# Agent system for energy consumption scheduling in an intelligent neighborhood – preliminary considerations

Karol Bocian* Maria Ganzha*†, Marcin Paprzycki†‡
* Warsaw University of Technology, Warsaw, Poland
† Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland
‡ Warsaw Management Academy, Warsaw, Poland

***Abstract*—It is extremely likely that, in the near future, a living community (consisting of multiple households) will be able to contract electricity according to its predicted needs. In this context, it is usually assumed that energy purchased within a (long-term) contract will be cheaper than energy purchased from the market (whenever needed). Hence, over-use of energy (going over the energy consumption limit, established in the contract), will result in purchase of additional energy and thus should be avoided. In this context, let us also assume that, in a smart home, devices are turned on and off by a centralized control mechanism. Furthermore, individual households can express preferences when each of their devices is to be used. In this case, the question that needs to be answered is: how to schedule devices in a way that will be satisfactory to the users (households) and match the parameters of the contract. The aim of this text is to describe an agent system that was developed to schedule device operation according to the contract-imposed limitations. Experimental results, illustrating its operation, in various scenarios, are presented.**

## I. Introduction

Recent years bring many trends that are likely to shape future of energy management. Let us summarize some of them. (1) Growing popularity of renewable energy sources. Locally owned and operated solar panels, wind turbines, energy storage systems, etc., lead to the situation when (slowly, but systematically increasing) part of energy is produced "locally" (and stored, as needed, for a limited time, e.g. using electric vehicle batteries). Furthermore, energy production, based on some forms of renewable sources, depends on "outside conditions", and its availability can be rather volatile (e.g. due to the dependency on wind / sunshine and clouds). This, in turn, makes the problem of stabilizing the power grid substantially more complex. (2) Advances in computer hardware and software, as well as theory and practice of pertinent areas of computer science, facilitated development of new methods of: (a) control of electricity consuming devices (creation of "smart devices") [1], (b) autonomous negotiations (including price negotiations) realized using software agents [2], (c) precise, real-time metering of use of energy for the households and, possibly, even on the level of individual devices [3], (d) trend prediction (e.g. using time series analysis) [4], etc. (3) Group purchases, as a way to reduce prices [5], [6]. (4) New generation of batteries (stationary and in electric cars), which lead to scenarios, in which energy is collected ; stored when cheap (e.g. at night) and used when expensive (e.g. during peak hours of the day) [7], [8], (5) learning of user behaviours (to develop a schedule matching their preferences, to increase the comfort of life) [9], (6) weather prediction (to schedule use and generation of energy) [10].

Combination of these trends results in novel ideas concerning energy management (on the level of individual households, which is the focus of this work). (A) Communities of houses / apartments; will be able to form groups to negotiate with energy providers contracts for volume purchases of energy. Here, it is assumed that energy purchased within the contract will be substantially cheaper than buying (additional) energy directly from the market. (B) Majority of energy consuming devices, in every household, will be controlled by a central controller, which will be capable of turning them on and off, in order to minimize energy use and/or to match the profile of the energy contract. Observe that, already today, in Poland there exist energy contracts with two prices "day" (expensive) and "night" (cheap). Hence, devices such as washing machines, should be turned on during cheap energy times. (C) With proliferation of "smart home assistants" (e.g. Alexa[1], Google Home[2], Lenovo Smart Assistant[3]) it is easy to envision that these software entities will be able to communicate with each other. One of possible topics of communication could be forming a group to jointly negotiate energy contracts (as in (A), above, see, also [11]), (D) Raspberry Pi[4], Arduino [12] and similar solutions give opportunity to build cheap home management systems.

Obviously, ideas mentioned here are limited in their scope. They do not include, among others, use of renewable energy sources, prosumer behaviors, etc. Nevertheless, they provide an initial set of issues that are worthy considering. Here, the assumption is that, as work continues, these (and other) aspects will be added to the developed simulator.

Based on presented ideas, in what follows, we focus our attention on scheduling of use of smart devices in a way that would, on the one hand, satisfy demands / preferences of

[1] https://www.alexa.com
[2] https://madeby.google.com/home/
[3] http://www3.lenovo.com/us/en/new-products/Lenovo-Smart-Assistant/p/99SD9EI1SA1
[4] https://www.raspberrypi.org

individual households; while, on the other, keep the total use of energy, of a group of households, below the threshold specified in a (hypothetical) contract. Here, it is easy to notice (see, also (C), above) that a very natural way of conceptualizing the considered problem would be to use software agents [13], [14], [15]. In this case, a separate agent could represent each individual household (and attempt at satisfying preferences of its owner(s)) and communicate with other agents (representing other households) to develop a joint schedule of energy use.

Following this line of reasoning, we have implemented a prototype agent system for management of energy use in an "intelligent community". The aim of this paper is to report on the creation of the system and on initial experiments that illustrate its potential. To this effect we proceed as follows. In the next Section we make explicit the assumptions and research questions addressed in this paper. We follow with a brief description of the AgentPlanner software and its modifications applied to adjust it to the energy management. Next, we summarize the experimental setup and present results of experiments for selected use case scenarios. Some observations concerning further development of the system complete this contribution.

