Approaching Decentralized Demand Side Management via Self-Organizing Agents

Christian Hinrichs Ute Vogel Michael Sonnenschein
Dept. of Environmental Computer Science, C.v.O. University of Oldenburg, Germany
{hinrichs,vogel,sonnenschein}@informatik.uni-oldenburg.de

ABSTRACT

This paper provides an introduction to self-organizing mechanisms in the domain of energy systems. We motivate the use of such mechanisms by outlining the drawbacks of present control systems and providing some advantageous properties of self-organizing systems. We address the problem of a supply and demand matching of a large number of distributed actors and give a mathematical formalisation of this optimization problem. Based on that we propose the algorithm DSS which uses the stigmergy paradigm to form a distributed search heuristic. Afterwards the DSS-T is introduced, a refinement of the DSS algorithm which incorporates mechanisms from the tabu search. To examine the performance of the algorithms, we implemented a simulation framework which lead us to the problem of local mutual exclusion with fairness. Therefore we describe this problem and its origin and then propose the ARP, a protocol which guarantees local mutual exclusion, fairness and starvation freedom.

Finally, we give some first insights into the performance of the DSS and DSS-T algorithms by examining a single simulation scenario for reference.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

Algorithms

Keywords

self-organization, emergence, distributed heuristics, demand side management, supply and demand matching

1. INTRODUCTION

The term *Demand-Side Management* (DSM) [4] appeared in the last third of the past century. First research focused on manually controlling large electrical loads (like industrial plants) in order to compensate stress in the power grid. On the one hand, these methods incorporated direct control signals which allowed immediate control over the loads. On the other hand, indirect signals (i.e. dynamic pricing) have been studied. These form incentives for appliance owners to regulate their energy use. This was followed by methods to sense and react to stress situations in the grid automatically. For an overview of these strategies the reader may refer to [5, 2, 8]. However, these approaches were centrally organized and not suitable for a large amount of individually controllable units.

Therefore, strategies of decentralized control have been investigated. Intelligent appliances which autonomously monitor properties of the power grid (i.e. frequency or voltage) and independently react to stress situations have been proposed and tested [19, 6, 17]. In these approaches however, the lack of coordination lead to oscillation effects due to synchronization of appliance reactions. So there is a need for communication to allow for intelligent coordination. Some work has been done to address the oscillation problem [9]. Though this – to a certain extent – can be done in a centralized fashion, a more scalable and elegant way would be given by a completely decentralized control and coordination scheme. This requires the appliances being able to autonomously act in a system-stabilizing way.

But how can we design such a distributed artificial intelligence? We investigated this question from a nature-driven perspective. In nature there are many systems that fulfill a specific task without central coordination (e.g. the foraging of ants). Those systems reveal a number of advantageous properties like adaptivity, self-healing, scalability, redundancy and more [16]. If we now examine the above question from a more abstract point of view, we come to the following conclusion: In a decentralized DSM-enabled system, a large number of agents perform autonomous actions and form a collective behaviour to fulfill a global objective. Such a system is called *self-organizing*.

In this paper, we present our ongoing work on self-organization and emergence in energy systems. We believe that such nature-inspired algorithms and models can effectively be applied to the task of stabilizing the electricity grid, i.e. by providing balancing power to compensate fluctuating energy resources. The paper is organized as follows: Section 2 describes the concept of self-organizing systems in detail. In section 3 we present related attempts (own and other) in this area. Section 4 then formalizes the addressed optimization problem of a supply and demand matching and proposes DSS and DSS-T, first approaches to self-organizing algorithms for this problem which form distributed search

Cite as: Approaching Decentralized Demand Side Management via Self-Organizing Agents, C. Hinrichs, U. Vogel and M. Sonnenschein, *2nd International Workshop on Agent Technologies for Energy Systems (ATES 2011), 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Rogers, Decker and Kok (eds.), May, 2, 2011, Taipei, Taiwan.

heuristics based on the stigmergy paradigm. Afterwards, in section 5 the simulation framework and the activity restriction protocol ARP are introduced before finally some tentative interpretations of the performance of the proposed algorithms are given.

2. SELF-ORGANIZATION AND EMERGENCE

There is no common definition of self-organization. In biology, a self-organizing process produces a global pattern purely on a basis of local actions, without knowledge of the global pattern or system [1]. In the field of multiagent systems there is a distinction between self-organization and *emergence*. In [16] the former is defined as a mechanism or process which allows a system to (re)organize during runtime without explicit external instruction. The latter is described as a phenomenon that becomes visible at the global (macro) level of the system through interactions of elements at the local (micro) level. This phenomenon can be a structure, behaviour or functionality of the system. Note that emergence is not an essential property of self-organizing systems. However, engineering such systems often aims at the emergence of specific properties, leaving self-organization being a tool to make emergence possible. In the following, we will use the term self-organization synonymously for self-organizing systems that exhibit emergent properties.

