Abstraction of heterogeneous supplier models in hierarchical resource allocation

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

Angaben zur Veröffentlichung / Publication details:

Schiendorfer, Alexander, Gerrit Anders, Jan-Philipp Steghöfer, and Wolfgang Reif. 2015. "Abstraction of heterogeneous supplier models in hierarchical resource allocation." In *Transactions on computational collective intelligence XX*, edited by Ngoc Thanh Nguyen, Ryszard Kowalczyk, Béatrice Duval, Jaap van den Herik, Stephane Loiseau, and Joaquim Filipe, 23–53. Cham: Springer. https://doi.org/10.1007/978-3-319-27543-7_2.

Abstraction of Heterogeneous Supplier Models in Hierarchical Resource Allocation

Alexander Schiendorfer[1(✉)], Gerrit Anders[1], Jan-Philipp Steghöfer[2], and Wolfgang Reif[1]

[1] Institute for Software and Systems Engineering, University of Augsburg, Augsburg, Germany
{schiendorfer,anders,reif}@isse.de
[2] University of Gothenburg, Gothenburg, Sweden
jan-philipp.steghofer@cse.gu.se

Abstract. Resource allocation problems such as finding a production schedule given a set of suppliers' capabilities are generally hard to solve due to their combinatorial nature, in particular beyond a certain problem size. Large-scale instances among them, however, are prominent in several applications relevant to smart grids including unit commitment and demand response. Decomposition constitutes a classical tool to deal with this increasing complexity. We present a hierarchical "regio-central" decomposition based on abstraction that is designed to change its structure at runtime. It requires two techniques: (1) synthesizing several models of suppliers into one optimization problem and (2) abstracting the direct composition of several suppliers to reduce the complexity of high-level optimization problems. The problems we consider involve limited maximal and, in particular, minimal capacities along with on/off constraints. We suggest a formalization termed *supply automata* to capture suppliers and present algorithms for synthesis and abstraction. Our evaluation reveals that the obtained solutions are comparable to central solutions in terms of cost efficiency (within 1 % of the optimum) but scale significantly better (between a third and a half of the runtime) in the case study of scheduling virtual power plants.

Keywords: Hierarchical resource allocation · Self-organization · Discrete optimization · Abstraction · Virtual power plants

1 Dynamic Resource or Task Allocation in a Hierarchical Setting

Resource or task allocation problems present themselves in a variety of domains addressed by multi-agent-systems [7], including distributed power management [11,36] or grid computing [1]. In this paper, we investigate resource allocation

This research is partly sponsored by the German Research Foundation (DFG) in the project "OC-Trust" (FOR 1085).

problems with a demand imposed by the environment that needs to be satisfied by a set of suppliers which are represented by software agents acting on their behalf. These problems also lie at the heart of the selection and payment function of several practical problems addressed by mechanism design including adaptations of the Vickrey-Clarke-Groves mechanism [11,44] where computationally efficient algorithms are required. These problems can, in general, be expressed as constraint satisfaction and optimization problems. Techniques from operations research and discrete optimization, such as constraint programming or mixed integer (linear) programming, have been proposed to find optimal solutions (see, e.g., [19,37]). This proves particularly useful when the characteristics of heterogeneous suppliers requiring different sets of variables and constraints need to be modeled.

With regard to power management systems, a specific resource allocation problem is to maintain the balance between production and consumption at all times to keep the mains power frequency in a small corridor to achieve stable power supply [18]. This is achieved by creating "schedules" for controllable power plants, i.e., instructions of how much power they have to produce at which point in time, based on the predicted demand and the predicted input of weather-dependent power plants at that time. A complicating factor is the suppliers' inability to switch production levels arbitrarily over time. In this context, three main challenges arise: First, the resource allocation problem involves a vast number of power plants and consumers but, at the same time, has to be solved in a timely fashion. Second, the balance has to be kept despite uncertain demand and output of weather-dependent power plants. Third, heterogeneity requires solutions that can deal with the power plants' individual characteristics in the form of technical limitations and preferences when creating schedules.

If the size of the system prohibits a centralized solution, either due to the communication overhead required in collecting all necessary information or due to the complexity of a centralized solution model, hierarchical decomposition offers a generic tool to deal with these issues (see, e.g., [1,6,14]). Here, the global problem is decomposed by forming a hierarchical structure of agent organizations [20,43]. To solve the global problem, each organization acts as an *intermediary* that has to recursively solve a sub-problem, which is achieved in a top-down fashion with regard to the hierarchy, as illustrated in Fig. 1. Given a hierarchy, the problem can be solved by means of an auction protocol [2] relying only on proposals submitted by subordinates. Alternatively, intermediaries can centralize information from a region of the system, i.e., the control models of their subordinates, and solve their sub-problem centrally. Hence, we term this approach *regio-central*. Independent sub-problems can be solved concurrently in both endeavors. Lacking global knowledge, overall proven optimality is generally not feasible but traded for tractable sub-problems and close-to-optimal solutions (as evidenced by our evaluation results in Sect. 5.1). To achieve this tractability, we propose algorithms to calculate *abstractions* [15]:

Abstraction: On higher levels, intermediaries decide on schedules that have to be fulfilled by their subordinate agents. To make optimal decisions, exact models

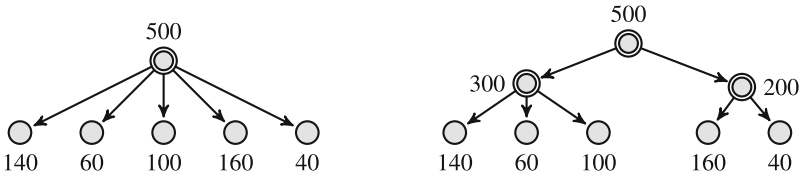

Fig. 1. A central and hierarchical solution example to a resource allocation problem. Inner nodes representing intermediaries are marked by double circles and redistribute their assigned share of an overall demand, e.g., power.

of all subordinates in composition would be required—eventually yielding the same complexity as a centralized solution. Because of the dynamic composition of organizations, abstraction has to be performed at runtime without manual interference.

In this paper, we focus on the regio-central approach to evaluate abstraction techniques in isolation, effectively testing the quality of the calculated models. However, abstracted models are also employed in the auction-based algorithm [3], making them a prerequisite for our hierarchical approach.

In the light of a dynamic environment and uncertainties, an organization might turn out to be unable to solve its corresponding sub-problem with sufficient quality or in due time, though. The organizations might thus have to or prefer to adapt their composition at runtime [3]. Furthermore, it is far from obvious how suitable structures have to be defined in terms of the size of organizations and the depth of the hierarchy in order to yield minimal runtimes (a challenge we discuss in our evaluation in Sect. 5.3) and solutions of acceptable quality at the same time. The size of the search space of possible hierarchies is $\Theta(B_n)$ where B_n is the n^{th} Bell number denoting the number of partitions since for each partition we can construct at least one corresponding hierarchy [4]. As a consequence, the system might thus depend on a number of successive reorganizations to establish a structure that provides an adequate runtime. In terms of optimization problems and models, this rises the challenge of separating concerns adequately, i.e., separating individual supplier characteristics from intermediaries' objectives, such that we can *synthesize* optimization problems at runtime:

Synthesis: To be able to solve the resource allocation problem at the different levels of the hierarchy considering the dynamic composition of organizations, *synthesis* defines necessary interfaces to convert a set of heterogeneous control models along with a predefined objective to obtain a regio-central optimization problem. The intermediary distributes the solution of this problem to its subordinate agents which, in turn, redistribute the solution if they are intermediaries themselves.

Throughout the paper, the creation of power plant schedules in autonomous power management systems serves as a case study to detail the life cycle and algorithms for this regio-central solution approach. We present a formalization of the resource allocation problem as a mixed integer program in Sect. 2 followed by considerations on how control models of suppliers are combined to one

such problem in Sect. 3. Three algorithms to obtain abstract control models of intermediaries are motivated and presented in Sect. 4. In Sect. 5 we discuss our experimental setup and present results to a number of evaluation questions. We present the related work in Sect. 6 and conclude with an outlook on achieved results and open questions in Sect. 7.

The paper is a substantially revised version of [41] and emphasizes the problem of cost-efficient scheduling rather than minimizing deviations between supply and demand only. We added the formal definition of supply automata to enable the systematic generation of synthesized optimization problems for a variety of generators. Also, the approach has been re-evaluated on larger experiments and the results are presented more rigorously. Moreover, the algorithms are discussed in more detail accompanied by additional examples and complexity considerations.

2 Problem Formalization

In the context of power management as a representative for the considered problem, the resource to be allocated is electric power and an allocation specifies how much (controllable) power plants need to supply at a certain point in time. The task is to assign schedules to controllable power plants so that their joint output meets the *residual load*, i.e., the difference between demand and supply from intermittent, weather-dependent sources. The overall problem is to continuously keep this balance at all times.

Formally, we abstract from this continuity by considering a finite set of discrete time steps $\mathcal{T} = \{1, \ldots, t_{\max}\}$ (e.g., in intervals of 15 min as used by energy markets) where we measure discrepancies and collect predictions of expected demand and intermittent production. Nonetheless, the problem cannot be solved optimally in advance for, e.g., the next day, due to the dynamic and stochastic nature mentioned in Sect. 1. An optimal schedule under the assumption of a sunny day might drastically underestimate the residual load and thus be detrimental to the system's objective. However, we need to consider at least a certain scheduling window $\mathcal{W} \subset \mathcal{T}$ ranging from the next time step $t_{\text{now}} + 1$ up to $t_{\text{now}} + W$ with $W = |\mathcal{W}|$ being the fixed width of that window. Inertia manifested in physical constraints such as limited rates of change or start-up times—both factors that present in slow-changing thermal power plants—necessitates this proactive behavior [18]. For instance, if a power plant's production is necessary to meet a high demand at time $t_{\text{now}} + W$, it may be required to start ramping up in $t_{\text{now}} + 1$.