## II. Assumptions and research questions

Let us now make explicit the assumptions underlying the use case scenario that has been modeled. (1) We consider a "smart neighborhood" consisting of $N$ households. Note that these households can be placed within a single compound (*physical community*), or can consist of any group of households that used some Internet-based infrastructure to form a *virtual community*. (2) These households have reached an agreement to group purchase electricity (to save money). (3) We assume that the contract is a static one (a specific upper limit of electricity consumption throughout the whole day has been contracted). Obviously, implicitly, we assume that it is possible to arrange such contract between a community and an energy provider. Here, let us note that the developed software can be modified to support different assumptions, e.g. to specify (differentiate) allowed electricity consumption for each hour of the day. (4) While the energy purchased within the contract is "cheap", use of additional energy / going above the limit is "expensive". However, no specific "price difference" is introduced (this can also be considered in the future, to make a more precise quantitative assessment of effects of various parameters of the model). Furthermore, there are no mechanisms that would allow the community to "sell back" unused energy. (5) Each household $h$ has a certain number ($M_h$) of smart energy consuming devices, which are "centrally controlled". In other words, there exists a mechanism that can turn them *on* and *off*. Here, we do not consider "small devices" that are often turned on and off, and that use relatively small amount of electricity (e.g. an electric tooth brush, or a cappuccino making machine). (6) Each of modeled (large) devices uses different amount of electricity (from a pre-specified interval), and is turned on for a specific amount of time. (7) Owners of each household can express their preferences concerning (a) when a given device should be turned on, and (b) how important it is that this device is turned on at a given time. Preferences are represented as integers from the interval $[1, 5]$ (here, $1$ represents a weak preference, while $5$ means that a given device *has to* be turned on at a given time. (8) There is no "prosuming" in the system (i.e. there are no solar panels, wind turbines, batteries, etc.). Similarly, there are no electric cars (which can be used in prosuming scenarios). On the one hand, we admit that this is an important limitation, on the other, adding prosumer behaviors to the considered scenario introduces level of complexity that should not be dealt with at this stage of the project. (9) Special class of devices, that must run at specific time periods (or always be available), e.g. light system within the physical community, has been introduced. However, for the time being, effects of their existence are not fully accounted for (see, below).

To model the smart community guided by these assumptions, we have decided to use software agents. Specifically, software agents are to be used to manage (negotiate) the schedule of use of devices in each household. Furthermore, software agents should be capable of adapting the schedule in case, for instance, of new device joining the mix, or one of the households resigns from turning on one (or more) of its devices at a given time. However, discussion of this capability is out of scope of current contribution. The main, initial, questions that we have decided to address were: (i) are software agents, actually, a good approach to deal with issues brought about in the outlined use case scenario? (ii) can they facilitate a robust scheduling approach that will allow devices to be used in a way that will assure that the total energy consumption is below the threshold, while user preferences remain satisfied? (iii) what can be done if it is impossible to obtain a "preferred / optimal schedule"? what will be the characteristics of a non-optimal schedule? how will such schedule affect individual preferences?

## III. AgentPlanner and its adaptation

Recently, software agents have been used to schedule university courses (lectures / laboratories / etc.), taking into account, among others: (a) maximal fulfillment of (explicitly defined) preferences of both, teachers and students, (b) necessary satisfaction of "logistics requirements" (i.e. total number of existing / available lecture halls / laboratory rooms cannot be exceed). Furthermore, the developed software took into account the following restrictions: (c) teacher / student can have only one activity in a single time unit, (d) only one activity can be scheduled in a single time unit in a given room, and (e) there is a limited number of "activity slots" per day (in other words, teaching activities should not start before 8:00 AM, and should end by about 8:00 PM). Complete description of this system (named AgentPlanner) can be found in [16], [17]. There, one can also find encouraging experimental comparison between the performance of the AgentPlanner and other timetabling systems.

Even a superficial comparison between the scenario, to which the AgentPlanner has been applied, and the needs of

scheduling energy use in the smart community, considered here, shows that there are important parallels between them. In both cases, there is a specific number of participants and activities (teachers / classes vs. devices and their use). Limited number of "slots" can be "filled with activities", due to the "natural restrictions" (number of available rooms vs. bound imposed by the contract). Activities have preferences assigned to them (some teachers prefer to teach in the evening vs. some inhabitants prefer to do laundry in the morning). Teacher preferences have two "aspects": day and hour and these preferences have "strength" associated with. The same concerns household preferences concerning day and time when a given device is to be turned on.

Obviously, there are also important differences. For instance, while the total number of rooms cannot be changed, if necessary; it is possible to go beyond the limit of energy consumption, stipulated in the contract, and pay the price of use of "expensive electricity". Nevertheless, we have decided to adapt the AgentPlanner software to work in the considered use case. To start, we have conceptualized the system that was to be developed (as a result of adaptation of the AgentPlanner), in the form of a use case diagram, presented in Figure 1.