Derived from the definitions above, self-organizing systems consist of at least two distinct activity levels. There is the macro level, which is an observation of the system from an external point of view. The individual system elements may be visible, but their internal processes are unknown. Observing a school of fish, for example, one can perceive the overall structure of the system, as well as its behaviour. That is, it forms a swarm which consists of several fishes. The swarm moves around, with no identifiable central leader to coordinate the movement. It is able to evade obstacles and enemies and may even split up into multiple parts without losing its abilities. These observable characteristics are the emergent properties of the system. To understand the origin of those, we must take a view into the system and inspect the micro level. That consists of fishes with individual properties (i.e. position, direction, velocity) and behaviours. Regarding the fishes as agents, the behaviours are expressed as rules which modify the agent's properties. The rules are *local*: they consider only the properties of the agent itself, and those of nearby agents. That is important, because it implies that no agent is able to perceive the whole system (and therefore: the macro level). The term "nearby", however, has to be defined specifically for each system. The behaviour of swarming animals like fishes, birds and gregarious land animals has been studied in [15]. The author has identified three principal rules that, when simulated in an artificial system, lead to the formation of swarms:

1. Collision Avoidance: avoid collisions with nearby objects.
2. Velocity Matching: attempt to match velocity and movement direction with nearby individuals.
3. Swarm Centering: attempt to stay close to nearby individuals.

Applying these simple behavioral rules (with descending priority) to a number of simulated fishes in parallel, they will form a swarm with the above characteristics. Note that none of these rules takes the whole system into account. But since the action of any indivdual *triggers* reactions of other individuals, a global behaviour emerges. So there is a mechanism of self-organization: every single individual acts on behalf of its rules, aiming for a local equilibrium (the fulfillment of a local goal, here rather: of all its rules). In a school of fish, an individual reaches its equilibrium condition if all its neighbours are equally far away positioned and have all the same movement direction and velocity as itself (see figure 1). In the next timestep however, this condition will be broken, because due to the third rule the outer fishes will try to move towards the swarm center, disturbing the equilibrium of the centered fish.

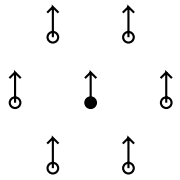

Figure 1: Equilibrium condition of the centered individual in a school of fish. Arrows represent direction/velocity vectors.

The example "school of fish" shows the following: There are individuals at the micro level of the system, acting upon simple local rules and each striving after an equilibrium (fulfillment of the local goal). If all individuals were in equilibrium condition, the system would have organized itself. But as soon as an external disturbance occurs, the individuals would try to fulfill their local goals again, effectively reorganizing the system. Since this reorganization occurs without external guidance, it is called self-organization. The emergent properties of the system are either produced by the reorganization process itself (as in the school of fish), or by reaching the equilibrium state. An example for the latter would be the termite nest-building. Here, the emergent property is the optimal protection of the termite queen. This is reached by building the nest equally around the spatial position of the queen. If the queen moves its permanent residence, the worker termites start to rebuild the nest so that the queen will afterwards again be positioned centrally in the nest. Upon reaching this goal, the system has reorganized and will again optimally protect the queen. This example shows an additional aspect: An internal *central* entity in a self-organizing system is allowed, as long as this entity only triggers the self-organization process but does not control the individual actions.

Self-organizing systems can be applied to a number of tasks, especially optimization problems. For a classification of self-organization mechanisms and possible applications, the interested reader may refer to [7, 12].

3. RELATED WORK

In the domain of energy systems, the concept of *virtual market places* has intensively been discussed in recent years. In those systems, electrical loads (consumers) as well as energy resources (producers) are actors at a virtual market place. There they offer or demand blocks of electrical power. The market clearing is reached by adjusting prices and therefore matching the offered and demanded amounts. Because

the actors react differently and autonomously to price adjustments, the system is partly self-organized. The emergent property is a *supply and demand matching* of electrical power. Two of such systems are described in [10] and [20]. Because of their statical hierarchical design however, market place systems are also centrally organized.

Another partly self-organized approach has been proposed in [11]. There, communication is carried out indirectly via a global black board. A central broker agent publishes the global optimization goal, while each other participating agent (producers and consumers) publishes its predicted operational mode (amount of electrical power demanded/supplied). These agents will then iteratively modify and publish their operational modes according to the global goal, published information from other agents and local degrees of freedom, until the system converges or is stopped by a special condition. The modification of local operational modes is carried out via a genetic algorithm.

