UiT
THE ARCTIC
UNIVERSITY
OF NORWAY

Faculty of Engineering Science and Technology

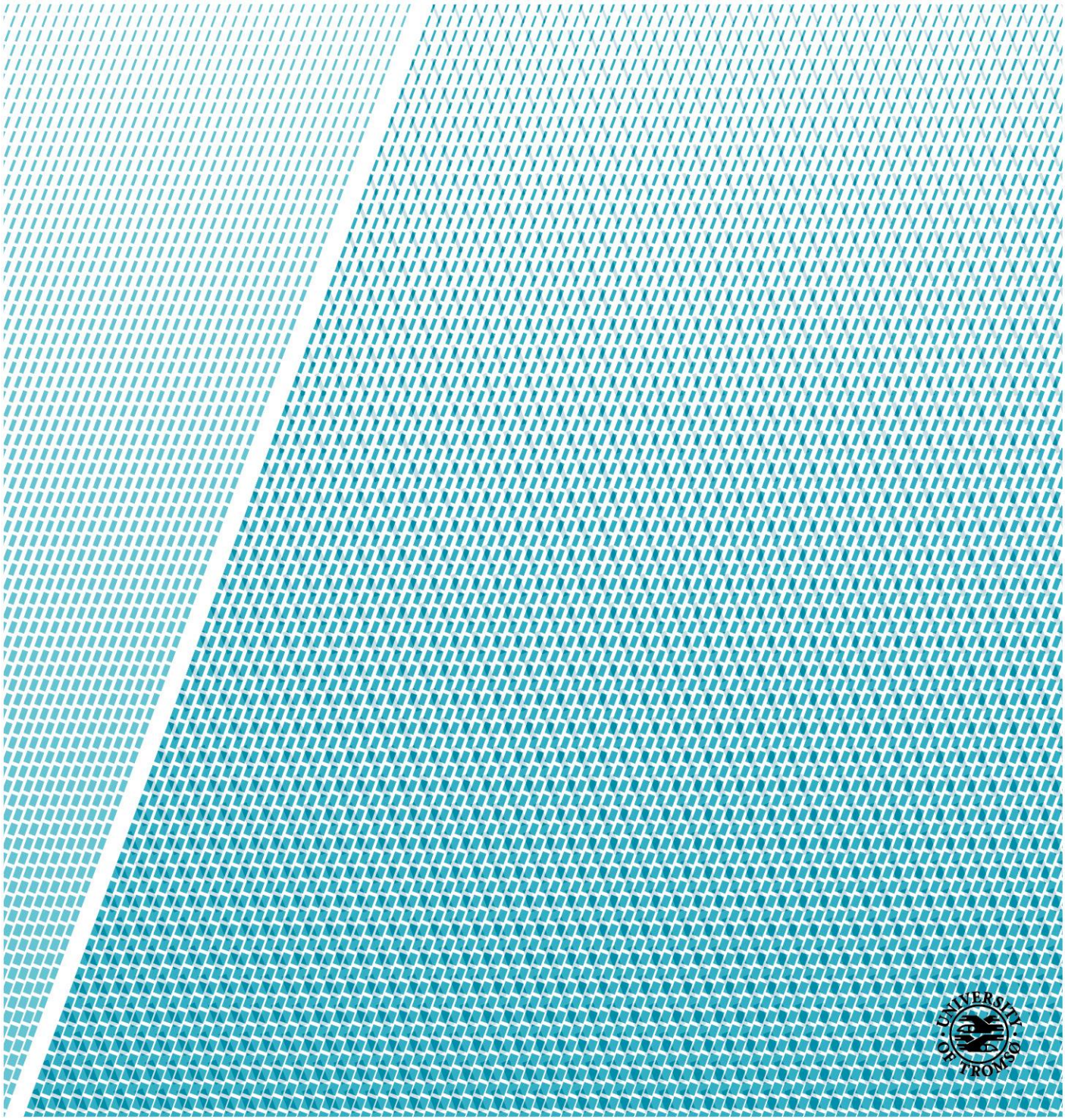
Department of Computer Science and Computational Engineering

Distributed Management of Resources in a Smart City using Multi-Agent Systems (MAS)

—

Igor Molchanov

Master's thesis in Computer Science - June 2018



Title: Distributed management of resources in a Smart City using Multi-Agent Systems (MAS)

Author: Igor Molchanov

Date: June 2018

Classification: Open.

Pages: 77

Attachments: Zip file

Department: Department of Computer Science and Computational Engineering

Study: Master of Science, Computer Science

Student number: 166120

Course code: SHO6264 Diploma Thesis - M-IT

Supervisor: Bernt A. Bremdal

Principal: UiT - The Arctic University of Norway (Campus Narvik)

Principal contact: Bernt A. Bremdal

Keywords: Smart City, multi-agent systems, smart house, smart parking, smart traffic, common good distribution, local market design, agent behavior

Abstract (English):This document describes study of distributed management of resources in the context of Smart City with support of multi-agent systems. The investigated points include theoretical concepts of Smart City and application of multi-agent systems, decentralized and centralized designs for agent-based solutions and aspects of interactions between different self-interested agents. The document explores design of an agent-based solutions for the set of proposed problems related to Smart City environment with the emphasis on sharing common good, models of agent interactions within the modeled environments and possibilities of multi-agent approaches in terms of collective problem-solving, adaptability and learning proficiency.

Acknowledgements

I want to thank my supervisor Bernt A. Bremdal and my co-advisor Kristoffer Tangrand for their suggestions and support which I got from them during the thesis work. I also want to thank Aleksei Degtiarev and Nikita Shvetsov and all my classmates for valuable insights and advises on design of my solutions.

In addition, I want to express my graditude to Tatiana Kravetc for the suggestions and advices which helped me to go furhter in my work.

Finally I would like to sincerely thank my family and friends for supporting me throughout this study and Master thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem description

In this master thesis, we present the research of the use of multi-agent systems in the context of Smart City and Internet of Things. Smart City concept represents an innovative way of thinking about urban space by presenting a model that integrates renewable energy resources, energy efficiency and smart systems for a living. In this new paradigm bottom-up approach plays an important role. This implies system structure, where offers are provided by different facilities with sensors and controllers using a variety of protocols and base technologies. The type of non-centralized development, common for the Smart City, corresponds to essential aspects of multi-agent systems.

This research project is focused on theoretical aspects of multi-agent systems application within the context of Smart City. In the thesis we investigate general aspects of Smart City environment, the ideas behind this concept and create a set of models for particular problems in order to find an answer how the distributed agent-based system which consists of low or zero-intelligent self-interested agents can manage different tasks and demonstrate a certain level of intelligence in terms of problem-solving capabilities, adaptability or learning proficiency.

The fundamental issue of this project relates how different self-interested agents which can manage different sensors and actuators and have to fulfill certain goals, can coexist and interact with each other with a goal to find ways to share a common good and distribute limited resource such as renewable energy. Agents can have parallel as well as opposing needs.

In our work we emphasize the following goals:

1. Exploring theoretical aspects of Smart City concept and application of multi-agent systems within Smart City environment

2. Design of system architecture and different structures of agent interactions
3. Creation of models and implementation of several approaches for them, studying the possibilities of applying similar methods for different cases and analyzing the results

Works by Roscia et al.[1], Longo et al.[2], as well as by Lom and Pribyl[3], are considered as a starting point of the study. In order to create models for considered problems, different ways of implementation were investigated. This includes studying different frameworks which are used for design and creation of multi-agent systems and simulation models. Information about investigated frameworks can be found in chapter 2.

1.2 Contribution and delimitation

As the amount of aspects which can be considered in terms of Smart City environment is considerably large, for our study we decided to scale the area of Smart City and consider the following set of problems:

- Smart house
- Smart parking and cars
- Smart traffic with toll station and barriers

These problems represent an examples of how multi-agent system can be applied in different areas. For the considered problems, we create models which consist of a number of agents and study their interactions in terms of learning and problem-solving capabilities.

For each model, we implement three approaches with the goal to investigate the possibilities of different structures for multi-agent systems. Implemented approaches include decentralized market-based approaches with zero-intelligent agents and learning agents as well as centralized approach with a controlling agent which manages the system.

1.3 Smart City concept

One of the key concepts of this thesis is Smart City and its domains. Although the concept of Smart City is quite common and mentioned in a large number of studies connected to different areas of innovations and urban development, it is hard to find a precise definition. Considering concepts of European Smart Cities, increasing quality of life, use of renewable energy sources

and developing the economy based on knowledge and innovations are some of the main goals which can lead to the development of Smart City[4].

One of the common themes in the studies of Smart City concept is technology. Use of Information Technology and Communication (ICT) is often considered as one of the most important aspects allowing the design of the innovative environments, which are often connected to the concepts of Smart City. For example, according to [1], we can define Smart City as place or territory where the use of the human and natural resources is planned and properly managed through the various integrated ICT structures. This provides an opportunities to create an ecosystem with intelligent and efficient ways of distribution of resources.

Considering structure of Smart City, we can also refer to [2] and define Smart City as a complex system which consists of many different subsystems integrated in the environment and connected together through the use of ICT and various concepts such as Internet of Things[5] in order to preserve safety and stability of work. Each subsystem or a domain of the Smart City represents a key element of the environment and can be also divided into smaller systems. With references to both [2] and the book by V. Angelakis et al.[4], we can identify the following general domains, which also represent key areas and main characteristics of the Smart City:

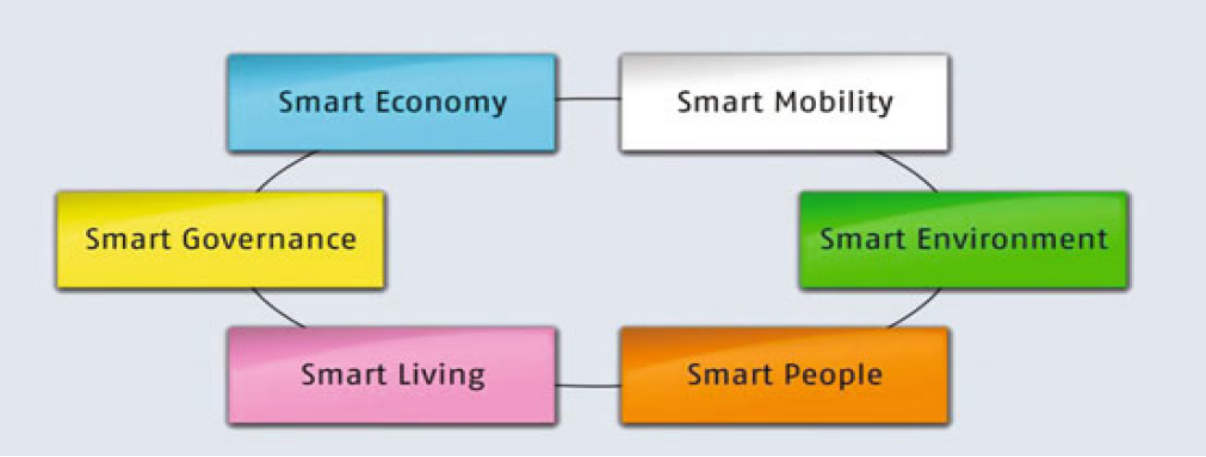


FIGURE 1.1: 6 characteristics of Smart City.

Source: Designing, Developing, and Facilitating Smart Cities by V. Angelakis et al[4].

Each of these characteristics is connected to one of the essential aspects of the Smart City. Within described general domains number of different aspects of Smart City such as transport and logistics, energy and smart buildings, urban public safety, smart health etc., can also be considered and divided further as every such aspect represents an important part of the system and also includes a variety of elements. This characterizes Smart City as a complex system which consists of a large number of subsystems.

With this, it can be said that even the small element of this system can be connected to several different domains and be also represented as a subsystem in a smaller scale. For example, one smart house can be considered as a system of particular elements and be a part of a larger system which can consist of several smart houses connected to the grid. Such system will also be connected to different domains such as Smart Environment and Smart Living.

According to [3], we can say that Smart City includes great number of interconnecting systems where each task can involve interactions and sharing data between different systems. In this environment, every process, for instance, constructing of the shopping center, can involve a large number of dependencies from different areas. Illustration of this principle can be seen on the figure 1.2.

FIGURE 1.2: Construction of shopping center in a Smart City.

Source: Modeling of Smart City Building blocks using Multi-Agent Systems by Lom and Pribyl[3].

We can say that in Smart City every problem, be it building a house or management of traffic flow, can be connected to different areas. This also implies that even particular aspect of Smart City can be reflected in a distributed manner.

Another important aspect of Smart City is the Internet of Things (IoT). In [5] we can find the following definition of IoT as an extensive network created by different sensors and actuators, which exists across all areas of the modern-day life. Integration of sensors in the environment give possibilities for the information to be shared across different platforms in order to develop a common operating space. If we try to define this concept more user-centric, we can refer to another definition from the mentioned paper, which describes IoT as an interconnection of sensors and actuators with the ability of share information across different platforms through the unified framework, creating a common environment for distributed applications.

1.4 Multi-agent systems in Smart City

As mentioned in the paper by Julien Nigon et al.[6], we can consider Smart City as a dynamic system with a complex variety of data flows, which include the information generated by different elements, such as generating and consuming devices. In such environment full understanding and use of produced data flows could be very hard to achieve using ordinary computational approaches.

This type of highly distributed system, which can include a great number of different elements, provides many challenges such as non-linearity, openness, heterogeneity and large-scale data. Many areas of Smart City system such as parking, traffic management, and smart house heating represents non-linear systems which could need sophisticated machine learning algorithms in order to manage full contextual data and non-linear patterns[7].

In this type of system, it is possible for the elements to dynamically enter and exit the system. With the use of IoT, Smart Cities becomes inherently open[8]. New devices and new sources of contextual data could be included continuously and old ones could be removed. For the intelligent algorithm designed to handle data in Smart City environment, it is important to freely operate with new sources and remove old ones. For instance, if we consider smart house environment, the system responsible for balancing energy consumption and comfort should be flexible and capable of reacting to environmental changes. In case of appearing new data sources, it should be possible for the system to include them in decision-making process. Such open structure could be difficult to achieve even with applying machine learning algorithms, such as neural networks or evolutionary algorithms.

Another important aspects are heterogeneity of data and large-scale data flows. Smart City system can generate a wide range of heterogeneous data. For instance, a traffic light control system for the Smart City would have to process output from various sensors of all the roads and crossroads to take multiple decisions about which light should be turned on or off. In order to optimize the performance of decision-making process for this kind of problem, distributed control should be favored over a central decision-making process[9], as every such decision should be at least partially independent. According to mentioned earlier, we can assume that the solution should be based on local decisions, made by the autonomous entities.

Considering different aspects of Smart City as a highly dynamic system with heterogeneous data flows, use of distributed multi-agent systems could be considered as a preferable solution for problem-solving in such type of environment. In [6] multi-agent systems (MAS) defined as systems which consist of multiple interacting and autonomous entities, able to make decisions within a common environment. MAS provide a methodological way to model and study complex systems with a bottom-up approach.

The MAS paradigm emphasizes the design of agents and their collective behaviors which lead to the solution of a particular task. It allows expressing a large variety of problems with a focus on the autonomous entities as elements of the problem environment and their interactions. Therefore, MAS paradigm can be used in different areas such as sociology, biology or engineering, in order to model, explore or simulate different complex problems for each of those areas. Natural aspects of MAS such as task distribution between different agents, adaptive behavior which emerges through the agent interactions within the MAS and availability of decentralized control and decision-making makes MAS suitable to overcome heterogeneity and complexity of Smart City context and adapt to challenges provided by this environment.

Chapter 2

State-of-the-art

2.1 Analysis of problem area

Smart City concept and its aspects include many areas such as innovative ways of energy management, smart transport, information security and many more. In consideration of Smart City features and possible problems which can occur in Smart City environment, many different studies were conducted.

The concepts which were studied include such aspects as smart buildings, smart street and smart light, smart house, smart parking, smart traffic, smart grid etc. Example of solutions connected to smart buildings and offices presented in a paper by Paul Davidsson and Magnus Boman [10]. In paper by W. Li et al.[11] presented one of various solutions for optimization of smart house environment. Example of traffic flow optimization problem is described in a paper by Erwin Walraven et al.[12]. Study of solutions for the smart grid problem presented in a paper by Ghezlane Halhoul Merabet et al.[13]. Within these concepts usage of multi-agent systems can be considered as one of frequently applied methods.

For instance, if we consider the concept of the smart street and smart lamp, described in [3], it can be said that distributed decision-making capabilities, provided by MAS can take an essential part in the design of system structure. In many cases concerning Smart City environment or one of its aspects, usage of MAS was proven to have the capabilities for optimization and intelligent problem-solving.

Application of MAS not limited to scientific simulations and modeling. It is also applied in different projects in order to improve theoretical and practical aspects of designing and implementation of complex systems.

Examples of such projects are alternative airline flight bidding prototype system, called SARDINE (System for Airline Reservations Demonstrating the Integration of Negotiation and Evaluation), which serves to improve the process of flight reservations. The SARDINE system uses software agents to coordinate the preferences and interests of each involved side. Using bidding system, it allows flexible reservation of available flights based on buyer's preferences and individual bids[14]. Another sphere of MAS applications is smart commerce. Various systems and prototypes, such as eMediator system, mentioned in the paper by Mihaela Oprea, or Robin system[15] were developed using multi-agent systems.

With the reference to the mentioned material, it can be concluded that multi-agent systems represent a universal tool which can be used in many different areas and is especially useful for simulations design and modeling of complex systems and innovative smart solutions.

2.2 Previous work

In relation to smart house, smart parking and smart traffic problems, which are considered as an example in this project, a number of different approaches with intellectual problem-solving capabilities, including applying of MAS, was investigated.

Smart house in many cases is considered as a complex system of smart agents with different goals. In such system, almost every element from heaters to TV and kitchenware can be considered as a smart agent.

In [11], smart house environment is considered as a smart network which includes a number of software agents. In this paper, agents are designed with a bottom-up approach and represent different devices which can be connected with each other and create a network system with capabilities to react on the environmental changes and solve different problems. The problem which is considered in this paper represents one of the common problems for smart house environment which is energy distribution. Agents are presented as demand and supply side of the smart house. However, this paper is focused more on communications between software agents as a part of the smart network with the hierarchical structure. In this paper, agents are presented as elements of the system without self-interest aspects and main decision-making function is done by a set of management agents.

Another aspect which is commonly considered in smart house environment is the management of different systems such as heating and light. Different concepts related to using distributed agent-based systems in the field of management of heating systems, light and electricity are described in studies such as [10], as well as many other studies.

These studies demonstrate possibilities of MAS in the management of comfort systems but they are focusing more on particular aspects of the smart house. In case of [10], the focus is more on modeling of embedded systems for maintaining the comfort level and less on agent interactions. In this paper, only two MAS behavior models are considered without such aspects as learning and market-based interactions.

Aspects of the smart distributed environment such as the usage of wireless networks with sets of different sensors and actuators in relation to smart parking also considered in the number of papers such as paper by Trista Lin et al.[16]. In this paper review of different solutions for smart parking development is presented. This article serves more as a review of possible smart parking solutions and does not reflect more aspects of multi-agent interactions.

Example of applying MAS and reinforcement learning in terms of smart traffic problem presented in [12]. This paper demonstrates the concept of traffic flow optimization using Markov Decision Process with learning agents and sectioned road. In this case, authors emphasize specific aspects of applying reinforcement learning for the traffic congestion problem with traffic prediction using a neural network. For this kind of problem market-based approach could also be applied as demonstrated in a paper by J. Kozlak and M. Zabinska[17]. However, in this paper, the problem is defined as the problem of route search for the agents with two type of agents acting with predefined policies. For the agents in this paper, only one behavior model is considered.

Most of the mentioned papers consider only specific aspects of considered problems while in our study we try to consider set of different problems in the more general way and apply a number of approaches to demonstrate possibilities and different features of MAS as well as flexibility which can be provided by agent-based systems. We also consider the distribution of common good as one of the key aspects of all our problems.

Regarding different approaches which were implemented in our work, we should mention that in the studies which were investigated, different architectures of multi-agent interactions were described. Most of them have a hierarchical structure with several layers, represented by the agents, for instance, the architecture described in [11], or market-based architecture with demand aggregation agents, described the paper by Baptiste Feron and Antonello Monti [18]. It can be said that with bottom-up approach decentralized solutions can be designed in a number of ways and even include complex networks composed of different agents. Although the architectures can vary, the main aspect in each of them is the same and related to interactions between different agents, which results in finding the solutions and maintaining work of the system.

For our decentralized approaches, we emphasize this aspect and use a market model based on the decentralized market design described in a paper by Esther Mengelkamp et al.[19].

In distributed market models with a number of agents, different behavior models for the agents can be considered. In our work we consider interactions between zero-intelligent agents in our market model as one of the approaches applied for the studied set of problems. The theory behind zero-intelligent (ZI) market is described by Dhananjay K. Gode and Shyam Sunder in [20]. The market model includes ZI agents which had some budget constraints to follow and specific restrictions for the trading process in order not to trade in a way that would lose them money.

As a result of the experiments conducted for this market model, it was concluded that the market efficiency for ZI agents was very high and the prices often converge to the standard equilibrium price. This result demonstrated that even agents with simple behavior models can be very efficient[21].

The centralized approach is focused more on decisions done by the one controlling agent, responsible for finding the solution to the problem. In order to model such agent, different ways can be applied. One of the possible ways is to use set of defined reactive policies for the agent, but in this case, it could cause lack flexibility. Another approach which can provide more capabilities for the agent to adapt to different situations is to use genetic algorithms.