In the diagram (Figure 1), on the "left hand side" we see the outside users of the system: *Administrator* and *Resident* (of the community). On the "right hand side" the "internal users" (i.e. agents) are represented. External users can add devices to the ecosystem. As a result, the scheduling algorithm will be started. Its result will be either a schedule, or information that the device was "rejected" (it cannot be added without breaking constraints). Scheduling algorithm starts from initialization of a *Creator Agent*, which, in turn, initializes all the remaining agents. Both, the *Administrator* and the *Resident* can provide input data that will be used by the scheduling algorithm. Input data is transferred to the database, managed by the *DatabaseInitAgent*. The *Administrator* and the *Residen*s can also add information concerning: new buildings, devices, power outlets, and preferences related to use of existing devices. This data is sent to the *DatabaseAgent* that places them in the database. Process of schedule creation is run by the *ScheduleAgent*. In what follows we are interested in the process of *Creation of a new timetable*, while we omit the *Reorganization of timetable* use case (understood on the level of the whole community).

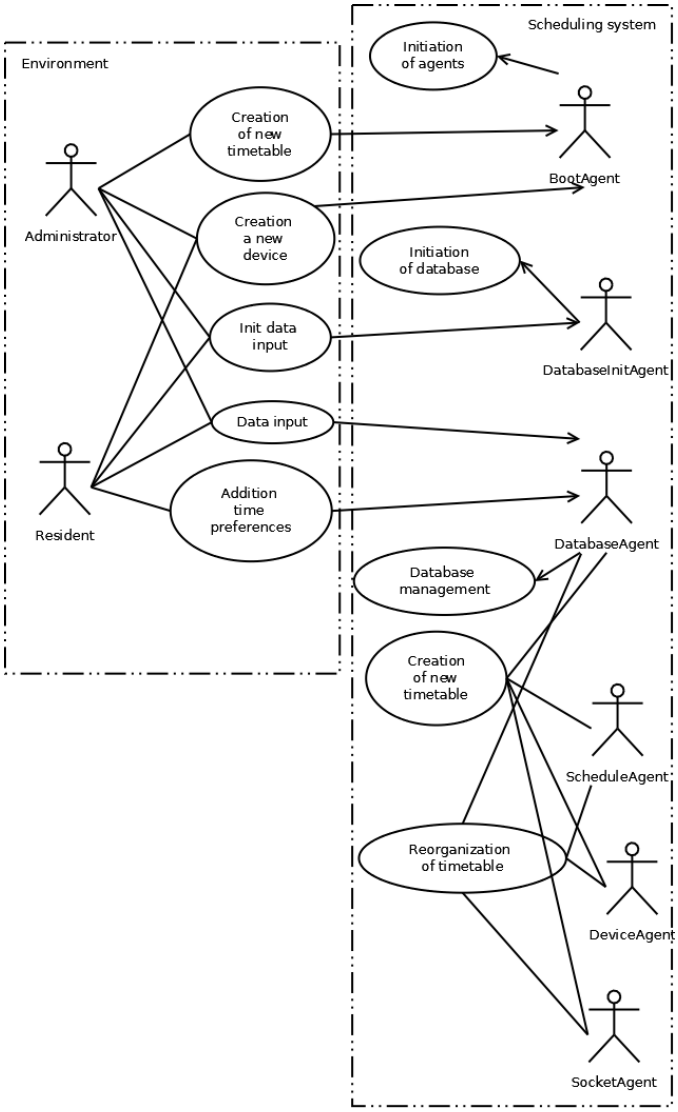


Fig. 1. Use case diagram for the developed system (foundation for modification of the AgentPlanner software)

### A. Modified AgentPlanner

Let us now present a brief description of the way that the original AgentPlanner created class schedule. Note that, here, only its key aspects (from the point of view of this paper) are described; for a complete description, see [16], [17].

The original AgentPlanner can be used in two situations. First, to build a schedule. Second, to reorganize an existing schedule. In the schedule building algorithm, which proceeds in rounds, the *Judge Agent* receives requests (concerning placing a specific activity in the plan) from agents representing teachers. Next, the *Judge Agent* puts into the schedule requests that are not in conflict, while for the remaining ones, the evaluation function is applied to select the "winners" (while the "losers" are rejected; in that round).

If, in a given round, there are rejected requests, the *Judge Agent* starts the schedule reorganization process. This proceeds as follows: (*First* agent), with (a) rejected request(s), sends the most important of them to obtain a spot in the plan. The *Judge Agent* checks who occupies this spot in the schedule (e.g. the *Second* agent). Then, it asks this agent to move its activity. If *Second* agent finds a new spot for its activity, then it accepts the change request, and *Judge Agent* moves activity of the *Second* agent to the new spot, and adds the activity belonging to the *First* agent to the requested spot. If the *Second* Agent rejects the *Judge Agent* request, then the *First* agent sends its next proposal (which spot this activity could occupy). If all its proposal(s) (concerning selected activity) are rejected, then this activity is placed in the *Rejected Activities* set.

Process continues until all activities are placed in the

schedule, or some of them cannot be scheduled at all (due to the conflicts and strength of preferences of individual agents). All such activities constitute the *Final Rejected Activities* set.