In [14], a system based on *overlay trees* has been proposed. In such a system, the agents organize themselves in a virtual tree structure and perform an iterative bottom-up supply and demand matching. For that task, each agent sends its degrees of freedom (in terms of possible operational modes) to its parent node in the tree. The parent node then selects the locally optimal operational modes and sends the selection back to the children, who have to adhere to the selected modes. These actions are performed at each level of the tree, until the root node has been reached. Coupled with a dynamically built tree structure, this system could be able to emerge an optimal global solution.

In [18] the concept of *holonic virtual power plants* has been studied. The idea is somehow related to the virtual tree structures. A holon can be composed of a single or more agents and even other holons. To the outside, the holon acts like a single agent. Therefore, the terms "agent" and "holon" can be used synonymously here. Additionally, a holon can be part of several holons at the same time. The structure is hierarchical, but overlapping and highly dynamic. In conjunction with distributed scheduling mechanisms, the author proposed a holonic system which is able to fulfill a global optimization goal (i.e. building a virtual power plant) in a self-organizing way.

4. APPROACHING SELF-ORGANIZATION

Most of the presented work in the previous section incorporated some kind of centrality or static organization. In this paper we propose a completely decentralized, highly dynamic and self-organized system of autonomous agents who work cooperatively on local goals in order to emerge a solution to a global optimization goal. For that objective, we presume that the monetary outcome of the system will be distributed equally among the agents. Hence, we are modelling agents which have no interest in competing with each other. Instead, they try to maximize their profit by cooperation. In the following section, we construct a formal model of the considered problem.

4.1 Formalisation

4.1.1 The optimization problem

Let $g : \mathcal{T} \to \mathbb{R}$ be a timeseries of amounts of electrical power. Time is discretized into time slots $t \in T$. The set of agents can be described by $\mathcal{A} = \{a_i \mid 0 \leq i < n\}$ with n being the number of agents. Each agent represents an actor which has the ability to adapt its mode of operation for each time slot. These operational modes therefore describe an either positive (producer) or negative (consumer) amount of electrical power. Additionally, they are rated by the agent with respect to how beneficial it is to choose the specific mode. For example, a CHP might rate its possible operational modes with respect to their efficiency. According to these ratings, the agents cannot be purely altruistic, but rather benevolent while being constrained by their device specific properties. So the set of rated operational modes for an agent a_i for a time slot t is defined as

$$\mathcal{P}_i(t) = \{(p, r)_k \mid 0 \leq k < m_i\} \tag{1}$$

where $p = p_{i,k}(t)$ is an operational mode, $r = r(p_{i,k}(t))$ is the rating for this mode and m_i is the number of possible modes for this agent in the given time slot.

The global goal of the system now is to select an operational mode for each agent for each time step so that a given load profile g is approximated as closely as possible. For the time being, we will neglect any constraints between operational modes.

A vector $s(t)$ of operational modes (one for each agent) with associated ratings for a time slot t is called a solution vector and is defined as

$$s(t) = (y_0, \ldots, y_n)\,, \quad y_i \in \mathcal{P}_i(t) \tag{2}$$

The matrix $s(\mathcal{T})$ then aggregates $s(t)$ for all $t \in \mathcal{T}$ and is called a *solution*. Now assume a rating function f which rates a solution vector $s(t)$ of selected operational modes for a specific time slot, and a second rating function F that aggregates the ratings given by f over time, effectively rating the solution $s(\mathcal{T})$. An example for f could be

$$f\left(g(t), s(t)\right) = \overline{r}_{s(t)} \cdot \left| \left(\sum_{p \in s(t)} p \right) - g(t) \right| \tag{3}$$

where

$$\overline{r}_{s(t)} = \frac{\sum_{r \in s(t)} r}{n} \tag{4}$$

is the mean of all ratings in $s(t)$. So f computes the absolute value of the difference between the superimposed operational modes and the load profile at the given time slot, and multiplies that with the overall rating. Then an example for F could be

$$F(g, s) = \sum_{t \in \mathcal{T}} f\left(g(t), s(t)\right) \tag{5}$$

so that F provides the sum of the ratings of f over time.

To find an optimal set of operational modes for a given load profile g, a solution s_{opt} has to be identified that minimizes F:

$$s_{opt} = \underset{s}{\operatorname{argmin}} \left(F\left(g, s\right)\right). \tag{6}$$

The solution space to this problem is given as

$$\mathcal{S} = \bigtimes_{t \in \mathcal{T}} \bigtimes_{i \in \mathcal{A}} \mathcal{P}_i(t) \tag{7}$$

yielding

$$|\mathcal{S}| = m^{n \cdot |\mathcal{T}|} \tag{8}$$

if we assume $m = m_i \; \forall i \in \mathcal{A}$.

```
repeat:
   if local environment changes:
      for each considered time slot:
         adapt own selection
      if selection has changed:
         publish new selection
```

Figure 2: DSS – A distributed search algorithm based on the stigmergy paradigm.

