Cooperative resource allocation in open systems of systems

Gerrit Anders, Alexander Schiendorfer, Florian Siefert, Jan-Philipp Steghöfer, Wolfgang Reif

Angaben zur Veröffentlichung / Publication details:

Anders, Gerrit, Alexander Schiendorfer, Florian Siefert, Jan-Philipp Steghöfer, and Wolfgang Reif. 2015. “Cooperative resource allocation in open systems of systems.” ACM Transactions on Autonomous and Adaptive Systems 10 (2): 11. https://doi.org/10.1145/2700323.

Cooperative Resource Allocation in Open Systems of Systems

GERRIT ANDERS, ALEXANDER SCHIENDORFER, and FLORIAN SIEFERT,
Institute for Software & Systems Engineering, Augsburg University, Germany
JAN-PHILIPP STEGHÖFER, Department of Computer Science and Engineering Chalmers | University of Gothenburg, Sweden
WOLFGANG REIF, Institute for Software & Systems Engineering, Augsburg University, Germany

Resource allocation is a common problem in many technical systems. In multi-agent systems, the decentralized or regionalized solution of this problem usually requires the agents to cooperate due to their limited resources and knowledge. At the same time, if these systems are of large scale, scalability issues can be addressed by a self-organizing hierarchical system structure that enables problem decomposition and compartmentalization. In open systems, various uncertainties—introduced by the environment as well as the agents' possibly self-interested or even malicious behavior—have to be taken into account to be able to allocate the resources according to the actual demand.

In this article, we present a trust- and cooperation-based algorithm that solves a dynamic resource allocation problem in open systems of systems. To measure and deal with uncertainties imposed by the environment and the agents at runtime, the algorithm uses the social concept of trust. In a hierarchical setting, we additionally show how agents create constraint models by learning the capabilities of subordinate agents if these are not able or willing to disclose this information. Throughout the article, the creation of power plant schedules in decentralized autonomous power management systems serves as a running example.

Categories and Subject Descriptors: I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multi-agent systems*

General Terms: Algorithms, Design, Performance

Additional Key Words and Phrases: Resource allocation problem, trust, uncertainty, open multiagent systems, systems of systems, constraint satisfaction optimization, electronic markets, smart grids

ACM Reference Format:
Gerrit Anders, Alexander Schiendorfer, Florian Siefert, Jan-Philipp Steghöfer, and Wolfgang Reif. 2015. Cooperative resource allocation in open systems of systems. ACM Trans. Autonom. Adapt. Syst. 10, 2, Article 11 (June 2015), 44 pages.
DOI: http://dx.doi.org/10.1145/2700323

1. RESOURCE ALLOCATION IN OPEN SYSTEMS

In many technical systems, such as power management systems, the basic task is to solve a *resource allocation problem* in which a number of system components have to provide a certain amount of resources in order to satisfy a specific demand [Chevaleyre et al. 2006]. This is often accomplished in a centralized and even optimal way. However, centralized approaches require detailed knowledge about the behavioral model of all system components and are intractable in large systems because the problem is

This research is sponsored by the research unit *OC-Trust* (FOR 1085) of the German Research Foundation.
Authors' addresses: G. Anders, A. Schiendorfer, F. Siefert, J.-P. Steghöfer, and W. Reif, Institute for Software & Systems Engineering, Augsburg University, Universitätsstr. 6a, 86159 Augsburg, Germany; emails: {anders, schiendorfer, siefert}@isse.de, jan-philipp.steghofer@cse.gu.se, reif@isse.de.

DOI: http://dx.doi.org/10.1145/2700323

NP-hard in the number of participants [Bar-Noy et al. 2001]. When giving the system components agency, that is, the ability to act autonomously in their environment, regionalized or decentralized approaches based on local decisions and coordination can be applied. These approaches are inherently suited for solving problems on the basis of imperfect information about the behavioral model and knowledge of other agents as well as the environment: information does not have to be disclosed to, pooled in, and kept consistent by some global entity. Since agents only have limited knowledge and capabilities, they have to rely on cooperation to achieve their own and the system's goals.

The willingness to cooperate, however, cannot be taken for granted, in particular in open Multi-agent Systems (MAS) [Artikis et al. 2009]. Such systems consist of a large number of heterogeneous agents that are embedded in a dynamic and potentially hostile environment. Since the agents' behavior and objectives are possibly not under the control of the system designer or other agents, only few assumptions can be made about them. As a consequence, the benevolence assumption, that is, the willingness to cooperate, has to be abandoned. On the other hand, the agents as well as the environment might show behaviors that can unintentionally put the system into jeopardy. As all this introduces uncertainty into the decision making process of each individual agent, without further measures, the agents might not be able to fulfill their own or the system's goals.

It is therefore crucial that the agents are able to identify and deal with these uncertainties at runtime. This is especially important when regarding mission-critical systems, such as power management systems. In such cases, the systems' resilience and dependability hinge on the ability to cope with uncertainties. The social concept of *trust*, more specifically *credibility*, has been proposed as a way to measure and deal with different kinds of uncertainties on the basis of experiences gained in the past [Steghöfer et al. 2010; Anders et al. 2014c]. In decision processes that use uncertain information, trust allows the agents to quantify the credibility of information sources and to use this data to form expectations of behavior. In incentive-compatible or trust-based mechanism design, trustworthy behavior is incentivized, for example, by providing compensation based on the trust others put in a participant [Dash et al. 2004], to promote cooperative and altruistic behavior.

Dynamic Resource Allocation Problems (DRAPs) that have to be solved in open MAS are a domain in which trust data are helpful. The DRAP we address in this article is dynamic in the sense that it is solved online in a running system and the circumstances under which the optimization takes place change. In particular, some of the information used to determine the future demand for resources stems from stochastic sources, which directly affects the objective function of the problem to be solved. This uncertainty has to be incorporated into the decision processes.

A further complication is introduced by the scale of open MAS. Even moderately small power management systems can consist of hundreds or thousands of prosumers (note that we use the term "prosumer" to refer to producers as well as consumers) that need to be integrated into a control system to ensure the system's basic functionality. Standard control approaches that use centralized models to solve problems such as the DRAP are no longer applicable to systems of such a size. Instead, a hierarchical decomposition of the task within a structure of systems of systems can be used effectively to split the problem into several smaller subproblems, whose collective solution provides a solution to the overall problem. In such a setting, the capabilities of agents within the hierarchical structure depend on the capabilities of their subordinates. However, these agents might not be able or willing to disclose all required information about their capabilities to their superordinate agent. Consequently, superordinate agents must be able to derive a model that describes their own capabilities, for example, from observations of their subordinates' behavior.

In this article, we introduce a **tru**st- and **c**ooperation-based **a**lgorithm for **o**pen MA**S**, called *TruCAOS*$^+$ (originally presented for nonhierarchical systems in Anders et al. [2013]), that allows one to solve the DRAP in large-scale systems by exploiting an existing hierarchical structure. The algorithm is based on the principle of electronic markets and thus uses cooperation between agents instead of centralized instances. Since we regard a system of systems, we limit cooperation to agents within the same subsystem. Each subsystem is represented on a higher level by an intermediary agent, that is, a single cooperation partner. To solve the DRAP, that is, to satisfy the demand, agents send proposals that contain predicted contributions to their superordinate intermediary. The intermediaries' goal is to assign the available resources according to the demand. Consequently, they have to decide which proposals to accept or reject. However, since we regard open MAS, the agents' predicted contribution of resources can (un)intentionally deviate from their actual contribution. The algorithm's trust-based principle allows the intermediaries to make informed decisions by taking the previously mentioned uncertainties into account. Further, it serves as a means to sanction misbehavior and incentivize cooperation. For situations in which agents do not provide their intermediary with all the necessary information about their capabilities, we outline a technique that allows intermediaries to learn constraint models through observation (along with a specific implementation using support vector data description). Such a behavioral model is orthogonal to trust models as they serve different purposes: The former approximates the possible controllability of an intermediary, while the latter captures the uncertainty inherent in the agent behavior and the environment.

Because of the DRAP's dynamic nature, TruCAOS$^+$ does not have to provide optimal solutions but solutions that are "good enough" and robust. Among other things, robustness can be achieved by proactively anticipating uncertainties, thereby improving the satisfaction of the demand. Another possibility is to search for solutions that increase the system's flexibility to react to unforeseen situations. In this article, we concentrate on the former characteristic, that is, solutions that are robust insofar as—despite the uncertainties—the system satisfies the demand as accurately as possible. This is achieved by using the concept of trust to (1) anticipate aleatoric, that is, intrinsic random and irreducible, uncertainties; (2) actively and effectively lower epistemic, that is, systematic, uncertainties; and (3) reduce the presence of aleatoric uncertainties. Both kinds of uncertainties result from the agents' as well as the environment's behavior. More precisely, the algorithm makes decisions that favor predictable agents, thereby putting the agents into states with less aleatoric uncertainties. This can be thought of as preferring a biased, thus more predictable, coin favoring tail with 90% to a perfectly fair, hence unpredictable, coin. Further, TruCAOS$^+$ reduces epistemic uncertainties by relying on trust-based expectations of the agents' and the environment's behavior instead of mere predictions. As more experiences are gained over time, the trust model improves and thus more knowledge about the underlying stochastic process influencing the agents' and the environment's behavior is acquired.

We give a basic formulation of the DRAP in Section 2, before we present our algorithm, TruCAOS$^+$, and show how it can be applied to systems of systems in Section 4 and Section 5. A major issue, the creation of models of the agents that participate in the scheme, is covered in Section 6. In Section 7, we show evaluation results and demonstrate that TruCAOS$^+$'s trust-based principle allows one to solve the DRAP despite uncertainties. Afterward, we discuss related work in Section 8. In Section 9, we conclude the article and give an outlook on future work. Throughout the article, the problem of creating power plant schedules in a decentralized autonomous power management system serves as a running example. This case study and the centralized approaches traditionally used in power management systems are introduced in Section 3.

2. A MODEL OF THE DYNAMIC RESOURCE ALLOCATION PROBLEM

We regard a DRAP that has to be solved by a set of possibly self-interested agents $\mathcal{A} = \{a_1, \ldots, a_n\}$, each with limited resources (see Section 2.3). The system's goal is to find an allocation so that, in each time step t, the agents make a *total* ***actual*** *contribution of resources* $A_{\mathcal{A}}[t]$ that matches an ***actual*** *demand for resources* $A_{env}[t]$ imposed by the environment *env* as accurately as possible. This means that the absolute deviation $|A_{env}[t] - A_{\mathcal{A}}[t]|$ between demand and contributions in a regarded time step t has to be minimized. While achieving this goal is paramount, the costs of providing these resources should be kept down.

This DRAP is a common problem in technical systems: In gas pipeline and water supply systems, for instance, the challenge is to maintain the system's pressure at a certain level [Mora et al. 2008; Dötsch et al. 2010]. Regarding district heating systems [Benonysson et al. 1995], in addition, the network temperature has to be kept between specific bounds. Similarly, the main task in power management systems is to maintain the balance between energy production and consumption at all times (see Section 1). Furthermore, in all these systems, the demand has to be satisfied despite fluctuations and unexpected events. We address this issue in Section 2.2.

2.1. Solving the Dynamic Resource Allocation Problem

We assume that the agents cannot arbitrarily change their contribution from one time step to another. Instead, their behavior is subject to inertia—a property which can be found in many systems that control physical devices, such as power generators or energy storage devices. As the agents' contribution might thus not change quickly enough to reactively adapt to the demand in all situations, their contribution has to be *proactively* fixed in the form of *schedules* beforehand. A schedule prescribes an agent's contribution for all time steps that are contained in the so-called *scheduling window* $\mathcal{W} = \{t_{\text{next}}, \ldots, t_{\text{now}} + H\}$. With regard to the current time step t_{now}, the scheduling window starts in the next time step $t_{\text{next}} = t_{\text{now}} + 1$. Its length corresponds to the *scheduling horizon* H.

Because a single agent cannot usually satisfy the environment's demand on its own, there is a need for cooperation. In order to cooperatively determine schedules that fulfill the goals at the macro (i.e., system) level, in each time step, each agent $a \in \mathcal{A}$ autonomously decides how much to contribute in future time steps $t \in \mathcal{W}$ by adjusting its own (internal and private) ***scheduled*** *contribution* $S_a[t]$ and announcing it in the form of a ***predicted*** *contribution* $P_a[t]$ (which might differ from its schedule). This decision is made on the basis of the predicted remaining demand $P_{env}[t] - P_{\mathcal{A}}[t]$, that is, the difference between the ***predicted*** *demand* $P_{env}[t]$ and the *total* ***predicted*** *contribution* $P_{\mathcal{A}}[t] = \sum_{a \in \mathcal{A}} P_a[t]$ in future time steps $t \in \mathcal{W}$. To achieve the system's goal, the agents adapt their scheduled contribution iteratively until the remaining demand is minimized. The remaining demand is therefore announced by the system in each iteration and time step. Feasible schedules can be described in terms of constraints over the decision variables representing the scheduled contribution per agent and time step (see Section 2.3).

2.2. Dynamics and Uncertainties

Because the actual demand $A_{env}[t]$ is based on a stochastic process driven by the environment (consider, e.g., meteorological influences), schedules are created on the basis of a predicted demand $P_{env}[t]$. Obviously, predicted and actual demand might thus differ. As we assume that the predictions become more accurate as the point in time t they are made for approaches, the agents have to adapt their schedules when predictions

change. A contribution scheduled for a time step t is thus not fixed but can be updated in the course of time.

The scheduled contribution $S_a[t]$ is the contribution a is actually *willing* to provide in t. However, for example, because of technical difficulties, $S_a[t]$ might *unintentionally* differ from a's ***actual*** *contribution* $A_a[t]$. In addition, since the agents might be self-interested, they might *intentionally* lie about their scheduled contribution $S_a[t]$. The predicted contribution $P_a[t]$ might therefore deviate from the scheduled $S_a[t]$ or actual $A_a[t]$ contribution. The same applies to the *total* ***actual*** *contribution* $A_{\mathcal{A}}[t] = \sum_{a \in \mathcal{A}} A_a[t]$, the *total* ***scheduled*** *contribution* $S_{\mathcal{A}}[t] = \sum_{a \in \mathcal{A}} S_a[t]$, and the total predicted contribution $P_{\mathcal{A}}[t]$.

Due to uncertainties introduced by inaccurate predictions, additional measures have to be taken. Because we assume that the system is able to observe the deviation between the predicted and the actual contribution or demand, it is possible to derive *expected values* for both. Thus, on the basis of experiences gained in prior time steps, an ***expected*** *demand* $E_{env}[t]$ can be derived from the demand $P_{env}[t]$ predicted by the environment, and a *total* ***expected*** *contribution* $E_{\mathcal{A}}[t] = \sum_{a \in \mathcal{A}} E_a[t]$ can be derived from predictions $P_a[t]$. This allows one, for example, to capture a permanently occurring prediction error in an expectation by adding a specific offset to the corresponding prediction. To create schedules that satisfy the actual demand as precisely as possible, the system thus announces the expected remaining demand $E_{env}[t] - E_{\mathcal{A}}[t]$.

Despite the agents' possibly self-interested behavior, at least a subset has to cooperate to be able to solve the system's task. On the one hand, that is because of the agents' limited resources; for example, if one of them decreases its contribution when solving the DRAP, others have to step in and raise their contributions in order to ensure that the balance between contributions and demand can be established. On the other hand, deviations from scheduled contributions can cause imbalances at runtime, which is why we need incentives to comply with agreed upon schedules. To incentivize agents to announce their actual contribution as accurately as possible, they receive higher rewards if the difference between the predicted and actual contribution is small. Hence, rational agents try to comply with their scheduled contribution and to support the system to achieve its goal.

2.3. Control Models

Each agent a's actual contribution $A_a[t]$ is subject to a minimum contribution $A_a^{\min}$ and maximum contribution $A_a^{\max}$. The property of inertia is regarded by additional restricting functions in the way that a's minimum contribution $\overrightarrow{A}_a^{\min}(A_a[t-1])$ and maximum contribution $\overrightarrow{A}_a^{\max}(A_a[t-1])$ for a time step t depends on a's contribution $A_a[t-1]$ in the previous time step $t-1$.[1] Therefore, we can further restrict a's contribution for a future time step t to a certain range L_a^t, which has to be recomputed in every time step. Because this range might be discontinuous, we specify L_a^t as a list of allowed intervals (see Section 6 for details). We assume that the agents' characteristics are attributes of the system and cannot be changed from outside. While they constrain the agents' actual behavior, the agents also have to regard these constraints when deciding about their scheduled contribution (however, they may still feel free to lie about their scheduled contributions). Recalling that the agents can represent physical devices such as generators, we refer to these constraint models as *control models* in the following.

[1] In the following, we refer to these functions also as (state-dependent) *min./max. successor functions*.

2.4. Extension to Hierarchical Systems

The DRAP is NP-hard; both with regard to the number of agents involved $|\mathcal{A}|$ and the number of time steps H schedules are created for in advance (see Section 1). Since it is not known whether there exists an efficient (polynomial) algorithm for this class of problems, we have to assume a complexity of $O(2^{|\mathcal{A}|\cdot H})$. To deal with the large number of agents typical for open MAS, we use hierarchical decomposition, meaning that the DRAP that is defined at the macro level and has to be solved by the entire system is decomposed into DRAPs that have to be solved at the meso level. The hierarchy constitutes a tree and is formed by the agents $\mathcal{A}$. In this hierarchy, agents that are not a leaf act as *intermediaries* $\mathcal{I} \subset \mathcal{A}$. Each intermediary $\lambda \in \mathcal{I}$ represents the group of its directly subordinate agents $\mathcal{A}_\lambda \subseteq \mathcal{A}$. Since we regard a hierarchy, $\mathcal{A}_\lambda$ might contain other intermediaries. Together, all agents represented by an intermediary form a subsystem within the overall hierarchical system of systems. Since the hierarchy is given by a tree, the sets of agents $\mathcal{A}_\lambda$ directly controlled by intermediaries $\lambda \in \mathcal{I}$ are disjoint. The hierarchy's root is represented by the so-called top-level intermediary $\Lambda \in \mathcal{I}$.

To satisfy the overall demand in this system of systems, each of the intermediaries has to solve a DRAP as described previously. While the top-level intermediary tries to satisfy the overall expected demand of the environment $E_{env}[t]$, all the other intermediaries have to ensure that their subordinate agents create schedules and, ultimately, make contributions that satisfy a specific fraction of the overall demand. This fraction is equal to the contribution $S_\lambda[t]$ they scheduled as they took part in the solution of the DRAP of their superior intermediary. While agents $a \in \mathcal{A} \backslash \mathcal{I}$ that are not intermediaries contribute directly, all intermediaries recursively distribute their contribution to the agents they represent. This necessitates cooperation between intermediaries and their subordinate agents.

Solving the DRAP in this hierarchical and top-down approach enables scalability with respect to the number of participating agents. However, since intermediaries participate in the resource allocation of their superior just like an agent that can contribute directly, they have to determine their possible contributions before having knowledge of the predicted contributions of their directly subordinate agents. Because the minimum and maximum contribution as well as the inertia of an intermediary λ depend on the individual capabilities of the agents $\mathcal{A}_\lambda$, the intermediary's behavior is subject to the constraints identified in Section 2.3. As we do not assume that all agents can or will voluntarily disclose their control models, or that such models exist in a format that can be interpreted by the intermediary, the aforementioned characteristics require each intermediary λ to create a control model for every agent $a \in \mathcal{A}_\lambda$ by observing and analyzing its behavior in the course of previous solutions of the DRAP (see Section 6). Evidently, this would not be necessary in a bottom-up approach as will be discussed in Section 6.

2.5. Formalization of the Optimization Problem

The DRAP that has to be solved by each intermediary $\lambda \in \mathcal{I}$ and its subordinate agents $\mathcal{A}_\lambda$ is a Constraint Satisfaction Optimization Problem (CSOP) [Tsang 1993] composed of a set of variables, their domains (which we do not assume to be *finite*), constraints that restrict valid assignments, and an objective function. Under the assumption that an intermediary has full knowledge about and control over its directly subordinate agents, this CSOP can be formulated as follows:

$$\begin{aligned} &\underset{S_a[t]}{\text{minimize}} \;\; \alpha_\Delta \cdot \Delta + \alpha_\Gamma \cdot \Gamma \qquad (1) \\ &\text{subject to} \;\; \forall a \in \mathcal{A}_\lambda, \forall t \in \mathcal{W} : \exists [x, y] \in L_a^t : x \leq S_a[t] \leq y, \end{aligned}$$

$$\overrightarrow{A}_a^{\min}(S_a[t-1]) \leq S_a[t] \leq \overrightarrow{A}_a^{\max}(S_a[t-1])$$
$$\text{with } \Delta = \sum_{t \in \mathcal{W}} \left| S_{\mathcal{A}_\lambda}[t] - S_\lambda[t] \right|, \Gamma = \sum_{a \in \mathcal{A}_\lambda,\ t \in \mathcal{W}} \kappa_a(S_a[t]).$$

In this optimization problem, the scheduled contribution $S_a[t]$ of agents $a \in \mathcal{A}_\lambda$ for time steps $t \in \mathcal{W}$ are the decision variables. The intermediaries' objective is to negotiate allocations with its agents $\mathcal{A}_\lambda$ so that the intermediaries' fraction $S_\lambda[t]$ of the expected overall demand $E_{env}[t]$ is met as closely and cost effectively as possible (in the case of the top-level intermediary Λ, $S_\Lambda[t]$ equals $E_{env}[t]$). Needless to say, these objectives can be conflicting. This is where the expressions Δ and Γ come into play. Δ sums up the absolute deviation between $S_\lambda[t]$ and the combined contribution $S_{\mathcal{A}_\lambda}[t]$ over all regarded time steps t (since we sum up *absolute* deviations, negative and positive deviations cannot cancel each other out), whereas Γ calculates the total costs of the contribution as allocated (using agent-specific cost functions $\kappa_a(S_a[t])$). We can weight these goals with the parameters α_Δ and α_Γ and thereby establish a lexicographical order with regard to the demand's satisfaction (primary objective) and the costs (secondary objective) as described at the beginning of Section 2.

