research work

Adaptive Building Intelligence

An approach to adaptive discovery of functional structure

Jonas Trindler
<trindler@ini.phys.ethz.ch>

Raphael Zwiker
<rzwiker@ini.phys.ethz.ch>

Advisors
Prof. Dr. Rodney Douglas, Institute of Neuroinformatics, ETH/University Zurich
Prof. Dr. Josef Joller, University of Applied Sciences Rapperswil

A cooperation between

Computer Science Department
University of Applied Science Rapperswil
Oberseestrasse 10
8640 Rapperswil, Switzerland

Institute of Neuroinformatics
University and ETH Zurich
Winterthurstrasse 190
8057 Zurich, Switzerland

September 16, 2003

Typeset by LaTeX

Abstract

Modern approaches to the architecture of living and working environments emphasize the simple reconfiguration of space to meet the needs, comfort and preferences of its inhabitants and to minimize the consumption of resources such as power. The configuration can be explicitly specified by a human building manager, but there is now increasing interest in the development of intelligent buildings equipped with standard sensors (e.g. presence, temperature, illumination, humidity) and effectors (e.g. lights, window blinds, wall-switches) that adapt to the needs of its inhabitants without human intervention.

There are several approaches to control such an intelligent building by learning the behavior of each room and take decisions based on experience. This approaches however all use a static structure where each sensor and effector is assigned statically to a room instead of considering the dynamic structure of the building.

We describe and demonstrate an algorithm that dynamically discovers and hierarchically clusters functionally related sensors and effectors. Based on the temporal occurrence of events, generated by sensors and effectors, we build up a weighted directed graph. These weights are event-dependent and are constantly adapted with a hebbian-like learning algorithm. Our approach is completly unsupervised and learns the relations passive.

In a second step the graph is partitioned using a normalized cut [**?**]. The partitions of the graph are small clusters with logically related devices.

The algorithms are implemented and tested using real data from a mixed intelligent / standard business environment. We illustrate that the discovered clusters can directly correspond to the office and suboffice structure of the building.

This approach to adaptive discovery of functional structure is part of a multi-agent-based intelligent building system project called *Adaptive Building Intelligence* [**?**].

Further documentation, the API reference documentation and the sourcecode can be found at:
`http://www.ini.unizh.ch/~trindler/abi/`.

Contents

Preface

by Prof. Dr. Joseph M. Joller, University of Applied Sciences Rapperswil, Switzerland

We are working towards a better understanding of the concept "Buildings as Multi-Agent systems". The previous work by U. Rutishauser and A. Schaefer tried to build a whole system from scratch in a very short period of time. Several concepts (hardware abstraction, clustering, learning) were implemented in a prototype system.

The overall architecture, that seems to emerge and to become more and more visible, consists of at least the following layers:

- Application Layer
  interfaces for several add-on devices, including specialized agents (location, remote control,)
- Core Agent Layer
  ABLE based agents, with learning algorithm(s) and rule bases and mechanisms for policy transfers.
- Agent Communication Layer
  this layer is more or less covered by JAS
- Hardware Abstraction Layer
  including structure agents

This term project had clear targets:

- to understand the spontaneous structuring of sensors and to study data clustering techniques in general.
- to compare controlled versus uncontrolled rooms. The uncontrolled rooms are of special interest, as we didn't have time yet in the other projects to systematically study their patterns. The results are fundamental and absolutely needed, before we can even think about controlling the room by agent technology. Only if we know, how the system behaves, we can start thinking about a model of the system, which we will implement using agent technology.

The results reported in this term project should, together with the raw data collected, provide a solid base on which we can now study in more details the Agent part of the system, including the learning algorithm, which probably has to be replaced completely.

The term project is an important step from a pure "try" project level to a "productive" level. It must be our goal to come up with

- a clear architecture
- solid software solutions, backed by field data
- well understood structures, rules and agent communities.

I look forward to the diploma thesis, where we will have to move upwards in the above layer architecture.

Part I

Introduction

Chapter 1

Introduction

1.1 Overview

In this paper we describe our approach to discover the functional structure of a typical commerical building. We define a learning algorithm, which is related to hebbian learning, to determine the relationship between all sensors and effectors in a building. This measure of similarity is used in a second step to group them into several independent clusters whereas these represent the functional structure. Our approach is event-driven and computes the new structure online.

Before we explain our approach, we illustrate the needs for discovering structure in a dynamic way. To make a typical commercial building really intelligent one has to understand the dynamic behavior of itself and its inhabitants. Previous research work of U.Rutishauser and A. Schaefer [**?**] focused on adaptive control and learning of such a building. It is necessary to introduce their work to understand what are the open issues and needs which have to be analysed and satisfied before we can illustrate the objectives of our work.

1.2 Adaptive Building Intelligence (ABI)

Other projects [**?**, **?**] have shown that it is possible to use results of interdisciplinary research fields (neuroscience, artificial intelligence, computer science and electrical engineering) to build intelligent rooms and buildings which are eventually also controlled automatically. The goal of our predecessor, U. Rutishauser and A. Schaefer [**?**], was to fix the lack which the others had and this is *learning* a rulebase to be able to control a building.

A building is a very complex system, since it behaves from a computational point of view completely non-deterministic. [**?**] says that such an environment is best controlled by a multi-agent-system and calls his approach *Distributed Artificial Intelligence (DAI)*. Our predecessor tackled this challenge by using a layered multi-agent-system. Different agents act independently and communicate indirectly with others about their goals and actions they take. Every agent has its own field of responsibility, but to achieve the overall goal of controlling a building they have to collaborate.

A *control agent* has a rulebase for controlling a local part of the building. These local areas are defined as clusters which contain sensors and effectors. These clusters can match with the physical structure of the building. The *control agent* receives sensory input and controls effectors by consulting a local rulebase which is used for reasoning. It starts with only a few basic rules, but learns adaptively the behavior of the users and environment by modifying its rulebase. These rules become more detailed and sophisticated whenever the learning unit receives feedback from the enviornment (punishment/reward).

In a typical office building all sensors and effectors are attached to a fieldbus (like [**?**, **?**]) and thier state can be requested or set by sending a specific command via this fieldbus. To reach each device independently,

every device has its own unique address (e.g. the network variable in LonWorks). Our building is equipped with the LonWorks system and therefore we pay no further attention to other products. But an integration of other systems would be possible.

The *bus agent* acts like a proxy between the multi-agent-system and the dedicated fieldbus. To be more exactly, it is strong coupled with a Lon-Network-Server (LNS). This server communicates over IP with the iLon gateway. iLon is the real LonWorks-IP gateway and so the only device which requires direct access to the LonWorks fieldbus.

A *structure agent* offers information about the structure of a building which each other agent who is interested has to request. It reads the static structure once and has no abilities to notify any changes of the structure to other agents. These changes could just be a new device or rather new reorganisation of a large office, which is not unlikely in a commercial building.

The aim of our project starts here by detecting the dynamics of a building which can have strong impacts on several other issues. Even the whole learning has to deal with a kind of *moving target* learning, it is essential to understand and determine the dynamic behavoir of our intelligent building. More reasons why we think it is worth to analyse this and discover the structure are mentioned in our motivation (see chapter 2).

For more details about the ABI system itself we refer to chapter 6 on page 24 of this document or directly to [**?**].

1.3 Content

This document is structured into the seven parts Introduction (part I), Approach to the problem of structure (part II), Architecture (part III), Design and Implementation (part IV), Results and discussion (part V), Conclusion and Future Work (part VI) and Glossary and Bibliography (part **??**).

The first part, Introduction (part I), gives a general introduction about the aim of our project. We introduce first in the chapter Introduction (1) the current state of our *Adaptive Building Intelligence* project and illustrates where open issues are and our work starts. We give also a list of papers which report about related work. In the chapter Motivation (2) we say why it is worth doing our work, what our objectives are and what results we expect.

The Approach to the problem of structure reviews first different possibilities to determine the relational structure of a building and group them in independent clusters (chapter 3). We explain why we defined a learning algorithm to discover the relationship and why to choose the *normalized cut* algorithm for clustering. In theory (4) we ground our inspiration for defining a *hebbian* related learning algorithm and give a brief summary of the *normalized cut*. And the chapter 5 gives an extensive explanation of our approach to the problem of structure.

Architecture, which consists of the chapters system architecture (6) and software architecture (7) gives a detailed introduction to the underlying architectural principles used within ABI. We illustrate also where our new agent, *Structure Discovery Agent*, is located in the layered multi-agent-system.

Details about the implementation of our agent are given in part IV. First we justify design decisions in chapter 8 and explain in chapter 9 why it is necessary to proceed the input data through filters to get ride of useless data caused by damaged devices. Because we have to deal with really sparse data, a simulation of the bus traffic is indispensable. The last chapter of this part, Simulation (10), explains its functions.

Chapter 11 evaluates the results of our approach. We discuss results and open issues in detail.

Part VI illustrates in the Conclusion (12) what has been achieved and what not within the context of the project. Future Work (13) suggests what could be done in the future and lists potential topics for future research projects to improve the *Adaptive Building Intelligence*.

1.4 Related work

The basic ideas and principles of *Adaptive Building Intelligence (ABI)* were first developed by our predecessors during the project *ADA - An artificial organism* [**?**] and *Adaptive Home Automation (AHA)* [**?**] which were conducted at the Institute of Neuroinformatics in Zurich, Switzerland.

Ada is a project which regards a room as an artifical organism. She has like every organism an emotional state and expresses herself and interacts with visitors. The goal of Ada is to dynamically change its overall functionality and quality through an active dialog with visitors.

The other project, *Adaptive Home Automation (AHA)*, was mainly an approach to control a building with a static fuzzy logic rulebase driven by a multi-agent system. ABI is build on top of the preceding AHA project and *learns* online to adapt the rulebase while running.

A variety of other related projects [**?**, **?**, **?**] try to make a building intelligent whereas the main objectives are mostly the same: minimize the consumption of power and/or on the other hand to maximize the comfort for its inhabitants.

Most of these project concentrate on making a room/building intelligent by learning to control it. But all are learning this on top of a *static* structure. They lack the capability of detecting the dynamics of a building's structure, which is really essential to control several rooms/floors together, even since the environment is completely inhomogeneous: people have different behaviors and can work on several places, large offices can be reorganized frequently or new sensors/effectors get installed and have be controlled. Up to now we haven't found any related work which tackles this challenge and discovers the functional structure of such an environment.

There are papers in the field of computer science, which are trying to cluster the architecture of the world-wide-web and its documents. E.g. [**?**] regards web links as association like in the brain, and the strength of the links can change depending on the frequency of use. They defined also a hebbian inspired algorithm which adapts the strength of links. Based on this they show how the web can be structured.

Algorithms for detecting the structure of a network (e.g. Ethernet) are also interesting to review. But they differ mainly in reverting to the amount of data and that each message carries a source- and destinationaddress.

Chapter 2

Motivation

Within this chapter we motivate why it is worth trying to discover the functional structure of a multisensor environment and mention what we expect to gain in terms of results.

A multisensor environment, like a commercial building, is itself a complex environment and thus very interesting and challenging to analyze and understand. It contains information about its state, but also records the dynamic behavior of such an environment. The difficult task, however, is to understand this information and cohesions.

We use a typical commercial building as our test environment equipped with many different sensors (e.g. presence, temperature, illumination, humidity) and effectors (e.g. lights, window blinds, wall-switches). We think such a building behaves not static like it was built, but rather totally dynamic and responds to its inhabitants and the current weather conditions by changing its internal state locally.

This itself is interesting enough to look into in more detail. But also by considering our preceding project, *Adaptive Building Intelligence (ABI)*, and be able to review their learning algorithm it would be very appropriate to have a feeling about the dynamics of a building. Even since the learning is a kind of *moving-target learning* and has to learn against a changing internal representation, it is indispensable to have a detailed analysis and understand possible impacts.

Modern buildings which are equipped with a dedicated fieldbus need nowadays a human building manager to explicitly specify which sensors can control which effectors. It would be thinkable to have a self-organized environment which adapts and customize itself.

We want to prove that it is possible to gain functionally structured knowledge out of such an environment. And furthermore we want to extend the current *Adaptive Building Intelligence (ABI)* system by an autonomous agent, which discovers the functional structure online and deploys this information into the whole multi-agent-system.

The learned structure has to be analyzed and compared with different models, like for example the physical structure. We want to answer if it is possible to discover a useful structure at all. If it is, we would like to analyze how many functional more or less independent groups can be detected and how does it perform considering stability and placity?

We have to handle here real data caused by real persons produced while doing real work. This makes it even more interesting but also challenging, not just to verify our approach in a more or less natural environment. Results obtained from research in such a real environment are likely to be much more substantial then results from a specially designed simulated environment.

These are surely enough reasons to justify a research project into that!

Part II

Approach to the problem of structure

Chapter 3

Determine functional relationship between devices and partition them into clusters

3.1 Overview

The main objective of our project is to group devices of a commercial building into clusters. Each cluster should contain just functionally related devices, but there are several thinkable possibilities to measure or define their relationship. Elements of a cluster have now a special similar feature like for example shape, type, function, metric distance, orientation etc. One of the simplest way would be to group them depending on their typ (blinds, lights, daylight, switch...).

But we are interested in the functional structure of a building and therefor like to group them into functional more or less independent clusters. For this we have to find/define a measurement of the functional relationship. Section 3.2 lists three different possibilities to determine this. We explain briefly their functions and give reasons for choosing one or not.

Although we can measure the similarity we need another step to group them into several clusters. Section 3.3 surveys three thinkable clustering algorithms and lists thier dis-/advantages. We reason also, which are the criterions to choose one and not the others.

3.2 Determine similartiy between devices

3.2.1 Correlation coefficient

The correlation coefficient (corrcoef) is a normalized measurement of the linear relationship between the states of two devices. Uncorrelated data results in a correlation coefficient of 0.0, equivalent data sets have a correlation coefficient of 1.0 and anti-correlated one of -1.0. The corrcoef can be graphical interpreted as the $\cos(\alpha)$, if α is the angle between the two data vectors (see figure 3.1).

For defining the corrcoef between two devices it is necessary to represent states over time as two vectors ($\overrightarrow{A}$ and $\overrightarrow{B}$). Thats possible when the n-th dimension of $\overrightarrow{A}$ corresponds with state at the n-th minute. The higher the resolution the more exact is the correlation.

Disadvantages:

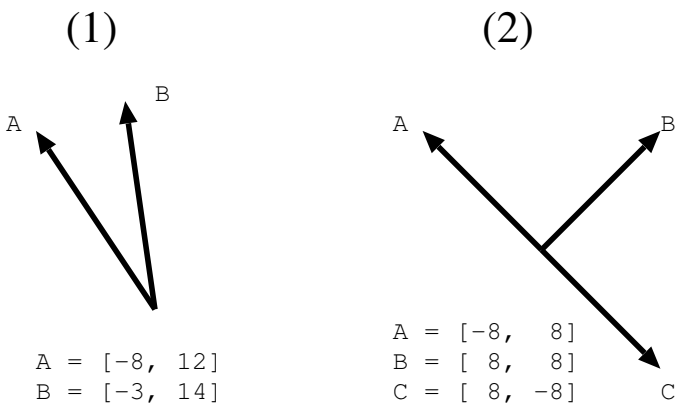

Figure 3.1: (1) shows high correlated vectors in two dimensional space. (2) Vector $\overrightarrow{A}$ ist anti-correlated to vector $\overrightarrow{C}$ and has no correlation to vector $\overrightarrow{B}$.

- Huge memory requirements because it is necessary to keep all states over the measure period in memory.
- Devices of a building show an different behavior at different timepoints. For example lights and blinds: During a sunny winter day the blinds are open and the lights are switched off. At the evening (if nobody is in the office) the blinds are closed (save energy) and the lights are also turned off. This two cases are mutually contradictory. Thus the corrcoeff is nearly zero.

3.2.2 Temporal dependent activity

The temporal connectivity methode detects the timedelay between two events of different devices. This timedelay represents the correlation. The shorter the timedelay the higher the correlation. Correlations in a graph can be represented in a directed, weighted graph, where each node is a device and edge weighted edge a correlation. The adjustment of the edges is event-driven.

Advantages:

- All calculations are event driven.
- The states of devices has no influence on the correlation. Thus there are also no problems with different behaviors on different timepoints.

Disadvantages:

- The resulting graph is a directed graph. That means that between two nodes A und B two correlations are defined ($A \Rightarrow B$ and $B \Rightarrow A$). But most clustering algorithm require a directed graph.
- A maximum border line for correlations does not exist. They can increase to infinity.
- The algorithm detects only behaviors which happen in a small timewindow, because if this would be to large every edge would be adjusted.

3.2.3 Genetic algorithm

Genetic algorithms seek after persistent patterns in the event stream. Depending on the complexity of patterns the algorithm requires huge amounts of memory and CPU time.

Advantages:

- Determined patterns represent the behavior of the users. The patterns could be used as an input for a further learning algorithm.

Disadvantages:

- The timedelay between two events isn't considered. Thus it is possible that a pattern ABC on a later date is exacly repeated (temporal viewed), but between the event sequence ABC other events can occur (e.g. ArBtC).
- It is a difficult task to determine the structure based on the founded patterns.
- For reliable predictions we would use a huge amount of data.

3.3 Clustering / partitioning

3.3.1 Minimal spanning tree

The basis of the minimal spanning tree (MST) is a correlation graph. To cluster this graph, MST deletes the edge with the largest lengths (length can be defined by the Euclidean distance). The devided subgraphs (figure 3.2) can further be devided recursively until it reaches a threshold [**?**].

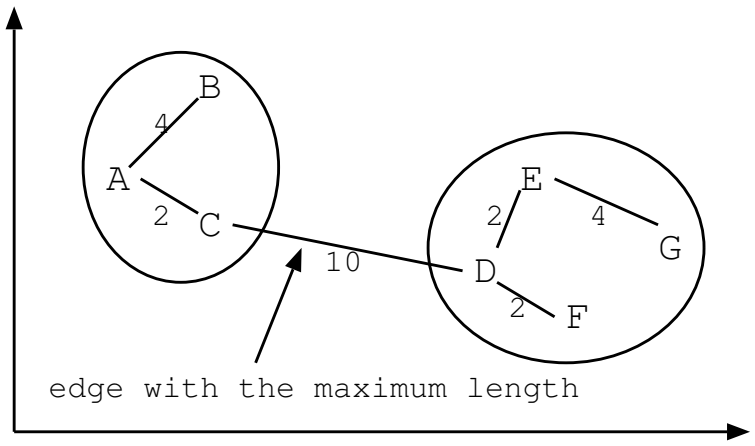

Figure 3.2: Divide graph on the base of Euclidean distance into two clusters by using the minimal spanning tree algorithm.

Advantages:

- Easy to understand and implement

Disadvantages:

- The stop criterion is a fix threshold.
- It first devides individual devices from the main graph.

3.3.2 K-means clustering

The *k-means* algorithm is popular and easy to implement. As a precondition, it is necessary to define the number of clusters, represented by k. The algorithm consists of four parts:

1. Choose the number of clusters k and define their center randomly.
2. Assign each device to the closest cluster center.
3. Recompute the center of each cluster (incoporate all devices of the cluster).
4. Reassign each device to the closest (new calculated) cluster center.
5. If the assignment has changed start again with step three. Otherwise the local minimum is found.

