies, in terms of sustainability and financial savings, is confirmed in the study of Avci et al. [AEA12]. Since the choice of algorithms implementing these ideas required lots of research and comparison of different solutions, we consider the need for pre-cooling/pre-floating a “complex problem requirement”. Therefore, this decision has been assigned to phase 2 of our approach for solving a problem with HTN planning (see Section 3.2.3).

## 3.2 Addressing Complex Problem Requirements

Most domains and problems have some key characteristics, whose representation with HTN planning constructs is complex and often requires developing novel approaches. Usually, a domain would include numerous such characteristics. This means that if we were to create a very exact model of the domain, the whole process would become overly complicated and time-consuming, and the result would be very computationally inefficient. Since such an outcome should be avoided, a step that addresses those complex problem requirements is necessary. In this phase, we evaluate each of those complex requirements to identify the ones that are vital for our domain and cannot be skipped. We determine the best possible abstraction level for every requirement so that we have a trade-off between complexity and realism. Afterwards, we look into strategies for their representation.

For our domain, we identify three areas, which cannot be omitted from the model, but the knowledge acquisition for them requires a lot of research. The areas are:

- Electricity pricing strategy in the smart grid: The smart grid is promised to offer dynamic pricing and the inclusion of multiple electricity providers. Since this is not valid for the grid nowadays, the different pricing strategies are not common knowledge. Therefore, this topic needs more attention and research.
- Uncertainty with respect to the amount of locally generated solar energy: According to the effect the factors of uncertainty have on the plans in the domain, we may need to design the domain model in different ways. For example, if we want to focus on the probability of success, for the plans to execute successfully, we can make the tasks in the domain more general. However, our solution is supposed to intrigue users by decreasing their electricity bills. Accordingly, we have to focus on the effect the uncertain amount of locally generated solar energy has on the cost of the generated plans. Since we have to work with the costs of the actions in the domain, its model becomes more complex.
- Automated calculation of optimal temperature set points for pre-cooling and pre-floating: Since there are a lot of studies on this topic, the choice of one algorithm is rather complicated. Furthermore, many of the algorithms incorporate lots of physics and complex mathematical operations, meaning a huge increase in the complexity of the domain model [LJ21; WTS20]. Therefore, finding an approach, which is efficient and can relatively easily be incorporated into a domain model, is challenging.

In the remainder of this section, we will discuss each of those areas separately.

### 3.2.1 Electricity Pricing Strategy

As explained in Section 3.1.1, a part of our load shifting strategy, apart from scheduling appliances with flexible work times for off-peak hours, is the planning of electricity sources for the household devices. The planning of electricity sources includes an evaluation of the market prices, which are of dynamic nature, and vary between the different providers. Since we aim at performing load shifting, we should be able to analyze the market prices before the start of the day. This would allow us to determine the peaks and the price differences for the next 24 hours. Those requirements oblige us to use *day ahead pricing (DAP)*. This means that the providers of electricity in the smart grid announce the prices for their electricity and the available amount of it for the next 24 hours one



day ahead. Accordingly, we have enough information about the pricing to be able to determine when the prices will be highest/lowest and plan in advance of the day. Usually, the lowest prices are given by the providers for the off-peak period and the highest - for the on-peak one. Therefore, we can just aim to make cheap purchases and so automatically support load-shifting.

### 3.2.2 Uncertain Amount of Solar Energy

As mentioned in the knowledge acquisition phase, the domain of sustainable buildings connected to a smart grid has to work with the factor of an uncertain amount of locally generated solar energy. Usually, uncertainty is taken into consideration during the modeling phase by making plans more general, so that they can be executed even under some unpredicted conditions. However, since our domain requires making day-ahead purchases of exact electricity amounts at exact hours, we cannot allow ourselves to be very general. Furthermore, the amount of locally generated electricity has a direct influence on the cost of the plan that will be constructed. For example, if we overestimate the amount, the cost will be higher than calculated by the planner, since this overestimation means that we have not bought enough electricity to cover our needs and will need to buy electricity during the on-peak hours.

As the amount of locally generated electricity is uncertain, the plan's cost, representing the electricity bill for a day, also becomes uncertain. Therefore, we have to, indeed, find a way to model uncertainty by making the cost of the operator for storing solar power uncertain. Let us assume that we store some amount and use this amount in our plans. But later we manage to store only half of the assumed amount. Of course, this leads to a deficit that has to be compensated for through unplanned purchases of electricity, resulting in a higher cost than the estimated one. Accordingly, the operator for storing the solar power should have had a (higher) cost. However, using variable costs is a new concept, which, to the best of our knowledge, has not been applied until now. Still, this requirement of the domain cannot be omitted from the domain model, since it is tightly connected to the usability of our solution. The quality of our plans is related to the money we can save when using them, and those savings should act as a primary motivation for the users to adopt our solution.

### 3.2.3 Set Points Calculation

In this section, we describe the algorithm we use to determine the set-points of the thermostats for the cooling and heating schedules, as well as how the comfort of the occupants is integrated into it.

As explained in Section 3.1.2, we are employing pre-cooling and pre-floating strategies, in order to ensure sustainable HVAC operation. Therefore, we needed an algorithm that estimates the temperature set points for the thermostats of a smart building in accordance with the pre-cooling/pre-floating requirements. The algorithm of our choice, which enables this, is the one developed by Avci et al. [AEA12]. Our decision to use exactly this solution is because it does not only implement the described concepts efficiently but it also incorporates the occupants' comfort as a factor.

The algorithm by Avci et al. [AEA12] is specified as the first part of their two-phase strategy for cooling and heating. The phases of the strategy are as follows: calculation of the temperature set points with the aforementioned algorithm, and the pre-cooling/floating itself. However, in our solution, we integrate only the first phase, since we are only interested in the creation of a plan and not in its execution.

The first phase is the calculation of a temperature set point for each hour of the day. The determined set points are always within a pre-defined interval  $[T_{min}, T_{max}]$ , in order to allow customization of the temperature range to the occupants' liking. The interval can be chosen according to the specific need. For example, people who prefer cooler temperatures will give a lower  $[T_{min}, T_{max}]$  than the ones who like warmer conditions. The set points are determined in accordance with the pre-cooling/floating techniques (Section 3.1.2.1), meaning that their values are dependent on the day-ahead prices for electricity for the specific hours. The algorithm identifies a price range for each of the 24 hours based on the day-ahead pricing. Then it assigns a value to every set point according to the price range in which its hour falls. It should be noted that here we use aggregated electricity prices instead of using all prices from all providers for all hours. We use the average prices for every hour (over all providers), resulting in 24 values. This reduces the computational complexity drastically, but also makes the algorithm susceptible to outliers.

There are three main factors which determine the value of a set point for a given hour  $t$ :

- $\rho_t$ : averaged day ahead price for electricity for hour  $t$
- $T_{min}, T_{max}$ : lower and upper bounds for the temperature set points to be calculated
- $\alpha \in \{-1, 0, 1\}$ : tolerance level of the occupants of the smart home. It represents the willingness of the inhabitants to reduce their comfort level respective to the indoor temperature, in order to have bigger financial savings and a more sustainable operation. The tolerance level of  $-1$  shows a preference for comfort, meaning lower temperatures in the summer and higher in the winter, whereas  $\alpha = 1$  represents acceptance of slight discomfort in order to have bigger savings.  $\alpha = 0$  is the neutral tolerance level, where the occupants want a comfort-savings tradeoff.

The algorithm 3.2 calculates the set points  $T_t^{set}$  for a **cooling** session as follows:

- In line 1 we determine the total number of **different** set points to calculate,  $n$ . We point out that  $n$  is not the total number of set points to calculate, as this number is always 24, one set point for every hour of the day.  $n$  is the number of different values one set point can get.

The value of  $n$  is dependent on  $k$ , which is the temperature set point interval. In our domain model, we have set  $k = 1$ , meaning that two different set points can have a minimal difference of 1. This decision allows for smaller jumps in the temperature settings for the HVAC system, promising no extreme temperature changes for short time intervals, which are both unhealthy and more energy-demanding.

Accordingly, if we would take the temperature interval  $[75,79]$  and keep  $k = 1$ , we would have  $79 - 75 + 1 = 5$  different possible values for our set points: 75,76,77,78,79.

- In line 2 we set the difference between the highest and the lowest price. We will use this difference to determine the upper bound of the price range  $r_j, j \in [0, n]$  for each of the  $n$  different set point values in lines 5-6 or 9-10. For the example with the temperature interval

[75,79] and  $k = 1$ ,  $r_1$  gives the upper price bound for setting the thermostat to 75F. This means that we can set the thermostat to the first set point value ( $j=1$ ), 75F, at hour  $\iota$ , if the price for electricity at this hour,  $\rho_\iota$ , is not higher than  $r_1$ . This is also the meaning of line 14.

- As already mentioned, in lines 5-6 or 9-10 we determine the upper bound of the price range  $r_j, j \in [0, n]$  for each of the  $n$  different set point values. Here we have two options for this calculation, the choice of which depends on the tolerance level chosen. For a neutral  $\alpha$ , the price range grows linearly, meaning that we favor neither lower, nor higher temperatures. On the other hand, for  $\alpha = -1$  we want overall cooler temperatures, meaning more often 75F and rarely 79F. Therefore, the price bounds here cannot grow linearly, which leads to the choice of the coefficient for  $RANGE \frac{2^{\alpha(j-1)}(1-2^\alpha)}{(1-2^{\alpha n})}$ . The analog applies for  $\alpha = 1$ .

---

**Algorithm 3.2** Set Point Calculation for Cooling [AEA12]
 

---

```

1:  $n \leftarrow \frac{T_{max}-T_{min}}{k} + 1$ 
2:  $RANGE \leftarrow \rho_{max} - \rho_{min}$  //  $\rho_{max}, \rho_{min}$  are the minimal and maximal  $\rho_\iota$ 
3:  $r_0 \leftarrow \rho_{min}$ 
4: if  $\alpha = 0$  then
5:   for  $j = 1$  to  $n$  do
6:      $r_j \leftarrow r_{j-1} + \frac{RANGE}{n}$ 
7:   end for
8: else
9:   for  $j = 1$  to  $n$  do
10:     $r_j \leftarrow r_{j-1} + RANGE * \frac{2^{\alpha(j-1)}(1-2^\alpha)}{(1-2^{\alpha n})}$ 
11:   end for
12: end if
13: for  $\iota = 1$  to 24 do
14:    $T_\iota^{set} \leftarrow k * [\text{argmin}\{j : \rho_\iota \leq r_j\} - 1] + T_{min}$ 
15: end for

```

---

It should be pointed out that Avci et al. [AEA12] provide the complete algorithm only for the cooling process. However, for our research study, we adapted their version to also calculate temperature set points for heating by altering line 14, Algorithm 3.3. The chosen modification ensures that the set points are bound to high temperatures when the prices are low, since then we are subtracting a very small value from  $T_{max}$ . When the prices reach the on-peak values, we increase the subtrahend, resulting in a smaller difference.

Table 3.2 shows example calculated temperature set points for the three different tolerance levels and the set point interval  $[T_{min}, T_{max}] = [68, 74]$  for both pre-cooling and pre-floating.

After the computation of the set points, they are to be used for adjusting the thermostat temperature settings during the next day.

**Algorithm 3.3** Set Point Calculation for Heating

---

```
1:  $n \leftarrow \frac{T_{max} - T_{min}}{k} + 1$ 
2:  $RANGE \leftarrow \rho_{max} - \rho_{min}$  //  $\rho_{max}, \rho_{min}$  are the minimal and maximal  $\rho_t$ 
3:  $r_0 \leftarrow \rho_{min}$ 
4: if  $\alpha = 0$  then
5:   for  $j = 1$  to  $n$  do
6:      $r_j \leftarrow r_{j-1} + \frac{RANGE}{n}$ 
7:   end for
8: else
9:   for  $j = 1$  to  $n$  do
10:     $r_j \leftarrow r_{j-1} + RANGE * \frac{2^{\alpha(j-1)}(1-2^{\alpha})}{(1-2^{\alpha n})}$ 
11:   end for
12: end if
13: for  $\iota = 1$  to 24 do
14:    $T_{\iota}^{set} \leftarrow T_{max} - k * [\text{argmin}\{j : \rho_{\iota} \leq r_j\} - 1]$ 
15: end for
```

---

### 3.3 Modelling the Domain

In this section, we explain how we map the concepts we acquire in phases 1 and 2, into HTN planning constructs. Firstly, we present our general approach for the mappings, and then, in the subsections, describe the details of the different areas of our domain model. In the end, we also offer our solution for the incorporation of uncertainty.

Our domain is built upon algorithms. This is due to the fact that the algorithms describe how to achieve the different declarative goals from the knowledge acquisition phase. The declarative goals, according to their nature, are to be mapped to tasks, primitive and compound. Respectively, the algorithms, the procedural goals, are to be realized through operators and methods. However, the structure of the algorithms is complex, including the definition of local variables, loops, conditionals, etc. Therefore, we will discuss our approach to representing this structure with planning constructs. We also provide a schematic domain model in Appendix B, which includes the compound and primitive tasks included in the model, as well as methods used to decompose the compound tasks, together with their preconditions. We point out that the preconditions are formulated using natural language and no exact predicates, in order to make the model easier to read and understand. A detailed description of how we model the set points algorithms is offered in Section 3.3.4. However, the size of the domain model does not allow us to explain the entire model, and, therefore, we focus more on our approach for the mapping of the acquired knowledge.