In Section 4, we show how intermediaries solve this CSOP *without* assuming full knowledge about and control over their directly subordinate agents.

3. RUNNING EXAMPLE: PROSUMER SCHEDULING IN AUTONOMOUS POWER MANAGEMENT SYSTEMS

In autonomous power management systems, current manual or semiautonomous scheduling—that is, the determination of the output levels of dispatchable prosumers,[2] such as coal and gas power plants or electric vehicles, for future time steps—is replaced by a fully autonomous mechanism. To be able to proactively participate in the creation of schedules and in maintaining the stability of the grid, we require prosumers to act as agents on their own behalf (see, e.g., McArthur et al. [2007] and Ramchurn et al. [2012]). Ideally, not only large power plants are used in the scheduling process (as is the case today), but small prosumers, such as micro-CHP units or biogas plants, which are usually under the control of individuals or small cooperatives, participate as well. As a result, the load of nondispatchable consumers can be satisfied locally, alleviating the need for complex distribution networks and allowing one to locally deal with complications such as voltage band constraints [Blank et al. 2011].

Such a scheme, however, puts a lot of strain on the computational system as hundreds or thousands of dispatchable prosumers have to be integrated. Since scheduling is an NP-hard problem with regard to the number of prosumers involved (see Section 2.4), scalability becomes a major concern for systems of this size. The introduction of intermediaries, forming a hierarchy [Steghöfer et al. 2013b] of so-called *Autonomous Virtual Power Plants* (AVPP) [Steghöfer et al. 2013a] as shown in Figure 1, allows the regionalization of scheduling and to scale the system to the number of agents required. Within an AVPP, the schedules of the controlled prosumers can be created in a centralized fashion. For this purpose, the AVPP must know the physical and economic limitations of the prosumers and combine them in a model. This model can then be solved by centralized optimization methods (see, e.g., Heo et al. [2006] and Zafra-Cabeza et al. [2008]). We call this approach *regio-central* since it combines regionalized information within the AVPP with a centralized scheduling algorithm.

[2]The physical constraints of dispatchable prosumers are usually a *superset* of those given in Section 2.3.

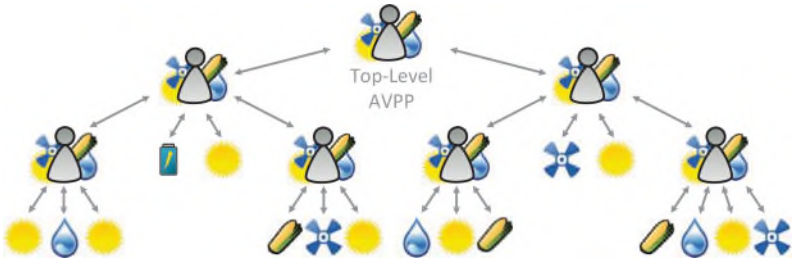

Fig. 1. Hierarchical system structure of a future autonomous power management system: Prosumers are structured into systems of systems represented by AVPPs acting as intermediaries, thereby decreasing the complexity of control and scheduling. AVPPs can be part of other AVPPs.

3.1. Decentralized Hierarchical Prosumer Scheduling

While centralized scheduling allows for very good solutions, has been investigated by the research community, and is in active use in the industry for large dispatchable power plants, it suffers from a number of drawbacks. Foremost, scheduling requires a model of each individual prosumer. Small cooperatives or individuals that own dispatchable prosumers, however, might have no interest in disclosing all information necessary to create adequate schedules to the organization that performs this task. But even if they are willing to disclose their control model, it is still possible that such a model is not available in a format that can be incorporated into the organization's decision making process (see Section 2.4). Cooperation-based approaches, such as TruCAOS$^+$ presented in this article (see Section 4), allow prosumers to formulate valid schedules without having to disclose their internal models. Minimizing the time needed to solve the DRAP with regionalized approaches requires the system to find a sweet spot between the size of AVPPs and the depth of the hierarchy [Anders et al. 2014a]. As deep hierarchies and rather small AVPPs fragment the system and introduce indirections, the utmost goal to hold the balance between production and consumption might not be properly achieved (see Schiendorfer et al. [2014] and Section 7.3.2). Cooperation-based approaches have the potential to scale better with the number of prosumers schedules have to be created for, thereby allowing the formation of greater AVPPs and flatter hierarchies, thus preventing fragmentation (see Section 7.3.2). Of course, such an approach has a need for increased communication and only yields suboptimal solutions, but since the scheduling problem is an instance of the DRAP presented in Section 2, optimal solutions are not necessary (see Section 1).

As explained in Section 2.4, each group of prosumers participating in the scheme constitutes a subsystem that is represented by an intermediary agent, that is, an AVPP. Each AVPP—except for the top-level AVPP—is, in turn, part of a superordinate group of prosumers. Within this hierarchical structure, each subsystem is transparent for superordinates to ensure scalability. The intermediary in the form of an AVPP thus acts as a regular prosumer and, therefore, as a participant in the cooperation scheme on the higher level. Since each subsystem only has to solve a partial problem, this approach allows dealing with large systems, while an overall resource allocation is still achieved.

With respect to the DRAP, in the scheduling problem, the actual demand for resources $A_{env}[t]$ corresponds to the *residual load*, that is, the load that has to be fulfilled by the dispatchable prosumers. It is calculated from the difference between overall nondispatchable load minus the output of intermittent power plants, such as wind turbines and solar power plants. The total actual contribution $A_A[t]$ is the sum of the

actual output of all dispatchable prosumers that participate in the scheduling process.[3] Each prosumer a participates with its actual output $A_a[t]$. The aim of the scheduling is to stipulate the future output $S_a[t]$ for all dispatchable prosumers a such that the deviation $|A_{env}[t] - A_A[t]|$ between the actual residual load and the actual output of prosumers is minimal. In contrast to the general formulation in Section 2, in centralized scheduling mechanisms, $S_a[t]$ is not autonomously defined by the prosumers but externally by the scheduling mechanism. Nowadays, it is usually assumed that prosumers are benevolent and stick to their schedule as closely as possible, even though $A_a[t]$ might differ from the scheduled output $S_a[t]$.

3.2. Uncertainties and Trust

In power management systems, a number of uncertainties, both originating in the system itself as well as in the environment, complicate its stable operation [Ramchurn et al. 2012]. Because power management systems are an example of mission-critical systems, their failure can have massive consequences for people, industries, and public services. It is therefore of utmost importance that they are stable and available at all times. There are three main sources of uncertainty: the actual nondispatchable load, the actual output of intermittent power plants, as well as the ability and willingness of dispatchable prosumers to stick to their schedules. In this article, we regard the residual load as part of the environment *env*. Since the former two factors both contribute to the residual load, they are subsumed in this stochastic process. The actual $A_{env}[t]$ and predicted $P_{env}[t]$ residual load is thus calculated from the actual behavior of and predictions made by nondispatchable consumers and intermittent power plants.

As mentioned in Section 2.2 and previously, $S_a[t]$ does not have to equal $P_a[t]$ or $A_a[t]$ if agent autonomy permits the agents more freedom in their choices. Consequently, predictions give only an indication of the actual output and residual load. To create robust schedules, that is, schedules that allow the system to satisfy the residual load despite uncertainties, the accuracy and credibility of the predictions has to be taken into account. As outlined in Section 5, we propose to use the concept of *trust* as a measure of prediction credibility in order to solve the DRAP on the basis of an expected demand $E_{env}[t]$ and expected contributions $E_a[t]$. Trust values are traditionally used to determine an interaction partner whose behavior is most likely to yield a beneficial result based on the prior experiences with the agent [Keung and Griffiths 2008]. This notion is reinterpreted here: Each agent interacts by providing predictions, the accuracy of the prediction is determined by comparing actual and predicted behavior, and the resulting experiences are used to estimate the accuracy of future predictions.

An additional complication arises by the fact that predictions can change at runtime. Changes in the status of system components, for example, can change the predictions of nondispatchable consumers and intermittent power plants which, in turn, changes the residual load prediction. Therefore, schedules have to be continuously adjusted.

4. THE TRUST- AND COOPERATION-BASED ALGORITHM

In this section, we propose TruCAOS$^+$, a market-based algorithm that solves the DRAP introduced in Section 2 in a cooperative and regionalized manner. With regard to our running example (see Section 3), each intermediary, that is, AVPP, satisfies a part of the overall residual load by allocating available resources of directly subordinate dispatchable prosumers. A market-based approach seems to be a natural choice because it inherently allows prosumers to keep their control model private, and electricity infeed as well as consumption is already subject to monetary rewards and costs. In open

[3]We assume that sealed and secure equipment, similar to smart meters, is available to measure the actual load and output.

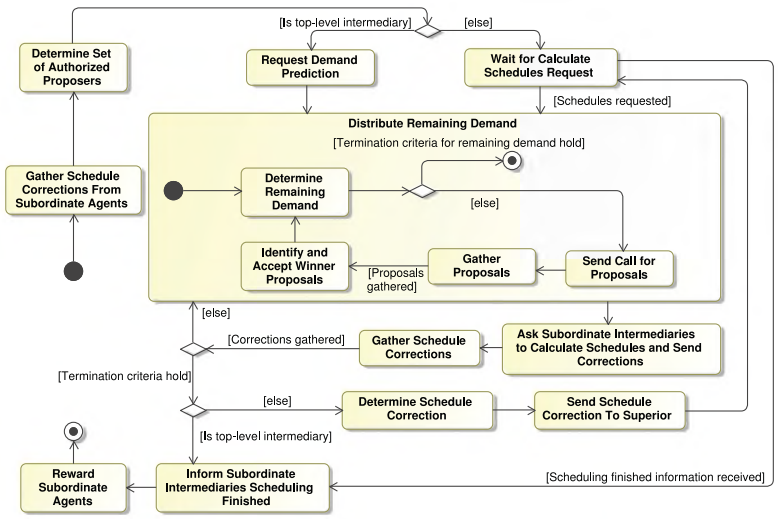

Fig. 2. TruCAOS$^+$'s basic procedure.

systems, the former characteristic is of particular importance as we cannot assume that the agents are willing to disclose information about their internal state. Even if they did, other agents would not know if this information is correct.

In its simplest form, TruCAOS$^+$ does not include any measures of prediction accuracy and does not allow intermediaries to identify agents that make erroneous proposals. To illustrate TruCAOS$^+$'s basic procedure (see Section 4.1), we therefore initially assume benevolent behavior (i.e., scheduled, predicted, and actual contributions or demands are identical). In Section 5, we abandon the benevolence assumption and extend TruCAOS$^+$ by trust-based measures, allowing one to find robust solutions in open environments (e.g., by forming expectations of behavior).

4.1. Basic Procedure

In principle, in each time step, all agents (in our case study, dispatchable prosumers) satisfy a part of the predicted demand by selling or buying resources to or from their superordinate intermediary (in our case study, the AVPP that controls dispatchable prosumers). This is done in an iterative and incremental process (see Figure 2) that, in its basic form, is reminiscent of an iteratively performed first-price sealed-bid auction (see, e.g., Klemperer [2002]).

Since the agents' own scheduled contribution should always reflect behavior that is feasible with respect to their control model (see Section 2.3), all intermediaries ask their subordinate agents to perform and communicate schedule corrections before this iterative and incremental process is started in the current time step t_{now}. More details on why this is necessary and how this step is accomplished are given in Section 4.1.4.

Since TruCAOS$^+$ calculates schedules in a top-down manner, the distribution of the demand is triggered by the top-level intermediary Λ by requesting demand predictions $P_{env}[t]$ for the next H time steps $t \in \mathcal{W} = \{t_{\text{next}}, \ldots, t_{\text{now}} + H\}$ from the environment (in our case study, the environment is represented by nondispatchable consumers and intermittent power plants). As stated in Section 2.4 and Section 2.5, all other intermediaries have to distribute the demand they promised to satisfy as they took part in the solution of the DRAP of their superior intermediary. As we can think of the predicted demand as the scheduled contribution of the top-level intermediary ($S_\Lambda[t] = P_{env}[t]$),

the demand that has to be distributed by each $\lambda \in \mathcal{I}$ in a specific time step t equals its scheduled contribution $S_\lambda[t]$.

4.1.1. Distribute Remaining Demand. In all *bidding iterations* $j \in \mathcal{J}$ in which this demand is distributed by an intermediary λ, it announces an auction in which a part of the *remaining demand* $R_\lambda[t] = S_\lambda[t] - P_{\mathcal{A}_\lambda}[t]$ (note that $R_\lambda[t]$ might also be negative in specific time steps and that $P_{\mathcal{A}_\lambda}[t]$ is the total predicted contribution of the set of λ's directly subordinate agents $\mathcal{A}_\lambda$) should be satisfied by allocating additional resources of at least one of the subordinate agents.[4] The corresponding *call for proposals* $CFP = (\langle R_\lambda[t_{\text{next}}], \ldots, R_\lambda[t_{\text{now}} + H]\rangle, \mathcal{W})$ contains a sequence of the remaining demand values as well as the regarded scheduling window $\mathcal{W}$. The call for proposals is sent to all *authorized proposers* $\mathcal{A}_{\text{auth}}$. In TruCAOS$^+$'s basic form, all agents are allowed to participate in the auction, that is, $\mathcal{A}_{\text{auth}} = \mathcal{A}$. In Anders et al. [2013], we showed how this set can be restricted to trustworthy agents, thereby incentivizing rational agents to behave benevolently. Each agent a that receives the call and wants to sell or buy resources to or from λ responds with a proposal, that is, a ***proposed*** *schedule*, $P_a^* = (\langle P_a^*[t_{\text{next}}], \ldots, P_a^*[t_{\text{now}} + H]\rangle, \overline{\kappa}_a^*)$ that includes a sequence of predicted contributions $P_a^*[t]$ as well as the *average costs* $\overline{\kappa}_a^*$, that is, a remuneration, for providing a contribution.[5] While the costs of the contribution specified in a proposed schedule P_a^* for a specific time step t are $\kappa(P_a^*, t) = \overline{\kappa}_a^* \cdot |P_a^*[t]|$, the total costs of P_a^* are defined as the sum $\kappa(P_a^*) = \sum_{t \in \mathcal{W}} \kappa(P_a^*, t)$ of the contributions' costs for all time steps in the scheduling window. The intermediary gathers all proposals received in this bidding iteration in a set $\mathcal{P}^*$ and completes this bidding iteration by identifying and accepting one or more suitable winner proposals $\mathcal{P}_W^* \subseteq \mathcal{P}^*$ (see Section 4.2). All other proposals are rejected.

For each agent a, its *most recently* accepted proposal P_a^* defines its ***predicted*** *schedule* $P_a = (\langle P_a[t_{\text{next}}], \ldots, P_a[t_{\text{now}} + H]\rangle, \overline{\kappa}_a)$ for the next H time steps. P_a is thus a contract between a and its superordinate intermediary: a has to comply with its prediction in exchange for remuneration (see Section 4.1.3). For this reason, whenever a wants to change its contribution, it has to make a proposal P_a^* that is accepted by λ and in turn replaces P_a. Hence, if P_a^* is not accepted, it does not change a's predicted schedule P_a. Note that P_a is created in the course of multiple time steps if $H > 1$. In case the most recently accepted proposal P_a was accepted in a previous time step, λ assumes that, for each future time step $t' \in \mathcal{W}$ that is not covered by P_a^*, a does not contribute, that is, $P_a[t'] = 0$. Moreover, in case there is no most recently accepted proposal P_a, we assume that $P_a[t] = 0$ and $\overline{\kappa}_a = 0$ for all $t \in \mathcal{W}$. Be aware that this default setting might actually not be feasible, for example, if $A_a^{\min} > 0$. In such a case a has to *correct* this infeasible assignment by means of a so-called *schedule correction* that is guaranteed to be accepted by the intermediary. Section 4.1.4 covers this issue in detail.

Each agent a creates its proposals with respect to the constraints introduced in Section 2.3 and its current state. Further, it has to take its scheduled contributions $S_a[t]$ as well as the information about the remaining demand included in the call for proposals into account. Basically, agents generate proposals by solving a CSOP that is very similar to the one specified in Section 2.5. Differences between the CSOP presented in Section 2.5 and those solved by the agents can stem from additional individual technical constraints and—because an agent might change its behavior at runtime—variable objectives. However, since each agent now only has to determine the scheduled contribution for itself, the CSOP is not subject to the scalability issues discussed earlier:

[4]Actually, only a fraction of the remaining demand is distributed to subordinate agents in each bidding iteration. While this measure increases fairness with respect to the contribution each agent makes in the end (and, in particular, the contribution capacity it has), for the sake of simplicity, we deliberately describe the algorithm without making explicit use of this concept.

[5]If $\overline{\kappa}_a^* < 0$, agent a offers money to its intermediary λ.

The complexity of solving the DRAP for a subsystem $\mathcal{A}_\lambda$ is only $O(j_{\text{aggr}} \cdot (|\mathcal{A}_\lambda| \cdot 2^H + 2^{|\mathcal{A}_\lambda|}))$ (where j_{aggr} is the aggregated number of iterations needed to solve the DRAP and $O(2^{|\mathcal{A}_\lambda|})$ the complexity of selecting winner proposals in the regarded subsystem) instead of $O(2^{|\mathcal{A}_\lambda| \cdot H})$ (see Section 2.4).[6] This even holds for an intermediary if it uses an abstracted control model representing the aggregated capabilities of all subordinate agents instead of their individual capabilities (see Section 6 for more details).

The iterative process of distributing the remaining demand terminates either if it is sufficiently satisfied, that is, if its absolute values are below a predefined threshold $R_{\max}$ (i.e., $\forall t \in \mathcal{W} : |R_\lambda[t]| \leq R_{\max}$), or if λ did not receive any proposals (i.e., $\mathcal{P}^* = \emptyset$), or if there does not exist a suitable proposal, or if a maximum number of bidding iterations $j_{\max}$ is exceeded.

4.1.2. Ask Subordinate Intermediaries to Calculate Schedules and Send Corrections. Subsequent to the bidding phase, the intermediary asks all directly subordinate intermediaries to solve the DRAP for their own subsystems (see Section 4.1.2). Whenever an intermediary receives such a request, it starts the procedure outlined in Section 4.1.1, that is, it distributes its remaining demand in a number of bidding iterations until at least one termination criterion holds.

In TruCAOS$^+$, intermediaries (except for the top-level intermediary) allocate resources when taking part in the solution of the DRAP of its superior *before* they know which contributions their subordinate agents actually can or want to make. To be able to make feasible proposals when taking part in the solution of the DRAP of their superior, intermediaries thus have to create approximations of the control models of their subordinate agents by observing their actual behavior as well as their behavior in the course of the auctions (in Section 6, we explain how these approximations are created in detail). However, since such a learned model might not perfectly reflect an agent's capabilities, it is possible that the demand cannot be completely satisfied at the end of the bidding iterations. In such a situation, an intermediary has to inform its superior about the mismatch between the allocated resources of its subordinate agents and its own predicted contribution, that is, the final remaining demand. Otherwise, it could not adhere to its predicted schedule. This information is sent in the form of a *schedule correction* $C_\lambda = (C_\lambda[t_{\text{next}}], \ldots, C_\lambda[t_{\text{now}} + H])$ to the superior. The predicted P_λ and scheduled S_λ contribution are modified accordingly by adding the different elements $C_\lambda[t]$ to $P_\lambda[t]$ and $S_\lambda[t]$. Furthermore, the superior adjusts the average costs $\overline{\kappa}_\lambda$ of P_λ in such a way that P_λ's total costs decrease by $\sum_{t \in \mathcal{W}} |\overline{\kappa}_\lambda| \cdot |C_\lambda[t]|$. Agents are thus incentivized to avoid corrections. Having sent a schedule correction, an intermediary waits for further instructions from its superior.

After collecting the corrections of all subordinate intermediaries, the intermediary adds the corrections to the remaining demand and adjusts the predicted contributions as described previously. If the remaining demand has changed, is not satisfied, and the maximum number of *correction iterations* $k_{\max}$ is not exceeded, the intermediary starts a new bidding phase as described in Section 4.1.1 to reallocate the received corrections. This procedure ensures that corrections made at lower levels do only propagate to the next higher level if other subordinate agents are able to adjust their contributions. This is advantageous as it decreases the probability that, first, the intermediary has to send a correction to its superior (which would reduce its reward) and, second, that more agents are incorporated into the solution of the problem. Moreover, due to limiting the locality to one intermediary first, several corrections can be dealt with in parallel before escalating to the next higher level.

[6]It should be noted that, in both cases, additional costs result from the creation of control models. Depending on their level of detail, these costs range between $O(|\mathcal{A}_\lambda| \cdot H)$ and $O(2^{2|\mathcal{A}_\lambda|})$ (see Section 6.2).

In case of the top-level intermediary and satisfied termination criteria, the top-level intermediary informs its subordinate intermediaries that the schedule calculation is finished (see Section 4.1.3). If the intermediary is not the top-level intermediary, it waits for further instructions from its superior.

4.1.3. Inform Subordinate Intermediaries about Termination and Reward Agents. The information that the calculation of schedules is finished originates from the top-level intermediary and propagates downward with respect to the hierarchy. Each intermediary that receives the information that the calculation is finished forwards this information to its subordinate intermediaries.