4.1.2 Communication

In a self-organizing system, agents act upon their local perception. In the present case, we model this perception via communication between agents. Hence, agents can be *connected* to other agents, which means they know each other and can communicate. Agents without a direct connection cannot communicate with each other. So the communication subsystem can be expressed as an undirected graph

$$G = (\mathcal{A}, \mathcal{E}) \tag{9}$$

where the set of vertices is the set of agents $\mathcal{A}$ and the edges $\mathcal{E}$ are communication links. Now, as suggested in section 2, we may define the term "nearby" as a kind of *locality*: Each agent a_i has a defined limited neighbourhood $\mathcal{N}_i$ of other agents with whom it can communicate. So $\mathcal{N}_i$ is defined by

$$\forall a_i \in \mathcal{A}\ \forall a_j \in \mathcal{N}_i : (a_i, a_j) \in \mathcal{E},\ \mathcal{N}_i \subseteq \mathcal{A},\ a_i \neq a_j. \tag{10}$$

Additionally

$$(a_i, a_j) \in \mathcal{E} \Rightarrow (a_j \in \mathcal{N}_i \wedge a_i \in \mathcal{N}_j) \tag{11}$$

guarantees that communication links are bidirectional.
Considering locality, we may introduce the *local solution vector* $s_i(t)$ of an agent a_i which reflects the local view of the agent at the system for a specific time slot t. It is defined as

$$s_i(t) = (y_{i_0}, \ldots, y_{i_l})\ , \quad y_{i_j} \in \mathcal{P}_j(t)\ , a_j \in \mathcal{N}_i \cup \{a_i\} \tag{12}$$

and is the subset of $s(t)$ which is currently visible to a_i with respect to its neighbourhood (including its own selection). Accordingly, $s_i(\mathcal{T})$ is the *local solution* visible to a_i.

4.2 Self-organizating algorithms

4.2.1 A first approach

As a first attempt to a completely distributed and self-organizing solution for the given problem, we have designed an algorithm which incorporates the *stigmergy* paradigm. In this paradigm, coordination is carried out via monitoring and modifying the environment rather than direct agreements between agents [12]. For the given case, we let each agent follow the *Distributed Stigmergy Search* (DSS) algorithm shown in figure 2. The local environment of an agent a_i here corresponds to the local solution without its own selection: $s_i(\mathcal{T}) \setminus \mathcal{P}_i(\mathcal{T})$. So if any of the neighbours changes any of its selected operational modes, the agent a_i will adapt its own selection to the new circumstances. To accomplish that, a_i selects those own operational modes (one for each time slot) which yield the best rating for the current local solution. This corresponds to a local search which yields

$$s_{opt_i} = \underset{s_i}{\operatorname{argmin}} \left(F\left(g, s_i\right)\right)$$

```
repeat:
   if local environment changes:
      for each considered time slot:
         sort op-modes by estimated improvement
         find first element s in the sorted list
                     which is not in the tabu list
         select s and add to tabu list
         if size of tabu list > l:
            remove oldest element from tabu list
      if selection has changed:
         publish new selection
```

Figure 3: DSS-T – Including tabu search in DSS.

(cmp. eq. 6), where only the components of s_i which belong to a_i are modified. If the new selection differs from the former, the agent then publishes that new selection, thus triggering activity in its neighbours.

In this mechanism, agents are in equilibrium condition if they are not able to improve the local solution by selecting other operational modes. If the whole system is in equilibrium condition, each agent has reached a pareto optimum in its local search space. Unfortunately this doesn't mean that the global solution $s(\mathcal{T})$ is a pareto optimal solution, too. For example consider a system $\mathcal{A}$ consisting of three agents connected in a row so that $a_0 \in \mathcal{N}_1$, $a_1 \in \mathcal{N}_2$ but $a_2 \notin \mathcal{N}_0$. Let $\mathcal{A}$ be in equilibrium condition, so no agent may choose an operational mode which improves its local solution. However, as a_0 and a_2 are not connected, there might be operational modes which would in fact improve the global solution of the system while impairing the local solution. Hence, the attraction of these local pareto optima of each agent rapidly leads to global equilibium condition and therefore a termination of the algorithm in a suboptimal state.