In the energy management setting, suppliers are further subject to both limited maximal and *minimal* capacities of production due to mostly technical but also economical reasons (e.g., a minimum generation of 20 % of the nameplate capacity for gas turbines or 40 % for coal based thermal plants [21]). Moreover, for a virtual power plant, it is imperative to be able to switch plants on and off selectively to achieve more favorable aggregate partial load behavior compared to a single conventional generator [24] that suffers from increased costs when not operated at optimal energy conversion efficiency. Based on these assumptions, we present the core optimization problem we consider in Eq. (1).

$$
\begin{aligned}
\underset{S_t^a, \sigma_t^a}{\text{minimize}} \quad & \alpha_{\Delta} \cdot \Delta + \alpha_{\Gamma} \cdot \Gamma \qquad (1)\\
\text{subject to} \quad & \forall a \in \text{sub}(v), \forall t \in \mathcal{W}: \\
& \exists [x, y] \in L^a : x \leq S_t^a \leq y, \\
& S_{\delta}^{a-}\left(\sigma_{t-1}^a, S_{t-1}^a\right) \leq S_t^a \leq S_{\delta}^{a+}\left(\sigma_{t-1}^a, S_{t-1}^a\right) \\
& \text{with } \Delta = \sum_{t \in \mathcal{W}} \left| \sum_{a \in \text{sub}(i)} S_t^a - D_t \right|, \\
& \text{and } \Gamma = \sum_{t \in \mathcal{W}, a \in \text{sub}(i)} \kappa_a(S_t^a)
\end{aligned}
$$

The problem is formulated for one intermediary i that controls its set of subordinate agents $\text{sub}(i)$. The task primarily consists of assigning scheduled contributions.

S_t^a for each agent $a \in \text{sub}(i)$ and time step $t \in \mathcal{W}$. Additional *state* of agents, such as it being on or off, is taken into consideration with the auxiliary variables σ_t^a. Two possibly conflicting objectives to rank schedules are given by minimizing the discrepancy Δ between the aggregate supply and the demand D_t given by the residual load as well as minimizing the cost for the scheduled supply Γ which depends on cost functions κ_a mapping production levels to costs. The prioritization of both goals is regulated by weights α_{Δ} and α_{Γ}.

Minimal and maximal production capacities $S_{\min}^a$ and $S_{\max}^a$ are generalized to finite lists of non-overlapping *feasible regions* L^a, representing minimal and maximal capacities in a particular *mode*, e.g., on or off. A single generator that can be switched off[1] is then represented by $L^a = \langle [0,0], [S_{\min}^a, S_{\max}^a] \rangle$. Section 4 sheds light on why the generalization to lists is necessary to deal with abstracted models of intermediaries representing a whole set of suppliers. Intuitively, the composition of n suppliers having 2 modes each leads to 2^n combined modes showing individual minimal and maximal boundaries.

Inertia is reflected by functions S_{δ}^{a-} and S_{δ}^{a+} restricting possible decreasing or increasing supply given a state and current supply. Their role is explored more thoroughly and exemplified in Sects. 3 and 4.2. A state σ_t^a can be seen as a (heterogeneous) store of variables depending on the type of a. For instance, a supplier showing minimal runtimes before switching off could store the current runtime in time steps as $\sigma_t^a.cr$. With respect to a scheduling window $\mathcal{W} = \{t_{\text{now}}+1, \ldots, t_{\text{now}}+W\}$, $\sigma_{t_{\text{now}}}^a$ contains the actual momentary state as constants, whereas for $t \in \mathcal{W}$, σ_t^a are decision variables that have to be assigned consistently by the optimizer (e.g., when to issue a start/stop signal). If a supplier $a \in \text{sub}(i)$ is an intermediary itself, it solves the identical optimization problem for its own subordinates $\text{sub}(a)$ with its assigned share S_t^a being the demand D_t to be redistributed. Guaranteeing that S_t^a is in fact achievable by $\text{sub}(a)$ requires calculating the schedules based on the composition of all underlying agents, making the problem utterly complicated to solve. To achieve a reduction in complexity, abstraction is employed instead (see Sect. 4).

[1] In the application domain, we also need to consider so-called must-run plants [48], leading to $L^a = \langle [P_{\min}^a, P_{\max}^a] \rangle$.

When allowing these optimization problems to be formulated dynamically at runtime, we need to separate concerns of individual suppliers and the optimization problem as a whole. A few terms help us to put these aspects in context. More specifically, they aid to decompose the challenge presented by the resource allocation problem into several subproblems:

Individual Agent Models (IAM) are variants of extended finite state machines ("supply automata", see Sect. 3) that describe the properties of *one* supplier, in particular possible supply trajectories over time as well as different modes.

Abstracted Agent Models (AAM) are approximations of the composition of a set of underlying agents and represent the joint behavior.

Synthesized Optimization Problems (SOP) combine *several* agent models with a predefined "template" objective to generate the optimization formulation presented in Eq. (1) at runtime. More specifically, the resulting SOP is a generated artifact in a suitable language, e.g., MiniZinc [32] or, in our case, the optimization programming language (OPL) [10].

3 Synthesis of Regio-Central Optimization Problems

The process of converting a set of control models into one optimization problem according to the template presented in Eq. (1) is termed "synthesis". We motivate the main ideas and techniques and refer to [42] for a reference implementation.

A driving factor is to model suppliers with heterogeneous physical requirements such that reconfiguration of the system's hierarchical structure at runtime is enabled. The suppliers we consider are in fact dynamical systems with a controller manipulating an underlying physical process to, e.g., ramp up, ramp down, or shut off the supplier. We search for a sufficiently detailed discrete-time model that we can employ in the scheduling problem to dismiss technically infeasible schedules yet abstract enough (e.g., not containing particular control signals) to solve the synthesized optimization problem efficiently.

First, we look at existing constraints in the power domain (see [5,34]) and discuss how to build ones appropriate for the problem we are concerned with. The types of constraints considered in the literature are:

Table 1. Cold and hot start-up times for different power plant types according to [23,30]. A cold start occurs if a plant is down for more than 48 h, a hot start if it is down for less than 8 h.

Plant type	Cold start-up (h)	Hot start-up (h)
Black coal	4 – 5	2
Brown coal	6 – 8	2 – 4
Gas turbine	0.5	0.25
Photothermal	4 – 5	2

Minimal up/down times: A supplier has to run (or be switched off) for a minimal number of steps before switching the mode from "on" to "off" or vice versa [34].

Ramp up/down rates: A fixed amount of ramp-up/ramp-down capacity between two consecutive time steps is assumed [45]; in thermal power plants, this is due to required heating and cooling.

Cold/warm start-up times: A thermal plant generally needs a minimum number of time steps to ramp up from 0 to its minimal production [5]. Depending on the supplier's type, this start-up duration depends on the down-time as "cold starts" differ from "warm starts" (see Table 1 for sample values).

Not all supplier types are subject to the *same* constraints. Consider, e.g., a small gas power plant where the ramp up/down rates are high enough to regulate from minimal to maximal production in one time step. Similarly, with respect to Table 1, the difference between a hot or cold start-up might be negligible (depending on the duration of one time step) for gas turbines but not for photothermal plants. Modeling this type of behavior requires storing the running or standing times as auxiliary variables and define their development over time by suitable relations.

In [5], the problem is solved by encoding the start-up using linear constraints and 0/1 decision variables directly in a MIP. We propose to take a short detour and first model a supplier as a variant off *extended finite state machines* (EFSM) that can be seen as a discrete-time variant of a rectangular hybrid automaton [17] and then *generate* decision variables and constraints in a suitable language, e.g., OPL for our synthesized CSOP. Different *modes* of operation correspond to, e.g., switching a plant on or off and private variables capture individual physical characteristics. If, for instance, starting-up takes several time steps, then turning a supplier on requires taking a transition from the off-mode to a dedicated start-up-mode (where several time steps have to be passed) before eventually arriving in the productive mode. Expressing the control models this way also paves the way for better modeling and debugging support by suitable tools such as simulations, reachability analysis ("Can a supplier reach $S_{\max}$ if started newly?"), or automated testing ("Is a given trajectory accepted by a supplier?") and offers a canonical transformation.

3.1 Supply Automata

A supply automaton[2] is a particular extended finite state machine that specifies how a supplier can change its output S as well as other *local* state variables over one time step. We use it to describe the set of feasible trajectories of a supplier.

[2] Supply is not limited to positive production, since negative production, e.g., consumption or storage of a resource, can be vital to meet the demand, reflecting the concept of a *prosumer*.

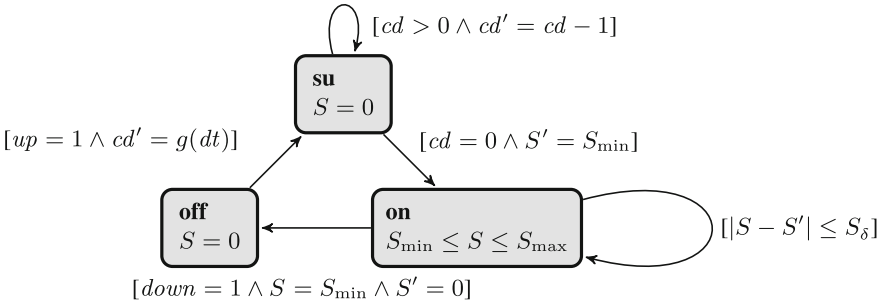

Fig. 2. Supply automaton to model adaptive start-up times depending on down time dt. A countdown cd is initialized with $g(dt) \in \mathbb{N}$, e.g., according to Table 1, and has the plant stay in the mode *su* (start-up) for $g(dt)$ steps. A plant can only contribute in the *on*-mode. Expressions in brackets contain jump predicates, invariants are written inside the modes.

Example 1. Figure 2 presents a supply automaton for a thermal unit considering start-up time. Consider a supplier that is currently off, needs constant $g(dt) = 1$ time steps to start up and then provide supply in $[5, 10]$ with a maximal change per time step $S_\delta = 1$. Then, exemplary trajectories $(0, 0, 5, 6, 7, 8, 9, 10)$ as well as $(0, 0, 0, 0)$ or $(0, 0, 5, 5, 0)$ are valid, in contrast to, e.g., $(0, 5, 5)$ which violates minimal start-up or $(0, 0, 5, 8)$ which violates maximal ramp-up.

We first explain the syntactic concepts used in Fig. 2 and provide semantics by defining valid transitions. We deliberately omit time indices in the presentation of supply automata as only transitions from one state to another are considered. The translation into an optimization problem similar to Eq. (1) including the underlying time series is presented in Sect. 3.2.