Finally, the intermediary distributes a reward r_a to each subordinate agent $a \in \mathcal{A}_\lambda$ for its contribution in the *previous* time step $t_{\text{now}} - 1$. The reward depends on a's predicted contribution $P_a[t_{\text{now}} - 1]$ and the average costs $\overline{\kappa}_a$ stipulated in the corresponding schedule P_a that was valid in $t_{\text{now}} - 1$:

$$r_a = \kappa(P_a, t_{\text{now}} - 1) = \overline{\kappa}_a \cdot |P_a[t_{\text{now}} - 1]| \,.$$

Note that agents must bear the costs resulting from corrections since these affect their net reward. As agents might not adhere to their predicted contribution, we redefine the reward in Section 5.2.1.

4.1.4. Gather Schedule Corrections from Subordinate Agents. As stated in Section 4.1.1, for each future time step $t' \in \mathcal{W}$ that is not covered by the most recently accepted proposal P_a, it is assumed that the agent a does not contribute (i.e., $P_a[t'] = 0$). However, this contribution might not be feasible for a if it contradicts its control model, for example, its inertia or minimum or maximum contribution. Each agent therefore has the opportunity to adjust its scheduled and predicted contribution by sending a schedule correction C_a to its intermediary *before* the iterative process of distributing the remaining demand is started. In contrast to Section 4.1.2, only those contributions can be adjusted that are not already covered by the most recently accepted proposal P_a.

These corrections are created in a bottom-up manner, meaning that all leaves in the hierarchy create and send their corrections to their superior intermediary. An intermediary, in turn, aggregates the corrections of its subordinate agents and forwards this aggregated correction to its own superior.

Again, the predicted and scheduled contributions are modified according to the correction. The average costs $\overline{\kappa}_a$ stipulated in P_a are changed so that P_a's total costs only increase by $\sum_{t \in \mathcal{W}} \frac{1}{2} \cdot \overline{\kappa}_a^{\min} \cdot C_a[t]$. Because this growth is based on $\overline{\kappa}_a^{\min}$ (i.e., the minimum average costs of the agent's accepted proposals) multiplied by the decreasing (arbitrary) constant $\frac{1}{2}$ and thus strictly less than $\overline{\kappa}_a^{\text{avg}}$ (i.e., the mean of the agent's average costs of accepted proposals) if $\overline{\kappa}_a^{\text{avg}} > 0$, the agent is incentivized to participate in the auctions where it can assess higher costs for its contributions instead of using corrections to increase its reward.[7]

4.2. Identify and Accept Winner Proposals

The regarded DRAP is a multiobjective optimization problem. While satisfying the demand as accurately as possible is the primary goal, the costs of providing the necessary resources should be kept down (see Section 2.5). TruCAOS$^+$ solves the DRAP by means of a multistage decision process that is based on heuristics. It is evident that this procedure does not allow one to find optimal solutions. But, as stated in Section 1,

[7]For the sake of simplicity, we assume here that $\overline{\kappa}_a^{\min} > 0$. The minimum average costs $\overline{\kappa}_a^{\min}$ as well as the mean of the average costs $\overline{\kappa}_a^{\text{avg}}$ are calculated over a fixed number of the last time steps.

Fig. 3. The steps performed for "Identify and Accept Winner Proposals" (see Figure 2).

we only need solutions that are "good enough" and robust in the sense that the system can satisfy the demand despite uncertainties.

The intermediary determines the set of winner proposals $\mathcal{P}^*_W \subseteq \mathcal{P}^*$ on the basis of a three-step process depicted in Figure 3. First, invalid proposals are filtered, then the quality of the remaining proposals is assessed and low-quality proposals are sorted out, and finally the best suited proposals are accepted:

(1) *Filter Invalid Proposals (see Section 4.2.1):* Proposals that do not improve the overall satisfaction of the total remaining demand by a minimum value are sorted out.
(2) *Filter Low-Quality Proposals (see Section 4.2.2):* The proposals' quality is assessed with regard to their price-performance ratio. Only those proposals whose quality is greater than a dynamic threshold pass this filtering stage. If no such proposal exists, the best-rated proposal is accepted in the third step to allow the algorithm to make any progress.
(3) *Determine Winner Proposals (see Section 4.2.3):* The set of winner proposals is determined by solving an optimization problem in which the gain in satisfaction of the total remaining demand is maximized. According to the DRAP's definition in Section 2, this objective should be achieved with minimal costs.

Proposals that do not pass the filtering stages or are not selected as a winner proposal are rejected. Having introduced the basic idea of each of the three steps, we explain each of them in more detail in the following subsections.

4.2.1. Filter Invalid Proposals. To satisfy the demand, only those proposals $P^*_a \in \mathcal{P}^*$ that would improve the overall satisfaction of the total remaining demand by a minimum value of $imp_{\min} > 0$ are regarded as valid and pass the filtering stage. To identify which proposals improve the satisfaction, we define $R^*_\lambda(P^*_a, t) = R_\lambda[t] + P_a[t] - P^*_a[t]$ as the remaining demand for time step t that would result in case P^*_a was accepted and thus replaced P_a. The resulting improvement is called the *gain in satisfaction* $\mathcal{G}(P^*_a)$ and is defined as follows:

$$\mathcal{G}(P^*_a) = \sum_{t \in \mathcal{W}} |R_\lambda[t]| - \left|R^*_\lambda(P^*_a, t)\right|.$$

As stated previously, a proposal P^*_a is valid if $\mathcal{G}(P^*_a) \geq imp_{\min}$ holds.

4.2.2. Filter Low-Quality Proposals. In order to keep the costs down, the quality $\mathcal{Q}(P^*_a)$ of each proposal P^*_a is evaluated with respect to its gain in satisfaction as well as the change in costs $\Delta\kappa(P^*_a)$ that would result in case P^*_a was accepted and replaced P_a. The quality $\mathcal{Q}(P^*_a)$ is thus defined as a tuple:

$$\mathcal{Q}(P^*_a) = (\mathcal{G}(P^*_a), \Delta\kappa(P^*_a))$$
$$\text{with } \Delta\kappa(P^*_a) = \sum_{t \in \mathcal{W}} \kappa(P^*_a, t) - \kappa(P_a, t).$$

To compare different proposals with regard to the previously mentioned quality criteria, we define a total order over the proposals where we distinguish proposals based on their cost impact. The previous decision stage guarantees that the gain in

satisfaction of all proposals is positive in this stage (see Section 4.2.1). All proposals P_a^* with $\Delta\kappa(P_a^*) > 0$ (i.e., the proposals that offer a better satisfaction for higher costs) can therefore be sorted according to their *price-performance ratio* $\Theta(P_a^*)$, which is defined as the ratio of its gain in satisfaction $\mathcal{G}(P_a^*)$ to its change in costs $\Delta\kappa(P_a^*)$:

$$\Theta(P_a^*) = \frac{\mathcal{G}(P_a^*)}{\Delta\kappa(P_a^*)}.$$

The greater the price-performance ratio, the higher the quality of the proposal. While this measure allows smaller agents with very limited resources to compete with larger agents, it might increase the number of bidding iterations needed to satisfy the demand. In case the change in costs $\Delta\kappa(P_a^*)$ of a proposal is not positive (i.e., we get a better satisfaction while not facing *any* additional costs but possibly even cost reductions), we do not assess the proposal's quality on the basis of its price-performance ratio but define that the quality is higher, the smaller $\Delta\kappa(P_a^*)$. We further specify that a proposal with $\Delta\kappa(P_a^*) \leq 0$ is better than any proposal with positive change in costs.

Summarized, these properties yield a quality relation $P_{a_1}^* \succ P_{a_2}^*$. In this more formal description, $P_{a_1}^* \succ P_{a_2}^*$ means that a proposal $P_{a_1}^*$ is "better" than a proposal $P_{a_2}^*$ if and only if the following condition holds:

$$(\Delta\kappa(P_{a_1}^*) \leq 0 \wedge \Delta\kappa(P_{a_1}^*) < \Delta\kappa(P_{a_2}^*)) \vee (\Delta\kappa(P_{a_1}^*) > 0 \wedge \Delta\kappa(P_{a_2}^*) > 0 \wedge \Theta(P_{a_1}^*) > \Theta(P_{a_2}^*)).$$

One possibility to ensure that the quality of accepted proposals is sufficiently high is to accept only one proposal per bidding iteration. Obviously, this procedure would lead to a high number of bidding iterations and, consequently, would increase the number of messages sent and proposals to generate. To allow intermediaries to accept multiple proposals in a single bidding iteration and, at the same time, to keep the quality of accepted proposals sufficiently high, each intermediary keeps track of the proposals it accepted in a fixed number of the last time steps and evaluates the average of each quality criterion separately. Based on this information, only those proposals whose quality is not worse than the average quality of accepted proposals can be accepted in the next decision stage (see Section 4.2.3). While this procedure certainly does not yield minimum costs, it assures that the average quality does not decrease. Furthermore, the overhead introduced by additional bidding iterations and communication can be reduced. In case no proposal is rated better than the average, the best proposal passes this filtering stage to allow the algorithm to make any progress.

4.2.3. Determine Winner Proposals. In case only one proposal is available in this stage, this proposal is accepted and wins the auction of this iteration. Otherwise, the intermediary determines the set of winner proposals $\mathcal{P}_W^*$ by solving a multiobjective combinatorial optimization problem. More precisely, the intermediary chooses $\mathcal{P}_W^*$ in such a way that there is no other subset of remaining proposals whose combination, that is, aggregation, yields a greater gain in satisfaction of the total remaining demand. The second and subordinate objective is to achieve the primary goal with a minimal aggregated change in costs (i.e., $\sum_{P_a^* \in \mathcal{P}_W^*} \Delta\kappa(P_a^*)$ should be as low as possible). Please note that these objectives as well as their lexicographical order correspond to the definition of the DRAP in Section 2.

Finally, for all $t \in \mathcal{W}$, the total predicted contribution $P_{\mathcal{A}_\lambda}[t]$ is updated according to the total predicted contributions $\sum_{P_a^* \in \mathcal{P}_W^*} P_a^*[t]$ of all winner proposals $\mathcal{P}_W^*$.

4.3. Identifying, Dealing with, and Leaving Local Optima

As intermediaries have to solve a combinatorial optimization problem (see Section 4.2.3), it is possible that the subsystem is situated in a local optimum in the

sense that no agent can propose to change its scheduled contribution to unilaterally improve the satisfaction of the total remaining demand.

To avoid such a situation, the intermediary would have to know which proposals will actually be made by its subordinate agents *in advance* so that it knew which proposals to accept or reject. Once stuck in a local optimum, it is very unlikely that the affected subsystem can leave it if the agents propose to change their schedule independently from the decisions of others. If we wanted to solve the problem in a decentralized manner, the agents would thus have to coordinate their decisions. On the other hand, it is even difficult for an intermediary to identify if its subsystem is situated in a local optimum since we cannot assume that it has perfect knowledge about the control models of its subordinate agents. The absence of proposals that reduce the remaining demand in a bidding phase indicates that such a situation might be present though.

We circumvent these problems by making use of the approximations of the subordinate agents' control models the intermediary learned in the course of time to be able to create its own proposals (how these models are learned is presented in Section 6).

The intermediary can use these control models to (1) identify if its subsystem is in a local optimum and (2) to propose to its subordinate agents how to change predicted schedules in order to leave it (evidently, such corrections should be as small as possible). Basically, the intermediary creates these proposals by solving the CSOP formalized in Section 2.5. If no proposals can be generated that increase the demand's satisfaction, the subsystem does not seem to be able to satisfy the demand the intermediary scheduled when taking part in the DRAP of its superior. This can be the result of imprecise approximations of the control models. In such a situation, the intermediary communicates the remaining demand to its superior in the form of a correction which, in turn, tries to distribute it to other subordinate agents (see Section 4.1.2). Otherwise, if the intermediary can identify suitable changes that improve the demand's satisfaction, it sends them in the form of a schedule correction C_a to the corresponding subordinate agents. Along with such a correction, the intermediary suggests a reward for adjusting the contribution. To be more specific, it suggests to change the average costs $\overline{\kappa}_a$ of the corresponding agent's most recently accepted proposal P_a in such a way that the total costs of P_a increase by $\sum_{t \in \mathcal{W}} \overline{\kappa}_a^{\text{avg}} \cdot |C_a[t]|$ (again, we assume that $\overline{\kappa}_a^{\text{avg}} > 0$). Since the agents' scheduled contributions have to comply with their control models and we do not assume that intermediaries have perfect knowledge about them, the agents are allowed to adjust suggested corrections. These adjustments have to be communicated to the corresponding intermediary. After the agents sent the adjusted corrections back to the intermediary, it selects and accepts suitable corrections that allow the subsystem to leave the local optimum with a procedure similar to the one described in Section 4.2.3.

5. ENABLING DYNAMIC RESOURCE ALLOCATION UNDER UNCERTAINTY

To be applicable in open systems, TruCAOS$^+$ must be able to deal with various uncertainties (see Section 2 and Section 3) that originate from *information asymmetry* [Ba and Pavlou 2002], among others. First, this means that TruCAOS$^+$ has to identify uncertainties that stem from deviations between predicted and actual contributions or demand. Second, TruCAOS$^+$ has to make informed decisions that allow dealing with these uncertainties at runtime. Third, TruCAOS$^+$ should provide measures that incentivize the agents to behave beneficially, at least if this is in their power.

We meet these challenges by extending TruCAOS$^+$ by a social system based on trust, a concept that is strongly connected to *risk* [Koller 1988]. Basically, trust mirrors expectations about others [Mui et al. 2002]. It is based on experiences [Jonker and Treur 1999] that stem from contracts in which two or more parties stipulate a desired result of an interaction [Ramchurn et al. 2004b]. Since trust represents an empirically

justified expectation, it can reduce uncertainties and risks associated with other agents' behavior [Ramchurn et al. 2004a]. In computing systems, trust is often measured as a numerical value, the so-called *trust value* [Marsh 1994]. In some trust models, additional measures, such as *confidence* [Kiefhaber et al. 2012], *certainty* [He et al. 2005; Wang and Singh 2007], or *reliability* [Huynh et al. 2006], are used to indicate the belief that the trust value accurately describes an agent's actual observable behavior. A current overview of trust management systems can be found in Yu et al. [2013].

Because trust stems from established contracts and their fulfillment, it allows one to quantify uncertainties, identify uncooperative agents, and effectively lower their utility by sanctioning misbehavior (see Section 5.2). Besides assessing the utility or risk to conclude a contract with a specific agent, trust thus serves as an enforcement mechanism that incentivizes rational agents to comply with their contracts [Ba et al. 1999].

5.1. Trust to Identify Uncertainties

As stated in Section 3.2, trust values are traditionally used to choose beneficial interaction partners on the basis of prior experiences [Keung and Griffiths 2008]. We extend this notion to use trust as a measure of uncertainty that originates from the agents' or the environment's behavior: Agents as well as the environment interact and conclude contracts with intermediaries by providing contribution or demand predictions (i.e., desired results) whose accuracy is evaluated in comparison with the actual values. The resulting experiences are used to determine the agents' or the environment's trustworthiness.

Our trust model consists of a trust value and an additional numerical value called *predictability*: The trust value represents an average deviation of the actual behavior from predictions and thus serves as an expectation of the accuracy of an agent's or the environment's predictions. However, when calculating the trust value, positive and negative deviations, that is, experiences, can cancel each other out. In such situations, the predictability captures the volatility of an agent's behavior and thus the risk that it does not make a contribution as stipulated or expected. The predictability is modeled as the standard deviation and thus corresponds to the criteria "variance of experiences" of the confidence metric defined in Kiefhaber et al. [2012].

In Section 5.1.1, we introduce a trust metric that measures uncertainties that stem from the agents' behavior and explain how trust values and the predictability are calculated in detail. Subsequently, we extend this metric to be applicable in the context of demand predictions in Section 5.1.2. Please note that we need to assess the agents' trustworthiness although, as outlined in Section 5.2, the system is designed to incentivize them to behave beneficially. This is mainly because we cannot assume that all agents can eliminate uncertainties in all situations (e.g., because of environmental impacts that are not under an agent's control) or adapt their behavior as a consequence of sanctions (e.g., because an agent simply lacks the ability to learn by experience as might be the case in systems whose development is rather a gradual evolution than a sudden shift [Sage and Cuppan 2001]).

5.1.1. Trust to Assess Uncertainties Originating from the Agents' Behavior. With respect to predicted contributions, TruCAOS$^+$ uses trust values to assess the risk that a contract, that is, a predicted schedule P_a, is violated. In each time step t and for each subordinate agent $a \in \mathcal{A}_\lambda$, an intermediary λ gains an experience $\epsilon_a[t] = (A_a[t], P_a[t], t)$ that captures the deviation $\upsilon(\epsilon_a[t]) = A_a[t] - P_a[t]$ between the actual and the predicted contribution at time step t.[8] An intermediary holds the $n_\mathcal{A}$ newest experiences with a specific agent

[8]We assume that λ is impartial and that it is able to measure the actual contribution as well as the actual demand (see Section 3).

a in a sequence $\mathcal{E}_a = \langle \epsilon_a[t_{\text{now}} - n_{\mathcal{A}} + 1], \ldots, \epsilon_a[t_{\text{now}}] \rangle$. We only regard the $n_{\mathcal{A}}$ newest experiences because an agent's behavior may change over time and we assume that more recent behavior can give a better indication of an agent's future behavior. In the case of former negative experiences, this also allows for *forgiveness* [Vasalou and Pitt 2005]. On the other hand, prior positive experiences lose their relevance over time if the corresponding agent now notoriously makes bad predictions.

From these experiences, an intermediary can derive a trust value $T_a(\mathcal{E}_a^c)$ that assesses its trust that a makes a certain stipulated contribution c. Please note that $T_a(\mathcal{E}_a^c)$ is derived from the set of experiences $\mathcal{E}_a^c \subseteq \mathcal{E}_a$ that contains the $n'_{\mathcal{A}} \leq n_{\mathcal{A}}$ experiences that are—with regard to their predicted contribution $P_a[t]$—most similar to c ($n'_{\mathcal{A}}$ being constant). That way, the intermediary can detect situations in which a's behavior depends on the amount of resources c it pretends to provide in the future. For example, because of technical difficulties, there might be a power plant that makes rather accurate predictions in the case its predicted contribution is rather low, but very inaccurate predictions if its predicted contribution is high. Because of the agents' possibly contribution-specific and dynamic behavior, when calculating the trust value, we further weight each experience $\epsilon_a[t] \in \mathcal{E}_a^c$ according to the dissimilarity $\delta_{\text{dis}}(\epsilon_a[t]) = |c - P_a[t]|$ between c and $\epsilon_a[t]$'s predicted contribution $P_a[t]$ on the one hand, and according to its age $\delta_{\text{age}}(\epsilon_a[t]) = t_{\text{now}} - t$ on the other hand. The less similar $\epsilon_a[t]$'s predicted contribution is to c and the older $\epsilon_a[t]$, the higher the values for $\delta_{\text{age/dis}}$ and, consequently, the lower should be its weight.[9]

The function $w_{\text{age/dis}}(\epsilon_a[t], \mathcal{E}_a^c) \in [0, 1]$ summarizes these properties in a more formal way. It determines the weight of an experience $\epsilon_a[t]$ either with regard to its age or dissimilarity to c ($\delta^{\max}_{\text{age/dis}}(\mathcal{E}_a^c)$ yields the maximum $\delta_{\text{age}}(\epsilon_a[t])$ or $\delta_{\text{dis}}(\epsilon_a[t])$ of all experiences $\epsilon_a[t] \in \mathcal{E}_a^c$; smallest values are provided by $\delta^{\min}_{\text{age/dis}}(\mathcal{E}_a^c)$):

$$w_{\text{age/dis}}\big(\epsilon_a[t], \mathcal{E}_a^c\big) = \begin{cases} \frac{\delta^{\max}_{\text{age/dis}}(\mathcal{E}_a^c) - \delta_{\text{age/dis}}(\epsilon_a[t])}{\delta^{\max}_{\text{age/dis}}(\mathcal{E}_a^c) - \delta^{\min}_{\text{age/dis}}(\mathcal{E}_a^c)} & \text{if } \delta^{\max}_{\text{age/dis}}\big(\mathcal{E}_a^c\big) \neq \delta^{\min}_{\text{age/dis}}\big(\mathcal{E}_a^c\big) \\ 1 & \text{else.} \end{cases}$$

The actual weight $w(\epsilon_a[t], \mathcal{E}_a^c) = \frac{w_{\text{age}}(\epsilon_a[t], \mathcal{E}_a^c) + w_{\text{dis}}(\epsilon_a[t], \mathcal{E}_a^c)}{2}$ of an experience is the mean of the weights according to age and dissimilarity.

Because of an agent's possibly contribution-specific behavior, we use the concept of *impartiality* with regard to its behavior in a certain contribution range Δc: If there is no experience $\epsilon_a[t]$ with $\delta_{\text{dis}}(\epsilon_a[t]) \leq \Delta c$, the intermediary adds an artificial experience $\epsilon_a[t_{\text{now}}] = (0, 0, t_{\text{now}})$ of assumed benevolent behavior to $\mathcal{E}_a^c$. When calculating the trust value, this experience has the greatest influence on the result. This measure of adding artificial experiences is similar to the well-known concept of initial trust [McKnight et al. 1998].