4.2.2 Including tabu search

Therefore, our work aims at providing a distributed heuristic that is able to generate "good" solutions in a short amount of time, providing all advantages of self-organizing systems but without guaranteeing a specific solution quality. Such a mechanism should in the long term converge to an optimal solution s_{opt}, while allowing temporary deteriorations like the famous *tabu search* algorithm in order to explore the whole search space. We incorporated the ideas of tabu search into the algorithm DSS-T by providing each agent with a local tabu list with a maximum length l_i (see figure 3). In this algorithm, for each time slot t an agent first calculates a list of its possible operational modes, sorted by their estimated improvement. It then selects the first element of this list which is not contained in the tabu list. Thus, the best operational mode which is not currently marked "tabu" is selected. Upon selection, the operational mode is added to the tabu list. As the size of this list is bounded by l_i and in each iteration it is truncated to this size, the re-selection of the currently added operational mode is effectively excluded from the next l_i iterations. After performing this procedure for all considered time slots, the new selection is published just like in the DSS algorithm, triggering activity in the neighbourhood of the agent.

The tabu list mechanism enables the agent to select operational modes which impair its local solution but might possibly lead to a global optimum. In contrast to the DSS algorithm, the DSS-T will not rapidly terminate. As every agent definitely will make a new selection when triggered by the environment, the system will never reach a global equilibrium condition (unless we set $l_i \geq m_i$, in that case every operational mode would be chosen exactly once before the algorithm terminates). This effect introduces the problem of *termination*: How can agents decide *not* to make a new selection but instead stick to the current one? And what kind of mechanism can guarantee that all agents make this choice when the system currently exhibits a near-optimal global solution? These questions cannot be answered yet, but will be addressed in our future work.

5. EVALUATION

5.1 Simulation framework

To evaluate the proposed (and prospective) self-organized algorithms for solving the given optimization problem, a simulation framework has been implemented. The framework is written in the Python programming language and is able to simulate an arbitrary number of autonomous software agents as defined in section 4.1. Each of these *workers* is identified by an URI of the form (*host:port*) and has the ability to send and receive messages to and from other agents. The messages are sent as network packages, so that the whole simulation can be carried out either locally or distributed. As stated in section 4.1.2, local perception of the workers is carried out via communication with their neighbours. We make the following assumptions about communication:

ASSUMPTION 1. *Identity*
All nodes have unique IDs.

ASSUMPTION 2. *No lost messages*
All sent messages will eventually be received.

ASSUMPTION 3. *Message travelling time*
The message travelling time is constant so all messages are received in the order they have been sent.[1]

5.2 Activity restriction protocol

In the so far outlined paradigm however, conflict situations could arise if two neighbouring workers sense each other and act upon this perception *simultaneously*, effectively hindering each other's goal achievement. For example, let a system $\mathcal{A}$ consist of two agents a_0, a_1 with $\mathcal{T} = \{t_0\}$ and $g(t_0) = 0$. Each agent has the same two equal rated possible modes of operation so that

$$\mathcal{P}_0(t_0) = \{(-1.0, 1.0)_0, (1.0, 1.0)_1\},$$
$$\mathcal{P}_1(t_0) = \{(-1.0, 1.0)_0, (1.0, 1.0)_1\}.$$

Now let an initial solution be

$$s_{init} = ((-1.0, 1.0), (-1.0, 1.0))$$

so that a_0 as well as a_1 have their first operational mode selected. The rating for this solution is $F(g, s_{init}) = 2$. Now both agents get active simultaneously. According to DSS or DSS-T, they will now both perform a local search and select an operational mode which improves the current solution. In their local view, the second operational mode $(1.0, 1.0)_1$ would reduce the rating to 0. Therefore, both agents choose their second mode so that afterwards

$$s_{post} = ((1.0, 1.0), (1.0, 1.0))$$

which still yields $F(g, s_{post}) = 2$. This procedure is then reversed in the next iteration and the system will begin to oscillate: $s_{init}, s_{post}, s_{init}, s_{post}, \ldots$ while the rating constantly remains at $F(g, s) = 2$.