Definition 1. *A supply automaton SA is described by* $(M, X, T, \mathsf{Inv}, \mathsf{Jump})$ *where:*

- M *is a finite set of modes,*
- X *is a finite set of real-valued, local variables with* $S \notin X$*; we write* $X_S = \{S\} \cup X$ *to include* S*, the real-valued supply required for all suppliers.* X'_S *represents the same variables* after *a jump.* $\Phi(X_S)$ *denotes the set of predicates over free variables taken from* X_S*.*[3]
- $T \subseteq M \times M$ *indicate possible mode transitions*
- $\mathsf{Inv} : M \to \Phi(X_S)$ *returns the invariant in one mode including a feasible interval per mode "*$x \leq S \leq y$*"* $(x \leq y \in \mathbb{R}$
- $\mathsf{Jump} : T \to \Phi(X_S \cup X'_S)$ *return a predicate specifying the conditions to take transitions*

Note that we assume each state satisfying its mode's invariant to be a possible initial state. Let $[X_S \to \mathbb{R}]$ be the set of all assignments that map from the

[3] The expressiveness depends on the function and relation symbols of the underlying constraint language, we assume basic linear arithmetic, boolean algebra and standard inequalities.

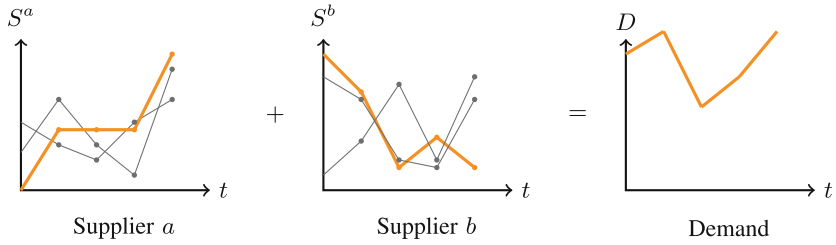

Fig. 3. Resource allocation over multiple time steps can be viewed as selecting the optimal combination of trajectories of two agents to meet a given demand.

variables X_S to $\mathbb{R}$. Let $p \in \Phi(X_S)$ be a set of predicates over X_S. We then write $v \models p$ to denote that an assignment $v \in [X_S \to \mathbb{R}]$ satisfies p.

Definition 2. *The semantics of a supply automaton* $SA = (M, X, T, \mathsf{Inv}, \mathsf{Jump})$ *are described by an associated transition system* $\mathsf{TS}(SA) = (\Sigma, \longrightarrow^*)$ *where:*

- *the (infinite) state space* $\Sigma \subseteq M \times [X_S \to \mathbb{R}]$ *is given by* $\Sigma = \{(m, v) \mid v \models \mathsf{Inv}(m)\}$.
- *the transition relation* $\longrightarrow \subseteq \Sigma \times \Sigma$ *is defined by* $(m, v) \longrightarrow (m', v')$ *if there exists a transition* $t = (m, m') \in T$ *such that* $v \cup v' \models \mathsf{Jump}(t)$.

We assume SA to be completed by τ*-transitions, i.e., take the reflexive closure* $\longrightarrow^*$ *to allow the supply automaton to preserve its state without explicit notations.*

A trajectory of a transition system $(\Sigma, \longrightarrow^*)$ is then a sequence $(\sigma_0, \sigma_1, \ldots, \sigma_k)$ such that $\sigma_i \in \Sigma$ for $i \in [0, \ldots, k]$ and $\sigma_i \longrightarrow \sigma_{i+1}$ for $i \in [0, \ldots, k-1]$. We write $\sigma \longrightarrow^{(k)} \sigma'$ to denote that σ' can be reached from σ in exactly k steps. With respect to Example 1, we have to reformulate the trajectories in terms of Σ to verify that $(0, 0, 5, 6)$ is indeed valid (if the supplier is already in the start-up phase). Consider the trajectory $((\mathrm{su}, cd = 1, S = 0) \longrightarrow (\mathrm{su}, cd = 0, S = 0) \longrightarrow (\mathrm{on}, cd = 0, S = 5) \longrightarrow (\mathrm{on}, cd = 0, S = 6))$. By contrast, $(\mathrm{su}, cd = 1, S = 0) \not\longrightarrow (\mathrm{on}, cd = 0, S = 5)$ since the jump condition $cd = 0$ is violated (see Fig. 2).

3.2 Translating Supply Automata into a Synthesized CSOP

In light of supply automata, the resource allocation problem presented in Eq. (1) corresponds to selecting optimal schedules from the set of feasible trajectories for each supplier in an agent organization such that the demand is met by the aggregate supply, as illustrated in Fig. 3. More specifically, we seek assignments of supply and states for future time steps that are consistent with invariants and jump conditions. This question naturally leads to the framework of constraint satisfaction and optimization problems (CSOP) treated in areas of discrete optimization including constraint programming and mathematical programming. We show how to systematically generate such a CSOP based on the syntactical representation of supply automata.

Quite generally, in constraint satisfaction problems (CSP) given by (X, D, C), the main task is to map a finite set of variables X to their associated domains $(D_x)_{x \in X}$ (i.e., construct assignments $v \in [X \to D]$) such that all constraints from a set C are satisfied.[4] An extension to optimization problems is achieved by the objective $f : [X \to D]$ that maps an assignment to a (possibly partially) ordered set $(F, \leq_F)$ and asking for a consistent assignment having an optimal value in F.

For an intermediary i with supply automata $(M^a, X^a, T^a, \mathsf{Inv}^a, \mathsf{Jump}^a)_{a \in \mathrm{sub}(i)}$, we can generate variables and constraints over a given time window $\mathcal{W}$ as follows (we write generated CSOP variables and constraints in typewriter font to avoid confusion with the variables in the supply automata):

Variables. We create the set of decision variables by "flattening" a particular trajectory over the window $\mathcal{W}$. In particular, we need variables $\mathtt{S[a][t]}$ to represent S^a_t for *every* agent $a \in \mathrm{sub}(i)$, a variable $\mathtt{x\_a[t]}$[5] for every *individually required* variable $x \in X^a$, and the mode m^a for every time step $t \in \mathcal{W} \cup \{t_{\mathrm{now}}\}$ written as $\mathtt{m[a][t]}$. The set corresponding to X^a_S for a particular time step t is written as $\mathtt{X^a_S[t]} = \{\mathtt{S[a][t]}\} \cup \{\mathtt{x\_a[t]} \mid x \in X^a\}$. For a predicate $\varphi \in \Phi(X^a_S)$, let $\varphi\langle X^a_S \to \mathtt{X^a_S[t]}\rangle$ denote the substitution of $x \in X^a_S$ with their counterparts in $\mathtt{X_S[a][t]}$, e.g., "$S^a \leq 50$" becomes "$\mathtt{S[a][t]} \leq 50$".

Initial States. The time step t_{now} is needed to consider the current state. For $S^a_{t_{\mathrm{now}}}$, we add the constraint "$\mathtt{S[a][t_{now}]} = S^a$", and similarly, we require "$\mathtt{x\_a[t_{now}]} = \sigma^a.x$" for $x \in X^a$ and "$\mathtt{m[a][t_{now}]} = m$".

State Invariants. For all time steps and all modes, the respective invariant has to hold, i.e., for every $t \in \mathcal{W}$ and $m \in M$, we add the constraint "$\mathtt{m[a][t]} = m \implies \mathsf{Inv}^a(m)\langle X^a_S \to \mathtt{X^a_S[t]}\rangle$"

Transition Constraints. For all transitions $(m, m') \in T$ and every time step $t \in \{t_{\mathrm{now}}\} \cup \mathcal{W} \setminus \{t_{\mathrm{now}} + W\}$, we add the constraint "$\mathtt{m[a][t]} = m \wedge \mathtt{m[a][t+1]} = m' \implies \mathsf{Jump}^a(m)\langle X^a_S \to \mathtt{X^a_S[t]}, X'^a_S \to \mathtt{X^a_S[t+1]}\rangle$".

The formulation of the CSOP is completed by adding the objective function f. Since the demand does not depend on the agents that should supply it and $\mathtt{S[a][t]}$ is included for every agent, we can use the objective presented in Eq. (1). Summing up, we write $\textsc{synthesize}((SA)_{a \in A}, f, (F, \leq_F))$ for the CSOP translated from a set of supply automata.

Example 2. Consider, for instance, two suppliers a and b: a is able to switch between on and off and contributes in the range $[5, 10]$ if on, whereas b only has one mode with its supply ranging in $[4, 8]$; both can maximally ramp up or down 1 unit per time step. Furthermore, upon being turned on, a has to provide its minimal production 5. Assume that both suppliers currently provide their respective minimum, 0 or 4. We consider scheduling $W = 4$ time steps and only care for minimizing discrepancies (i.e., $\alpha_\Gamma = 0$, $\alpha_\Delta = 1$). Constructing the supply automata, we arrive at the following optimization problem that is actually a mixed integer program (presented in pseudocode-OPL):

[4] We adopt a functional view, i.e., each constraint $c \in C$ maps an assignment $v \in [X \to D]$ to $\mathbb{B} = \{\mathrm{true}, \mathrm{false}\}$ and consequently, c is satisfied w.r.t. v iff $c(v) = \mathrm{true}$.

[5] Note that the distinction between $\mathtt{x_a}$ and $\mathtt{S[a]}$ emphasizes that $\mathtt{S}$ is defined for *every* agent and $\mathtt{x_a}$ is generated *specifically* for a.

$$
\begin{aligned}
&\texttt{range window} = \texttt{t}_{\text{now}} \,..\, \texttt{t}_{\text{now}} + 4 \qquad (2)\\
&\qquad\texttt{agents} = \{\texttt{a}, \texttt{b}\}\\
&\quad\texttt{minimize}\ \ \texttt{sum(t in window) abs}\,(\texttt{sum(a in agents) S[a][t]} - \texttt{D[t]})\\
&\texttt{subject to}\quad \texttt{forall(t in window)}\ \{\\
&\qquad\qquad \texttt{m[a][t]} = \texttt{off} \rightarrow \texttt{S[a][t]} = 0\\
&\qquad\qquad \texttt{m[a][t]} = \texttt{on} \rightarrow \texttt{S[a][t]} \geq 5 \ \texttt{and}\ \texttt{S[a][t]}\ \leq 10\\
&\qquad\qquad \texttt{m[b][t]} = \texttt{on} \rightarrow \texttt{S[b][t]} \geq 4 \ \texttt{and}\ \texttt{S[b][t]}\ \leq 8\\
&\qquad\qquad \texttt{m[a][t]} = \texttt{on or m[a][t]} = \texttt{off}\\
&\qquad\qquad \texttt{m[b][t]} = \texttt{on}\\
&\qquad\}\\
&\qquad \texttt{forall(t in } \texttt{t}_{\text{now}} \,..\, \texttt{t}_{\text{now}} + 3)\ \{\\
&\qquad\qquad \texttt{m[a][t]} = \texttt{off and m[a][t} + 1] = \texttt{on} \rightarrow \texttt{S[a][t} + 1] = 5\\
&\qquad\qquad \texttt{m[a][t]} = \texttt{on and m[a][t} + 1] = \texttt{on}\ \rightarrow\\
&\qquad\qquad\quad \texttt{abs(S[a][t]} - \texttt{S[a][t} + 1]) \leq 1\\
&\qquad\qquad \texttt{m[b][t]} = \texttt{on and m[b][t} + 1] = \texttt{on}\ \rightarrow\\
&\qquad\qquad\quad \texttt{abs(S[b][t]} - \texttt{S[b][t} + 1]) \leq 1\\
&\qquad\}\\
&\qquad \texttt{m[a]}[\texttt{t}_{\text{now}}] = \texttt{off},\ \texttt{m[b]}[\texttt{t}_{\text{now}}] = \texttt{on},\ \texttt{S[a]}[\texttt{t}_{\text{now}}] = 0,\ \texttt{S[b]}[\texttt{t}_{\text{now}}] = 4
\end{aligned}
$$