Definition of an Agent's Trust Value. Based on the experiences $\mathcal{E}_a^c$, the trust value $T_a(\mathcal{E}_a^c)$ represents the expectation that agent a makes a contribution c. It is defined as the weighted mean of measured differences between actual and predicted contributions, recorded in experiences $\epsilon_a[t]$:

$$T_a\big(\mathcal{E}_a^c\big) = \begin{cases} \frac{1}{\sum_{\epsilon_a[t] \in \mathcal{E}_a^c} w(\epsilon_a[t])} \cdot \sum_{\epsilon_a[t] \in \mathcal{E}_a^c} \left(w(\epsilon_a[t]) \cdot \frac{v(\epsilon_a[t])}{v^{\max}_{\mathcal{A}_\lambda}} \right) & \text{if } v^{\max}_{\mathcal{A}_\lambda} > 0 \\ 0 & \text{else.} \end{cases}$$

To assess the trust value for a in relation to all subordinate agents $\mathcal{A}_\lambda$ of an intermediary λ, we use the maximum deviation $v^{\max}_{\mathcal{A}_\lambda} = \max\{|v(\epsilon_{a'}[t])| \mid \epsilon_{a'}[t] \in \mathcal{E}_{a'} \wedge a' \in \mathcal{A}_\lambda\}$ recorded in an experience gained with some agent $a' \in \mathcal{A}_\lambda$ as a normalizing factor. Note that $T_a(\mathcal{E}_a^c)$ is from the interval $[-1, 1]$ and represents a's *expected*

[9]Note that we write "age/dis" to either refer to "age" or "dissimilarity."

deviation $\mathbb{E}(v_a^c) = T_a(\mathcal{E}_a^c) \cdot v_{\mathcal{A}_\lambda}^{\max}$ from a stipulated contribution $c = P_a^{(*)}[t]$ ($P_a^{(*)}[t]$ can either stand for a predicted $P_a[t]$ or a proposed contribution $P_a^*[t]$). The greater $|T_a(\mathcal{E}_a^c)|$, the less trustworthy a. If $T_a(\mathcal{E}_a^c) < 0$, λ expects that a's actual contribution $A_a[t]$ will be lower than $P_a^{(*)}[t]$. Otherwise, it is expected that $A_a[t]$ is greater than $P_a^{(*)}[t]$ if $T_a(\mathcal{E}_a^c) > 0$, or that a complies with its stipulated contribution (i.e., $A_a[t] = P_a^{(*)}[t]$) if $T_a(\mathcal{E}_a^c) = 0$.

Definition of an Agent's Predictability. Because the trust value reflects the *mean* deviation from predictions, positive and negative experiences can cancel each other out, that is, $T_a(\mathcal{E}_a^c)$ might be 0 although a always deviates from predictions. So if we want to assess an agent's predictability, that is, the risk that it does not make a contribution as stipulated or expected, we need an additional sound source of information. Since an agent's predictability decreases with the volatility of its behavior, we use the standard deviation as a metric. With respect to the trust value $T_a(\mathcal{E}_a^c)$—the weighted mean of experiences, that is, observed deviations—, we thus define the *predictability* $T_a^\sigma(\mathcal{E}_a^{\Delta c,c}) \in [0, 1]$ of an agent a in the context of a contribution c as the corresponding weighted standard deviation of the experiences $\mathcal{E}_a^{\Delta c,c} \subseteq \mathcal{E}_a^c$. Please note that the set $\mathcal{E}_a^{\Delta c,c}$ only contains those experiences $\epsilon_a[t]$ whose dissimilarity $\delta_{\text{dis}}(\epsilon_a[t])$ is in a certain range $[0, \Delta c]$ (this set is nonempty since it contains at least the artificial experience). That way, we take account of the characteristic that an agent's behavior can depend on its stipulated contribution, which could result in a high standard deviation of experiences in $\mathcal{E}_a^c$ although the agent's behavior is quite predictable in the different contribution ranges. Expected variations are defined as $\mathbb{E}(\sigma_a^{\Delta c,c}) = T_a^\sigma(\mathcal{E}_a^{\Delta c,c}) \cdot v_{\mathcal{A}_\lambda}^{\max}$.

5.1.2. Trust to Quantify Uncertainties Originating from the Environment. The uncertainties associated with demand predictions are identified in a similar way as described in Section 5.1.1. Please note that only the top-level intermediary Λ gains experiences with the demand since it is the only intermediary that requests demand predictions from the environment. Since we assume that predictions become more accurate as the point in time they are made for approaches, the top-level intermediary quantifies the accuracy of demand predictions with respect to the temporal distance to a future point in time. Consequently, the intermediary gains multiple experiences per time step. This is in contrast to the trust metric presented in Section 5.1.1 where intermediaries only evaluate the quality of predictions made for the next time step and thus only gain one experience per agent and time step.

More precisely, since the demand is predicted for the next H time steps in every time step, there are exactly $H = |\mathcal{W}|$ demand predictions for every time step t_{for}. These were obtained in the H previous time steps $t_{\text{at}} \in \{t_{\text{for}} - H, \ldots, t_{\text{for}} - 1\}$. As a result, in each time step t_{for}, Λ acquires H experiences. Each experience $\epsilon_{env}[t_{\text{for}}, t_{\text{at}}] = (A_{env}[t_{\text{for}}], P_{env}^{t_{\text{at}}}[t_{\text{for}}], t_{\text{for}}, t_{\text{at}})$ reflects the difference $v(\epsilon_{env}[t_{\text{for}}, t_{\text{at}}]) = A_{env}[t_{\text{for}}] - P_{env}^{t_{\text{at}}}[t_{\text{for}}]$ between the actual demand $A_{env}[t_{\text{for}}]$ observed in t_{for} and the predicted demand for t_{for} made at time step t_{at}, denoted by $P_{env}^{t_{\text{at}}}[t_{\text{for}}]$.

Since predictions are assumed to become more accurate as the point in time they are made for approaches, the uncertainty of a predicted demand $P_{env}^{t_{\text{at}}}[t_{\text{for}}]$ depends on the distance $\Delta t \in \{1, \ldots, H\}$ between the time step $t_{\text{at}} \in \{t_{\text{for}} - H, \ldots, t_{\text{for}} - 1\}$ and t_{for}. Figure 4 illustrates the connection between predictions $P_{env}^{t_{\text{at}}}[t_{\text{for}}]$ and the distance Δt (in the following, we deliberately omit the exponent t_{at}). To capture this behavior with regard to the context Δt, Λ manages H sequences of experiences $\mathcal{E}_{env}^{\Delta t} = \langle \epsilon_{env}[t_{\text{now}} - n_{env} + 1, t_{\text{now}} - n_{env} + 1 - \Delta t], \ldots, \epsilon_{env}[t_{\text{now}}, t_{\text{now}} - \Delta t] \rangle$, one for each distance $\Delta t \in \{1, \ldots, H\}$. Each $\mathcal{E}_{env}^{\Delta t}$ contains the n_{env} newest experiences $\epsilon_{env}[t_{\text{for}}, t_{\text{at}}]$, where $\Delta t = t_{\text{for}} - t_{\text{at}}$. Again, the intermediary assesses the expected accuracy of a demand prediction $d = P_{env}[t]$ on the basis of the set $\mathcal{E}_{env}^{\Delta t,d} \subseteq \mathcal{E}_{env}^{\Delta t}$ that represents the $n'_{env} \leq n_{env}$ experiences that are—in terms of the predicted demand—most similar to d (n'_{env} being constant).

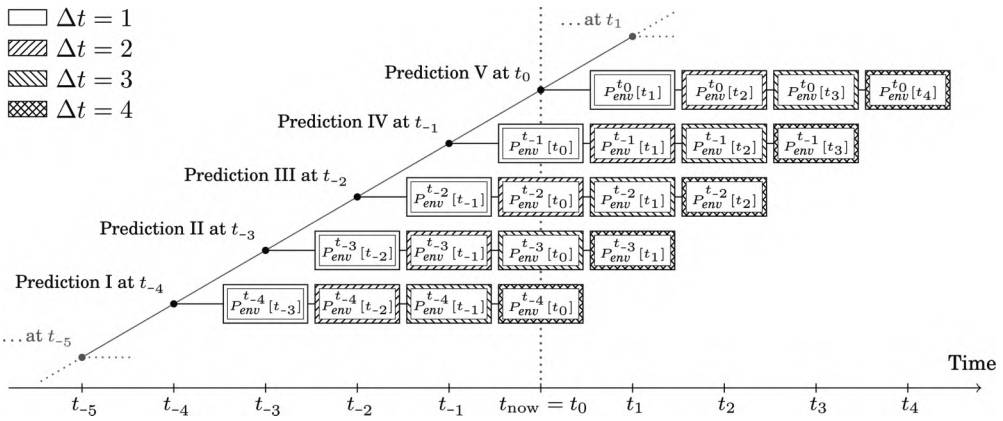

Fig. 4. Five demand predictions I, II, ..., V made for $H = 4$ consecutive time steps at different points in time t_{at}, illustrating the relation between a demand prediction $P_{env}^{t_{\text{at}}}[t_{\text{for}}]$ for a specific time step t_{for} and the distance $\Delta t = t_{\text{for}} - t_{\text{at}}$ (in this example, $\Delta t \in \{1, \ldots, 4\}$): As indicated in t_{now}, the top-level intermediary gains H experiences in each time step. In this figure, $t_i = t_{\text{now}} + i$ (with $i \in \{-5, -4, \ldots, 4\}$).

The corresponding trust value $T_{env}(\mathcal{E}_{env}^{\Delta t,d})$ in the context of a distance $\Delta t = t - t_{\text{now}}$ (with $t \in \mathcal{W}$) and a demand prediction $d = P_{env}[t]$ is based on a metric that is analogously defined to $T_a(\mathcal{E}_a^c)$. Instead of $v_{\mathcal{A}_\lambda}^{\max}$, it uses $v_{env}^{\max}$, the maximum deviation from a demand prediction, to normalize the experiences. $T_{env}(\mathcal{E}_{env}^{\Delta t,d}) \in [-1, 1]$ thus represents an expected deviation $\mathbb{E}(v_{env}^{\Delta t,d}) = T_{env}(\mathcal{E}_{env}^{\Delta t,d}) \cdot v_{env}^{\max}$ from a predicted demand $P_{env}[t]$.

5.2. Dealing with Uncertainties

Intermediaries use trust values in various situations to identify and deal with uncertainties, for example, by incentivizing beneficial behavior through indirect, trust-based sanctions that lower the utility of uncooperative agents. Under the assumption that an intermediary's superior acts as described in the following, it is therefore in its best interest to maintain a good trust value itself. Since an intermediary's contribution is made by its subordinate agents, it should only accept predicted contributions the agents will make in the future. We assume that the agents know how misbehavior is sanctioned and the opportunity cost they incur imply that it is in the agents' best interest to behave benevolently—that is, we assume them to be *rational*.

The following enumeration summarizes the adjustments we make to TruCAOS$^+$:

(1) *Rewarding Agents (see Section 5.2.1):* To incentivize agents to comply with their predicted contributions, we adjust the reward function presented in Section 4.1.3. Deviations from predicted contributions are punished by lowering the corresponding agent's reward.
(2) *Expected Demand and Contributions (see Section 5.2.2):* Intermediaries use trust values to form expectations of the remaining demand and scheduled contributions. As our evaluation shows, scheduling on the basis of expectations reduces the probability of deviations between actual demand and contributions because our trust model allows intermediaries to lower epistemic uncertainties (see Section 7.2).
(3) *Redefining Valid Proposals (see Section 5.2.3):* A proposal is now regarded as valid if the *expected* gain in satisfaction is improved. This expectation is also derived from the information provided by trust values.
(4) *Expected Quality and Price-Performance Ratio (see Section 5.2.4):* A proposal's quality is now assessed by using an *expectation* of the change in costs on the basis of the idea of the revised reward function. Furthermore, we make use of the

concept of predictability. The predictability constitutes a risk that the goal to balance contributions and demand is not achieved and thus quantifies aleatoric uncertainties. Because of an agent's possibly contribution-specific behavior, its predictability might change as a result of the acceptance of its proposal. We therefore map the change in predictability to costs that are incorporated into a proposal's quality assessment. This measure allows us to efficiently put the agents into states with less aleatoric uncertainties (see Section 7.2). As before, only those proposals can be accepted that feature a sufficient quality.

These enhancements are discussed in more detail in the following subsections.

5.2.1. Rewarding Agents. First, as a's actual contribution $A_a[t]$ in a time step t might differ from the predicted contribution $P_a[t]$, we redefine the reward r_a agent a receives:

$$r_a = \kappa(P_a, t_{\text{now}} - 1) - \left(\max\{|\overline{\kappa}_a|, |\overline{\kappa}^{\text{avg}}_{\mathcal{A}_\lambda}|\} \cdot |P_a[t_{\text{now}} - 1] - A_a[t_{\text{now}} - 1]|\right).$$

With regard to the definition in Section 4.1.3, we now charge the agent for the deviation $|P_a[t_{\text{now}} - 1] - A_a[t_{\text{now}} - 1]|$ from its predicted contribution. The punishment is based on the maximum of the absolute value of the stipulated average costs $\overline{\kappa}_a$ in the corresponding schedule P_a and the absolute value of the subsystem's average costs of accepted proposals $\overline{\kappa}^{\text{avg}}_{\mathcal{A}_\lambda}$ that is calculated over a fixed number of the last time steps. Using absolute values ensures that the agent is punished for its behavior, regardless of whether its actual contribution is greater or lower than its predicted contribution.

Please note that schedule corrections that originate from situations in which an intermediary cannot distribute its fraction of the overall demand to subordinate agents also lower the reward the intermediary receives (see Section 4.1.2). However, because its superior still has the opportunity to compensate for this schedule correction *before* the contribution actually has to be made, the reduction of the reward caused by the previously mentioned form of schedule corrections is not greater but usually lower than the punishment for a noncommunicated deviation from a predicted contribution.

5.2.2. Expected Demand and Contributions. The top-level intermediary Λ makes use of the trust values for the environment to determine expectations of the future demand (see Section 5.1.2). The expected demand $E_{env}[t] = P_{env}[t] + \mathbb{E}(v^{\Delta t,d}_{env})$ (with $\Delta t = t - t_{\text{now}}$) in a time step $t \in \mathcal{W}$ is defined on the basis of a demand prediction $d = P_{env}[t]$ and the expected deviation $\mathbb{E}(v^{\Delta t,d}_{env})$ that results from the corresponding trust value $T_{env}(\mathcal{E}^{\Delta t,d}_{env})$.

Analogously, every intermediary λ forms expectations of future contributions. The expected contribution $E^{(*)}_a[t] = P^{(*)}_a[t] + \mathbb{E}(v^c_a)$ of an agent a is defined on the basis of a stipulated contribution $c = P^{(*)}_a[t]$ and the expected deviation $\mathbb{E}(v^c_a)$ that originates from the corresponding trust value $T_a(\mathcal{E}^c_a)$ (see Section 5.1.1). Similarly to P_a and P^*_a, we define the ***expected*** *schedule* $E_a = (\langle E_a[t_{\text{next}}], \ldots, E_a[t_{\text{now}} + H]\rangle, \overline{\kappa}_a)$ as well as the expected schedule E^*_a that would result *in case a proposal* P^*_a *was accepted*. Note that the average costs $\overline{\kappa}_a$ are the same for $E^{(*)}_a$ and $P^{(*)}_a$. Based on these expectations, the call for proposals now contains the *expected* remaining demand $R_\lambda[t] = E_{env}[t] - E_{\mathcal{A}_\lambda}[t]$.

5.2.3. Redefining Valid Proposals. The remaining demand $R^*_\lambda(P^*_a, t) = R_\lambda[t] + E_a[t] - E^*_a[t]$ for time step t that would result if P^*_a was accepted (see Section 4.2.1) is now based on the expected change in contribution, which results from the trust value $T_a(\mathcal{E}^c_a)$. The gain in satisfaction $\mathcal{G}(P^*_a)$ is thus an expectation. Because of contribution-specific uncertainties associated with an agent a, $E^*_a[t]$ might be smaller than $E_a[t]$, although $P^*_a[t]$ is greater than $P_a[t]$. For the same reason, regarding two proposals $P^*_{a_1}$ and $P^*_{a_2}$ of two agents a_1 and a_2, the expected gain in satisfaction $\mathcal{G}(P^*_{a_1})$ might be higher than $\mathcal{G}(P^*_{a_2})$, although the overall contribution predicted in $P^*_{a_1}$ is smaller than in $P^*_{a_2}$.

5.2.4. Expected Quality and Price-Performance Ratio. As defined in Section 4.2.2, a proposal's quality $\mathcal{Q}(P_a^*)$ and price-performance ratio $\Theta(P_a^*)$ are based on the change in costs $\Delta\kappa(P_a^*)$. Here, we redefine the change in costs to use expectations. More precisely, we seize the idea of the updated reward function introduced in Section 5.2.1 to determine the expected change in costs $\Delta\kappa(\boldsymbol{E}_a^*)$. Instead of actual contributions and actual deviations from predicted contributions (both values are obviously not known in advance), the intermediary uses expected contributions $\boldsymbol{E}_a^{(*)}[t]$ as well as expected deviations $\mathbb{E}(v_a^c)$ to evaluate $\Delta\kappa(\boldsymbol{E}_a^*)$:

$$\Delta\kappa(\boldsymbol{E}_a^*) = \sum_{t\in\mathcal{W}} \mathbb{E}(\overline{\kappa}(\boldsymbol{E}_a^*, t)) - \mathbb{E}(\overline{\kappa}(\boldsymbol{E}_a, t)),$$
$$\text{with } \mathbb{E}\big(\overline{\kappa}\big(\boldsymbol{E}_a^{(*)}, t\big)\big) = \kappa\big(\boldsymbol{E}_a^{(*)}, t\big) - \left(\max\left\{|\overline{\kappa}_a|, \left|\overline{\kappa}_{\mathcal{A}_\lambda}^{\text{avg}}\right|\right\} \cdot \left|\mathbb{E}\big(v_a^c\big)\right|\right), \text{ and } c = \boldsymbol{E}_a^{(*)}[t].$$

Apart from the change in costs, a proposal's quality $\mathcal{Q}(P_a^*)$ and price-performance ratio $\Theta(P_a^*)$ also rely on the corresponding agent's trustworthiness as the gain in satisfaction $\mathcal{G}(P_a^*)$ is based on the trust value $T_a(\mathcal{E}_a^c)$. However, as discussed in Section 5.1, positive and negative deviations can cancel each other out, which might result in a good trust value and quality although the agent's compliance with proposed contributions is hard to predict. To master such situations, we incorporate the uncertainty of a contribution—by means of the predictability $T_a^\sigma(\mathcal{E}_a^{\Delta c,c})$ of an agent's behavior—when assessing the quality of its proposals. Since we want to effectively reduce the presence of aleatoric uncertainties (see Section 1), intermediaries evaluate the *change in predictability* $\Delta_\sigma(\boldsymbol{E}_a^*)$ when deciding whether a proposal P_a^* should be accepted and replace P_a:

$$\Delta_\sigma(\boldsymbol{E}_a^*) = \sum_{t\in\mathcal{W}} \mathbb{E}\big(\sigma_a^{\Delta c,c_1}\big) - \mathbb{E}\big(\sigma_a^{\Delta c,c_2}\big), \text{ with } c_1 = \boldsymbol{E}_a^*[t] \text{ and } c_2 = \boldsymbol{E}_a[t].$$

If $\Delta_\sigma(\boldsymbol{E}_a^*) < 0$, the intermediary assumes that the predictability increases and the risk of imbalances between demand and contributions decreases. By multiplying the change in predictability $\Delta_\sigma(\boldsymbol{E}_a^*)$ with a factor κ_σ, we map the value to costs (e.g., costs that would occur to compensate for a deviation from an expected contribution at runtime). This mapping allows us to incorporate the change in predictability into the intermediary's decisions by adding the corresponding costs to the change in costs $\Delta\kappa(\boldsymbol{E}_a^*)$. The result is the so-called *risk-based change in costs* $\Delta\kappa_{\text{risk}}(\boldsymbol{E}_a^*)$ that would result in case P_a^* was accepted and replaced P_a:

$$\Delta\kappa_{\text{risk}}(\boldsymbol{E}_a^*) = \Delta\kappa(\boldsymbol{E}_a^*) + \Delta_\sigma(\boldsymbol{E}_a^*) \cdot \kappa_\sigma.$$

The greater the factor κ_σ, the more risk-averse an intermediary's decisions. Certainly, intermediaries have to come to a compromise between costs and risk. When evaluating the quality $\mathcal{Q}(P_a^*)$ of a proposal P_a^* or its price-performance ratio $\Theta(P_a^*)$, an intermediary now uses $\Delta\kappa_{\text{risk}}(\boldsymbol{E}_a^*)$ instead of $\Delta\kappa(\boldsymbol{E}_a^*)$.

This trust-based influence on a proposal's quality and price-performance ratio results in price premiums and price discounts [Ba and Pavlou 2002] because trustworthy agents can charge higher prices for a contribution than untrustworthy agents.