Therefore an activity restriction protocol has been designed which prevents such conflicts. Given a set of interconnected agents expressed as a graph G as defined in section 4.1.2, the activity restriction protocol prevents the simultaneous action execution of neighbouring agents. A related problem is known as *mutual exclusion in networks*. However, in the given case we aim at mutual exclusion only between neighbouring agents. So any number of nodes in the graph is allowed to be active at the same time, as long as they are not directly connected. This is similar to the *generalized dining philosophers problem*. In the following solution we included *fairness* so as to when a set of neighbouring nodes repeatedly and simultaneously request activity, it is granted rotatory to the involved nodes so that all nodes will equally often become active in the long term. To accomplish that, we have assigned each node a_i a state variable $st_i \in \{I, R, A\}$ where $I := inactive$, $R := requesting$ and $A := active$, and an activity value $act_i \in \mathbb{N}$. All nodes start with $st_i = R$ and $act_i = 0$. A node a_i may at any time change its state from I to R, signaling an activity request. Upon changing the state from R to A however, the activity value of the node is incremented by 1. Additionally, this transition is bounded by the condition

$$\forall a_j \in \mathcal{N}_i : f_{mutex}(i, j) > 0 \tag{13}$$

where

$$f_{mutex}(i, j) = \begin{cases} 1 & \text{if } st_j = I, \\ 2 & \text{if } st_j = R \ \wedge \ f_{act}(i, j) > 0, \\ 0 & \text{else,} \end{cases} \tag{14}$$

and

$$f_{act}(i, j) = \begin{cases} j - i & \text{if } act_i = act_j, \\ act_j - act_i & \text{else.} \end{cases} \tag{15}$$

This condition ensures that no neighbouring node is already active or is requesting with a lower activity value. In case of equal activity values, the node id is crucial. So the transition R to A can be expressed as

$$R \xrightarrow[act_i = act_i + 1]{\forall a_j \in \mathcal{N}_i :\, f_{mutex}(i,j) > 0} A \tag{16}$$

where the condition is placed above the transition arrow, and the required action which has to be taken below.

A node may stay arbitrary but finitely long in the active state so that we assume the following:

ASSUMPTION 4. *Finiteness*
A node in the active state will eventually become inactive.

The possible state transitions are shown altogether in figure 4.

[1] This is a quite restrictive assumption which will very likely become a topic in our future work.

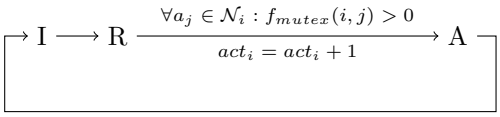

Figure 4: State transition diagram with $\frac{condition}{action}$ annotation for the activity restriction protocol.

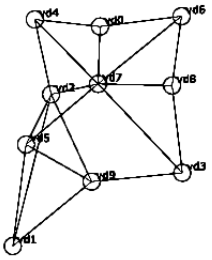

Figure 5: Connection graph of the examined scenario.

The correctness of the activity restriction protocol is shown in appendix A.

5.3 Simulation results

As this is a report on ongoing work, we will only examine a simplistic scenario here. We consider just one time slot

$$\mathcal{T} = \{t_0\}$$

where the global optimization goal is zero:

$$g(t_0) = 0.$$

There are ten agents ($n = 10$) with each four random modes of operation ($m = 4$) in the range [-1000,1000] so that

$$p_{i,k} = random(-1000, 1000),\ 0 \leq i < n,\ 0 \leq k < m.$$

Furthermore, all mode ratings are equal:

$$\forall p_{i,k} :\ r(p_{i,k}) = 1.$$

The connection graph is randomized and is shown in figure 5. And finally, as we use the DSS-T algorithm, the tabu list length is

$$\forall a_i :\ l_i = \frac{m_i}{2} = 2.$$

According to eq. 8 there are 1.048.576 possible solutions in this scenario. We calculated these with a brute force algorithm for reference.

Figure 6 shows the progress of the simulation of the scenario. The y-axis denotes the rating value while the x-axis displays the iterations of the simulation. An iteration is defined as a selection of an operational mode of any single agent. So after each of the ten agents has chosen its initial operational mode, the simulation would have proceeded at least ten iterations. Note that because the agents start without a selected operational mode, each agent must make its initial choice before the solution vector is called valid (see eq. 2). This point in time t_{init}, where all agents have chosen their initial solution, is marked by the vertical bar in the diagram. In the present case $t_{init} = 14$ because some agents became active more than once before all agents did their initial choice. The straight horizontal lines indicate ratings for the best (lower) and worst (upper) reference solutions. The optimal solution in this case has a rating of zero. In between, the progress of the algorithm is visible.

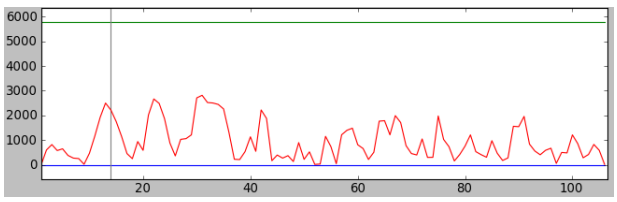

Figure 6: Exemplary simulation run.