Once the CSOP is synthesized, we can include *soft constraints* [31] regarding individual preferable states and transitions (e.g., "try to schedule at 240 KW" or "never ramp up more than 25 % in 15 min, even if technically possible"). If not all soft constraints can be satisfied simultaneously, suppliers may establish *constraint relationships* [40] prioritizing these goals, i.e., "avoiding strong ramp-up" is more important than "stay at 240 KW". Soft constraints lead to a special kind of codomain for the objective function $(F, \leq_F)$, viz. elements of a c-semiring or valuation structures (partially ordered monoids). Intuitively, an assignment $v \in [X \rightarrow D]$ is graded by a set of soft constraints and the overall grading $f(v)$ is found using the multiplication operator of the c-semiring or valuation structure. The precise semantics of these statements along with their integration with Eq. (1) are outside the scope of this work but are described in detail in [25,38].

4 Abstraction of Composite Control Models

Except for the root of the hierarchy, intermediaries can be regarded as another type of supplier that is also controlled by a superior intermediary. Instead of merely using composed control models on higher hierarchy levels (which would effectively just result in a centralized solution), we introduce some reduction of complexity by an automated abstraction algorithm. Clearly, abstraction may cause errors due to imprecision but leads to a scalable resource allocation scheme as discussed in our evaluation in Sect. 5.1. The task at hand is consequently to

formalize the construction of an *abstracted agent model* (AAM), as introduced in Sect. 2, for an intermediary.

Since both AAM and *individual agent models* (IAM) are of the same type to have a superior agent only deal with agent models denoted by supply automata, an AAM also defines possible production ranges and transitions between productions for S_t^i. More abstractly, it describes the set of feasible trajectories of an intermediary's aggregate production. Technically, this set is represented by means of suitable constraints. We are interested in finding the "corners" of the possible supply space spanned by an intermediary as well as "holes", i.e., contributions that cannot be achieved by the set of subordinate agents due to discontinuities, e.g., resulting from discrete on/off transitions. We propose three abstraction algorithms that result in an abstract supply automaton representing the intermediary.

General abstraction (Sect. 4.1) calculates feasible production ranges corresponding to joint modes of the underlying suppliers by considering on/off settings and minimal/maximal supply. *Temporal abstraction* (Sect. 4.2) aims at dismissing provably infeasible schedules due to inertia (e.g., limited ramping rates and start-up times). *Sampling abstraction* (Sect. 4.3) probes a collective's functional relationships such as a mapping from supply to costs and uses a simpler representation on higher levels.

4.1 General Abstraction

The first important abstraction consists of describing the feasible regions of an intermediary based on its subordinate agents. As the contribution of an individual supplier may be discontinuous due to distinct operation modes (e.g., when a supplier shows a strictly positive minimal contribution when *on* but may also not contribute at all when *off*), the production space is discontinuous in general.

Example 3. Consider an intermediary i responsible for two suppliers a and b with their respective possible contributions given by $\{[0,0],[1,4]\}$ and $\{[0,0],[7,10]\}$ where $[0,0]$ indicates that the suppliers might be off. Then every production from $[1,4]$ can be reached by switching supplier a on and supplier b off. Dually, the intermediary can produce $[7,10]$ if only supplier b is running. Engaging both suppliers leads to a combined production interval $[8,14]$ and analogously $[0,0]$ with both being excluded entirely. However, no output in the intervals $(0,1)$ and $(4,7)$ can be provided. Abstraction enters the picture in the sense that it is not relevant whether, e.g., the output $S^i = 8$ is created by setting S^1 to 1 and S^2 to 7 or by S^1 to 0 and S^2 to 8. Multiple concrete configurations collapse to one abstract view, beneficially to the complexity of solving the resource allocation problem. We can thus contract the overlapping intervals $[7,10]$ and $[8,14]$ to yield $[7,14]$. To sum up, the feasible regions of that intermediary are given by $\{[0,0],[1,4],[7,14]\}$ with supply holes $(0,1)$ and $(4,7)$. Hence, 0 and 14 constitute the intermediary's *actual* minimal and maximal production under consideration of underlying on/off-modes.

From this example, we derive a formulation of general abstraction that returns the possible productions of an intermediary. Let $+$ be the standard plus operation in interval arithmetic such that $[x_1, y_1] + [x_2, y_2] = [x_1 + x_2, y_1 + y_2]$ which will be used to calculate the combined production of two suppliers each operating in one particular mode.

These possible contributions of a supplier a are given by a sorted list L^a of non-overlapping intervals. Since a supplier may contribute in *any* of the offered intervals due to varying modes, we need to match any two intervals for combination. For this purpose, we lift the combine operation $+$ to lists and write $\times$ for the combination of two lists. First, we consider the set of all combinations of intervals for two lists:

$$L^a \times L^b := \{[x, y] + [x', y'] \mid [x, y] \in L^a, [x', y'] \in L^b\} \tag{3}$$

For a set of n suppliers A each having k distinct intervals, the set $\prod_{a \in A} L^a$ contains $O(k^n)$ elements. This comes as no surprise since our pivotal application scenario with $L^a = \langle [0, 0], [S^a_{\min}, S^a_{\max}] \rangle$ enumerates the power set of *running* power plants. A brute-force implementation consequently suffers from this exponential behavior. Fortunately, considerable savings can be achieved by composing the overall result from smaller partial results and merging overlapping intervals. In addition, the problem is solvable in linear time if $k = 1$, i.e., if all suppliers have to be *on* and provide in one continuous interval. With respect to our case study, this situation corresponds to so-called "must-run" power plants [48]. Nonetheless, we consider the general case ($k \geq 1$).

For the aforementioned contraction, we recursively define a normalization operation $\downarrow$ which takes a sorted list of intervals (in ascending order of the lower bounds of the intervals) and merges overlapping intervals such that, e.g., $\langle [7, 10], [8, 14] \rangle \downarrow = \langle [7, 14] \rangle$. Concatenation of lists is written as $L_1 \cdot L_2$.

$$\begin{aligned} \langle\rangle\downarrow &= \langle\rangle \\ \langle [x, y] \rangle\downarrow &= \langle [x, y] \rangle \\ (\langle [x_1, y_1], [x_2, y_2] \rangle \cdot L)\downarrow &= \\ &\begin{cases} \langle [x_1, y_1] \rangle \cdot (\langle [x_2, y_2] \cdot L)\downarrow) & \text{if } y_1 < x_2 \\ (\langle [x_1, \max\{y_1, y_2\}] \rangle \cdot L)\downarrow & \text{else} \end{cases} \end{aligned} \tag{4}$$

Equipped with this operation, we can implement general abstraction by iteratively calculating the set of possible combinations as shown in Algorithm 1.

As a consequence, the list of feasible regions for an intermediary i is given by $L^i = \textsc{general-abstraction}((L^a)_{a \in \mathrm{sub}(i)})$, also written as $\bigotimes_{a \in \mathrm{sub}(i)} L^a$. Technically, the ability to calculate partial results and normalize them, originates in the associativity of $\times$ (inherited from $+$) and the fact that normalization obtained by $\downarrow$ is only a different representation for the same set of feasible contributions. Regarding supply automata, each interval $j = [x, y] \in L^i$ translates to one mode m_j with $\mathsf{Inv}(m_j) =$ "$x \leq S^i \leq y$". Transitions and jump conditions are added in Sect. 4.3.

Algorithm 1. General Abstraction to find feasible regions

Require: L^a is a finite list of feasible regions for a supplier $a \in A \neq \emptyset$, A is finite
Ensure: L is a list of feasible regions reachable by the suppliers in A

```
1: procedure GENERAL-ABSTRACTION((L^a)_{a∈A})
2:     Choose a ∈ A                                  ▷ arbitrary first supplier
3:     L ← L^a
4:     for all b ∈ A \ {a} do
5:         L' ← ⟨⟩
6:         for all [x, y] ∈ L × L^b do
7:             Insert-Sorted(L', [x, y])             ▷ sort by x ascendingly
8:         L ← L'↓                                   ▷ normalize
9:     return L
```

4.2 Temporal Abstraction

While general abstraction describes feasible regions of an intermediary, it fails to consider momentary *states*, such as the current productions, that further restrict inertia if some suppliers internally are at peak level or switched off. *Temporal* abstraction thus calculates infeasible ranges of an intermediary i for all time steps $t \in \mathcal{W}$ given the initial state $\sigma^i_{t_{\text{now}}}$ and supply $S^i_{t_{\text{now}}}$. For an intermediary, a possible production level at time step t is composed by the supply offered by its children. Starting from a given initial state, we determine temporally feasible ranges by *both* minimizing and maximizing every child's production respecting jump conditions until time step t and merging the resulting intervals using $\otimes$. Values below or beyond those ranges are guaranteed to be *infeasible* and must therefore not be allocated in the abstract view.

Example 4. Consider the same intermediary i presented in Example 3 responsible for two suppliers a and b specified by $L^a = \langle [0,0], [1,4] \rangle$ and $L^b = \langle [0,0], [7,10] \rangle$. Now assume that a is off at the moment and needs to be off for one more time step before contributing the minimum 1 and b currently producing 8. Both suppliers feature a maximal rate of change of 1 per time step. Provided that we intend to schedule 2 time steps, we analyze possible contributions after each future time step under the present constraints. At $t_{\text{now}}+1$, a may still only be considered with 0 as there is one step more to wait, whereas b's output can be either decreased to 7 or increased to 9 but not be switched off completely. Consequently, the intermediary can only contribute in the range of $[7,9]$ at that point. At $t_{\text{now}}+2$, a can contribute the minimum 1 (or be still off if we chose not to start it) and b can go up to 10 maximally or be already switched off to yield 0 since the minimum of 7 is reachable in the previous step $t_{\text{now}}+1$. Therefore, the intermediary can contribute in $\langle [0,0], [1,1] \rangle \otimes \langle [0,0], [7,10] \rangle = \langle [0,0], [1,1], [7,11] \rangle$. If we incorrectly assumed the full range of general abstraction to be available, wrong allocations, such as $S^i_{t_{\text{now}}+1} \leftarrow 2$ and $S^i_{t_{\text{now}}+2} \leftarrow 3$, would be possible in the abstract view of the intermediary i and need systematic treatment.