Summarizing, all these measures are aimed at reducing epistemic as well as the presence of aleatoric uncertainties. The latter is achieved on the basis of the concept of predictability, which allows intermediaries to anticipate aleatoric uncertainties with regard to the agents' states. In turn, this information is used to actively put the agents into states with less aleatoric uncertainties. In future work, we want the agents to create schedules that improve the system's ability to deal with aleatoric uncertainties that emerge in the form of deviations between demand and contributions at runtime

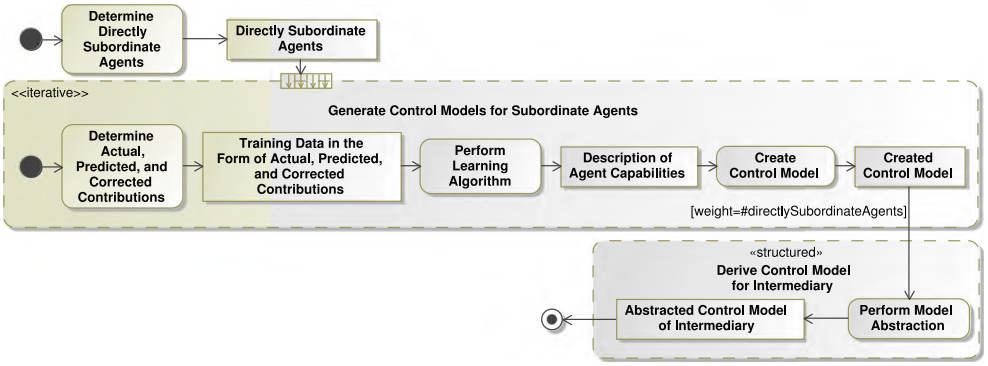

Fig. 5. Generation of an intermediary's control model based on learned control models of subordinate agents.

(this corresponds to the second characteristic of robust solutions we mentioned in Section 1). For this purpose, we will need a revised estimation of the environment's future behavior, for example, by predicting multiple possible future developments of the demand by means of trust-based scenarios [Anders et al. 2014c]. If the system creates schedules for multiple scenarios, it self-improves its ability to cope with uncertainties since it increases its flexibility by scheduling reserves. As has been shown for a regio-central scheduling approach in Anders et al. [2014b], the resulting schedules can guide a reactive mechanism (see, e.g., Anders et al. [2012]) in order to effectively compensate for deviations.

6. CREATING CONTROL MODELS THROUGH OBSERVATION

As mentioned in Section 2.4, intermediaries cannot rely on an explicit control model of their subordinate agents. On the one hand, privacy concerns might be raised given that, for example, many energy producers correspond to small cooperatives, private households, or farmers that do not want to disclose information that can be used to derive their strategic options. In particular, a power plant that is capable of regulating down to 50% of its maximum output may not want to have this information public in order to avoid having to run at this low level. On the other hand, such explicit models of the underlying technical system (including predictive aspects and dynamics) might simply not be available due to a low level of maturity [Derksen et al. 2013]. Certainly, some agents might not be reticent to offer basic information such as the minimum and maximum contribution and *can* offer a control model to their superordinate. In general, we refrain from imposing this requirement on all agents.

However, an intermediary has to know about its possible contribution space in order to make feasible proposals it can send to its own superior. In absence of existing control models, we thus resort to learning appropriate models from data, more precisely from the proposals and actual values that were observed. Certainly, according to the *no free lunch theorem* [Wolpert 1996], some assumptions have to be made in order to obtain useful control models which, in our case, are encoded in the constraints introduced in Section 2.3: a set of possible production intervals and limited controllability due to inertia. We are interested in a description of all feasible contributions of subordinate agents such that an intermediary can combine these descriptions to find out its own possible contributions.

Figure 5 shows the basic procedure of how an intermediary can create its own control model on the basis of observed contributions of its subordinate agents:

(1) *Generate Control Models for Subordinate Agents (see Section 6.1):* For each of its directly subordinate agents, the intermediary stores actual, predicted, and corrected contributions. These will serve as *training data* for a learning algorithm that obtains a description of a single agent's capabilities (see Section 6.1.1). In particular, we use support vector data description as a one-class classification model that is well suited for the data expected from agents since it only requires positive examples and is able to capture discontinuous intervals and min./max. successor functions as basic features of the considered agent's behavior. Subsequently, the intermediary extracts a control model from the learned description (see Section 6.1.2).
(2) *Derive Control Model for Intermediary (see Section 6.2):* Based on these individual control models, the intermediary's capabilities are determined. In principle, for each proposal generation, an intermediary could solve a combinatorial problem taking the control models of all subordinates into account. To scale better with the number of subordinate agents, an intermediary calculates and uses an *abstracted* control model for proposal generation.

Summarizing, an intermediary records the contributions of its subordinate agents to approximate their control models. This information, along with the current states, is used to gain an abstracted control model that represents the feasible operation range of the intermediary itself. The intermediary uses the abstracted model to generate its own proposals.

Before we turn our attention to the incorporated algorithms, we need to address one issue faced in any endeavor to construct such control models from data: During the initial phase of the system's runtime, intermediaries need to make proposals but have no data about their subordinate agents' possible or actual contributions. Instead of creating control models for a top-down assignment of the demand, all intermediaries could propagate the information about the overall demand $A_{env}[t]$ through the hierarchy and collect proposals in a bottom-up way. However, as described in Section 5.2, an intermediary would have to accept suitable predicted contributions before being able to decide about its own proposal. If $A_{env}[t] \gg \sum_{a \in \mathcal{A}_\lambda} A_a^{\max}$, an intermediary λ could sum up all received proposals $\mathcal{P}^*$ to calculate an aggregated proposal to hand to its superior—otherwise it would have to preselect which proposals to include. Hence, every intermediary would have to explore the power set of $\mathcal{P}^*$—effectively solving a DRAP in every iteration of the algorithm. By contrast, the top-down approach gets along with much less communication and computation and thus allows intermediaries to efficiently steer and intervene in the behavior of their subsystem. Nevertheless, the described bottom-up approach could be used in the initial phase of the system to harness a sufficiently large training set of feasible schedules but gets replaced by the top-down approach at runtime.

The following sections treat the previously described steps for learning control models in more detail. We rely on techniques from machine learning and constraint-based modeling.

6.1. Generating Control Models for Subordinate Agents

Many well-studied machine learning algorithms can be categorized into classification or regression problems—neither is particularly suited for our problem of "describing" the feasible contribution space of an agent, given a set of exemplary contributions. Instead, we employ a third kind of learning algorithm—the so-called "data description" or "one-class classification," which approximates the space the presented training data (coming only from one "class") reside in.

In particular, we generally expect agents to submit only feasible schedules and we thus observe more positive than negative examples. A particular instance of one-class classification is Support Vector Data Description (SVDD) [Tax and Duin 1999], which is inspired by support vector machines [Vapnik 1995] and has been used successfully to encode search spaces of distributed energy resources by Bremer et al. [2011] and Hinrichs et al. [2013a]. Intuitively, SVDD creates an abstract description of the positive training examples by fitting a tight hypersphere that encompasses most training examples. As Figure 6 motivates, all considered information about agents (feasible intervals as well as min./max. successor functions) can be extracted from an SVDD trained on observed actual and predicted contributions. We thus specialize the step *Generate Control Models for Subordinate Agents* in Figure 5 using SVDD as a learning algorithm.

6.1.1. Learning Descriptions from Data using SVDD. We revisit the basic properties of SVDD and refer to Tax and Duin [2004] for a detailed introduction. The intuition is to find the smallest hypersphere (characterized by center $\mathbf{a}$ and radius r) that contains most training examples $\{\mathbf{x}_i \in \mathbb{R}^n \mid i \in 1, \ldots, m\}$:

$$\underset{r,\mathbf{a},\xi}{\text{minimize}} \quad r^2 + C \sum_i^m \xi_i \quad \text{subject to} \quad \|\mathbf{x}_i - \mathbf{a}\|^2 \leq r^2 + \xi_i, \xi_i \geq 0, \forall i \in 1, \ldots, m,$$

where ξ is a vector of slack variables, one for each training example $\mathbf{x}_i$, to allow for some inputs to lie outside the hypersphere in order to avoid having a too high volume (and thus high false-positive errors) due to outliers. C is a constant that regularizes the trade-off between the volume of the hypersphere and allowed errors (the greater C, the stronger outliers are punished, leading to a necessary increase in r and therefore higher resulting volumes). The basic formulation relying on inner products in $\mathbb{R}^n$ would select an optimal circle. Data representing the contributions of agents might, however, be nonlinear and show discontinuous feasible intervals (see Section 6.2.1), which calls for more flexible data descriptions. We can achieve this flexibility by replacing the standard inner products by applications of kernel functions (implicitly mapping training data to a higher dimensional space—the well-known "kernel trick" involving Mercer kernels) that measure the degree of similarity of two points. This can be done as it turns out that the optimization problem to be solved only relies on occurrences of any $\mathbf{x}_i$ in inner products with another $\mathbf{x}_j$. A popular instance is the Gaussian kernel, which we used for our implementation:

$$k(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma \|\mathbf{x}_i - \mathbf{x_j}\|^2},$$

where $\gamma = \frac{1}{2\sigma^2}$ is a hyperparameter regulating how the Gaussian kernel evaluates objects as similar. A small value for γ indicates a large value for the variance σ^2 of the Gaussian function and thus "tolerantly" evaluates the similarity of points lying farther apart in the input space, resulting in hyperspheres of larger volumes.We shall see that C and (mostly) γ are the parameters that influence the shape and accuracy of the learned SVDD classifiers (see Figures 6, 7, and 8). Finding suitable values for γ corresponds to the well-known bias-variance trade-off found in statistical learning [James et al. 2014, Chapter 2]. A high bias limits the flexibility of the classifier leading to possible underfitting, whereas a high variance yields flexible classifiers prone to overfitting. Please note the terminological subtlety that a high variance problem is faced when γ is high and consequently σ^2 (the *variance* of the Gaussian kernel) *low*.

In our setup, an intermediary collects sequences of actual and predicted contributions for each subordinate agent a and splits them into $(A_a[t], A_a[t+1])$ or $(P_a^{(*)}[t], P_a^{(*)}[t+1])$ pairs to serve as the training examples for SVDD_a (for ease of presentation, we will

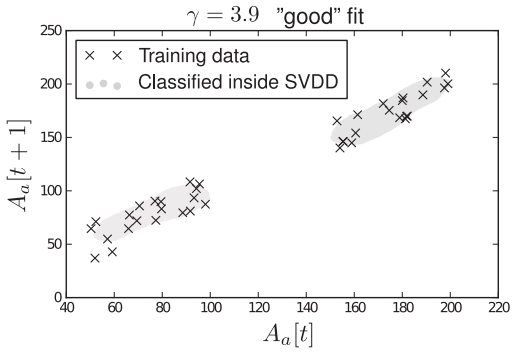

Fig. 6. A well fit model that describes the feasible ranges and inertia of an observed plant.

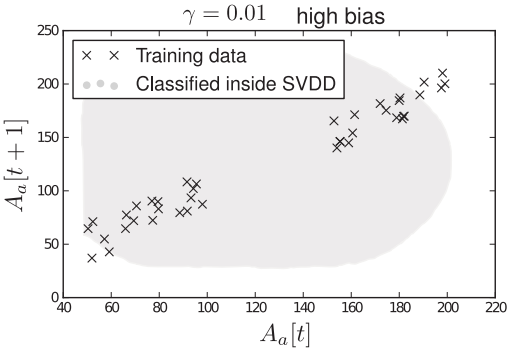

Fig. 7. A too small γ results in underfitting the data due to the large σ^2 in the Gaussian kernel considering almost all points equal and a merely spherical structure.

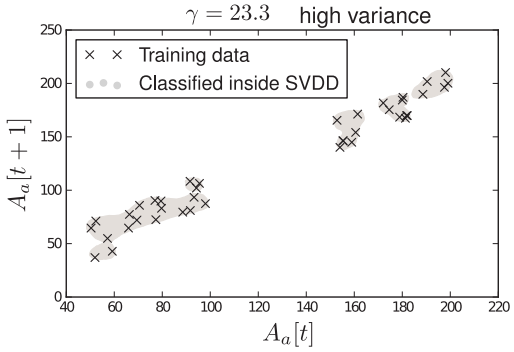

Fig. 8. A too large γ results in overfitting the data and "cluttered" structures since only very close training points are considered by the Gaussian kernel when using a very small σ^2.

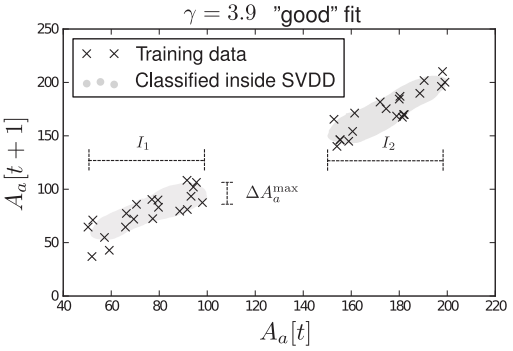

Fig. 9. Control models are derived by extracting the relevant information from an SVDD model.

stick to actual contributions in the following). Note that observed and proposed behavior leads to positive examples, whereas corrected behavior leads to negative examples. We applied feature scaling to the data, thus having each dimension reside in $[0, 1]$. This numerically stabilizes and improves the performance of SVDDs significantly [Juszczak et al. 2002].

An interesting problem in the context of one-class classification is that of *model selection* as, by definition, only representatives of one class need to exist in the training set (although negative examples can also be used during training to improve the accuracy of the obtained bound [Tax and Duin 2004]). In the case of an SVDD with the Gaussian kernel, the model selection problem boils down to the choice of the kernel hyperparameter γ and C to obtain good generalization qualities. Typically, this problem is addressed using k-fold cross-validation, where the training set is separated into k "folds" and each one serves as the test set to evaluate some error measure (e.g., misclassification rate) exactly once when the classifier is trained on the $k - 1$ other folds. Different combinations for (C, γ) are evaluated with respect to the misclassification rate over all folds and the best one is picked. Figures 6, 7, and 8 show the effects of increasing values of γ on the decision boundaries that become more flexible with higher values of γ.

Negative examples (outliers) need to be at hand in order to avoid high bias in the resulting models as shown in Figure 7 where a static "description" always returning 1 (for being inside the SVDD) would perform optimally on a test set containing only positive examples. Therefore, standard k-fold cross-validation for SVDDs (as suggested by Pan et al. [2005]) cannot directly be applied since they favor highly biased models, which would lead to high inaccuracies during the extraction of control models. For instance, when using the model generated in Figure 7, a contribution between 100 and 150 would be accepted by the SVDD, whereas by projecting all training points onto the x axis we would assume that this agent is not able to provide a value in that interval and thus prefer a more fine-grained model as the one depicted in Figure 6. When having negative data at hand, the problem addressed with SVDD is still different from regular classification in that data description always finds a bounded region (or multiple regions), whereas a classifier could also just draw a boundary between two classes instead of detecting outliers that do not belong to either class [Tax and Duin 2004]. In particular, to achieve the same boundedness with a conventional classifier, negative examples surrounding all positive examples need to be at hand.

This issue has been identified by Bremer et al. [2011] for virtual power plants in particular, where the choice of γ and C remained open problems. Different approaches try to automatically generate outliers to improve the accuracy of an SVDD classifier (e.g., Wang et al. [2009] add artificial boundaries using the maxima of the training sets; Mack and Waske [2011] identify potential outliers by taking positively classified examples that have a high mean distance to other positively classified examples) or propose other methods to directly optimize the complexity (the flexibility to adapt to data) of a one-class classifier [Tax and Müller 2004]. Our approach allows for a domain-specific source of outliers. As discussed in Sections 4.1.2 and 4.1.4, agents are incentivized to avoid corrections since they negatively affect the reward they receive. Given their assumed rationality, we may reason that a correction is only issued in the case a predicted contribution is not feasible and represents a restriction that should be considered by a learned control model. For instance, a correction of $\langle 500, 100\rangle$ to $\langle 500, 150\rangle$ indicates that changing the contribution by -400 from 500 to 100 is not possible. Apparently, only a negative change of 350 is possible if the current contribution is 500. Therefore, we can assume that corrections submitted by agents correspond to infeasible schedules that we can feed into the training and into the cross-validation of SVDD models as *negative examples* in order to achieve a higher accuracy.

6.1.2. Extracting Control Models of Subordinate Agents. Once the SVDD classifiers have been trained on observed schedules, we can use them to extract control models in terms of constraints, as Figure 9 illustrates. We use a grid of test points spanning the observed range of contributions with a desired resolution and obtain a predicted class (inside the described data or not) for each point as a Boolean matrix. The set SVDD_a denotes all grid points that were accepted by the SVDD based on data recorded from a. We may use knowledge about the agents' basic behavior to normalize this matrix, for example, if we see a point $(A_a[t], A_a[t+1]) \in \text{SVDD}_a$, we may assume that $(A_a[t], A_a[t])$ and $(A_a[t+1], A_a[t+1])$ are in SVDD_a as well since contributions do not have to change from one time step to another. That way, we can project the positively classified test points onto one axis only to find feasible intervals by scanning from left to right. Figure 9 shows an example that could have resulted from an AVPP consisting of two plants operating in different ranges where only one can be switched on at a time and therefore presents two disjoint feasible intervals.

The inertia functions can be read from an SVDD model by finding the maximum or minimum value for $A_a[t+1]$ given a value for $A_a[t]$. If we opt for a constant maximal rate of change $\Delta A_a^{\max}$ that represents $\overrightarrow{A}_a^{\max}$, we could use several ways to extract

this value from the SVDD, including the maximum, mean, or median change we face when scanning the SVDD. However, the min./max. successor function could be more complicated if we consider an AVPP that can disconnect some of its power plants. Then, a more precise approximation of the inertia-based limited change would be to collect a set of samples $(A_a[t], \widehat{A}_a[t+1])$ where $\widehat{A}_a[t+1]$ is the maximum or minimum contribution (predicted by SVDD) if the current contribution is $A_a[t]$. These input-output pairs can then be used to form a piecewise linear function representing $\overrightarrow{A}_a^{\min}$ and $\overrightarrow{A}_a^{\max}$ in the control model.

6.2. Deriving the Intermediary's Control Model by Abstraction

Once we have collected the data, trained an SVDD classifier, and extracted the control models for all subordinate agents, we could, in principle, define the CSOP to generate proposals with $|\mathcal{A}_\lambda|$ agents instead of one agent. However, this problem becomes more difficult to solve as the number of subordinate agents is greater than one. We therefore use the techniques described in Schiendorfer et al. [2014] to *abstract* the set of subordinate models and form one control model for the intermediary. Model abstraction consists of three consecutive steps:

(1) *General Abstraction (see Section 6.2.1):* First, we determine the intermediary's *generally* feasible contribution ranges.
(2) *Temporal Abstraction (see Section 6.2.2):* Afterward, the so-called *temporal abstraction* restricts feasible contributions on the basis of the current *state* and constraints on the possible behavior of the subordinates up to a specified point in time.
(3) *Sampling Abstraction (see Section 6.2.3):* Finally, we derive the intermediary's inertia as an approximation using *piecewise linear functions*.

In the following, we explain these three steps in more detail.

6.2.1. General Abstraction. We need to determine the possible contributions an intermediary can make. If we allow that an intermediary excludes some agents from its plans, some agent a could run at 0 or contribute a value $x \geq y > 0$ (hence $L_a = \langle [0,0], [y, A_a^{\max}] \rangle$). Then the range the intermediary itself can contribute in becomes discontinuous in general. To see that, consider an intermediary λ with two agents, a_1 having boundaries of $[1,4]$ and a_2 of $[7,10]$. Ranges $[1,4]$ and $[7,10]$ are obviously feasible if exactly one agent contributes, and some value between $[8,14]$ if both and $[0,0]$ if no agent contributes. This leads to $A_\lambda^{\min} = 0$ and $A_\lambda^{\max} = 14$ where no contribution in $(0,1)$ or $(4,7)$ is possible, thus $L_\lambda = \langle [0,0], [1,4], [7,14] \rangle$.

However, as the agents are limited by inertia, not all contribution intervals are available at all times, given a current state. Consider, for instance, an intermediary λ that is responsible for two agents we assume to have a constant maximal rate of change (i.e., $\overrightarrow{A}_a^{\min}(x) = x - \Delta A_a^{\max}$, $\overrightarrow{A}_a^{\max}(x) = x + \Delta A_a^{\max}$): The first one, a_1, is able to contribute in $[50, 100]$ with $\Delta A_{a_1}^{\max} = 25$, and the second one in $[40, 100]$ with $\Delta A_{a_2}^{\max} = 10$. Now assume that $A_{a_1}[t_{\text{now}}] = 95$ and $A_{a_2}[t_{\text{now}}] = 45$; hence, a_1 is almost fully used to capacity and a_2 runs close to minimum. But in the abstract view, that means that $A_\lambda[t_{\text{now}}] = 140$ and λ can contribute within $[90, 200]$. If we summed up the rates of change (assuming every agent can fully change as would be true and could have been previously observed for, for example, $A_{a_1}[t] = 70 \rightarrow A_{a_1}[t+1] = 95$ and $A_{a_2}[t] = 70 \rightarrow A_{a_2}[t+1] = 80$, implying $A_\lambda[t] = 140$, $A_\lambda[t+1] = 175$) , we would determine $\Delta A_\lambda^{\max} = 35$. We would then deduce that $A_\lambda[t_{\text{next}}] = 175$ were valid and could be considered in a proposal, whereas we can in fact only leverage five of a_1's change capacity and 10 from a_2, arriving at an actual maximum for $A_\lambda[t_{\text{next}}]$ at 155. Using these tighter boundaries,

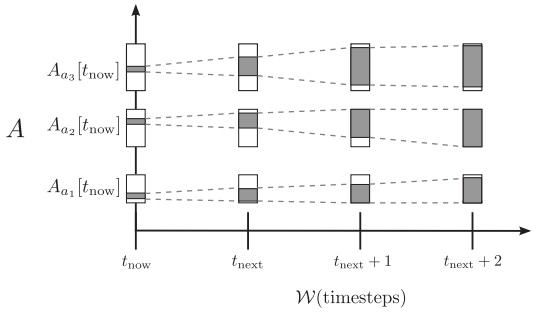

Fig. 10. Temporal abstraction finds boundaries for the contributions of λ for time steps $t \in \mathcal{W}$ by minimizing and maximizing contributions based on the current state of its subordinate agents.

we can avoid having many corrections due to imperfections in the control model the intermediary uses.