#### 3.3.1 Choice of Predicates

As explained in the background information, the automated planning process requires an initial state, a domain model, and an initial task network. The initial task network contains the tasks to be performed in the initial state. Those tasks are addressed by the methods and operators defined in the domain model. Their application depends on the satisfaction of their preconditions, which are given through lists of predicates. This means, that our domain model should use a set of predicates, which

h	$\rho$	cooling			heating		
		$\alpha = 0$	$\alpha = -1$	$\alpha = 1$	$\alpha = 0$	$\alpha = -1$	$\alpha = 1$
1	0.0416	68.0	68.0	69.0	74.0	74.0	73.0
2	0.037	68.0	68.0	69.0	74.0	74.0	73.0
3	0.033	68.0	68.0	68.0	74.0	74.0	74.0
4	0.032	68.0	68.0	68.0	74.0	74.0	74.0
5	0.032	68.0	68.0	68.0	74.0	74.0	74.0
6	0.0336	68.0	68.0	68.0	74.0	74.0	74.0
7	0.0367	68.0	68.0	69.0	74.0	74.0	73.0
8	0.0496	68.0	68.0	70.0	74.0	74.0	72.0
9	0.0572	68.0	68.0	71.0	74.0	74.0	71.0
10	0.0709	68.0	68.0	71.0	74.0	74.0	71.0
11	0.092	69.0	68.0	72.0	73.0	74.0	70.0
12	0.1154	70.0	68.0	72.0	72.0	74.0	70.0
13	0.1518	70.0	68.0	73.0	72.0	74.0	69.0
14	0.25	73.0	69.0	74.0	69.0	73.0	68.0
15	0.2927	74.0	71.0	74.0	68.0	71.0	68.0
16	0.308	74.0	72.0	74.0	68.0	70.0	68.0
17	0.319	74.0	73.0	74.0	68.0	69.0	68.0
18	0.3038	74.0	71.0	74.0	68.0	71.0	68.0
19	0.221	72.0	69.0	73.0	70.0	73.0	69.0
20	0.13604	70.0	68.0	72.0	72.0	74.0	70.0
21	0.127	70.0	68.0	72.0	72.0	74.0	70.0
22	0.107	69.0	68.0	72.0	73.0	74.0	70.0
23	0.0756	69.0	68.0	71.0	73.0	74.0	71.0
24	0.0569	68.0	68.0	71.0	74.0	74.0	71.0

**Table 3.2:** Calculated temperature set points for the three different tolerance levels and the set point interval  $[T_{min}, T_{max}] = [68, 74]$  for both pre-cooling and pre-floating using Algorithm 3.2 for pre-cooling and Algorithm 3.3 for pre-floating.

enable us to represent the initial state of the environment very precisely, but also don't increase the complexity. Respectively, the majority of predicates we used are the ones delivering the input for our various algorithms. The rest of them correspond to the local variables for our algorithms, e.g., all  $r_j$ 's for the set point algorithm. Those predicates are being added to the state of the environment with different operators upon their calculation, and then removed from it with a different operator once the algorithm terminates. From the HTN domain model perspective, the termination of an algorithm means a successful decomposition of the compound task representing the invocation of the algorithm, with a matching method. The advantage of this approach is that it follows the variable visibility concept from programming and is therefore intuitive for the modelers.

An exact example for such a local-variable-predicate with its "managing" operators is given bellow. The predicate stores the value  $r_j$ , as defined in lines 6 and 10 of Algorithm 3.2, and Algorithm 3.3

- predicate:  $\langle r_j, (j, r_j\_value) \rangle$

- operator which adds it to the state of the environment:

$$calc\_r\_j = \langle \langle calc\_r\_j, (j, last\_calc\_r, range\_mult\_coeff) \rangle, \\ pre(calc\_r\_j), (eff^-(calc\_r\_j), eff^+(calc\_r\_j)), 0 \rangle$$

, where:

- last\_calc\_r is the last r we calculated
  - range\_mult\_coeff is the coefficient for *RANGE* from Algorithm 3.2 and Algorithm 3.3. Accordingly, the coefficient is either  $\frac{1}{n}$ , when the occupants have chosen the neutral tolerance level, or  $\frac{2^\alpha(j-1)(1-2^\alpha)}{(1-2^{\alpha n})}$ , otherwise. More specific information about the representation of the algorithms for set points' calculations are offered in Section 3.3.4 and Figure 3.5.
  - $pre(calc\_r\_j) = ()$
  - $eff^-(calc\_r\_j) = ()$
  - $eff^+(calc\_r\_j) = \langle \langle r\_j, (j, last\_calc\_r + range\_mult\_coeff) \rangle \rangle$ , which is the formula for the computation of  $r_j$  from Algorithm 3.2 and Algorithm 3.3.
- a basic operator for the deletion of this predicate could have the structure:

$$del\_r\_j = \langle \langle del\_r\_j, (j, r\_j) \rangle, \\ pre(del\_r\_j), (eff^-(del\_r\_j), eff^+(del\_r\_j)), 0 \rangle$$

with:

- $pre(del\_r\_j) = \langle \langle r\_j, (j, r\_j\_value) \rangle \rangle$
- $eff^-(del\_r\_j) = \langle \langle r\_j, (j, r\_j\_value) \rangle \rangle$
- $eff^+(del\_r\_j) = ()$

However, in our domain model, we do not use this exact definition of  $del\_r\_j$  presented above. This definition is given with the sole purpose of clarifying the concept. For  $del\_r\_j$  we perform an optimization, which alters its implementation. The explanation of the optimization we use is not of relevance to our mapping strategy and is therefore omitted.

### 3.3.2 Iteration Loops, Conditionals, Recursion

Since the algorithms explaining the logic of our domain include loops, conditionals, and recursion, we need to map those control flow statements to HTN planning constructs.

During the modeling phase, we wanted to ensure a cohesive style and an acceptable length of the model. Due to the broad functionality and complex algorithms we include, it was our priority to build the model with an understandable structure, supporting maintenance and further development. Therefore, we decided upon modeling both iteration loops (for, while, do-while) and recursion through compound-task-method-recursion. We are allowed to convert iteration to recursion, since both are equally powerful. We define compound-task-method-recursion as follows:

**Definition 3.3.1 (Compound-task-Method-Recursion)**

Let  $m = \langle ct, pre(m), tn(m) \rangle$  be the method decomposing the compound task  $ct$  with the task network  $tn(m)$ . Then  $m$  uses Compound-task-Method-Recursion, iff  $ct$  is included in  $tn(m)$ , or  $ct$  appears in any of the hierarchical levels resulting from multiple decompositions of  $tn(m)$ .

The conditional statements are also realized using methods. Let us say we want to transform the following part of an algorithm:

**Algorithm 3.4** Part of an Algorithm with Conditional Statements

---

```

1: ...
2: procedure CONDITIONAL
3:   if Conditions_1 then
4:     Tasks_1
5:   else if Conditions_2 then
6:     Tasks_2
7:     ...
8:   else
9:     Tasks_n
10:  end if
11: end procedure
12: ...

```

---

As can be seen in line 2, we encapsulate the entire conditional part in the procedure *Conditional*. *Conditional* would be a compound task for us, for which we will offer  $n$  possible decompositions with  $n$  different task networks  $tn$ . Each task network  $tn_i$  will correspond to *Tasks\_i*. The task networks will be used for methods as follows:

$$\forall i \in [1, n] : \text{define } m = \langle \text{Conditional}, \text{Conditions}_i, \text{Tasks}_i \rangle$$

The compound task *Conditional* is then included in the task network decomposing the whole algorithm.

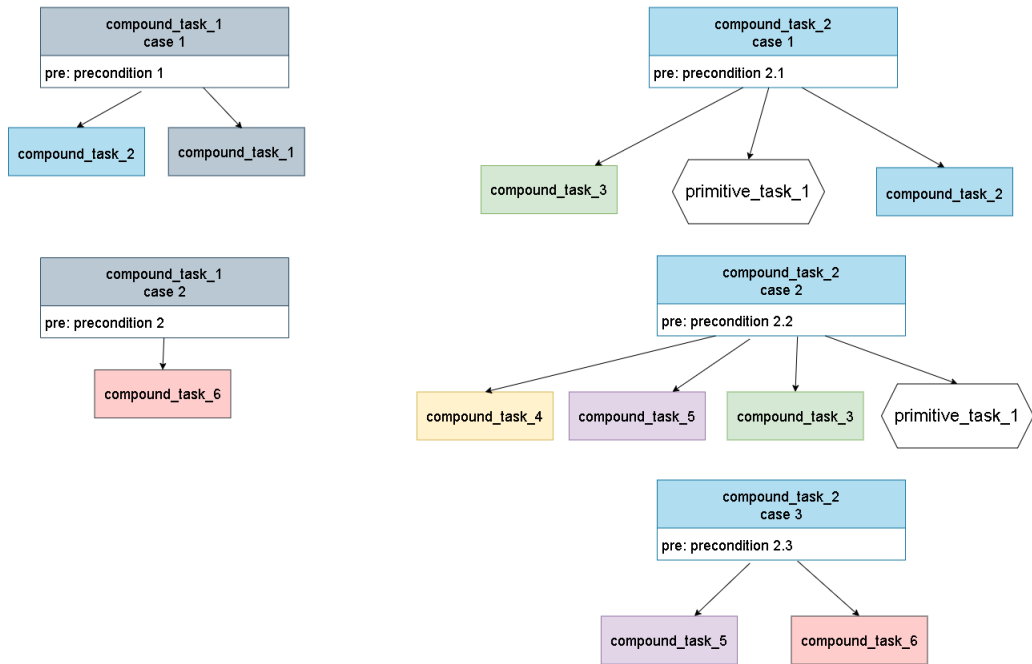
**3.3.3 Realization of Good Programming Principles**

In computer programming, there are various principles that are to be followed when creating a program that is understandable, comprehensible, and easy to debug and maintain. In order to ensure that our domain model satisfies these qualities, even though it is not a program, we followed some of those principles when building it.

The first principle we paid attention to is SRP, or the single-responsibility principle. According to it, every class, function, etc. should be responsible for only one function. The second principle we chose is about the size of functions in programs. The functions should always be small in order to allow for debugging and fast understanding.

Because of those principles, we split our algorithms into different cohesive sections (realization of SRP), which are also small enough to be comprehensible (small size). The different parts are transferred to compound or primitive tasks, depending on the granularity, which together build a

possible task decompositions for `compound_task_1`      possible task decompositions for `compound_task_2`



**Figure 3.4:** Example for size reduction of task networks decomposing compound tasks

task network. This task network is the result of the decomposition of the compound task we are addressing with the algorithm. If a certain cohesive section is too big, it is mapped to a compound task and is split further. We provide an example in Figure 3.4. As we can see, in order to keep the decompositions of `compound_task_1` and their preconditions simple and manageable, we map a big part of the first decomposition to another compound task (`compound_task_2`).

### 3.3.4 Set Points Calculation

The computation of the set points is modeled as a part of the domain and not as an external function. We provide the input for the algorithms (Algorithm 3.2 and Algorithm 3.3) with the following predicates, which occur as ground predicates in the problem instance (initial state):

- $\langle average\_price\_at\_h, (h, average\_price) \rangle$
- $\langle averaged\_price\_min\_max, (min\_price, max\_price) \rangle$
- $\langle tolerance\_level, (\alpha) \rangle$

As it can be seen, we choose to include the information about the average prices in the initial state. The reason for this decision is that even though they can be computed by the planner, this would require calling external functions, and would considerably increase the complexity. At the same time, determining the arithmetic mean should be of no difficulty for the external system, constructing the initial state. Additionally, `averaged_price_min_max` provides the minimal and maximal average prices, which are used to compute *RANGE*.



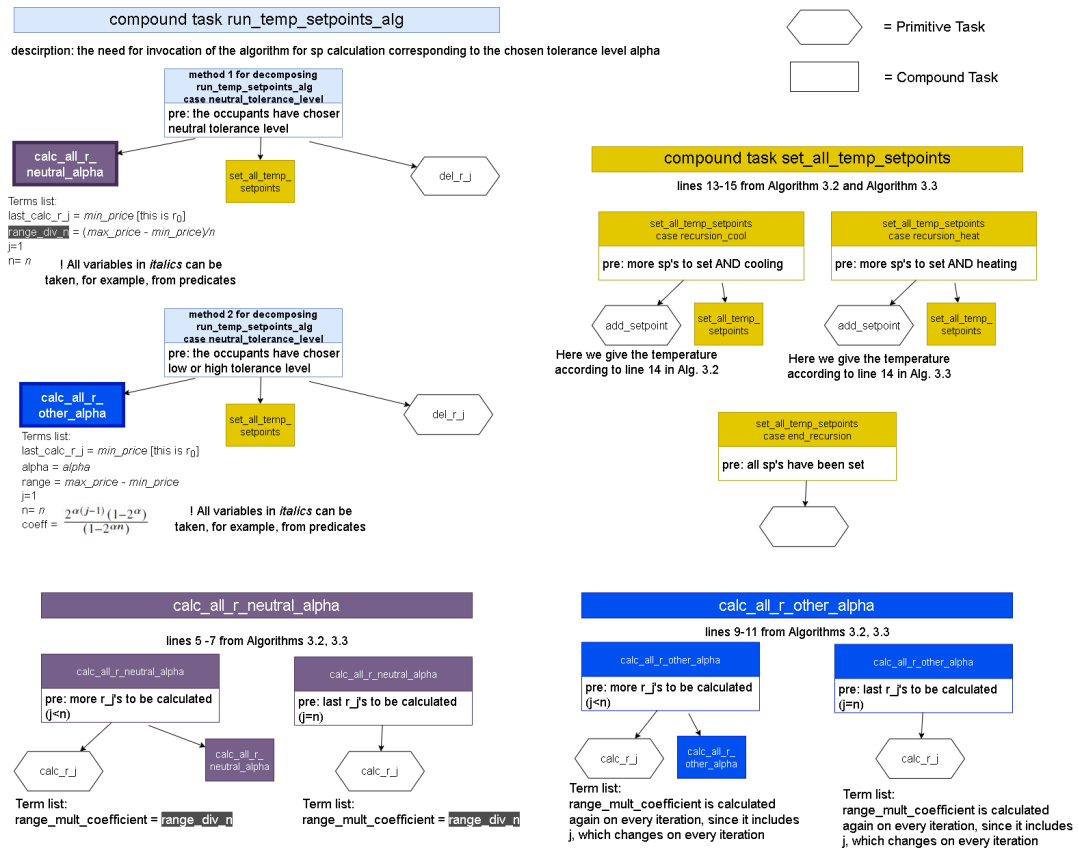


Figure 3.5: Part of the domain model which represents Algorithm 3.2 and Algorithm 3.3

The algorithm itself is to be realized with different methods and operators, following the approach for algorithm transformation described in Section 3.3. We show those methods and operators, as well as their connections, in Figure 3.5. We point out that the way *min\_price*, *max\_price* and other terms are obtained, can vary with different implementation approaches. E.g., they can be given through predicates, or found through sorted preconditions. As shown in the figure, compound task *run\_temp\_setpoints\_alg* represents the task to perform the algorithm. There are two methods which can decompose the task, the first, here called *method 1*, is used when the neutral tolerance level is chosen. The second, *method 2*, if for the other  $\alpha$  values. The reason behind this choice for two possible decompositions is the if - else construct, as well as the two different formulas for computing  $r_j$  (see lines 6 and 10 from Algorithm 3.2 and Algorithm 3.3). The formula in line 10 includes  $j$ , which is different for every iteration, while  $\frac{RANGE}{n}$  in line 6 stays always the same. This requires fundamentally different approaches, and, therefore, we define two different compound tasks - *calc\_all\_r\_neutral\_alpha* and *calc\_all\_r\_other\_alpha*. Compound task *set\_all\_temp\_setpoints* represents the need to execute lines 13-15 of the algorithms. The operator for the primitive task *add\_setpoint* follows exactly the strategy for local-variable-predicates described in Section 3.3.1.