Advantages:

- Easy to understand and implement
- complexity is linear - O(n), where n is the number of devices

Disadvantages:

- The number of clusters is fixed.
- A metric distance is used for correlation (to define center of cluster).
- Derived clusters may be placed only on local minimum.

3.3.3 Normalized cut

Recently several new algorithms were developed in the field of computer vision for segmenting an image. One of them, Normalized cut developed and published by J. Shi and J. Malik [**?**], uses not just the minimum cost to find an optimal cut to segment an image into two sets, rather it computes the *normalized cut* as a fraction of the total edge connections to all other nodes in the graph. Thus the optimal one does not devide first individual pixels at the border of an image.

They transform the image into an undirected, weighted graph before segmenting this. The weights represents a feature (brightness, distance, color, etc.).

Advantages:

- Does not divide individual devices
- Builds up a binary tree
- No metric distance required

Disadvantages:

- Criterion for cancel subpartitioning a graph is also a threshold

3.4 Which one fits best?

3.4.1 Measure similarity

Within the limits of this work, we survey only the first and second illustrated algorithms. We tried first to determine the correlation coefficient with a constant timestep of 30 seconds. But the originated graph exhibits to diffuse correlations. Probably the result can be improved with a much smaller timestep. Thereby

the requirement of memory would increase. Furthermore the problem of different behaviors at different timepoints isn't solved.
We assumed that the most information about relationship between devices is provided by the temporal activity of devices. Thus we choose an Hebbian style learning algorithm to measure the similarity .

3.4.2 Clustering / partitioning

The k-mean algorithm bases on a metric system. But our reward map is not unambiguous convertible into a metric graph. Also the starting position of defining the number of clusters is not desirable.
Our first implementation was a kind of minimal spanning tree (MST) algorithm. But mostly it has clustered groups with single devices and such with lots of devices.
The normalized cut algorithm has the benefit that it discovers also the dependency between the clusters and represents this as a binary tree. Because it tries also to maximize the cohesions within the subsets, it does not segment predominantly individual devices.

Chapter 4

Theory

Like mentioned in the last chapter 3.4 we have chosen two concepts/algorithms to determine the functional relationship between devices and afterwards group them, if possible, hierarchically. In this chapter we explain briefly the theory of the hebbian learning to build a directed and weighted graph. And secondly we also give a short introduction into the normalized cut algorithm which we use to partition the graph and determine subgraphs.

After each section we try to explain where we see the bridge to our problem and why we have chosen this kind of algorithm. But for closer details of this theory we refer to the literature where these two concepts are discussed in several papers.

The results of applying these concepts/algorithms to our problem are discussed and mentioned in the result chapter 11 at the end of this document.

4.1 Hebbian Learning

In 1949, Donald Hebb [**?**] wrote: *When the axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.*

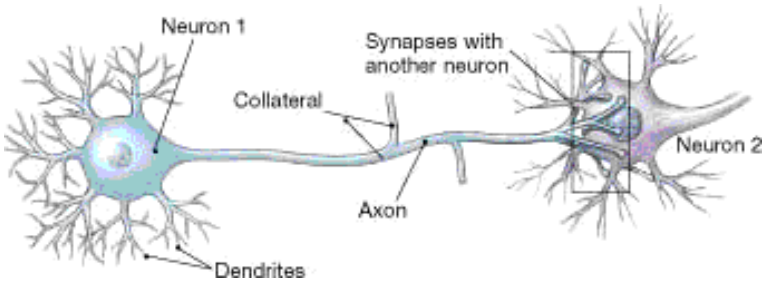

Figure 4.1: Two neurons, which are connected together over a synapse.

As a result, a learning method evolved for artificial neuronal networks, which was named after his inventor Hebbian learning. It is an instance of an unsuperviced learning procedure, where weights between each node from a network are ajusted so that they represent the relationship between the nodes. Each node stands for a neuron and each weight between neurons represents a directed synapse. The change of a synapse depends on the firing activity of the post- and presynaptic neuron. But this adjustmant can only be influenced by

these two neurons, a third neuron which is connected with a synapse to one of the others can't change the weight (see figure 4.2).

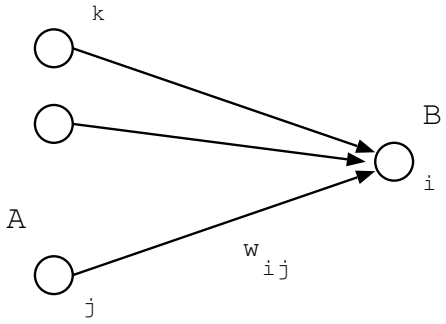

Figure 4.2: Only the firing activity of neuron A and B can adjust the weight w_{ij}. Another neuron k can't influence it.

Hebb's proposal was an inspiration for several experiments on the synaptic plasticity in the last three decades. But he formulated his principle on purely theoretical grounds. More than 20 years later [**?**, **?**], it was proved by Bliss and Lomo the first time experimentally. Though Hebb realized that a such meachanism could help to stabilize specific neuronal activity patterns in the brain. He also said if neurnal activity patterns correspond to behavior, then stabilization of specific patterns implies learning of specific types of behaviors.

Hebbian learning is interesting in cognitive science because it is unsupervised and has a local learning rule which means that it can be applied to a network in parallel. It is simple and needs very little computation and furthermore it is biologically plausible.

W. Gerstner and W. M. Kistler [**?**] defines Hebb's ideas more precise and mentions that it requires at least six aspects to formulate a useful plasticity model:

Locality means that the learning rule for the adjustment of synapse w_{ij} connecting neuron j to neuron i shuold depend only on the firing activity of j and i and not on the state of other neurons(see also figure 4.2).

Cooperativity Hebb's formulation 'takes part in firing it' implies that both the pre- and postsynaptic neuron must be active to induce a weight increase. It furthermore suggests a casual relationship between the firings.

Synaptic depression Hebb's original proposal gives no rule for a decrease of synaptic weights. It only refers how a synaptic can be strenghtened. But because there is no bound on the synapsis weights it is not unproblematic. Often a decay function can fix this problem.

Boundedness The synaptic weights should be bounded in a range of $0 \leq w_{ij} \leq w_{max}$ where w_{max} is the maximal weight value that a synapse can have.

Competition It is maybe a good feature to introduce *competivity* which means: if one synaptic weight increases, it does this at the expense of others. These have to decrease first.

Long-term stability In many learning theories weights are only changes during a *learning session* and are taken afterwards as fixed parameters. But here the weights are adapded online and care must be taken that previously learned information is not lost eg. by overwriting it.

Weight w_{ij} of a connections from neuron j to i is considered as a parameter that can be adjusted. The process of parameter adaption is called *learning* and the procedure for adjusting the weights is referred to as a *learning rule*:

The hebbian learning rule specifies how much between two neurons should be increased respectively decreased and this in propotion to their activation. If x_i and x_j are the activations of two neurons, W_{ij} is the synaptic weight between them and γ the learning rate. The rule is defined (in its basic form) as:

$$\Delta W_{ij} = \gamma \cdot x_i \cdot x_j$$

where ΔW_{ij} is the change of the weight W_{ij}.

4.1.1 Temporal aspects

However Hebb formulated, *...and repeatedly or persistently takes part in firing it...*, it's a little vague. It means that both neurons have to be active in order to strengthen a synapse. And active itself can also have several meanings. When are two neurons active together? This temporal aspect has to be defined for each problem itself, but it has an important influence in the value of strenghtening / decreasing the synaptic weights.

The Δt between the firing of the pre- and the postsynaptic neuron is essential for the value of strengthening synaptic weights. There are experiments which have a negative exponential weight function 4.3, which calculates the ΔW_{ij} by taking Δt as its argument.

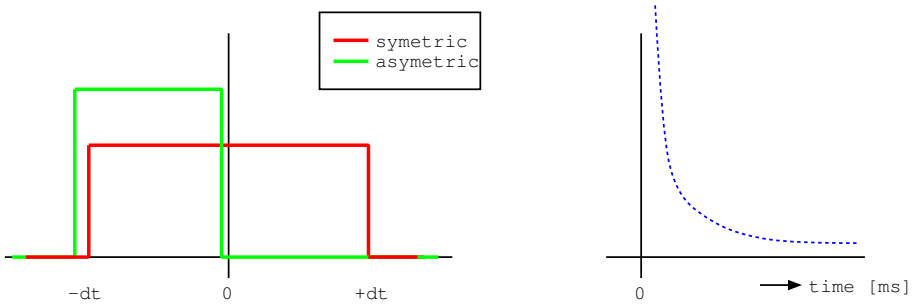

Figure 4.3: The left figure illustrates the two learning windows. Within the asymetric windows the weight of the synapse gets an additional meaning: causality. The right figure shows a proposed exponential weight function to strenghten a synapse. How much the synapse gets potentiated is depending on the timedelta between the two firing neurons.

If now a a neuron A fires slightly before another B, the synapse weight w_{AB} between them will be increased. Contrary, if the neuron B fires before A, the connection w_{AB} will be depressed. If the trend of this weight is persistent increasing, it is called long-term potentiation (LTP) and if it is decreasing long-term depression (LTD).

Often a so called learning window 4.3 defines the border how long two firing neurons are considered to be active together. And if this window is asymetric the weights can also say something about the casual relationship between two neurons, because only firing sequences are considered where the presynaptic neuron was active slightly before the postsynaptic one.

4.1.2 Adaption to a multisensor environment

In the previous section the theory of hebbian learning is explained. Here we want to make the bridge from this theory to a multisensor environment, like for example a typical commercial building. Such a building is equipped with several different sensors and effectors (lights, blinds, wallswitches, presence detector, outside sensors etc.). We think now an environment like this could have some similarities to neurons. The functional structure of them can maybe derived from the firing activity of each device (which stands for a neuron).

We want to examine, if we can also apply the hebbian learning here to discover the functional structure of a building. We think that devices which are active together have a stronger casual relationship than others which implies that they belong together somehow. This can even correspond to the physical structure. For this we build a network (we call it graph) where every node has an edge to every other node. The weight

of each edge is dynamically adapted and this online. The edges are directed and presents therefore also the casual relationship.

We illustrate this mentioned process in the next chapter 5 in detail.

4.2 Normalized cut algorithm

J. Shi and J. Malik [**?**] propose a new automatic grouping algorithm, especially for segmentation of images in the field of computer vision, where they try to extract the global impression of an image. We get inspired by this algorithm, because they treat an image as an weighted undirected graph. The weights of the graph represents the distance between two nodes as well some features (eg. of the image: brightness, color, etc.) which can influence the weight.

Other graph partitioning algorithms measure the cut as the sum of all edges which have to be removed:

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v) \tag{4.1}$$

where u are the devices which are in set A and v those in set B and the weight of the edge between node u and v is defined as $w(u, v)$.

The goal is to minimize this cut, but this means that it first partitions individual nodes (see figure 4.4).

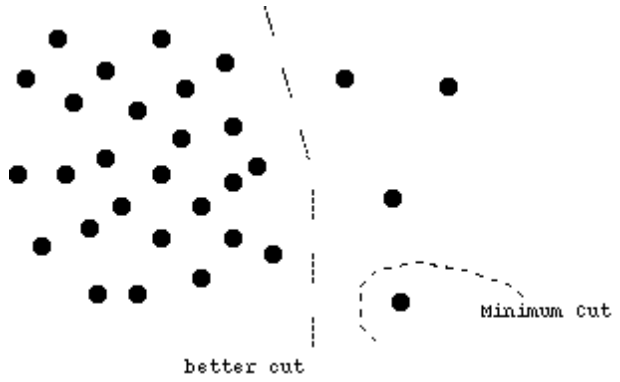

Figure 4.4: On the right a bad minimum cut which partitions just one individual node. It would be better, to partition first the group on the left with a higher density from the right one. The normalized cut tries to meet this need with a different calculation of the minimum cut.

Shi and Malik measure the disassociation between two groups by the ratio between the minimal cut and the total of edges from a group to all other nodes, and accumulated for each group. They call this size *normalized cut (Ncut)*:

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)} \tag{4.2}$$

where $assoc(A, V) = \sum_{u \in A, v \in B} w(u, v)$ is the total connection from nodes in A to all nodes in the graph V and the same for $assoc(B, V)$ with nodes in B.

Now an individual node has a higher Ncut to partition it alone. This results in finding a cut trough the graph and not just at the boundary.

They measure also the normalized association within a group by the ratio between the sum of edges inside of a group and the sum of edges connecting nodes inside the group with all others in the graph.

$$Nassoc(A, B) = \frac{assoc(A, A)}{assoc(A, V)} + \frac{assoc(B, B)}{assoc(B, V)}$$

Beside to have a Ncut as small as possible, they try to achieve a high normalized association within groups. And after transforming equation 4.2 they get:

$$Ncut(A, B) = 2 - Nassoc(A, B)$$

where both goals, minimizing the disassociation between groups and maximizing the association within a group, can be satisfied together.

4.2.1 Computation

First they compute several matrices: W is a symmetric nxn matrix with each individual edge weight. D is a nxn matrix with d on its diagonal, where $d(i) = \sum_j w(i, j)$. Further $b = \frac{\sum_{x_{i>0}} d_i}{\sum_{x_{i<0}} d_i}$, where x is a n-dimensional vector. If $x_i > 0$ node i is in partition A, and if smaller it is in B.

After several transformations (where a term that contains x is substituted with y) they prove that they can minimize

$$\frac{y^T(D - W)y}{y^T Dy} \tag{4.3}$$

with the constraint that $y_i \in \{1, -b\}$ and $y^T D1 = 0$ (where 1 is a vector with ones), if they solve the follwing generalized eigenvalue system:

$$(D - W)y = \lambda Dy \tag{4.4}$$

Further they say that the eigenvector y of the second smallest eigenvalue, by solving equation 4.4, is a good approximation for the normalized cut problem. An element of this eigenvector belongs to the first partition A if $y_i > 0$ or if $y_i < 0$ it belongs to the other partition B.

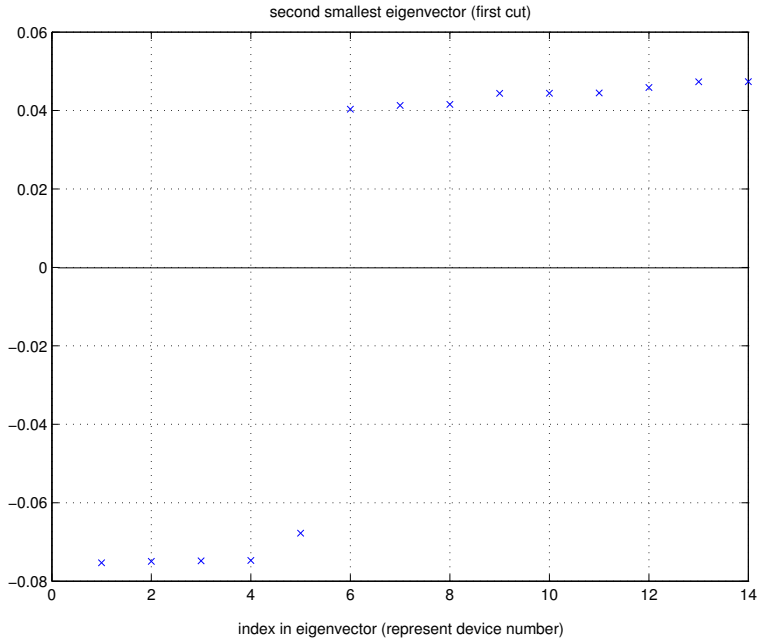

Figure 4.5: A second smallest eigenvector: All elements above zero are in one partition, the others which are below zero are in the other partition.

Figure 4.5 shows an eigenvector from a second smallest eigenvalue. This vector is an approximation to the discrete one. There are 14 nodes, whereas the first five are belonging to one group and node 6-14 to the

other. Shi and Malik say that in their experience the second smallest eigenvalue is very close to the optimal normalized cut value, but there is little theory on how much they can deviate.

To conclude, Shi and Malik show that an approximation for the Ncut can be found efficiently by solving a generalized eigenvalue problem, although minimizing the Ncut itself is exactly NP-complete.

4.2.2 Algorithm

Their algorithm is defined like this following enumeration:

1. Construct a weighted undirected graph $G = (V, E)$, where each edge represents the similarity between two nodes.
2. Solve $(D - W)x = \lambda Dx$ for eigenvectors with the smallest eigenvalues.
3. Use the eigenvector with the second smallest eigenvalue to partition the graph into two groups.
4. Decide if the current partitions can be divided again by comparing the eigenvalue with a threshold. Partition each group recursively as long as its eigenvalue is smaller than the threshold.

4.2.3 Transfer to clustering our hebbian learned connection graph

And why should we even use this normalized cut algorithm to partition our graph? First it has the good feature that it does not try to cut individual nodes first. In a typical commercial building it is not likely that a device is installed somewhere alone, most of them appear in a group (eg. inside of a room). And even if it is our goal to determine groups which are the basis of any further controlling build on top of this, we can't controll an area with just one device because we need other input parameters.

Another nice feature is that it partitions the graph in a divisive way where the result is a binary tree. Maybe we can benefit from this hierarchical relationship between groups by transfering any policy between groups in a future step.

Chapter 5

Our approach to the problem of structure

After surveing the different measurements of similarity and clustering algorithms, we decided to use a kind of hebbian learning to determine the relationship between devices and the normalized cut algorithm for partitioning our graph in chapter 3.4.

This chapter explains our process in detail and refers for discussion also to chapter 11. In short, our process is defined as follows:

1. Determine the relationship between devices by the temporal occurrence of thier state updates.
2. Transform directed graph in a undirected graph.
3. Partition the graph and try to find any subgraphs in it.
4. Track the result what means to merge the new derived structure as good as possible with the old structure.

It is also worth to mention that our process is totally event-driven and thus adapts online to any change. The following sections illustrates each step in detail whereby any problems are also discussed.

5.1 Determine functional relationship

Our analysis (see time analyis results 11.2) has shown, that our assumption is mostly true, where we say if a local area of our multisensor environment changes, it was caused by more than just one state update of a device. E.g. if a user enters a room and switches on the lights or if the blind are closed it gets dark and the user leaves or needs artificial light by switch them on.

Like we mentioned in the theory chapter 4.1 scientists use a hebbian learning rule to adapt the strength of a synapse between two neurons. And these synapse can represent a learnt behavoir if they are strong enough. We think a commercial building can also be seen like a network of neurons and we try to determine the functional relationship with the activity of each device.

We build a graph with n nodes (corresponds to the number of devices) with a maximum number of $n \cdot (n-1)/2$ edges. Each edge has a weight and represents the realtionship with the node on the other side of the edge. To gain also a casual relationship we use $n \cdot (n-1)$ edges to have a directed, weighted graph.

Now, each time two devices (nodes) are active together, we adjust the weight of the directed edge depending on which one was slightly before. The most important information to adjust the weight of an edge is the

temporal aspect. Like other experiments in neuroscience have shown, the synapse gets potentiated with a maximum if the two neurons are firing almost together. An often mentioned model is for example the spike-time-dependent-plasticity (STDP). There it is proposed [**?**, **?**, **?**] to use an exponential function to calculate the value of strengthening a synapse.

This was an inspiration for us to design a similar function but analyse also different other functions. We call this functions reward functions and have also defined some constraints: *maxreward* is the maximum value which a edge can be potentiated at once, s is a smooth factor to stretch the exponentional function horizontally. Because we have seen that our system can have a delay to deliver the messeges, *gap* is a small number to bypass such delay errors. The result chapter 11.2 gives a more detailed explanation about this problem.