Methods based on genetic and evolutionary algorithms for multi-objective optimization could be applied in many different fields such as reconfiguration of power distribution system, described in a paper by Bogdan Tomoiag et al.[22] and optimization of hybrid energy systems, described in a paper by Myeong Jin Ko et al.[23]. Similar methods could also be used to optimize energy consumptions in a smart house environment, as described in a paper by Nadeem Javaid et al.[24].

2.3 Instruments and methods

The variety of instruments for design and development of multi-agent systems is large and still continue to expand. From the vast amount of different methods, we can distinguish general purpose frameworks and special purpose frameworks. In this section we present information about some of the multi-agent frameworks which were investigated and tested during work on this thesis. Presented information about the frameworks is based on official documentation and review of agent platforms done by Florin Leon et al.[25].

General purpose frameworks include frameworks for modeling and simulations of multi-agent environments such as Repast, Mesa, Gama, NetLogo, MASON, GOAL and many other. These frameworks designed for rapid prototyping of multi-agent systems and can serve to model social complexity, physical modeling, abstract modeling or machine learning.

Special purpose multi-agent frameworks include various tools for designing distributed systems and environments. Example of such tools could be Orleans[26], framework for design and development of highly-distributed large-scale applications for cloud computing.

Referring to a paper by Florin Leon et al., we can say that amount of tools related to the design of distributed environments is very big. One of the examples described in this paper is JADE - Java Agent Development Framework. This framework is used to design distributed applications composed of autonomous entities. It is implemented in Java and has a purpose to simplify the implementation of multi-agent systems using a middle-ware which complies with the FIPA specifications and use a set of graphical tools to support the debugging and deployment phases[27].

This is the open-source tool allows communication between agents with FIPA standards[28]. The framework has many extensions which provide an opportunities to define the behavior of the agents through established BDI (Believe-Desire-Intention) model. This model represents the architecture of intelligent agent modeling based on concepts of practical reasoning. More information about this type of architecture can be found in [3].

Many frameworks use the same model as JADE, for example, Madkit - library for designing and simulating of multi-agent systems. This library provides opportunities to design multi-agent solutions using AGR (Agent/Group/Role) organizational model, which implies agents to play specific roles in groups and through this create artificial societies. Main features of these frameworks are connected to faster and more simpler ways of deploying solutions for distributed environments and development applications. Most of them have predefined sets of rules for the agents. Interaction process between agents is mostly based on real-time request-response models.

Frameworks which more focused on modeling of agent-based environments include such tools as NetLogo, Repast, and Mesa.

Repast represents one of the platforms for modeling and simulation specifically designed for social science applications. It can be used to develop agent-based models using computing clusters and has two versions for general modeling and large complex computing.

One of the most famous frameworks for multi-agent modeling is NetLogo. It is particularly well suited for educational purposes and has a large number of predefined models for different scenarios. Users can give instructions to the agents which operate independently. This framework can be used in many areas including biology and social science but is less suitable for modeling of intelligent agent behavior as the main purpose of this framework is simulations of natural and social phenomena.

Another framework for agent-based modeling is Mesa - an Apache2 licensed general purpose framework in Python[29]. It allows users to quickly create agent-based models using built-in core components (such as spatial grids and agent schedulers) or customized implementations and analyze their results using Python's data analysis tools. Its goal is to be the Python 3-based alternative to NetLogo, Repast, or MASON.

2.4 Ways of implementation

In this thesis, we investigate the advantages and problem-solving capabilities which can be provided by MAS in relation to Smart City environment. In order to do it, we created several models for the set of problems connected to some of the areas of Smart City environment. Through these models, we can demonstrate how agent-based systems can manage different tasks and provide flexible universal solutions.

For the implementation of the models we considered two major ways:

1. Usage of frameworks with the real-time software agents such as JADE and osBrain[30]
2. Usage of frameworks for the agent-based modeling such as Mesa

Frameworks for modeling real-time environments provides opportunities to simulate interactions and communication between software agents with usage of specific communication protocols. It allows us to simulate data flows in the multi-agent environment and is very useful in case of design solutions for distributed applications. JADE framework can be used for implementation of fully distributed systems with agents possibly running on different machines with the support of FIPA ACL for communication between agents. In case of osBrain, ZeroMQ as the transport layer for the asynchronous communication between agents is used. Using Pyro4 library, osBrain provides an opportunity to treat every remote agent as a local object which can be reconfigured. Agents are considered as independent system processes and can be run on different machines.

Although these tools provide enough functionality to design distributed applications and smart networks, they have a set of restrictions. For instance, in order to implement agent networks in JADE or osBrain, communication structures between agents should be established and models for agent behavior should be designed separately. In many cases, implementation of additional agents to support communication between different layers of the network could be needed. Particular aspects of behavior templates, which are used for implementation of software agents, can make it harder to implement custom behavior models for different cases. Usage of real-time communications also makes the design of solution and simulation process more complex to develop.

With regards to mentioned earlier and experience of testing the solutions we can assume that usage of frameworks focused on creating real-time distributed multi-agent systems can be an efficient way to design and deploy distributed applications or implement a particular solution for a complex multi-threaded, multi-machine systems, but it is less suitable for modeling or simulation of the more general set of problems, similar to our case.

On the other hand, one of the biggest advantages of frameworks designed for agent-based modeling is an opportunity to create simple prototypes of desired multi-agent systems as models and test these solutions using designed discrete environment. It helps to save time as some aspects of distributed system development such as handling different protocols, request-response time-outs and losing data can be omitted. Although, logical templates and behavior models often need to be implemented from scratch.

In our work, we decided to use Mesa framework for agent-based modeling. This framework allows us to simulate interactions between autonomous agents to see and evaluate their effects on the system. It provides a fast and efficient way to create prototypes and, as the templates for the agents are basic but have very flexible structure, this makes implementation of custom behavior models for the agents easier. Mesa also allows us to make simulations which involve large numbers of agents and provides opportunities to freely experiment with parameters of the environment for different simulations.

The basic structure of models in Mesa represented by several core components such as *Agent* class, which serves as a template for implementation of individual agents, *Model* class, which is used to define our model and *Schedule* class, which defines the scheduler component of the model. This component controls the order in which agents are activated.

In Mesa, it is possible to use different built-in scheduler classes with the common interface. This allows us to change the activation regime of a given model and implement custom activation schedules. For our models, we customized the default *BaseScheduler* class in order to make different groups of agents be activated in specific order based on schedule from one of the examples for the Mesa project[31].

Every agent in Mesa has a method called *step*, which defines actions of the agent at the moment of activation. After adding all the agents to the schedule, the scheduler's method *step* can be called to activate agents according to the established order. Schedule mechanism allows us to simulate the interaction between different groups of agents. In our models, we use discrete time steps with agents activation according to schedule at each time step.

Each agent in the system has a unique identifier. Using these identifiers we can differentiate agents and update conditions for the particular agents.

2.5 Roth-Erev learning algorithm

Reinforcement Learning (RL) represents a concept which is closely tied to the concept of the agent as entity characterized by such aspects as the perception of the environment, reasoning, and action. In order to implement learning aspect for the agents in our work, we decided to use a variant of Roth-Erev algorithm. Roth-Erev is a stateless algorithm which uses propensity values and probabilities which derived from these propensities. This type of algorithm is commonly used in market-based multi-agent systems and flexible enough to be used for different cases.

The learning process for this algorithm based on probability values of choosing a particular strategy from the set strategies. For example, we can consider n learning agents with j strategies[32]. In time step t propensity value $q_{nk}(t)$ of learning agent n corresponds to his strategy k and represents its willingness to play this particular strategy. At the start of the simulation, we have set of initial propensities $q_{nk}(0)$, where all propensities are equal. After particular strategy k is played, the payoff is received and corresponded propensity value is updated.

We derive probabilities of choosing a particular strategy from the propensity values. During learning process we can observe the law of effect in action as more successful strategies, which lead to a higher payoff, would be played more often due to higher corresponded propensity values and, as a result, the higher probability to be chosen.

The general Roth-Erev algorithm, described in a paper by Mridul Pentapalli[33], is shown in the equation 2.1.

$$q_j(t+1) = [1-r]q_j(t) + E_j(e, N, k, t) \tag{2.1}$$

$$E_j(e, N, k, t) = \begin{cases} \pi_k(t)[1-e], & \text{if } j = k \\ \pi_k(t)\frac{e}{N-1}, & \text{if} j \neq k \end{cases}$$

where $q_j(0)$ is the initial propensity of action j at time $t = 0$, $q_j(t)$ is the propensity of action j at time t, $\pi_k(t)$ is the reward obtained for taking action k at time t, $r \in [0, 1]$ is recency parameter, $e \in [0, 1]$ is the experimentation parameter and N is the number of available actions. Probabilities are derived from the propensity values using the formula, described by the equation 2.2.

$$p_j(t) = \frac{q_j(t)}{\sum_{i=0}^{N-1} q_j(t)} \tag{2.2}$$

where $p_j(t)$ is the choice probability of action a_j at time t

Chapter 3

Method

As we mentioned in our work, in order to get a deeper understanding of possibilities, provided by MAS regarding Smart City context, it is important to find an example which can demonstrate how different components can interact with each other to fulfill certain goals in the Smart City environment. For this purpose, we consider several problems which can be found in Smart City environment, such as smart house, smart parking and smart traffic with toll stations and barriers. Due to complex nature of the Smart City and variety of aspects connected to intersections of the Smart City domains we scale our problems to make them more suitable for modeling.

Although these problems are connected to different domains, they can be represented in a similar way. For example, the problem of energy distribution between different electrical devices in the smart house can be considered similar to the distribution of free parking places between cars which want to park, as in both cases we are dealing with the distribution of some restricted amount of common good which should be divided between a number of consumers.

We implemented several approaches for every model in order to demonstrate collective problem-solving capabilities of multi-agent systems. These approaches include market-based models with zero-intelligent and learning agents and centralized approach.

In the Smart City environment, it is possible to consider different models of agent interactions. One of such models is the interaction between several self-interested agents. In contradiction to cooperative models with groups of agents having a common goal to achieve, in models with self-interested agents every agent has its own goal and acts in accordance to certain rules in order to achieve it. This leads to one of the important aspects of self-interested agent behavior model as the agent can have an individual description of desirable states of the environment and act in an attempt to bring these states to reality[34]. This can also be beneficial for other agents and overall system.

For example, the heater in the smart house can have a goal to maintain certain temperature range, or smart lamp agent can regulate brightness to maintain a certain level of light on the street depending on particular conditions. These agents have their local goals and restricted amount of information about the environment in order to fulfill them.

3.1 Approaches

In this work, we consider two main types of approaches used to model interactions between agents. For the decentralized approach, we model interactions within the system with zero-intelligent agents as well as interactions within the system which includes learning agents. Problem-solving capabilities in the decentralized approach emerge from the interactions between different agents. For the centralized approach, we consider model structure where the process of finding the solution to the problem is mainly controlled by the single agent.

3.1.1 Decentralized approach

For the decentralized approach, we implement a market model based on peer-to-peer (P2P) distributed approach, described in [19]. This approach allows buyers and sellers trade with each other individually on the pay-as-bid basis. Trading process is divided into time slots, in every time slot, each buyer, represented by the agent with a demand iteratively paired with randomly chosen seller until it satisfies its demand or it has been paired with all available sellers.

When the two agents are connected, certain condition for the performance of a transaction must be fulfilled. For example, if the bid price of the buyer is greater than or equal to the ask price of the seller, buyer pays its bid price and allocates the desirable amount of good.

The amount of traded good can be restricted ,for instance, determined as the minimum amount of seller's bid and buyer's ask amounts. Amount of demand, which cannot be acquired on the local market, has to be satisfied from the external market if possible. In case of a smart house, if some of the loads are impossible to satisfy with available renewable energy, they need to be satisfied from the outer sources such as smart grid. In this case, the price of acquiring the energy could be depended on current peak hours for energy demands on the grid.

In this market design, the order of paired buyers and sellers is important as the amount of good which can be traded is limited. In order to avoid competitive advantages every buyer and seller which would be paired, are chosen randomly. Each P2P transaction results in an individual price for the deal. The basic scheme of the decentralized approach can be seen in the figure 3.1.

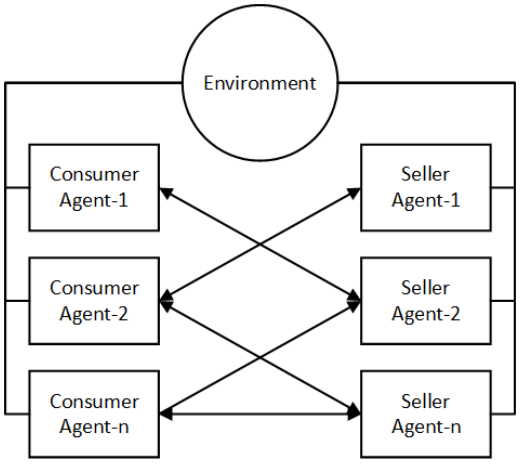


FIGURE 3.1: Decentralized market model.

Every consumer agent is randomly paired with the seller agent. At the start of each transaction, the conditions for the deal are checked. If the conditions are satisfied, the consumer can acquire the amount of good it needs from the seller. If the conditions are not met, seller and buyer are disconnected and their ask and bid prices are updated to increase the possibility of the next deal to be successful.

If consumer manages to acquire necessary amount of good it is excluded from the current trading process. The trading process continues until all the consumers are able to satisfy their demands or all available common good is distributed.

We use described market model for both zero-intelligent and learning decentralized approaches.

Decentralized zero-intelligent approach

At the start of each trading in the ZI decentralized approach agents choose a price which is randomly assigned within the boundaries of upper and lower price limits. In our approach, we use the average price for the deal to update ask price of the seller during trading. In case of rejected deals, it helps to make it more possible for the next deal to be successful. It also helps to increase the speed of convergence for the simulations. We do not consider it as a learning due to the fact that at the start of each trading new prices are chosen and the overall behavior pattern of seller agents is not affected. Although it makes sellers not pure ZI agents, they still correspond to essential aspects of ZI trading such as randomness of initial choices made, the absence of experience gathered through the multiple trading episodes and lack of any strategies.

For the ZI approach bid price is chosen randomly but we can assume that buyers could have a desire to pay more as the higher bid price implies higher probability for a successful deal which is essential as the amount of traded good is limited.

Decentralized approach with learning agents

As the ZI agents are not able to memorize the results of their past actions and do not have

any strategies to use, they always act in accordance with defined constraints. In contrast to ZI agents, intelligent agents can learn from their previous decisions and trading results during the time. This allows them to be more flexible and choose a strategy which could lead to better results, for example, reduce energy consumption in case of lack of energy or use strategies which could increase the probability of a successful deal.

In order to make agent learn from its past decisions, RL algorithms are used. For our model we decided to use modified Roth-Erev algorithm, described in [19]. Scheme of implemented learning intelligent agent can be seen in the figure 3.2.

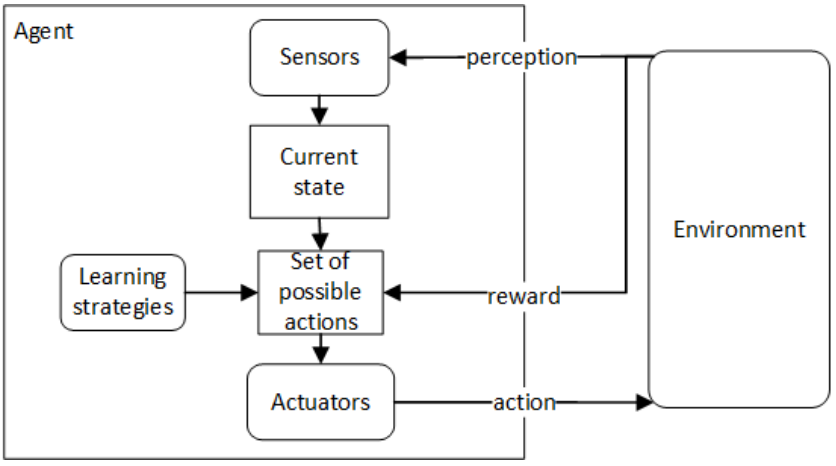


FIGURE 3.2: Learning agent with states.

Learning agent can have set of possible states and sets of strategies for every state. Using information from the environment, the agent can define the state it is in and choose a particular strategy with the probability derived from the propensity value, which corresponds to the particular strategy. Using reward we can update propensity value and increase the probability of choosing the particular strategy. As the Roth-Erev algorithm is initially stateless, it can give us the opportunity to modify structures of our learning agents and apply different approaches, which include learning agents with sets of states and strategies, as well as stateless agents with only sets of strategies.

The algorithm which we use in our work described by the equation 3.1.

$$q_{i,j}(t+1) = (1-r) \cdot q_{i,j}(t) + E_{i,j}(e, N, k, t) \tag{3.1}$$

$$E_{i,j}(e, N, k, t) = \begin{cases} R_k(t) \cdot (1-e), & \text{if } j = k \\ q_{i,j}(t) \cdot \frac{e}{S_i - 1}, & \text{if} j \neq k \end{cases}$$

where $q_{i,j}(0)$ is the initial propensity of action j for agent i at time $t = 0$ (aspirational level), $q_{i,j}(t)$ is the propensity of action j for agent i at time t, $R_k(t)$ is the reward obtained for taking action k at time t, $r \in [0, 1]$ is recency parameter, $e \in [0, 1]$ is the experimentation parameter,

S_i is number of possible strategies for the agent i. Recency parameter r represents memory aspect of agent and determines how fast agent can forget about its past decisions. It maps the weight of propensity value at time step t into the updated propensity value for time step $t+1$. Larger r implies faster speed of forgetting. It helps to prevent an agent of falling into specific behavior pattern too fast. Experimentation parameter e determines influence of last action on the future decisions. Therefore, it can be considered as speed of learning for an agent.

Each learning agent i updates its propensity value $q_{i,j}$ to choose particular strategy j in time step $t+1$. Probabilities of choosing the strategies are derived from the propensity values using the equation 3.2.

$$p_{i,j,t} = \frac{q_{i,j}(t+1)}{\sum_{i=0}^{S-1} q_{i,j}(t+1)} \tag{3.2}$$

where $p_{i,j}(t+1)$ is the choice probability with which agent i will chooses strategy j. It is derived from the propensity values at each time step t.

With this algorithm, we can update probabilities for all available strategies based on the amount of reward and propensity values. As the law of effect implies, a strategy which results in a higher reward will have a higher probability to be chosen for the next time step.

The amount of reward is one of the important aspects of the learning. In case of low initial propensities, high amount of reward can negatively affect the learning process for the agent by increasing the probability of choosing particular strategy too much comparing to other possible strategies. Using reward of zero as a penalty for the agent, we can reduce the probability for the particular strategy to be chosen and increase the chance for an agent to choose a better strategy.

By tuning parameters of learning algorithm such as the size of initial propensities, amount of reward and sizes of experimentation and recency parameters, we can regulate speed and results of learning for our agents and save the possibility for them to learn different behavior patterns in case of changes within the environment.

3.1.2 Centralized approach

Contrary to the decentralized approach, in which distribution of common good is based on interactions between different agents in the environment and their decisions, in the centralized approach, we have one main controlling agent responsible for distribution of common good between consumers. This agent has the necessary information about the environment, such as amounts of demand and available supply, and can act in accordance with certain rules to maintain system stability.

The basic scheme of the centralized approach can be seen in the figure 3.3.

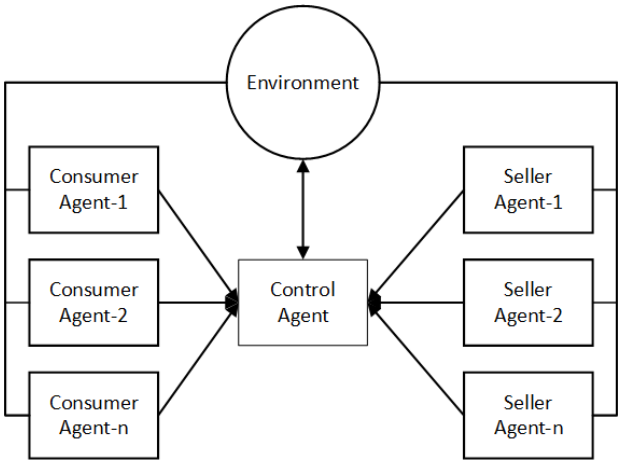


FIGURE 3.3: Centralized model.

For this work, we use an approach based on genetic algorithm to encode consumer loads, which can be energy demands of particular electrical devices, demands of smart cars for parking or for going through the road fast. The idea of this approach is based on load scheduling, described in [24] and encoding of access to the energy lines, described in [22].