After the calculation of the set points, the domain requires the performance of the primitive task of adjusting the thermostats according to the set points  $\langle set\_thermostat, (h, temperature, room) \rangle$ . For this, we provide the operator

$$set\_thermostat = \langle \langle set\_thermostat, (h, temperature, room) \rangle, \\ pre(set\_thermostat), (eff^-(set\_thermostat), eff^+(set\_thermostat)), 0 \rangle$$

, where:

- $pre(set\_thermostat) = \langle \langle wanted\_temperature, (room, h, temperature) \rangle \rangle$
- $eff^-(set\_thermostat) = \langle \langle wanted\_temperature, (room, h, temperature) \rangle \rangle$
- $eff^+(set\_thermostat) = ()$

We point out that the predicate  $\langle wanted\_temperature(room, h, temp) \rangle$  means that a wish for temperature  $temp$  at hour  $h$  in room  $room$  has been expressed. Therefore, it is included in  $pre(set\_thermostat)$  and removed by  $eff^-(set\_thermostat)$ .

#### 3.3.5 Flexibility and Occupants' Comfort

One of the key qualities we wanted to ensure in the development of our approach was its usability. The domain model is created according to two goals, which it should satisfy: Firstly, its usage should optimize the operation of the smart building in terms of sustainability. Secondly, its functionality has to meet the occupants' expectations and wishes. We address the second goal by allowing the domain model to work for different levels of wanted automation.

##### 3.3.5.1 Appliance Scheduling and Load-Shifting

The domain model supports two options regarding planning for household appliances. The first option is the determination of both the working hours for a device and the electricity sources to power it during those hours. In this case, we perform classic load shifting. The second option is for the inhabitants to give the hours in which they want to operate a specific appliance. In this case, we only choose the electricity sources. Both variants are described below.

Since many people like their schedule a certain way and are less likely to compromise about it, the basic functionality of our solution is just determining the cheapest electricity sources for an appliance. In this case, the working hours of the appliance are given in the problem instance through the predicate  $\langle to\_schedule, (appliance, start\_h, stop\_h) \rangle$ , e.g.,  $\langle to\_schedule, (fridge, 0, 24) \rangle$ . The example for the fridge shows that this input could be determined by the user, or extracted as a pattern by a separate system, if the system of the smart building allows this. Patterns such as fridge usage are very easily recognized and thus fit perfectly the domain model. An appliance for which only the electricity source is to be chosen, is given with the predicate  $\langle general\_appliance, (appliance) \rangle$ , e.g.,  $\langle general\_appliance, (fridge) \rangle$ . In this case, we only calculate the cheapest way to cover the electricity need of the specified appliances without scheduling them for off-peak hours. Here we realize Algorithm 3.1, again according to the approach for algorithm transformation described in Section 3.3.

If the occupants are willing to have a specific appliance scheduled according to the load-shifting strategy, they are still able to do that. Such devices receive the type dishwasher (predicate  $\langle \text{dishwasher}, (\text{appliance}) \rangle$ ), e.g.,  $\langle \text{dishwasher}, (\text{robot\_roomba\_kitchen}) \rangle$ ). Dishwasher is used as a metaphor for a device whose scheduling does not have a drastic influence on the comfort of the inhabitants. For example, for most families, it would not be a problem to have their Roomba vacuum cleaner clean their kitchen during the late night or early morning off-peak hours. We want to point out that even though type dishwasher exists, the dishwasher machine itself can still be entered as a general\_appliance in order to avoid scheduling.

Additionally, the domain uses the predicate  $\langle \text{appliance\_policy}, (\text{appliance}, \text{working\_hours}, \text{kWh}) \rangle$  to acquire knowledge about the electricity consumption of the different appliances. For example,  $\langle \text{appliance\_policy}, (\text{robot\_roomba\_kitchen}, 1, 0.583) \rangle$  means that the Roomba robot needs to work for one hour and has an energy consumption of 0.583kWh.

### 3.3.5.2 Automation Levels for the Cooling/Heating Process

Our solution also provides multiple automation levels for the cooling/heating process in the smart home:

1. Individual temperature set points and cooling/heating window: An occupant has specified that they want the HVAC to operate in a specific time slot, with temperature settings also defined by them. Our solution also allows the occupants to define different settings for different rooms. In this case, the planer only plans the electricity supply for the HVAC system and the wanted adjustments of the thermostats. The preferences of the inhabitants are represented in the problem instance through the predicate  $\langle \text{wanted\_temperature}(\text{room}, \text{h}, \text{temp}) \rangle$ , where the variable room should be bound to a constant of the type room (predicate  $\langle \text{room}(\text{room}) \rangle$ ).
2. Pre-cooling/pre-floating with occupant-chosen set point interval: An occupant has specified that they want specific room temperatures, but have not given working hours for the HVAC system (predicate  $\langle \text{user\_temp\_set\_points}(\text{lower\_set\_point}, \text{upper\_set\_point}, \text{room}) \rangle$ ). This setting implies the execution of the pre-cooling/floating technique with user-defined temperature set points interval. This case also allows the occupants to define different settings for different rooms. Here the planer plans the electricity supply for the HVAC system and the wanted adjustments of the thermostats as before, but also the temperature set points for the whole day. For the latter, we use the algorithm described in Section 3.2.3.
3. Fully automated cooling/heating process: The user only specifies that the control of the HVAC system is a responsibility of the smart home. This setting implies the execution of the pre-cooling/floating technique with the set points interval of 72F-81F during summer [Sle] and 66F-72F during winter [Ene]. The relevant interval is applied to all rooms. As in level 2, the planer plans the electricity supply for the HVAC system and the wanted adjustments of the thermostats as before, but also the temperature set points for the entire day. For the latter, we use the algorithm described in Section 3.2.3.

Levels 3 and 2 are chosen through the predicates  $\langle auto\_cooling, () \rangle, \langle auto\_heating, () \rangle$  in the problem instance. In these cases, the domain will plan the cooling/heating process, only when the weather forecast predicts temperatures above or below the ones wanted by the inhabitants. According to the predicates in the initial state, a corresponding compound task decomposition is selected, which ensures the wanted functionality.

In order to be able to plan the electricity sources for the cooling, the problem instance should include ground predicates for the predicates  $\langle ac, (ac) \rangle, \langle appliance\_policy, (ac, working\_hours, kWh) \rangle$ , where the variable *working\_hours* can be set to 0 or just “not\_given”. Accordingly, if the smart home has no air-conditioning, the cooling option will never be used. For the heating, the situation is different. The thermostat settings can always be changed (operator *set\_thermostat*, which we described earlier), but we plan the electricity sources only when the building uses electric heating. This knowledge is given through the predicate  $\langle electric\_heating, (eh) \rangle, \langle appliance\_policy, (eh, working\_hours, kWh) \rangle$ .

For levels 2, 3 our strategy for electricity source planning is to plan the sources for the off-peak hours, when the HVAC system is working on high. If used properly by the occupants without deviating too strongly from the recommended settings, the pre-cooling/floating should ensure that none or minimal electricity is used during the on-peak period. We identify the peak periods according to the data provided by the U.S. Energy Information Administration (EIA) as shown in Figure 3.2. More specifically, we analyze the curves representing the demand in January and July, which are considered representative of the winter and summer periods. The results are hard-coded in the domain model.

#### 3.3.6 Modelling the Uncertainty

As previously explained, we are modeling the uncertain amount of locally generated solar energy by making the operator for storing solar energy cost-variable:

$$store\_solar\_o = \langle \langle store\_solar, (amount, h) \rangle, \\ pre(store\_solar\_o), eff(store\_solar\_o), cost(store\_solar\_o) \rangle$$

,where  $cost(store\_solar\_o)$  represents the variable cost. The easiest way to achieve such costs is to create a probability distribution of their different possible values and take the expected value over it. The exact process is as follows:

1. Track the differences between the predicted amount of solar power and the real one at the end of every day. The answer to the question for how many days we save a difference depends on the system of the smart building, the available memory and computing power.
2. Use the noted differences to create a probability distribution over the differences.
3. Calculate the expected value over the created probability distribution,  $E(X)$ . This value represents the difference we expect between our forecast value  $F$  and the one we will have.
4. Use  $E(X)$ ,  $F$  and the average price of electricity at the hour of the storage,  $p_h$ , to compute the cost with Equation (3.1):

$$(3.1) \quad cost(store\_solar\_o) = p_h * \min(E(X), F)$$

The semantics of Equation (3.1) are the following: if we usually overestimate the generated amount,  $E(X)$  will be positive. By multiplying the positive value with the price, we get positive costs, meaning a higher electricity bill. This corresponds to the implication of overestimating the generated energy and having to buy during hour  $h$ . Accordingly, the calculated cost is the price to pay to cover the electricity deficit, that is expected to occur. However, if we typically underestimate the produced amount,  $E(X)$  would be negative. Since in this scenario we will have more electricity than supposed at the end, we will be able to sell the surplus, and this decrease the electricity bill. We represent this through the resulting negative costs of multiplying the negative  $E(X)$  with a positive price.

The reason for us to take the minimum of the expected value for the difference and the wanted amount is that it could happen that we want to store a smaller amount than the difference. In this case, it would be illogical to use the bigger value,  $E(X)$ , which is why we use the minimum of the two.

However, the tracking of differences, maintenance of the file, the creation of the probability distribution, as well as the determination of the expected value cannot be incorporated into our domain model. Especially step 1 is to be performed by another component of the smart building. The rest of the steps can easily be exported to an external function. Since many planners support the integration of external functions, which are not a direct part of the domain model, this option can be easily achieved. However, the exact implementation and logistics are determined later, in the implementation phase, since they do not affect the domain model. Important at this point is our decision to use cost-variable operators and knowing the exact semantics behind those variable costs.



## 4 Implementation

### 4.1 Planner

Before we implement our domain model, we have to choose the planner with which we will be working. The choice of the planner is an important step of the process since it defines the syntax to use for the implementation, the search algorithm, and other specifics which we have to take into consideration. The planner has to be a state-based HTN planner, since we use state-based HTN planning. Another preference is that it uses a pretty standard syntax. Those two criteria are met by Java Simple Hierarchical Ordered Planner 2 (JSHOP2), a Java implementation of the Simple Hierarchical Ordered Planner 2 (SHOP2), which is one of the most popular HTN planners [Ilg06].

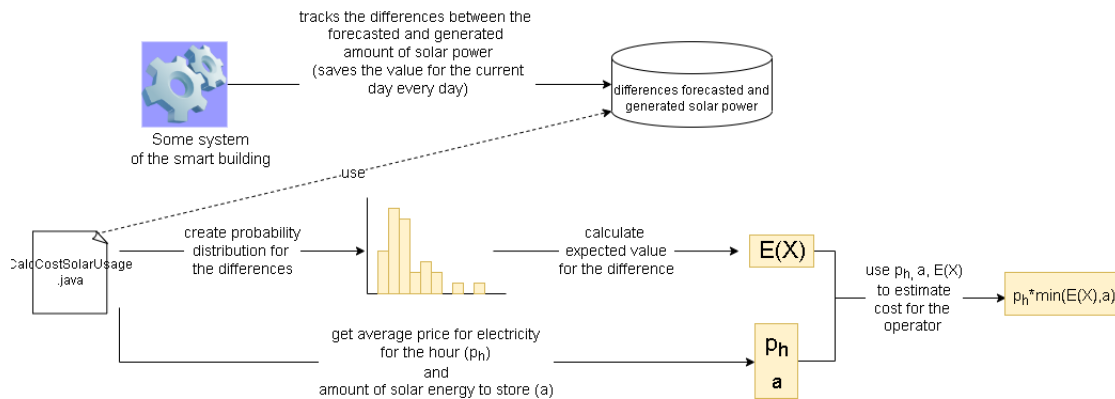
A significant characteristic of JSHOP2 is its large expressive power. For example, it offers the usage of sorted preconditions, which allows for easy and efficient implementation of the *min*, *max*, *argmin*, *argmax* functions that often appear in the algorithms for our domain. An additional feature of JSHOP2 is its built-in support for calls of many basic arithmetic functions, e.g., multiplication, division, etc., but also basic comparators. Again, this allows us to implement our algorithms, especially the one for the calculation of the set points, more easily. The integrated comparators facilitate, for example, finding the ground predicate representing the cheapest offer for electricity. Furthermore, this planner supports the execution of external calls, which also enabled us to implement the cost-variable operators. The reason for this is that the creation of the probability distribution and the calculation of the expected value couldn't be incorporated directly into our domain model. Last but not least, JSHOP2's syntax bears many similarities to PDDL, making it easily understandable for everyone who uses AI planning.

### 4.2 Domain Model

For the implementation of the domain model we use the syntax for JSHOP2 as defined in JSHOP2's official documentation [Ilg06]. The exact implementation is not shown in this work due to its length (670 lines of code). However, we include two example problem instances and the plans which were produced for them in Appendix C. Since the problem instances are used for qualitative evaluation, their construction is explained in detail in Section 5.2.2.

An important characteristic of JSHOP2 is that the preconditions are considered in the order, in which they are included in the domain model. This means that a precondition that is appearing later can be evaluated only if all the earlier preconditions cannot be satisfied. This allows us to implement the conditional statements (if - else if - else) from our algorithms very easily, just by ordering the preconditions correspondingly. However, we want to point out that many HTN planners use a non-deterministic decomposition technique, as described in Section 2.2, [GA15]. Accordingly, if one is interested in using the created model but with a different planner, specifically

## 4 Implementation



**Figure 4.1:** Implementing variable operator costs

one implementing this non-deterministic choice, the preconditions of our methods would need alterations to ensure correct functionality. The alterations consist in implementing all preconditions in full. Let us assume that in our implementation we have *method1* and *method2* which decompose the same compound task. Let us also assume that *method1* precedes *method2* in the file. This means that the planner will always try to apply *method1* before *method2*. It will evaluate the preconditions of *method2* only if the ones of *method1* cannot be satisfied. Accordingly, in our implementation, we do not include in the preconditions of *method2* that those of *method1* should not apply. However, in the case of non-deterministic choice, this information should be included.

### 4.2.1 Incorporating Uncertainty

As explained before, we manage to incorporate the factor of weather uncertainty in our domain model by applying the concept of cost-variable operators. More specifically, we make the cost of the operator for storing locally generated solar energy variable. As already known, it is determined with the help of the expected value over the probability distribution of differences between the predicted and generated amount of solar power. However, the creation of this probability distribution, as well as the computation of the expected value over it, cannot be incorporated into the domain model itself. Luckily, JSHOP2 supports the calls of external functions, which enables us to easily implement all calculations in a separate .java file and call this functionality whenever we need it.