Now, the exponential reward function is defined like this:

$$\Delta t = t_{i+1} - t_i \tag{5.1}$$

$$R(\Delta t) = \frac{\text{maxreward}}{e^{\frac{\Delta t - \text{gap}}{s}}} \tag{5.2}$$

where t_i and t_{i+1} are the timestamps of the pre- and postsynaptic firing.

A known problem of hebbian learning is that the reward value could increase to infinity. Often a decay is used to bound this value in a specific range. We use a constant decay value, which depresses the weight each time a neuron is firing. If slightly afterwards another is also firing, this synapse gets potentiated again. But if not it decreases, what can be an sign that this synapse was falsly potentiated once or is now a unvalid learned sequence. The total and delta reward are defined as

$$r_{i+1} = r_i \cdot \text{decay} + \Delta r \tag{5.3}$$

$$\Delta r = R(\Delta t) \tag{5.4}$$

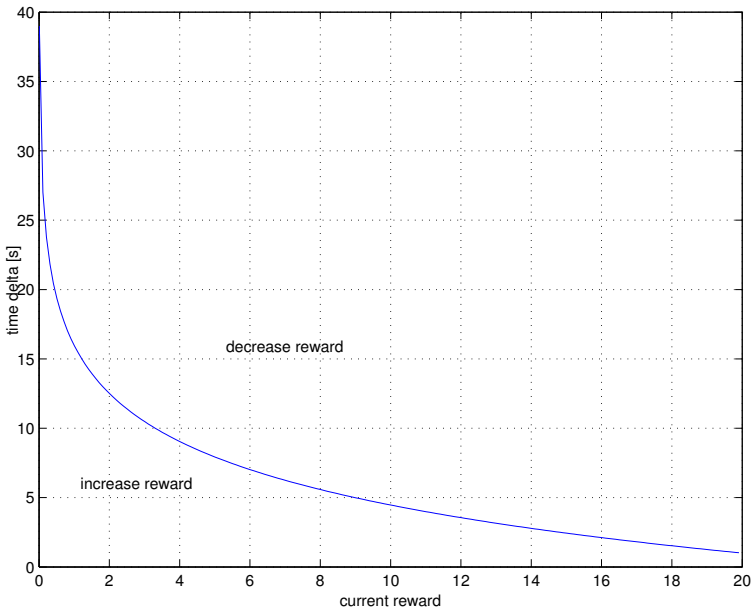

Figure 5.1: Limit value for keeping a constant reward. If Δt is larger it decreases, if it is smaller it increases.

Figure 5.1 shows if a weight gets rewarded with a specific Δt or if the reward is smaller than the decay value. The blue curve shows the limit value where the reward value gets decreased by a decayvalue and

afterwards strengthened for a specific Δt with exactly the same value. The total reward value is constant. It also illustrates that connections which have a high reward can only keep this by small Δt (smaller than 1s). Otherwise it receives less rewards than it gets decreased.

5.1.1 Transform directed reward graph into undirected

Our hebbian algorithm produces as result a directed reward graph. But the normalized cut algorithm needs as input a undirected graph. Thus it is necessary to convert each bi-directed relationship into an undirected (figure 5.2). There are two opportunities thinkable:

1. mean of both associations
2. choose higher association

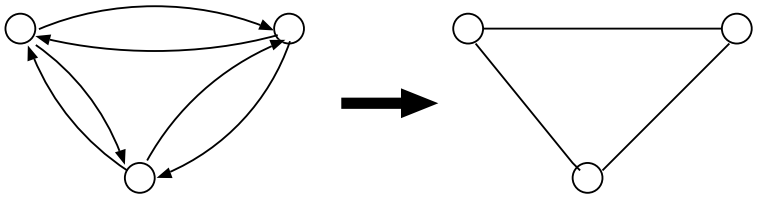

Figure 5.2: Both directed association must transform into undirected

If two devices mostly fire in the same order then the choice of the maximum is a good idea. In this case the correlation in one direction is extremly high and in the other nearly zero. In our system it is not guaranteed that two sequenced events always arrived in the same order (we use asynchrone communication). Due to this fact we calculate the directed association by the mean of both (see also "Blind pairs" 11.3.3 p. 53) but we also know that this is a very critical point.

5.2 Partition graph into subgraphs

Also event-driven we partition our graph into several subgraphs and try to extract clusters which are almost independent, but contain devices which have a high relationship. The normalized cut algorithm calculates the minimum cost to cut several edged of the graph. It always partitions a graph into two subgraphs as long as the sum of all removed edges is lower than a threshold.

The result is a structure of several clusters that are hierachically dependent. Each cluster represents a group of related devices and these groups can be the basis of any further learning.

5.3 Track structure of clusters

Each time we partitioning a snapshot of the hebbian learnt graph, the normalized cut alorithm builds a new hierarchical structure. Up to now we transfer the old structure into this new structure, but it would be imaginable to apply changes not until it verifies itself over a longer time. This could be a possibility to stabilize the structure.
In terms of running a control agent for each cluster it is necessary to keep as much information (about learned rules) as possible. Also we can't destroy all old clusters and create the new ones in accordance with the new structure. Each learnt cluster should transform into the most approximated new cluster by add and remove devices.

Reduce into a flat hierachy

It is a complex task to track the hierarchical information and use this to transform the old into the new structure. Because of that we transform it at the moment into a flat hierarchy. As a result we gain a one layer structure with the bottom clusters of the binary tree (figure 5.3) and loose the hierachical dependencies.

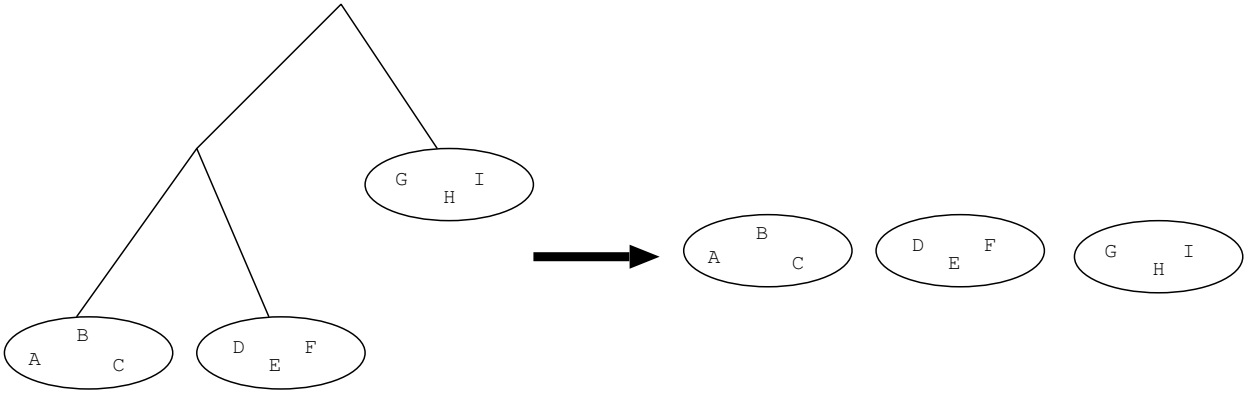

Figure 5.3: Transmutation of a hierarchical structure into a flat structure.

We know that it could be possible to track the structure in more sophisticated way, but up to now we haven't such an implementation.

Reallocation of clusters

First we compute which cluster from the old structure belongs most to the ones form the new structure. Then we add/delete devices from each old cluster until they match with the new structure. This means not to have any losses of information, as we ensure that each cluster corresponds to one of the new determined structure.

Part III

Architecture

Chapter 6

System Architecture

This chapter explains the architecture of our test environment. We mention how we have access to all devices of the building and how they are notifying any state updates.

6.1 Basic principles

A typical office building is equipped with several sensor and effector devices. Mostly these are connected to a fieldbus and can be controlled/accessed from one point. There a building manager can set some fix constraints between the devices (eg. wall-switch 1 controls light 1). Figure 6.1 shows the principle of our test environment at the Institute of Neuroinformatics. Beside the devices inside of the building, there are also some installed outside and measures the sun illumination, humidity, etc.

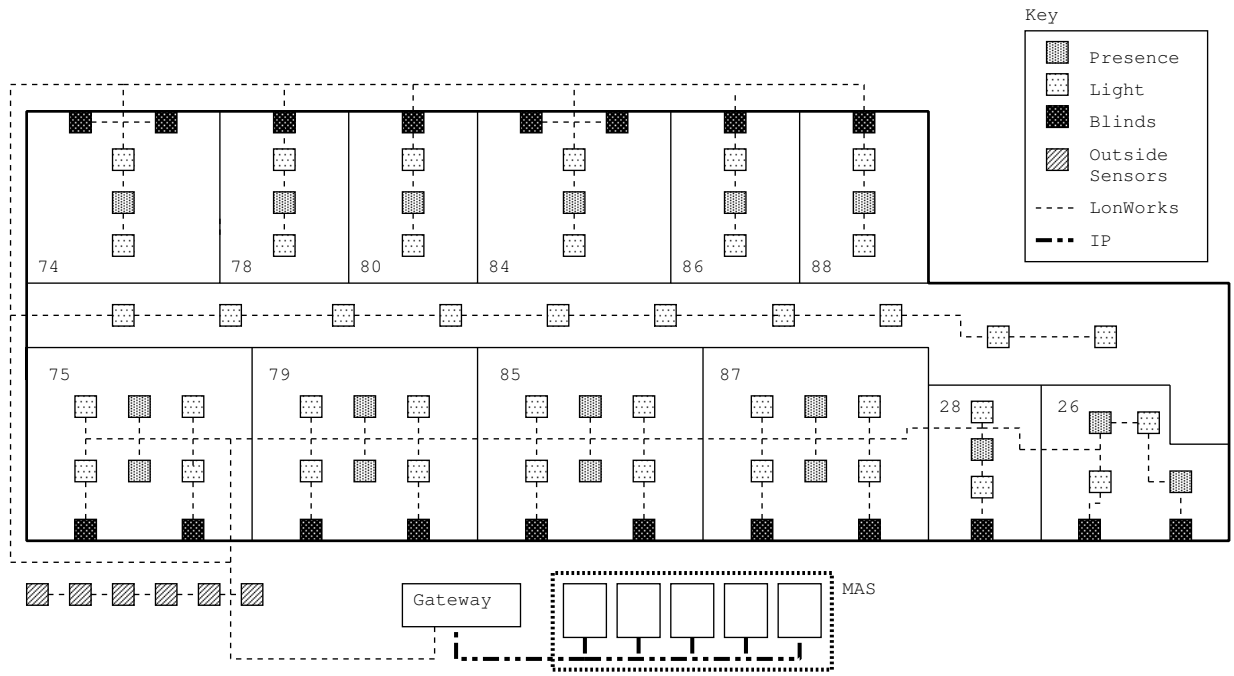

Figure 6.1: A floor of a building structured into rooms. Each rooms has sensors (eg. presence, daylight) and effectors (eg. lights, blinds, wall-switches).

All sensors and effectors of all rooms are connected to the same network (LonWorks). This network is,

with the help of a gateway, accessible from an IP based network. Each device has an unique LonWorks - Networkvariable (NV), and trough this gateway each NV can be set or requested. It is also possible to subscribe an interest in state updates for a specific NV.

Unfortunately we have seen that we can not subscribe to an arbitary number of NV. If there are too many subscribers, the system does not deliver any update messages. We guess it could be an overload of the whole fieldbus, because there was also a problem with the presence detectors because of that.

Chapter 7

Software Architecture

This chapter gives a brief overview of the AHA system which was developed by U. Rutishauser and A. Schaefer. For more details we refer to their diploma thesis [**?**]. First a short overview explains the architecture of their multi-agent-system (MAS) and used standards. Afterwards we show the dependencies between all agents and where our new agent is located and who are its communication partners.

7.1 Overview ABI System / Standards

The AHA System consists of a MAS which can run distributed on several machines. This MAS is structured into several independent and autonomous agents, where each of these agents pursues it is own goals independently and cooperates with other agents if necessary.

The AHA multi-agent-system is developed with the ABLE (Agent building and learning environment) framework [**?**, **?**], which fulfilles the java agent specification (JAS) [**?**]. JAS itself is an implementation of the FIPA abstract agent standart. FIPA (Foundation for Intelligent Physical Agents) [**?**] defines a set of abstract concepts like *agent*, *agent-direcotry-service* and *agent-communication-language (ACL)* but it does not provide any concrete implementation. A *lifecyle service*, which is responsible for creating new instances of agents and destroying them, is not defined in FIPA and JAS, because it is too implementation specific. The AHA system uses the ABLE *lifecycle service* implementation.

The whole messaging between agents is done asynchronously, which makes agents on the one hand independently but on the other hand more difficult to design. Agents can communicate directly (1:1) or indirect (1:n) by consulting a distribution agent. This agent is resposible for the indirect message flow, whereas each agent can subscibe to a topic and it maintains a list with all topics and subscribers. To support a fast communication it uses a threadpool which is capable to process a high amount of messages. In the MAS serveral message types can be sent, and it is exactly defined which agent can send/receive which message types. Consult [**?**] for a detailed definition of all possible message types.

7.2 Agents and their interaction

As mentioned the AHA system consists of several independent agents, which are trying to reach a goal together. Whereby an agent can by no means be restricted to be some part of software, as it can also be a person or a piece of hardware.

Figure 7.1 shows the dependencies between the agents of our AHA system. The solid line shows the directed dependency. It is also show by the dashed lines that each agent needs the distribution agent to communicate with the others asynchronously.

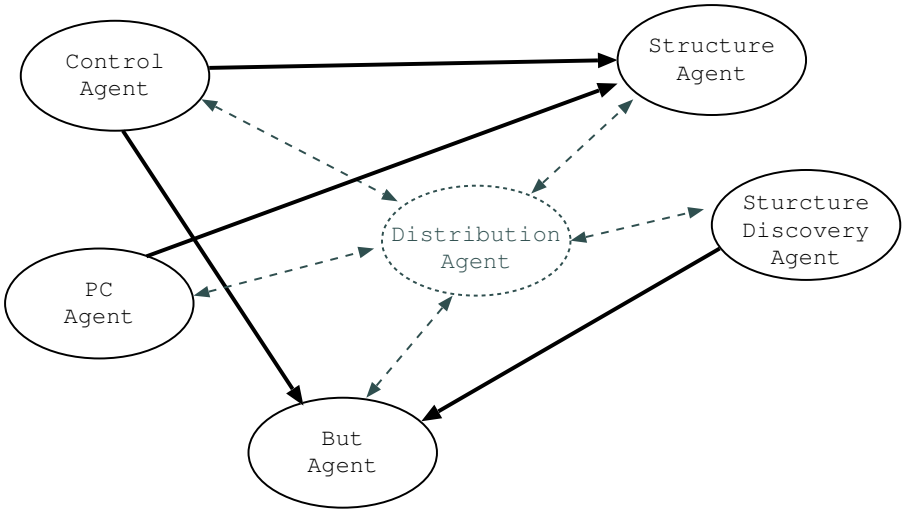

Figure 7.1: Dependencies between agents in our multi-agent-system (MAS).

Each agent has a different specific function/goal which it trys to reach. The following sections give a short description of every agent and explain their responsibilities.

7.2.1 Control agent

The control agent is responsible for the learning and controlling of a single room itself. Thereby we treat a room more abstract as a functional cluster and not just with its physical borders. Such a cluster could also contain devices which have another similarity than just being in the same room (eg. the same orientation). However, it requests this information from a structure agent which delivers all devices belonging to the requested cluster.

It controls all effectors and subscribes interests in all sensors of its cluster. A typical cluster can also have subclusters, because the structure is build hierachically. Depending on the number of subclusters the control agent instanciates a number of learning units.

7.2.2 Structure agent

The task of the sturcture agent is to offer any information of the **static** structure of a building. While starting it loads a configuration file, which contains each device in a hierarchical structure.

7.2.3 Structure discovery agent

The structure agent can only provide information about the static structure whereas the building behaves in a dynamic style or rather its inhabitants. To learn about such an environment it is essential to understand and record its **dynamic behavoir**. The insights must be the basis for any further learning. The structure discovery agent subscribes interests to all active devices and learns the functional relationship between them. In a second step it partitions them hierachically into several clusters.

7.2.4 Bus agent

The Bus agent provides the one and only interface to the fieldbus and also the access to every sensor/effector-device in our environment. It directly communicates with the LonWorks gateway and acts like a proxy

between the agents and the gateway. It is also the bus agent who notifies any interest in a NV to the gateway and does a subscribtion.

7.2.5 PC agent

The PC agent is a new agent developed by Tobi Delbruck (tobi@ini.phys.ethz.ch). It serves as a better presence detector, because we had several problems with the normal presence detector caused by a problem of the manufacturer.

The new agent acts as a server with serveral thin clients running on different workstations. And so they can also serve as a personal presence detector, because everytime the user works with/leaves his computer, the client notifies the PC agent about this. The PC agent spoofes now the control agent by send an artificial presence message.

Another feature is, that the agent can give each client individual accessions to control several effectors. A user can so control for example the lights and blind in his room.

7.2.6 Distribution agent

Like already mentioned, the distribution agent makes sure that an asynchronous communication between in our MAS is possible. To serve every request as fast as possible it has a threadpool. It also maintains for each interest topic a table with its subscribers. If it receives message under a spefic topic, it delivers this to all subscribers. As a result the agents do not have to know something specific about eachother.

7.2.7 Virtual person agent

Another agent which was developped mainly to personalize the system was the virtual person agent. In future it is thinkable that many are carrying any kind of mobile device (mobile phone, pda, etc.) and these have to register at the closest access point to gain access e.g. for the internet. This agent detects an individual person when he is subscribing at an specific access point.

Part IV

Design and Implementation

Chapter 8

Structure Discovery Agent

8.1 Functionality

The aim of the *Structure Discovery Agent (SDA)* is to discover the functional relationships between all devices in a office building and group them to independet clusters. Further devices of this clusters can be controlled by the AHA system. To achieve this, the SDA knows all devices and will be informed about all state updates of them.
For a productive use, it would be necessary to integrate some management and autosave function into the SDA. After a restart the system should be able to recover itself.

8.2 Messages in AHA

The asynchronous communication between agents is realized via the *Distribution Agent* and special defined *AHAMessages*. An *AHAMessage* contains always, following information:

- Reference of the sender
- Message type
- List of parameters (key/value pair)

The SDA use three different message types: SUBSCRIBE_ INTEREST, UNSUBSCRIBE_INTEREST and VARIABLE_UPDATE. Each message type must contain some defined information as parameter list (more information [**?**]).

SUBSCRIBE_INTEREST

At each startup, the SDA register its interest to each device by sending a SUBSCRIBE_INTEREST message.

UNSUBSCRIBE_INTEREST

If a device is no longer interesting for the SDA, it has to unsubscribe its interests by sending an UNSUBSCRIBE_INTEREST message.

VARIABLE_UPDATE

Every state update of a device is notified by the *Bus Agent* by distributing a topic-based message (which contains the device ID and the new value) via the *Distribution Agent*. This update message is delivered to all subscribed agents.

8.3 Implementation

8.3.1 Design

The SDA is composed of sevaral classes. The diagram 8.1 shows the associations between the most important classes. Here we mention just briefly the responsibilities and functions of each class. For any further details consult our javadoc documentation.