Algorithm for the distribution of common good is described by the following steps:

Algorithm 1 Distribution algorithm for the centralized approach

```
Data: (f(x), NI, pop, n, d, NIT, p, rand, d_mutated)
get information about demand and supply
for i = 0 : NI do                                   ▷ NI - number of iterations
    generate population pop of random binary vectors of size n
    calculate fitness of every vector x based on constraint function f(x)
end for
find the fittest vector d ∈ pop
for j = 0 : NIT do                                  ▷ NIT - number of iterations for partner set
    choose number of vectors which can be used for crossover
end for
select the fittest vector p as a partner for crossover
for i = 0 : NI do
    choose condition coefficient rand, (0 ≤ rand ≤ 1)
    if rand > 0.8 then
        make a mutation of vector d
        calculate fitness
        if mutated version d_mutated of d has better fitness then
            d = d_mutated
        end if
    else
        make a crossover of vector d and vector p
        calculate fitness
        if the any result of crossover has better fitness then
            d = result of crossover
        end if
    end if
end for
decode vector d and distribute common good accordingly
```

In addition to described parameters in the algorithm, we have $f(x)$ as fitness function, which is used to determine the fitness of solution x and n as a number of consumers which determines the size of DNA vectors. The purpose of this algorithm is to find a binary DNA vector representing the number of loads which can be satisfied in regards to available supply, and distribute supply accordingly. Distribution is done by decoding chosen DNA vector, in which we satisfy particular demand if the corresponding gene in the vector is equal to 1. With the diversity of solutions genetic algorithm can provide and fitness constraints it is possible to find the solution which

allows us to maximize the number of satisfied demands and usage of available common good, such as renewable energy.

For the mutation, we go through the DNA vector, every gene of which can be switched out with the opposite element. The probability of mutation is determined by the *mutation_chance* variable, which is in our case is equal to 100 to give every gene 1/100 chance to be changed. This helps to increase the diversity of solutions.

The crossover function takes two DNA vectors, slices them into two parts at random index within their length and merges them. For both vectors, it keeps their initial sublist up to the crossover index and swaps their ends. Code used for implementation of crossover and mutation functions is based on [35].

3.2 Models

3.2.1 Smart house model

For this model, we consider interactions between several entities in the smart house environment, modeled using a bottom-up approach.

In our model we have two agents which responsible for controlling electrical heaters, agent controlling light system, agent, controlling heated floor, agent controlling energy battery storage, as well as agents responsible for PV panel and wind energy. Together these agents represent energy loads and energy generation of the smart house.

The goal, in this case, is to minimize energy costs in the house and maximize usage of renewable energy, produced by several facilities such as solar PV panel and wind turbines.

Example of diagram of implemented agent classes presented in the figure 3.4

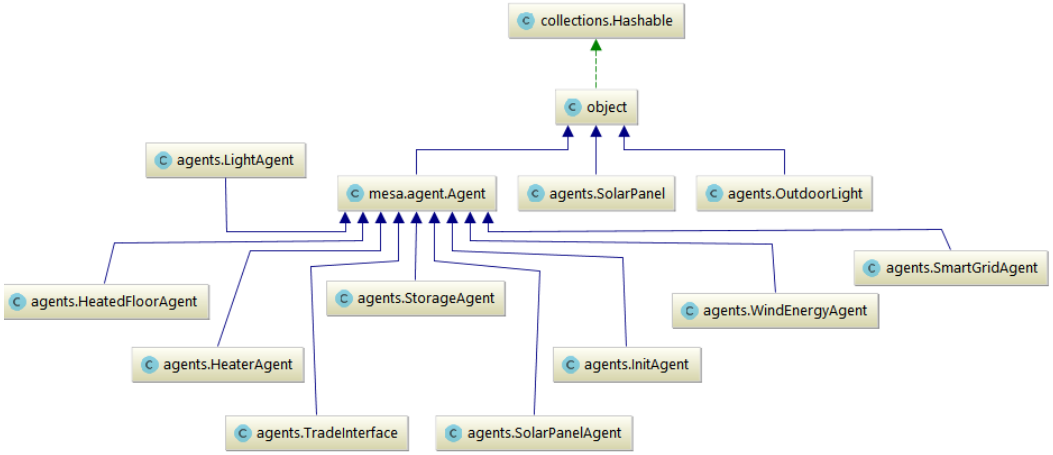


FIGURE 3.4: Smart house model.

The demand side of the model includes two heater agents, represented by $HeaterAgent$ class, agent, responsible for the heated floor, represented by $HeatedFloorAgent$ class, and the agent responsible for the light system, represented by $LightAgent$ class.

The supply side of the model includes agent responsible for PV panel, represented by $SolarPanelAgent$ class, the agent responsible for energy acquired from the wind turbine, represented by the $WindEnergyAgent$ class and agent, responsible for energy battery, represented by the $StorageAgent$ class.

Energy battery agent is a prosumer, as it could serve as supply-side agent and be the main energy source in case of lack of renewable energy, as well as have a demand depending on current battery energy level.

In order to initialize the state of the environment at the start of every time step, we use $InitAgent$ class.

$SmartGridAgent$ class represents agent responsible for the interaction with the smart grid and possible buying energy from the grid or selling the energy surplus back to the grid for the price, which depends on peak hours. This concept provides opportunities for creating more complex interactions for the smart house model to make it reduce overall peak energy loads by selling renewable energy to the grid, but due to time constraints and the fact that it was not the main aim of this work, it was not completed.

In order to provide trading between agents according to modeled P2P market approach for the decentralized models, we use $TradeInterface$ class. This class provides connections of supply and demand side agents in random order with the goal to distribute available renewable energy until the requirements for the end of trading are fulfilled. For the centralized approach, the $ControlAgent$ class is used, which represents controlling agent, responsible for energy distribution.

We consider agents responsible for the heaters to be placed in different rooms and have a goal to maintain temperature for the room in the interval, defined by minimum temperature and desired temperature value which is considered as a simulated input from the person in the smart house. In reality, it can be input from the mobile device sent through the ZigBee or WiFi networks[36]. They calculate their demands based on the difference between the current indoor temperature in the room and desired temperature.

We use the formula, described by the equation 3.3 to calculate the amount of energy needed to heat the room.

$$U = c_p m \Delta T \tag{3.3}$$

where c_p is dry air specific heat value, p is dry air density, A is room area, h is room height, m is mass of air and ΔT is temperature difference.

This equation is based on specific heat formula: heat energy = (mass of substance)(specific heat value)(change in temperature)[37].

We use Monte-Carlo methods and normal distribution to simulate output from the environment, such as to define the level of solar energy or current indoor temperature. We also simulate output from the set of sensors inside the house to check movement, temperature or light levels for the particular time step. This is used to present additional dynamics to the simulated environment and make it closer to the real world as in smart house we can operate with a variety of information about current condition inside the house and user preferences in order to reduce energy consumption and maintain comfort level.

The probability density function of the distribution described by the equation 3.4.

$$f(x) = \frac{exp(-x^2/2)}{\sqrt{2\pi}} \tag{3.4}$$

Example of probability density function plotted on the set of random samples can be seen in the figure 3.5.

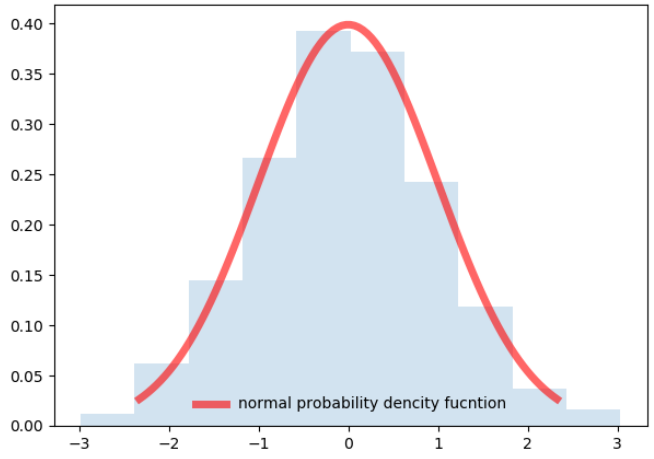


FIGURE 3.5: Normal probability density function.

In each time step, heater agents check if somebody is in the room, get information about current indoor temperature and desired temperature and calculate their demands. If nobody is in the room at the current time step, the desired temperature is set to default minimum value. If current temperature in the room is equal or more than desired temperature, heater turns off in order to save energy. Probability of person being in the room or be away is based on current hour and day to follow average routine of people.

In order to generate indoor temperature input for the heaters, different possible ways were considered including thermodynamical model, described in [10]. After studying and testing the approaches it was decided to use normal distribution and data on the average indoor temperature, described in [38]. This allows us to reduce number of parameters for the model and omit complex thermodynamical calculations and provides us with sufficient diversity of temperature data. For this model we do not consider such aspects as temperature loss in time as we are mostly interested in studying of agents behavior based on state of the environment. In reality ad-hoc situations such as open window or door, can happen and influence temperature levels. In order to properly calculate influence of outdoor temperature, many parameters such as thickness of the walls, amount of furniture in the room or even position of the house should be considered, which is not the main focus of this study. Considering these aspects it was decided to use generated values for the indoor temperature in order to simulate changes in the environment.

The light agent represents an example of a light system in the room. It calculates energy demand based on current outdoor light level and desired light level both measured in lux. We calculate the energy load for light using the equation 3.5, which is based on the formula for converting lux to watts from [39].

$$U = \frac{l_{desired} * A}{lm_w} \tag{3.5}$$

where $l_{desired}$ is amount of illumination needed to reach desired light level, measured in lux, A is room area, lm_w is parameter of luminous efficiency per watt.

For the purpose of reducing the complexity of calculations, we do not consider such aspects as light loss and coefficient of utilization. To simulate input from the user for the light agent we set several profiles for the desired light intensity, based on recommended light levels described in [40]. By calculating the difference in lux between desired light level set according to chosen profile and current outdoor light, the light agent can define its load for the current time step.

At the start of every time step, the light agent checks the movement in the room similarly to heater agents, get information about current outdoor light and desired light level and define its load accordingly. If the current outdoor light level is higher than the desired light level or if nobody is in the room for the time of defining the demand, lights turn off.

The demand of the heated floor agent calculated based on running costs for the heated floor in typical bathroom area[41]. It acts as a simple reactive agent and defines its demand solely based on the simulated output from the movement sensors which is made similarly to the agents mentioned earlier.

Agent, responsible for energy battery, monitors the state of the battery and based on battery discharge depth it calculates necessary amount of energy. Guided by the restrictions for the battery discharge described in the following resource[42] we have the set of conditions based on a percentage of discharge, described in table 3.1.

states	percentage
maximum	$100\% \geq x \geq 70\%$
stable	$70\% > x \geq 50\%$
unstable	$x < 50\%$

TABLE 3.1: Battery conditions

where x is the amount of energy stored in the battery. If battery condition is unstable, it needs to be charged in order to prevent degradation of the battery. In this case, storage agent plays the role of the consumer and participate in trading to obtain the amount of energy needed to fully charge the battery or to return to the stable condition. Otherwise, storage agent is considered as a supply-side agent. We also consider that the energy surplus after every trading can be stored in the energy storage if maximum capacity of storage allows it.

Agent, responsible for PV panel represents a source of solar energy for the smart house. Amount of solar energy for every time step is defined using normal distribution function, data on solar energy output from the dataset, provided by my advisors and the weather coefficients. Using normal distribution we create new values based on average hourly data from the dataset for 24 hours. As we do not know precise information about weather conditions, size, number or angle of inclination of the solar panels in the place from which information is obtained, we use weather coefficients and time to make values of solar energy output be dependent on the state of our environment.

Wind energy agent represents a source of wind energy for the smart house. For a generation of wind energy output, we use hourly data from the wind farm for the period from 1/2/2018 to 23/03/2018, obtained from the following source[43]. As initial data consists of energy output from the wind farm with at least several big wind turbines, it has to be scaled to fit our modeled environment.

For this model we have the following order of agents: before the energy distribution, we initialize our environment to set the level of solar energy and outdoor light based on weather state and the current time. Then we calculate the amount of supply which can be distributed. After every supply-side agent defines the available amount of energy it can provide at the current time step, demand-side agents define their demands based on the state of the environment they are in. When all loads and supply are established, distribution of available renewable energy is conducted.

Decentralized zero-intelligent approach

For the decentralized smart house model we use P2P market-based approach described earlier, in case of ZI agents we represent smart house environment as a local market with renewable energy and make agents interact accordingly to the market model and distribute available energy during every trading between agents responsible for sources of renewable energy and agents which need

energy for the trading period. Prices for trading are based on energy prices for kWh from the Nordpool[44]. We assume that prices for the local market are less than prices for buying energy from the outer sources such as smart grid as using internal renewable energy is more beneficial for the house system comparing to buying it from the grid.

Decentralized approach with learning agents

In order to maintain stability of the system and reduce costs, connected to buying additional energy for satisfying energy demands, it is important to consider flexibility of the demands. As most of the demand-side agents represent types of services which have to be maintained every time step, the flexibility of the demand can be expressed in reducing consumption. For this approach, we consider heater agents as learning agents with the sets of strategies based on temperature range they need to maintain. We implement set of states for the heater agents based difference between current indoor temperature and desired temperature set by the user. For every state, the agent has set of strategies it can use. For this model, we adapt our learning method to use it with states. In the table 3.2 sets of states and actions implemented for the heater agents, can be seen.

states	actions
Critical	minimum
	intermediate
	ideal
Low	intermediate
	ideal
Intermediate	stay
	ideal
Ideal	pass

TABLE 3.2: Table of states and actions for heater agents

The *Critical* state corresponds to the situation when the current indoor temperature is less than default minimum temperature value and desired temperature is higher than minimum temperature. For this state, the agent has three actions it can take. Action *minimum* corresponds to heating the room up to the minimum temperature value. *Intermediate* action implies heating up to the temperature close to ideal desired temperature, in our case, it is defined as temperature one degree lower than the ideal desired temperature. The last action for this state implies heating up to the maximum desired temperature set as an ideal temperature to maintain.

Low state corresponds to the situation when the difference between current indoor temperature and desired temperature is bigger than one degree. In our model, this state implies lesser temperature difference between indoor and desired temperature values than for the *Critical* state. As we assume that heater agent does not have the desire to increase the difference between current and desired temperature we reduce the number of strategies for the *Low* state and make agent choose between two options: heat up to *intermediate* or to *ideal* state. As

the *Intermediate* state is closest to the ideal temperature, we make it possible for the agent to stay in this state or heat up the room to *ideal* temperature. If the indoor temperature in the room is equal to the desired temperature, which corresponds to *Ideal* state, we assume that no actions from the heater agent required and heater turns off.

This state-action structure allows heater agents to change their energy demands based on action choices and act according to minimum policies in case of lack of renewable energy as well as use maximum amount of energy if possible. In order to make agent learn that action to choose for every state the reward system is set. Amount of reward depends on the choices of agents and distributed based on results of trading. We assume the following possible situations for the trading results:

- "energy left": if all the demands are satisfied and some energy surplus exists
- "not enough energy": if amount of renewable energy is not enough to satisfy all the demands
- "no buyers and sellers left": situation when amount of renewable energy is equal to the amount of the demands

Similar sets of situations are considered in all our models as it reflects possible results of common good distribution which were modeled. The reward for the agent is evaluated based on agent action and situation after trading. We assume that existing of energy surplus after trading yields possible increasing of demands to maximize usage of renewable energy and choices which leads to increasing the demand such as heating up room to ideal temperature instead of minimum, results in a bigger reward. On the other hand, acting within the minimum policies such as heating up to a minimum in case of energy surplus, is penalized using reward, equal to zero. For the case of lack of renewable energy reward evaluation is opposite and reducing the demand in order to save energy is rewarded.

In case if all renewable energy was distributed successfully between the agents and no surplus or lack of energy exists, both learning agents get a maximum reward as we assume this scenario as most desirable for us. Based on the amount of occurrence of every situation, agents can learn which strategy to choose.

Amount of reward for agents is scaled based on maximum temperature values from the temperature range as for learning it is important to keep the size of reward within the certain limits depending in order to stabilize learning rate and reduce the possibilities for an agent to fall into specific behavior patterns too fast.

Centralized approach

For the centralized approach in smart house model, we consider energy demands of agents

as loads which can be encoded in a binary vector. In every time step, the controlling agent receives information about current energy loads and available renewable energy. Based on the number of customers and their demands, the set of loads for this time step is created. Using the distribution algorithm 1, described earlier, controlling agent distributes available energy according to a founded solution with the best fitness. In order to calculate fitness for this model, we implement fitness function which, with slight abuse of notation, can be described by the equation 3.6.

$$f(x) = \begin{cases} N - (p - d_x), & \text{if } d_x \leq p \\ 0, & \text{otherwise} \end{cases} \tag{3.6}$$

where N is positive numerical constant defined as $N \geq (p + d)$ in order to ensure that $f(x)$ will be always positive, p is total amount of available renewable energy, d_x is sum of demand for particular binary vector x, d is total amount of energy demand. The lesser is the difference between available energy and amount of demand represented by vector x, the closer is this solution to Pareto optimality as it implies that the energy cannot be distributed differently without worsening the result for at least one agent[45]. The ideal situation implies zero difference between supply and satisfied demand which means that all the energy can be successfully distributed. In terms of Pareto optimality, this solution will be dominant and will have the best fitness. Demands which cannot be satisfied with the renewable energy assumed to be satisfied from the outer source such as smart grid.

3.2.2 Smart parking and cars model

For the smart parking problem, we implement a model which consists of car agents as smart cars with a desire to park for the particular amount of time and parking slot agents as available parking slots with the goal to sell their acceptability to cars. A limited number of available parking slots in every time step represents common good which is distributed between cars during every time step.

In order to generalize our model and highlight the aspect of multi-agent interactions, we decide to model every parking slot as an agent. In theory, every parking slot can have set of sensors and act as a part of a smart network created by the interactions with drivers or smart cars using different communication methods as was described in [16]. In the figure 3.6 example of diagram of implemented classes can be seen.

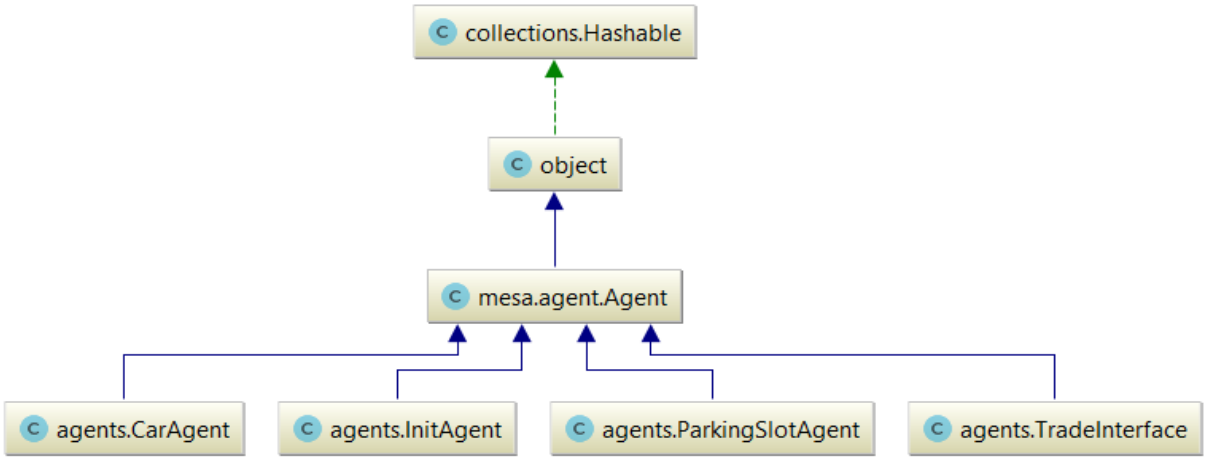


FIGURE 3.6: Smart parking model.

Car agent, represented by the *CarAgent* class has a desire to park for the period of time, chosen within the defined boundaries. The probability for the car agents to have a demand for parking increasing during peak hours for traffic congestion[46],[47] as the bigger amount of cars on the roads is considered during these hours. In order to park, agent forms a bid using price defined within the boundaries.

Parking slot agent, represented by the *ParkingSlotAgent* class sells its availability to the car agents. The availability depends on the amount of time parking slot is occupied by the car. Each parking slot has two possible states:

$$state = \begin{cases} busy, & \text{if occupied time } > 0 \\ free, & \text{otherwise} \end{cases}$$

In each time step, available parking slots are distributed between cars which want to park. Then the car is paired with the parking slots they both excluded from distribution for the amount of time car desired to park.

The *TradeInterface* class in this model has the same role as for the smart house model and also replaced by the *ControlAgent* class in the centralized approach.

The *InitAgent* class is used for the ZI decentralized approach to set initial price limits for the parking slots.

Decentralized zero-intelligent approach

In the decentralized ZI approach for parking we slightly modify the initial market model and simulate interactions between car agents as buyers with individual bids and parking slots as sellers with an initial price limit, defined before the start of every trading period. When the car agent is paired with the parking slot, it suggests its bid and if the deal is successful, they both excluded from the trading for the amount of time bought by the car agent.

After each deal price of available parking slots, waited to be paired, is updated based on average price of the deal. As price limit is updated after every deal regardless of the outcome, we still can have each transaction be resulted in individual price. On the other hand, using this principle we can dynamically update price for the parking based on demands of the consumers and results of the deals.

Decentralized approach with learning agents

For the decentralized approach with learning agents in this model, we apply principles of the distributed market with the intelligent agents, demonstrated in [19]. In contradiction to ZI approach, we consider both car agents and parking slots as learning agents with sets of strategies. Strategies for agents represented by price choices which are discretized to integer values between determined price boundaries. This implies that all strategies of agents are contained in the set $s \in S = \{p_{min}, .., p_{max}\}$, where p_{min} and p_{max} are minimum and maximum price boundaries. For every trading period car agents and parking slot agents make a price derisions by choosing a particular strategy from the set. When trading is conducted, agents which participate in trading receive rewards according to results of the deals.