Figure 4.1 depicts the structure behind our implementation of the cost-variable operators. Firstly, it should be pointed out that we are delegating the collection and management of the data about the differences between the forecasted and the generated amount of solar power. This data should be easy to collect from the sensors of any smart building which has solar panels and has access to the weather forecast. Accordingly, we assume the tracking and logging of this information to be a responsibility of a separate system. The rest of the calculations and their invocations are realized in the .java and domain model files we provide.

JSHOP2 allows defining our own external functions and calling them in our domain model, given that the implementation fulfills the following criteria:

- The function should be implemented in a Java class, whose name should be the name of the function. The class should also implement the Java interface *Calculate*.



- In order to implement the Calculate interface, the developer should implement the *call* function from the interface in the created class. The only parameter of *call* is of type *List*, representing a list as defined in JSHOP2’s documentation. This list contains all parameters which the function will need. E.g., the call (call CalcCostSolarUsage 20 0.14) would result in calling the *call* function from *CalcCostSolarUsage.java* with the parameter [20, 0.14]. The return value of the *call* function should be of type *Term*, a term, again, as defined by JSHOP2.

It is important to mention that JSHOP2 only creates one object per class, even when the corresponding function is called more than once. During the planning, there can never exist two objects of the same external-function-class. We utilize this design decision by defining the class attribute *expectedDifference*, which is set in the constructor of the class *CalcCostSolarUsage*. In the constructor, as shown in Listing 4.1, we load and read the file with the logged differences, use it to create a probability distribution over the values for the differences, and finally calculate the expected value over it and save it in *expectedDifference*. In this way, we don’t have to run all those computations during every call of the *CalcCostSolarUsage* function, since the field value is shared for all calls. Accordingly, this makes the calculation of the cost of the cost-variable operators way faster.

The *call* function *CalcCostSolarUsage* takes one parameter, of type *List*, which includes two *Terms*. The first one has to be the amount of solar power we expect to be generated and stored for a specific hour, and the second - the average price for electricity during this hour. We then calculate the expected cost for the application of the operator with the formula  $cost = price * Math.min(amount, expectedDifference)$ , representing Equation (3.1), whose interpretation is offered in Section 3.3.6. We return the value by creating a *TermNumber* object representing it.

## 4.3 Input and Output

Since our domain model supports planning for different automation levels, each of which requires different data, and is invoked through different logical atoms, the ground predicates for the problem instances have to be constructed with precision. In Appendix A, we provide an overview of the different logical atoms that are to be used when creating problem instances. However, all initial states that we used in the evaluation phase are constructed automatically. For this, we create our own Java *generator* program, which is also available in our Gitlab repository.

The program has different settings for the generation of different problem instances. The purpose of the problem instance is chosen through *args[0]*, whose value is to be set to “evaluation”, “automated\_hvac”, “hvac\_user\_chosen\_sp”, “user\_hvac\_schedule”, “no\_hvac\_only\_appliances”. We want to point out that the “evaluation” setting generates the files for the quantitative evaluation, the specifics of which are described in the next chapter. With “automated\_hvac”, the program creates a scenario, in which the highest automation level (fully automated HVAC control) is wanted. Accordingly, “hvac\_user\_chosen\_sp” is for the second level, where the occupants specify intervals for the set points for the different rooms. The lowest level is covered by “user\_hvac\_schedule”, which generates an instance containing exact settings for the HVAC. In case no HVAC control is wanted, but only appliances, for example, a scenario during spring, when no heating or cooling is necessary, “no\_hvac\_only\_appliances” can be used. With *args[1]* the user can select a name

## 4 Implementation

---

### Listing 4.1 CalcCostSolarUsage.java

---

```
import JSHOP2.*;

import java.io.*;

public class CalcCostSolarUsage implements Calculate{

    private double expectedDifference;

    public CalcCostSolarUsage(){
//differences.txt saves forecasted value - produced amount
        String pathDoc = System.getProperty("user.dir")+"\\differences.txt";

        try(BufferedReader br = new BufferedReader(new FileReader(pathDoc))) {
            int numberDifferences=Integer.parseInt(br.readLine()); //should be saved in the
first row
            double meanInProgress=0;
            for(String line; (line = br.readLine()) != null ;) {
                meanInProgress+=Double.parseDouble(line);
            }
            this.expectedDifference=meanInProgress/numberDifferences;
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @Override
    public Term call(List l){
        double amount =((TermNumber)l.getHead()).getNumber();
        l = l.getRest();
        double price =((TermNumber)l.getHead()).getNumber();

        return new TermNumber(price*Math.min(amount,expectedDifference));
    }
}
```

---

for the problem instance to be generated, however, this is optional. The files for the quantitative evaluation are saved in the *generated\_evaluation* folder, whereas the other problem instances - in *generated\_problems*.

The output of the planning process is a plan, constructed by JSHOP2. However, since the plan is the linearization of all operators which were applied during the planning process, some of our plans can be unnecessarily long. The reason for this is the algorithm for the set points calculation, which requires the application of operators such as *!calc\_r\_j*. Accordingly, such operators can be removed from the produced plan without any loss. This is the functionality of the *Postprocessing.java* file, which is also part of our *generator* program. We alter the *make.bat* file for Windows for JSHOP2, so that we can create a plan for any problem instance and have it post-processed directly after the generation of the plan. The whole process is automated and is started with the command *make plan %problem*, where *%problem* is the name of the problem instance. The post-processed plan is saved in *results/produced\_plans*.



# 5 Evaluation

## 5.1 Evaluation Framework

As explained before, after the implementation of our solution, we conduct an extensive evaluation of it, in order to test its quality. We cover three different types of evaluation, as defined by Georgievski et al. [GA16] - quantitative, qualitative, and evaluation through demonstrations. For the demonstrations, we construct different scenarios, covering the entire functionality of the domain. Additionally, we use the habitual patterns described by Abreu et al. [APF12] to construct our problem instances to ensure their realism. The produced plans are studied in terms of credibility. Since, upon multiple examinations, no logical errors have been identified, we assume that produced plans are indeed credible.

For the quantitative evaluation, we review the performance of the planner in terms of the scalability of our solution. For this, we look at the planning time for problem instances with increasing complexity of the initial state. We construct two experiments for this. In the first, the size increases because of the growth in the number of appliances to schedule. In the second, we keep widening the list of electricity providers.

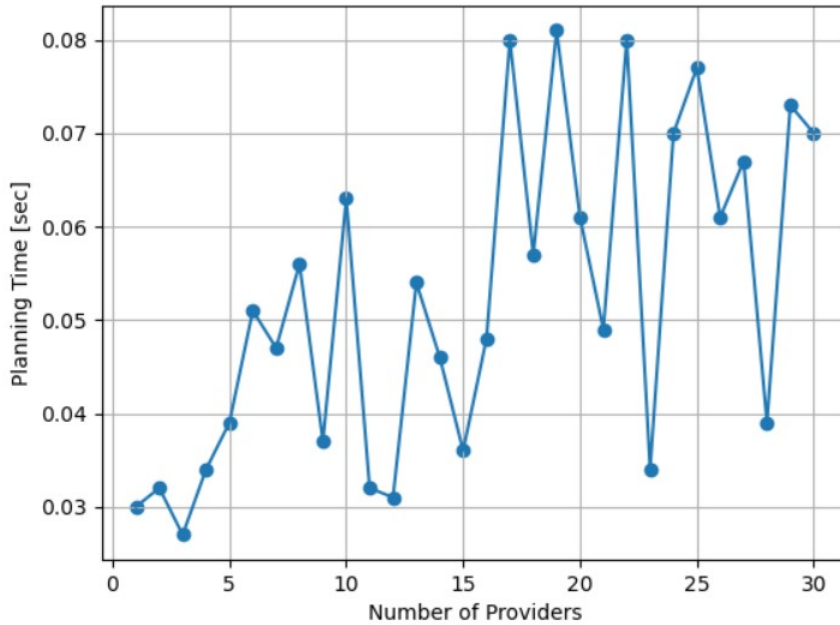
Finally, for the qualitative evaluation, we compare the cost of plans when using the highest automation level (completely automated set-points determination), and the lowest one (the user chooses the exact thermostat settings). Of course, we ensure that the temperature goals with both strategies are comparable, in order to ensure very similar experimental conditions.

## 5.2 Results

### 5.2.1 Quantitative Evaluation

For the quantitative evaluation, we explore the connection between the time necessary to create a plan for a given problem instance, and the complexity of this instance. In our case, an increasing complexity can be defined both as an increasing number of appliances to schedule, and as having more electricity offers from the smart grid, respectively, more providers. As those factors are connected to the size (length) of the problem instance, this experiment also covers the question of the relation between the planning time and the size of the initial state.

As our two definitions of complexity, number providers and number appliances, are not connected, we explore their effects through separate experiments. The problem instances are generated with our *generator* program, which has settings for creating the evaluation initial states.

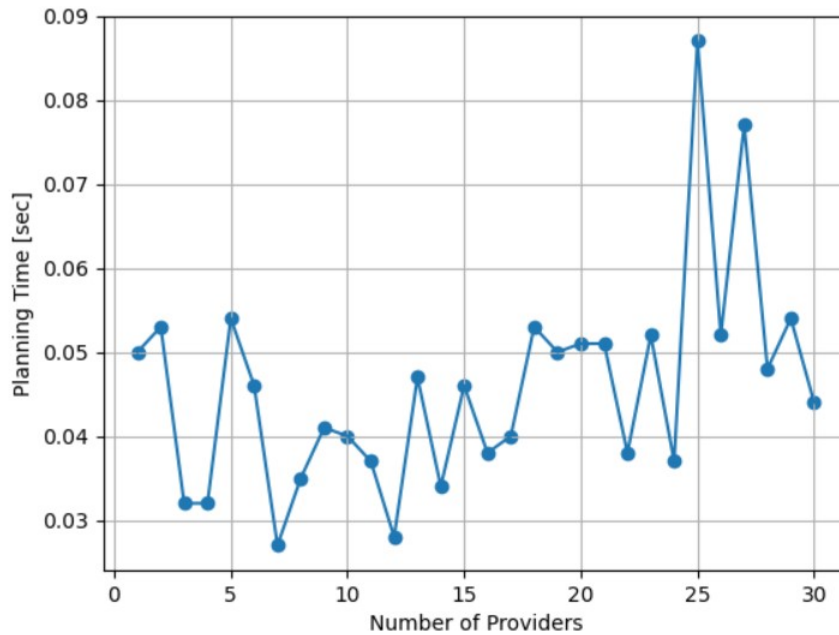


**Figure 5.1:** An overview of the planning time for an initial state with a varying number of providers and varying types of appliances.

The set-up of the initial states with varying numbers of providers is the following: every instance has 4 appliances to schedule, randomly generated, the instances are named *problem<sub>i</sub>*, where  $i \in \{1, \dots, 30\}$  and represents the number of different providers included. The upper bound for this number is 30, as above it JSHOP2 could not process the problem instances due to their length (they include  $> 30 * 24$  offers, since every provider is registered with an offer for every hour of the day). We want to point out that the type of the appliances, their electricity consumption, and working times vary for the different problem instances. We make this construction decision because the offers from the providers differ. This means that using only one configuration for the appliances will benefit some instances and disadvantage others. The resulting planning times are shown in Figure 5.1. As can be seen from the values, the planning time does not seem to be affected by the number of providers systematically.

To disprove the hypothesis that the lack of a strong tendency is due to the variation in the types of the appliances among the problem instances, we conduct a follow-up experiment. The experiment is built as the original one, but here we fix the types of appliances to be the same in all problem instances - one dishwasher and three general appliances. The planning times are shown in Figure 5.2. The values, once again, do not reveal any clear dependency.

The set-up for the second experiment with the problem instances with varying numbers of appliances is as follows: every instance has 1 provider, respectively 24 offers, one for each hour. The provider offers are randomly generated. The instances are named *problem<sub>j</sub>*, where  $j$  starts at 1 and goes up to 101, using the increment of 5. We use  $j$  to denote the number of appliances to schedule in the constructed initial state. The working hours, as well as their energy consumption, are chosen randomly. The resulting planning times are shown in Figure 5.3. The results show an extreme value



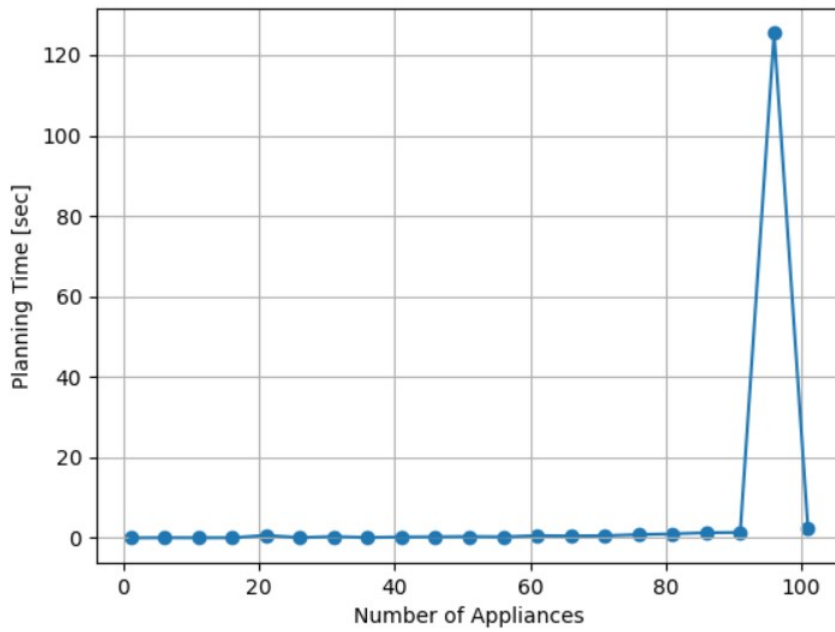
**Figure 5.2:** An overview of the planning time for an initial state with a varying number of providers and fixed types of appliances for every planning instance.

at  $j = 96$ , the cause for which, upon closer examination, is identified to be problematic values for the amount of electricity offered by the provider at specific hours. Since the amount was too low, the planner has to take additional steps, which is not the regular case. However, we do not provide additional information about that case, since it occurs because of the completely random choice of values to use in the problem instance, which in this case are very unrealistic.