To match the needs of the smart community scenario, the AgentPlanner has been modified as follows:

- For representation of power consumption of individual devices, an appropriate table in the database has been augmented, by including objects representing devices and information about power consumption of each one of them. The fact that each device can have different power consumption is being taken into account when the assessment function is calculated.
- The assumption that there is threshold of allowable power consumption (above which cost increases considerably) had to be taken into account. This information is used in the planning / scheduling algorithm in the following way: each hour has its own limit of power consumption and the algorithm does not allow to turn on device, which can use energy above the limit. Limit is also used in the evaluation function of each device that is to be placed in the schedule. Here, note that, for obvious reasons, the order (round number), in which devices are submitted to the *Judge Agent* matters. It is particularly important when each device has different power consumption level.
- Let us recall that, within the community, there are devices that have to work non-stop. A simple example would be various life-supporting devices (e.g. in an ambient assisted living type situations [18]). These devices should have "priority" above all other devices (and have to be powered-up as long as there is available electricity, and regardless of the cost). Furthermore, if the community is a physical one (e.g. it is a "gated community"), then running its infrastructure (or, at last, a part of it) has priority higher than household devices, other than these that support life. Hence, emergency lights, or the mechanism that opens and closes the gate(s) have to work (they are more important than the possibility of drying hair at 9:15PM, in household 34B). To represent such situations – devices that have to be on, "unconditionally", at specified times – a *non-negotiable priority* (with value 6) was added.
- Division into time-slots had to be modified. Instead of a limited number of blocks every hour, during a certain time interval, day was divided into 24 time-slots (periods). As a result, the developed schedule is always completed for the whole day.
- The evaluation function had to be adjusted, to take into account representation of power consumption of individual devices, contract-imposed energy consumption limit, and the non-negotiable priority. The new function (*ECAL*) has the following form:

$$ECAL \quad (Day, Time, Socket) = (A \times B \times C + E + F \times 5 + G \times 5) \times (1 + L) \qquad (1)$$

Here, note that, depending on the specific scenario that is to be realized, some of these terms may be equal to zero – be absent from the formula:

- *Day* – day for which the plan is created,
- *Time* – time interval (one hour blocks are used in the current version of the system), for which the schedule is being constructed, *Socket* – socket, to which a number of specific devices is connected, it is represented by the *SocketAgent*; *SocketAgent*s communicate with *DeviceAgent*s representing individual devices (*DeviceAgent*s turn devices on and off),
- *A* – building priority – it is possible to set different priority for individual buildings,
- *B* – preference of a given device to be turned on at a given day,
- *C* – preference of a given device to be turned on at a given hour,
- *E* – number of devices that, during specific *Day*, are plugged in a particular *Socket*,
- *F* – term that is equal 1 if a device that is plugged to the given *Socket*, was scheduled to work during the previous *Time*, else it is equal 0; in this way continuity of usage of electricity associated with a given *Socket* is preferred (this is a somewhat controversial decision, following the ideas underlying the original AgentPlanner that may be removed in the future),
- *G* – term that is equal 1 if a device that is plugged to the given *Socket* is scheduled to work during the next *Time*, else it is equal 0 (another parameter that originates from the original AgentPlanner that may be removed in the future),
- *L* – is a function that establishes the "usefulness" of a given device "being turned on" during given *Time*; in other words, the role of this parameter is to favor situations when there are no gaps caused by lack of devices that could fit them; the function has the form:

$$L = \lfloor ECAL_{MAX} * K \rfloor, \qquad (2)$$

  where *K* represents how well a device "fills the gap"; specifically, the better the power consumption of a given device fits into the "left-over gap" in a given time slot, the higher the score; (if it does fill it completely, the maximum score is granted). $K$ belong to the interval $[0, 1]$,

$$K = \frac{used}{limit(Day, Time)}, \qquad (3)$$

- $ECAL_{MAX}$ – maximum value of the *ECAL* function, before addition of the device under consideration
- *Used* – power consumption of a given device
- limit(*Day*, *Time*) – limit of energy consumption established in the contract for a specific *Day* and *Time*
- "Softening" user preferences – a modification that allows the system to "soften" user preferences; this is

achieved by adding "imposed preferences" according to the argument *Fr*; this functionality was added to see the effect of residents being more flexible (benevolent) in setting their preferences); it works as follows: if user sets his preference to 000005000000 (device has to definitely be turned on at a given hour and no other preferences are specified) and $Fr = 1$, then the softened preference has the form 000045400000 (i.e. while the user-selected hour has the strongest preference, it is also possible to turn the device on before or after, with a slightly lesser preference), if user sets her preference to 000005000000 and $Fr = 5$ then the softened preference has the form 012345432100, making the requested schedule quite flexible.

The scheduling and rescheduling code and processes, from the original AgentPlanner, had to be only slightly modified:

- Agents with changed names and roles: *TeacherAgent* became *DeviceAgent* and now it represents devices; *RoomsAgent* became *SocketAgent* and represent sockets in buildings. These changes allow also to connect specific devices to correct households in specific buildings.
- While the evaluation function was modified (see, above), its results are used like in the original algorithm.