If the deal between two agents is successful, both of them are rewarded. A reward of car agent is based on the difference between chosen bid price and outer price, which represents price for other parking places. This implies bigger reward for the lower bid price. For the parking slot agent amount of reward is based on ask price as we assume that higher ask price in the case of the successful deal is more profitable for the seller. In order to reduce differences between rewards for buyers and sellers and avoid falling into specific bidding pattern too fast, all rewards are normalized using the maximum value from the set of prices. If the deal is rejected, both agents are penalized and have to choose another price strategy. This allows agents to learn that strategy is more likely to lead to the successful deals. As a result of their interactions, both selling and buying side agents can learn to use particular price strategy.

Centralized approach

In the centralized approach, we encode every car agent with the desire to park as a load and create a vector of loads based on a number of cars. The available amount of parking slots serves as the boundary for fitness function similarly to the available amount of renewable energy for the smart house model. Calculation of fitness for every binary vector of loads is based on following conditions:

$$f(x) = \begin{cases} b_x, & \text{if } s_x \leq p \\ 0, & \text{otherwise} \end{cases} \tag{3.7}$$

where b_x is the sum of bids of car agents in vector x, whose demands could be satisfied, s_x is the number of agents in vector x which could park in regards to the available amount of parking slots and p is the total amount of available parking slots. Every car agent, encoded in binary vector x as 1, which implies that its demands are going to be satisfied, is added to the number

of agents suggested to distribution by this vector. If this amount is not bigger than a number of available parking slots, vector x can be considered as a possible solution. As every car agent has individual bid, the higher is the sum of bids from the combination of agents, suggested for the distribution, the higher is fitness value of the vector, representing this solution. Using these constraints we try to find the combination of car agents with the highest bids, whose demands can be satisfied in accordance with available parking slots at the current time step.

For the distribution in accordance with the chosen solution, car agents are randomly assigned to the available parking slots and excluded from trading similarly to decentralized approaches.

3.2.3 Smart traffic and toll stations model

For this problem, we consider a limited system which consists of car agents, toll agent for the main road and two barrier agents for sideways. Car agents considered as consumers with the desire to pass on the road. Road agents represent sellers side, they can sell the opportunity for the car to pass for a price. Barrier agents can close access to the sideways during rush hours, defined based on traffic congestion data[46],[47], but it can be opened if the level of car density on the main road is too high. This model represents an idea of interactions between road elements such as toll station and barriers, and cars with gathering and sharing information about congestion levels on the roads and prices, using different communication methods and sensors.

Common good in this case represented by the capacity of the roads as the only limited amount of cars can pass on the available roads in every time step. Cars which want to pass represent the loads and must be distributed between the available roads based on capacity and prices.

The figure 3.7 shows the example of diagram of classes in this model.

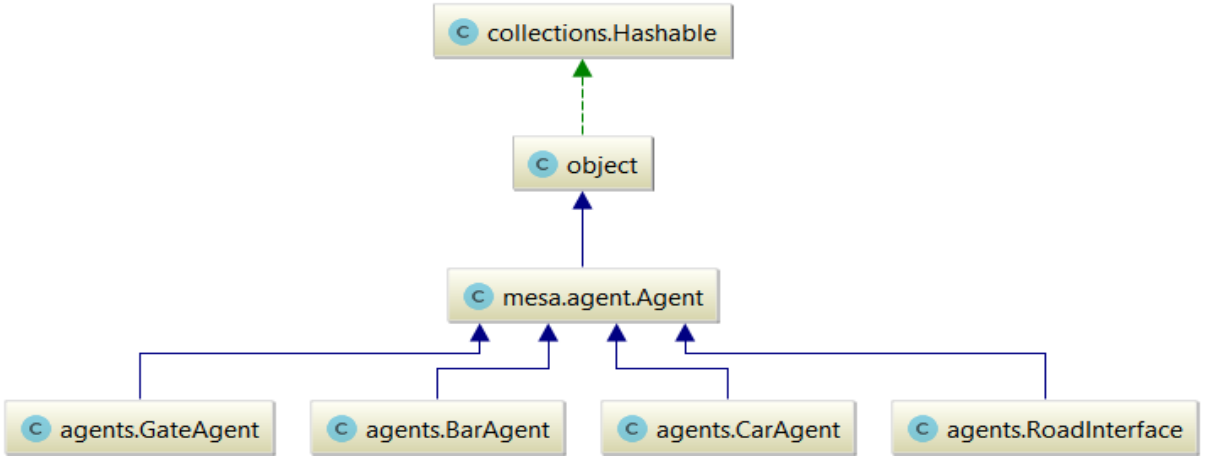


FIGURE 3.7: Smart traffic model.

CarAgent class corresponds to car agent as a demand-side agent similarly to smart parking model. For this model, we differentiate cars by three categories and set individual price limits for every category based on group rates and price data from the FJELLINJEN[48]. Every car agent can have one of the following types:

- *car*: corresponds to vehicles with an allowed total weight of 3500 kg and below, for example a passenger car.
- *lorry*: corresponds to vehicles with an allowed total weight more than 3500 kg, for example a truck or similar transport
- *emergency*: corresponds to emergency services such as an ambulance or fire trucks

Emergency services have the priority to pass. In case of the decentralized approaches, cars with this type can pass without paying the price in order to guarantee that the car will have a successful deal with the chosen road agent and pass immediately. *BarAgent* class represents barrier agents responsible for two sideways. *GateAgent* class represents gate agent responsible for the main road. In contradiction to the sideways, it is always open and serves as the main seller for the cars. The main road assumed to have more available traffic throughput per time step. Every road can have different parameters such as length and maximum speed limit. Speed parameter of the road determines the speed of driving through the road. When the road is free it is equal to the maximum speed limit. With increasing number of cars on the road speed parameter decreases. The formula, described by the equation 3.8, is used to calculate speed parameter of the roads[49].

$$U_i = \left(\left(1 - \frac{k_i}{k_{max}}\right)^m\right)^n \tag{3.8}$$

where U_i is speed parameter for the road i, k_i is the current car density level of the road i and k_{max} is the road maximum capacity value per time step, m and n are positive constants. The $RoadInterface$ class represents an interface for trading between agents for the decentralized approaches. It serves to provide interactions between sellers and buyers in a P2P manner similarly to $TradeInterface$ class, described earlier.

Decentralized zero-intelligent approach

For the decentralized ZI approach, we create a model based on similar aspects as ZI parking. Car agents, which have a desire to pass, choose their bid prices defined within the boundaries of maximum and minimum price limits. Price limits are set individually for every type of car agent. As the emergency services do not need to pay in order to pass, they do not need to choose bid price. Sellers, represented by the gate agent for the main road and barrier agents for the sideways, choose their ask prices, defined within the same boundaries. The process of trading is similar to previously described ZI approaches with a structure, described earlier in section 3.1.1.

When the car agent is paired with one of the available sellers, it can get the right to pass in case of successful deal or be rejected and wait to be paired again. Trading is conducted until all the cars are distributed between the available roads or the maximum capacity for the current time step is reduced to zero, which implies that no more cars can pass. The cars which have not manage to pass at the current time step form a queue of waiting cars and become first to be distributed at the next time step.

Decentralized approach with learning agents

In this approach we consider behavior model of car agents to be similar to reactive agents, which react to the input from the environment based on defined policies[50]. In contradiction to ZI approach, car agents can decide which road to choose based on current state of the road and price for a pass.

In order for car agent to make a choice, we introduce parameter of time for pass, calculated using equation 3.9.

$$T_i = \frac{l_i}{U_i} \tag{3.9}$$

where l_i is the length of the road i and U_i is the speed parameter of the road from the formula 3.8. With increasing number of cars on the road, the parameter of time for a pass for this road increases. In every time step car agents with the desire to pass choose sellers based on the sum of the time for pass and price for a pass.

Trading process is similar to that of the ZI approach, but instead of being randomly paired with the sellers, car agent searches for the available road with the lowest time and price for a pass. This reflects the desire of car agent to find faster and possibly cheapest way.

In this approach, we combine together several elements from the previous models such as a discretized set of states from the smart house learning agents and price strategies from the smart parking learning approach. Seller agents, represented by Gate agent and Barrier agents are considered as learning agents with sets of discretized states based on density levels. For every state, agents have set of price decisions discretized to integer values within the maximum and minimum price boundaries. Price choices for car agents determined within the same boundaries.

During the trading process, car agents are paired with available sellers and in case of a successful deal, car agent acquires the right to pass from the seller, updating its density level. In case of rejected deal car agent is assumed to wait for other offers and seller is penalized in order for it to change its price decision.

States for selling agents are assigned based on current density levels. Amount of reward received by the agent depends on price decision made by this agent and the state it is in. We assume that passing on the roads with low levels of car density implies higher ask prices to be rewarded more. On the other hand, in case of high density levels, the reward for lower ask prices is higher

in order to make car agents pay less for possibly higher travel time. The purpose of these aspects is to differentiate prices for the selling agents depending on the demands and density situation on the roads.

As a result of this approach, sellers try to learn specific price strategies based on results of the trading process and density levels. Car agents are distributed between the available roads based on changing density levels on the roads and price decisions of the sellers. Emergency services have the priority to pass similarly to ZI approach. Cars which do not manage to pass in the current time step, form a queue and have a priority for the distribution at the next time step.

Centralized approach

In the centralized approach, we encode car agents with the desire to pass as loads similarly to centralized smart parking and consider an available capacity for the current time step as a resource which can be distributed. Fitness function for this approach described by the equation 3.10.

$$f(x) = \begin{cases} b_x, & \text{if } s_x \leq c \\ 0, & \text{otherwise} \end{cases} \tag{3.10}$$

where b_x is the sum of car agents bids for a pass in vector x, s_x is the number of agents in vector x which could be distributed between the roads in accordance to total capacity value c for the current time step. The process of distribution is similar to centralized parking as in both cases we have car agents with the desire to get a supply be it the right to park or the right to pass on a road.

For these model we have emergency services and cars, waiting in a queue as prioritized groups which have to be distributed first in order to reduce congestion. After all emergency services and waiting cars are distributed, distribution of other cars is conducted in regards to available capacity and sums of the bids. We assume controlling agent to know information about current density levels on the available roads and distribute cars accordingly using speed parameter, described in formula 3.8.

Together with Mesa framework, described in section 2.4, we use several additional libraries: Numpy and Scipy.stats for the simulation of the output from the environment using normal distribution, Pandas and Xlrd for working with data from the wind farm and Matplotlib for creating the graphs.

Chapter 4

Results

As a result, the study of the theory made it possible for us to understand considered problems from different angles and to focus on the goals of the experimental part. In this chapter the results of conducted simulations for the set of considered problems are shown.

4.1 Smart house simulation

For the smart house, we model interactions between demand-side and supply-side agents. Demand-side agents represent parts of the smart house system, such as two heaters in different rooms, light system, and heated floor. Energy battery agent also considered as a possible consumer as it needs to maintain certain energy level for the battery. Supply-side agents represent renewable energy sources for the smart house such as solar and wind energy. For our model, we calculate amounts of energy in kWh and consider hourly time steps.

In order to calculate the demand for heating the rooms, we use specific heat formula, represented by the equation 3.3. Parameters for the formula, which were used in the simulation, described in table 4.1.

parameter	value
c_p	$1.00 J/gK$
p	$\approx 1275 g/m^3$
A	$15 m^2$
h	$2.5 m$
m	pAh
ΔT	$t_{desired} - t_{indoor}$

TABLE 4.1: Heater formula parameters

Indoor temperature is simulated using normal distribution and data from [38]. Example of simulated levels of indoor temperature for two heaters can be seen in the figure A.1 in Appendix A.

Amount of occurrences on this figure shows how many times during simulation period particular temperature level was simulated. Each room corresponds to different heater agent. Heaters have a parameter of minimum temperature level which should be maintained, equal to $20°C$.

Energy demand from the light system depends on the difference between levels of outdoor light and desired light level. The parameter of the desired level for the indoor light represented by the set of profiles, each of which corresponds to the particular level of light intensity. In the table 4.2 implemented set of profiles described.

User profiles	illumination level (lux)
4	1500
3	500
2	100
1	0

TABLE 4.2: Desired indoor light level profiles

User profile 4 corresponds to the maximum available illumination level based on recommended light level value for very detailed work. User profile 3 corresponds to the recommended illumination level for office and study work. User profile 2 corresponds to the light level for the basic orientation. User profile 1 implies that no illumination is needed so the light turns off.

As the outdoor illumination level depends on many various conditions we decide to use discretized values based on illumination levels for the overcast day, sunset and sunrise[40]. In order to reduce the difference between possible outdoor and indoor illumination levels, for the outdoor illumination, we decide to use values close to indoor illumination as in reality influence of outdoor illumination on the indoor light level can be reduced by different factors. Using dependence on time and weather types we can still maintain logically correct relations between outdoor and indoor light while reducing the complexity of our model.

If the current time is less than 7 a.m. we consider light level to be minimal, which for our model is equal to 20 lux.

At 7 a.m. we consider a sunrise with a maximum light value of 400 lux for the sunny weather, 100 lux for the partly cloudy weather and 40 lux for the cloudy weather. In case of rainy weather illumination level is set to minimum. For the time of a day between 7 a.m. and 18 p.m., we have maximum illumination level of 2000 lux in case of sunny weather, 200 lux for the partly cloudy weather and 100 lux for cloudy weather.

At 18 p.m. we consider a sunset, which implies illumination level of 400 lux for the sunny weather and 80 lux for the partly cloudy weather. For the time after a sunset and before the next sunrise illumination level is set to minimum.

Example of simulated outdoor light levels during 24 hours can be seen in the figure A.2 in Appendix A.

For the formula 3.5, which is used to calculate amount of energy needed for the light agent, we use the same room area parameter A = $15m^2$ as for the heaters, parameter of lumens per watt lm_w is equal to 90 $lumens/W$, which is assumed to be close to this parameter for the LED lamps based on data, described in [39].

The demand of agent, responsible for the heated floor is set based on data about heating floor running costs for the typical bathroom[41] and is equal to 0.8 kWh.

For some of the agents such as agents, responsible for heaters, light and heated floor, we assume the fact of somebody being in particular room where device corresponded to the agent is placed, to influence the demand. Probability of a person being in the room or outside depends on day and time. Example of the simulated probability of a person being in any room during 24 hours is shown in the figure A.3 in Appendix A.

In order to simulate solar energy, we use values from the dataset, provided by my advisor. After calculating average hourly amounts of solar energy, we use these values and normal distribution formula 3.4 to create new values for our simulation. For the purpose of making amounts of acquired solar energy vary based on time of the day and possible weather condition, we use set of weather coefficients, presented in the table 4.3.

weather types	coefficient
sunny	1.3
cloudy	0.8
partly cloudy	0.3
rainy	0

TABLE 4.3: Weather type coefficients

By multiplying our energy output values on these coefficients, we can simulate the influence of weather condition on the amount of solar energy.

Example of simulated solar energy levels during 7 days can be seen in the figure A.4 in Appendix A.

In contradiction to solar energy, wind energy is less predictable. In our model, we use scaled data from the wind farm to simulate wind energy. In the figure A.5 in Appendix A example of simulated levels of wind energy during 7 days is shown.

4.1.1 ZI decentralized approach

One of the main goals of this simulation is to test how agents can share common good which is, in this case, represented by a renewable energy. Figures 4.1 and 4.2 show results of energy distribution between demand-side and supply-side agents in the smart house. Period of simulation is 7 days or 168 hours.

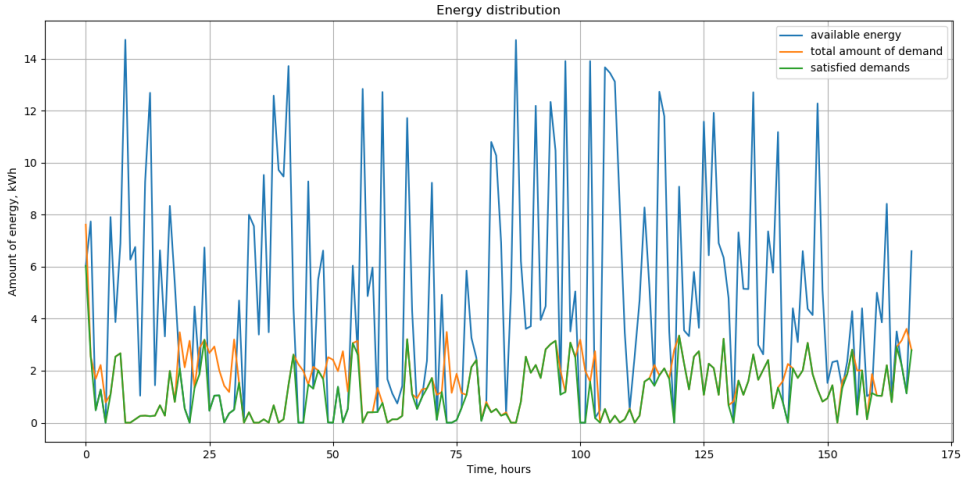


FIGURE 4.1: Energy distribution during 7 days.

For this simulation we use pure renewable energy sources such as solar energy and wind without additional energy stored in battery. This help us to show how amount of satisfied demands vary depending on available energy. Example of energy distribution in case of storage and all renewable energy sources included can be seen on the figure A.6 in Appendix A.

In the figure 4.2 example of energy distribution in case of only solar energy source available is shown. The simulated period is the same as for the previous case.

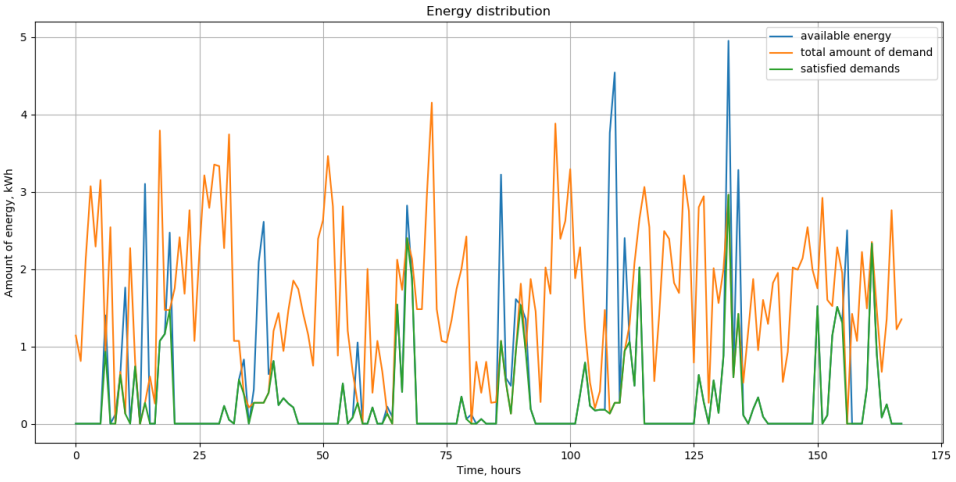


FIGURE 4.2: Energy distribution during 7 days with only solar energy.

In this case amount of demands would be often higher than available amount of energy supply. But still, in each trading episode maximum of available renewable energy would be distributed in regards to supply and demand amounts.

4.1.2 Decentralized approach with learning

For the smart house, we consider two learning heater agents with the goal to maintain particular temperature range. For these agents, we use discretized set of states and actions they can perform, described in table 3.2. Based on the energy balance of the smart house, heater agents can change their decisions and reduce or maximize the amount of desired energy.

For the learning algorithm, described in 3.1, we use recency parameter $r = 0.8$ and experimentation parameter $e = 0.5$. Initial propensity values are equal to 1. As we do not have many different strategies for each state of the agents, even with high recency parameter both agents are able to learn that strategy to use fast. On the other hand, with experimentation and recency parameters we can give agents an opportunity to change their behavior model.

In the figures 4.3 and 4.4 we can see how decisions of agents change in regards to the state of the environment during simulation period of 7 days.

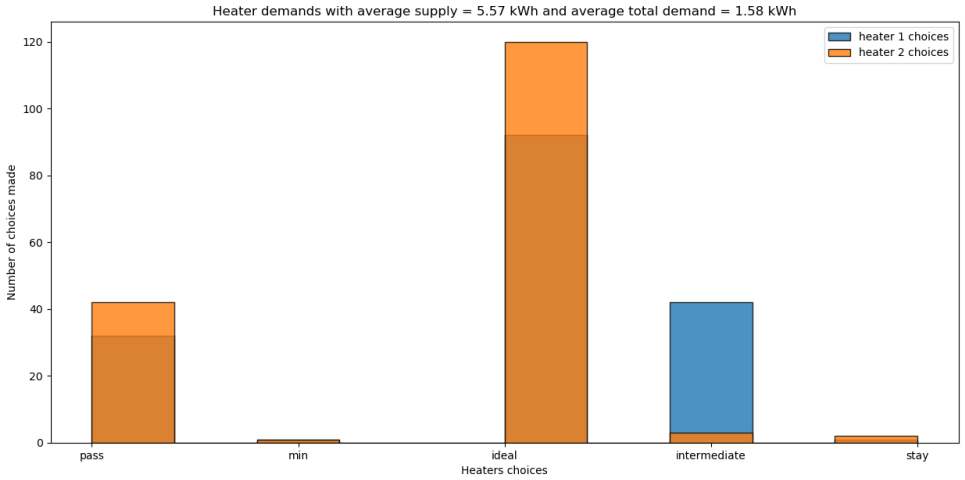


FIGURE 4.3: Heater agents choices during 7 days in case of high level of energy supply.