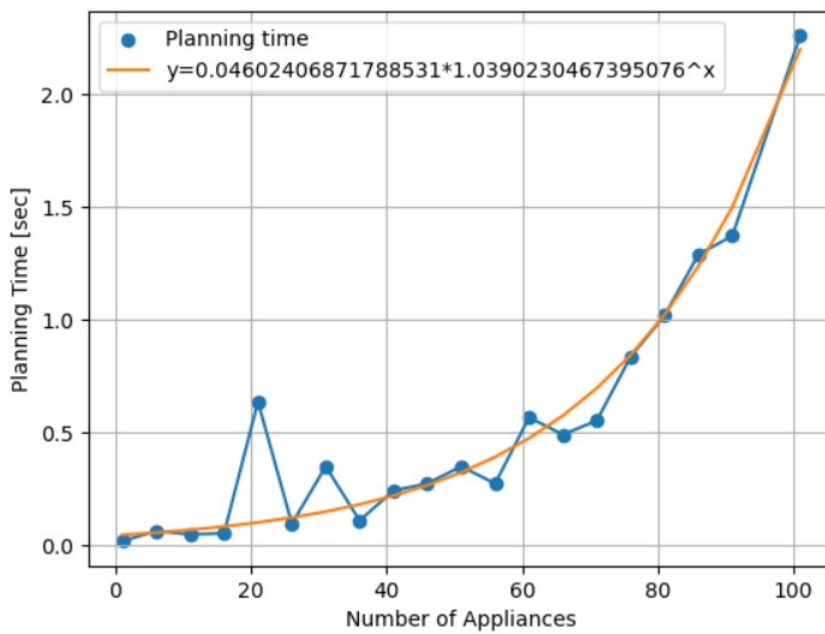
Accordingly, we remove the extreme value from the data set with the results and use the altered one for the evaluation. The plot of the new data set is depicted in Figure 5.4. We also use exponential regression to recognize a tendency. Since the exponential regression delivers promising results, we can conclude that the planning time grows exponentially with regard to the number of appliances. However, even though exponential complexity would usually mean a restriction in the applications of a program, this is not the case with our solution. The reason for this is that the domain of smart homes is very unlikely to ever include a number of appliances to schedule, that comes anywhere near 100. Since the results up to the border of 100 appliances show very short planning times, we can conclude that our solution delivers satisfactory results.

### 5.2.2 Qualitative Evaluation

As mentioned, we create two different problem instances *problem\_highest*, which requires fully automated cooling, where the set points are automatically chosen, and *problem\_low*. For the latter, the occupants have given their specific cooling requirements throughout the day - 77F between 10:00 am and 6:00 pm, and 79 after 6:00 pm. This means that in the second case the air conditioner will be cooling between 10:00 am and 6:00 pm, and in the first case - we will be performing a pre-cooling strategy with a longer duration than the strategy in the second case.



**Figure 5.3:** An overview of the planning time for an initial state with a varying number of appliances to schedule.



**Figure 5.4:** An exponential regression over the data for planning time for an initial state with a varying number of appliances to schedule. Extreme values have been removed.



In order to ensure that the plans for the two initial states will be comparable, we construct them identically, with their only differences being the HVAC preferences. More specifically, this means that we include exactly the same providers and electricity offers in both, remove all additional appliances, in order to focus on the HVAC, set the BESS capacity, and produce solar power to 0. The day-ahead prices we use, are constructed according to <https://www.pjm.com/>. The two problem instances, as well as the produced (and post-processed) plans, can be found under Appendix C.

The created plan for the low automation level has a cost of 2.856. The cost with the highest automation level is 1.013249999999998. Those results show that without the usage of our solution, the user would pay around 282% of the price they would pay otherwise. Accordingly, the difference in the two values is high enough to show that the plans produced with our solution are of better quality.



## 6 Related Work

The topic of how to make buildings and especially smart buildings more sustainable is widely researched. However, the majority of studies do not explore this question from the perspective of AI planning. Additionally, to the best of our knowledge, none of the works which use AI planning, incorporate the notion of uncertainty in the domain model using variable costs.

The closest study to ours is [GDP+12], which also focuses on AI planning for smart buildings as a part of the smart grid. However, unlike ours, it does not use HTN planning, but describes a general approach for AI planning, which could also be applied to HTN. Additionally, the objective of the work is defined differently - the team does not focus on a more sustainable operation, but on reducing the electricity bill. Accordingly, our approach for scheduling devices for off-peak hours and choosing providers according to their prices for electricity can also be found there. What makes the proposal exceptionally good is the usage of so-called “policies”, which describe the “behavior type” of appliances regarding their electricity usage. In our work, we have a similar notion recognizable in the types “dishwasher” and “general appliance”. However, the policies by Georgievski et al. are more comprehensive. Though, no solutions targeting other areas (e.g., optimizing HVAC set points) are suggested. The work by Georgievski et al. discusses possible architectures for the proposed solution. We believe that their architecture could easily be adapted to our work.

Additional information on architecture, but also different solution approaches to common HTN planning approaches, are offered by Georgievski [Geo15]. His work discusses solutions based on HTN planning for coordinating different services or appliances. Even though our study also offers some type of coordination of appliances, its focus is on scheduling them for load balancing and determination of electricity sources, a day ahead. Georgievski’s work, on the hand, is about reacting to changes as they occur, e.g., reacting to a lamp being turned on, or adjusting a desk. Additionally, of very big importance is the provided proof of concept, since Georgievski deploys his proposed solution in an actual environment and describes the results.

Regarding the modeling of uncertainty, numerous scientific works make suggestions on how to approach the challenge. [GA16] gives a very comprehensive overview of the different possible approaches. What makes their work so important, however, is that their proposals are split into categories, each of which corresponds to a type of uncertainty. E.g., the team suggests using nondeterministic outcomes and conditional effects of operators when working with, what they call, unexpected events and action contingencies. For uncertainty due to partial observability, they introduce the idea of utilizing a memory of previous actions. Very noteworthy is also their overview of studies on the different uncertainty types, which can be used as a starting point when modeling uncertainty.

Another research work in this field is by Alnazer et al. [AGA22]. Their study not only develops the concept of variable-cost operators, which are used in our work, but also focuses on different uncertainty types. We consider the study exceptionally relevant for domains that require risk-aware

planning, since the paper provides extensive guidelines on this subject. [BK04] is of interest for domains where uncertainty is not expressed through uncertain action costs, but through uncertain success rate. In this case, we do not ask what will the cost of performing a specific action be, but how likely is it that we will be able to perform the action. Accordingly, [BK04] describes an approach to finding the plans with the highest probability of success.

Another option is to approach the challenge through probabilistic planning. For example, [Ric17] models domains with the uncertainty feature as Partially Observable Markov Decision Processes (POMDP). [CC07], on the other hand, choose Bayesian Networks as a basis for the modeling.

However, outside the planning area, there are also numerous research studies that address different viewpoints on how to ensure sustainability in smart buildings. One very important aspect, which complements this work, is occupant behavior.

Understanding occupant behavior, in order to be able to motivate them to adopt certain sustainable habits, is vital if real advances are to be made. Furthermore, the process of learning how to adapt the systems to the occupants maximally cannot be neglected either. It is what allows us to make the usage of sustainable systems more popular, and also motivates the users to overlook some discomforts because of the advantages being offered to them. In [MMA+22], Malik et al., give a starting point in regard to thinking about occupant behavior and interests in the context of energy efficiency. Heidari et al. [HMK22] use reinforcement learning methods to understand and predict user behavior and accordingly optimize energy usage. Additionally, the habitual patterns described by Abreu et al. [APF12] deliver good insight into the occupants' needs and constraints. The patterns can be used to develop concepts of comfort for the inhabitants, and classify which appliances in a household are eligible for scheduling. Additionally, as mentioned in Section 4.3, we constructed our problem instances in accordance with the patterns, to ensure their credibility.

## 7 Conclusion and Future Work

HTN planning has been used to address problems in different domains. In our research study, we examine the domain of smart homes in order to “improve” it, in terms of sustainable operation, via HTN planning. For this, we perform a systematic analysis of the domain. With this knowledge acquisition phase, we identify the smart buildings’ utilization of the smart grid as a field that can be optimized. We pinpoint load shifting as a suitable strategy, which can also be realized with automated planning. However, we also provide an argumentation that load shifting is a strategy that is very beneficial on a large scale, but not on a smaller one. In some cases, it could even be considered disadvantageous for the occupants. Accordingly, we identify a second strategy that would allow for a more environmentally-friendly operation, while providing benefits for the users - HVAC optimization. For this, we use pre-cooling and pre-heating strategies.

As a next step, we discuss and “polish” the more complex requirements of the domain, e.g., the incorporation of uncertainty. We model the domain with a special focus on its quality and usability. In order to ensure its applicability for different scenarios with different occupants personas, we introduce different automation levels both regarding the load shifting, and the HVAC operation. We also introduce our approach to reflecting the uncertain nature of the domain in its model through the usage of cost-variable operators, as defined by Alnazer et al. [AGA22]. Since our cost-variable operator is related to the amount of locally generated solar power, we also improve the estimations of the amount of this energy, which can be vital for a smart building.

The evaluation of our implementation of the domain model shows very low planning times, meaning that the plans can be computed efficiently. Furthermore, the qualitative evaluation reveals that we can reduce electricity costs by almost 2/3 when using the proposed solution. Accordingly, the developed approach seems to meet the primary criteria of the users - it facilitates considerable savings, is fast, and still allows the user to increase or decrease its responsibility and involvement with the environment of the building.

However, our solution shows that achieving a more sustainable operation of a smart building is not only dependent on the technology in it. Our caution to use scheduling for all devices or to allow only HVAC operation with the set points we calculate, illustrates the relevance of occupants’ behavior and their willingness to decrease their comfort level. Additionally, our usage of pre-cooling and pre-floating strategies implies also the importance of the architecture. Decisions such as the choice of building materials, the thickness of walls, the floor type, etc., all have a huge effect on indoor temperature, thermal mass, and respectively on the success of the pre-cooling and pre-floating techniques.

The solution proposed in this study is easily extendable with different strategies for improving sustainable operation. For example, future research could focus on the management of “energy vampires”, devices that aren’t actively used but still consume electricity. Another good field would

be the transport one, since electric personal vehicles are becoming increasingly popular. However, powering them requires a lot of energy. Therefore, more insight into this topic would be of great importance.

Another direction for future work could be the adaptation of our domain model implementation for other planners. Because of JSHOP2's characteristic to choose the decomposition to use according to their order in the domain model file, our domain model has a special arrangement of the methods which is not to be changed. Accordingly, by interest to use the created model but adapt it for a different planner, specifically one implementing non-deterministic choice, the preconditions of our methods would need alterations to ensure correct functionality.

Lastly, as we mentioned at the beginning of this work, here we are looking only at the residential sector, excluding the commercial buildings from the smart buildings' domain. Therefore, further development of the domain by extending it with knowledge about office buildings is recommendable.

## Bibliography

- [AEA12] M. Avci, M. Erkok, S. Asfour. “Residential HVAC load control strategy in real-time electricity pricing environment”. In: May 2012, pp. 1–6. ISBN: 978-1-4673-1836-5. DOI: [10.1109/EnergyTech.2012.6304636](https://doi.org/10.1109/EnergyTech.2012.6304636) (cit. on pp. 18, 39, 41–43).
- [AGA22] E. Alnazer, I. Georgievski, M. Aiello. *Risk Awareness in HTN Planning*. Apr. 2022 (cit. on pp. 18, 28, 29, 67, 69).
- [Age] I. E. Agency. *Gas Market Report, Q2-2022*. URL: <https://www.iea.org/reports/gas-market-report-q2-2022/executive-summary>. (accessed: 28.06.2022) (cit. on p. 17).
- [APF12] J. Abreu, F. Pereira, P. Ferrão. “Using pattern recognition to identify habitual behavior in residential electricity consumption”. In: *Energy and Buildings* 49 (June 2012), pp. 479–487. DOI: [10.1016/j.enbuild.2012.02.044](https://doi.org/10.1016/j.enbuild.2012.02.044) (cit. on pp. 61, 68).
- [BK04] A. Bouguerra, L. Karlsson. “Hierarchical Task Planning under Uncertainty”. In: (Oct. 2004) (cit. on p. 68).
- [BMB14] A. Buckman, M. Mayfield, S. Beck. “What is a Smart Building?” In: *Smart and Sustainable Built Environment* 3 (Sept. 2014), pp. 92–109. DOI: [10.1108/SASBE-01-2014-0003](https://doi.org/10.1108/SASBE-01-2014-0003) (cit. on pp. 26, 27).
- [CC07] B. Carolis, G. Cozzolongo. “Planning the Behaviour of a Social Robot Acting as a Majordomo in Public Environments”. In: Sept. 2007, pp. 805–812. ISBN: 978-3-540-74781-9. DOI: [10.1007/978-3-540-74782-6\\_72](https://doi.org/10.1007/978-3-540-74782-6_72) (cit. on pp. 18, 68).
- [Che20] A. Cheshmehzangi. “COVID-19 and Household Energy Implications: What are the main Impacts on energy use?” In: *Heliyon* 6 (Oct. 2020), pp. 1–18. DOI: [10.1016/j.heliyon.2020.e05202](https://doi.org/10.1016/j.heliyon.2020.e05202) (cit. on p. 17).
- [Com] E. Commision. *EU Buildings Factsheets*. URL: [https://ec.europa.eu/energy/eu-buildings-factsheets\\_en](https://ec.europa.eu/energy/eu-buildings-factsheets_en). (accessed: 28.06.2022) (cit. on p. 17).
- [Ene] D. Energy. *Recommended Thermostat Settings in the Winter*. URL: <https://www.directenergy.com/learning-center/recommended-thermostat-settings-winter>. (accessed: 03.07.2022) (cit. on p. 51).
- [FWR] D. A. Ozimek. *Future Workforce Report 2021: How Remote Work is Changing Businesses Forever*. URL: <https://www.upwork.com/research/future-workforce-report>. (accessed: 28.06.2022) (cit. on p. 17).
- [GA15] I. Georgievski, M. Aiello. “HTN planning: Overview, comparison, and beyond”. In: *Artificial Intelligence* 222 (Feb. 2015), pp. 124–156. DOI: [10.1016/j.artint.2015.02.002](https://doi.org/10.1016/j.artint.2015.02.002) (cit. on pp. 23, 55).
- [GA16] I. Georgievski, M. Aiello. “Automated Planning for Ubiquitous Computing”. In: *ACM Comput. Surv.* 49 (Dec. 2016). DOI: [10.1145/3004294](https://doi.org/10.1145/3004294) (cit. on pp. 18, 22, 31, 33, 61, 67).