It is noticeable that the system oscillates between excellent and average solutions, which is due to the tabu list. The algorithm seems to be able to generate near-optimal solutions in a very short amount of time in comparison to the size of the search space. However, this is based on a very few experiments done so far and therefore quite speculative. We are presently conducting a more formal analysis of a large number of simulation runs with varying random seeds and also different scenarios.

6. CONCLUSIONS

In this paper we have motivated the use of self-organizing mechanisms in the domain of energy systems. We provided an introduction to some relevant topics in this area: emergence, disturbance, equilibrium condition, cooperation, locality. The addressed optimization problem has been formalized in mathematical terms. We proposed the distributed search algorithms DSS and DSS-T which are based on the stigmergy mechanism in combination with a local search. A protocol for local mutual exclusion with fairness in a network of distributed processes has been proposed in order to avoid oscillations of suboptimal solutions. We proved the correctness of this protocol. Finally we gave some tentative interpretations of the performance of the algorithms.

In our future work we will address the formal analysis of the proposed algorithms. We will examine the reduction of problem difficulty which is given by distributing the calculations. Furthermore, we will compare the proposed approach to the well known `Adopt` algorithm [13] from the field of *Distributed Constraint Optimization* as well as to the `Max-Sum` algorithm which has been proposed in [3]. Besides that, we will develop other self-organizing algorithms for the presented problem.

7. ACKNOWLEDGMENTS

We would like to thank the Universitätsgesellschaft Oldenburg e.V. (UGO) for supporting this work. Furthermore, we thank the reviewers of this paper for their very constructive contributions.

8. REFERENCES

[1] S. Camazine, J.-L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau. *Self-Organization in Biological Systems*. Princeton University Press, 2001.

[2] J. Eto. The past, present, and future of U.S. utility demand-side management programs. Technical Report

LBNL-39931, Lawrence Berkeley National Laboratory, 1996.

[3] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. Decentralizsed coordination of low-power embedded devices using the max-sum algorithm. In Padgham, Parkes, Müller, and Parsons, editors, *Proc. of 7th In. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, pages 639–646, 2008.

[4] C. Gellings and J. Chamberlin. *Demand-Side Management: Concepts And Methods*. The Fairmont Press, Inc., 1988.

[5] C. Goldman and M. Kito. Review of demand-side bidding programs: Impacts, costs, and cost-effectiveness. Technical Report LBL-35021, Lawrence Berkeley National Laboratory, 1994.

[6] D. J. Hammerstrom, J. Brous, D. P. Chassin, G. R. Horst, R. Kajfasz, P. Michie, T. V. Oliver, T. A. Carlon, C. Eustis, O. M. Jarvegren, W. Marek, R. L. Munson, and R. G. Pratt. Pacific Northwest GridWise(tm) Testbed Demonstration Projects: Part II. Grid Friendly(tm) Appliance Project. Technical Report PNNL-17079, Pacific Northwest National Laboratory, Richland, WA, 2007.

[7] L. Hassas, G. Serugendo, A. Karageorgos, and C. Castelfranchi. On self-organising mechanisms from social, business and economic domains. *Informatica*, 30:63–71, 2006.

[8] G. Heffner. Configuring load as a resource for competitive electricity markets - review of demand response programs in the U.S. and around the world. Technical Report LBNL-51496, Lawrence Berkeley National Laboratory, 2002.

[9] C. Hinrichs, U. Vogel, and M. Sonnenschein. Modelling and evaluation of desynchronization strategies for controllable cooling devices. In I. Troch and F. Breitenecker, editors, *Proc. Mathmod 2009 - 6th Vienna International Conference on Mathematical Modelling*, number 35 in Argesim Report, Vienna, Austria, 2009.

[10] K. Kok, M. Scheepers, and R. Kamphuis. *Intelligent Infrastructures*, chapter Intelligence in electricity networks for embedding renewables and distributed generation, pages 179–209. Intelligent Systems, Control and Automation: Science and Engineering Series. Springer, 2009.

[11] J. Li, G. James, and G. Poulton. Set-points based optimal multi-agent coordination for controlling distributed energy loads. In *Proceedings of the Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO'09)*, San Francisco, California, USA, 2009.

[12] J.-P. Mano, C. Bourjot, L. Gabriel, and P. Glize. Bio-inspired mechanisms for artificial self-organised systems. *Informatica*, 30:55–62, 2006.

[13] P. Modi, W. Shen, M. Tambe, and M. Yokoo. Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence Journal (AIJ)*, 161:148–180, 2005.

[14] E. Pournaras, M. Warnier, and F. Brazier. Local agent-based self-stabilisation in global resource utilisation. *International Journal of Autonomic Computing*, 1(4):350–373, 2010.

[15] C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph.*, 21:25–34, August 1987.