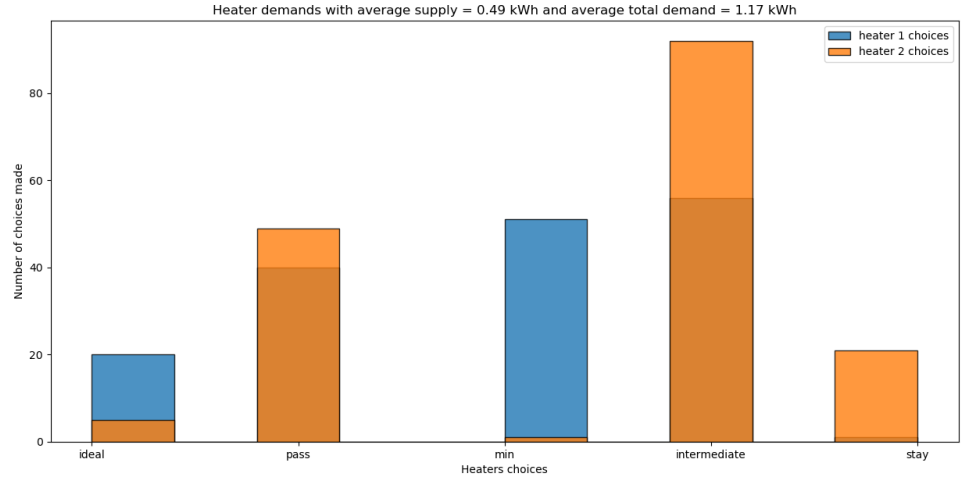


FIGURE 4.4: Heater agents choices during 7 days in case of low level of energy supply.

Parameter of number of choices made shows how many times particular type of action was chosen during the simulation period.

Figures 4.5 and 4.6 show changes in amounts of energy demands from the heaters based on available renewable energy.

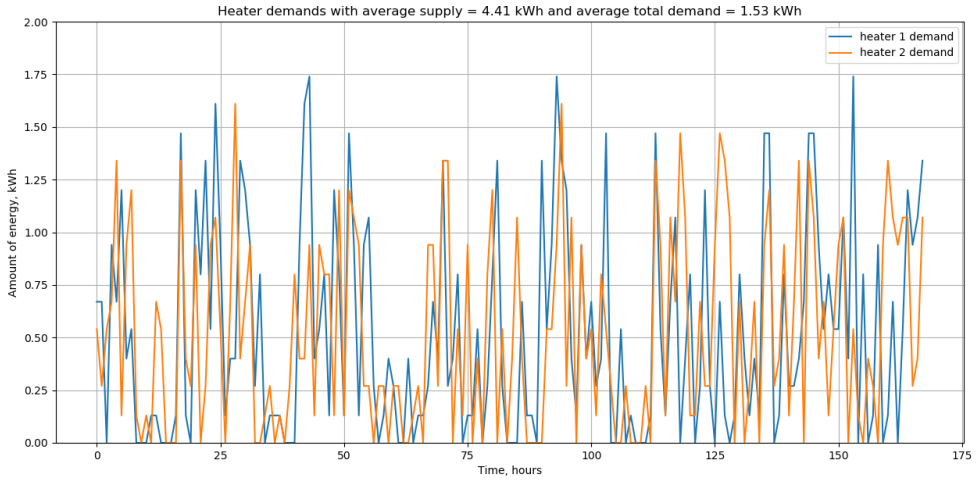


FIGURE 4.5: Heater agents demand values during 7 days in case of high level of energy supply.

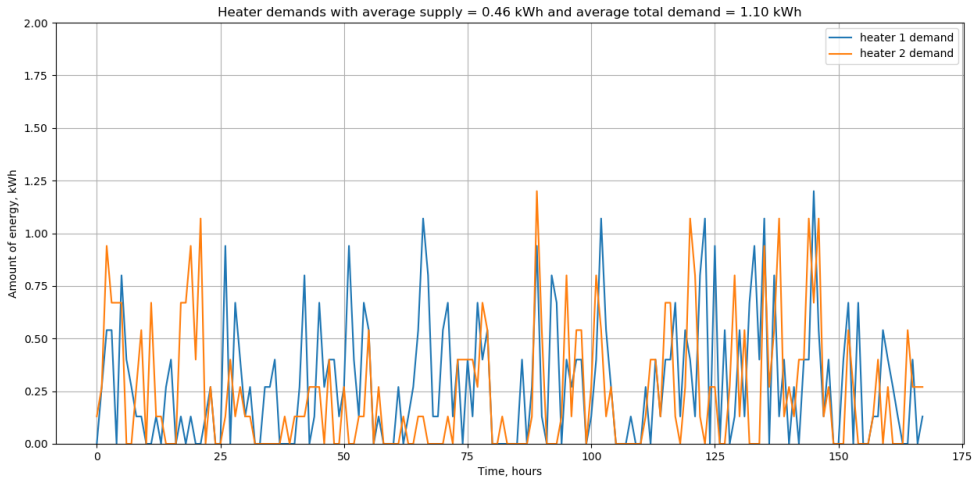


FIGURE 4.6: Heater agents demand values during 7 days in case of low level of energy supply.

4.1.3 Centralized approach

For the centralized approach, problem of energy distribution is handled by the controlling agent using the algorithm 1. For this case fitness of each solution is calculated by the fitness function, described in 3.6. For the algorithm we use number of iterations $NI = 300$ and number of iterations for partner set $NIT = 5$. Figures 4.7 and 4.8 demonstrate energy distribution during simulation period of 7 days.

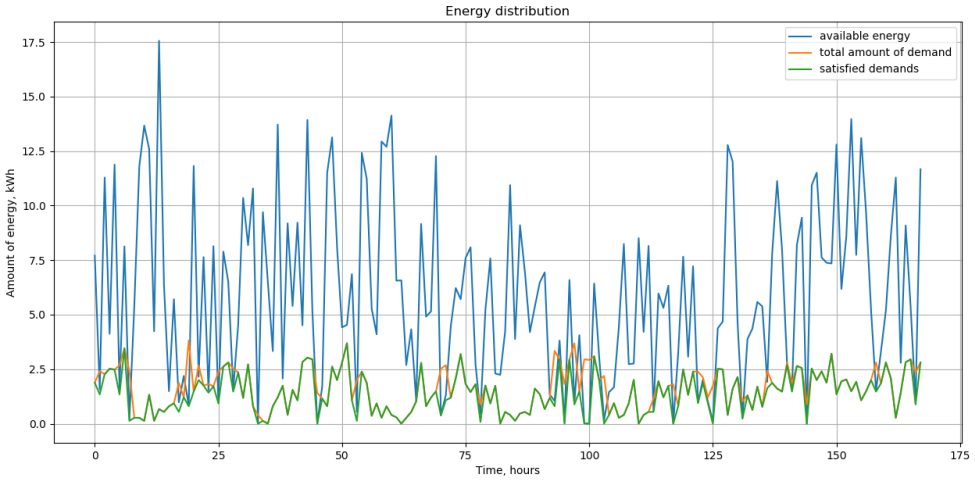


FIGURE 4.7: Energy distribution with centralized approach during 7 days.

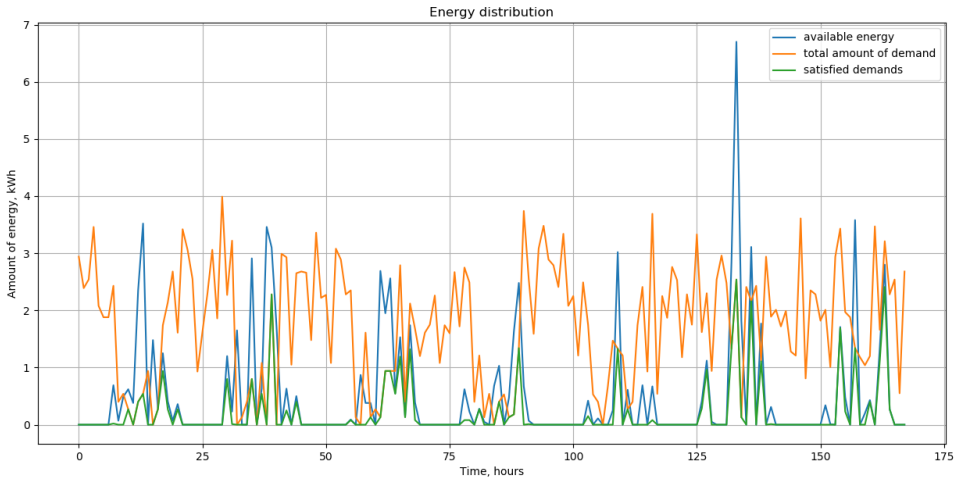


FIGURE 4.8: Energy distribution with centralized approach during 7 days with only solar energy.

These figures demonstrate energy distribution in case of wind and solar energy as available energy sources in the figure 4.7 and only solar energy in the figure 4.8.

4.2 Smart parking simulation

For this simulation, we model the distribution of parking slots between car agents. Amount of free parking slots represents common good which is distributed between car agents with the desire to park. For the simulation, we use hourly time steps. Each car has a parameter of

desired time for parking $t \in [1, .., 5]$, with the minimum value equal to one hour or one time step.

Amount of cars with the desire to park depends on the time of the day and increases during peak hours based on traffic congestion data from [46] and [47] with probability of the car to have a desire to park increasing from 7 a.m. to 9 a.m. and from 15 p.m. to 17 p.m. In the figure B.1 in Appendix B an example of simulated levels of demands for the parking during 24 hours is shown.

4.2.1 ZI decentralized approach

Figure 4.9 shows distribution of parking slots during simulated period of 7 days. The total amount of car agents is equal to 100, the total amount of parking slot agents is 80. As we use probability values for car agents to simulate their desire to park, amount of car agents can vary for each trading period.

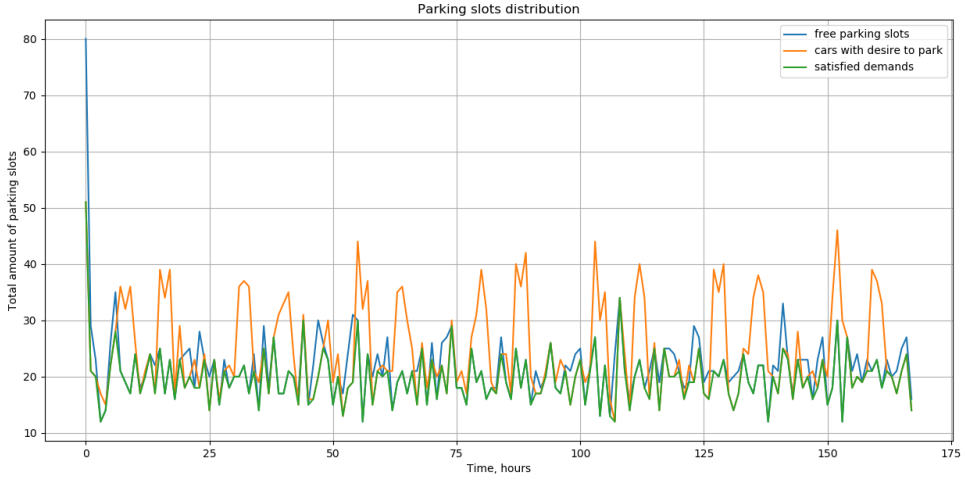


FIGURE 4.9: Distribution of parking slots during 7 days.

In order to distribute parking slots, we use the similar market-based model as for the smart house. Distribution of parking slots conducted through the trading process between car agents as a buyers and parking slot agents as a sellers. But in this case, we focus more on price choices for bid and ask prices. For ZI approach each agent has set of price choices $p \in \{50, 200\}$.

In the figure B.2 in Appendix B levels of average price for the deals during the simulated period of 7 days are shown.

At the start of each trading car agent chooses its bid price from the price set and parking slot agent gets its ask price. If bid price of the buyer is more or equal to ask price of the seller, they

can have a deal. Otherwise, buyer's bid is rejected and buyer needs to wait to be paired with one of the sellers again. Levels of rejected and satisfied deals during the simulation period of 7 days can be seen in the figure 4.10.

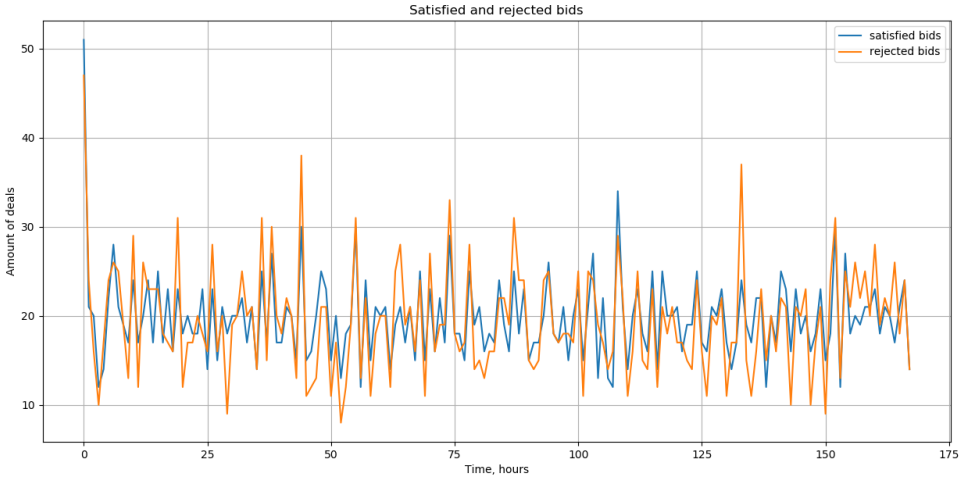


FIGURE 4.10: Satisfied and rejected deals during 7 days.

4.2.2 Decentralized approach with learning

In this approach, we consider car agents and parking slot agents as learning agents with sets of bid decisions. Bidding decisions are discretized to integer values between lower and upper bound of market prices, therefore it makes every price strategy for the agents be in the set $s \in S = \{50, 60.., 240\}$. In order not to make the number of possible price strategies too big we increased the gap between them. The outer price which represents price for other parking slots and used to calculate the amount of reward for the buyer is set to 175 NOK, which is slightly higher than the average price for the local market and is increased during peak hours, based on traffic congestion. Maximum outer price is equal to 262.5 NOK which is 1.5 times higher than usual outer price.

Example of levels of rejected and satisfied deals during the simulated time period of 7 days can be seen in the figure 4.11.

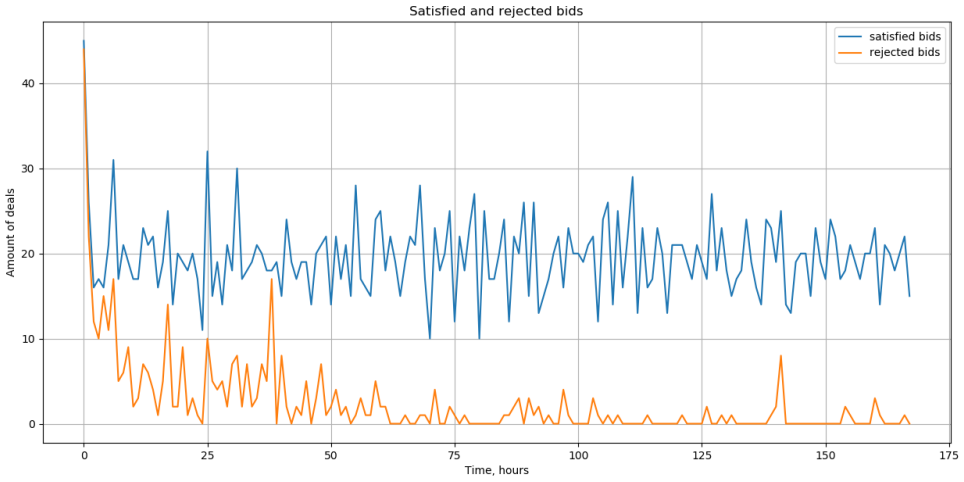


FIGURE 4.11: Satisfied and rejected deals during 7 days with learning agents.

For the learning algorithm, we tried to use different values for the parameters. For this simulation we decided to use the same propensity values equal to 1, recency parameter $r = 0.8$ and experimentation parameter $e = 0.5$, as for the smart house model.

4.2.3 Centralized approach

For the centralized approach we use the same algorithm as for the previous model, but instead of energy loads, cars with a desire to park are encoded. The fitness of every solution is calculated using fitness function, described in 3.7. Figure 4.12 shows the distribution of parking slots during the simulated period of 7 days. As in this case solutions include more elements and more possible combinations, size of initial population and amount of iterations for finding a suitable solution for the algorithm are increased. For this simulation we use number of iterations $NI = 600$ and number of iterations for partner set $NIT = 5$.

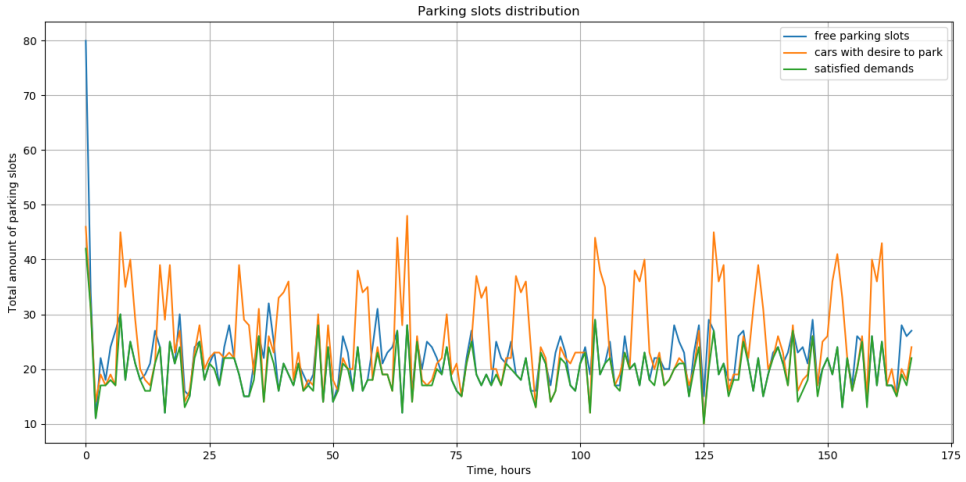


FIGURE 4.12: Distribution of parking slots with centralized approach during 7 days.

4.3 Smart traffic simulation

For the traffic problem, we consider the hourly capacity of the roads as common good which is distributed between car agents with a desire to pass through the gate on the main road or two barriers on the sideways. Gate and barriers are represented by the agents which can provide the right for the cars to pass. For this simulation, we implement 3 types of cars with different sets of prices for each type. Prices for each type described in the table 4.4.

Car types	Price limits
car	$price \in \{46, 56\}$
lorry	$price \in \{132, 162\}$
emergency	$price = 0$

TABLE 4.4: Price limits for car types

Type *emergency* is considered as the type with priority to pass without paying. We consider each car agent to choose the type during initialization with particular probability for each choice. With this, we can have different amounts of cars for each type.

Example of type distribution during the simulated period of 7 days is shown in the figure 4.13.

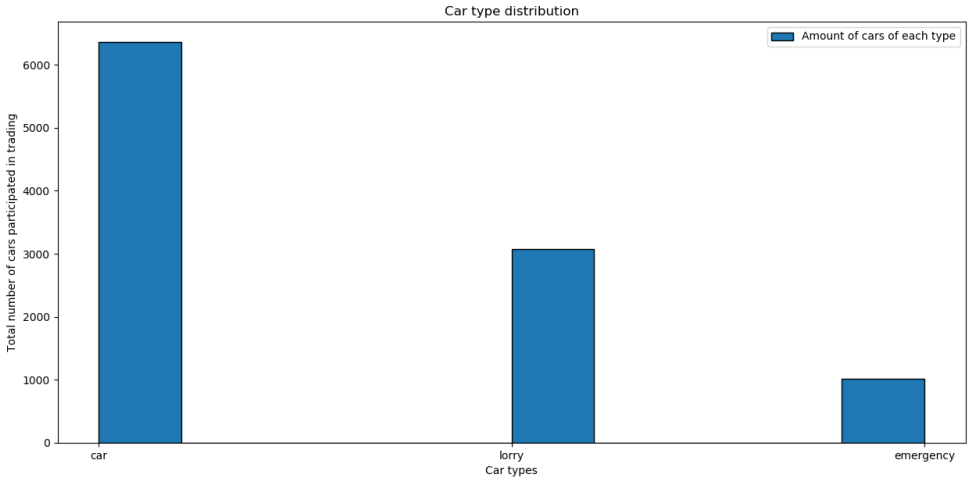


FIGURE 4.13: Levels of cars of each type during 7 days.

Total number of cars participating in trading shows the number of cars of each type which took part in the trading process during the whole simulated period.

4.3.1 ZI decentralized approach

For this case, we simulate interactions between 100 car agents, one gate agent responsible for main road and two barrier agents responsible for the sideways. Similarly to other cases, we use hourly time steps for the simulation. Car agents represent buyers with the desire to pass, road agents represent sellers which provide the right for the cars to pass.

Road agents have the different hourly capacity - gate agent has an hourly capacity equal to 40 as we assume the main road to be able to have the higher amount of cars per hour. Barrier agents have an hourly capacity equal to 20. As a result, we have the total hourly capacity for the roads equal to 80 cars per hour. For the distribution process, we use the same market model with P2P trading between buyers and sellers. Example of the roads capacity distribution during the simulated period of 7 days can be seen in the figure 4.14.