The whole process remained practically unchanged. During schedule creation, the *Judge Agent* receives propositions from the *Device Agents* (via the *Socket Agents*) and gives them answers: permissions and rejections. The rescheduling process works as before. The *Jugde Agent* asks agents to start their device at another time slot and informs about success or failure of attempted rescheduling process.

### B. Implementation details

The modified AgentPlanner has been implemented in using JADE agent platform (version 4.3.0;[5]). The database component was MySQL database (version 5.1.69;[6]). NetBeans (version 7.0.1;[7]) have been used as a programming environment. Finally, as in the original AgentPlanner, GUI has not been developed, a simple line interface, CSV files, and set of images remained as the way to run experiments.

## IV. Experimental setup

To test the developed solution, we have run multiple experiments. Let us list parameters common for them:

- contract-based energy limit was set at 2000 (this is an arbitrary value that does not have any practical / real-world counterpart),
- number of buildings in the community was set to $N = 20$ (running experiments with "granularity" at the level of individual buildings, rather than experimenting with multi-household buildings, does not make difference for the results presented in this contribution),
- each simulation of energy consumption dealt with a single day (24 hours),
- all devices started working on the hour, and were turned on for one hour; in other words, for each device, the day was divided into 24 available time-slots.
- each experimental scenario was completed 10 times for random values; presented results are averages of these 10 runs

The two main scenarios were as follows.

- *Scenario 1* – all devices have power consumption equal to 250 (again, this is an arbitrary value).
- *Scenario 2* – power consumption, for individual devices, was set at a random number, a multiple of 10, from the interval $[200, 400]$.

For each scenario two situations have been considered. First, all device preferences were set at 5 and each device had only a single specific time of the day that it was to be turned on. Second, it was assumed that the system will soften user preferences (using the *Fr* parameter, defined above).

## V. Experimental results

Let us now describe the results of completed simulations. Depending on the specific scenario we have recorded and analyzed the following "performance measures":

1) Total (for the one day schedule) number of violated time preferences (e.g. user has set time for 10AM, while the device was turned on at 11AM).
2) How large was the scheduled energy use, vis-a-vis the contract.
3) Number of devices that did not fit into the constructed schedule.

To illustrate the way that the modified AgentPlanner works, in Figures 2,3 and 4, we show examples of calculated schedules. In these Figures, the vertical axis presents scheduled energy, while the horizontal axis presents time of the day, divided into 1-hour intervals. Horizontal red line demarcates the energy consumption limit for that day (recall, that we assume that there is a single energy limit for a given day). Small rectangles represent energy consumption of individual devices. Presented schedules were calculated for Scenario 1. Similar results have been obtained also for Scenario 2.

Figure 2 shows schedule for the buildings having 5 devices each, and $Fr = 0$. Here, we can see that all devices have been fitted into the schedule (none of them exceed the limit of the contract). This means that all $20 * 5 = 100$ devices have been scheduled for times, which were specified by their owners. Obviously, the number of devices scheduled at different hours varies, representing user preferences one to one. Here, let us note that for all 10 experiments, with random preferences of 20 households, and possibility of scheduling $2000/250 = 8$ devices per time slot, each time we were able to schedule all requests.

Figure 3 shows a sample schedule for the buildings having 6 devices and $Fr = 0$. Here, we can see that it was impossible to build the schedule without exceeding the limit. Here, going

[5]http://jade.tilab.com/
[6]https://www.mysql.com/
[7]https://netbeans.org/


Fig. 2. Example of calculated schedule for: Scenario 1, 5 devices per building, $Fr = 0$