Basic classes

EventCollector
The *EventCollector* is like a controller for structure discovery. It receive event notifications from the *StructureDiscoveryAgent* (or *EventSimulator*). In chapter 8.3.2 the data flow of *EventCollector* is showed.

HebbianRewardMap
It manages the directed reward graph and updates all edges driven by notifications from the *EventCollector*. At every update it increments a statistic of an individual fire counter for each device itself.

NormalizedCut
In a constant timestep the *NormalizedCut* is called by the *EventCollector*. It manages the partitioning of the reward graph. The actual partitioning algorithm is implemented in the *Partitionizer*.

Partitionizer
It partitions the reward graph into subclusters. This process is called again (recursively) until it reaches a defined threshold.

ClusterTracker
The *ClusterTracker* try to transfer the old structure into the new derived structure. This transfer should keep as many information as possible from the old system.

StructureDiscoveryAgent
This agent suits as an interface between the AHA system and the *EventCollector*. It subscribes itself to topics in order to receive update messages of the devices. In further plans it should manage any backup and recovery functions and inform cluster agents of modifications.

EventSimulator
For testing the algorithms we used this simulator instead of the *StructureDiscoveryAgent*. Based on the log file *busTrafficLogger.txt* it simulates a real event stream.

EventFilter
This is an interface for an specific implementation of a filter. Filter are useful to get rid of useless events caused by the fieldbus. Chapter 9 gives more information about our realized filters.

Cluster
The class *Cluster* represents a found cluster. It contains devices and subclusters.

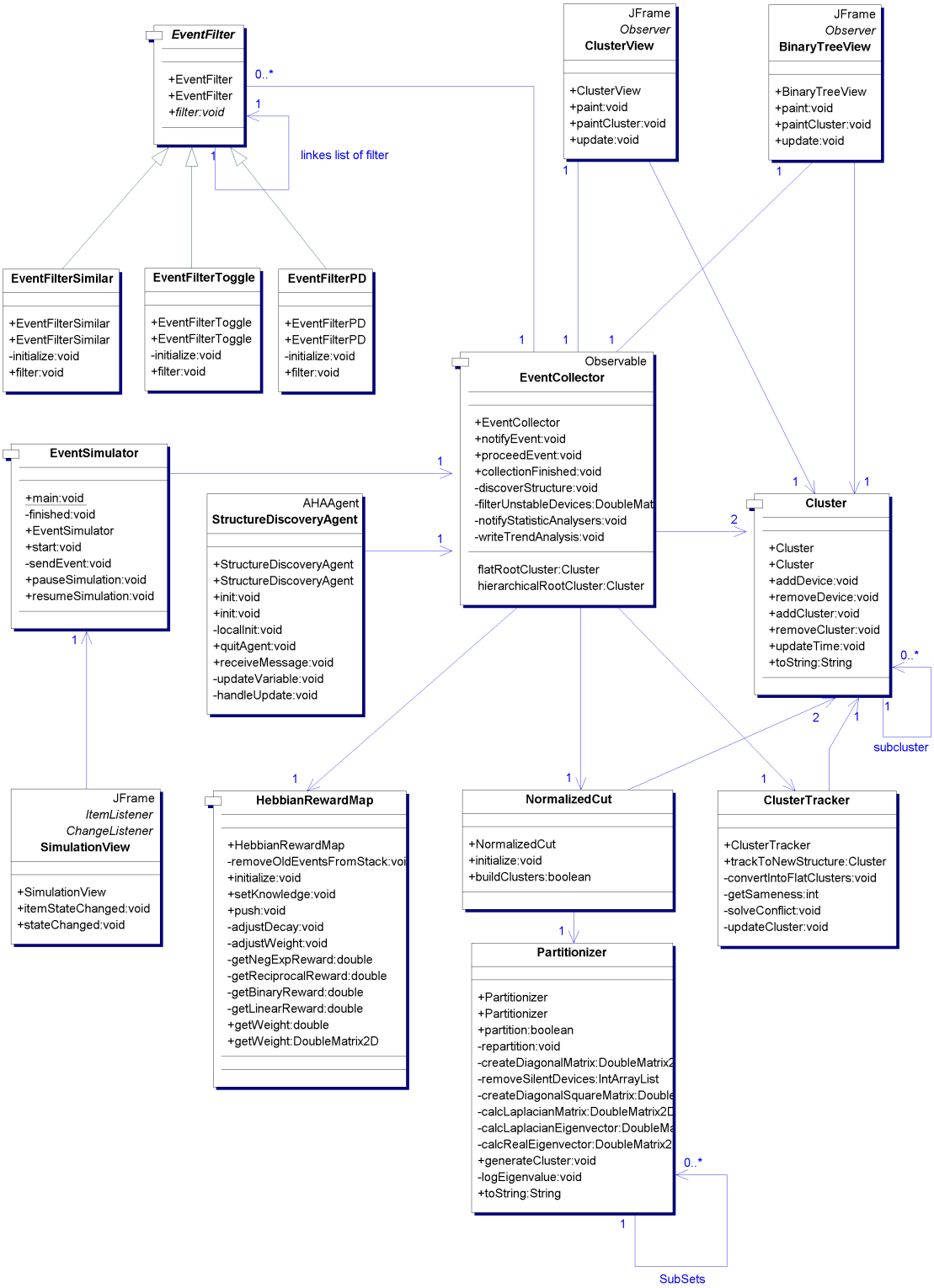

Figure 8.1: UML class diagram of SDA

Helper classes for analysing

To analyse different aspects of the SDA we have implemented some helper classes. In regular work they wouldn't be imporant.

ElementInfoMapper
The class *ElementInfoMapper* containes all nativ information about a physical building (existing rooms and their devices). This information are used to compare the discovered structure with the physical structur.

ClusterStatistic
Generate a statistic of occurence of different cluster sizes and of occurence of different cluster combinations.

ClusterStatisticAllocation
Create for each device a statistic of the correctness allocation.

ClusterStatisticQuality
Each derived cluster is allocated to the physical room with its highest similarity. This class records the mean quality of allocated clusters for each room. Quality mean: How many devices (percent) does a cluster contain just from one physical room.

EventDelayAnalyser
The *EventDelayAnalyser* records all timedelays and generate the following statistic:

- Lists of timedelays inside each room and between each room combination.
- How often which device-device sequence occurs (inside each room and between all rooms).
- Listing of timedelays for each device-device combination

EventSequenceAnalyser
Helper class for *EventDelayAnalyser*.

EventOccurrenceCounter
The *EventOccurrenceCounter* counts the number of released rewards for each device. Thereby it separates between rewards inside and outside the same room.

8.3.2 Data flow

The *EventCollector* is event driven by the SDA or the *EventSimulator*. The general process flow is illustrated in the collaboration diagram 8.2. To save CPU time we do not cluster the reward graph each time we receive an event. Steps 1.3 and 1.4 of our approch 5 are only executed once in an hour.

8.3.3 Visualisation

We implemented two different GUIs for illustrating the discovered structure. The first implementation (screenshot 8.4) shows a flat structure. Because the source data for drawing this structure are tracked by the *ClusterTracker* the order is more stable as in the other, second (hierarchical) GUI. The hierachical visualisation (8.3) shows a binary tree as grouped sets. Two groups are always combined with a rectangle and the label describes the position of the cluster and its relatives (A means left and B right pratition).

8.4 Used frameworks

Our SDA uses other frameworks, which we want to mention here briefly:

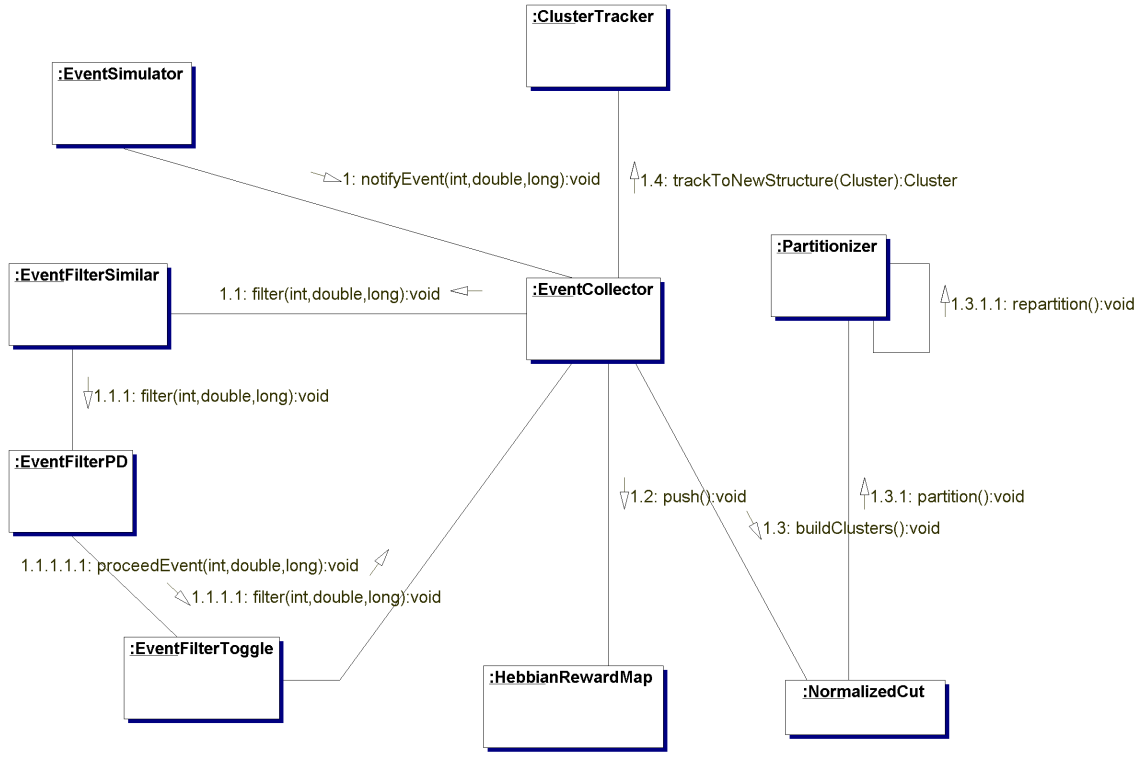

Figure 8.2: Collaboration diagram for incoming event and reclustering

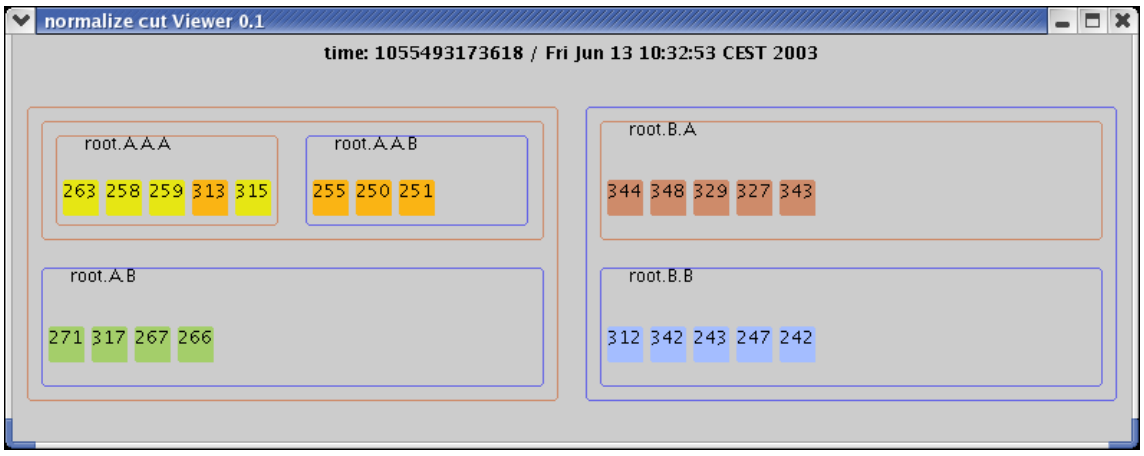

Figure 8.3: Screenshot: Hierarchical cluster viewer

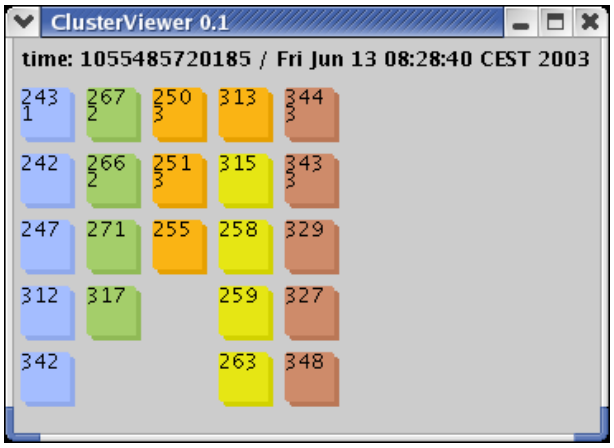

Figure 8.4: Screenshot: Viewer for flat cluster structures

ABI

ABI is an abbreviation for *Adaptiv Building Intelligence* [**?**]; a framework for a building control system based on a multi-agent-architecture developed by Ueli Ruthishauser and Alain Schaefer. The SDA expands this framework with new features (more details about ABI in chapter 7 on page 26).

ABLE

The Agent Building and Learning Environment (ABLE) [**?**, **?**] is a java framework, component library, and productivity toolkit for building intelligent agents utilizing machine learning and reasoning. The ABLE research project is made available by the IBM T.J. Watson Research Center.

Colt

Colt [**?**] are open source libraries for high performance scientific and technical computing in java. We use them especially for matrix computing.

Log4j

Log4j [**?**] allows a controlled debug logging with different log levels.

Chapter 9

Filtering

Our test environment uses the LonWorks network which is a popular commercial building fieldbus (described in chapter 6). But this network and the attached devices don not work reliable enough. We found several dubious behaviors, e.g. a wallswitch was uptight and sent a SET_OFF command every 30 seconds or presence detectors which are toggling all the time and sending this in an interval of 90 seconds.

Because of this we had to implement some filter functions. Every event must first pass all filters before it can eventually reach our discovery algorithm.

We list here the most important filters and explain their functions:

9.1 Similar Event Filter

In principle this filter blocks all events from a device if they are similar to the last passed one. Sometimes the transport layer lose some events. Thereby it is possible that the light is turned off and we never know something about this. Later if the light is switched on this filter blocks the event (because it thinks that the light was never switched off). For this case the filter has a timeout to block similar events (this timeout is re-triggered with each blocked event).

9.2 Presence Detector

The real signal of the presence detectors (PD) just reports movements. By every movement of a person the PD sends a OCCUPIED signal. But if the person moves it is likely that he is just moving a little and does not change the state of his environment by switching on/off any devices. And this would result that the hebbian learning reduces the correlations. Due to a firmware bug in the PD's they switch into an unstable state and send every 90 seconds an OCCUPIED and UNOCCUPIED signal.

This filter smooths the signal of PD's with a timeout of 5 minutes (figure 9.1). So we can reduce all this signals to a few OCCUPIED signals. The filter sends only a OCCUPIED signal at each rising edge of its state (e.g. two times in figure 9.1).

9.3 Toggle Filter

The current controlling algorithm of the AHA system isn't perfected. On special conditions, blinds and lights can switch their states within few seconds automatically. Such behavior adulterates the hebbian learning.

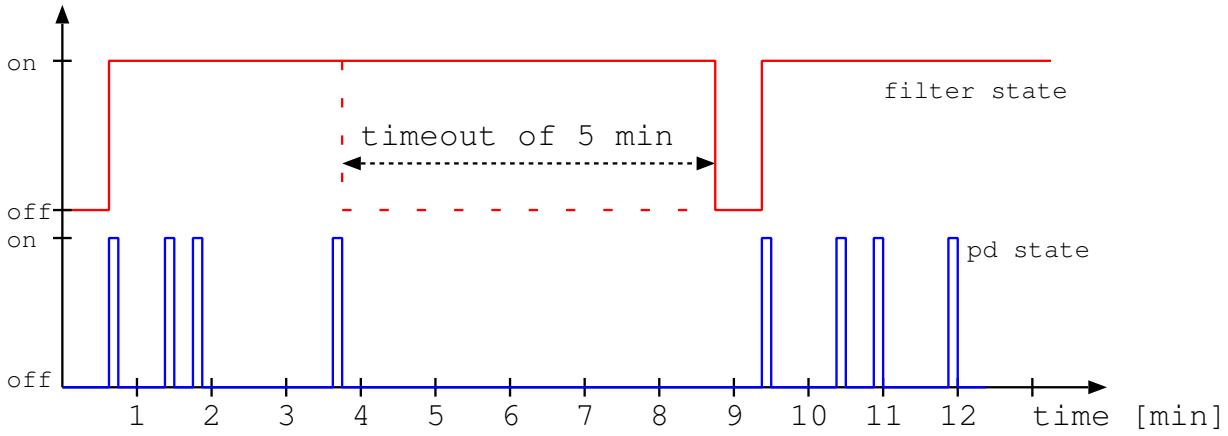

Figure 9.1: The red line represent the state inside of the PD filter and the blue one the state of the PD itself. Only events at the rising edge of inner state are passed by the filter.

This filter interprets two states which switches within 30 second as a toggle and blocks the second one (figure 9.2). The timeout is also re-triggered with each blocked event.

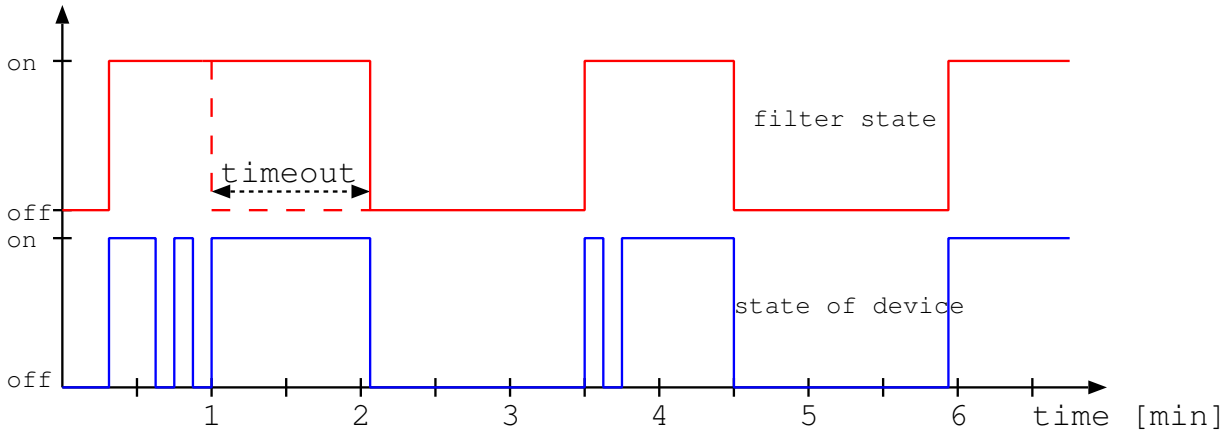

Figure 9.2: The red line represent the state of the filter and the blue one the state of the device itself. Only the events at the rising or falling edge of the filter state are passed by the filter.

Chapter 10

Simulation

By using the *structure discovery agent (SDA)* in a multi-agent-system, it receives messages from the *DistributionAgent* for each state update. For testing and analyzing our algorithm it is necessary to simulate this changes of states in a shorter time, but still based on real data.