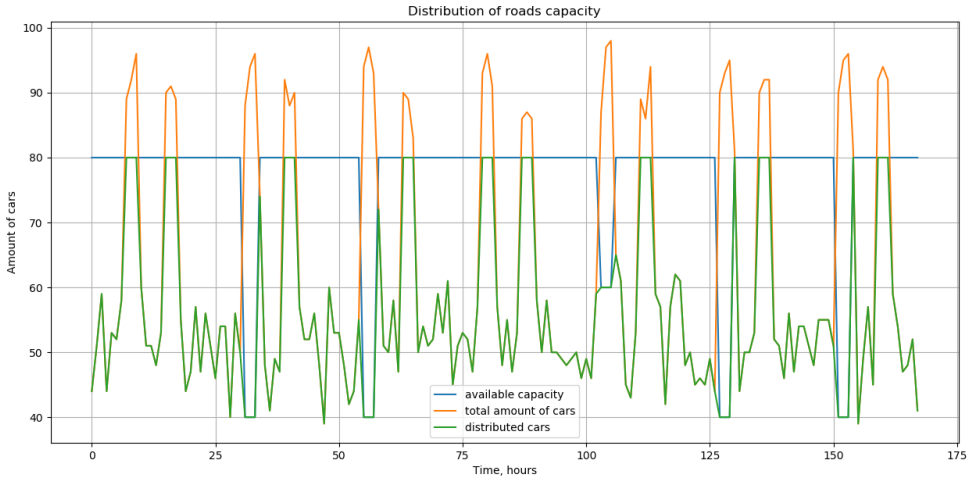


FIGURE 4.14: Distribution of roads capacity during 7 days.

For this case, we do not consider different parameters such as length or maximum speed limit for the roads as ZI agents assumed to ignore information about the environment and act only according to constraints defined by the market model. This makes distributed amounts of cars between the roads be dependent only on results of the trading. An example which shows how cars are distributed between the roads during the simulated period of 7 days, can be seen in the figure 4.15.

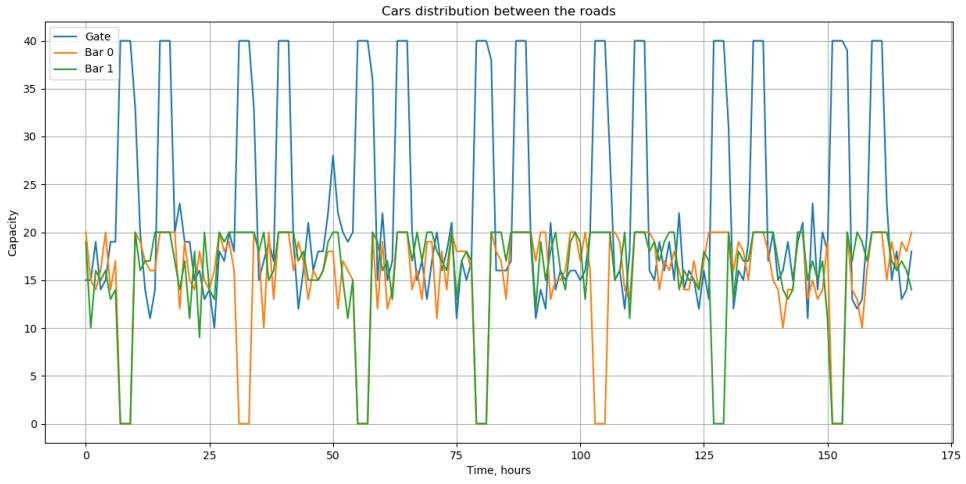


FIGURE 4.15: Distribution of cars between the roads during 7 days.

4.3.2 Decentralized approach with learning

For the decentralized approach, we consider car agents be similar to reactive agents with the goal to find fastest and possibly cheapest way from the available roads. For every road, we

consider parameter of length and maximum speed limit. For the main road, represented by the gate agent we consider length be equal to 7 km and speed limit be 70 km/h. For both sideways, we have length be equal to 10 km and speed limit be equal to 60 km/h. In our calculations, we assume some amount of freedom for the interpretations as the main purpose of these parameters is to distinguish the main road and sideways and make the main road more likely to be selected in case of equal car density levels on all available paths.

Example of cars distributed between the roads during the simulated period of 7 days using described approach can be seen in the figure 4.16.

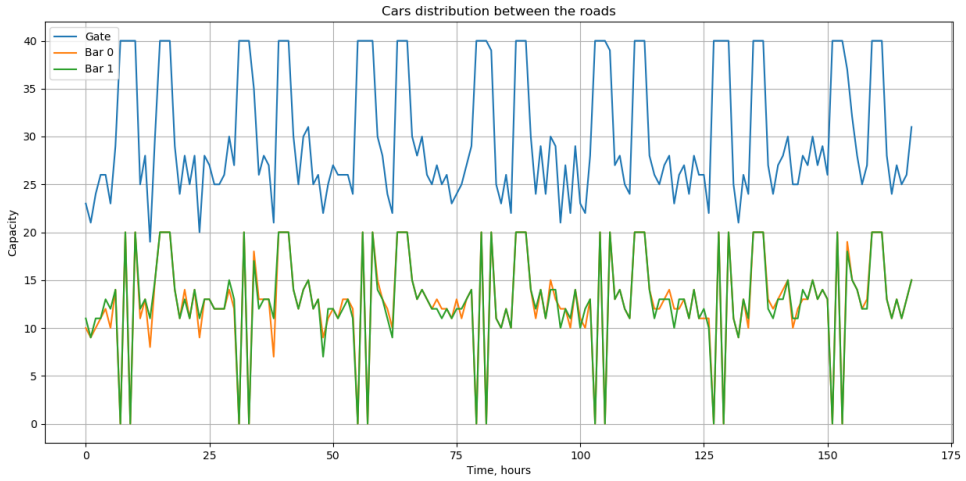


FIGURE 4.16: Distribution of cars between the roads during 7 days with car agents choosing the roads.

Capacity parameter shows the highest capacity value between all the roads, which is for this simulation is equal to 40 cars per hour for the main road.

An important aspect of this learning approach lies in implemented behavior model for the road agents which are considered as learning agents with the discretized states based on car density levels. For each state road agents have set of price strategies, based on price sets, shown in the table 4.4, discretized to integer values. In the table 4.5 states and corresponding density levels for them are shown.

states	density levels
empty	$0\% \leq x \leq 20\%$
low	$20\% < x \leq 40\%$
intermediate	$40\% < x \leq 60\%$
packed	$60\% < x \leq 80\%$
full	$x \geq 80\%$

TABLE 4.5: States of the roads based on density levels

where x is the current density level. Percentage is based on maximum capacity value for the road.

To show how price strategies are chosen according to density levels we consider the distribution of price strategies for the *lorry* type of the car for the gate agent during the simulated time period of 7 days.

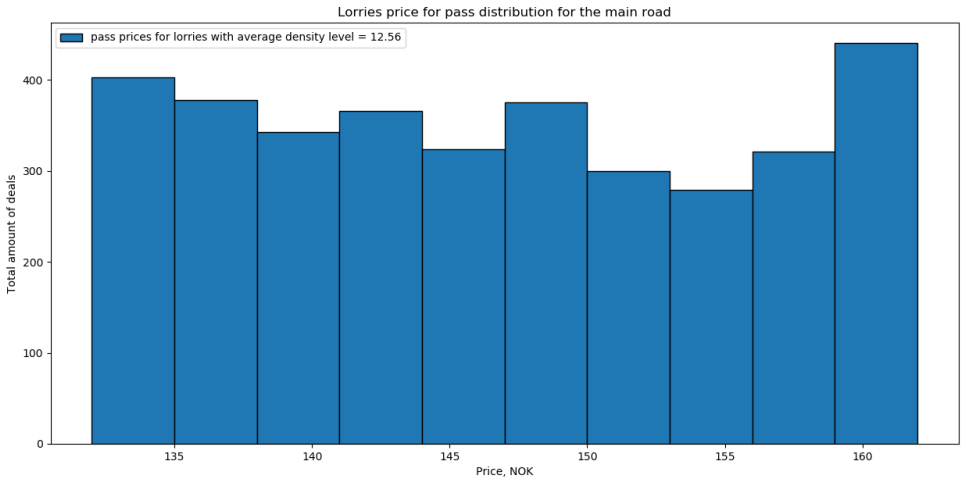


FIGURE 4.17: Distribution of prices for the lorry on the main road with low density during 7 days.

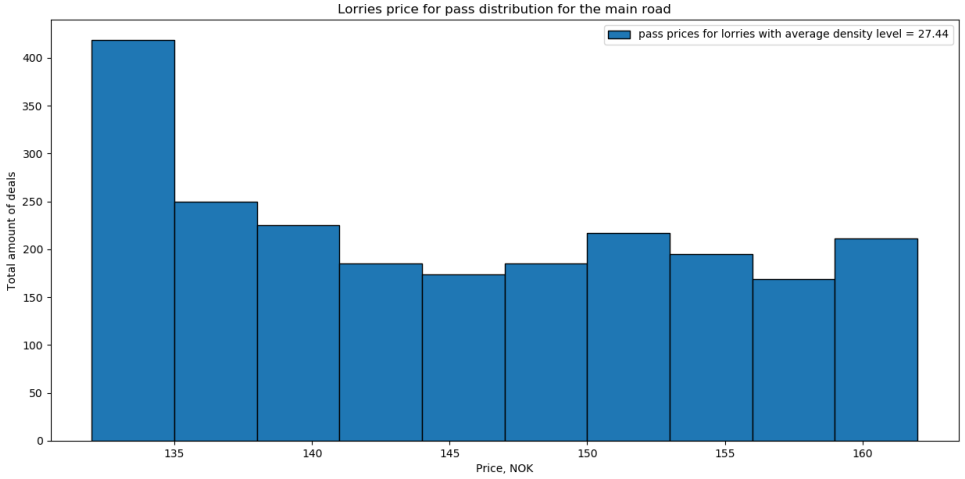


FIGURE 4.18: Distribution of prices for the lorry on the main road with high density during 7 days.

In the figure 4.17 distribution of prices in case of relatively low level of density on the main road is shown, with average density value during a simulated time period equal to 12.56 considering the maximum capacity of the main road being equal to 40.

In the figure 4.18 situation with the relatively high density level on the main road is shown with average density during the simulated period being equal to 27.44 with the same maximum capacity for the road as for the previous case. Set of prices is also the same.

For this case, we change recency and experimentation parameters to 0.1 in order to increase the speed of learning due to the high amount of choices and desire to make our agents learn specific behavior models for every state. Initial propensity values which were used are the same as for other cases.

4.3.3 Centralized approach

For this case, we encode car agents with a desire to pass through the gate at the main road or barriers at two sideways similarly to energy loads or parking cars. In order to calculate fitness for each possible solution, we use fitness formula, described in 3.10. As we need to handle emergency cars with the priority to pass and cars waiting in a queue, we distribute them first and use an algorithm 1 to distribute other cars according to available capacity. Number of iterations NI and number of iterations for the partner set NIT are the same as for the smart parking simulation.

Example of distributed capacity between car agents during the simulated period of 7 days can be seen in the figure 4.19.

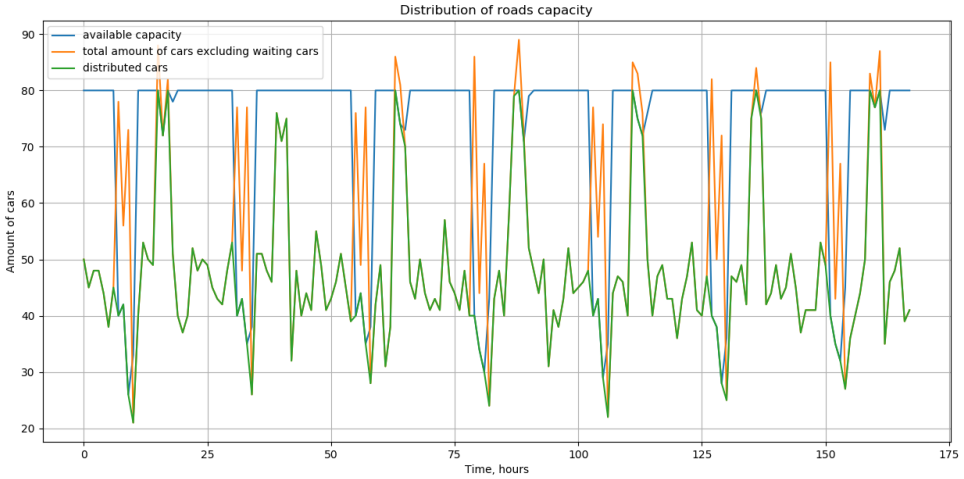


FIGURE 4.19: Distribution of roads capacity with centralized approach during 7 days.

Chapter 5

Discussion

5.1 Results interpretation

The primary objectives of this work were to create a set of models for different problems related to Smart City with several approaches applied for each model, study how for each problem agents can find a way to share a common good and demonstrate intelligence in adaptability and problem-solving capabilities, as well as investigate the possibilities of applying similar approaches for different cases in order to demonstrate flexibility and universality of multi-agent solutions.

With the chosen way of implementation and considered set of problems we managed to design multi-agent system architecture for different cases and create several models for each problem. Such concepts as P2P market design, a market model with self-interested zero-intelligent agents, reinforcement learning in regards to agent behavior, decentralized and centralized agent-based models, were studied and applied to the design of different approaches for the modeled problems. For the decentralized models, we used market design, described in [19], which allows interactions between sellers and buyers in the P2P local market. Distribution of common good, in this case, was provided via interactions between the active seller and buyer agents on the local market. For the centralized approach, we used a controlling agent with an algorithm for the distribution based on genetic algorithm to encode energy loads or other demands and find a solution which can be satisfied in regards to the available amount of supply.

As we consider each problem in terms of distribution of common good, for every simulation we have agents as consumers, which have a goal to acquire the particular amount of limited resource, provided by supply agents. In case of smart house, as described in section 3.2.1, agents, responsible for different house systems such as two heaters, light and heated floor, are designed and implemented as pure consumers with energy demands, which can be satisfied by

renewable energy, provided by sellers such as agents responsible for solar energy and wind energy with agent, responsible for energy battery as prosumer which can also buy energy to maintain battery balance. Due to the overall energy balance of the house, for the simulation results, we used pure renewable energy sources to show changes in satisfied demands in case of high and low energy levels. For initializing amounts of supply and demands we used data, acquired from the different sources including data, derived from the datasets, provided my advisors.

In the model, presented in section 3.2.2 we considered car agents as consumers with a desire to park and parking slot agents as sellers which can sell the right for the car to park at the certain parking slot. Amount of available parking slots was considered as common good in this case. Using data about peak traffic hours we were able to differentiate amounts of demands according to the time of the day.

For the model of the smart traffic with gate and barriers, presented in section 3.2.3, we considered the available hourly capacity of the roads as a common good, distributed between different types of car agents with the desire to pass through the main gate or two barriers.

In chapter 4 we presented results of the simulations of implemented approaches for each considered problem with various parameters.

For the smart house we simulate the distribution of renewable energy between smart house agents for the particular time-period equal to 7 days or 168 hours as we have hourly time steps for the models. Changes in amounts of energy demands and supply per hour are supported with different weather states and time of the day. Results of the smart house simulation described in section 4.1.

Results of smart house simulation with zero-intelligent agents, described in section 4.1.1 aimed to show how renewable energy is distributed between agents according to implemented market-based model. We considered cases with different levels of available energy. As expected, the process of distribution follows the same rules regardless of the amount of available energy, traded on the local market, and amount of distributed energy during each hour corresponds to amounts of supply and demand so that maximum amount of demands would be satisfied if possible. As can be seen from the figure 4.2, even in case of low energy levels every consumer will acquire as much energy as it can in regards to the available amount of energy.

With results of the simulation with learning agents, described in section 4.1.2 we wanted to show how agents can learn to use particular strategy in accordance with the states of the environment on the example of two heater agents with sets of states and strategies for each state.

Figures 4.3 and 4.4 show that, in case of sufficient amount of energy both heater agents learn to maximize their demands and heat the rooms to ideal desired temperature (which corresponds to *ideal* choice) or close to ideal desired temperature (which corresponds to *intermediate* choice)

instead of heating to minimum (which corresponds to *min* choice). On the other hand, in case of lack of energy, heater agents more frequently choose actions which allow reducing energy demand such as heating to the minimum temperature if possible or heating to intermediate temperature level, rather then heating to the maximum value.

From the figures 4.5 and 4.6 it can be seen that, in case of sufficient amount of energy, heater agents have higher demands as a result of deciding to heat the room to the higher temperature more frequently. However, in case of low level of energy supply, heater agents reduce their demands which allow saving more renewable energy. As a result, less amount of energy is needed to be bought from the grid. This shows how learning agents can adapt their behavior according to changes in the environment.

Results of the simulation of the centralized approach, presented in section 4.1.3 had the goal to show the distribution of renewable energy, handled by the controlling agent using the implemented algorithm. As a result, we observed that, using genetic algorithm-based approach with sets of simple constraints, a controlling agent trying to find optimal or close to optimal solutions in terms of energy distribution according to demands and available supply for each hour during the simulation. For only one agent to do it, sufficient amount of information about supply and demands is needed. For the implemented algorithm high initial population and increasing amount of iterations for finding the solutions can influence the quality of results. As in this case we do not consider the flexibility of the energy demands, each demand is satisfied in regards to the available amount of renewable energy, if demand is higher than available amount of energy, it needs to be satisfied from the other sources such as a grid. This implies that particular energy demands which cannot be fully satisfied, are omitted from the solution. For this case, we implemented additional check after main distribution in order to at least partly satisfy energy demands which were omitted.

Results of smart parking simulation, presented in section 4.2 had the goal to show how a limited number of parking slots can be distributed between cars. We considered 7 days as the simulated time period. Amount of demands for the parking corresponds to traffic peak hours. For the simulation we used a total number of car agents equal to 100 and the total amount of parking slots equal to 80.

Observing the results of parking slots distribution, shown in figure 4.9 for the zero-intelligent approach, described in section 4.2.1, we can say that distribution is following amounts of demands and available amounts of parking slots. As for each hour amount of cars with a desire to park can be different, situations where available parking slots were enough to satisfy all demands, and the situations where not every demand was satisfied, occurred during the simulated period.

At the start of simulation amount of demands is higher as from the total amount of cars no cars are currently parked. After the first trading period, some amount of cars was able to acquire

available parking slots and was excluded from further trading episodes for some periods of time. This cause total amount of demands to be reduced.

In the results of simulation of smart parking with learning agents, shown in section 4.2.2 we demonstrated how agents can learn to use particular strategies through their interactions on the local market. Comparing levels of successful and rejected bids during the same simulated time period for zero-intelligent agents, shown in the figure 4.10, and learning agents, shown in the figure 4.11, we could see that amount of rejected bids in case of learning agents is decreasing during simulated time period, as a result of both sellers and buyers are learning to use specific price strategies. As both seller and buyer get a reward, equal to zero as a penalty in case of a rejected deal, they will try to find a way to reduce the amount of rejected bids regarding amounts of individual rewards, which can be beneficial for the market efficiency.

During simulation tests, we considered different values for the recency and experimentation parameters for the learning agents. As a result, we managed to observe that with a decrease of recency and experimentation parameters influence of individual rewards for the agents increases. As we reward buyer and seller agents differently, with higher reward for the higher ask price of the seller and higher reward for the lower bid price of the buyer in case of successful deal, the amount of rejected deals in case of recency and experimentation parameters, equal to 0.1 decreases slower comparing to higher recency and experimentation parameters, used to acquire the results, shown in section 4.2.2. This is also caused by the fact that agents are less prone to try other strategies in case of low experimentation and recency parameters and it can take more time for them to change their decision patterns.

For the centralized approach for the smart parking model, we considered each car with a desire to park similarly to the energy load in case of a smart house. According to results for this approach, shown in the figure 4.12, we can conclude that even with increasing number of agents in the system, with clearly defined sets of constraints, the controlling agent manages to find a solution and distribute common good in order to satisfy as many demands as possible. Although, with increasing number of possible combinations it can take more iterations for the algorithm to find a better solution.

In the smart traffic simulation, hourly capacity of the roads is considered as common good, distributed between car agents with a desire to pass through the gate on the main road or barriers on the two sideways. Agents, which corresponds to the gate and barriers are considered as sellers, which provide the right to pass for the cars.

For the car agents we implemented three different types with specific prices for the pass for each type, as shown in table 4.4. Among implemented types *emergency* type has the priority to pass, but also the lowest chance of occurrence during all trading episodes, as demonstrated on the figure 4.13. For the simulation, we considered the total amount of car agents equal to

100 and the total amount of hourly capacity equal to 80 with a capacity of 40 for the main road and 20 for both sideways.

As shown in the figure 4.14, distribution of the cars corresponds to capacity values of the roads. During traffic peak hours amount of cars with a desire to pass through the gate or barriers increases. During these peak hours, barriers can block sideways which cause the amount of total available capacity to be reduced but both sideways can be open based on density situation on the main road. This allows the amount of available capacity to vary for every hour. Amount of cars waiting in the queue also influence the amount of available capacity for the buyers as these cars have the priority to be distributed at the start of each trading.