6.2.2. Temporal Abstraction. Figure 10 outlines how we use *temporal abstraction* to calculate these boundaries systematically by taking the states of subordinate agents into account: We first derive the boundaries of each individual agent for the considered time steps by maximizing and minimizing. We then combine them for every future time step and conclude that values not in the resulting intervals cannot be reached as we simulated the minimal and maximal trajectory for every subordinate agent. Note that for clarity, we assume agents to be limited by a single, continuous feasible interval $L_a = \langle [A_a^{\min}, A_a^{\max}] \rangle$ as shown by the white rectangles in Figure 10. It is straightforward to weaken this restriction and have the algorithm work with sets of intervals for each agent and time step.

More formally, for every subordinate agent a, we determine $A_a^{\min}[t]$ and $A_a^{\max}[t]$, given $A_a[t_{\text{now}}]$, for $t \in \mathcal{W} \cup \{t_{\text{now}}\}$ as follows:

$$\begin{aligned} A_a^{\min}[t_{\text{now}}] &= A_a^{\max}[t_{\text{now}}] = A_a[t_{\text{now}}], \\ A_a^{\max}[t+1] &= \min\left\{A_a^{\max}, \overrightarrow{A}_a^{\max}\left(A_a^{\max}[t]\right)\right\}, \\ A_a^{\min}[t+1] &= \max\left\{A_a^{\min}, \overrightarrow{A}_a^{\min}\left(A_a^{\min}[t]\right)\right\}. \end{aligned}$$

The resulting boundaries of subordinate agents can then be combined to find intervals that the intermediary has to contribute in for each time step $t \in \mathcal{W}$: $L_\lambda^t = \bigoplus_{a \in \mathcal{A}_\lambda} [A_a^{\min}[t], A_a^{\max}[t]]$ where $[x_1, y_1] \oplus [x_2, y_2] = [x_1 + x_2, y_1 + y_2]$. Certainly, as t increases, $A_a^{\min}[t]$ and $A_a^{\max}[t]$ converge to $A_a^{\min}$ and $A_a^{\max}$ such that no further restriction is imposed. The behavior of an intermediary can be described (precisely) by the composition of all of its subordinate agents' models or as a single abstracted model. As we lose the information about the states of the underlying agents when turning to the abstracted model, temporal abstraction provides boundaries that exclude *definitely* infeasible schedules. Hence, some schedules that are consistent with the abstracted model might not lead to feasible schedules in the compositional view, but any schedule that is not accepted by the abstracted model is certainly not valid in the composition.

However, these boundaries still do not inform us about the actual inertia-representing functions $\overrightarrow{A}_\lambda^{\min}$ and $\overrightarrow{A}_\lambda^{\max}$ of the intermediary, that is, an *aggregate* of its underlying agents. For instance, reaching $\min L_\lambda^{t_1}$ at time step t_1 and $\max L_\lambda^{t_2}$ at time step $t_2 = t_1 + 1$ with $t_1, t_2 \in \mathcal{W}$ (where for an interval $i = [x, y]$, $\min i = x$ and $\max i = y$) might be perfectly legal with the previously obtained bounds L_λ^t although the transition could not be feasible. For similar reasons as in the previous example, it might be overly "optimistic" to assume the full change potential available, that is, a fixed change

$\Delta A_\lambda^{\max} = \sum_{a \in \mathcal{A}_\lambda} \Delta A_a^{\max}$ since some agents with high possible changes in contribution could already be at their respective maximum, whereas other (e.g., slower) agents still have more potential to steer toward their maximum. Our previous example (i.e., $A_{a_1}[t_{\text{now}}] = 95$, $A_{a_2}[t_{\text{now}}] = 45$, and thus $A_\lambda[t_{\text{now}}] = 140$) illustrated this circumstance: Following the reasoning, we could conclude that $A_\lambda[t_{\text{next}}] \leq 155$, $A_\lambda[t_{\text{next}}+1] \leq 165$, and $A_\lambda[t_{\text{next}}+2] \leq 175$. Only based on these constraints, a schedule $\langle 140, 140, 175 \rangle$ would be allowed for λ, although the transition from 140 to 175 is not possible in one time step.

6.2.3. Sampling Abstraction. In Schiendorfer et al. [2014], we addressed this problem using *sampling abstraction*, that is, we sampled the intermediary at several points in its contribution range by asking for the maximum contribution it can provide in the next step if its current contribution is set to each *sampling point*. Then, these sample points lead to a *piecewise linear function* that is used to approximate the actual min./max. successor functions.

However, this approach suffers from two drawbacks: First, we would need to solve a simplified CSOP (with only a scheduling window of two steps for the change from one step to another) with complexity $O(2^{2|\mathcal{A}_\lambda|})$ repeatedly (one time for each sampling point). In addition, this procedure leads to very high maximally possible changes since the optimal configuration for a certain contribution is sought (e.g., finding the max. successor function for $A_\lambda[t] = 140$ could return the previously discussed solution of $\langle 70, 70 \rangle$ and claiming 175 to be feasible for $A_\lambda[t+1]$).

By contrast, we need to approximate inertia based on the *actual* states of the underlying agents and use them not only for the current time step but extend them to all steps in $\mathcal{W}$. Nevertheless, we can reuse the same principle using the changes observed in temporal abstraction (as they do take the current states into account). More concretely, we use the set of pairs $\{(\max L_\lambda^{t-1}, \max L_\lambda^t) \mid t \in \mathcal{W}\}$ for the max. successor function and $\{(\min L_\lambda^{t-1}, \min L_\lambda^t) \mid t \in \mathcal{W}\}$ for the min. successor function. Note that $\min L_\lambda^{t_{\text{now}}} = \max L_\lambda^{t_{\text{now}}} = A_a[t_{\text{now}}]$. These pairs then induce a piecewise linear function representing $\overrightarrow{A}_\lambda^{\min}$ and $\overrightarrow{A}_\lambda^{\max}$ more conservatively in the intermediary's model at only polynomial costs $O(|\mathcal{A}_\lambda| \cdot H)$ for temporal abstraction. In our example, these functions would then correctly restrict a transition from 140 to maximally 155—regardless of the time step the transition is performed in.

7. EVALUATION

In the context of our case study, we evaluate TruCAOS$^+$ in three different settings. In the first setting regarded in Section 7.2, we investigate in which way TruCAOS$^+$ can deal with uncertain demand and contributions. Subsequently, we analyze TruCAOS$^+$'s performance in a hierarchy of AVPPs in Section 7.3. In Section 7.4, we examine the quality of the control models intermediaries create for their subordinate agents using the SVDD-based approach. For each single evaluation scenario, we performed 200 simulation runs. All presented results are average values. Before we discuss our results, we introduce our general test bed.

7.1. General Test Bed

We base our evaluation on a hierarchical structure of AVPPs, consisting of power plants of different types (hydro, biofuel, gas power plants as well as solar plants and wind generators). While each power plant is modeled as an individual agent, we only use a single agent to represent all consumers. Weather-dependent power plants and the consumer constitute the environment *env*. Physical power plant data[10]—such as

[10]EnergyMap (Bavaria), 2012: http://www.energymap.info

production boundaries—, load curves[11], and weather data[12] influencing the output of weather-dependent power plants are based on real-world data (we scaled the residual load dependent on the number of power plants, their maximum output, as well as their inertia in the different evaluation scenarios). Each dispatchable power plant's inertia is defined within typical boundaries. The production costs, that is, the average costs of providing a contribution, range from $6.50\frac{\text{EUR cent}}{\text{kWh}}$ to $17.50\frac{\text{EUR cent}}{\text{kWh}}$. When making proposals, the intermediaries base their average costs on the subsystem's average costs of accepted proposals $\overline{\kappa}^{\text{avg}}_{\mathcal{A}_\lambda}$ that is calculated over the last five time steps (see Section 5.2.1). With regard to the DRAP, power plants have to satisfy a prescribed residual load over an evaluation-scenario-specific period of $\mathcal{T}$ time steps with a resolution of 15min (a standard practice in current power management systems). Schedules are created for the next $H = 4$ time steps, that is, for the next hour.

The evaluation is implemented in a sequential, round-based execution model in which each round corresponds to a specific time step $t \in \mathcal{T}$. At the beginning of each round, the top-level AVPP Λ requests residual load predictions $P_{env}[t]$ from the environment for the next $H = 4$ time steps. Subsequently, all agents in the hierarchy of AVPPs solve the DRAP as explained in Sections 4 and 5. The remaining residual load is satisfied in case its absolute values are below $R_{\max} = 1\text{kW}$. According to the DRAP and the regarded time frame $\mathcal{T}$, the system's goal is to minimize the aggregated absolute deviation $DEV = \sum_{t\in\mathcal{T}} |A_{env}[t] - A_{\mathcal{A}}[t]|$ between the actual residual load $A_{env}[t]$ and the total actual contribution of resources $A_{\mathcal{A}}[t]$ over all time steps $t \in \mathcal{T}$. Further, the costs of providing the resources should be kept down.

The optimization problems of determining winner proposals (see Section 4.2.3) and leaving local optima (see Section 4.3) are formulated for and solved with the standard mathematical programming software IBM ILOG CPLEX.[13]

Further, we also formulated all power plant models as well as the CSOP of generating proposals for CPLEX. In the context of the proposal generation, the power plants' objective is to minimize the remaining residual load specified in the intermediaries' call for proposals.

We compare the results obtained with TruCAOS$^+$ to the solutions found by the regio-central approach (referred to as "CPLEX" in the following) explained in Section 3 that basically uses CPLEX to solve the CSOP introduced in Section 2.5. As the utmost goal is to satisfy the residual load, followed by the goal to provide energy at a low price, we use $\alpha_\Delta = 100.0\frac{\text{EUR cent}}{\text{kWh}}$ and $\alpha_\Gamma = 1.0$ (regarding Equation (1), note that the absolute deviations Δ are given in kWh and the costs $\kappa_a(S_a[t])$ of a scheduled contribution $S_a[t]$ are given in EUR cent). It is thus more expensive to not produce a requested amount of energy than to produce it. Note that the CPLEX-based regio-central approach yields—with regard to a specific AVPP—optimal results.

7.2. Dealing with Uncertain Demand and Contributions

To evaluate TruCAOS$^+$'s performance under uncertainty, we focus on a system consisting of a single AVPP. We compare the results we achieved with TruCAOS$^+$ in two different scenarios. In the first scenario "S-NT," TruCAOS$^+$ does not make any use of trust values, whereas it uses trust values in the second scenario "S-T." After introducing our test bed in more detail, we present our evaluation results in Section 7.2.2.

---

[11]LEW, 2012: http://www.lew-verteilnetz.de

[12]LfL (Bavaria), 2010: http://www.lfl.bayern.de

[13]IBM ILOG CPLEX Optimizer, Version 12.4, 2011: http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/

7.2.1. Test Bed. The single AVPP controls 20 dispatchable power plants (hydro, biofuel, and gas power plants) with different physical properties. Together, these power plants have to satisfy a predefined actual residual load $A_{env}[t] \in [1{,}342\text{kW}, 2{,}157\text{kW}]$, that is, a demand, over a period of four days, which corresponds to 384 discrete time steps $t \in \mathcal{T} = \{1, \ldots, 384\}$.

The environment creates residual load predictions for the $H = 4$ future time steps by adding a random prediction error—generated by using a Gaussian distribution—to $A_{env}[t]$. The residual load shows two different behaviors env_1 and env_2. For env_1, we use a mean prediction error of $\mu_{env_1} = 50\text{kW}$ and a standard deviation of $\sigma_{env_1} = 5\text{kW}$. The residual load behaves according to env_1 in time steps $t \in \{1, \ldots, 48\} \cup \{97, \ldots, 144\}$. In time steps $t \in \{49, \ldots, 96\} \cup \{145, \ldots, 384\}$, we use env_2 with $\mu_{env_2} = -100\text{kW}$ and $\sigma_{env_2} = 10\text{kW}$. Uncertainty regarding future time steps is implemented by increasing the mean prediction error with the time horizon of predictions by 10kW for env_1 and $|-20\text{kW}|$ for env_2.

In S-T, the AVPP uses the trust metrics presented in Section 5.1 ($n_{\mathcal{A}} = 100$, $n'_{\mathcal{A}} = 5$, $n_{env} = 50$, and $n'_{env} = 5$ proved to be beneficial in preliminary tests) to determine the expected demand $E_{env}[t]$ and the expected contributions $E^{(*)}_{\alpha}[t]$. The evaluation shows that, due to our trust metrics, $E_{env}[t]$ is 81.30% more accurate than $P_{env}[t]$.

Except for a "Malicious Power Plant" (MPP) with a maximum output of $A^{\max}_{\text{MPP}} = 450\text{kW}$ (this is about a fifth of the maximum residual load in this setting), all power plants behave benevolently, meaning that they comply with their predicted contributions. The MPP predicts future outputs $P_{\text{MPP}}[t]$ that deviate from its actual and scheduled output $A_{\text{MPP}}[t] = S_{\text{MPP}}[t]$ according to two different predefined behaviors mpp_1 and mpp_2. Regarding mpp_1 / mpp_2, the MPP adds a random prediction error that is, on average, 30% / 10% higher than predicted if $P_{\text{MPP}}[t] = A^{\min}_{\text{MPP}} = 0\text{kW}$, and, on average, 30% / 10% lower than predicted if $P_{\text{MPP}}[t] = A^{\max}_{\text{MPP}} = 450\text{kW}$. Again, random numbers are generated by using a Gaussian distribution with a standard deviation of $\sigma_{mpp_1} = 13.5\text{kW}$ / $\sigma_{mpp_2} = 4.5\text{kW}$. Between $A^{\min}_{\text{MPP}}$ and $A^{\max}_{\text{MPP}}$, the prediction error changes linearly with $P_{\text{MPP}}[t]$ such that the power plant does not deviate from predictions at its sweet spot $P_{\text{MPP}}[t] = 225\text{kW}$. The MPP shows behavior mpp_1 in time steps $t \in \{1, \ldots, 72\}$ and behavior mpp_2 in time steps $t \in \{73, \ldots, 384\}$. In S-T, we use $\Delta c = 112.5\text{kW}$, that is, 25% of $A^{\max}_{\text{MPP}}$, as contribution range to estimate the predictability of MPP. To investigate TruCAOS$^+$'s decisions with regard to the MPP, the AVPP contains a power plant, called the MPP's "twin" in the following, that is identical to the MPP but behaves benevolently.

7.2.2. Results. In S-NT, the AVPP allocates resources according to the predicted residual load P_{env}. As depicted in Figure 11, the mean absolute deviation $\Delta(A_{env}, P_{env})$ between the actual A_{env} and the predicted P_{env} residual load varies with the environment's behavior env_1 and env_2 (the average of $\Delta(A_{env}, P_{env})$ is 60.87kW, the standard deviation $\sigma = 15.56\text{kW}$). Regarding S-T, the AVPP assesses the quality of predictions by means of the trust metrics and solves the DRAP on the basis of the expected residual load E_{env}. The small mean absolute deviation $\Delta(A_{env}, E_{env})$ between A_{env} and E_{env} indicates that the AVPP is able to anticipate systematic prediction errors, that is, epistemic uncertainties, very accurately (the average of $\Delta(A_{env}, E_{env})$ is 11.38kW, $\sigma = 11.66\text{kW}$).

The rather high values of $\Delta(A_{env}, E_{env})$ around the time steps 48, 96, and 144 coincide with the change in the environment's behavior. While $\Delta(A_{env}, E_{env})$ sometimes exceeds $\Delta(A_{env}, P_{env})$ at these time steps, the short time frames in which these deviations are present indicate that the trust values quickly adapt to behavioral changes. This can be attributed to our trust metric that weights experiences according to their age. Further, only the last n_{env} experiences are in line for the calculation of the trust value.

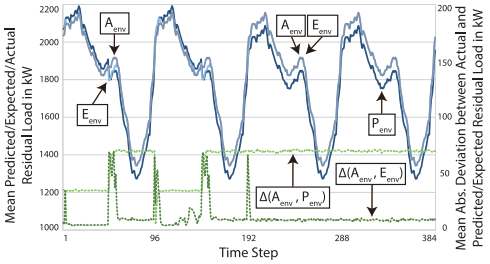

Fig. 11. Mean predicted P_{env}, expected E_{env}, and actual A_{env} residual load. The mean absolute deviation $\Delta(A_{env}, P_{env})$ between A_{env} and P_{env} shows the predefined randomly generated prediction error. The mean absolute deviation $\Delta(A_{env}, E_{env})$ between A_{env} and E_{env} reflects the accuracy of E_{env}, which was calculated by means of the presented trust metrics in S-T.

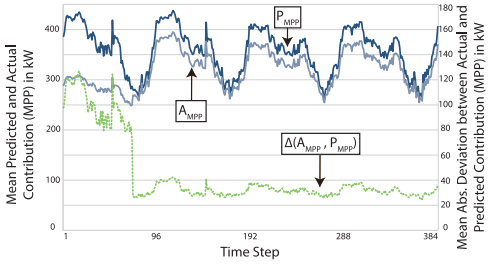

Fig. 12. The predicted P_{MPP} and actual A_{MPP} contribution of the MPP in S-NT. The mean absolute deviation $\Delta(A_{\text{MPP}}, P_{\text{MPP}})$ between A_{MPP} and P_{MPP} mirrors the MPP's varying and contribution-specific behavior.

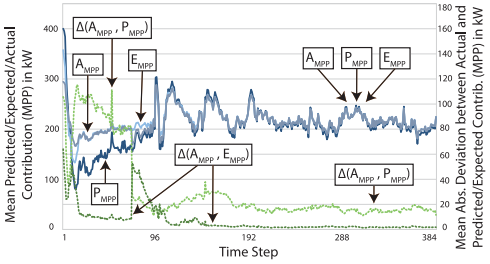

Fig. 13. The predicted P_{MPP}, expected E_{MPP}, and actual A_{MPP} contribution of the MPP in S-T. As in Figure 12, the mean absolute deviation $\Delta(A_{\text{MPP}}, P_{\text{MPP}})$ between A_{MPP} and P_{MPP} mirrors the MPP's varying behavior. The small mean absolute deviation $\Delta(A_{\text{MPP}}, E_{\text{MPP}})$ between A_{MPP} and E_{MPP} shows that TruCAOS^{+}'s trust-based decisions mitigate epistemic uncertainties.

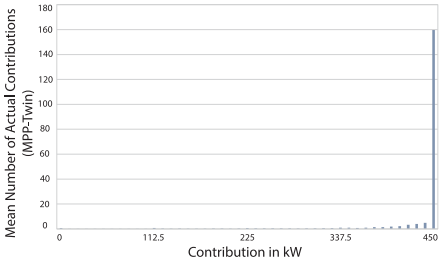

Fig. 14. The number of actual contributions of the MPP's twin for time steps between 192 and 384 (the width of the intervals is 10kW), resulting in a mean actual contribution of 430.59kW with a standard deviation of $\sigma = 62.50$kW.

With regard to S-NT, the predicted contribution P_{MPP} of the MPP correlates with the predicted residual load P_{env} (see Figure 12), which is also mirrored by the shapes of the two corresponding curves. In time step 72, the change in MPP's behavior from mpp_1 to mpp_2 is clearly visible. As the mean absolute deviation $\Delta(A_{\text{MPP}}, P_{\text{MPP}})$ of actual contributions A_{MPP} from predicted contributions P_{MPP} depends on the MPP's predicted output, $\Delta(A_{\text{MPP}}, P_{\text{MPP}})$ varies with P_{MPP}. The MPP's malicious behavior causes an average of $\Delta(A_{\text{MPP}}, P_{\text{MPP}})$ of 44.25kW with $\sigma = 27.55$kW.

With respect to S-T, Figure 13 depicts how the AVPP allocates resources of the MPP in case trust-based decisions are made. The AVPP needs approximately 18 time steps until the trust value accurately reflects the MPP's behavior. After these time steps, TruCAOS^{+}'s trust-based principle allows the AVPP to significantly reduce the deviation between the expected E_{MPP} and the actual A_{MPP} contribution. The MPP's malicious behavior causes an average of $\Delta(A_{\text{MPP}}, E_{\text{MPP}})$ of only 7.34kW with $\sigma = 10.77$kW. At time step 72, the MPP changes its behavior, which is reflected in the peak of $\Delta(A_{\text{MPP}}, E_{\text{MPP}})$. About 20 time steps are needed to adapt to the new behavior. In the course of these time steps, E_{MPP} deviates from A_{MPP} more than P_{MPP} does. However, starting at time

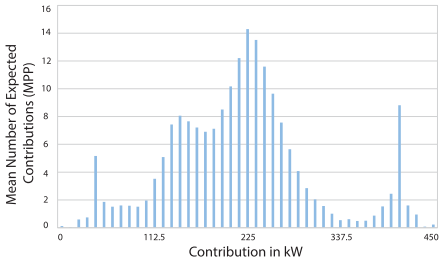

Fig. 15. The number of expected contributions of MPP for time steps between 192 and 384 (the width of the intervals is 10kW), resulting in a mean expected contribution of 215.66kW with a standard deviation of $\sigma = 84.91$kW.