- [GDP+12] I. Georgievski, V. Degeler, G. A. Pagani, T. A. Nguyen, A. Lazovik, M. Aiello. “Optimizing Energy Costs for Offices Connected to the Smart Grid”. In: *Smart Grid, IEEE Transactions on* 3 (Dec. 2012), pp. 2273–2285. DOI: [10.1109/TSG.2012.2218666](https://doi.org/10.1109/TSG.2012.2218666) (cit. on p. 67).
- [Geo15] I. Georgievski. “Coordinating services embedded everywhere via hierarchical planning”. In: 2015 (cit. on pp. 38, 67).
- [GNN+17] I. Georgievski, T. A. Nguyen, F. Nizamic, B. Setz, A. Lazovik, M. Aiello. “Planning meets activity recognition: Service coordination for intelligent buildings”. In: *Pervasive and Mobile Computing* 38 (Feb. 2017). DOI: [10.1016/j.pmcj.2017.02.008](https://doi.org/10.1016/j.pmcj.2017.02.008) (cit. on pp. 18, 23).
- [GNT04a] M. Ghallab, D. Nau, P. Traverso. In: *Automated Planning*. Ed. by M. Ghallab, D. Nau, P. Traverso. The Morgan Kaufmann Series in Artificial Intelligence. Burlington: Morgan Kaufmann, 2004. ISBN: 978-1-55860-856-6. DOI: <https://doi.org/10.1016/B978-155860856-6/50004-1>. URL: <https://www.sciencedirect.com/science/article/pii/B9781558608566500041> (cit. on p. 18).
- [GNT04b] M. Ghallab, D. Nau, P. Traverso. “Chapter 11 - Hierarchical Task Network Planning”. In: *Automated Planning*. Ed. by M. Ghallab, D. Nau, P. Traverso. The Morgan Kaufmann Series in Artificial Intelligence. Burlington: Morgan Kaufmann, 2004, pp. 229–261. ISBN: 978-1-55860-856-6. DOI: <https://doi.org/10.1016/B978-155860856-6/50017-X>. URL: <https://www.sciencedirect.com/science/article/pii/B978155860856650017X> (cit. on pp. 23, 24).
- [Gre] GreenSpec. *Thermal Mass*. URL: <https://www.greenspec.co.uk/building-design/thermal-mass/>. (accessed: 03.07.2022) (cit. on p. 39).
- [Hei20] L. Heiland. *Domain-independent AI planning for demand-side management in office buildings*. 2020 (cit. on p. 38).
- [HMK22] A. Heidari, F. Maréchal, D. Khovalyg. “Reinforcement Learning for proactive operation of residential energy systems by learning stochastic occupant behavior and fluctuating solar energy: Balancing comfort, hygiene and energy use”. In: *Applied Energy* 318 (May 2022). DOI: [10.1016/j.apenergy.2022.119206](https://doi.org/10.1016/j.apenergy.2022.119206) (cit. on p. 68).
- [Hod] U. E. I. A. Hodge T. *Hourly electricity consumption varies throughout the day and across seasons*. URL: <https://www.eia.gov/todayinenergy/detail.php?id=42915>. (accessed: 03.07.2022) (cit. on pp. 35, 36).
- [Hoy16] M. Hoy. “Smart Buildings: An Introduction to the Library of the Future”. In: *Medical reference services quarterly* 35 (July 2016), pp. 326–331. DOI: [10.1080/02763869.2016.1189787](https://doi.org/10.1080/02763869.2016.1189787) (cit. on p. 26).
- [HT] A. Hussain, M. Torres. *Time to pick up pace of dynamic electricity pricing*. URL: <https://www.frontier-economics.com/uk/en/news-and-articles/articles/article-i6106-time-to-pick-up-pace-of-dynamic-electricity-pricing/#>. (accessed: 02.07.2022) (cit. on p. 37).
- [Ilgh06] O. Ilghami. “Documentation for JSHOP2”. In: (Apr. 2006) (cit. on p. 55).
- [JEG] J. E. G. Inc. *What Temperature Should I Set My Air Conditioner in Summer?* URL: <https://justenergy.com/blog/what-temperature-should-i-set-my-air-conditioner-in-summer/>. (accessed: 03.07.2022) (cit. on p. 38).



- [KFS+21] R. Karimi, L. Farahzadi, S. Sepasgozar, S. Sargolzaei, S. Sepasgozar, M. Zareian, A. Nasrolahi. “Smart Built Environment Including Smart Home, Smart Building and Smart City: Definitions and Applied Technologies”. In: Dec. 2021. ISBN: 978-1-83881-141-9. DOI: [10.5772/intechopen.95104](https://doi.org/10.5772/intechopen.95104) (cit. on p. 26).
- [KJU+13] I. Khan, N. Javaid, M. Ullah, A. Mahmood, M. Farooq. “A Survey of Home Energy Management Systems in Future Smart Grid Communications”. In: (July 2013). DOI: [10.1109/BWCCA.2013.80](https://doi.org/10.1109/BWCCA.2013.80) (cit. on pp. 18, 28, 35).
- [LCL16] G. Lobaccaro, S. Carlucci, E. Löfström. “A Review of Systems and Technologies for Smart Homes and Smart Grids”. In: *Energies* 9 (May 2016), pp. 1–33. DOI: [10.3390/en9050348](https://doi.org/10.3390/en9050348) (cit. on p. 28).
- [Lio92] Y.I. Liou. “Knowledge Acquisition: Issues, Techniques and Methodology”. In: *SIGMIS Database* 23.1 (Mar. 1992), pp. 59–64. ISSN: 0095-0033. DOI: [10.1145/134347.134364](https://doi.org/10.1145/134347.134364). URL: <https://doi.org/10.1145/134347.134364> (cit. on p. 31).
- [LJ21] Z. Liu, G. Jiang. “Optimization of intelligent heating ventilation air conditioning system in urban building based on BIM and artificial intelligence technology”. In: *Computer Science and Information Systems* 18 (Jan. 2021), pp. 27–27. DOI: [10.2298/CSIS200901027L](https://doi.org/10.2298/CSIS200901027L) (cit. on p. 40).
- [Mai18] A. f. E. E. e. Maier M. “Metaanalyse: Die Digitalisierung der Energiewende”. In: (Aug. 2018), p. 19 (cit. on pp. 35, 37).
- [Mat19] C. L. Matt Harding Kyle Kettler. “Environmental and Social Benefits of Time of Use Electricity Pricing”. In: (Jan. 2019) (cit. on p. 17).
- [MMA+22] J. Malik, A. Mahdavi, E. Azar, H. chandra putra, C. Berger, C. Andrews, T. Hong. “Ten questions concerning agent-based modeling of occupant behavior for energy and environmental performance of buildings”. In: *Building and Environment* 217 (Mar. 2022). DOI: [10.1016/j.buildenv.2022.109016](https://doi.org/10.1016/j.buildenv.2022.109016) (cit. on p. 68).
- [MOG+10] D. K. Mcglinn, E. O'Neill, A. Gibney, D. O’Sullivan, D. Lewis. “SimCon: A Tool to Support Rapid Evaluation of Smart Building Application Design using Context Simulation and Virtual Reality”. In: *J. UCS* 16 (Jan. 2010), pp. 1992–2018 (cit. on p. 26).
- [MZH] D. Byers. *Mark Zuckerberg: Half of Facebook may work remotely by 2030*. URL: <https://www.nbcnews.com/tech/tech-news/mark-zuckerberg-half-facebook-may-work-remotely-2030-n1212081>. (accessed: 28.06.2022) (cit. on p. 17).
- [Offa] E. S. Office. *Energy consumption in households*. URL: [https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Energy\\_consumption\\_in\\_households](https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Energy_consumption_in_households). (accessed: 31.08.2022) (cit. on p. 38).
- [Offb] E. S. Office. *Energy statistics - an overview*. URL: [https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Energy\\_statistics\\_-\\_an\\_overview#Final\\_energy\\_consumption](https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Energy_statistics_-_an_overview#Final_energy_consumption). (accessed: 28.06.2022) (cit. on p. 17).
- [RC17] J. Reyna, M. Chester. “Energy efficiency to reduce residential electricity and natural gas use under climate change”. In: *Nature Communications* 8 (May 2017), p. 14916. DOI: [10.1038/ncomms14916](https://doi.org/10.1038/ncomms14916) (cit. on p. 17).

## Bibliography

---

- [Ric17] F. Richter. “Hierarchical planning under uncertainty”. en. PhD thesis. Universität Ulm, 2017. DOI: [10.18725/OPARU-5243](https://doi.org/10.18725/OPARU-5243). URL: <https://oparu.uni-ulm.de/xmlui/handle/123456789/5300> (cit. on pp. 18, 28, 68).
- [Sle] SleepAdvisor.org. *What’s The Best Temperature For Sleep?* URL: <https://www.sleepadvisor.org/best-temperature-for-sleep/>. (accessed: 03.07.2022) (cit. on p. 51).
- [Sta21] C. Stanullo. *AI planning for improved ventilation management in buildings*. 2021 (cit. on p. 38).
- [WTS20] J. Wang, C. Tang, L. Song. “Design and Analysis of Optimal Pre-Cooling in Residential Buildings”. In: *Energy and Buildings* 216 (Mar. 2020), p. 109951. DOI: [10.1016/j.enbuild.2020.109951](https://doi.org/10.1016/j.enbuild.2020.109951) (cit. on p. 40).
- [ZZS+21] Z. Zeng, W. Zhang, K. Sun, M. Wei, T. Hong. “Investigation of pre-cooling as a recommended measure to improve residential buildings’ thermal resilience during heat waves”. In: *Building and Environment* 210 (Dec. 2021). DOI: [10.1016/j.buildenv.2021.108694](https://doi.org/10.1016/j.buildenv.2021.108694) (cit. on p. 39).

All links were last followed on September 26, 2022.

# A Predicates for Defining Problem Instances

## Appliances

- (dishwasher ?dw)
- (general\_appliance ?appl)
- (appliance\_policy ?appl ?h\_to\_work ?kWh)
- (to\_schedule ?appl ?start\_h ?stop\_h)
- (ac ?ac)
- (electric\_heating ?eh)

## DAP prices

- (price ?provider ?h ?price ?amount)
- (average\_price\_at\_h ?h ?price)
- (averaged\_price\_min\_max ?min ?max)

## Weather forecast

- (weather\_max\_bounds ?higher\_bound ?lower\_bound)
- (forecast\_tomorrow ?max ?min)

## Storage

- (price\_tolerance\_to\_avoid\_storing ?t)
- (bess\_capacity ?capacity)
- (bought\_energy ?amount ?h)
- (stored\_energy 0 ?h 0 0 0)

## Locally generated solar power

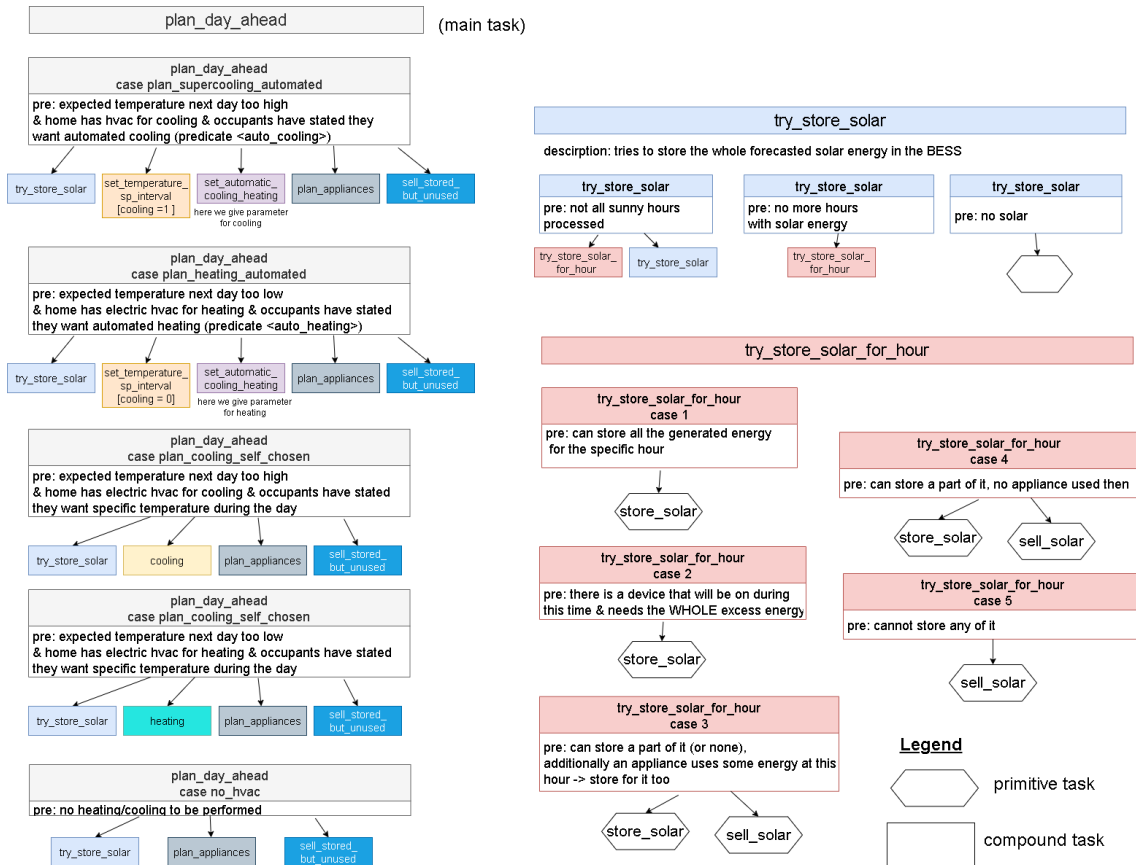
- (hours\_with\_solar ?start\_h ?end\_h)
- (solar ?amount ?h)