The intuition behind temporal abstraction directly stems from Example 4. For each future time step, we perform a *maximization step* as well as a

minimization step that provides us with the feasible regions of all subordinate agents, which are combined to find abstract boundaries. Technically, for a supplier a modeled by SA^a with its associated transition system $\mathsf{TTS}(SA^a) = (\Sigma, \longrightarrow)$, we look for $S_t^{a,\max} = \max\{S \mid \exists \sigma \in \Sigma : (m^a_{t_{\text{now}}}, \sigma^a_{t_{\text{now}}}) \longrightarrow^{(t-t_{\text{now}})} \sigma \wedge \sigma.S = S\}$ and dually for the minimization. For the constraints showed in Sect. 3, we can calculate these boundaries by ramping-up and down step-wise but in general this need not hold.[6]

Restricted transitions regarding possible supply are encoded in a supplier's jump conditions Jump^a. Given a state and supply, we assume that all conditions allow to derive functions that present minimal and maximal supply and states in the following time step and call them *inertia functions*. For each mode $m \in M^a$ and $c = \mathsf{Jump}^a(m)$, we require associated functions $c_{\min}$ and $c_{\max}$. We write $\sigma_t^{a,\max}$ and $S_t^{a,\max}$ to denote the states and supplies for supplier a at the maximization step t (dually for the minimization). To illustrate the concept of such inertia functions, consider a fixed rate of change condition $\mathsf{maxChange}$: $|S^a - S'^a| \leq S^a_\delta$. We can derive bounding functions that depend on a state as follows: $\mathsf{maxChange}_{\min}(\sigma^a, S^a) := (\sigma^a, S^a - S^a_\delta)$ and $\mathsf{maxChange}_{\max}(\sigma^a, S^a) := (\sigma^a, S^a + S^a_\delta)$. Similarly, in the case of a start-up condition su, $\mathsf{su}_{\max}(\sigma^a,)$ equals $(\sigma^a\langle m \to \text{on}\rangle, S^a_{\min})$ if $\sigma^a.cd = 0$ (the remaining "count down" in that state) and $(\sigma^a\langle cd \to cd - 1\rangle, 0)$ if $\sigma^a.cd > 0$. In addition, minimization and maximization steps ought to respect minimal and maximal productions of the respective mode.

Based on these inertia functions, we derive Algorithm 2 to exclude infeasible parts of the search space. For a future time step t, we identify the minimal and maximal contribution of each subordinate a returned by its inertia functions. To obtain the minimal and maximal output of the intermediary in this time step, we combine and merge the resulting intervals with $\otimes$. We write L_t^i to represent the feasible regions of the intermediary i for time step t which corresponds to the combined gray intervals in Fig. 4. These intervals *further* constrain feasible schedules in addition to the general bounds represented by the combined white intervals established by general abstraction.

Temporal abstraction thus adds constraints excluding infeasible regions for specific time steps in the synthesized optimization problem (on higher levels), thereby further reducing the abstracted search space. These constraints are as well only concerned with S_t^i and therefore seamlessly integrate with the AAM found by general abstraction.

$$\forall t \in \mathcal{W} : \exists [x, y] \in L_t^i : x \leq S_t^i \leq y \tag{5}$$

In addition to the general boundaries L^i returned by GENERAL-ABSTRACTION(i) (see Algorithm 1) that hold for all time steps, we constrain S_t^i (the supply of i in time step t) to lie in an interval specified by L_t^i. Section 5.4 investigates the effects of temporal abstraction.

[6] Consider a supplier with limited fuel resource in-flow such as, e.g., a biogas plant. At a future time step t, the supply could actually be higher if no gas had been spent in the previous steps rather than ramping-up upfront and needing to ramp-down at t due to the lack of fuel.

Algorithm 2. Temporal Abstraction to exclude infeasible ranges

Require: i is an intermediary; $\sigma_{t_{\text{now}}}$ contains state t_{now} for all suppliers $a \in \text{sub}(i)$
Ensure: $(L_t^i)_{t \in \mathcal{W}}$ consists of the feasible regions reachable by the intermediary at time step t

procedure TEMPORAL-ABSTRACTION(i, $\sigma_{t_{\text{now}}}$)
  intermediaries $\leftarrow \{a$ is an intermediary $\mid a \in \text{sub}(i)\}$
  for all $j \in$ intermediaries **do**
    $(L_t^j)_{t \in \mathcal{W}} \leftarrow$ TEMPORAL-ABSTRACTION$(j, \sigma^j_{t_{\text{now}}})$
  $\forall a \in \text{sub}(i) \setminus$ intermediaries : $\sigma^{a,\min}_{t_{\text{now}}}, \sigma^{a,\max}_{t_{\text{now}}} \leftarrow \sigma^a_{t_{\text{now}}}$
  for all $t \in \mathcal{W}$ **do**
    for all $a \in \text{sub}(i) \setminus$ intermediaries **do**
      $S_t^{a,\min}, \sigma_t^{a,\min} \leftarrow c_{\min}(\sigma_{t-1}^{a,\min}, S_{t-1}^{a,\min})$ ▷ $c = \mathsf{Jump}^a(\sigma_{t-1}^{a,\min}.m)$
      $S_t^{a,\max}, \sigma_t^{a,\max} \leftarrow c_{\max}(\sigma_{t-1}^{a,\max}, S_{t-1}^{a,\max})$ ▷ $c = \mathsf{Jump}^a(\sigma_{t-1}^{a,\max}.m)$
    $L_t^a \leftarrow \{[S_t^{a,\min}, S_t^{a,\max}]\}$
  $L_t^i \leftarrow \bigotimes_{a \in \text{sub}(i)} L_t^a$
  return $(L_t^i)_{t \in \mathcal{W}}$

4.3 Sampling Abstraction

In addition to finding the feasible regions of an intermediary, we are interested in functional relationships between aggregate variables, such as mapping the total production of all subordinate agents to total costs. Similarly, inertia functions for the intermediary depending on current productions ought to be found. Temporal abstraction only goes so far as to exclude *definitely infeasible* productions at time t. It does not restrict the transition between two independently feasible but not consecutively reachable productions for future time steps.

Example 5. In the scenario described in Example 4, we found that $L^i_{t_{\text{now}}+1} = \langle[7, 9]\rangle$ and $L^i_{t_{\text{now}}+2} = \langle[0, 0], [1, 1], [7, 11]\rangle$ are feasible. Assume a schedule leaving suppliers a and b at their current state during time step $t_{\text{now}} + 1$ (thus $S^i_{t_{\text{now}}+1} = 8 \in L^i_{t_{\text{now}}+1}$) but asking $S^i_{t_{\text{now}}+2} = 11 \in L^i_{t_{\text{now}}+2}$. Clearly, both scheduled productions are individually attainable with respect to temporal boundaries but ignore the fact that $S^i_{t_{\text{now}}+2} = 11$ is reachable *only if* $S^i_{t_{\text{now}}+1} = 9$. More precisely, b needs to already ramp up to 9 in step $t_{\text{now}} + 1$ to reach 10 in step $t_{\text{now}} + 2$ when a is then able to provide its minimum level 1, leading to a combined production of 11. With regard to Fig. 4, consider a schedule leaving the suppliers at their current output for the time steps $t_{\text{now}} + 1$ and $t_{\text{now}} + 2$ but demanding S^i_3 to be maximal. This schedule ignores that, starting from state $\sigma_{t_{\text{now}}}$, only regions in $L^i_{t_{\text{now}}+1} \subset L^i$, i.e., the gray intervals in time step $t_{\text{now}} + 1$ are possible.

Hence, inertia functions for the intermediary mapping a certain aggregate production to possible successor states are of interest. Similarly, an intermediary's cost function is needed to be able to make better allocations on higher levels. We propose to acquire an abstract representation of these functional relationships by *sampling*, i.e., solving several optimization problems that "probe" the collective behavior of an intermediary. Concretely, these sampling problems consist of the constraints given by the composition of agent models and introduce

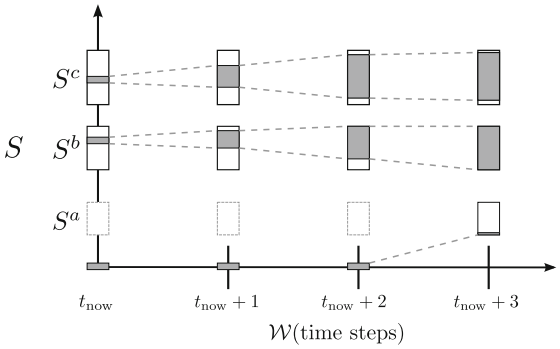

Fig. 4. Temporal abstraction for an intermediary consisting of three suppliers given their current states shown in time step t_{now}. White boxes indicate general bounds, gray areas represent the temporal boundaries at time step t. Supplier a needs two time steps to start up and is then available at its minimal output.

an additional constraint that binds the input variable to some particular value. Regarding inertia functions, as objective, the output variable is minimized or maximized.

For instance, consider an AVPP with 400 being included in its feasible regions obtained by general abstraction. Multiple configurations of its children can lead to these 400, resulting in varying possible increases in production. To find the maximal positive change, we enforce $400 = S^i_{t_{\text{now}}} = \sum_{a \in \text{sub}(i)} S^a_{t_{\text{now}}}$ and ask to maximize $S^i_{t_{\text{now}}+1}$. Assuming that the solution yields 450, we collect the pair $(400, 450)$ as sampled information. Similarly, we find the minimal costs by using the objective to minimize $\kappa(S^i_{t_{\text{now}}}) = \sum_{a \in \text{sub}(i)} \kappa(S^a_{t_{\text{now}}})$ given $S^i_{t_{\text{now}}} = 400$.