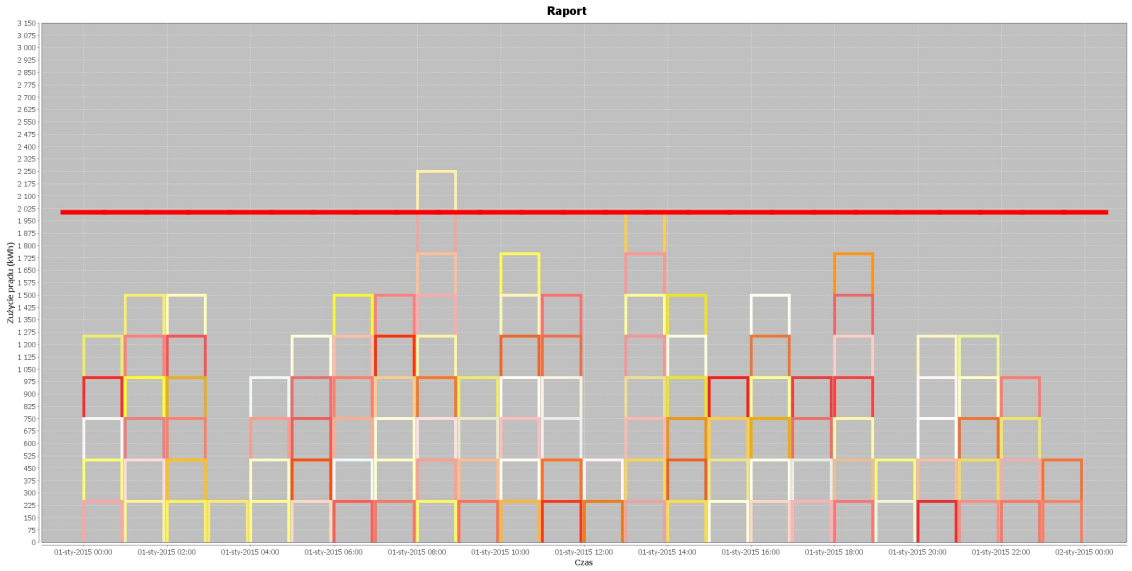


Fig. 3. Example of calculated schedule for: Scenario 1, 6 devices per building, $Fr = 0$

above the limit was necessary to avoid violation of user preferences. For the time period between 8:00 and 9:00AM, 9 devices had to be scheduled and thus the contract has been violated. This example was selected deliberately, to illustrate the situation of violation of contract. Obviously, there is space available, and it would have been possible to fulfill the contract by violating one user preference and moving one of the devices that were scheduled for the offending time slot to another one.

Figure 4 shows a sample schedule for buildings having 9 devices and $Fr = 5$. Here, it would be impossible to build a satisfactory schedule for all devices (without violating the contract). Hence, we have decided try softening user preferences" (see, above). Once we have set $Fr = 5$, which means that all requests for all devices had the form 123454321, we were able to schedule all devices. Observe that the "available space" is almost completely filled by scheduled devices (only 12 slots remain unused). However, let us stress it again, using this approach means that user preferences have been violated multiple times (see, next).

### *A. Increasing number of devices until threshold has been reached*

Let us look now into the results of experiments from a "quantitative perspective". In the first series of experiments we have observed how "good" is the proposed approach in scheduling devices for the increasing number of devices per building. We have done this for both scenarios (the simplistic

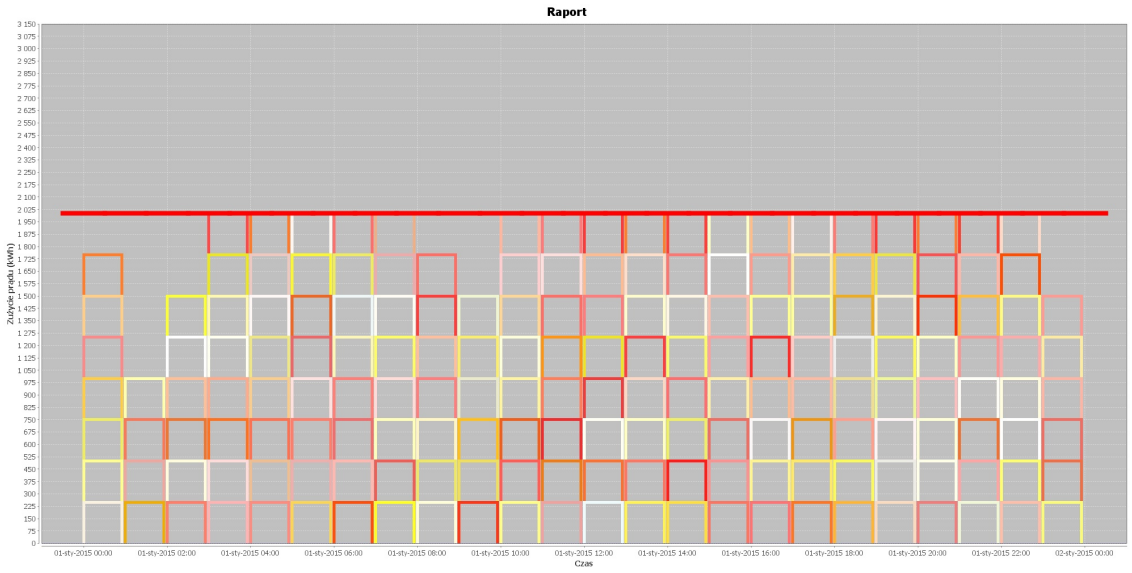


Fig. 4. Example of calculated schedule for: Scenario 1, 9 devices per building, $Fr = 5$

one, with all devices having the same power consumption, and for the mixed power consumption levels). Furthermore, we have considered the effect of parameter $Fr$.

The results are presented in Tables I, II, and III, illustrating various aspects of produced schedules. In the Tables, in columns number of devices per building varies from 6 to 11 (as mentioned above, for 5 devices per building the proposed approach was able to schedule all devices. For both scenarios, cases of $Fr = 0$ and $Fr = 5$ are presented.