In our simulation we replace the SDA with the *EventSimulator* which reads a log file and simulates all recorded events one by one, see figure 10.1. The current ABI system (to be exactly the *Bus agent*) maintains this logfile by writing each event in it.

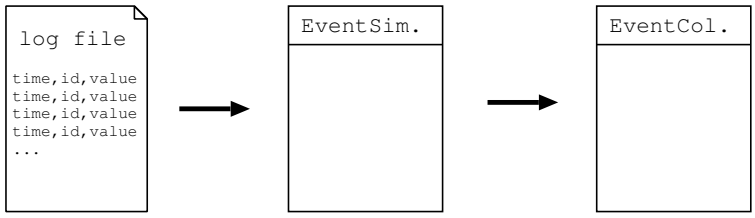

Figure 10.1: The *EventSimulator* parses the logfile and sends simulated events to the *EventCollector*.

10.1 Who records event data

The quality of a simulation is directly dependent on the underlying data. We record real data produced by human and system behavior. We expanded the *BusAgent* of the existing ABI system with this feature. This agent knows about all updates and can easy write a log file. The logfile consists of events where each carries a timestamp, value and device ID.

10.2 Features

In some cases it is interesting to analyze just some specific rooms. Therefore we implemented a feature that allows us to enable/disable each device. We can also define a start time for simulation. All events before this startpoint get ignored.

Part V

Results and Discussion

Chapter 11

Results and Discussion

This chapter gives a detailed explanation of our results. After illustrating our algorithm, the architecture and implementation we discuss here the gained structure information.

First we prove our assumption which is the basis for the whole hebbian learning by analyzing the time dependencies in and between rooms. After we illustrate and discuss the results from the hebbian learning, before we finally present our determined structure information.

11.1 How sparse is the data?

To give *sparse data* a more precise meaning the following tables show how many times a device has notified a changed state. Each device is located in a room and has a unique LonWorks ID which is also listed in the tables. We refer in this chapter several times to these devices, and use for this the Device ID.

Intelligent rooms:

Room Number	Device Type	Device ID	LonWorks ID	Fire Counter
55G74	light 1	1	242	424
55G74	light 2	2	243	304
55G74	blind 1	3	342	365
55G74	blind 2	4	312	367
55G74	presence	5	247	154
55G84	light 1	6	343	505
55G84	light 2	7	344	632
55G84	blind 1	8	327	378
55G84	blind 2	9	329	379
55G84	presence	10	348	176

Table 11.1: Firecounter of all devices in rooms controlled by AHA. Observation period: 06 may - 13 june (38 days)

Human controlled rooms:

Room Number	Device Type	Device ID	LonWorks ID	Fire Counter
55G86	light 1	11	266	77
55G86	light 2	12	267	73
55G86	blind	13	317	30
55G86	presence	14	271	118
55G80	light 1	15	258	293
55G80	light 2	16	259	11
55G80	blind	17	315	50
55G80	presence	18	263	113
55G78	light 1	19	250	39
55G78	light 2	20	251	38
55G78	blind	21	313	11
55G78	presence	22	255	180
55G26	light 1	23	180	20
55G26	light 2	24	181	181
55G26	blind 1	25	168	11
55G26	blind 2	26	170	33
55G26	presence	27	185	296
55G26	presence	28	189	195
55G75	light 1	29	60	56
55G75	light 2	30	61	14
55G75	light 3	31	76	134
55G75	light 4	32	77	7
55G75	blind 1	33	103	51
55G75	blind 2	34	105	40
55G75	presence	35	65	772
55G75	presence	36	67	803

Table 11.2: Firecounter of all devices in rooms controlled by people. Observation period: 06 may - 13 june (38 days)

11.2 Time delay

We show in chapter 4.1 that the temporal appearance of events is a criteria for the correlation. We assume that devices inside of a physical room are predominantly used within short-time periods. E.g. often two lights are turned on or off together. Such temporal compliances between different rooms are mostly randomly.

The analysis of frequency distribution about timedelays **inside** of physical rooms shows that the most correlations occur in the first three seconds (figures 11.1 and 11.3).
In contrast of this the timedelay **between** physical rooms are distributed over the whole time (figure 11.5). The peak at five seconds results from several unique device-device combinations. Whereas the peaks in histogram 11.1 and 11.3 results from few high-frequent combinations (table 11.2, 11.4 and 11.6).

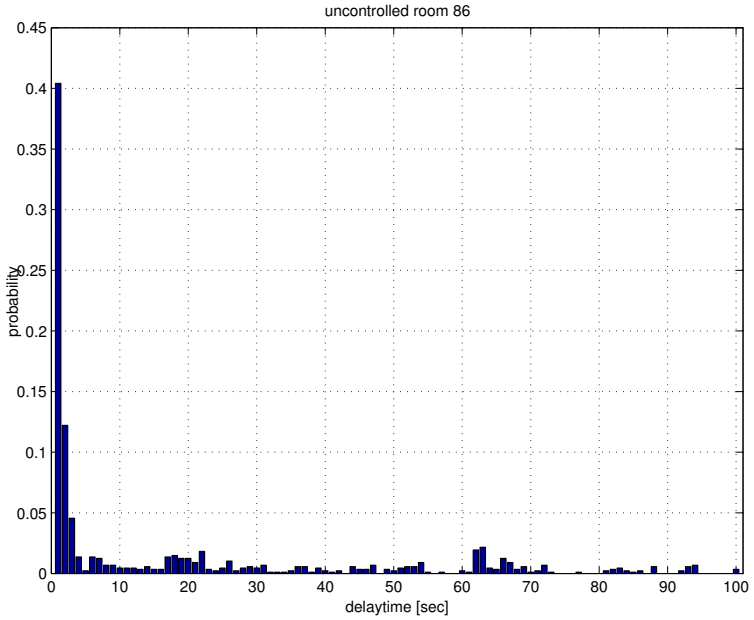

Figure 11.1: Distribution of timedelay of all event correlations inside room 86

type-type	ID-ID	counter	percent
light1-light2	266-267	64	74.4%
light1-pd1	266-271	12	14.0%
light2-pd1	267-271	8	9.3%
light1-blind1	266-317	1	1.4%
light2-blind1	267-317	1	1.4%

Figure 11.2: Number of timedelays (less than 5000ms) for device-device combinations inside room 86

Inside AHA controlled rooms there are more event sequences within a small delaytime (figure 11.7 (a)). Between the rooms the timedelay is arranged like an equal distribution (figure 11.7 (b)). This must be a consequence of the learning rules which are restricted to the physical room boundaries.

We prove our assumption by considering rooms in our test environment. But it is possible that there exists situations where our assumption fails, e.g. when a user never uses the blinds because the sun is never shining into his window. If that is true we could never detect any (short) event sequences with his blind.

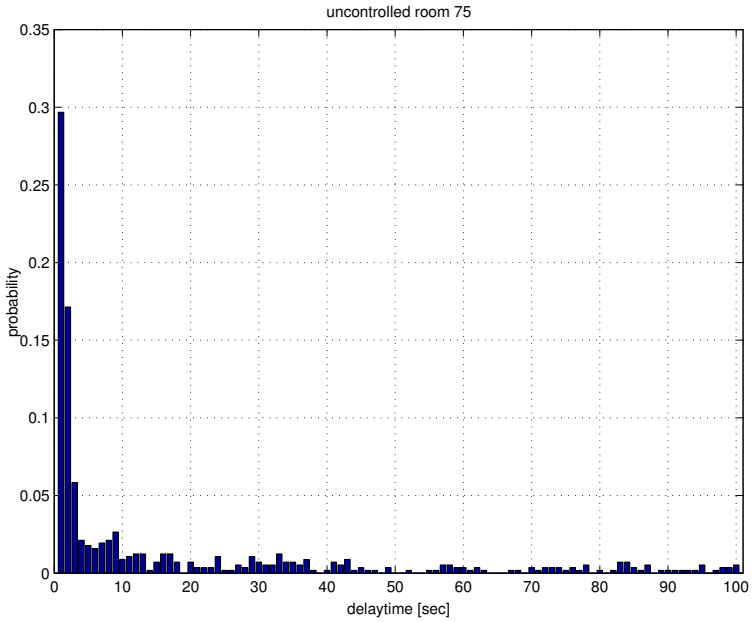

Figure 11.3: Distribution of timedelay of all event correlations inside room 75

type-type	ID-ID	counter	percent
pd1-pd2	65-67	174	57.6%
blind1-blind2	103-105	37	12.6%
pd1-light3	65-76	26	8.6%
light1-light3	60-76	21	7.0%
light1-pd1	60-65	11	3.6%
pd2-light3	67-76	9	3.0%
light1-pd2	60-67	5	1.7%
light1-light2	60-61	4	1.3%
light2-light4	61-77	2	0.7%
light2-light3	61-76	2	0.7%
light2-pd2	61-67	1	0.3%
light3-light4	76-77	1	0.3%
light1-light4	60-77	1	0.3%
light4-blind1	77-103	1	0.3%
light3-blind1	76-103	1	0.3%
light2-blind1	61-103	1	0.3%
light1-blind1	60-103	1	0.3%
light4-blind1	77-105	1	0.3%
light3-blind1	76-105	1	0.3%
light2-blind1	61-105	1	0.3%
light1-blind1	60-105	1	0.3%

Figure 11.4: Number of timedelays (less than 5000ms) for device-device combinations inside room 75

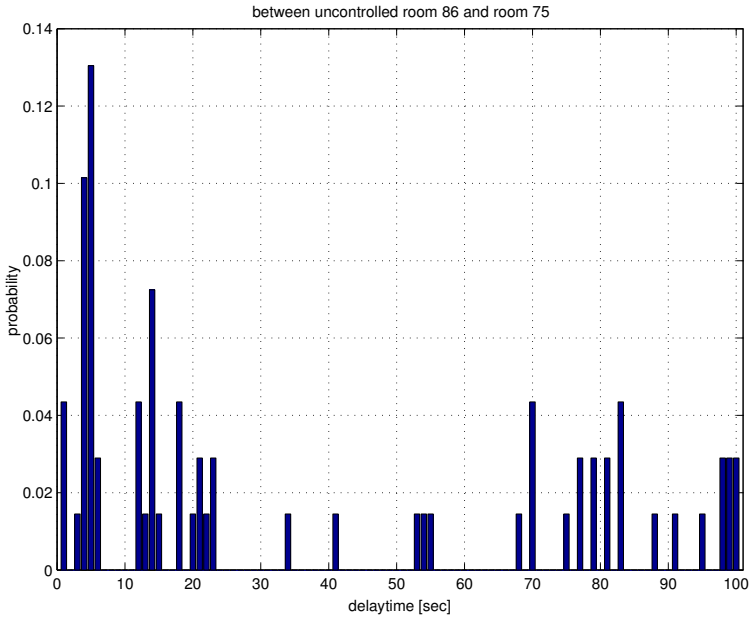

Figure 11.5: Distribution of timedelay of all event correlations between room 86 and 75

type-type	ID-ID	counter	percent
blind2_75-light2_86	105-267	2	10.0%
light1_75-blind1_86	60-317	1	5.0%
light1_75-light1_86	60-266	1	5.0%
light1_75-light2_86	60-267	1	5.0%
light2_75-blind1_86	61-317	1	5.0%
light2_75-light1_86	61-266	1	5.0%
light2_75-light2_86	61-267	1	5.0%
light3_75-blind1_86	76-317	1	5.0%
light3_75-light1_86	76-266	1	5.0%
light3_75-light2_86	76-267	1	5.0%
light3_75-pd_86	76-271	1	5.0%
light4_75-blind1_86	77-317	1	5.0%
light4_75-light1_86	77-266	1	5.0%
light4_75-light2_86	77-267	1	5.0%
blind1_75-blind1_86	103-317	1	5.0%
blind1_75-light2_86	103-267	1	5.0%
blind2_75-blind1_86	105-317	1	5.0%
pd2_75-pd1_86	67-271	1	5.0%
pd2_75-light2_86	67-267	1	5.0%

Figure 11.6: Number of timedelays (less than 5000ms) for device-device combinations between room 86 and 75

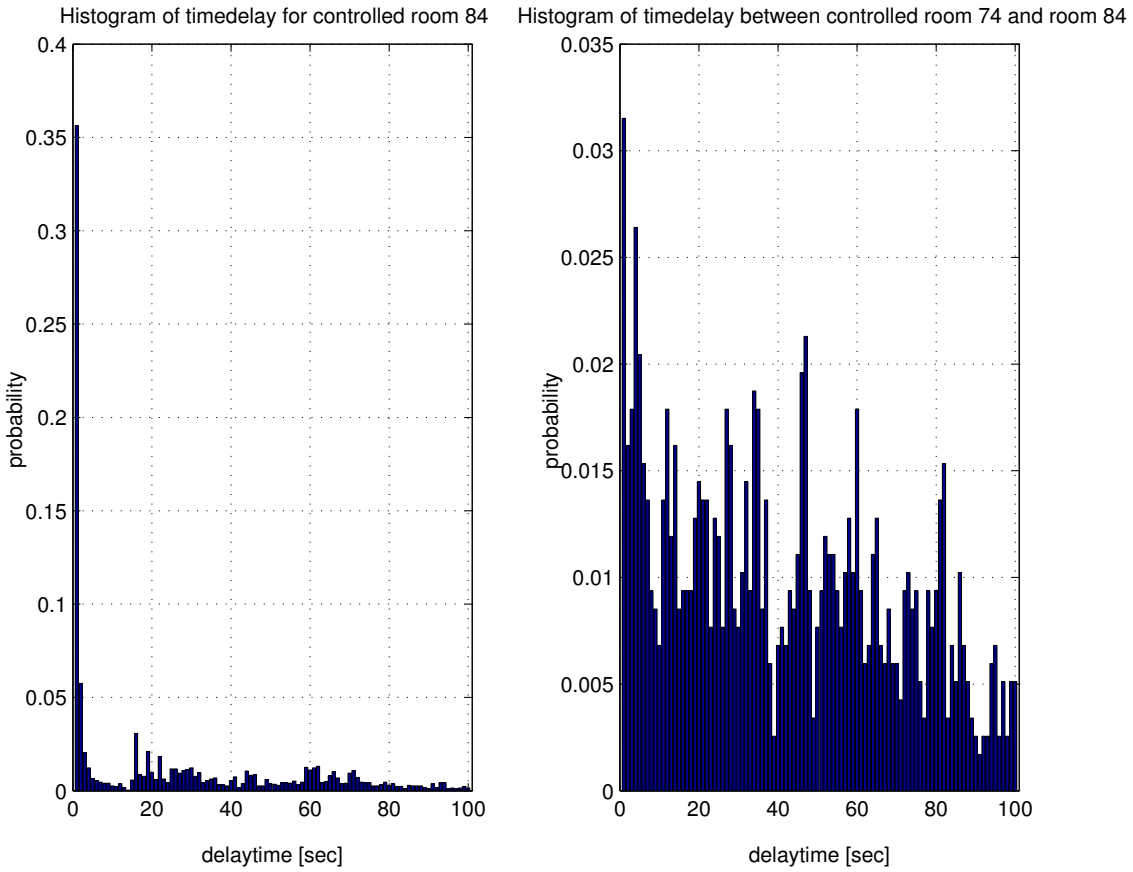

Figure 11.7: Distribution of timedelay of inside and between controlled rooms.

11.3 Hebbian learning

In this section we give a detailed explanation of several different reward functions and survey them. We also review our algorithm to determine the relationships by analysing trends of several device-device combinations and discuss open issues.

11.3.1 What's a good approximation for a reward function?

As we showed in the theory part, most information lies in the time between firing of different devices. The analysis has shown that the likelihood that two devices are somehow structurally dependent on each other is much higher if the time delta between two firing nodes is small.

Based on this analysis we wanted to potentiate a connection with a maximum reward points, if the time delta is small enough. The value of reward should depend on the time delay between them. We defined four different reward functions (11.1,11.2,11.3,11.4) which cover different areas (see also plot 11.8).

The reward function is only defined in a fixed learning $window = 300'000ms$, sequences which have a larger Δt are not rewarded. Long term behavoir should be learnt by an other instance, like for example the control agent of the ABI [**?**].

To exclude any system delay we have defined a $gap = 1000ms$ during which all connections get the maximum reward ($maxreward = 1.0$). The binary reward function only rewards connections which have a Δt equal or smaller than *gap*. All others reward in the whole *window*.

$$r_{linear}(\Delta t) = \frac{(\text{window} - \Delta t) \cdot \text{maxreward}}{\text{window}} \tag{11.1}$$

$$r_{negexp}(\Delta t) = \begin{cases} \text{maxreward} & \text{if } \Delta t <= gap \\ \frac{\text{maxreward}}{e^{\frac{(\Delta t - \text{gap}}{5000ms}}} & \text{if } \Delta t > gap \end{cases} \tag{11.2}$$

$$r_{reciproc}(\Delta t) = \begin{cases} \text{maxreward} & \text{if } \Delta t <= gap \\ \frac{\text{maxreward}}{\frac{\Delta t}{1000ms}} & \text{if } \Delta t > gap \end{cases} \tag{11.3}$$

$$r_{binary}(\Delta t) = \begin{cases} \text{maxreward} & \text{if } \Delta t <= gap \\ 0.0 & \text{if } \Delta t > gap \end{cases} \tag{11.4}$$

Figures in 11.9 shows the mean value of each connection over an observation period of 38 days (12may - 13june) and figure 11.10 shows their variance. Figure 11.9b clearly points out devices which have a high correlation and always change its states together. These are for example blind pairs, light pairs, presence and some other devices combinations. They can good be detected in intelligent rooms. But human-controlled rooms often do not have such short term sequences, that's the reason because the bottom right corner has not so intensive colors.

In contrast, figure 11.9a shows a very noisy image diagram. Sequences which have a high Δt are still rewarded and produce such noisy areas. Even if human-controlled rooms have much more sparse data they become more noisy because there are more sequences between rooms with a large learning *window*.

The negative exponential and reciprocal reward function 11.9c,d point out a much better image diagram. They still have some noisy values, but less than the linear rewarded one. They try to find connections between devices inside rooms, which typically have a small Δt, but rewards also sequences between different rooms if they occur often. We think these functions fits best as a reward function. Also other neuroscience projects are dealing with a negative exponential reward function, like we have mentioned in the theory chapter 4.1 on page 12.