Performing this procedure not for one but a set of sampling points evenly distributed over the respective input range is intended to capture the intermediary's characteristics. The resulting input-output pairs can be represented by a suitable approximation method. We currently employ *piecewise linear functions* since they are readily supported by MIP or constraint solvers and have already been applied in model abstraction in simulation engineering [13]. Algorithm 3 illustrates this idea for finding maximal productions given the current state. Since absolute time indices are not needed for this calculations, we may fix S^i_0 and maximize S^i_1. Note that we formulate constraints and the objective syntactically using quotes rather than presenting them semantically in terms of, e.g., their extension.

As of now, the sampling points are selected equidistantly across the full range although more informative points can be selected systematically using techniques borrowed from active learning [39]. The resulting sampling optimization problems (see line 7 in Algorithm 3) are NP-hard in general. Therefore, a robust implementation ought to consider bounding the time spent on optimization by, e.g., setting a time limit. If then no solution is found for a particular input, we give up on it and deal with the next input. Guidance by properties of the

Algorithm 3. Sampling Abstraction for rates of change

Require: $(SA)_{a \in \mathrm{sub}(i)}$ is a family of supply automata
Ensure: S_δ^{i+} collects pairs of the positive change speed

procedure SAMPLING-ABSTRACTION(i, s)
  $(X, D, C, f) \leftarrow$ SYNTHESIZE$((SA)_{a \in \mathrm{sub}(i)}, f, (\mathbb{R}, \leq))$ ▷ see Sect. 3.2
  $I \leftarrow$ CHOOSE-SAMPLING-POINTS(s, L^i), $S_\delta^{i+} \leftarrow \emptyset$ ▷ select s feasible points
  for all $inp \in I$ **do**
    $C' \leftarrow C \cup \{\text{“}S_0^i = inp\text{”}\}$
    $f' \leftarrow$ “maximize S_1^i” ▷ new objective function
    $out \leftarrow$ solve (X, D, C', f')
    $S_\delta^{i+} \leftarrow S_\delta^{i+} \cup \{(inp, out)\}$
  return CONVERT-TO-PIECEWISE-LINEAR(S_δ^{i+})

function, such as monotonicity ($x \leq y \rightarrow f(x) \leq f(y)$) or extensivity ($x \leq f(x)$), can help to shrink the search space.

It has to be stated that this form of abstraction leads to over-approximations of the actually possible rates of changes and minimal costs. This is due to the fact that among all configurations yielding the input, we select an *extremal* one. For instance, for a certain level of production and possible ramp-up, we select a configuration that offers the most potential (all suppliers far enough below their maximum). In other cases showing the same aggregate supply, some subordinate agents might however be already peaking and cannot offer additional ramp-up. This leads to an over-approximation of the ramp-up. Similarly, when considering a schedule in the abstract view, we assume to achieve the optimal costs for each scheduled production individually (i.e., for each time step), whereas transitions among them might not be technically feasible in the concrete view. More sophisticated abstraction techniques to allow for more robust estimations (e.g., by considering minimal as well as maximal costs for a given supply) have yet to be investigated.

However, regarding our current setting, our evaluation in Sect. 5.2 shows promising results for using the proposed scheme for cost function approximation. Mostly, this is due to the fact that regionally optimal solutions at the lowest level of the hierarchy can improve upon possibly suboptimal decisions made on higher levels. As expected, Table 4 and Fig. 7 show that more sampling points lead to increased accuracy in abstraction observable by better overall costs at the cost of higher running times. An implementation exploiting this principle could start by collecting an initial small set of sampling points to offer a first crude approximation faster while collecting more sampling points in a background process. Finally, the sampled inertia function is used to define the set of transitions and suitable jump conditions for the abstracted supply automaton. Consider two modes m_j, m_l with $j = [x, y]$, $l = [x', y'] \in L^i$ obtained by general abstraction such that $y < x'$. We add transitions (m_j, m_l) if $S_\delta^{i+}(y) \geq x'$ and (m_l, m_j) if $S_\delta^{i-}(x') \leq y$. Furthermore, we add reflexive transitions (m_j, m_j). All of these transitions include the jump condition $S_\delta^{i-}(S^i) \leq S'^i \leq S_\delta^{i+}(S^i)$.

5 Evaluation

To give the presented algorithms empirical grounding, we evaluated the approach on problems taken from decentralized energy management using power plant models built from freely available data [12]. A centralized (optimal) solution planning the outputs of all power plants at once is compared to the regio-central approach using abstraction. These solutions are used for benchmark purposes offering insight in how close to the optimum the regio-central approach gets.

More specifically, we consider biofuel, hydro, and gas plants in the region of Swabia in Bavaria. Nameplate capacities, i.e., maximal productions are drawn from a distribution according to this data. Minimal productions depend on the type of plant and are given as percentage of the nameplate capacity. Similarly, maximal rates of change per minute and costs per kWh are selected based on the type and taken from [21,26], respectively. To sum up, our data is based on the quantities in Table 2.

This statistical data forms the basis of a generative model that we can use to sample realistic power plants for our simulation. First, we sample the type according to their relative frequency and then draw the maximal production depending on the selected type according to normal distributions ($\mathcal{N}(368.72, 1324.62)$ for biofuel, $\mathcal{N}(264.63, 746.55)$ for hydro, and $\mathcal{N}(275.88, 494.80)$ for gas) within the bounds listed in Table 2. Rates of change and costs are added based on type and maximal production. The demand, i.e., the residual load, to be fulfilled by a set of power plants is based on consumer data of [29] and scaled such that the peak loads map to 110 % of the drawn plants' combined maximal productions – in order to have a representative load test for the system. Regarding Eq. (1), we set α_Δ to a high value[7] to prioritize the goal of meeting the demand but still minimize costs among all demand-satisfying schedules.

Upon drawing a set of power plants, hierarchies are created similar to a B+ tree, i.e., only AVPPs at the lowest level control physical power plants. The hierarchies' shapes, i.e., depth and width, are controlled by restricting the maximal number of physical power plants per AVPP at the leaf level and the maximal number of directly subordinate AVPPs at the inner node levels. First, the "leaf" AVPPs

Table 2. Initially assumed distribution of input data including characteristics [21,26]. We list minimal and maximal bounds for the maximal production. Standard deviations for cost distribution are given in parentheses.

Type	Rel. Frequency	Max. Power	Min. Power	Rate of Change	Costs
	[%]	[KW]	[%]	[%/min]	[€ / KW]
Biofuel	54	[3.0, 17374.0]	35	6	0.175 (0.014)
Hydro	43	[2.0, 7800.0]	0	50	0.15 (0.017)
Gas	3	[1.0, 2070.0]	20	20	0.0865 (0.004)

[7] The value of α_Δ effectively acts as a "market price" since violations would have to be compensated by buying additional energy.

are created by taking a random permutation of the physical plants and picking clusters of the maximal physical plant count. Then, new hierarchy levels in the form of intermediate AVPPs, i.e., inner nodes, are introduced when needed.

We export the synthesized optimization problems (be it regio-central or central ones) as mixed integer programs that are solved by IBM ILOG CPLEX [10] which is indicative for the state-of-the-art in commercial energy management software [35,47]. Given the same power plants and initial states, the problem is solved, unless otherwise stated, for a period of half a day (i.e., 48 time steps in 15 min intervals, being a standard in energy markets) both centrally and regio-centrally — called a *run*. We obtain comparable results by taking care of random seeds and reproducibility. Consequently, all our experiments follow the basic structure:

1. Draw n power plants
2. Load consumer data for half a day (unless otherwise stated, $t = 48$ steps)
3. Solve the resource allocation problem in a hierarchical way
4. Solve the resource allocation problem in a central way
5. Repeat k times (k depends on the experiment)

We investigate parametrizations for the regio-central approach depending on the specific evaluation questions that can be seen as the independent variables in our experiment setup. That includes the number of sampling points to be used, the maximal number of concrete plants per AVPP as well as the maximal number of AVPPs per AVPP to control depth and width of hierarchy structures.

Several dependent variables are measured to compare the performance and are introduced when needed. The most prominent ones are clearly overall costs and runtimes per run as well as per time step which results in either $k \cdot t$ or k data points that are statistically analyzed.

The experiment suite and full source code including an instruction on how to run the experiments can be found online[8] in an attempt to provide replicable research. Each presented experiment was run on a machine having 4 Intel Xeon CPU 3.20 GHz cores and 14.7 GB RAM on a 64 bit Windows 7 OS with 8 GB RAM offered to the Java 7 JVM running the abstractions as well as CPLEX.

All central models used for comparison were solved with a 30 min time limit per time step. When planning for 15 min intervals, a solution must be available much earlier. We still wanted to collect optimal solutions as benchmarks and therefore allowed twice that period for the central solver. We examine questions of interest and present the results of the experiment runs in the following sections.

5.1 Scalability

Does the size of the problem impact the performance in terms of time and quality? We expect that upon reaching a certain number of power plants, the runtime per time step scales better in the hierarchical setting than in a central benchmark. We vary the number of power plants considered and compare the achieved

[8] https://github.com/Alexander-Schiendorfer/TCCI-Abstraction.

runtime and costs. Given n power plants, we construct a hierarchy by grouping 35 physical plants at the leaf level as suggested by Sect. 5.3. Inner AVPPs may control up to 10 other AVPPs. By inspecting the runtime behavior of a centralized solution from 5 to 100 plants (see Fig. 6), we found that, although the median runtimes grow linearly with the number of scheduled suppliers, the spread increases strongly and, in particular, outliers showing high runtimes are more probable. We drew 50 times a set of power plants of the respective size and simulated 48 time steps, i.e., half a day.

Table 3 presents the results per time steps aggregated over 2400 data points. For each input size, the difference between monetary costs per step as well as the difference between runtimes per time step were statistically significant using a paired t-test at $\alpha = 0.01$. Figure 5 visualizes this effect, indicating that the central runtimes strictly grow faster than the hierarchical ones. However, the overhead costs incurred by using the hierarchical scheme are in the order of magnitude of 1 % while speed-ups of up to 50 % compared to the central solution could be achieved. Note that we compare the runtime of the centralized approach to the aggregated sequential runtime of all intermediaries in the regio-central approach. Further runtime improvements can be achieved when parallelizing the schedule creation of independent intermediaries, i.e., those that belong to different subtrees in the hierarchy.