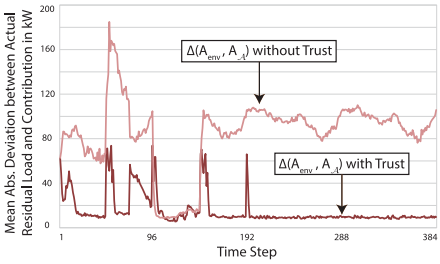

Fig. 16. The mean absolute deviation $\Delta(A_{env}, A_A)$ between the actual residual load A_{env} and the total actual contribution A_A in S-NT compared to S-T. TruCAOS$^+$'s trust-based decisions significantly decrease $\Delta(A_{env}, A_A)$.

step 144, the curves of A_{MPP}, P_{MPP}, and E_{MPP} almost coincide. This is mainly because the AVPP allocates the resources of the MPP such that P_{MPP} is near its sweet spot of 225kW. At its sweet spot, the MPP does not deviate from its predictions and is thus very predictable (the average of $\Delta(A_{\text{MPP}}, P_{\text{MPP}})$ is 31.43kW with $\sigma = 28.93$kW). This behavior is also visualized in the histograms in Figures 14 and 15: While the MPP's *twin* is almost always allowed to contribute with its maximum capacity (see Figure 14), the contributions of the MPP are shifted to the area around its sweet spot (see Figure 15). This shows that TruCAOS$^+$'s trust-based decisions actively and effectively mitigate the presence of aleatoric uncertainties by accepting contributions in a way that allows one to increase the agents' predictability. Because the contributions of the MPP are much lower than those of its twin, TruCAOS$^+$'s trust-based decisions incentivize agents to behave beneficially.

Figure 16 summarizes our observations: By using trust-based techniques in S-T, the average of the mean absolute deviation $\Delta(A_{env}, A_A)$ between A_{env} and the total actual contribution A_A is only 15.74kW ($\sigma = 14.24$kW), compared to 84.27kW ($\sigma = 31.76$kW) without trust-based measures in S-NT. In S-T, this results in an aggregated absolute deviation of $DEV = 6{,}044.55$kW. This is 81.32% lower than in S-NT, where we have $DEV = 3{,}2360.15$kW. On average, TruCAOS$^+$ terminated after 5.58 ($\sigma = 0.44$) bidding iterations in S-NT and 5.55 ($\sigma = 0.40$) iterations in S-T.

To judge TruCAOS$^+$'s performance with regard to scheduling times and production costs, we additionally evaluate this setting *without uncertainties* and compare the results to those achieved with the regio-central approach. In this additional evaluation scenario, TruCAOS$^+$ as well as CPLEX can satisfy the demand in all time steps. CPLEX needs 260.10ms ($\sigma = 66.42$ms) on average to create schedules for all dispatchable power plants. By incorporating cheap hydro power plants only (this is possible because their aggregated maximum output exceeds the maximum residual load), CPLEX obtains optimal average costs of $6.50\frac{\text{EUR cent}}{\text{kWh}}$ ($\sigma = 0.0\frac{\text{EUR cent}}{\text{kWh}}$) to satisfy the demand. While TruCAOS$^+$ also achieves these optimal costs in this evaluation scenario, the mean scheduling time is only 57.13ms ($\sigma = 15.23$ms). In this setting, TruCAOS$^+$ is thus 78.04% faster than CPLEX. Please note that, in the *worst case*, the algorithms could choose an allocation of resources that perfectly satisfies the demand but leads to mean costs of $17.50\frac{\text{EUR cent}}{\text{kWh}}$, which are 169.23% higher than those found by TruCAOS$^+$ and CPLEX.

Although we only regarded a single malicious agent in our empirical analysis, we expect our system to behave in a very similar way in the presence of multiple malicious agents. This means that we expect the system to shift the contributions of malicious

Table I. Evaluation Results for the Regio-Central CPLEX-Based Approach and TruCAOS$^+$ (We Use the Shorter Form TruC$^+$ in the Table) for Different Heights of the Hierarchical Structure of AVPPs

All Results are Averaged over 200 Simulation Runs for Each Scenario. Values in Parentheses Denote Standard Deviations. As before, "DEV" Stands for the Aggregated Absolute Deviation Between the Contributions and the Demand.

Height of the AVPP Hierarchy	1 (superflat)		2 (flat)		3 (deep)		5 (deeper)	
	CPLEX	TruC$^+$	CPLEX	TruC$^+$	CPLEX	TruC$^+$	CPLEX	TruC$^+$
Mean #Dispatchable Power Plants per AVPP	173.00 (0.00)	173.00 (0.00)	14.23 (0.00)	14.23 (0.00)	6.55 (0.00)	6.55 (0.00)	2.42 (0.00)	2.42 (0.00)
Max. Seq. Scheduling Time [ms]	13,612.19 (6,264.49)	228.89 (59.87)	405.96 (139.57)	369.08 (645.09)	412.03 (143.70)	510.18 (321.91)	467.26 (128.86)	740.50 (464.30)
Mean Scheduling Time per AVPP [ms]	13,612.19 (6,264.49)	228.89 (59.87)	131.35 (70.38)	46.29 (5.08)	72.30 (49.25)	33.14 (2.70)	40.17 (33.48)	18.35 (1.79)
Mean #Bidding Iterations per AVPP	–	3.97 (0.66)	–	2.49 (0.53)	–	3.01 (0.53)	–	1.88 (0.51)
Mean #Detected Local Optima per Time step	–	0.00 (0.00)	–	0.18 (0.04)	–	2.96 (0.20)	–	14.24 (1.31)
Mean DEV per Simulation Run [kW]	0.00 (0.00)	2.40 (0.02)	0.00 (0.00)	6.98 (0.05)	926.97 (479.21)	80.99 (2.75)	2,905.38 (1,144.81)	140.93 (1.44)
Mean Actual Costs $\left[\frac{\text{EUR cent}}{\text{kWh}}\right]$	10.32 (1.11)	10.32 (1.11)	11.22 (0.74)	11.46 (0.91)	12.07 (0.49)	12.69 (0.72)	12.71 (0.29)	12.69 (0.61)

agents into ranges in which they behave particularly trustworthy as long as a sufficient amount of trustworthy agents is present that is able to compensate for these contribution adjustments.

7.3. Solving the Dynamic Resource Allocation Problem in Hierarchical Systems

In this section, we investigate the impact of the height of the AVPPs' hierarchical structure on TruCAOS$^+$'s performance, for example, with regard to scheduling times.

7.3.1. Test Bed. TruCAOS$^+$as well as the CPLEX-based regio-central approach solve the DRAP in a hierarchy of AVPPs with a predefined height of 1 ("superflat"), 2 ("flat"), 3 ("deep"), and 5 ("deeper"). We randomly generated these structures before the actual evaluation so that a structure of a specific height does not change from one simulation run to another. All four hierarchies consist of the same 173 dispatchable power plants that have to satisfy the residual load over a period of one day, which corresponds to 96 discrete time steps $t \in \mathcal{T} = \{1, \ldots, 96\}$. As the regio-central approach does not deal with uncertainties, we do not regard uncertainties in this evaluation setting, that is, the power plants as well as the environment adhere to their predictions.

As we focus on different performance attributes, we assume that AVPPs have perfect knowledge about their subordinate power plants' control model. That way, TruCAOS$^+$ as well as the regio-central approach have the same information to solve the DRAP.

7.3.2. Results. Table I shows the results for the different hierarchical structures. Unsurprisingly, CPLEX struggles with the number of power plants schedules have to be created for in the "superflat" scenario. While CPLEX needs 13,612.19ms to create valid schedules, TruCAOS$^+$ only needs impressive 228.89ms, which is 98.32% faster. As discussed in Section 4, that is because power plants have to solve the DRAP only for a

single agent when using TruCAOS$^+$. Further, TruCAOS$^+$ is able to satisfy the demand at the same *optimal* costs of $10.32\frac{\text{EUR cent}}{\text{kWh}}$ as CPLEX. Please note that the cheapest power plants have production costs of $6.50\frac{\text{EUR cent}}{\text{kWh}}$ and that, in the *worst case*, the algorithms could opt for an allocation that accurately satisfies the demand but leads to mean costs of $17.33\frac{\text{EUR cent}}{\text{kWh}}$. Under these circumstances, TruCAOS$^+$'s results are quite remarkable.

CPLEX significantly benefits from the DRAP's decomposition in the case of the "flat" hierarchy. The maximum sequential scheduling time (i.e., the maximum of the sums of the scheduling times in each branch originating from the root of the AVPP structure) drops to 405.96ms and an AVPP only needs 131.35ms on average to calculate schedules. We can explain this by the lower number of power plants (14.23 on average compared to 173) schedules have to be created for. On the other hand, the costs slightly increase by 8.72% due to abstraction and indirections. While the mean scheduling time per AVPP significantly decreases to 46.29ms for TruCAOS$^+$, we see that the mean max. seq. scheduling time increases by 61.25% compared to the "superflat" hierarchy. We explain this behavior by two observations: First, due to the complexity of the AVPPs' control models, AVPPs need more time for generating proposals than physical power plants. Second, additional hierarchy layers introduce additional overhead, for example, because multiple AVPPs now have to select suitable proposals. However, in spite of the increase of the max. seq. scheduling time, TruCAOS$^+$ still yields max. seq. scheduling times that are *better* than those of CPLEX. The costs TruCAOS$^+$ obtains are still comparable to those of CPLEX.

As can be seen in Table I, with regard to TruCAOS$^+$, the max. seq. scheduling time continues to grow, while the mean scheduling time per AVPP further decreases for the hierarchies "deep" and "deeper." The costs TruCAOS$^+$ achieves remain comparable to the costs of CPLEX. Since the scheduling times also increase for CPLEX in these scenarios, we can conclude that—with respect to CPLEX—the hierarchy "flat" is a sweet spot that allows one to achieve good scheduling times at low costs.

Interestingly, the aggregated absolute deviation between the power plants' output and the residual load increases with the height of the hierarchy, that is, the utmost goal to hold the balance between production and consumption cannot be properly achieved. This can be attributed to the increased fragmentation of the system and errors introduced by abstraction (we already mentioned this phenomenon in Section 3), which is mirrored by the higher number of local optima detected by the AVPPs in deeper hierarchies. While this deviation can get very high when using CPLEX (2,905.38kW for the hierarchy "deeper"), in TruCAOS$^+$, intermediaries try to iteratively compensate for this deviation as best as possible (140.93kW for the hierarchy "deeper"), which is also reflected in the longer max. seq. scheduling times.

Summarizing, as TruCAOS$^+$ allows big AVPPs to create schedules in short time, it prevents the system's fragmentation that goes along with high costs and imbalances.

7.4. Accuracy of SVDD-Based Control Models

In this section, we evaluate the SVDD-based approach we presented in Section 6 to create control models of subordinate agents.

7.4.1. Test Bed. The evaluation of the SVDD-based approach is performed in a setting similar to the one used in Section 7.2.1: The physical power plants as well as the residual load that has to be satisfied over a period of four days (i.e., $\mathcal{T} = \{1, \ldots, 384\}$) are the same. However, the system consists of two instead of a single AVPP. While the top-level AVPP Λ has five subordinate physical dispatchable power plants, its subordinate AVPP λ controls 15. In this setting, λ makes use of the SVDD-based approach (we used the SVDD implementation provided by the *libsvm* library [Chang and Lin 2011]) to

Table II. Results for the Evaluation of the SVDD-Based Approach
With Regard to TruCAOS$^+$, We Compare a Setting in which AVPPs have Perfect Knowledge about Existing Control Models to Two Settings with Learned Control Models. In the Latter Case, we use Manually Fixed Parameters in the Case of SVDD-FP and Parameters that were Determined at Runtime in the Case of SVDD-SP. All Results are Averaged Over 200 Simulation Runs. Values in Parentheses Denote Standard Deviations. As Before, "DEV" Stands for the Aggregated Absolute Deviation Between the Contributions and the Demand.

	CPLEX perfect knowledge	TruCAOS$^+$ perfect knowledge	TruCAOS$^+$ SVDD-FP	TruCAOS$^+$ SVDD-SP
Mean *DEV* per Simulation Run [kW]	0.00 (0.00)	11.19 (0.02)	10.68 (0.02)	10.65 (0.02)
Mean Actual Costs $\left[\frac{\text{EUR cent}}{\text{kWh}}\right]$	9.83 (1.05)	9.59 (1.08)	9.60 (1.10)	9.60 (1.10)

learn the control models of its own subordinate power plants. The data used to train the SVDD, that is, the positive examples, are the power plants' actual contributions as well as their predicted schedules (see Figure 5). As discussed in Section 6.1.1, the quality of the results an SVDD yields hinge on a suitable choice of the parameters C and γ.

In the following, we compare two different configurations: one called SVDD-SP, that uses k-fold cross-validation to find reasonable values for C in a range of $[\frac{1}{l}, 1]$ (where l is the number of training examples) as well as for γ in a range of $[1, 8]$ at runtime (as explained in Section 6.1.1, we use schedule corrections as negative examples), and another one that uses manually fixed $C = 0.10$ and $\gamma = 1.90$ found from manual preliminary testing (called SVDD-FP). As discussed in Section 6, during the initial phase of the system's runtime, λ has no data about its subordinate agents' contributions that could be used as training data. For our evaluation, we use a pragmatic solution: During the initial phase, we assume that λ knows the exact control models of its subordinate agents until a certain number of contributions have been recorded. Concretely, in SVDD-SP and SVDD-FP, λ gathers contributions during the first 96 time steps of the simulation before it learns the SVDD-based control models for the first time. The SVDD-based control models are then periodically recreated every 48 time steps. In addition to the training, λ performs k-fold cross-validation in the case of SVDD-SP.

7.4.2. Results. We observed no significant differences between running the system with perfect knowledge about existing control models compared to learned ones—which mainly results from the small size of the considered system (see Table II). However, the type (and consequently the associated costs) of power plants affect the quality of the extracted control models drastically. Table III(a) shows that both SVDD-FP and SVDD-SP can approximate the control models' parameters with a mean accuracy of more than 94% already after 96 time steps. After 384 time steps, in SVDD-SP, λ automatically selects $C = 0.014$ ($\sigma = 0.00$), and $\gamma = 1.60$ ($\sigma = 1.24$) as the optimal parameters, which comes close to the preselected choices found by manual testing. Apparently, these parameters work well for the given problem since the learning performance with regard to the accuracy is higher with a fixed parameter set. This indicates that the exact combination ($C = 0.10$, $\gamma = 1.90$) is either not inspected during grid search or does not perform well enough on the validation sets. Therefore, the resolution of the grid could be increased along with collecting more outliers—if possible. Furthermore, both scenarios indicate that more training data increase the accuracy of the extracted constraint models.

Table III(b) draws a different picture when considering hydro power plants that—as they are comparably cheap ($6.5\frac{\text{EUR cent}}{\text{kWh}}$ compared to $17.5\frac{\text{EUR cent}}{\text{kWh}}$ for biofuel power plants)—are preferably selected when they are at their maximum contribution level.

Table III. The Accuracy of the Learned Control Models of Biofuel/Hydro Power Plants with Regard to the Minimum $A^{\min}_{\text{Biofuel/Hydro}}$ and Maximum $A^{\max}_{\text{Biofuel/Hydro}}$ Contribution, as Well as the Constant Maximal Rate of Change $\Delta A^{\max}_{\text{Biofuel/Hydro}}$

The Accuracy is Measured in Percent Relative to the Actual Values. Further, the Table Shows the Number of Training Examples as well as the Number of Outliers used to Perform the *k*-Fold Cross-Validation. SVDD-FP96 and SVDD-SP96 Represent the Results after the First Model Creation in Time step 96, whereas SVDD-FP384 and SVDD-SP384 Represent the Results for Time step 384. Values in Parentheses Denote Standard Deviations.

(a) The accuracy of learned control models of biofuel power plants

SVDD	$A^{\min}_{\text{Biofuel}}$	$A^{\max}_{\text{Biofuel}}$	$\Delta A^{\max}_{\text{Biofuel}}$	#Training Exa.	#Outliers
SVDD-FP96	99.97% (0.16%)	98.61% (0.36%)	95.48% (8.46%)	230.79 (42.69)	–
SVDD-FP384	99.98% (0.12%)	98.70% (0.41%)	96.40% (7.78%)	436.65 (62.56)	–
SVDD-SP96	99.94% (0.35%)	98.64% (0.37%)	94.39% (8.54%)	231.48 (43.34)	11.67 (5.23)
SVDD-SP384	99.96% (0.25%)	98.70% (0.40%)	96.25% (7.89%)	433.77 (63.79)	22.13 (7.68)

(b) The accuracy of learned control models of hydro power plants

SVDD	$A^{\min}_{\text{Hydro}}$	$A^{\max}_{\text{Hydro}}$	$\Delta A^{\max}_{\text{Hydro}}$	#Training Exa.	#Outliers
SVDD-FP96	20.41% (15.20%)	99.08% (0.59%)	2.58% (2.10%)	64.37 (8.43)	–
SVDD-FP384	18.14% (7.03%)	99.32% (0.53%)	2.24% (0.96%)	81.92 (15.40)	–
SVDD-SP96	17.48% (9.51%)	99.14% (0.49%)	2.14% (1.33%)	64.45 (8.48)	0.00 (0.00)
SVDD-SP384	18.06% (7.58%)	99.27% (0.52%)	2.24% (1.07%)	80.74 (14.49)	0.00 (0.00)

Therefore, the resulting training sets do not vary to the same extent as those recorded in the case of biofuel power plants that affects, in particular, the accuracy of the extracted minimum contributions (comparing $A^{\min}_{\text{Biofuel}}$ and $A^{\min}_{\text{Hydro}}$) and rates of change that deviate by a percentage of about 98% on average. The problem we are facing is an interesting instance of *concurrent optimization*, which is typically found in interwoven systems where conflicting goals influence each other [Tomforde et al. 2014]. For our system's primary goal, steering cheap power plants at maximum is optimal (*exploitation*). But this also results in a very limited training set (duplicate training entries are removed) and therefore reduces the required *exploration*. Investigations on how to manage these concurrent and conflicting goals are subject to future work.

As the evaluation results show, our SVDD-based approach is able to learn the minimum and maximum output as well as the power plants' inertia very precisely (i.e., the constraints originally considered in Anders et al. [2013]) as long as sufficiently varying training examples are available.

8. RELATED WORK

While resource allocation has been an important topic of research, the incorporation of uncertainties is a relatively recent development. An important aspect of this is the institutionalization of procedural rules in the exploitation of Common Pool Resources (CPR), as considered by Pitt et al. [2011] based on Elinor Ostrom's Principles for Enduring Institutions. To sustainably exploit a CPR, agents need to cooperate as governed by a set of modifiable rules. Uncertainties arise when agents behave selfishly, for example, by consuming more resources than they were appropriated. In such cases, other agents (i.e., the institution they form) have to sanction them in a way that incentivizes them to behave better in the future. While the work of Pitt et al. [2011] is focused on endogenic resources that can be fully exhausted and have to be managed in a sustainable way, we are working with exogenic ones that are supplied constantly. It is also necessary that the full demand is covered by the agents' contributions even if that means that some agents have to contribute more than they would like to.

As a specialization of this kind of incentive-based mechanism design, Dash et al. [2004] introduce the notion of trust-based mechanism design, which adds the concept of trust to the factors costs and valuations in decision making. The presented algorithm is applied to a task allocation scenario in which agents post tasks to an auctioneer that in turn decides on an allocation on the basis of received proposals. When deciding on this allocation, the auctioneer takes the agents' trustworthiness into account, which is derived from experiences reported by their former interaction partners. The auctioneer's goal is to maximize the sum of the expected utilities of the agents. A trust-based payment and allocation scheme is introduced that chooses the most trustworthy and cheapest agents to fulfill an allocation, on the one hand, and incentivizes the agents to tell the truth about their costs, valuations, and the experiences with other agents, on the other hand. While Dash et al. [2004] served as a theoretical foundation for our investigations, our DRAP differs from the task allocation problem regarded there, for example, in its objective. In contrast to Dash et al. [2004], TruCAOS$^+$'s use of trust-based techniques allows it to identify and cope with agents that unintentionally show nonbeneficial behavior.

Based on the principle of trust-based mechanism design, the literature presents various marked-based approaches in the context of supply and demand management in smart grids that prevent strategic misbehavior and gambling through pricing mechanisms (e.g., Chalkiadakis et al. [2011] and Vytelingum et al. [2010]). In contrast to Vytelingum et al. [2010], TruCAOS$^+$ can proactively identify and cope with agents that unintentionally show nonbeneficial behavior. To reactively deal with unintentional prediction errors, Vytelingum et al. [2010] propose an additional online balancing mechanism that compensates for deviations between demand and contributions. The mechanism presented in Chalkiadakis et al. [2011] mitigate these uncertainties by forming coalitions of agents such that prediction errors cancel each other out.

In the domain of power management systems, there are multiple approaches that try to solve the scheduling problem on the basis of a definition that is similar to our DRAP. Apart from centralized algorithms that require a detailed model of the agents (see, e.g., Heo et al. [2006] and Zafra-Cabeza et al. [2008]), there are several decentralized approaches that are based on the principles of electronic markets (see, e.g., Wedde [2012], Kok et al. [2005], and Li et al. [2010]). DEZENT [Wedde 2012] is a market-based approach to balance energy supply and load in a hierarchical system structure in a bottom-up manner. In Section 6, we sketched why we prefer a top-down approach. A further approach based on a hierarchical system structure is the PowerMatcher [Kok et al. 2005] in which the root of a tree balances supply and load by determining an equilibrium price, based on aggregated load, supply, and price predictions, to establish a market equilibrium. The auctioneer is thus a central component of the system. Unlike TruCAOS$^+$, DEZENT and PowerMatcher only create schedules for the next instead of multiple future time steps. Stigspace [Li et al. 2010] is a coordination mechanism that uses a blackboard, called stigspace, as the medium of communication between distributed energy resources in order to create schedules in an iterative process. Initially, the stigspace is used to announce the load that has to be fulfilled by the distributed energy resources. These in turn revise their schedule (i.e., their contribution for the regarded time frame) in order to minimize the remaining load. The new schedule is then posted to the stigspace where the remaining load is updated accordingly. This process is repeated until the load is sufficiently satisfied. Because every schedule posted to the stigspace is "accepted," only the DRAP's primary objective—to allocate resources according to a given demand—might be achieved. DEZENT, PowerMatcher, Stigspace, and TruCAOS$^+$ have in common that the agents do not have to disclose their behavioral model. However, in contrast to TruCAOS$^+$, the other approaches do not deal with

uncertainties associated with load and supply predictions. With regard to open MAS, we therefore expect that TruCAOS$^+$ can solve the DRAP in a more robust way.