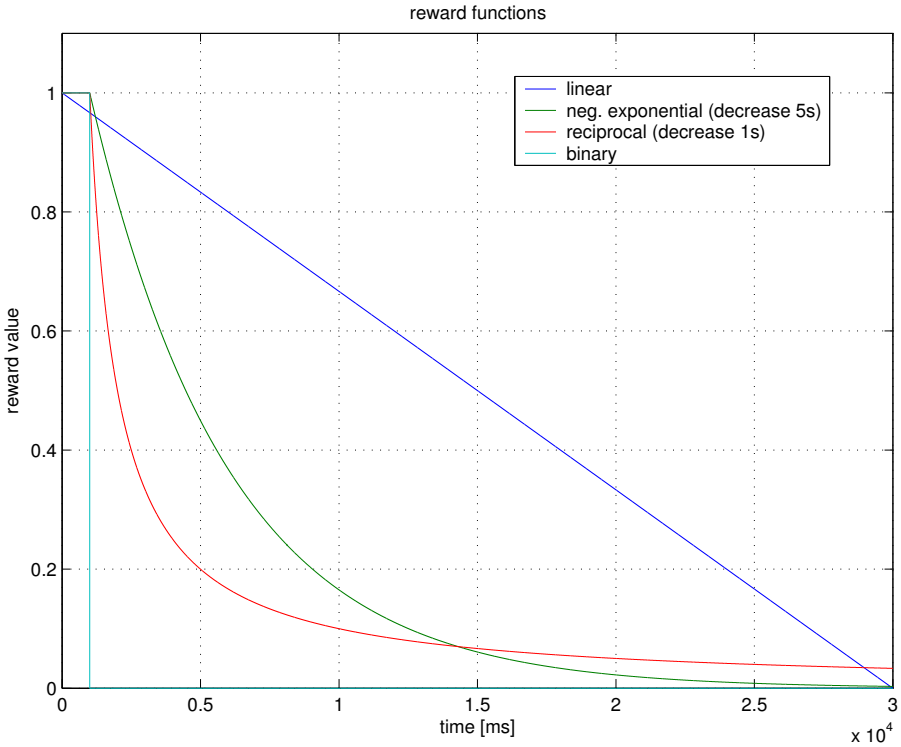

Figure 11.8: Four different reward functions (linear, neg. exponential, reciprocal, binary)

Figure 11.10 describes the variance of the mean 11.9 and shows that that the system is really dynamic. The mean values is taken over an observaton period of 38days. The variance is just between devices in the same room higher than zero. Other combinations are almost zero, which means that devices from different rooms do not influence eachother by looking at them over a longer period.

11.3.2 Decay

As mentioned above, it is a known problem that there is no bound on weight of the connections in hebbian learning. We solve this with a decay function which decreases a connection between two nodes if just one fires in an observed time window. This decreas depends on the current value of the connection. This means a connection with the maximum reward decreases more than a connection which was only once discovered. An example is shown in plot 11.11, where the reward value is 1.0. This value can result of a sequence which was rewarded with the maximum or with several rewards of different sequences. The plot illustrates now, how much the total reward weight between two devices decreases, if just one device fires within the learning window. It takes approximately 100 firings of one device, until the weight reaches zero. This would mean, that the devices are not dependent anymore or the learnt sequence was just a random one.

The drawback of this decay function is that knowledge gets lost, which was learned in the past. But on the other hand it is maybe not essential to keep as much as possible in memory. We think a building is not static as it is built, rather it behaves dynamic and adapts to the needs of its inhabitant and the current weather conditions outside. For this reason we think a decay function is justifiable which decreases sequence patterns from the past if they do not occur anymore.

Another issue is the integration of new devices. If there is no decay, it would take a long time until this device can build up rewards to other devices. With the decay we bound the reward indirect to a range and make sure that a new device can be integrated quite fast.

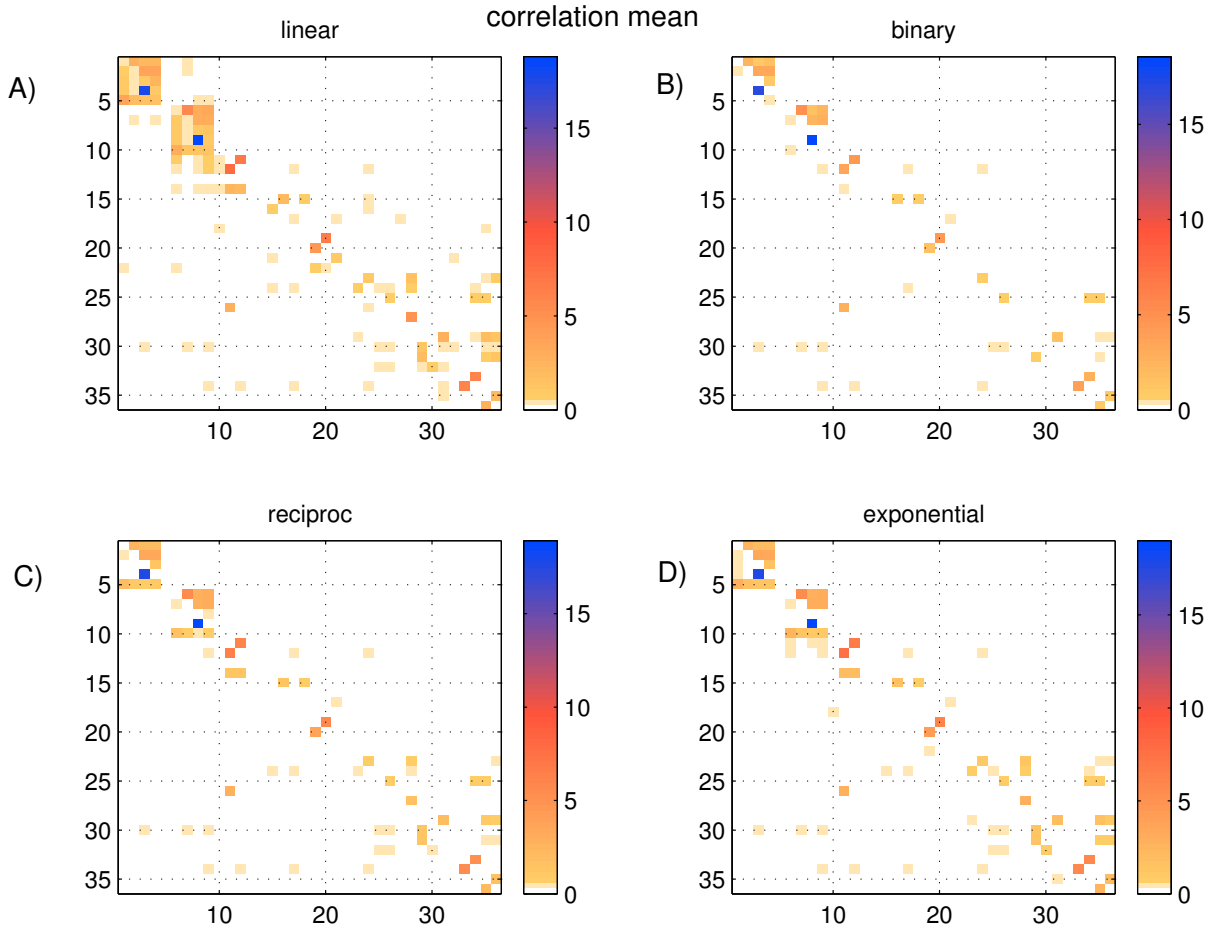

Figure 11.9: Correlation mean between each device pair gained by different reward functions: A) linear, B) binary, C) reciprocal and D) negative exponential. D presents only high correlated devices, whereas B is much noisier. All images have seven different rooms on thier diagonal, and the devices are willful ordered for a better understanding. Both axes reprents 36 devices, to understand the id consult table 11.2 and 11.1 on page 41 for the Device ID translation. gap=1s, learning window=30s, decay=.95

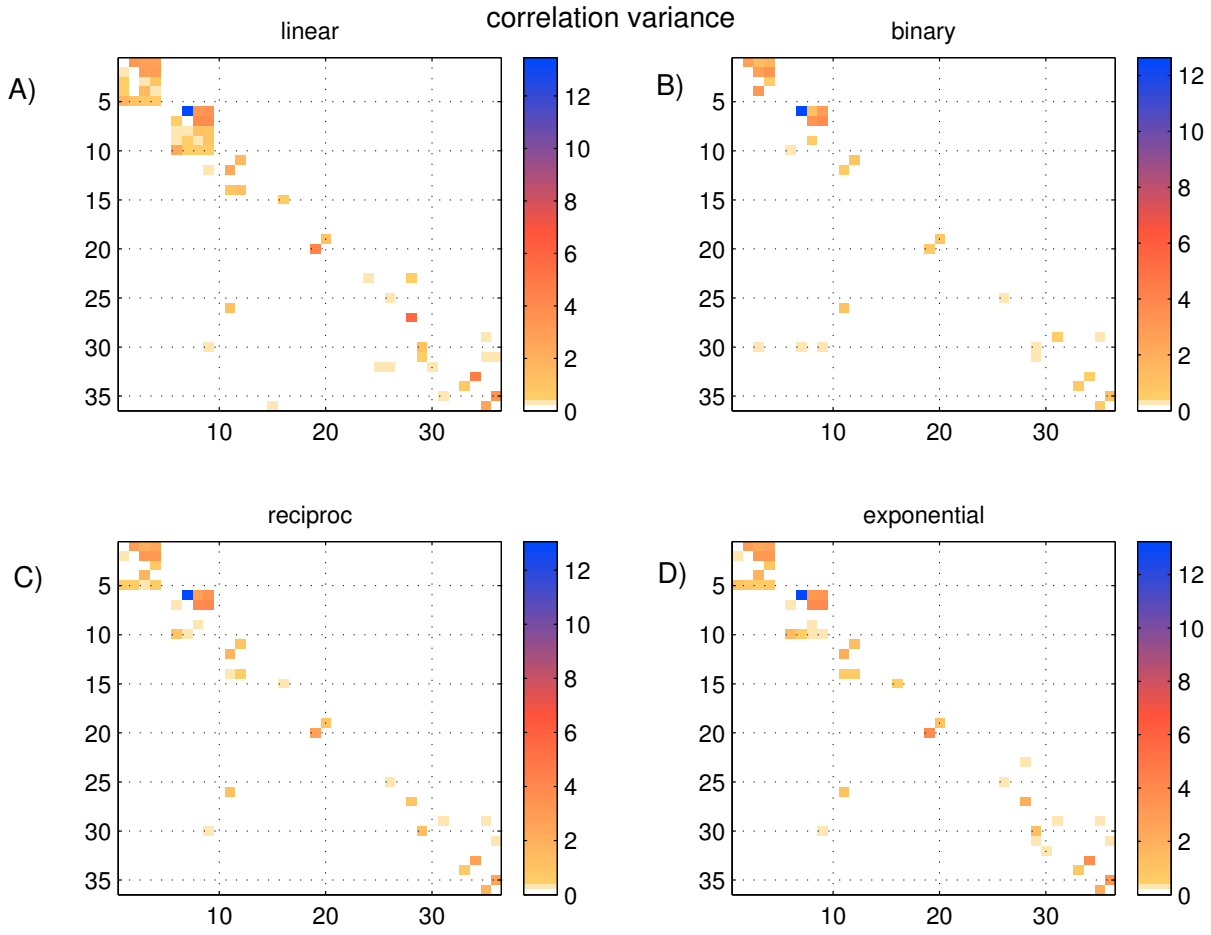

Figure 11.10: Correlation variance of the four different reward functions (A) linear, B) binary, C) reciprocal and D) neg. exponential), both axes represents also devices with thier id (see figure 11.9), gap=1s, learning window=30s, decay=.95

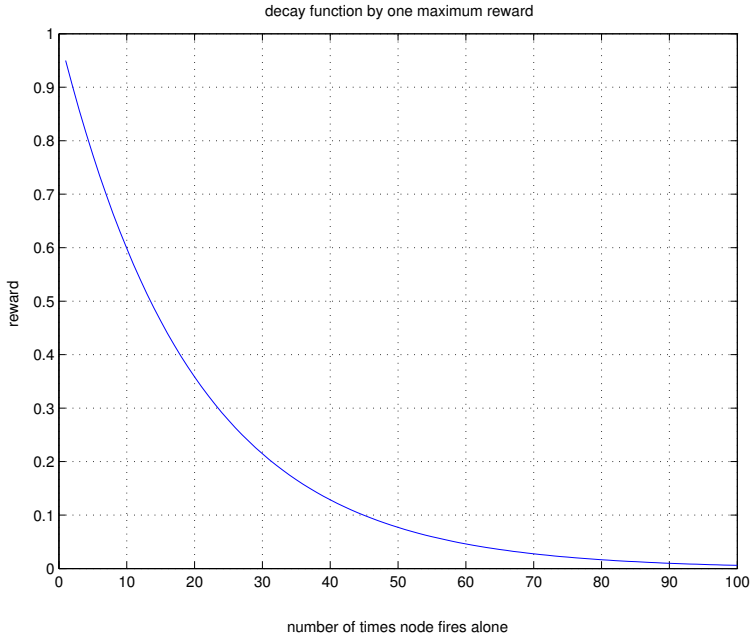

Figure 11.11: decay (95%) function: if a connection has a total reward value of 1.0, it takes approximately 100 firings where just one of the two devices is active until the weight decreases to zero.

11.3.3 Trend of connections over time

Because an intelligent room (which is controlled by AHA) has at the moment much more events than a normal *human-controlled* room, it is nessecary to analyse them separatly. Table 11.2 and 11.1 on page 41 shows how many times each device has fired.

The following figures show the trend of each connection between two node. Figure 11.12 shows the trends in human-controlled rooms and figure 11.13 shows the mean value over the last 38 days in an image diagram. In the image diagrams columns and rows are willful arranged that devices from the same room are neighbors because then all rooms are represented with rectangles around the diagonal. An optimal result would have along the diagonal some rectangles, which would represent the individual rooms.

Figure 11.14 and 11.15 shows the same, but this time also with the intelligent rooms. These two intellient rooms (room 74 and 84) have together 10 devices which are in the image 11.15 on the top left corner. It is evident that they have a much better mean correlation, because they have much more state updates.

Figure 11.14 shows also two connections which are very high, almost at the limit value. These are two blind pairs from room 55G74 and 55G84, which are always controlled together and changed quite often. This is the reason why they have such a high connection value.

All other connections are lower than 10, and most are even near zero, which is good visible in both images diagrams. And they show also, that all higher values are located around the diagonale, which means that they have a connections to a device inside its own physical room. Devices are almost independent between rooms, which is indicated with the dark blue color.

Problem: Presence Detectors

The last two image diagrams show clearly that the correlation between a presence detector and other devices in its room is not very high. We expect a much better value, but with the current sensors its very difficult to gain a real presence signal. First, they send a UNOCCUPIED event also if still someone is present, but is working more or less motionless and secondly they have a software bug. This leads to a constant toggle

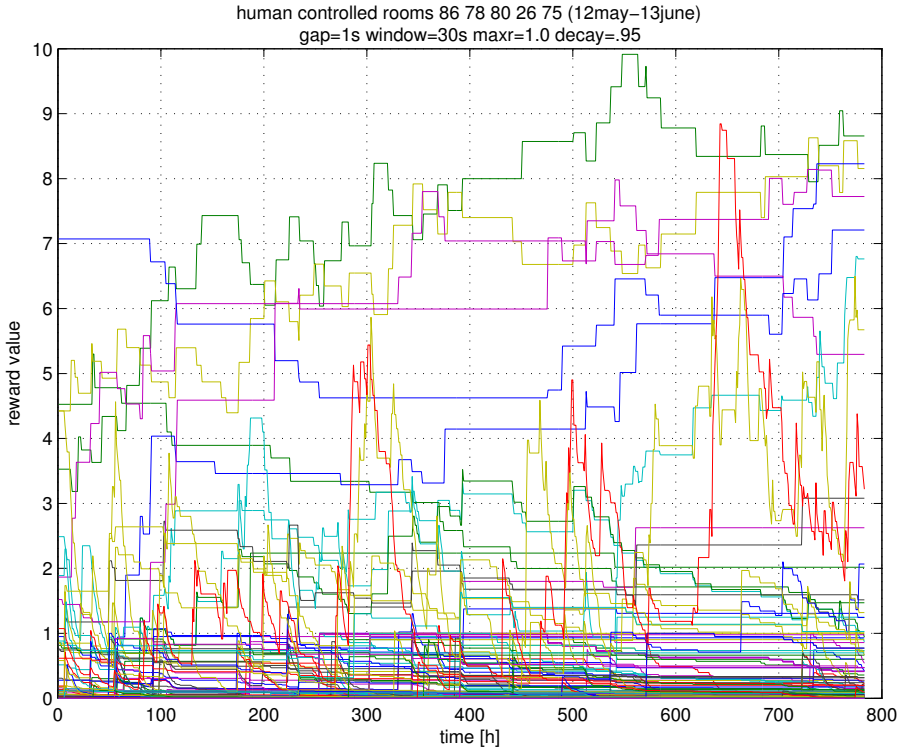

Figure 11.12: Human-controlled rooms (86 80 78 26 75), 12 May - 13 June 2003, gap=1000ms, neg. exp. reward func

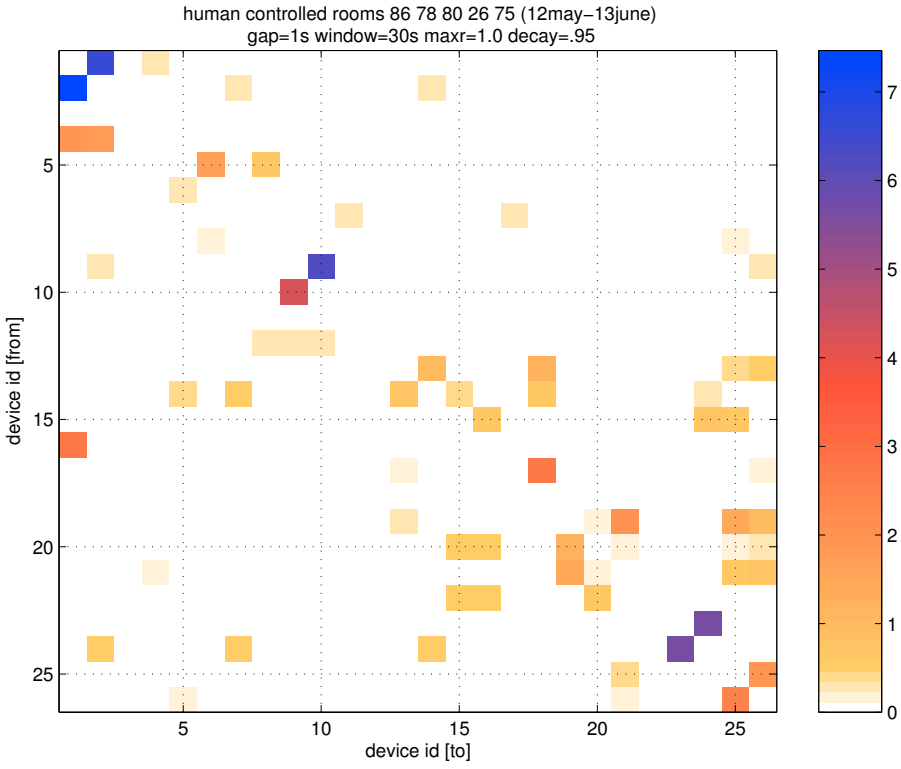

Figure 11.13: Human-controlled rooms (86 80 78 26 75), 12 May - 13 June 2003, gap=1000ms, neg. exp. reward func; Consult table 11.2 on page 41 for the Device ID translation (substract the id by 10).