In addition to the speed-up per time step, one has to consider the runtime to be invested on the abstraction itself.[9] For a period of 48 time steps, the "fixed costs" (in terms of runtime) for abstraction consist of the computational effort for general and sampling abstraction. With only 50 plants, amortization of the hierarchy is not reached as the mean execution times for the central solution (57.49 s) are far below the hierarchical case (82.57 s). At 100 plants, the central times (158.87 s) begin to exceed the hierarchical case (154.91 s). This development culminates in a difference of 5232.75 s (central) vs. 3210.62 s (hierarchical) when

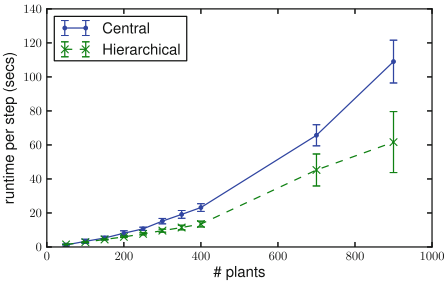

Fig. 5. Runtimes central vs. hierarchical per time step. $\frac{1}{6}$ of the standard deviation is presented for visualization.

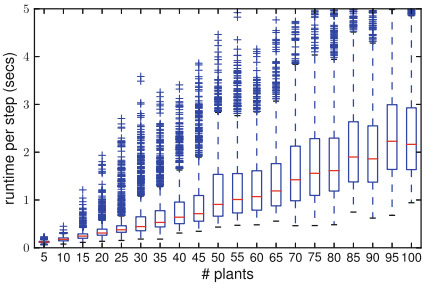

Fig. 6. Runtime development of *central* solutions. Only runtimes less than 5 s are plotted. With rising n, the probability of outliers increases drastically.

[9] Note that the *temporal* abstraction execution is already included in the runtime per time step for the hierarchical setting.

Table 3. Comparison of central and hierarchical approach over varying numbers of plants. Costs (Γ) and runtimes (T) are presented per time step averaged over 2400 data points from 50 drawn sets of plants each considering 48 time steps. Values in parentheses denote standard deviations.

# plants	T_{centr} (sec)	T_{hier} (sec)	rel.	Γ_{centr} (€)	Γ_{hier} (€)	rel.
50	1.2 (3.42)	1.49 (0.3)	124.18 %	2395.6 (536.14)	2398.21(536.16)	+ 0.11 %
100	3.31 (7.43)	2.82 (0.57)	85.19 %	4777.65 (845.56)	4782.79(845.81)	+ 0.11 %
150	5.27 (6.19)	4.43 (1.32)	83.94 %	7008.41 (1170.96)	7017.81(1169.98)	+ 0.14 %
200	8.09 (8.54)	5.92 (2.45)	73.23 %	9431.81 (1322.07)	9442.39(1321.87)	+ 0.11 %
250	10.67 (5.77)	7.71 (3.62)	72.31 %	11805.68 (1686.42)	11819.61(1686.49)	+ 0.12 %
300	15.2 (9.58)	9.6 (5.37)	63.14 %	14120.96 (1818.34)	14138.42(1818.0)	+ 0.13 %
350	19.15 (13.7)	11.44 (8.16)	59.71 %	16343.07 (2139.34)	16362.32(2139.0)	+ 0.12 %
400	23.2 (13.46)	13.49 (10.16)	58.14 %	18695.82 (2388.73)	18721.15(2390.16)	+ 0.14 %
700	65.68 (37.26)	45.28 (56.43)	68.93 %	32576.8 (3930.08)	32910.47(3906.97)	+ 1.05 %
800	81.19 (45.26)	36.07 (65.84)	44.43 %	37376.75 (4435.96)	37664.91(4431.76)	+ 0.79 %
900	109.02 (75.49)	61.64 (107.57)	56.54 %	41936.44 (4890.82)	42244.31(4835.42)	+ 0.76 %

considering 900 plants. The average relative share of abstraction runtime (excluding temporal abstraction time that belongs to every time step) with respect to the full 48 time step simulation decreases from 13.5 % (50 plants) to 7.8 % (900 plants). Concluding, we see a significant speed-up but, perhaps more importantly, that the abstracted models obtained at reasonable time investments are accurate enough to provide solutions close to the optima.

5.2 Sampling Accuracy

How does the extra running time spent on collecting sampling points pay off in terms of accuracy? Since we expect the abstracted models to increase in accuracy by using more sampling points, considerable improvements should be obtained at the cost of getting these sampling points (involving more sampling optimization problems) upfront. For varying numbers of power plants (50, 200, and 500) and varying numbers of sampling points (5, 10, 15, 25, and 35), we drew 50 times a set of plants from our generative model and calculated 5 time steps both regio-centrally and centrally (optimal) for comparison, totaling in a number of 250 data points. A maximal number of 20 power plants per AVPP and 5 AVPPs per AVPP were set to consider at least 3 AVPPs (in the case of 50 plants) up to 25 AVPPs (in the case of 500 plants), where we also need an additional intermediate layer to allow for several abstraction steps. Table 4 summarizes the results that are visualized in Figs. 7 and 8.

As expected, the runtimes rise with the number of sampling points due to the number of optimization problems that have to be solved in abstraction. Since 250 identical steps (50 draws with 5 time steps each) were solved by the algorithm with parameters, we tested our hypothesis of varying costs per time step using a pairwise t-test at $\alpha = 0.01$. Significantly different costs per step were shown for all combinations of sampling points other than 25 vs. 35 sampling points with 50 plants, 5 vs. 15 and 25 vs. 35 sampling points with 200 plants, and 5 vs. 10

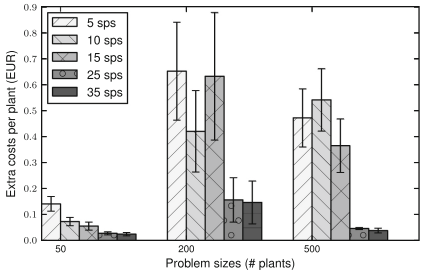

Fig. 7. The influence of sampling points on the accuracy of abstracted models. Extra costs are normalized w.r.t. the number of considered plants. We plot $\frac{1}{4}$ of the std.dev. for visualization purposes.

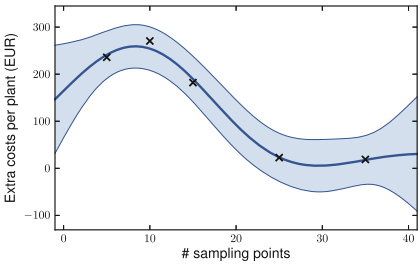

Fig. 8. The influence of the number of sampling points on the overhead costs in the case of 500 plants taken from Table 4. A Gaussian Process regression model is fit to the data to visualize the nonlinear functional relationship.

sampling points with 500 plants. In general, one can see a tendency that very high accuracy is achievable by selecting many points. However, offering more sampling points alone does not guarantee actual improvement due to the fact that the points are just selected equidistantly over the full range and therefore, *informative* points may appear randomly in sets consisting of fewer points but not in the larger one. We observe this behavior, e.g., when comparing 5, 10 and 15 sampling points for 200 plants. By chance, 10 sampling points lead to a significant improvement over 5, whereas 15 points compare unfavorably to 10 and yield accuracy similar to just 5 points. This confirms our findings in [39], emphasizing that more systematic selection is needed. Figure 8 provides more insight into the effect of increasing the number of sampling points on the achieved accuracy. This relationship is clearly nonlinear and one can observe a classical "diminishing returns": adding points to an already saturated set of observed sampling points does provide less benefit than to a comparably smaller set.

5.3 Hierarchy Influence

How does the hierarchy depth and breadth affect the quality and runtime? For this experiment, we varied the maximal number of physical plants per AVPP (ppAVPP) as well as the maximal number of AVPPs per AVPP (avppAVPP) resulting in different system structures for 600, 700, and 800 plants in total. Both parameters lead to broad structures if they are large and deep structures if they are small, thus provoking more abstraction steps.

Table 5 lists the results for different numbers of physical plants as well as ppAVPP/avppAVPP-combinations. Using a paired t-test at $\alpha = 0.01$, we compared all ppAVPP/avppAVPP-combinations in terms of runtime and costs, showing significant differences in most cases. Setting ppAVPP to a small value (5 or 15) leads to fragmented structures where accuracy and efficiency can be wasted as the centralized solver still has capacities for higher numbers of plants, confirming

Table 4. Comparison of different sampling point settings. Costs are presented per time step averaged over 250 data points from 50 drawn sets of plants each considering 5 time steps. Overhead shows relative and absolute extra costs compared to the optimal solutions. Runtime consists of the time needed to schedule 5 steps and overhead for sampling abstraction. Values in parentheses denote standard deviations. Optimal costs are written in bold.

# sampling points	costs per step	rel. overhead	abs. overhead	runtime
50 power plants:	**2180.71** € (453.91)			
5 sps	2187.73 € (453.57)	+ 0.32 % (0.29)	+ 7.03 € (2.37)	27.29 s (2.82)
10 sps	2184.31 € (453.87)	+ 0.17 % (0.16)	+ 3.6 € (1.79)	41.42 s (4.16)
15 sps	2183.45 € (453.72)	+ 0.13 % (0.15)	+ 2.75 € (1.77)	52.81 s (2.52)
25 sps	2182.05 € (454.01)	+ 0.06 % (0.06)	+ 1.34 € (1.1)	78.11 s (2.92)
35 sps	2181.9 € (453.86)	+ 0.05 % (0.07)	+ 1.19 € (1.15)	106.86 s (3.67)
200 power plants:	**8587.1** € (938.36)			
5 sps	8717.41 € (1014.05)	+ 1.52 % (1.53)	+ 130.31 € (12.28)	107.05 s (3.61)
10 sps	8671.07 € (975.07)	+ 0.98 % (1.36)	+ 83.97 € (11.2)	172.89 s (33.44)
15 sps	8713.46 € (995.13)	+ 1.47 % (2.22)	+ 126.36 € (14.02)	225.95 s (43.85)
25 sps	8618.24 € (938.57)	+ 0.36 % (0.83)	+ 31.14 € (8.27)	329.84 s (20.76)
35 sps	8616.31 € (952.83)	+ 0.34 % (0.7)	+ 29.21 € (8.14)	460.12 s (51.72)
500 power plants:	**21157.72** € (1853.51)			
5 sps	21393.53 € (1885.9)	+ 1.11 % (1.08)	+ 235.81 € (14.97)	244.0 s (11.28)
10 sps	21428.24 € (1868.18)	+ 1.28 % (1.16)	+ 270.52 € (15.51)	357.59 s (10.71)
15 sps	21339.92 € (1870.11)	+ 0.86 % (0.99)	+ 182.2 € (14.37)	470.32 s (18.24)
35 sps	21176.55 € (1853.2)	+ 0.09 % (0.09)	+ 18.83 € (4.28)	934.63 s (26.98)

our findings in Fig. 6. The largest value for ppAVPP, 35, led to the best runtime and cost efficiency in all considered cases which is why it was used as setting for Sect. 5.1. Interestingly, the tradeoff between cost and runtime performance seems to be regulated by the depth of the hierarchy; deep structures (avppAVPP = 5) led to the best runtime performance at slightly higher costs due to more abstractions. In the case of 600 plants, however, the differences in cost with 35 plants per AVPP over all AVPP per AVPP settings was not significant. Similarly, the difference in cost between 35/5 and 35/15 was not significant for 700 plants. With 900 plants, however, the flat structure with 15 AVPPs per AVPP showed significantly different costs compared to 5 and 10 AVPPs per AVPP. Concluding, our results suggest that AVPPs at the leaf level need to schedule an adequately number of power plants, whereas the speed-up benefits of having a deep structure controlled by small inner nodes seem to outweigh the (in most cases insignificant) additional costs. A more thorough analysis including probabilistic models of the system structures and more ppAVPP/avppAVPP-combinations is planned.