TABLE I
AVERAGE NUMBER OF DEVICES THAT WERE NOT SCHEDULED

| | | Number of devices per building | | | | | |
|---|---|---|---|---|---|---|---|
| | | 6 | 7 | 8 | 9 | 10 | 11 |
| Scenario 1 | $Fr = 0$ | 02.4 | 06.6 | 13.6 | 20.6 | 31.2 | 43.5 |
| | $Fr = 5$ | 00.0 | 00.0 | 00.4 | 03.3 | 14.0 | 28.9 |
| Scenario 2 | $Fr = 0$ | 10.9 | 19.7 | 31.7 | 47.2 | 61.8 | 82.6 |
| | $Fr = 5$ | 00.0 | 02.3 | 10.5 | 29.8 | 51.1 | 74.0 |

Table I illustrates fast growing average number of devices that were not scheduled in case of assuring that the preferences of residents are upheld. This is particularly visible in the case of Scenario 2, showing (once more) that packing schedule consisting of uneven size requests is more complex than packaging equal sized requests. This illustrates one of important directions of possible improvements of the proposed approach. Softening (at level $Fr = 5$) owner preferences has definite positive effect on the results of scheduling. Here, for 6 devices it is possible to make the algorithm work in both Scenarios. Moreover, in the case of even power consumption, it is still possible to make the schedule work for up to 8 devices (here, only very few cases were "unschedulable"). Again, focusing on the contract, while violating user preferences had more positive effect in the case of even-sized power requests.

Table II shows average number of violated *Time* preferences; in other words, in how many user requests have not been satisfied. Conceptual difference between this Table and Table I is that here the "spread" of contract violations is captured. In rows with parameter $Fr = 0$ there are only zeros. This is because this approach does not allow actual violation of *Time* preferences. Here, the result is rejection of one, or more, devices. These devices are not scheduled at all. Hence, while user preferences and the contract are not violated, the price is that some devices will not run. Obviously, this is unrealistic, but it illustrates the nature of the problem that is being dealt with.

TABLE II
AVERAGES NUMBER OF VIOLATED *Time* PREFERENCES

| | | Number of devices per building | | | | | |
|---|---|---|---|---|---|---|---|
| | | 6 | 7 | 8 | 9 | 10 | 11 |
| Scenario 1 | $Fr = 0$ | 00.0 | 00.0 | 00.0 | 00.0 | 00.0 | 00.0 |
| | $Fr = 5$ | 02.4 | 06.6 | 13.2 | 17.3 | 17.2 | 14.6 |
| Scenario 2 | $Fr = 0$ | 00.0 | 00.0 | 00.0 | 00.0 | 00.0 | 00.0 |
| | $Fr = 5$ | 10.9 | 17.4 | 21.2 | 17.4 | 10.7 | 08.6 |

For $Fr = 5$ we see that, in all cases, some *Time* preferences have been violated. This means, for instance, that for Scenario 1, for 7 devices, on average 7 requests (out of 140) have not been scheduled as requested. We can see, again, that preferences in Scenario 2 were more often violated than in Scenario 1. However, the average number of violated *Time* preferences does not increase "forever". Starting for 10 devices in Scenario 1 and for 9 devices in Scenario 2 we can see that these numbers decrease. The explanation is as follows: the difficulty of a problem was getting too high; thus, increasing number of violated *Time* preferences did not help solving the problem.

Finally, Table III shows average amount energy that was scheduled in various situations (in thousands of energy units). Recall that the daily limit is equal $48000$ $(2000 \times 24)$ en-

TABLE III
AVERAGES AMOUNT OF SCHEDULED ENERGY USE; IN THOUSANDS

| | | Number of devices per building | | | | | |
|---|---|---|---|---|---|---|---|
| | | 6 | 7 | 8 | 9 | 10 | 11 |
| Scenario 1 | $Fr = 0$ | 29.4 | 33.3 | 36.6 | 39.8 | 42.2 | 44.1 |
| | $Fr = 5$ | 30.0 | 35.0 | 39.9 | 44.1 | 46.5 | 47.7 |
| Scenario 2 | $Fr = 0$ | 32.4 | 36.2 | 39.2 | 41.2 | 43.3 | 43.7 |
| | $Fr = 5$ | 35.2 | 40.6 | 44.5 | 45.7 | 46.1 | 46.0 |

ergy units. Results presented in this Table should be viewed together with those presented in the remaining two Tables. They show that "forcing users" to soften their preferences helps developing a more efficient schedule. In Scenario 1, for 11 devices per household, almost all allowable energy is going to be used. Even in the case of Scenario 2, only 2000 energy units are not scheduled. These results have been obtained for the total requested power consumption larger than the limit (for Scenario 1, 55000 energy units vs available 48000). More realistic situation, for Scenario 1 (where the exact number of units and their energy consumption are known), is when 9 devices reside in each household. Here, the total requested energy ($9 \times 20 \times 250$) is 45000 energy units. Hence, a perfect schedule should be able to run all of them (with 3000 power units to spare)). As can be seen, for $Fr = 0$, 39800 energy units are scheduled, with approximately 20 devices rejected. However, for $Fr = 5$, 44100 energy units are scheduled to be used, with approximately 3 units rejected due to contract violations. This latter result seems quite reasonable, if softening user preferences is possible.