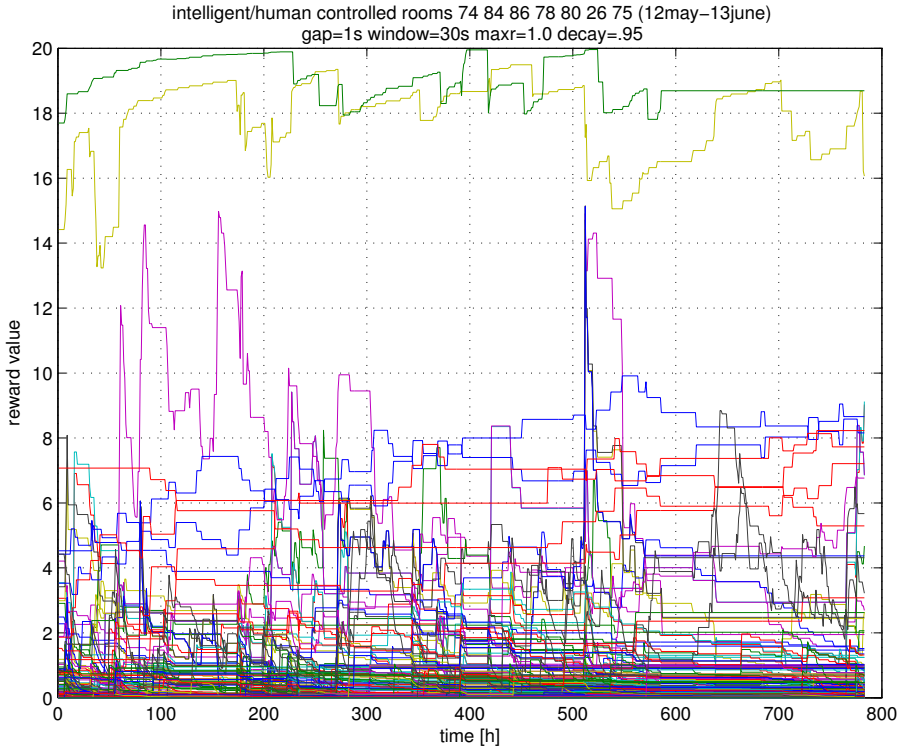

Figure 11.14: Intelligent- and human-controlled rooms (74 84 86 80 78 26 75), 12 May - 13 June 2003, gap=1000ms, neg. exp. reward func

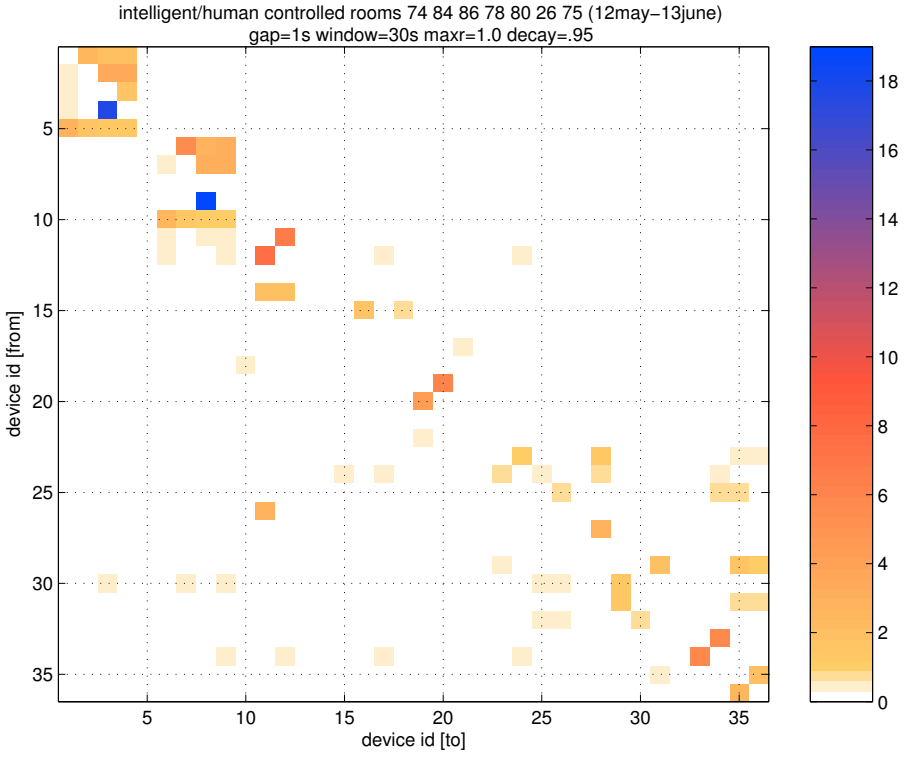

Figure 11.15: Intelligent- and human-controlled rooms (74 84 86 80 78 26 75), 12 May - 13 June 2003, gap=1000ms, neg. exp. reward func; Consult table 11.2 and 11.1 on page 41 for the Device ID translation.

stream with a period of ca. 90s.

And at last an intelligent room sets off all lights if nobody is present. A human-controlled room does not act similar (also because of the sw bug). These three points are almost impossible to kept in our filter. If the building system would run properly we expect better correlation between presence detector and other devices. Every morning when a person enters, he changes typically the state of the room (switch lights on, rise blinds, etc.) and even these sequences are difficult to obtain.

Blind pairs

We expect devices which are "hard-wired" by the Lonworks Software have a high correlation. Room 74 and 84 have blind pairs and each is controlled with one wall-switch. Figure 11.16 presents that this assumption is right. But it also shows a problem of the temporal order of events. Blind 1 and blind 2 in room 74 are controlled together and our system receives mostly first an event from blind 1 and immediately an event from the other blind. This sequence strengthens the connection between blind 1 and 2 but not contrary. However there are also other sequences where the system receives first an update from the second blind and then strengthens the connection in the other direction.

This was also a reason why we thought about a gap in the reward function. But to solve this problem for example the reward should be given to both directed connection. Because we transform this graph in a undirected before we partition it, this was not yet nessecary.

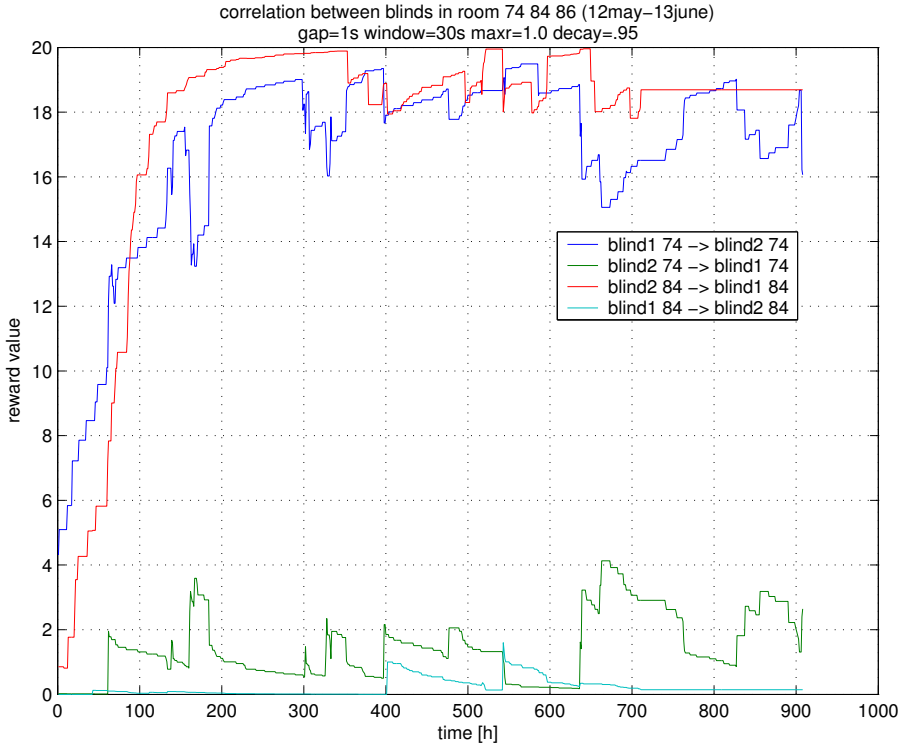

Figure 11.16: Blind pairs in rooms 74 84, 12 May - 13 June 2003, gap=1000ms, neg. exp. reward func. 312/342 are blinds from room 74, 327/329 from room 84.

11.3.4 Discussion

As illustrated in the last sections, we can determine the functional relationship between devices. But we also mentioned some problems, where a structure discovery can be difficult. Maybe there are also users, which behaves totally different as our assumption, and work in a environment without tough any effectors. This

makes it almost impossible to gain any reliable structure information, but do we need knowledge of these devices if they do not change over more than for example 4 days? Is it possible to control them in a further step? These are questions which have to be answered in future.

But even if we look at the mean correlation between devices in a controlled environment. Room 74, 84 and 86 have quite active devices and figure 11.17 shows this also clearly with three rectangles. The $5x5$ area on the top-left corner are the five devices from room 74, in the center is also a $5x5$ area with the devices from room 84 and on the bottom-left corner the $4x4$ area which represents room 86. Translation of the devices ids are defined in the tables on 41. Another interesting point which this figures illustrates is the influence of presence detectors to the other devices. Row 5, 10 and 14 represent these relationships.

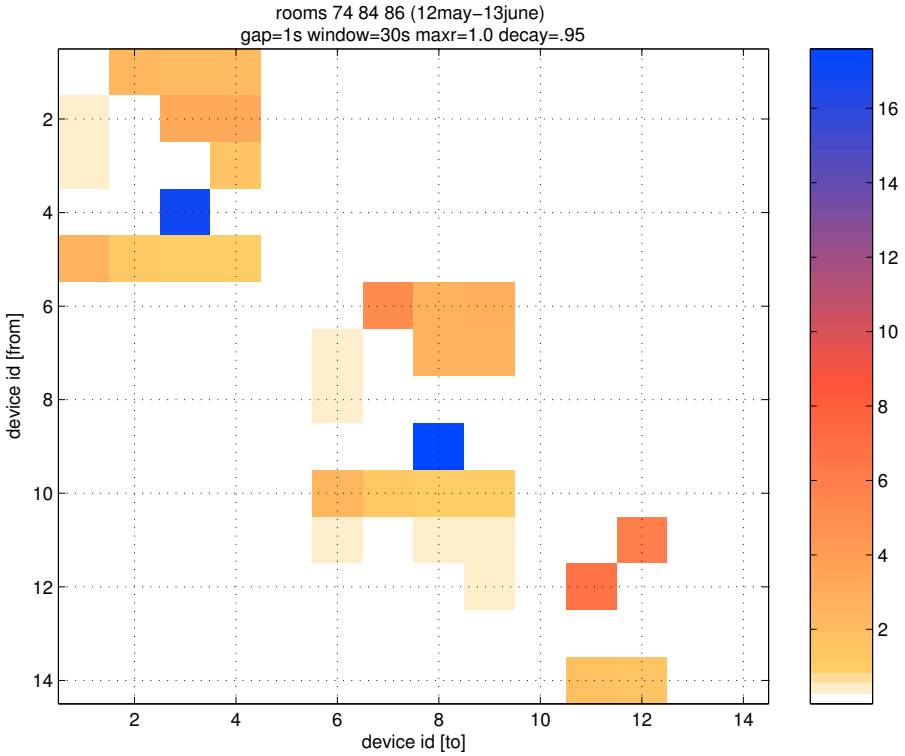

Figure 11.17: A closer look to the correlation mean in 74 84 86, 12 May - 13 June 2003, gap=1000ms, nexp. reward func.

We also think that the reward and decay function can be improved. If we would have more data, we think a better approximation of the reward function could be possible. A negative exponential function seams to deliver good results, but maybe the stretch factor should be resized.

Figure 11.18 illustrates our algorithm in a nice way. The lights in room 86 are switched on/off mostly together. But not always in the same order, which results that one relationship decreases and the other increases. But there are also situations where just one is switched on because the sum is below 20.

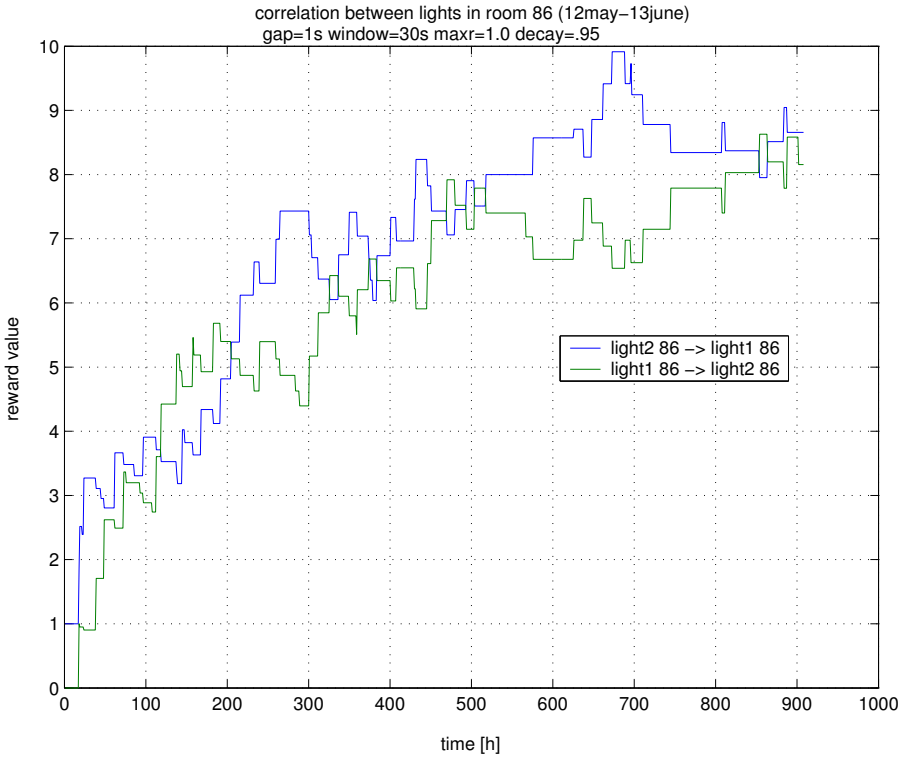

Figure 11.18: A nice illustration of our algorithm for example by considering the lights in room 86.

11.4 Clustering

11.4.1 Threshold

The normalized cut algorithm divides a set of devices into two subsets. This can be done recursively until each device is in its own set. Thus we have to define a threshold to stop any further partitioning. To determine this threshold we measured all second smallest eigenvalues while subpartitioning.
Figure 11.19 shows the histogram of this eigenvalues in AHA controlled rooms. It illustrates that there are many small eigenvalues between 0.0 and 0.2 and almost none between 0.2 and 0.4. Afterwards there are again more eigenvalues. Based on this result we interpret that the lowest eigenvalues separate all physical rooms in there own clusters and the much higher eigenvalues divide the rooms into subclusters. Human

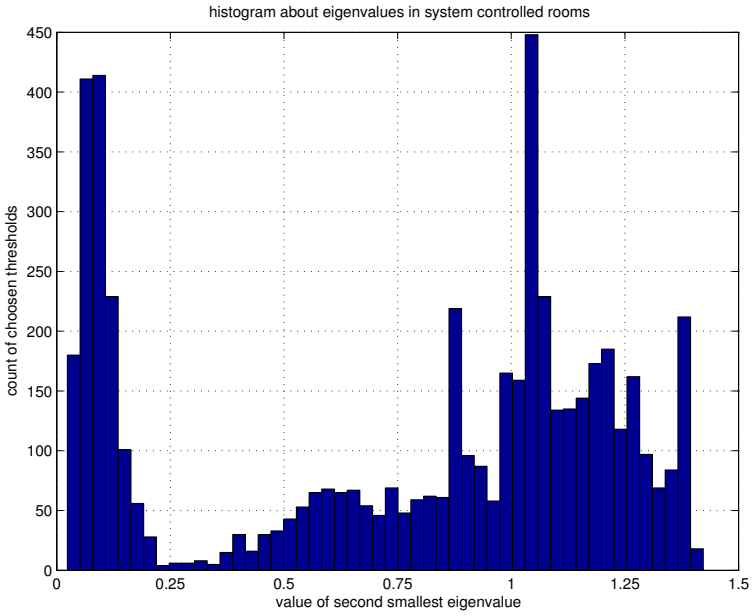

Figure 11.19: Histogram of all second smallest eigenvalue. These are gained by partition a graph recursively with devices from AHA controlled rooms.

controlled rooms are not as active as controlled rooms. Thus the association between the rooms are higher and the cohesion inside rooms lower. Figure 11.20 shows that there is not such a good threshold to determine like in controlled rooms. Inspired by figure 11.19 we defined threshold=0.6 and achieved good results. But we know that this threshold can vary from optimal one.

11.4.2 Eigenvalue and eigenvector

The normalized cut approximates the NP-complete problem to a generalized eigenvalue problem. The eigenvector of the second smallest eigenvalue delivers the partitioning solution. Each device is represented in this eigenvector as an element. In a discrete solution the elements of this eigenvector are 1.0 or -1.0. All devices with an element value of -1.0 are divided into a set A, all with +1.0 into set B.
But the calculated eigenvector is a continuous approximation. In most cases the allocation to a discrete solutions is distinct (figure 11.21). Whereas in figure 11.22 some devices are located around zero. At first glance it seems that devices six to nine build a third cluster but they have a light relationship among themselves. It seams that devices which are located near zero are not strongly related among themselves. Also it is not bad to allocate this group of devices to two other subsets. But this observation should constitute in further analysis.

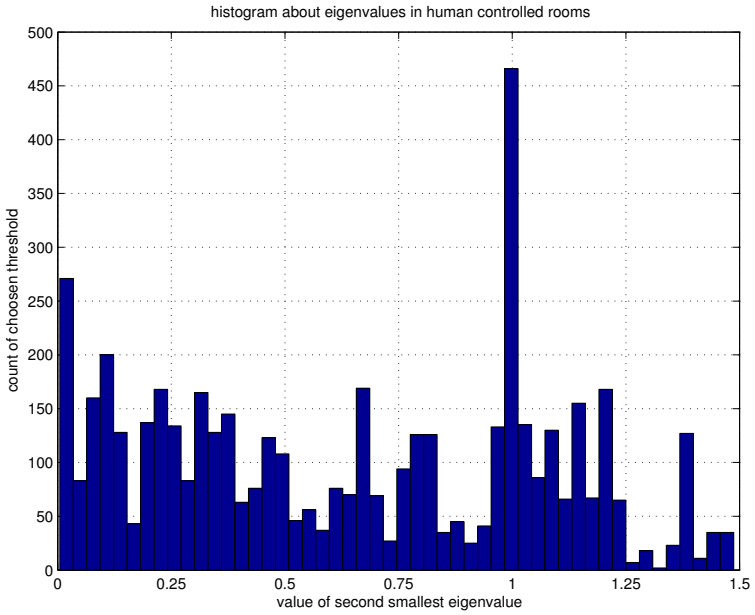

Figure 11.20: Histogram of all second smallest eigenvalue. These are gained by partition a graph recursively with devices from human controlled rooms.