## Automated HVAC

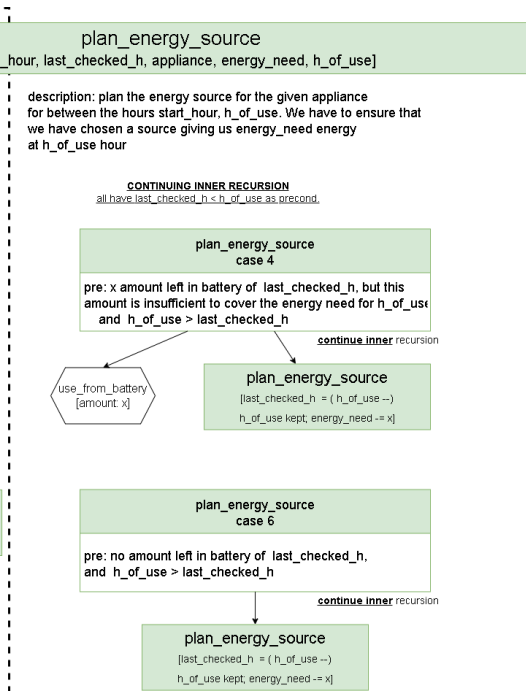
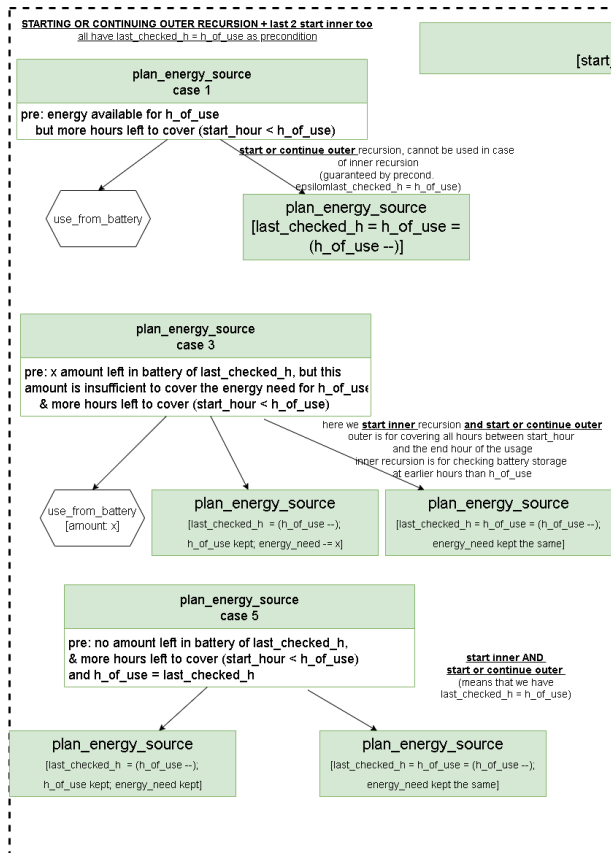
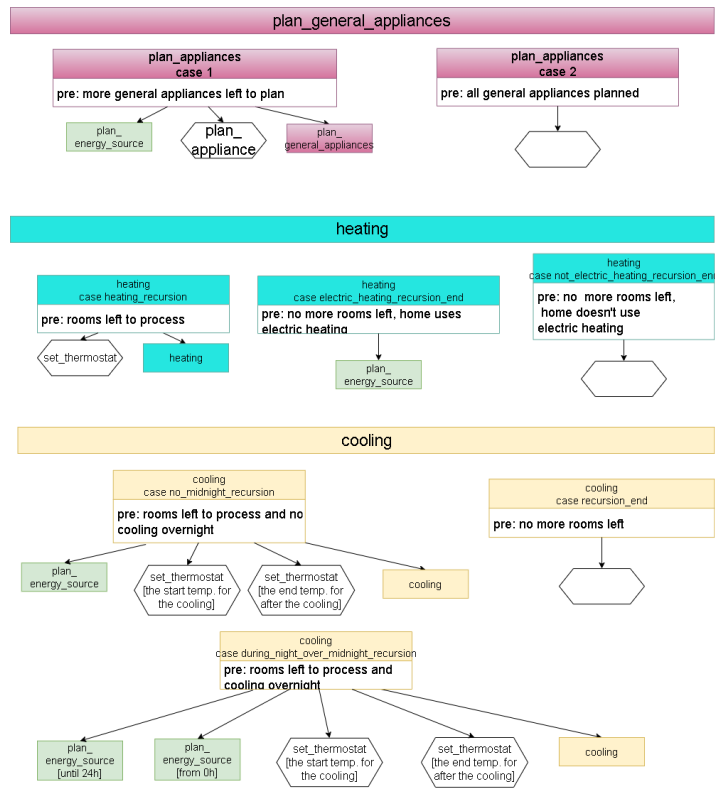
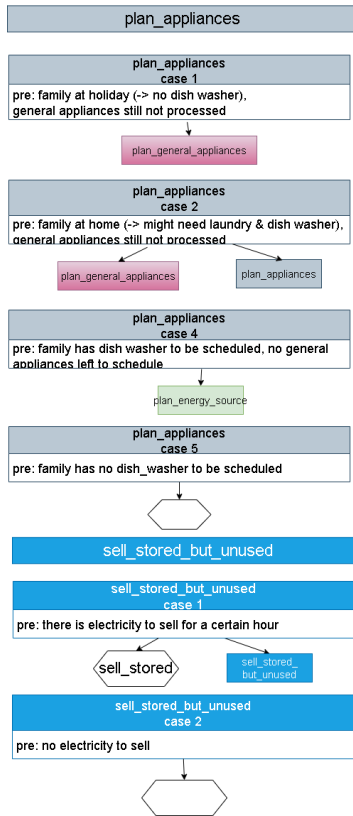
- (auto\_heating)
- (auto\_cooling)
- (room ?room)
- (tolerance\_level ?alpha)
- (user\_temp\_set\_points ?low ?high ?room)
- (wanted\_temperature ?room ?h ?temp) - not for automated HVAC, but for user-chosen thermostat settings

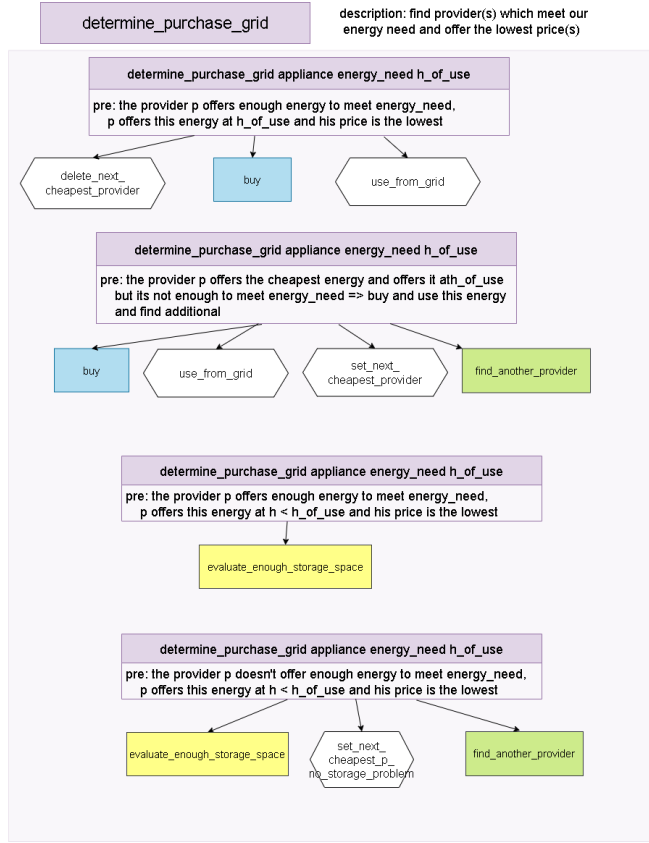
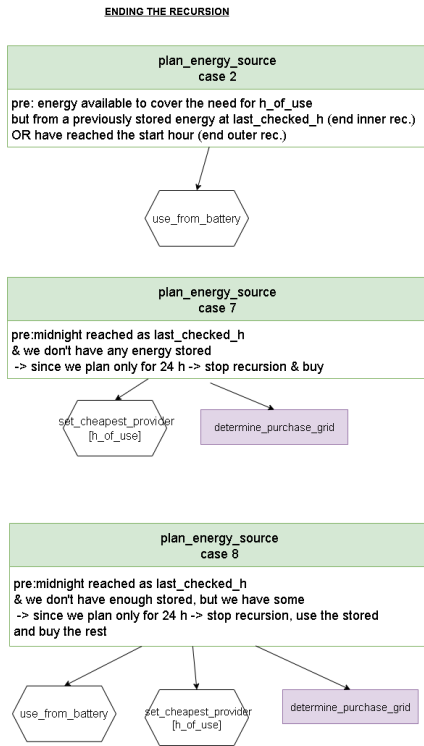


## B Structure of Domain Model



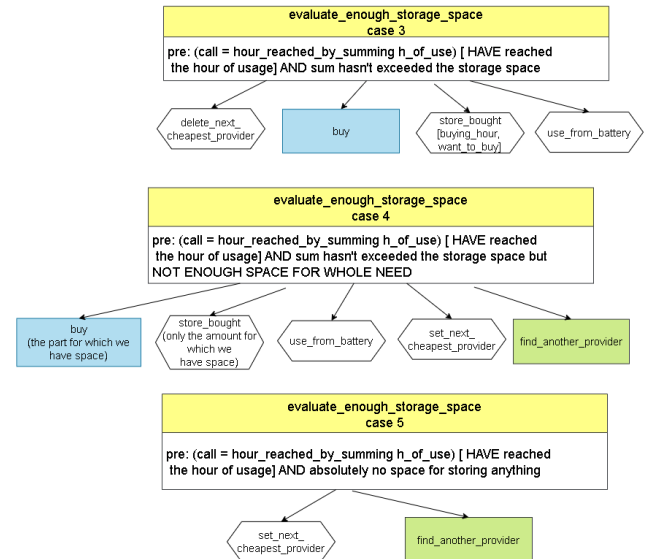
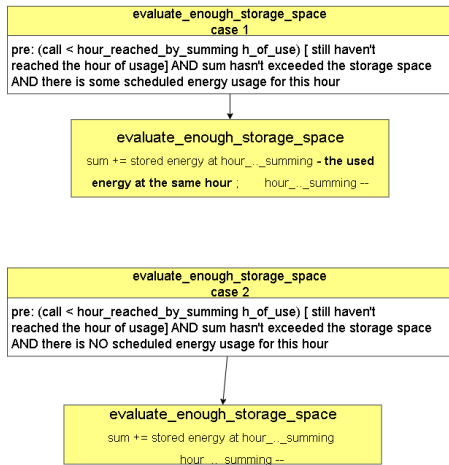
## B Structure of Domain Model



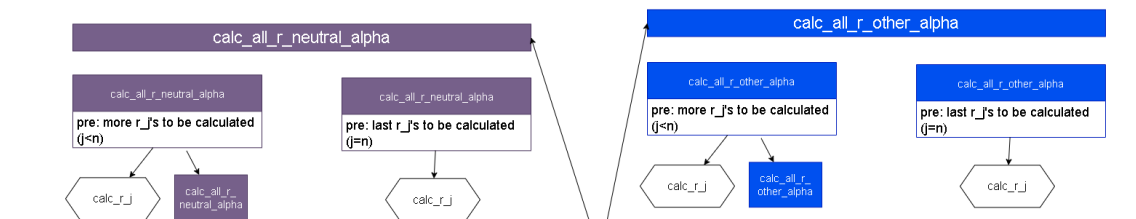
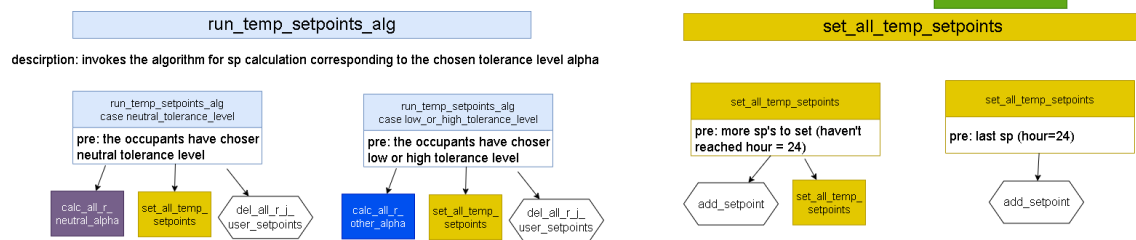
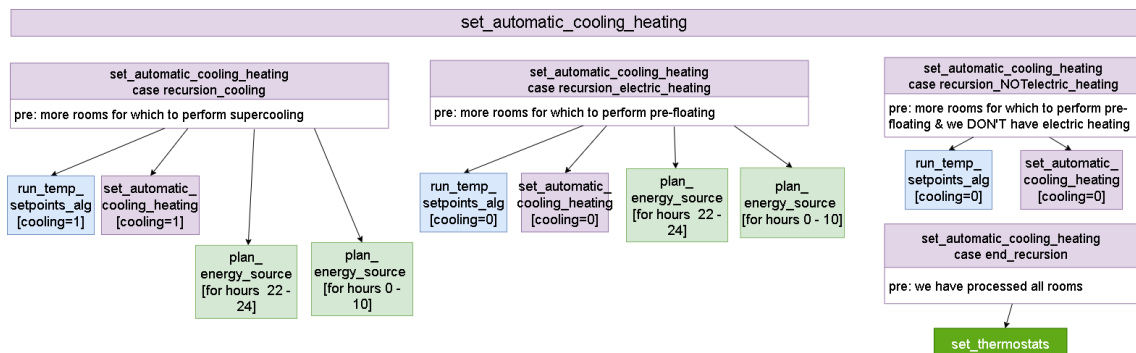
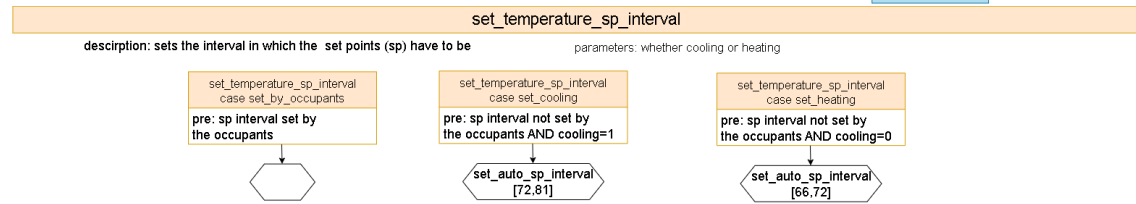
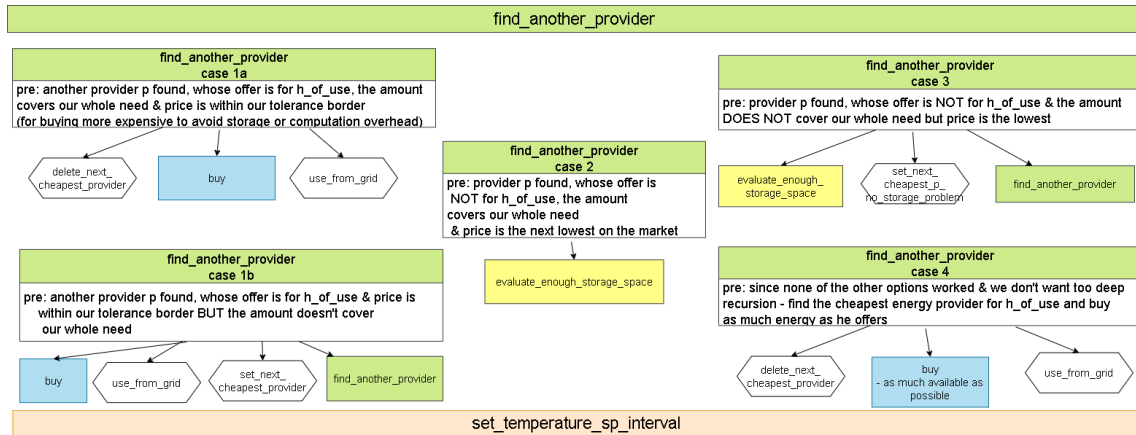