Hinrichs et al. [2013b] present a self-organizing heuristic, called COHDA$_2$, that solves a combinatorial optimization problem. The algorithm is based on a neighborhood structure in which agents iteratively negotiate about their configuration in order to achieve a global as well as individual objectives. The former can be thought of as the objectives of our DRAP. In COHDA$_2$, every time an agent changes its configuration, it informs its neighbors about its new configuration as well as the configuration of its neighbors and the neighbors' neighbors, etc., it received in a prior time step. Although each agent thus only has current information about its direct neighborhood, it builds a complete representation of the configuration of the other agents in the system over time. To decide on suitable own configurations, the agents take this locally perceived global configuration into account. As it is very likely that this configuration differs from the actual configuration of the other agents, the agents keep track of and inform their neighbors about the best configuration they have achieved so far. While COHDA$_2$ solves the combinatorial optimization problem in a truly decentralized manner, the quality of its solutions highly depends on the topology of the overlay network defining the agents' neighborhood. Similarly to TruCAOS$^+$, the agents do not disclose information about their behavioral model. However, in contrast to TruCAOS$^+$, COHDA$_2$ does not explicitly deal with uncertainties.

Bremer et al. [2011] and Hinrichs et al. [2013a] inspired us using data description and, in particular, SVDD to encode and describe feasible spaces of distributed energy resources. Whereas they considered devices as pure black boxes, we assume some properties about their behavior—the constraints shown in Section 2. These assumptions enable one to *extract* a formal control model that can then be used by an agent to calculate proposals and be thus integrated into TruCAOS$^+$. By contrast, Bremer used SVDD directly in an evolutionary algorithm (in particular, the fact that the SVDD is described by a set of support vectors and coefficients) because this algorithm class mainly relies on evaluating solution candidates, which can be done with an SVDD classifier and no semantic information. They can therefore consider 96-dimensional vectors representing feasible schedules (with time steps for one day sampled in 15min intervals). Since we assume the possible outputs to depend only on the previous time step, we are able to use two-dimensional vectors as training data. That way, we can avoid the curse of dimensionality as harvesting 96-dimensional examples would take a significant online running time—which is relevant to our case as the optimization problem we deal with may change over time. A future extension to a larger observation horizon to accommodate, for example, start-up times could, however, be fruitful.

The resource allocation problem described in Anders et al. [2012], is very similar to our DRAP. However, the algorithms in Anders et al. [2012] solve the problem by reacting to deviations between demand and contributions. While they could also be used to proactively allocate resources, they do not take uncertainties into account.

9. CONCLUSION AND FUTURE WORK

In this article, we introduced a dynamic resource allocation problem (DRAP) in the context of large-scale open multi-agent systems. Because we assume that the agents' contribution is subject to inertia—a typical property of physical devices, such as power generators—the DRAP has to be solved for a specific future time frame in advance. Based on the DRAP, we presented a **tru**st- and **c**ooperation-based **a**lgorithm for **o**pen MA**S**, called TruCAOS$^+$, that solves instances of the DRAP in an iterative and incremental process. To be able to scale with the number of agents that participate in the DRAP, TruCAOS$^+$can be applied to a self-organizing hierarchical system structure in which it solves the problem in a top-down manner. Because of TruCAOS$^+$'s

market-based principle, the agents do not have to disclose their behavioral model—a property that is of particular interest in open MAS. With regard to the hierarchical system structure, we showed that intermediaries can learn the control models of subordinate agents to be able to decide about the contribution of their subsystems. Since contribution and demand predictions might (un)intentionally deviate from their actual values, TruCAOS$^+$ further uses the concept of trust to form expectations of the actual contribution and demand on the basis of experiences gained in the past. Our evaluation shows that TruCAOS$^+$'s trust-based decisions allow one to anticipate epistemic as well as aleatoric uncertainties, effectively reduce epistemic as well as the presence of aleatoric uncertainties, sanction misbehavior, and incentivize cooperation. Therefore, TruCAOS$^+$ yields results that are robust in the sense that—despite the uncertainties and the dynamic nature of the DRAP—they satisfy the demand very accurately. We illustrated our investigations on the basis of a self-organizing power management system consisting of a hierarchy of AVPPs that have to create schedules for dispatchable power plants and consumers to maintain the balance between energy production and consumption. As stated in Section 2, other instances of the DRAP can be found in the domains of gas pipeline and water supply as well as district heating systems, among others.

In future work, we will improve the solutions' robustness by scheduling reserves on the basis of different possible developments of the demand (see second characteristic of robust solutions in Section 1 and Section 5.2.4). As discussed in Anders et al. [2014b], such solutions enable the combination of the proactive principle of TruCAOS$^+$ with a reactive mechanism, such as one of those proposed in Anders et al. [2012], to compensate for deviations between actual demand and contributions at runtime without having to recalculate schedules. Moreover, we will dig deeper into the creation of control models. In particular, we plan to investigate further techniques that allow one to derive more realistic and complex control models in the context of smart grids and uncertainties. Further, epistemic uncertainties introduced by model abstraction will be reduced by using an adaptive abstraction refinement that improves the abstract model in specific ranges.

REFERENCES

Gerrit Anders, Christian Hinrichs, Florian Siefert, Pascal Behrmann, Wolfgang Reif, and Michael Sonnenschein. 2012. On the influence of inter-agent variation on multi-agent algorithms solving a dynamic task allocation problem under uncertainty. In *Proceedings of the 2012 6th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO'12)*. IEEE Computer Society, Washington, D.C., 29–38.

Gerrit Anders, Alexander Schiendorfer, Jan-Philipp Steghöfer, and Wolfgang Reif. 2014a. Robust scheduling in a self-organizing hierarchy of autonomous virtual power plants. In *Proceedings of the 2014 27th International Conference on Architecture of Computing Systems (ARCS'14)*. 1–8.

Gerrit Anders, Florian Siefert, Michael Mair, and Wolfgang Reif. 2014b. Proactive guidance for dynamic and cooperative resource allocation under uncertainties. In *Proceedings of the 2014 IEEE 8th International Conference on Self-Adaptive and Self-Organizing Systems (SASO'14)*. 21–30. DOI:10.1109/SASO.2014.14

Gerrit Anders, Florian Siefert, Jan-Philipp Steghöfer, and Wolfgang Reif. 2014c. Trust-based scenarios—Predicting future agent behavior in open self-organizing systems. In *Self-Organizing Systems*, Lecture Notes in Computer Science, Vol. 8221, Wilfried Elmenreich, Falko Dressler, and Vittorio Loreto (Eds.). Springer Berlin Heidelberg, 90–102. DOI:10.1007/978-3-642-54140-7_8

Gerrit Anders, Jan-Philipp Steghöfer, Florian Siefert, and Wolfgang Reif. 2013. A trust- and cooperation-based solution of a dynamic resource allocation problem. In *Proceedings of the 2013 IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems (SASO'13)*. IEEE Computer Society, Washington, D.C., 1–10. DOI:10.1109/SASO.2013.33

Alexander Artikis, Marek Sergot, and Jeremy Pitt. 2009. Specifying norm-governed computational societies. *ACM Transactions on Computational Logic (TOCL)* 10, 1 (2009), 1.

Sulin Ba and Paul Pavlou. 2002. Evidence of the effect of trust building technology in electronic markets: Price premiums and buyer behavior. *MIS Quarterly* 26, 3 (2002), 243–268.

Sulin Ba, Andrew B. Whinston, and Hang Zhang. 1999. Building trust in the electronic market through an economic incentive mechanism. In *Proceedings of the 20th International Conference on Information Systems*. Association for Information Systems, 208–213.

Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph Naor, and Baruch Schieber. 2001. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM (JACM)* 48, 5 (2001), 1069–1090.

Atli Benonysson, Benny Bøhm, and Hans F. Ravn. 1995. Operational optimization in a district heating system. *Energy Conversion and Management* 36, 5 (1995), 297–314.

Marita Blank, Sebastian Gerwinn, Olav Krause, and Sebastian Lehnhoff. 2011. Support vector machines for an efficient representation of voltage band constraints. In *Proceedings of the 2nd IEEE PES International Conference and Exhibition on Innovative Smart Grid Technologies (ISGT Europe)*. IEEE, 1–8.

Jörg Bremer, Barbara Rapp, and Michael Sonnenschein. 2011. Encoding distributed search spaces for virtual power plants. In *2011 IEEE Symposium on Computational Intelligence Applications In Smart Grid (CIASG)*. 1–8. DOI: http://dx.doi.org/10.1109/CIASG.2011.5953329

Georgios Chalkiadakis, Valentin Robu, Ramachandra Kota, Alex Rogers, and Nicholas R. Jennings. 2011. Cooperatives of distributed energy resources for efficient virtual power plants. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'11)*, Vol. 2. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 787–794.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2, 3 (2011), 27:1–27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

Yann Chevaleyre, Paul E. Dunne, Ulle Endriss, Jérôme Lang, Michel Lemaître, Nicolas Maudet, Julian Padget, Steve Phelps, Juan A. Rodríguez-aguilar, and Paulo Sousa. 2006. Issues in multiagent resource allocation. *Informatica* 30, 1 (2006), 3–31.

Rajdeep K. Dash, Sarvapali D. Ramchurn, and Nicholas R. Jennings. 2004. Trust-based mechanism design. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*. IEEE Computer Society, 748–755.

Christian Derksen, Tobias Linnenberg, Rainer Unland, and Alexander Fay. 2013. Unified energy agents as a base for the systematic development of future energy grids. In *Multiagent System Technologies*, Lecture Notes in Computer Science, Vol. 8076, Matthias Klusch, Matthias Thimm, and Marcin Paprzycki (Eds.). Springer, Berlin, 236–249. DOI: http://dx.doi.org/10.1007/978-3-642-40776-5_21

Florian Dötsch, Jörg Denzinger, Holger Kasinger, and Bernhard Bauer. 2010. Decentralized real-time control of water distribution networks using self-organizing multi-agent systems. In *Proceedings of the 2010 4th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*. 223–232. DOI: http://dx.doi.org/10.1109/SASO.2010.20

Rui He, Jianwei Niu, and Guangwei Zhang. 2005. CBTM: A trust model with uncertainty quantification and reasoning for pervasive computing. In *Parallel and Distributed Processing and Applications*, Yi Pan, Daoxu Chen, Minyi Guo, Jiannong Cao, and Jack Dongarra (Eds.). Lecture Notes in Computer Science, Vol. 3758. Springer, Berlin, 541–552.

Jin S. Heo, Kwang Y. Lee, and Raul Garduno-Ramirez. 2006. Multiobjective control of power plants using particle swarm optimization techniques. *IEEE Transactions on Energy Conversion* 21, 2 (2006), 552–561.

Christian Hinrichs, Jörg Bremer, and Michael Sonnenschein. 2013a. Distributed hybrid constraint handling in large scale virtual power plants. In *Proceedings of the 2013 4th IEEE/PES Innovative Smart Grid Technologies Europe (ISGT EUROPE)*. 1–5. DOI: http://dx.doi.org/10.1109/ISGTEurope.2013.6695312

Christian Hinrichs, Michael Sonnenschein, and Sebastian Lehnhoff. 2013b. Evaluation of a self-organizing heuristic for interdependent distributed search spaces. In *Proceedings of the International Conference on Agents and Artificial Intelligence (ICAART'13)*, Volume 1—Agents. SciTePress, 25–34.

Trung Dong Huynh, Nicholas R. Jennings, and Nigel R. Shadbolt. 2006. An integrated trust and reputation model for open multi-agent systems. *Journal of Autonomous Agents and Multi-Agent Systems* 13, 2 (2006), 119–154.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2014. *An Introduction to Statistical Learning: with Applications in R*. Springer, New York.

Catholijn Jonker and Jan Treur. 1999. Formal analysis of models for the dynamics of trust based on experiences. In *Multi-Agent System Engineering*, Francisco Garijo and Magnus Boman (Eds.), Lecture Notes in Computer Science, Vol. 1647. Springer, Berlin, 221–231.

Piotr Juszczak, David M. J. Tax, and Robert P. W. Duin. 2002. Feature scaling in support vector data description. In *Proceedings of the ASCI 2002 8th Annual Conference of the Advanced School for Computing and Imaging*. Citeseer, 95–102.

Sarah Keung and Nathan Griffiths. 2008. Towards improved partner selection using recommendations and trust. In *Trust in Agent Societies*, Rino Falcone, Suzanne Barber, Jordi Sabater-Mir, and Munindar Singh (Eds.). Lecture Notes in Computer Science, Vol. 5396. Springer, Berlin, 43–64.

Rolf Kiefhaber, Gerrit Anders, Florian Siefert, Theo Ungerer, and Wolfgang Reif. 2012. Confidence as a means to assess the accuracy of trust values. In *Proceedings of the IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE Computer Society, 690–697.

Paul Klemperer. 2002. What really matters in auction design. *Journal of Economic Perspectives* 16, 1 (2002), 169–189. DOI:http://dx.doi.org/doi:10.1257/0895330027166

Koen Kok, Cor Warmer, and René Kamphuis. 2005. PowerMatcher: Multiagent control in the electricity infrastructure. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'05)*. ACM, New York, NY, 75–82. DOI:http://dx.doi.org/10.1145/1082473.1082807

Michael Koller. 1988. Risk as a determinant of trust. *Basic and Applied Social Psychology* 9, 4 (1988), 265–276.

Jiaming Li, Geoffrey Poulton, and Geoffrey James. 2010. Coordination of distributed energy resource agents. *Applied Artificial Intelligence* 24, 5 (2010), 351–380.

Benjamin Mack and Björn Waske. 2011. Optimizing support vector data description by automatically generating outliers. In *Proceedings of the EARSeL 7th SIG-Imaging Spectroscopy Workshop*.

Stephen Paul Marsh. 1994. *Formalising Trust as a Computational Concept*. Ph.D. dissertation. University of Stirling.

Stephen D. J. McArthur, Euan M. Davidson, Victoria M. Catterson, Aris L. Dimeas, Nikos D. Hatziargyriou, Ferdinanda Ponci, and Toshihisa Funabashi. 2007. Multi-agent systems for power engineering applications—Part I: Concepts, approaches, and technical challenges. *IEEE Transactions on Power Systems* 22, 4 (2007), 1743–1752.

D. Harrison McKnight, Larry L. Cummings, and Norman L. Chervany. 1998. Initial trust formation in new organizational relationships. *The Academy of Management Review* 23, 3 (1998), 473–490.

Thamar Mora, Abu B. Sesay, Jörg Denzinger, Hossein Golshan, Gene Poissant, and Cameron Konecnik. 2008. Cooperative search for optimizing pipeline operations. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems: Industrial Track (AAMAS'08)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 115–122. http://dl.acm.org/citation.cfm?id=1402795.1402817.

Lik Mui, Mojdeh Mohtashemi, and Ari Halberstadt. 2002. A computational model of trust and reputation. In *Proceedings of the 35th Hawaii International Conference on System Sciences*. 188–196.

Ming-Qing Pan, Su-Xiang Qian, Liang-Yu Lei, and Xiao-Jun Zhou. 2005. Support vector data description with model selection for condition monitoring. In *Proceedings of the 2005 International Conference on Machine Learning and Cybernetics*, Vol. 7. 4315–4318. DOI:http://dx.doi.org/10.1109/ICMLC.2005.1527696

Jeremy Pitt, Julia Schaumeier, and Alexander Artikis. 2011. The axiomatisation of socio-economic principles for self-organising systems. In *Proceedings of the 2011 5th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO'11),* 138–147. DOI:http://dx.doi.org/10.1109/SASO.2011.25

Sarvapali D. Ramchurn, Trung Dong Huynh Huynh, and Nicholas R. Jennings. 2004a. Trust in multi-agent systems. *The Knowledge Engineering Review* 19, 01 (2004), 1–25.

Sarvapali D. Ramchurn, Nicholas R. Jennings, Carles Sierra, and Lluis Godo. 2004b. Devising a trust model for multi-agent interactions using confidence and reputation. *Applied Artificial Intelligence* 18, 9–10 (2004), 833–852.

Sarvapali D. Ramchurn, Perukrishnen Vytelingum, Alex Rogers, and Nicholas R. Jennings. 2012. Putting the "Smarts" into the smart grid: A grand challenge for artificial intelligence. *Communications of the ACM* 55, 4 (2012), 86–97.

Andrew P. Sage and Christopher D. Cuppan. 2001. On the systems engineering and management of systems of systems and federations of systems. *Information, Knowledge, Systems Management* 2, 4 (2001), 325–345.

Alexander Schiendorfer, Jan-Philipp Steghöfer, and Wolfgang Reif. 2014. Synthesis and abstraction of constraint models for hierarchical resource allocation problems. In *Proceedings of the 6th International Conference on Agents and Artificial Intelligence (ICAART'14), Volume 2*. SciTePress, 15–27.

Jan-Philipp Steghöfer, Gerrit Anders, Florian Siefert, and Wolfgang Reif. 2013a. A system of systems approach to the evolutionary transformation of power management systems. In *Proceedings of*

INFORMATIK 2013—Workshop on "Smart Grids", Lecture Notes in Informatics, Vol. P-220. Bonner Köllen Verlag.

Jan-Philipp Steghöfer, Pascal Behrmann, Gerrit Anders, Florian Siefert, and Wolfgang Reif. 2013b. Hi-SPADA: Self-organising hierarchies for large-scale multi-agent systems. In *Proceedings of the 9th International Conference on Autonomic and Autonomous Systems (ICAS'13)*. IARIA.

Jan-Philipp Steghöfer, Rolf Kiefhaber, Karin Leichtenstern, Yvonne Bernard, Lukas Klejnowski, Wolfgang Reif, Theo Ungerer, Elisabeth André, Jörg Hähner, and Christian Müller-Schloer. 2010. Trustworthy organic computing systems: Challenges and perspectives. In *Proceedings of the 7th International Conference on Autonomic and Trusted Computing (ATC'10)*, Bing Xie, Jürgen Branke, S. Sadjadi, Daqing Zhang, and Xingshe Zhou (Eds.). Lecture Notes in Computer Science, Vol. 6407. Springer, Berlin, 62–76.

David M. J. Tax and Robert P. W. Duin. 1999. Data domain description using support vectors. In *Proceedings of the European Symposium on Artificial Neural Networks*. 251–256.

David M. J. Tax and Robert P. W. Duin. 2004. Support vector data description. *Machine Learning* 54, 1 (2004), 45–66.

David M. J. Tax and Klaus-Robert Müller. 2004. A consistency-based model selection for one-class classification. In *Proceedings of the 17th International Conference on Pattern Recognition (ICPR'04)*. Vol. 3, 363–366. DOI:http://dx.doi.org/10.1109/ICPR.2004.1334542

Sven Tomforde, Joerg Haehner, Hella Seebach, Wolfgang Reif, Bernhard Sick, Arno Wacker, and Ingo Scholtes. 2014. Engineering and mastering interwoven systems. In *Proceedings of the 2nd International Workshop on "Self-optimisation in Organic and Autonomic Computing Systems" (SAOS'14) in conjunction with ARCS 2014*, Vol. 2.

Edward Tsang. 1993. *Foundations of Constraint Satisfaction*. Vol. 289. Academic Press, London.

Vladimir N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.

Asimina Vasalou and Jeremy Pitt. 2005. Reinventing forgiveness: A formal investigation of moral facilitation. In *Trust Management*, Peter Herrmann, Valérie Issarny, and Simon Shiu (Eds.). Lecture Notes in Computer Science, Vol. 3477. Springer, Berlin, 39–90.

Perukrishnen Vytelingum, Sarvapali D. Ramchurn, Thomas D. Voice, Alex Rogers, and Nicholas R. Jennings. 2010. Trading agents for the smart electricity grid. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, Vol. 1. International Foundation for Autonomous Agents and Multiagent Systems, 897–904.

Chi-Kai Wang, Yung Ting, Yi-Hung Liu, and G. Hariyanto. 2009. A novel approach to generate artificial outliers for support vector data description. In *Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE'09)*. 2202–2207.

Yonghong Wang and Munindar P. Singh. 2007. Formal trust model for multiagent systems. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*. Morgan Kaufmann, San Francisco, CA 1551–1556.

Horst F. Wedde. 2012. DEZENT—A cyber-physical approach for providing affordable regenerative electric energy in the near future. In *Proceedings of the 2012 38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA'12)*. 241–249. DOI:http://dx.doi.org/10.1109/SEAA.2012.73

David H. Wolpert. 1996. The lack of a priori distinctions between learning algorithms. *Neural Computing* 8, 7 (Oct. 1996), 1341–1390. DOI:http://dx.doi.org/10.1162/neco.1996.8.7.1341

Han Yu, Zhiqi Shen, C. Leung, Chunyan Miao, and V. R. Lesser. 2013. A survey of multi-agent trust management systems. *IEEE Access* 1 (2013), 35–50.

Ascensión Zafra-Cabeza, Miguel A. Ridao, Ignacio Alvarado, and Eduardo F. Camacho. 2008. Applying risk management to combined heat and power plants. *IEEE Transactions on Power Systems* 23, 3 (2008), 938–945.