As some of the cars are not able to pass during particular trading episodes they are distributed at the following trading. This, together with varying total capacity value and peak traffic hours, can create higher peaks of demand, but due to constraints of the market model only demand which corresponds to the amount of available supply, would be satisfied. This allows us to distribute the maximum available amount of cars in accordance with provided capacity for every hour.

In the simulation results of the decentralized approach with learning for the smart traffic, described in section 4.3.2 we gave car agents an opportunity to decide which road to choose based on density situation on the roads and price for the pass. In each trading, car agents find the most suitable seller based on ask price for the pass and parameters, calculated using formulas, described in 3.8 and 3.9. If we compare the distribution of cars between the roads in case of zero-intelligent approach, shown on the figure 4.15 and distribution of cars for this approach, shown on the figure 4.16, we can see that cars, which chose sellers based on road parameters and price, are distributed more evenly.

In this approach, we also showed how road agents can learn to use specific price strategies based on density levels on the roads and trading results. For instance, we wanted road agents to have higher ask prices in case of low level of car density and have lower prices in case if density is more than 60% of maximum road capacity in order to make car agents pay less in this case. To implement learning aspect for this model we combined such aspects as states with sets of choices from the smart house and price strategies from smart parking. As a result, for our road agents, we have set of states based on density levels and two sets of price strategies based on types of cars. For this approach, we used low recency and experimentation parameters equal to 0.1 as they occurred to be more suitable for increasing the speed of learning for the agents in case of a high number of possible strategies. On the other hand, higher recency and experimentation parameters provide more opportunities for the agents to change their behavior as in this case each choice made has a lesser influence on probabilities for other strategies to be used.

The figure 4.17 shows the distribution of prices for *lorry* type of car in case of relatively low level of density on the main road. In this case, we increased reward value for the road agent to use higher ask prices. As we consider seller agent to have a reward, equal to zero as a penalty in case of a rejected deal, results of the trading process have a high influence on the learning results. Despite this, it can be seen that number of times when the higher ask price for the deal was chosen during all simulated period is higher than that for the lower prices, which is close to the behavior pattern we wanted our agent to learn.

On the figure 4.18 we show the situation with the relatively high density level on the main road. We used the same time period for simulation and the same set of price strategies. In case of high density level, we increased the reward for the lower ask prices. This corresponds to the higher amount of satisfied deals, and, as a result, a lesser amount of penalty. Consequently, this behavior pattern is easier for the agent to learn. As expected, a number of times lower ask price was chosen during trading episodes within the simulated period is indeed higher, which corresponds to desirable behavior model for this state.

The centralized approach for the smart traffic, described in section 3.2.3 is modeled similarly to smart parking with car agents encoded as loads. Results of the simulation of the centralized approach for the smart traffic described in the section 4.3.3. The main difference in this approach is related to the *emergency* type of cars with priority to pass and cars, waiting in queue to pass. We distribute them first and then other cars are distributed according to available capacity. On the figure 4.19 we can see an example of how the available capacity of the roads is distributed between the car agents.

On the assumption of information, described above, we can conclude that regardless of the number of agents in the system with clearly defined sets of rules and constraints multi-agent system can maintain stability and demonstrate the same levels of problem-solving capabilities. According to demonstrated results, it can be said that even self-interested zero or low-intelligent agents can maintain the stability of the system and demonstrate problem-solving capabilities and adaptability. With market-based model agents which have different needs, are able to find ways to share limited resource through the interaction with each other. Using reinforcement learning, we can increase flexibility if agents behavior and make them better adapt to changes in the environment. On the market, memory and experience can help agents to find a particular strategy which can lead to more efficient interactions, such as, reduce the number of rejected deals.

For the centralized approach with one controlling agent, usage of the method, based on genetic algorithm with a set of constraints and high population of possible solutions allowed the agent to find a suitable solution. On the other hand, with increasing amount of encoded parameters time for finding a better solution can increase. We can assume that for the centralized approach more complex behavior model for the controlling agent could be needed in order to find better

solutions, whereas the decentralized multi-agent approach allows complex system behavior to emerge from the simple behavior models of agents.

Although each problem which was considered in our work was connected to different areas of Smart City, we could see that in terms of distribution of limited resource different problems could be considered similarly, as in each case we had some amount of common good which had to be distributed, be it parking slots, energy, or right to pass. In each problem we managed to represent distribution of common good through the distributed market-oriented approach or centralized genetic-algorithm-based approach and acquire the results.

5.2 Encountered problems

In this section, we discuss problems which were encountered during work on this thesis. Many problems were connected to design of architecture for the agent interactions on our models. For considered problems, we had to adapt our approaches in order to create a solution which corresponds to specific aspects of each problem. During the study of state-of-the-art, a number of papers, related to considered problems were analyzed in order to better formulate conditions for the models. As each considered problem can involve a significant number of parameters, which is a common aspect of the Smart City environment, some simplifications and scaling for the modeled problems had to be done.

Choosing the better way of implementation was also one of the encountered problems. For the implementation of the models, we considered different frameworks, several of which, including JADE, Madkit, osBrain, and Mesa, were tested in order to find the most suitable tool. Problems with JADE and Madkit were connected to specific aspects of these frameworks and architectures for agent behavior, which were these frameworks used. Some of these aspects were discussed in chapter 2. As for the thesis we had to model set of problems and apply several approaches for them, we considered general purpose MAS framework as more suitable tools. Framework osBrain was tested as it allows relatively fast implementation of multi-agent structures, but the implementation of the environment with different parameters which can be changed for different simulations was harder to do using such type of framework. We considered aspects of modeling and simulating of different solutions more important than the practical usage of communication protocols and therefore we used Mesa as framework made for general purpose agent-based modeling.

One of the most noticeable problems encountered during implementation of the models was the implementation of learning aspects. Among considered algorithms we chose Roth-Erev as it was relatively simple to modify and adapt for different solutions, but this algorithm also has several problems. First and furthermost was a need to tune parameters of the algorithm and amount

of reward. Depending on learning parameters and size of initial propensity values, the influence of reward can vary. In some cases, too high or too low amount of reward negatively affected learning results. In order to prevent this, we had to evaluate and scale reward differently for each implemented approach.

Another problem, connected to implemented learning algorithm was that reward, equal to zero as a penalty caused propensity values to decrease very fast, especially in case if agents had to be penalized a number of times until they manage to find a more suitable choice of actions. In the implemented algorithm each propensity value is updated according to received reward, which caused propensity values not only for penalized choice but for other choices also be slightly decreased. After getting a large number of penalties, propensity values could become almost indistinguishable from zero. To prevent this we had to update all propensity values using small positive constant. As we update every propensity value equally, higher propensity values still remain distinguishable.

With the increasing number of possible strategies, time of convergence for Roth-Erev algorithm also increases, so recency and experimentation parameters for the algorithm had to be tuned accordingly to increase the accuracy of results. On the other hand, with a small number of possible strategies even with high recency parameter agents were able to learn specific strategy relatively fast.

It should be also mentioned that in case of smart house models we modeled only parts of the possible energy loads but used the data about full renewable energy productions for the house. This caused the balance between demands and supply to be changed in a way that overall supply was often bigger than demands. For the simulations, we managed to consider different scenarios, but in case of modeling such type of the problem higher amount of demands and consumers could be considered in order to reduce the gap between supply and demand values and get more correct results.

The implemented way of encoding the loads for the centralized approach can be not very effective in some cases, especially with high amount of encoded elements as the algorithm will need more iterations and additional constraints in order to find the best solution.

Chapter 6

Further development

6.1 Possible improvements

In this section, we discuss possible further development for this work. By reference to the problems, encountered during work on the thesis, described in the section 5.2 we can propose a number of improvements for the further work. One of the ways to improve current results is to upgrade structure of simulated models with additional elements, other market designs and bidding strategies for the agents. Further development of this work can be related to models structure, implemented approaches and ways of implementation of the models.

Smart house model can be improved with increasing number of elements in the environment. Adding electrical devices such as TV, washing machine, air condition and elements such as an electric vehicle, as agents, modeled with bottom-up approach can provide an opportunity to create more diverse and dynamic local market for the smart house and consider the problem of energy distribution in more detail. The individual behavior for each agent also could be improved, for example, in our models we consider battery agent as a zero-intelligent agent with no learning capabilities, but with the learning aspect, we can implement more complex and flexible behavior for this agent in terms of energy savings and energy management.

Another way for the smart house model to be improved is to implement interactions with the smart grid and possibly other smart houses. This implies a different scale of the model, one or several households can be considered as participants of the local market and trade renewable energy between each other and the smart grid in order to reduce energy peak demands for the grid. Regardless of the scale, a P2P market model which described in this work can be applied and modified.

One of the possible modifications could be the implementation of blockchain transactions for the market model. As we mentioned earlier in section 3.1.1, the decentralized P2P market model

provides trading between buyers and seller in randomized order to avoid possible competitive advantages and results in individual price for each transaction. This corresponds to essential aspects of blockchain technology as blockchain serves as distributed digital registry, that can record individual encrypted transactions across P2P network[51]. The decentralized structure of blockchain allows market participants to conduct P2P transactions without centralized authority. For the local market with a high number of different elements as individual agents, such structure can contribute to efficiency and integrity of designed market approach.

Smart parking model can be improved with the implementation of different market designs for the parking slots distribution. Different services for the parking such as car washing can be introduced as additional elements for the model. In addition to original market design, the auction-based market approach can be considered for the cars in order to increase the number of different interactions between agents. The scale of the model also can be changed, as it could be represented as the problem of finding the optimal parking place for the car in terms of price and position. As the peak traffic hours can influence the number of cars which desire to find a parking place, the correlations between traffic levels and distribution of parking slots also can be considered in more detail.

For the smart traffic problem, a possible extension of the model can include representation of the roads as graphs with different segments, each of which can be monitored and controlled by the agent.

In our work, we considered discretized states for the learning, but in case of more complex and dynamic environment, discretization is not always possible. Implementation of reinforcement learning algorithm for the continuous and multidimensional space of states and actions to model agents behavior can make implemented solutions more suitable for real-world appliance and can be beneficial for handling problems with a large number of elements. Another important aspect is the reward evaluation, as it greatly impacts learning results. For this purpose actor-critic mechanism can be introduced and adapted for the specific solutions.

For the centralized approach with genetic-algorithm-based solution additional constraints and parameters, used for encoding and finding solutions can increase efficiency and accuracy of results. Evolutionary algorithms such as NSGA and SPEA can be used in order to acquire better results in case of more complex problems.

For the implementation of the solutions usage of frameworks such as JADE and osBrain which support communication between agents and multi-threading can provide more opportunities to create simulations, close to real word and demonstrate full capabilities of distributed multi-agent solutions. In case of osBrain, we can use different communication patterns and multi-threading to create simulations which involves complex calculations with a large number of variables and data.

Chapter 7

Conclusion

The objective of this thesis was to explore how multi-agent system which consists of different self-interested agents, can demonstrate a certain level of intelligence in terms of problem-solving capabilities, adaptability or learning proficiency and provide flexible solutions for the set of problems, related to Smart City context. Different aspects of Smart City and multi-agent systems were studied and described.

The emphasis was on studying how different agents can coexist and interact with each other in order to share common good. We managed to design and create a set of models for the considered problems and applied several approaches for each modeled problem. With these models we were able to analyze how through the interactions between agents, agent-based systems were able to manage different tasks. With introducing learning aspects we saw how different learning agents are able to adapt to changes in the environment and change their behavior. As a result, we were able to justify that even with simple behavior models of agents, multi-agent system can demonstrate more complex overall behavior and problem-solving capabilities. With defined set of constraints even zero or low-intelligent agents are able to find desired solutions and maintain stability of the system.

Through the thesis we ensured that decentralized control and decision-making mechanisms, provided by the multi-agent systems could be effective for solving the problems which include number of different parameters. With help of distributed market-base approach, agents in the system were able to coexist and demonstrate good results in terms of sharing limited resource. In case of centralized approach, agent, responsible for handling the problem had to have sufficient amount of information about the environment and clearly defined set of constraints in order to adapt to different situations.

On the assumption of study of Smart City concepts done and acquired results we can conclude that multi-agent systems indeed are able to demonstrate impressive results in finding the solutions for the problems, common for the Smart City. Using multi-agent systems we can design different behavior models for the agents and structures of multi-agent interactions, which allow us to adapt our solutions for the different context.

Bibliography

[1] M Roscia, Michela Longo, and George Cristian Lazaroiu. Smart city by multi-agent systems. *Proceedings of 2013 International Conference on Renewable Energy Research and Applications, ICRERA 2013*, pages 371–376, 10 2013.

[2] Michela Longo, M Roscia, and George Cristian Lazaroiu. Innovating multi-agent systems applied to smart city. *Research Journal of Applied Sciences, Engineering and Technology*, 7:4296–4302, 05 2014.

[3] Michal Lom and Ondej Pibyl. Modeling of smart city building blocks using multi-agent systems. *Neural Network World*, 27:317–331, 01 2017.

[4] Vangelis Angelakis, Elias Tragos, Henrich C. Phls, Adam Kapovits, and Alessandro Bassi. *Designing, Developing, and Facilitating Smart Cities*. Springer, Reading, MA., 2017.

[5] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645 – 1660, 2013. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2013.01.010. URL http://www.sciencedirect.com/science/article/pii/S0167739X13000241. Including Special sections: Cyber-enabled Distributed Computing for Ubiquitous Cloud and Network Services and Cloud Computing and Scientific Applications Big Data, Scalable Analytics, and Beyond.

[6] Julien Nigon, Nicolas Verstaevel, Jrmy Boes, Frdric Migeon, and Marie-Pierre Gleizes. Smart is a matter of context. pages 189–202, 06 2017.

[7] Eleni Vlahogianni, Konstantinos Kepaptsoglou, Vassileios Tsetsos, and Matthew Karlaftis. A real-time parking prediction system for smart cities. *Journal of Intelligent Transportation Systems*, 20:00–00, 04 2015.

[8] J. Jin, J. Gubbi, S. Marusic, and M. Palaniswami. An information framework for creating a smart city through internet of things. *IEEE Internet of Things Journal*, 1(2):112–121, April 2014. ISSN 2327-4662. doi: 10.1109/JIOT.2013.2296516.

[9] M. C. Choy, D. Srinivasan, and R. L. Cheu. Neural networks for continuous online learning and control. *IEEE Transactions on Neural Networks*, 17(6):1511–1531, Nov 2006. ISSN 1045-9227. doi: 10.1109/TNN.2006.881710.

[10] Paul Davidsson and Magnus Boman. Distributed monitoring and control of office buildings by embedded agents. *Information Sciences*, 171(4):293 – 307, 2005. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2004.09.007. URL http://www.sciencedirect.com/science/article/pii/S0020025504003111. Intelligent Embedded Agents.

[11] W. Li, T. Logenthiran, and W. L. Woo. Intelligent multi-agent system for smart home energy management. *2015 IEEE Innovative Smart Grid Technologies - Asia (ISGT ASIA)*, pages 1–6, Nov 2015. doi: 10.1109/ISGT-Asia.2015.7386985.

[12] Erwin Walraven, Matthijs T.J. Spaan, and Bram Bakker. Traffic flow optimization: A reinforcement learning approach. *Engineering Applications of Artificial Intelligence*, 52: 203 – 212, 2016. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2016.01.001. URL http://www.sciencedirect.com/science/article/pii/S0952197616000038.

[13] G. H. Merabet, M. Essaaidi, H. Talei, M. R. Abid, N. Khalil, M. Madkour, and D. Benhaddou. Applications of multi-agent systems in smart grids: A survey. *2014 International Conference on Multimedia Computing and Systems (ICMCS)*, pages 1088–1094, April 2014. doi: 10.1109/ICMCS.2014.6911384.

[14] Mihaela Oprea. Applications of multi-agent systems. page 32, 2004.

[15] Robin trading hub official page. http://robintradinghub.com/, 2018. Accessed: 2018-05-20.

[16] Trista Lin, Herve Rivano, and Frdric Le Moul. A survey of smart parking solutions. *IEEE Transactions on Intelligent Transportation Systems*, 18:3229 – 3253, 04 2017.

[17] J. Kozlak and M. Zabinska. Agent-based simulation of road traffic using market approach. *2017 International Conference on Behavioral, Economic, Socio-cultural Computing (BESC)*, pages 1–6, Oct 2017. doi: 10.1109/BESC.2017.8256361.

[18] B. Feron and A. Monti. Development and assessment of a market-based multi-agent system for domestic heat and electricity management. *2016 IEEE International Energy Conference (ENERGYCON)*, pages 1–6, April 2016. doi: 10.1109/ENERGYCON.2016.7514114.

[19] Esther Mengelkamp, Philipp Staudt, Johannes Grttner, and Christof Weinhardt. Trading on local energy markets: A comparison of market designs and bidding strategies. page 6, 06 2017.

[20] Dhananjay K. Gode and Shyam Sunder. Allocative Efficiency of Markets with Zero-Intelligence Traders: Market as a Partial Substitute for Individual Rationality. *Journal of Political Economy*, 101(1):119–137, March 1993. URL https://www.scribd.com/document/269718338.

[21] Zero Intelligence Traders: Gode and Sunder (1993). http://people.brandeis.edu/~blebaron/classes/agentfin/GodeSunder.html, 2018. Accessed: 2018-05-11.

[22] Bogdan Tomoiaga, Mircea Chindri, Andreas Sumper, Antoni Sudria-Andreu, and Roberto Villafafila-Robles. Pareto optimal reconfiguration of power distribution systems using a genetic algorithm based on nsga-ii. *Energies*, 6(3):1439–1455, 2013. ISSN 1996-1073. doi: 10.3390/en6031439. URL http://www.mdpi.com/1996-1073/6/3/1439.

[23] Myeong Jin Ko, Yong Shik Kim, Min Chung, and Hung Chan Jeon. Multi-objective optimization design for a hybrid energy system using the genetic algorithm. *Energies*, 8: 2924–2949, 04 2015.

[24] Nadeem Javaid and Urva Latif. Cost optimization in home energy management system using genetic algorithm, bat algorithm and hybrid bat genetic algorithm. pages 5–6, 02 2018.

[25] Florin Leon, Marcin Paprzycki, and Maria Ganzha. A review of agent platforms. pages 2–16, 11 2015.

[26] Microsoft orleans official page. https://dotnet.github.io/orleans/index.html, 2017. Accessed: 2018-05-19.

[27] Java agent development framework. http://jade.tilab.com/, 2018. Accessed: 2018-05-20.

[28] The foundation for intelligent physical agents. http://www.fipa.org/, 2018. Accessed: 2018-05-20.

[29] Mesa documentation page. http://mesa.readthedocs.io/en/latest/overview.html, 2018. Accessed: 2018-05-20.

[30] osBrain documentation page. http://osbrain.readthedocs.io/en/stable/index.html, 2018. Accessed: 2018-05-20.

[31] Wolf-sheep predation model. https://github.com/projectmesa/mesa-examples/tree/master/examples/WolfSheep, 2018. Accessed: 2018-05-31.

[32] Uni Heidelberg. Lecture 5 reinforcement learning. https://www.uni-heidelberg.de/md/awi/forschung/reinforcement_learning.pdf, 2018. Accessed: 2018-05-16.

[33] Mridul Pentapalli. A comparative study of roth-erev and modified roth-erevreinforcement learning algorithms for uniform-price double auctions. pages 6–11, March 2008. URL http://www2.econ.iastate.edu/tesfatsi/MRidulPentapalli_ThesisPresentation.FinalRevs.pdf.

[34] Yoav Shoham and Kevin Leyton-Brown. *MULTIAGENT SYSTEMS*. Cambridge University Press, 2009.

[35] A genetic algorithm in python. https://colindrake.me/post/a-genetic-algorithm-in-python/, 2018. Accessed: 2018-05-31.

[36] W. Li, T. Logenthiran, and W. L. Woo. Intelligent multi-agent system for smart home energy management. *2015 IEEE Innovative Smart Grid Technologies - Asia (ISGT ASIA)*, pages 1–6, Nov 2015. doi: 10.1109/ISGT-Asia.2015.7386985.

[37] Specific heat. http://hyperphysics.phy-astr.gsu.edu/hbase/thermo/spht.html. Accessed: 2018-05-11.

[38] O. Irulegi, A. Serra, and R. Hernndez. Data on records of indoor temperature and relative humidity in a university building. *Data in Brief*, 13:248 – 252, 2017. ISSN 2352-3409. doi: https://doi.org/10.1016/j.dib.2017.05.029. URL http://www.sciencedirect.com/science/article/pii/S2352340917302202.

[39] RapidTables. Converting lux to watts. https://www.rapidtables.com/calc/light/how-lux-to-watt.html, 2018. Accessed: 2018-05-11.

[40] Recommended light levels (illuminance) for outdoor and indoor venues. https://www.noao.edu/education/QLTkit/ACTIVITY_Documents/Safety/LightLevels_outdoor+indoor.pdf, 2018. Accessed: 2018-05-11.

[41] RadiantFloorHeating. Heated floor running costs. http://www.radiantfloorheating.com.au/shop/view/running_costs/5, 2018. Accessed: 2018-05-11.

[42] SINO VOLTAICS. Battery discharge: solar battery bank discharge explained. http://sinovoltaics.com/learning-center/storage/battery-discharge-solar-battery-bank/, 2018. Accessed: 2018-05-11.

[43] SOTAVENTO. Wind generation data. http://www.sotaventogalicia.com/en/real-time-data/historical, 2018. Accessed: 2018-05-11.

[44] Nordpool. Historical market data. https://www.nordpoolgroup.com/historical-market-data/, 2018. Accessed: 2018-05-14.