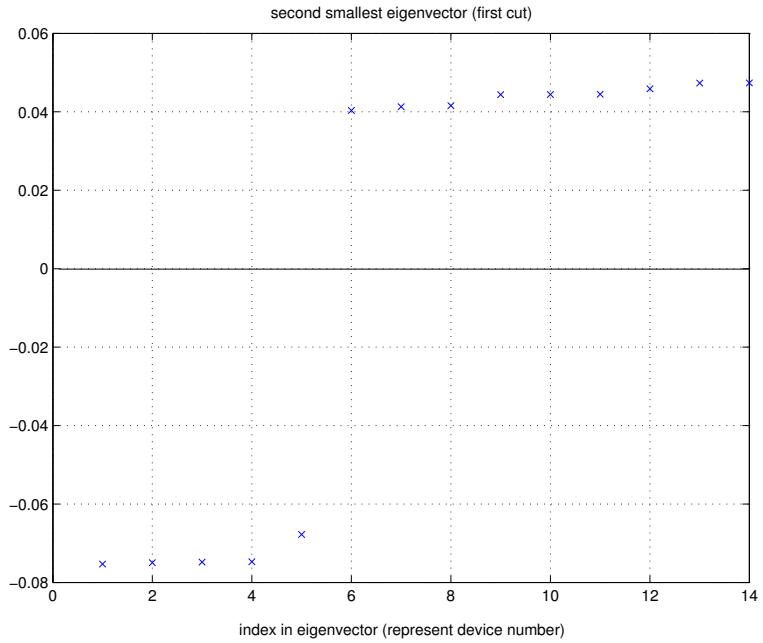

Figure 11.21: Cut by zero divide devices (x axis) in two groups

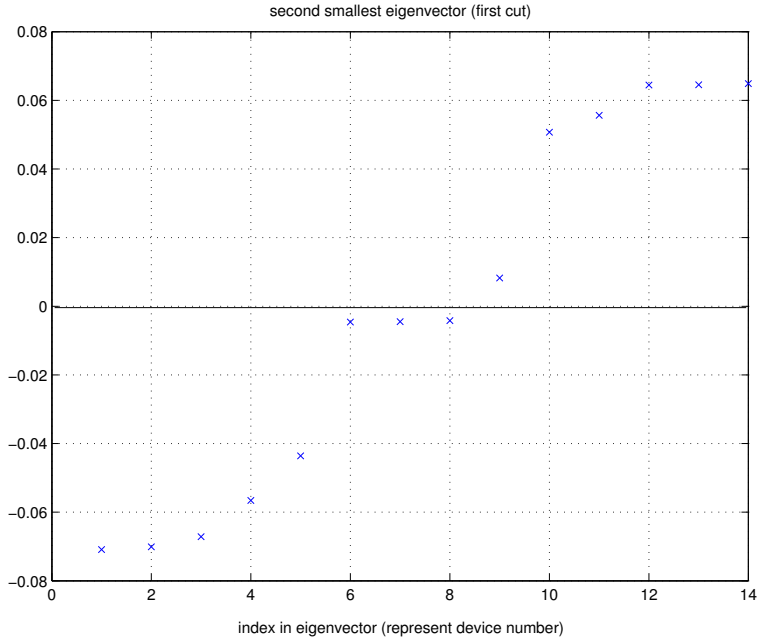

Figure 11.22: Few devices are arranged near zero. The zero cut splits this devices to the two clusters.

11.4.3 Quality of clustering

To measure the quality of the computed structure, we have to compare this with a predefined model. A thinkable model is the physical structure, which is also the most evident one. But there are also other logical models like for example a structure which clusters all blind on the east side of the building.
Although the normalized cut algorithm builds a hierarchical cluster tree (binary tree) we do not consider this information for analysing up to now (for reasons see chapter 5.3).

To compare the determined structure with the physical model we allocate each cluster to the most corresponding a room. We do this by analysing the number of devices in a cluster which belongs to the different rooms. A cluster gets allocated to the physical room with the highest number (figure 11.23). This assignment is the basis for our further analysis.

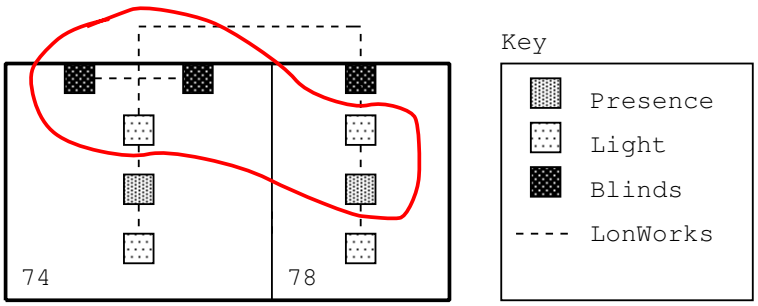

Figure 11.23: This cluster, indicated by the red line, belongs with 60% to the room 74 and with 40% to room 78. As described we regard this cluster as one which belongs to room 74.

Correctness of devices

Table 11.3 shows the correct allocations (in percent) for each device. Correct means that the device is in a cluster which is allocated to the correct physical room of this device. For example in figure 11.23 the device light1 in room 74 is correct allocated because the cluster also belongs more to physical room 74 than room

84.

Room Description	Devices Type	LonWorks ID	Correct Allocation (%)
Room 75	light1	60	100
	light2	61	91
	light3	76	100
	light4	77	92
	blind1	103	100
	blind2	105	100
	pd1	65	99
	pd2	67	80
Room 78	light1	250	100
	light2	251	100
	blind1	313	55
	pd1	255	65
Room 80	light1	258	100
	light2	259	100
	blind1	315	6
	pd1	263	84
Room 86	light1	266	100
	light2	267	100
	blind1	317	100
	pd1	271	100
Room 26	light1	180	93
	light2	181	60
	blind1	168	9
	blind2	170	7
	pd1	185	74
	pd2	189	100

Table 11.3: correctness of devices (only human controlled rooms)

Table 11.3 shows that almost all devices are allocated to the right room. But some devices (mostly blinds) are often allocated to other rooms (blind1-78, pd1-78, blind1-80, light2-26, blind1-26, blind2-26). Whereby the device light4-75 fires only seven times during 38 days.

Device Type	Fire Counter	Room26	Room75	Room78	Room80	Room86
blind1-80	50	8.7%	28.7%	55.2%	5.7%	1.6%
blind1-26	11	8.8%	91.2%	0.0%	0.0%	0.0%
blind2-26	33	7.3%	37.0%	0.0%	0.0%	55.7%

Table 11.4: List of allocation of perverse devices

Table 11.4 shows for these devices how strong they are allocated to other rooms. We see that the blind1-80 is allocated most of the time with devices from the room 78 and 75. If we look into the reward graph we can detect the relevat device-device combinations for this behavior (figure 11.24). While the first 180 hours it looks correct, the highest reward are inside of the room 80. But this timepoint the blind1 in room 78 gets the maximum of reward by mischance. Also the blind1-75 triggers at the timepoint 250 hours a reward. Although this rewards decrease with every event from the blind1-80, they are higher than inside correlations for a longer time. Therefore the blind1-80 seems to be more releated to the blind1-78 and blind1-75 for the remaining time although this rewards are accidentally.

An interesting aspect is that devices with the three lowest allocations are blinds. It looks that our reward function does not reward correlations for blinds good enough. But we are positiv that our reward function can detect relationships between the correct presence detector as soon as the software bug of presence detectors is fixed.

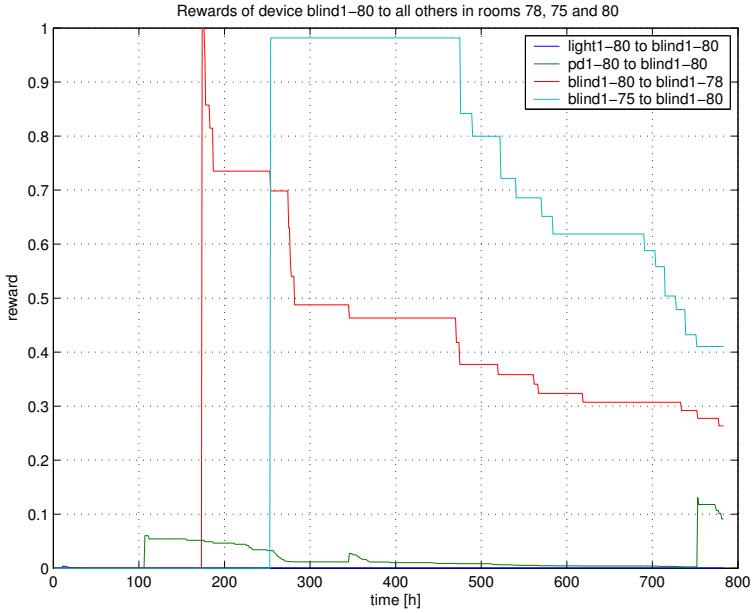

Figure 11.24: All reward weights of devices in rooms 78,86 and 80 from and to device 315 which are greater than zero.

Quality / clarity of clusters

A cluster should be allocated to the corresponding right physical room as well as only containing devices from the same room. For instance a cluster with five devices have a quality of 4/5 (80%) if four devices are from the allocated physical room. Table 11.5 shows the quality of clusters for each physical room.

Room Number	Clarity of Allocated Clusters (%)
86	89%
78	77%
80	75%
26	93%
75	83%

Table 11.5: quality of builded clusters (only human controlled rooms)

Also we see that the recognized clusters contains almost only devices from one physical room. More than 50 percent of clusters includes only devices from the same physical room (11.25).

Medium number of clusters for physical rooms.

And now we want the answer in how many clusters gets a room divided. This is one of the most interesting aspects of the clustering results. Table 11.6 illustrates the medium number of clusters. These values mean for each room bases also on the allocation of clusters to a physical room (described in the introduction of 11.4.3).

Most small rooms are divided into one or temporary more clusters. The big office (room 75) is splitted into nearly three subclusters. Histogram 11.26 shows the probability of different cluster sizes. Clusters which

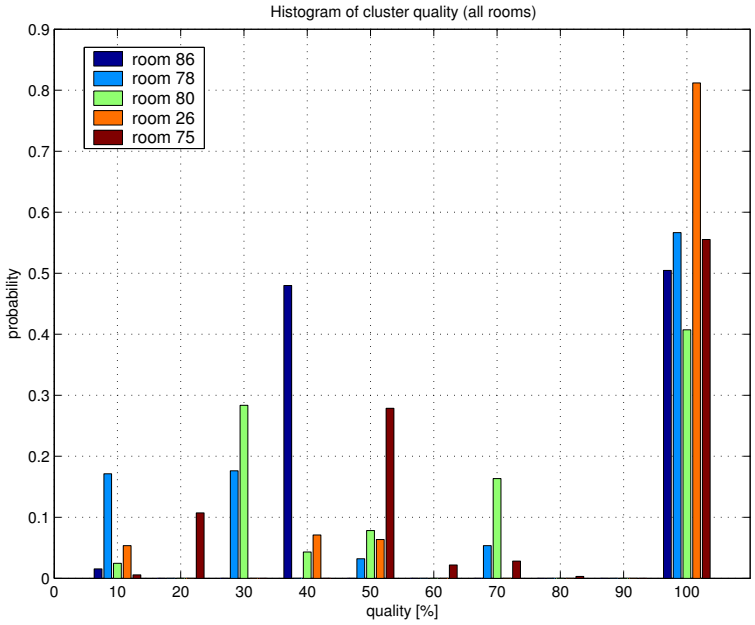

Figure 11.25: Distribution of cluster quality for each room.

Room Description	**Medium number of cluster for physical room**
Room 86	1.0
Room 78	1.53
Room 80	1.36
Room 26	1.20
Room 75	2.87

Table 11.6: Medium cluster size grouped by rooms

includes only two devices contain mostly light or blind pairs. It also illustrates that clusters mainly include less than four devices.

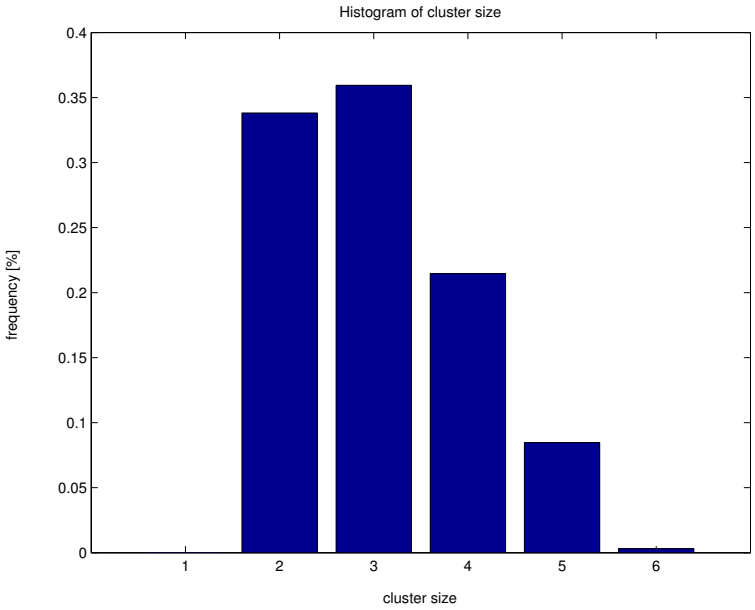

Figure 11.26: Distribution of cluster size of all built clusters.

Part VI

Conclusion and Future Work

Chapter 12

Conclusion

We proved that it is eventually possible to determine a very detailed functional structure from the data available to the system. The result chapter demonstrates that our approach discovers a functional structure from very sparse data which matches almost perfectly with the physical model.

Small rooms equipped with about four devices are determined as being independent, even if they contain devices which have just one update every fourt day on average. But we have seen also, that our assumption can not be valid in general within a small timeframe of 40 days. There are some devices which are rarely active and do not strengthen their relationship with others because of this. Additionally there are still presence detectors which do not operate correctly and because of this we have to filter incoming events. Filtering is not lossless and we thus also lose some data due to this. If the presence detectors would run properly, especially rarely active blinds could potentially strengthen their relation to a presence detector and would be grouped into the correct cluster.

Bigger rooms within which more people are working are divided into subrooms. But up to now we can not prove this in general, because we only have real data from one such room. And also to gain there better results we think they have to be equipped with more sensors (especially the presence detectors have to be more sensitive).

Nevertheless our approach is capable to discover the functional structure in a pretty robust way. And this even in an environment with real data of a real building equipped with a variety of sensors and effectors, which can be damaged or behave weired.

Another advantage of our approach is that it is able to compute the realtionship between devices locally and thus in parallel. The partitioning can be triggered event-driven.

Chapter 13

Future work

Here we deal with possible extentions of our approach and also possible future directions for research about intelligent buildings in general.

13.1 Include event values and analog devices

As this document illustrated, we consider up to now just devices which have a digital output value. We do not analyze the value of their update while their are sending an event over the fieldbus. It could be possible to gain a more detailed reward graph, if we would also regard the actual value of the events. But we think this would make the graph larger and even more complicated for partitioning.

A more reasonable issue is to include first the analog devices in our discovery of structure. But, unlike the digital one, they are sending much more events because e.g. the illumination changes on average slowly but continuously. This analog value should be quantified somehow, for example by decreasing a new reward with the delta of its notified state value: $r = r * \Delta s$ where $\Delta s = s_{i+1} - s_i$ is the delta between the new and old state value.

13.2 Different reward functions

We could prove our assumption in most cases, but there are a few devices which possibly never strengthen their relationship with others. It is thinkable to have more than one reward function. One could for example just strengthen sequences where two devices are active together at the same time, and another could reward sequences which have a bigger Δt. The first one would obviously strengthen the weights with a higher reward than the second can ever do.

13.3 Deploy dynamic structure information to multi-agent-system

Beside to get a better understanding about the dynamic structure of our intelligent building, it is also the objective to deploy this knowledge to the learning units. They should use this information to control just this cluster which they are associated with. And if there is a new device inside of its cluster, it should learn how this one can be involved in the rulebase and eventually also controlled.

If a device belongs no longer to a specific cluster, the learning unit should notice this and break off to control it.

13.4 Policy transfer

If structure adapts to a new state, especially the control agents have to be informed or have to request the new structure. If a device belongs no longer to a specific cluster it is assigned to another or was removed from the dedicated fieldbus. When the device is now just in another cluster, two agents have to notice this. The one which has lost the device, and the other which has gained it. To be able to control this new device, the second agent has to collaborate with the first one. They exchange their learned knowledge about one or a set of devices. But how this challenge can be solve is still an open issue.

Part VII

Glossary and Bibliography

Bibliography

[ABL] Agent building and learning environment (able). http://www.alphaworks.ibm.com/tech/able.

[BDY99] Magnus Boman, Paul Davidsson, and Håkan L. Younes. Artificial decision making under uncertainty in intelligent buildings. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 65–70, 1999. ftp://ftp.dsv.su.se/users/mab/uai99.ps.

[Bol96] J. Bollen. Algorithms for the self-organisation of distributed, multi-user networks, 1996.

[Bro91] Rodney A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

[BSP+02] J. P. Bigus, D. A. Schlosnagle, J. R. Pilgrim, W. N. Mills III, and Y. Diao. Able: A toolkit for building multiagent autonomic systems. *IBM Systems Journal, Applications of Artificial Intelligence*, Vol 41(Nr.3):350–371, 2002.

[Col] Open source libraries for high performance scientific and technical computing in java. http://hoschek.home.cern.ch/hoschek/colt.

[EIB] European installation bus. http://www.eiba.com.

[EKB+03] K Eng, D Klein, A Baebler, U Bernadet, M Blanchard, M Costa, T Delbruck, R J Douglas, K Hepp, J Manzolli, M Mintz, F Roth, U Rutishauser, K Wassermann, A M Whatley, A Wittmann, R Wyss, and P F M J. Verschure. Design for a brain revisited: The neuromorphic design and functionality of the interactive space ada. *Reviews in the Neurosciences*, 2003.

[FIP02] Fipa abstract architecture specification (xc00001k). Technical report, Foundation For Intelligent Physical Agents, Geneva, Switzerland, 2002.

[FLoA02] Inc. Fujitsu Laboratories of America. Jas agent services (jsr-87) specification. http://www.java-agent.org, 2002.

[GK] Wulfram Gerstner and Werner M. Kistler. Mathematical formulations of hebbian learning.

[GK02] Wulfram Gerstner and Werner M. Kistler, editors. *Spiking Neuron Models – Single Neurons, Populations, Plasticity.* Cambridge University Press, August 2002. ISBN : 0-521-81384-0.

[Heb49] Donald Hebb, editor. *The Organization of Behavior.* Wiley, New York, 1949.

[JMF99] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[L4J] Log4j logging framework. http://jakarta.apache.org/ant.

[LON] Lonworks networking platform. http://www.echelon.com/products.

[MvRT00] G.Q. Bi M.C.W. van Rossum and G.G. Turrigiano. Stable hebbian learning from spike timing-dependent plasticity. *The Journal of Neuroscience*, Vol 20(8812-8821):8812–8821, 2000.

[RS02a] Ueli Rutishauser and Alain Schaefer. Adaptive building intelligence – a multi-agent approach. Technical report, University of Applied Sciences Rapperswil, Switzerland and Institute of Neuroinformatics, Swiss Federal Institute of Technology, Zurich, Switzerland, 2002.

[RS02b] Ueli Rutishauser and Alain Schaefer. Adaptive home automation – a multi-agent approach. Technical report, University of Applied Sciences Rapperswil, Switzerland and Institute of Neuroinformatics, Swiss Federal Institute of Technology, Zurich, Switzerland, 2002.

[SM00] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[SMA00] S. Song, K. Miller, and L. Abbott. Competitive hebbian learning through spiketime-dependent synaptic plasticity. *Nature Neuroscience*, 3:919–926, 2000.

[VC00] Graham Clarke Victor Callaghan. A soft-computing dai architecture for intelligent buildings. Technical report, Department of Computer Science, University of Essex and Department of Computer Science, University of Hull, 2000. http://cswww.essex.ac.uk/intelligent-buildings/publications/springerverlag.pdf.

[Wei99] Gerhard Weiss, editor. *MULTIAGENT SYSTEMS, A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge, Massachusetts, 1999. ISBN : 0-262-23203-0.