**evaluate\_enough\_storage\_space**

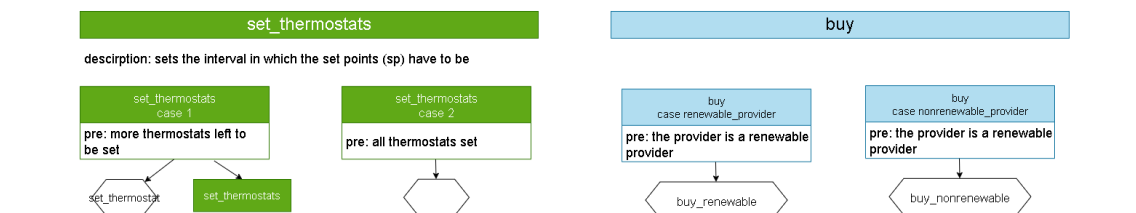
evaluate whether we can buy this energy at the wanted hour and store it (whether enough space)  
 [want\_to\_buy, h\_of\_use, wanted\_provider, sum, hour\_reached\_by\_summing, buying\_hour price appliance]



## B Structure of Domain Model



The difference between those are that we pass different coefficients to calc\_r\_j





# C Problem Instances for Qualitative Evaluation and Produced Plans

## C.1 Highest Automation Level

### C.1.1 Problem Instance

Line 4 is marked, since it is the only line which is not included in the problem instance for the lowest automation level, Appendix C.2.1.

```
(defproblem problem energy_plan_for_smart_home
```

```
(
```

```
(ac ac)
```

```
Line 4: (auto_cooling)
```

```
(weather_max_bounds 79 50)
```

```
(forecast_tomorrow 90 77)
```

```
(room bedroom1)
```

```
(appliance_policy ac not_given 1.5)
```

```
(tolerance_level 0)
```

```
(price_tolerance_to_avoid_storing 1.12)
```

```
(bess_capacity 0)
```

```
(bought_energy 0 0)
```

```
(bought_energy 0 1)
```

```
(bought_energy 0 2)
```

```
(bought_energy 0 3)
```

```
(bought_energy 0 4)
```

```
(bought_energy 0 5)
```

```
(bought_energy 0 6)
```

```
(bought_energy 0 7)
```

```
(bought_energy 0 8)
```

(bought\_energy 0 9)  
(bought\_energy 0 10)  
(bought\_energy 0 11)  
(bought\_energy 0 12)  
(bought\_energy 0 13)  
(bought\_energy 0 14)  
(bought\_energy 0 15)  
(bought\_energy 0 16)  
(bought\_energy 0 17)  
(bought\_energy 0 18)  
(bought\_energy 0 19)  
(bought\_energy 0 20)  
(bought\_energy 0 21)  
(bought\_energy 0 22)  
(bought\_energy 0 23)  
(bought\_energy 0 24)  
(stored\_energy 0 0 0 0 0)  
(stored\_energy 0 1 0 0 0)  
(stored\_energy 0 2 0 0 0)  
(stored\_energy 0 3 0 0 0)  
(stored\_energy 0 4 0 0 0)  
(stored\_energy 0 5 0 0 0)  
(stored\_energy 0 6 0 0 0)  
(stored\_energy 0 7 0 0 0)  
(stored\_energy 0 8 0 0 0)  
(stored\_energy 0 9 0 0 0)  
(stored\_energy 0 10 0 0 0)  
(stored\_energy 0 11 0 0 0)  
(stored\_energy 0 12 0 0 0)  
(stored\_energy 0 13 0 0 0)  
(stored\_energy 0 14 0 0 0)

(stored\_energy 0 15 0 0 0)  
(stored\_energy 0 16 0 0 0)  
(stored\_energy 0 17 0 0 0)  
(stored\_energy 0 18 0 0 0)  
(stored\_energy 0 19 0 0 0)  
(stored\_energy 0 20 0 0 0)  
(stored\_energy 0 21 0 0 0)  
(stored\_energy 0 22 0 0 0)  
(stored\_energy 0 23 0 0 0)  
(stored\_energy 0 24 0 0 0)  
(hours\_with\_solar 0 1)  
(solar 0 0)  
(price provider1 0 0.0818 25.5)  
(price provider2 0 0.0833 77.5)  
(price provider3 0 0.039 65.5)  
(average\_price\_at\_h 0 0.05)  
(price provider1 1 0.047 28.5)  
(price provider2 1 0.0585 74.5)  
(price provider3 1 0.047 65.5)  
(average\_price\_at\_h 1 0.0416)  
(price provider1 2 0.033 25.5)  
(price provider2 2 0.052 70.5)  
(price provider3 2 0.056 85.5)  
(average\_price\_at\_h 2 0.037)  
(price provider1 3 0.05 58.5)  
(price provider2 3 0.042 47.5)  
(price provider3 3 0.025 38.5)  
(average\_price\_at\_h 3 0.033)  
(price provider1 4 0.048 41.5)  
(price provider2 4 0.029 65.5)  
(price provider3 4 0.035 61.5)

(average\_price\_at\_h 4 0.032)  
(price provider1 5 0.0227 15.5)  
(price provider2 5 0.045 23.5)  
(price provider3 5 0.043 10.5)  
(average\_price\_at\_h 5 0.032)  
(price provider1 6 0.0351 49.5)  
(price provider2 6 0.0489 98.5)  
(price provider3 6 0.0264 58.5)  
(average\_price\_at\_h 6 0.0336)  
(price provider1 7 0.05174 20.5)  
(price provider2 7 0.0494 28.5)  
(price provider3 7 0.0416 74.5)  
(average\_price\_at\_h 7 0.0367)  
(price provider1 8 0.0733 93.5)  
(price provider2 8 0.0729 13.5)  
(price provider3 8 0.0789 37.5)  
(average\_price\_at\_h 8 0.0496)  
(price provider1 9 0.0592 1.5)  
(price provider2 9 0.0696 19.5)  
(price provider3 9 0.0703 77.5)  
(average\_price\_at\_h 9 0.0572)  
(price provider1 10 0.0733 10.5)  
(price provider2 10 0.1153 98.5)  
(price provider3 10 0.068 65.5)  
(average\_price\_at\_h 10 0.0709)  
(price provider1 11 0.1433 20.5)  
(price provider2 11 0.15 34.5)  
(price provider3 11 0.0744 40.5)  
(average\_price\_at\_h 11 0.092)  
(price provider1 12 0.106 52.5)  
(price provider2 12 0.178 15.5)

(price provider3 12 0.0929 19.5)  
(average\_price\_at\_h 12 0.1154)  
(price provider1 13 0.1743 95.5)  
(price provider2 13 0.2128 65.5)  
(price provider3 13 0.217 92.5)  
(average\_price\_at\_h 13 0.1518)  
(price provider1 14 0.3403 26.5)  
(price provider2 14 0.239 53.5)  
(price provider3 14 0.278 84.5)  
(average\_price\_at\_h 14 0.25)  
(price provider1 15 0.392 32.5)  
(price provider2 15 0.294 33.5)  
(price provider3 15 0.391 34.5)  
(average\_price\_at\_h 15 0.2927)  
(price provider1 16 0.4552 84.5)  
(price provider2 16 0.392 65.5)  
(price provider3 16 0.4574 38.5)  
(average\_price\_at\_h 16 0.308)  
(price provider1 17 0.2264 86.5)  
(price provider2 17 0.3317 97.5)  
(price provider3 17 0.2498 91.5)  
(average\_price\_at\_h 17 0.319)  
(price provider1 18 0.4 23.5)  
(price provider2 18 0.413 93.5)  
(price provider3 18 0.343 91.5)  
(average\_price\_at\_h 18 0.3038)  
(price provider1 19 0.254 69.5)  
(price provider2 19 0.337 32.5)  
(price provider3 19 0.199 53.5)  
(average\_price\_at\_h 19 0.221)  
(price provider1 20 0.1791 13.5)

```
(price provider2 20 0.1494 38.5)
(price provider3 20 0.1269 35.5)
(average_price_at_h 20 0.13604)
(price provider1 21 0.1534 21.5)
(price provider2 21 0.1656 3.5)
(price provider3 21 0.17 71.5)
(average_price_at_h 21 0.127)
(price provider1 22 0.102 81.5)
(price provider2 22 0.152 3.5)
(price provider3 22 0.136 69.5)
(average_price_at_h 22 0.107)
(price provider1 23 0.108 75.5)
(price provider2 23 0.0607 65.5)
(price provider3 23 0.11647 70.5)
(average_price_at_h 23 0.0756)
(price provider1 24 0.052 76.5)
(price provider2 24 0.049 84.5)
(price provider3 24 0.096 60.5)
(average_price_at_h 24 0.0569)
(averaged_price_min_max 0.03199 0.31993)
(renewable_provider provider2)
)
(:unordered
(plan_day_ahead)
))
```

### C.1.2 Post-processed Plan

Plan cost: 1.0132499999999998

(!buy\_renewable provider2 24.0 1.5 0.049)

(!use\_from\_grid 24.0 1.5 ac)

(!buy\_renewable provider2 23.0 1.5 0.0607)

(!use\_from\_grid 23.0 1.5 ac)

(!buy\_nonrenewable provider1 22.0 1.5 0.102)

(!use\_from\_grid 22.0 1.5 ac)

(!buy\_nonrenewable provider3 10.0 1.5 0.068)

(!use\_from\_grid 10.0 1.5 ac)

(!buy\_nonrenewable provider1 9.0 1.5 0.0592)

(!use\_from\_grid 9.0 1.5 ac)

(!buy\_renewable provider2 8.0 1.5 0.0729)

(!use\_from\_grid 8.0 1.5 ac)

(!buy\_nonrenewable provider3 7.0 1.5 0.0416)

(!use\_from\_grid 7.0 1.5 ac)

(!buy\_nonrenewable provider3 6.0 1.5 0.0264)

(!use\_from\_grid 6.0 1.5 ac)

(!buy\_nonrenewable provider1 5.0 1.5 0.0227)

(!use\_from\_grid 5.0 1.5 ac)

(!buy\_renewable provider2 4.0 1.5 0.029)

(!use\_from\_grid 4.0 1.5 ac)

(!buy\_nonrenewable provider3 3.0 1.5 0.025)

(!use\_from\_grid 3.0 1.5 ac)

(!buy\_nonrenewable provider1 2.0 1.5 0.033)

(!use\_from\_grid 2.0 1.5 ac)

(!buy\_nonrenewable provider1 1.0 1.5 0.047)

(!use\_from\_grid 1.0 1.5 ac)

(!buy\_nonrenewable provider3 0.0 1.5 0.039)

(!use\_from\_grid 0.0 1.5 ac)

(!set\_thermostat 1.0 72.0 bedroom1)

(!set\_thermostat 2.0 72.0 bedroom1)  
(!set\_thermostat 3.0 72.0 bedroom1)  
(!set\_thermostat 4.0 72.0 bedroom1)  
(!set\_thermostat 5.0 72.0 bedroom1)  
(!set\_thermostat 6.0 72.0 bedroom1)  
(!set\_thermostat 7.0 72.0 bedroom1)  
(!set\_thermostat 8.0 72.0 bedroom1)  
(!set\_thermostat 9.0 72.0 bedroom1)  
(!set\_thermostat 10.0 73.0 bedroom1)  
(!set\_thermostat 11.0 74.0 bedroom1)  
(!set\_thermostat 12.0 74.0 bedroom1)  
(!set\_thermostat 13.0 76.0 bedroom1)  
(!set\_thermostat 14.0 79.0 bedroom1)  
(!set\_thermostat 15.0 81.0 bedroom1)  
(!set\_thermostat 16.0 81.0 bedroom1)  
(!set\_thermostat 17.0 81.0 bedroom1)  
(!set\_thermostat 18.0 81.0 bedroom1)  
(!set\_thermostat 19.0 78.0 bedroom1)  
(!set\_thermostat 20.0 75.0 bedroom1)  
(!set\_thermostat 21.0 75.0 bedroom1)  
(!set\_thermostat 22.0 74.0 bedroom1)  
(!set\_thermostat 23.0 73.0 bedroom1)  
(!set\_thermostat 24.0 72.0 bedroom1)

Time Used = 0.047



## C.2 Lowest Automation Level

### C.2.1 Problem Instance

Same as Appendix C.1.1 but instead of [Line 4](#):

(wanted\_temperature bedroom1 10 77)

(wanted\_temperature bedroom1 18 79)

(appliance\_policy ac not\_given 1.5)

### C.2.2 Post-processed Plan

Plan cost: 2.856

(!buy\_nonrenewable provider3 18.0 1.5 0.343)

(!use\_from\_grid 18.0 1.5 ac)

(!buy\_nonrenewable provider1 17.0 1.5 0.2264)

(!use\_from\_grid 17.0 1.5 ac)

(!buy\_renewable provider2 16.0 1.5 0.392)

(!use\_from\_grid 16.0 1.5 ac)

(!buy\_renewable provider2 15.0 1.5 0.294)

(!use\_from\_grid 15.0 1.5 ac)

(!buy\_renewable provider2 14.0 1.5 0.239)

(!use\_from\_grid 14.0 1.5 ac)

(!buy\_nonrenewable provider1 13.0 1.5 0.1743)

(!use\_from\_grid 13.0 1.5 ac)

(!buy\_nonrenewable provider3 12.0 1.5 0.0929)

(!use\_from\_grid 12.0 1.5 ac)

(!buy\_nonrenewable provider3 11.0 1.5 0.0744)

(!use\_from\_grid 11.0 1.5 ac)

(!buy\_nonrenewable provider3 10.0 1.5 0.068)

(!use\_from\_grid 10.0 1.5 ac)

(!set\_thermostat 10.0 77.0 bedroom1)


(!set\_thermostat 18.0 79.0 bedroom1)

Time Used = 0.052



## Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

Stuttgart, 28.09.2022, 

place, date, signature