## VI. CONCLUDING REMARKS

In this paper we have shown that use of software agents to schedule energy consumption in a smart community is quite promising. We have also shown a number of areas where the proposed approach needs improvements. Furthermore, due to the space limitation, we have not reported on other experiments that we have completed (that show correctness of the developed system). For instance, we have checked that using the "non-negotiable priority" (value 6) allows the proposed approach to correctly schedule the required services. Obviously, more simulations are needed to capture all factors that are already available in the developed system. Here, rescheduling on the fly, is the next important area, which we will report in subsequent publications.

## REFERENCES


[1] I. Koutsopoulos, V. Hatzi, and L. Tassiulas, "Optimal energy storage control policies for the smart power grid," in *2011 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Oct 2011, pp. 475–480.

[2] I. Praa, C. Ramos, Z. Vale, and M. Cordeiro, "Intelligent agents for negotiation and game-based decision support in electricity markets," Computer Science Department, Polytechnic Institute of Engineering, Porto, Portugal, http://www.gecad.isep.ipp.pt/GECAD/Files/Reports/Report0307/Papers/IESJournals/Isabel_paper_CRL.pdf, Tech. Rep., 2015.

[3] J. E. Petersen, V. Shunturov, K. Janda, G. Platt, and K. Weinberger, "Dormitory residents reduce electricity consumption when exposed to realtime visual feedback and incentives," *International Journal of Sustainability in Higher Education*, vol. 8, no. 1, pp. 16–33, 2007. [Online]. Available: \url{https://doi.org/10.1108/14676370710717562}

[4] E. Gonzlez-Romera, M. ngel Jaramillo-Morn, and D. Carmona-Fernndez, "Forecasting of the electric energy demand trend and monthly fluctuation with neural networks," *Computers & Industrial Engineering*, vol. 52, no. 3, pp. 336 – 343, 2007, planning and Management of Energy and Infrasturcture Engineering Projects. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360835207000204

[5] J. Xu, L. Duan, and R. Zhang, "Energy group buying with loading sharing for green cellular networks," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 4, pp. 786–799, April 2016.

[6] R. J. Kaufman and B. Wang, "Bid together, buy together: On the efficacy of group-buying models in internet-based selling," *Handbook of Electronic Commerce in Business and Society: Research Collection School Of Information Systems*, pp. 99–137, 2002.

[7] G. Berdichevsky, K. Kelty, J. Straubel, and E. Toomre, "The tesla roadster battery system," http://large.stanford.edu/publications/power/references/docs/tesla.pdf, Tesla Motors, August 2016.

[8] B. Dunn, H. Kamath, and J.-M. Tarascon, "Electrical energy storage for the grid: A battery of choices," *Science*, vol. 334, no. 6058, pp. 928–935, 2011. [Online]. Available: http://science.sciencemag.org/content/334/6058/928

[9] R. Yang and M. W. Newman, "Living with an intelligent thermostat: Advanced control for heating and cooling systems," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, ser. UbiComp '12. New York, NY, USA: ACM, 2012, pp. 1102–1107. [Online]. Available: http://doi.acm.org/10.1145/2370216.2370449

[10] F. Oldewurtel, A. Parisio, C. N. Jones, D. Gyalistras, M. Gwerder, V. Stauch, B. Lehmann, and M. Morari, "Use of model predictive control and weather forecasts for energy efficient building climate control," *Energy and Buildings*, vol. 45, pp. 15 – 27, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378778811004105

[11] S. Nowak, M. Ganzha, M. Paprzycki, C. Badica, M. Ivanovic, and S. Simionescu, "Agent-based system for highway gasoline price negotiations," in *Proceedings for BCI Conference*, 2017, in press.

[12] N. Saparkhojayev and A. Kanatbekkyzy, "Smart home assistant," *World Applied Sciences Journal*, vol. 28, no. 8, pp. 1075–1081, 2013.

[13] C. Carabelea and O. Boissier, "Multi-agent platforms on smart devices: Dream or reality," in *ECCBR Workshop on Case Based Reasoning and Personalisation*, 2002.

[14] C. Carabelea, O. Boissier, and F. Ramparany, "Benefits and requirements of using multi-agent systems on smart devices," in *Euro-Par 2003 Parallel Processing*, 2003, pp. 1091–1098.

[15] V. Terziyan, "Semantic web services for smart devices based on mobile agents," *Mobile Computing: Concepts, Methodologies, Tools, and Applications*, pp. 630–641, 2009.

[16] R. Tkaczyk, M. Ganzha, and M. Paprzycki, "Agentplanner - agent-based timetabling system - preliminary design and evaluation," in *2013 17th International Conference on System Theory, Control and Computing (ICSTCC)*, Oct 2013, pp. 795–800.

[17] ——, "Agentplanner - agent-based timetabling system," *Informatica*, vol. 40, no. 1, pp. 3–17, 2016.

[18] A. Solanas, C. Patsakis, M. Conti, I. S. Vlachos, V. Ramos, F. Falcone, O. Postolache, P. A. Perez-martinez, R. D. Pietro, D. N. Perrea, and A. Martinez-Balleste, "Smart health: A context-aware health paradigm within smart cities," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 74–81, Aug 2014.