[45] Dana Nau. Cmsc 421, intro to ai - spring 2010, section 17.6 - game theory. https://www.cs.umd.edu/~nau/cmsc421/game-theory.pdf, 2018. Accessed: 2018-05-15.

[46] Traffic congestion statistics for Oslo based on TomTom's historical database for 2016. https://www.tomtom.com/en_gb/trafficindex/city/oslo, 2018. Accessed: 2018-05-11.

[47] Aud Tenny, Paal B. Wangsness, Jrgen Aarhaug, and Fredrik Alexander Gregersen. Experiences with capacity reductions on urban main roads rethinking allocation of urban road capacity. *International Scientific Conference on Mobility and Transport Transforming Urban Mobility*, pages 8–10, June 2016. URL https://www.researchgate.net/publication/316080419.

[48] Congestion charge and environmental differentiation. https://www.fjellinjen.no/private/prices/, 2017. Accessed: 2018-05-11.

[49] F. Ho and P. Ioannou. Traffic flow modeling and control using artificial neural networks. *Control Systems,IEEE*, pages 16:16–26, 1996.

[50] Michael Wooldridge. *An Introduction to MultiAgent Systems*. JOHN WILEY & SONS, LTD, 2002.

[51] Nikita Shvetsov. Exploring auction based energy trade with the support of mas and blockchain technology. pages 17–21, 06 2017.

Appendix A

Smart house model additional figures

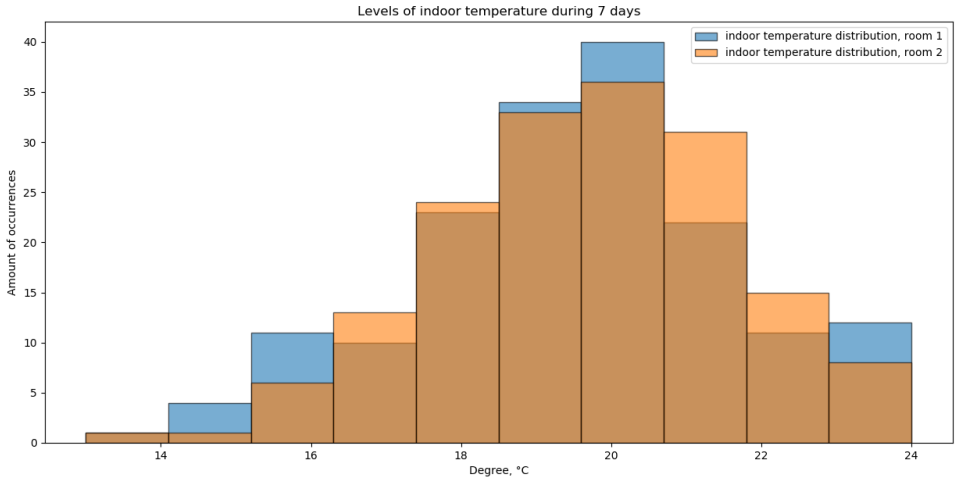


FIGURE A.1: Indoor temperature levels during 7 days.

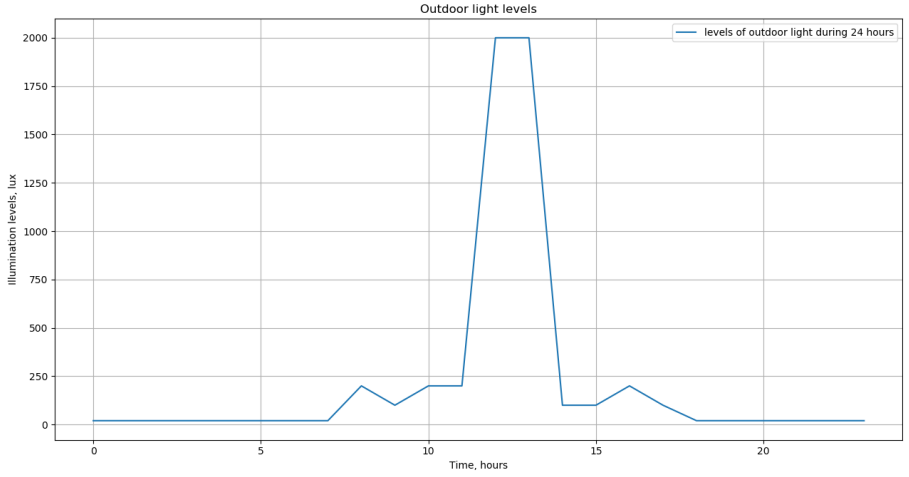


FIGURE A.2: Outdoor light levels during 24 hours.

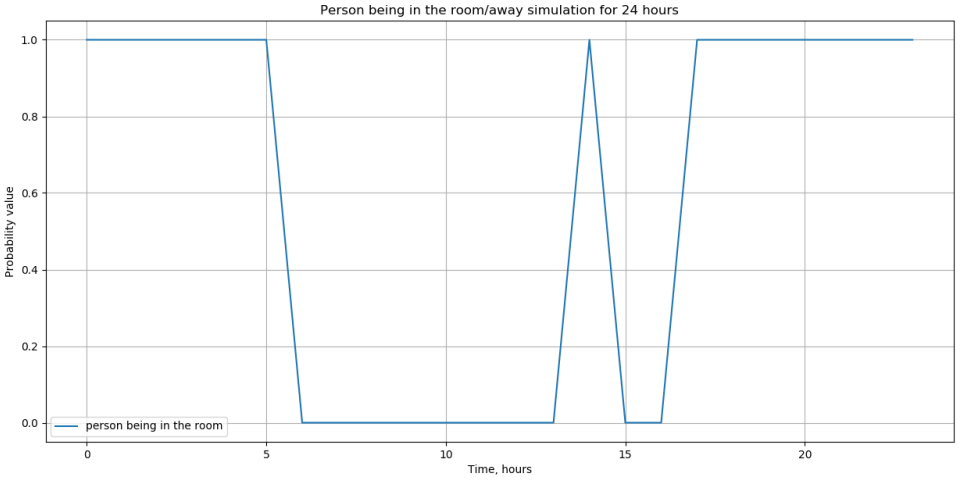


FIGURE A.3: Person being in the room/being away probability graph.

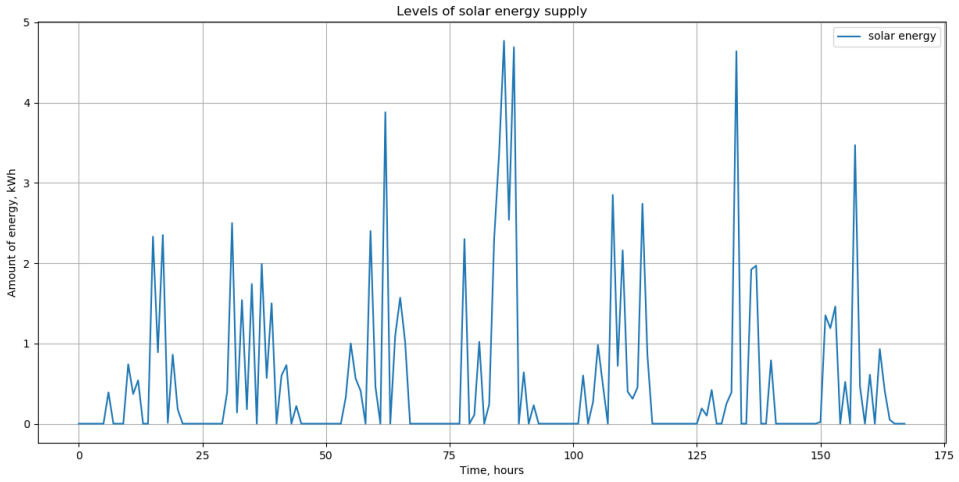


FIGURE A.4: Levels of solar energy during 7 days.

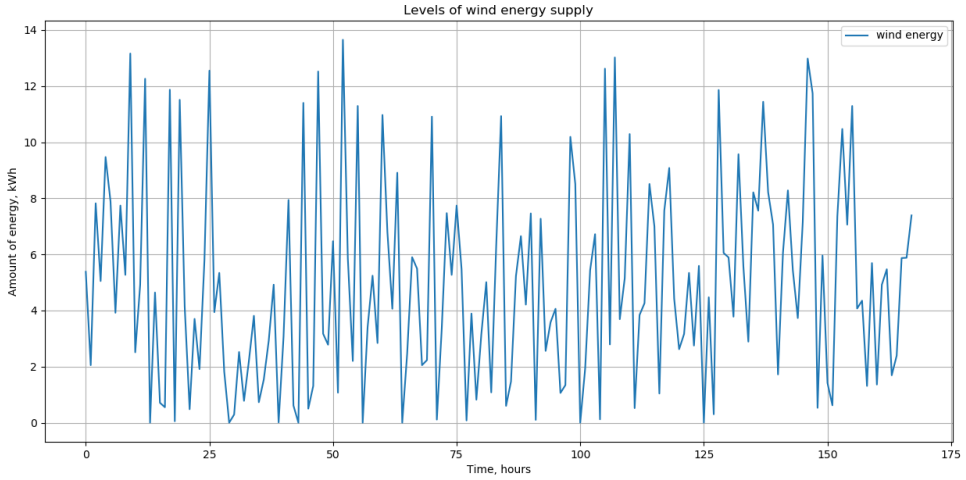


FIGURE A.5: Levels of wind energy during 7 days.

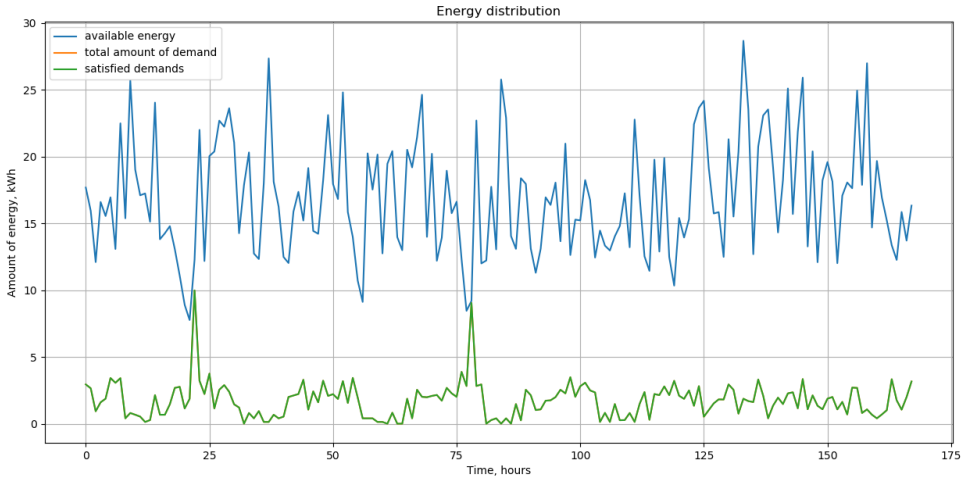


FIGURE A.6: Energy distribution with battery and all renewable energy sources during 7 days with ZI approach.

Appendix B

Smart parking model additional figures

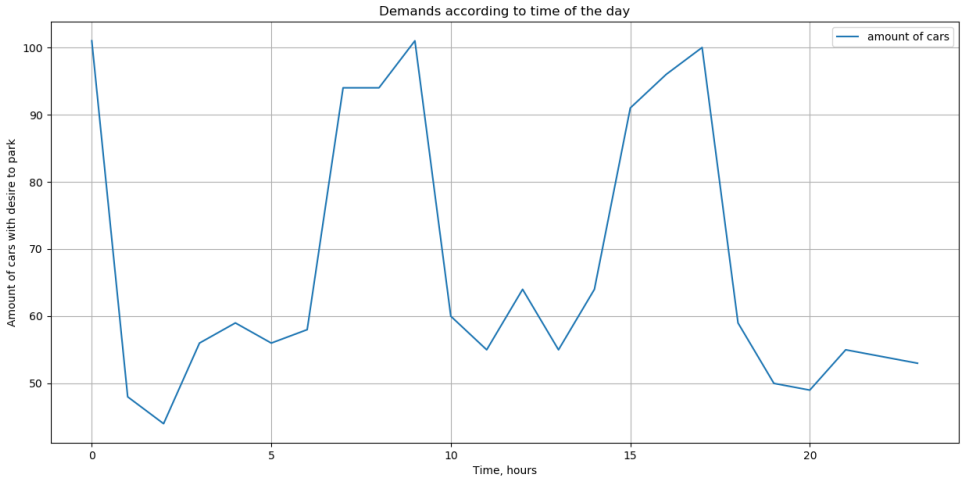


FIGURE B.1: Demands for the parking during 24 hours.

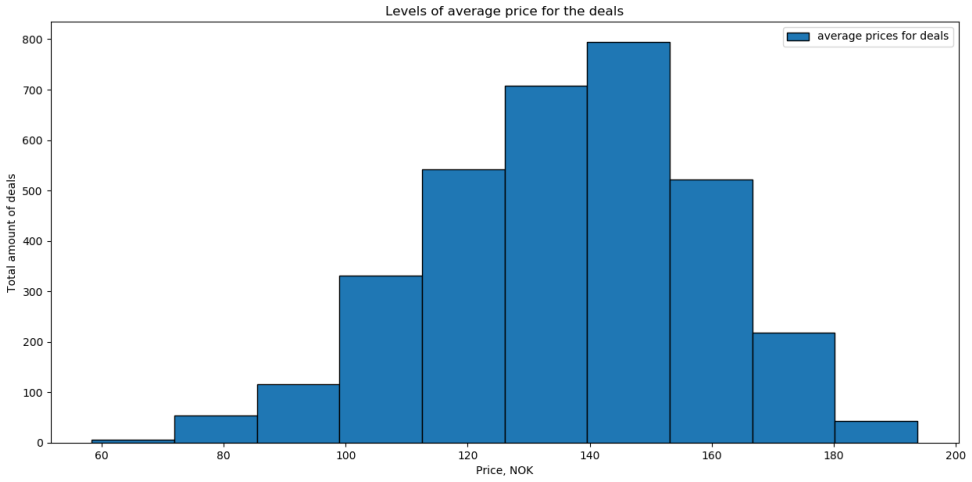


FIGURE B.2: Levels of average price for the deals during 7 days.

Appendix C

Source code

The complete source code and requirements.txt file for the installation may be found at the following links:

Smart house

- ZI smart house model: https://github.com/BlindBird993/ZiSmartHouseModel
- smart house model with learning: https://github.com/BlindBird993/LSmartHouseModel
- smart house centralized model: https://github.com/BlindBird993/CSmartHouseModel

Smart parking

- ZI smart parking model: https://github.com/BlindBird993/ZiParkingModel
- smart parking model with learning: https://github.com/BlindBird993/LParkingModel
- smart parking centralized model: https://github.com/BlindBird993/CParkingModel

Smart traffic

- ZI smart traffic model: https://github.com/BlindBird993/ZiTrafficModel
- smart traffic model with learning: https://github.com/BlindBird993/LTrafficModel
- smart traffic centralized model: https://github.com/BlindBird993/CTrafficModel

Appendix D

Project description

Faculty of Engineering Science and Technology
Department of Computer Science and Computational Engineering
UiT The Arctic University of Norway

Distributed Management of Resources in a Smart City using Multi-Agent Systems (MAS)

Igor Anatolyevich Molchanov
Thesis for Master of Science in Computer Science

Problem description

Smart City represents a new way of thinking about urban space by shaping a model that integrates green energy resources and systems, energy efficiency, sustainable mobility, smart and sustainable living for all as well as maintainable economy for the future. It is a new paradigm that resides on modern Information and Communication Technology (ICT), not least the accelerating emergence of what is called Internet of Things (IoT) that enables and depends on distributed and ubiquitous data processing and information in a connected form. Much of the development will take place bottom-up. This means that vendors of sensors, controllers and processors may not provide offer according to specifications and standards provided by a centralized body. Instead, initiatives will be driven by needs defined for different niches using different protocols and different base technologies. Multi-Agent Systems (MAS) cater for a non-centralized development of intelligent Smart City applications that can be developed in independently and in parallel, and still be able to cooperate. An inherent capability of MAS is that even with low or zero-intelligence agents the collective performance can demonstrate intelligence and learning capabilities that make MAS adaptable to change, extensions and reductions, and still be able to create optimal or close to optimal solutions.
The candidate is going to explore the use of MAS in the context of IoT and Smart City. The basic question to be answered is how a distributed set of disparate set of sensors and controllers that operate a variety of facilities such as heaters, charging spots, electric vehicles etc. can be connected in a "bottom up" fashion and together and demonstrate consolidated intelligence. Intelligence can be defined in terms of collective problem-solving capabilities, optimization capabilities or adaptability/learning proficiency. In distributed systems, a market oriented approach is often applied. A fundamental issue is related to how agents that manage sensors and are tuned to fulfill certain goals and desires can coexist and interact with agents of parallel as well as opposing needs. This is an issue of self-interest versus cooperation.
The thesis work will be organized around two sub-goals/tasks:

1. Based on work by Roscia et al[1]., Lono et al.[2] as well as Lom and Pribyl[3] and related literature the candidate is going to explore the concept of MAS for the benefit of Smart City and design a system architecture that can manage communication/interaction between different agents managing different sensors and controllers based on different hardware and that apply different protocols. Exchange of requests and responses,

---

1 Rosica, Longo, Lazaroiu: Smart City By Multi-Agent Systems, Int. conference on Renewable Energy Research and Applications, Madrid Oct 2013
2 Longo, Roscia ansd Lazaroiu: Innovating Multi-Agent Systems Applied to Smart City, Research Journal of Applied Sciences, Engineering and Technology 7(20), 4296-4302, 2014
3 M.Lom, O.Pribyl: Modeling of Smart City Building Blocks using multi-agent systems, CTU FTS 2017

management of conflicts of interest as well as remuneration concepts should be addressed.

2. Using the principles defined under sub-goal 1 the candidate should create a computer model/limited system whereby a set of agents of both confluence and opposing interests can find ways to share a common good i.e. a road or parking space (in the case of electric vehicles) and distribute a limited resource such as renewable energy produced by one or more facilities. For the benefit of his effort he may use standard multi-agent platforms written in Java or Python i.e. ZEUS, Java. The model should be demonstrated and documented according to UiT standards for MSc theses.

Dates

Date of distributing the task: <12.01.2018>

Date for submission (deadline): <01.06.2018>

Contact information

Candidate — Igor Anatolyevich Molchanov
imo031@post.uit.no

Advisor at UiT-IVT — Bernt A. Bremdal
bernt.a.bremdal@uit.no

Advisor at UiT-IVT — Kristoffer Tangrand
kristoffer.tangrand@uit.no

General information

This master thesis should include:

- ❄ Preliminary work/literature study related to actual topic
  - A state-of-the-art investigation
  - An analysis of requirement specifications, definitions, design requirements, given standards or norms, guidelines and practical experience etc.
  - Description concerning limitations and size of the task/project
  - Estimated time schedule for the project/ thesis
- ❄ Selection & investigation of actual materials
- ❄ Development (creating a model or model concept)
- ❄ Experimental work (planned in the preliminary work/literature study part)
- ❄ Suggestion for future work/development

Preliminary work/literature study

After the task description has been distributed to the candidate a preliminary study should be completed within 3 weeks. It should include bullet points 1 and 2 in "The work shall include", and a plan of the progress. The preliminary study may be submitted as a separate report or "natural" incorporated in the main thesis report. A plan of progress and a deviation report (gap report) can be added as an appendix to the thesis.

In any case the preliminary study report/part must be accepted by the supervisor before the student can continue with the rest of the master thesis. In the evaluation of this thesis, emphasis will be placed on the thorough documentation of the work performed.

Reporting requirements

The thesis should be submitted as a research report and could include the following parts; Abstract, Introduction, Material & Methods, Results & Discussion, Conclusions, Acknowledgements, Bibliography, References and Appendices. Choices should be well documented with evidence, references, or logical arguments.

The candidate should in this thesis strive to make the report survey-able, testable, accessible, well written, and documented.

Materials which are developed during the project (thesis) such as software / source code or physical equipment are considered to be a part of this paper (thesis). Documentation for correct use of such information should be added, as far as possible, to this paper (thesis).

The text for this task should be added as an appendix to the report (thesis).

General project requirements

If the tasks or the problems are performed in close cooperation with an external company, the candidate should follow the guidelines or other directives given by the management of the company.

The candidate does not have the authority to enter or access external companies' information system, production equipment or likewise. If such should be necessary for solving the task in a satisfactory way a detailed permission should be given by the management in the company before any action are made.

Any travel cost, printing and phone cost must be covered by the candidate themselves, if and only if, this is not covered by an agreement between the candidate and the management in the enterprises.

If the candidate enters some unexpected problems or challenges during the work with the tasks and these will cause changes to the work plan, it should be addressed to the supervisor at the UiT or the person which is responsible, without any delay in time.

Submission requirements

This thesis should result in a final report with an electronic copy of the report including appendices and necessary software, source code,

simulations and calculations. The final report with its appendices will be the basis for the evaluation and grading of the thesis. The report with all materials should be delivered according to the current faculty regulation. If there is an external company that needs a copy of the thesis, the candidate must arrange this. A standard front page, which can be found on the UiT internet site, should be used. Otherwise, refer to the “General guidelines for thesis” and the subject description for master thesis.

The advisor(s) should receive a copy of the the thesis prior to submission of the final report. The final report with its appendices should be submitted no later than the decided final date.