[16] G. Serugendo, M.-P. Gleizes, and A. Karageorgos. Self-organisation in multi-agent systems. *The Knowledge Engineering Review*, 20(2):65–189, 2005.

[17] J. Short, D. Infeld, and L. Freris. Stabilization of grid frequency through dynamic demand control. *IEEE Transactions on Power Systems*, 22(3):1284–1293, 2007.

[18] M. Tröschel. *Aktive Einsatzplanung in holonischen Virtuellen Kraftwerken*. PhD thesis, Universität Oldenburg, 2010.

[19] D. Trudnowski, M. Donnelly, and E. Lightner. Power-system frequency and stability control using decentralized intelligent loads. In *Proceedings of the 2005/2006 IEEE PES T&D Conference and Exposition*, Dallas, TX, 2006.

[20] H. Wedde, S. Lehnhoff, C. Rehtanz, and O. Krause. Bottom-up self-organization of unpredictable demand and supply under decentralized power management. In *Proceedings of the 2nd IEEE International Conference on Self-Adaptation and Self-Organization (SASO'08)*, Venice, Italy, 2008. IEEE Press.

APPENDIX

A. CORRECTNESS OF ARP

A.1 Local mutual exclusion

THEOREM 1. *The action restriction protocol guarantees that no two neighbouring nodes are in the active state at the same time, so that*

$$\forall(a_i, a_j) \in \mathcal{E} : (st_i = A \Rightarrow st_j \neq A) \wedge (st_j = A \Rightarrow st_i \neq A).$$

PROOF. Assume two connected nodes are simultaneously in the active state, so that

$$\exists(a_i, a_j) \in \mathcal{E} : st_i = st_j = A.$$

As all nodes initially started in the inactive state, the nodes a_i, a_j had each to perform the transitions $I \rightarrow R \rightarrow A$ at least once, both conforming to the condition of $R \rightarrow A$. Then there are two possible cases:

1. The transitions $R \rightarrow A$ were performed sequentially, so that after the first transition $st_i = A \neq st_j$. Then because of $f_{mutex}(j, i) = 0$ the transition $R \rightarrow A$ of a_j cannot be performed, contradicting our assumption.
2. The transitions $R \rightarrow A$ of both nodes were performed simultaneously. This implies $st_i = st_j = R$ at the time of the transition. Then according to eq. 14:

   $$f_{act}(i, j) > 0 \ \wedge \ f_{act}(j, i) > 0$$

   and because of eq. 15:

   $$(act_i = act_j \quad \Rightarrow \quad i > j \ \wedge \ j > i)$$
   $$\vee$$
   $$(act_i \neq act_j \quad \Rightarrow \quad act_i > act_j \ \wedge \ act_j > act_i)$$

   which contradicts our assumption.

Therefore $\nexists(a_i, a_j) \in \mathcal{E} : st_i = st_j = A$. $\square$

A.2 Progress

THEOREM 2. *The action restriction protocol is deadlock free.*

PROOF. Assume a deadlock where an arbitrary number of nodes (at least one) are in the requesting state but none of them is able to perform the $R \rightarrow A$ transition, so that

$$\exists \mathcal{R} \subseteq \mathcal{A}, |\mathcal{R}| > 0 : \quad \forall a_i \in \mathcal{R} : \quad st_i = R \ \wedge \\ \forall a_j \in \mathcal{N}_i : f_{mutex}(i,j) \leq 0.$$

Let a_i be a node which has no neighbour with a smaller act_j so that

$$\exists a_i \in \mathcal{R} : \forall a_j \in \mathcal{N}_i : act_i \leq act_j.$$

There are two cases to be considered:

1. $\exists a_j \in \mathcal{N}_i : act_i = act_j$, then

   $$f_{mutex}(i,j) > 0 \quad \text{iff} \quad i < j$$

   and

   $$f_{mutex}(j,i) > 0 \quad \text{iff} \quad j < i.$$

2. $\forall a_j \in \mathcal{N}_i : act_i < act_j$, then

   $$f_{mutex}(i,j) > 0.$$

Both cases lead to

$$\exists a_k, a_l \in \{a_i, a_j\} : f_{mutex}(k,l) > 0 \ , \ k \neq l$$

which contradicts the assumption. □

A.3 Starvation freedom

THEOREM 3. *Any node in the requesting state will eventually become active.*

PROOF. Let a_i be a node in the requesting state which cannot become active so that

$$\exists a_j \in \mathcal{N}_i : act_j \leq act_j.$$

Considering the progress property (theorem 2) and the fact that every node must increase its activity value upon entering the active state, all neighbouring nodes will eventually get a higher activity value than a_i, enabling a_i to become active. □