5.4 Temporal Abstraction for Inertia

Is temporal abstraction required for accurate abstracted models? We want to investigate the effects of temporal abstraction on the achieved accuracy. Therefore, we analyzed a small setting consisting of 50 power plants that are either

Table 5. Influence of the hierarchy on costs and runtimes per time step. Costs are presented per time step averaged over 250 data points from 50 drawn sets of plants each considering 5 time steps. Left column contains runtimes in seconds; right column contains costs in Euros; values in parentheses denote standard deviations. Best configurations per number of plants are written in bold.

plants per AVPP	5 AVPPs per AVPP		10 AVPPs per AVPP		15 AVPPs per AVPP	
600 plants:						
5 plants per AVPP	34.27	25895.19	43.92	25840.34	34.43	25842.65
	(1.63)	(2110.12)	(2.16)	(2107.79)	(1.4)	(2106.21)
15 plants per AVPP	33.66	25780.93	31.07	25765.1	31.64	25784.81
	(1.42)	(2137.22)	(1.39)	(2120.32)	(1.28)	(2122.12)
35 plants per AVPP	**25.1**	25729.94	27.96	25712.88	28.1	**25712.09**
	(2.29)	(2080.1)	(9.93)	(2108.26)	(6.89)	(2106.86)
700 plants:						
5 plants per AVPP	39.65	30109.09	50.27	30048.11	40.02	30045.2
	(1.61)	(2444.44)	(1.67)	(2422.91)	(1.56)	(2418.81)
15 plants per AVPP	39.51	29992.62	35.91	29966.94	36.51	29972.93
	(3.9)	(2449.15)	(1.57)	(2447.9)	(1.4)	(2429.5)
35 plants per AVPP	**28.77**	29897.34	30.18	29934.82	38.46	**29896.27**
	(1.73)	(2431.65)	(4.19)	(2445.54)	(18.2)	(2430.87)
800 plants:						
5 plants per AVPP	56.47	34542.19	59.43	34456.06	45.8	34458.11
	(1.98)	(2747.09)	(2.64)	(2737.62)	(1.85)	(2737.22)
15 plants per AVPP	41.93	34502.89	41.36	34310.24	42.25	34377.36
	(2.19)	(2738.86)	(1.49)	(2742.44)	(1.62)	(2750.6)
35 plants per AVPP	**32.81**	34377.9	34.52	34383.45	41.5	**34289.73**
	(2.18)	(2722.67)	(2.28)	(2711.98)	(14.74)	(2722.37)

grouped in 10 AVPPs of 5 power plants or in 5 AVPPs of 10 power plants that are managed by the root AVPP. Both cases show a hierarchy of depth 2 and therefore need one step of abstraction. Furthermore, the length of one time step has been reduced from 15 min to 1 min to consider increased effects of inertia. Errors are defined as differences between schedules assigned to intermediaries and their possible redistribution to their own subordinates.

As Table 6 shows, the error between assigned schedules to an intermediary and actually achievable schedules, i.e., the difference between its own subordinates' overall supply and assigned share of the residual load, can be reduced by employing temporal abstraction. This difference in errors is significant using a pairwise t-test at $\alpha = 0.01$ for both 5 as well as 10 plants per AVPP. Also, the error roughly doubles with 5 plants since more abstractions are performed and more possible deviations are introduced. Interestingly, the tighter rates of change induced by the shorter scheduling time steps makes the problem

Table 6. Influence of temporal abstraction. Costs are presented per time step averaged over 2400 data points from 50 drawn sets of plants each considering 48 time steps. Runtime is measured per time step (excluding fixed times for general and sampling abstraction); error denotes the overall discrepancy between supply and demand. Values in parentheses denote standard deviations.

plants per AVPP	T_{centr}	T_{hier}	Rel.	Error
without temporal abstraction:				
5 plants	12.25 (127.69)	4.36 (1.92)	35.55 %	3.55 (55.06)
10 plants	10.67 (116.23)	4.1 (0.56)	38.37 %	1.37 (33.9)
with temporal abstraction:				
5 plants	14.33 (141.99)	3.89 (0.58)	27.15 %	**0.26** (6.25)
10 plants	15.01 (143.04)	4.41 (2.49)	29.38 %	**0.01** (0.09)

significantly harder for CPLEX. This setting closely resembles the one used in [41], explaining the proportionally higher speed-up of about 70 % compared to Sect. 5.1 with about 50 %. Therefore, the proposed abstraction techniques can prove particularly useful when short intervals are considered to react timely on updated prognoses, slow power plants have to be scheduled, or schedules *should* not utilize the full technical potential of a power plant's rate of change, i.e., are restricted by soft constraints.

6 Related Work

Our proposed approach draws inspiration from many well-established concepts from diverse areas of computer science, including abstract interpretation, model abstraction, decomposition, and finite state machines.

The methodology of using abstracted models that are then refined on lower levels is closely related to abstract interpretation [9] used in program verification or approximation techniques used in the analysis of hybrid systems [17]. The latter technique analyzes minimal and maximal derivatives $\dot{x}$ of a state variable x (e.g., supply) w.r.t. time given a model in the form of differential equations and produces so-called rectangular hybrid automata that have constant upper and lower bounds. Hence, in one mode, x can perform continuous time transitions according to $\dot{x}_{\min} \leq \dot{x} \leq \dot{x}_{\max}$, similar to our considerations in Sects. 4.2 and 4.3. Merging output variables shared by a composition of agents (the aggregate supply) as we do in general abstraction is not considered. However, approximations obtained by their technique could serve as supply automata.

Abstractions have been treated from many different angles and in a variety of fields. First attempts toward automatic "model abstraction" to make pre-defined abstractions of complex physical system models obsolete were made in [46]. From an AI perspective, [15] was an attempt to systematically describe different varieties of abstraction and their formal properties, especially with regard to deduction and representation. A similar systematization was done in [13] where a

taxonomy of common model abstraction techniques from the perspective of the simulation engineering community is provided. Crucially, [13] states that abstraction techniques "must maintain the validity of the simulation results with respect to the question being addressed [...]" which holds true in the abstraction of composite control models as well. Another approach is to differentiate structural and behavioral abstraction [28]. The authors demonstrate abstraction methods and, e.g., use neural networks to get a behavioral abstraction of subcomponents that were given as state machines.

Abstraction methods for task and resource allocation problems are presented in [8]. One of the techniques, summarization, has similarities to our general abstraction since it combines time intervals in much the same way that we combine production intervals (cf. Sect. 4.1). The other technique, generalization, uses a taxonomy of the domain concepts used in the constraints to find more general formulations, e.g., to express that a certain operating room belongs to a certain unit within a hospital. Instead of assigning a surgeons to concrete operating rooms, they are assigned to units, making the problem smaller. We do not employ a similar technique. The paper argues for the inclusion of user decisions in the abstraction process to ensure that abstraction errors are avoided.

An approach to extract constraint models from an architectural description of the system for use in runtime optimization is presented in [16]. A model of the instantiation of an abstract architecture is used to derive constraints that describe the correct configurations of a system. The utility function takes into account the utility of the configurations as well as the utility and cost of the reconfiguration and the decision making itself. The resulting problem can either be solved with a MIP approach or by pseudo-boolean optimization (PBO) in order to find the best configuration of the system given the circumstances. Interestingly, the evaluation results suggest that the PBO method that intuitively should yield more accurate results failed to deliver and an approach based on MIP is sufficient. Similar to our work, the optimization problems are constructed at runtime, but no abstraction is attempted. Since the systems regarded do not follow a hierarchical structure, the problem solved has a very different structure.

The problem we address has similarities with lot size problems, especially those with minimal order quantities [33]. These problems address the optimal size of orders in cases where the machines producing them have a setup cost as well as a unit production cost and there is the possibility to store an inventory of items for later delivery. Minimal order quantities give a lower bound of the production per time step that is similar to the one for power generators: either a machine is switched off completely or it produces at least a minimal number of units. Formulations with a capacity constraint also limit the maximum inventory [27]. Such a limit can be used to model the maximum capacity of storage power plants in a power grid. However, the formulations from operations research do not address two important issues in energy management: change rates and loading times. The proposed models assume that it is possible to switch from maximum production (usually bounded by the cumulative demand) to zero production within one time step. Likewise, the inventory can take up as many units as

produced within a single time step. Power generators, however, have ramp up times that limit the change in their production from one time step to another. Power storage facilities, on the other hand, have loading rates that limit the energy that can be stored per time step. These issues are reflected in the supply automata we present in Sect. 3.

7 Conclusions and Future Work

In this paper, we addressed a resource allocation problem that is very relevant to energy management systems. The variant we consider shows *minimal* in addition to maximal supply capacities, discontinuity by on/off switches, and inert suppliers. We have motivated the need for self-organizing hierarchical structures due to scalability concerns and the large search space of "good hierarchies" in terms of runtime and monetary efficiency. To allow for self-organization, we need to separate individual properties, such as the start-up behavior of suppliers, from overall goals: meeting the demand cost-efficiently. In the process, we suggested supply automata to model individual, heterogeneous supplies as suitable formalism for common problems in the power systems literature that can be automatically translated into synthesized optimization problems and provided an example.

Based on supply automata, we discussed techniques to calculate abstractions of the composite models of a sub-system, i.e., intermediaries. Our evaluation shows that the hierarchical setting achieves a quality within 1 % of the optimal solutions at about 30 % to 50 % of the runtime by mere decomposition, i.e., without a parallel execution of the individual sub-systems. The runtimes of the hierarchical approach for given input sizes showed more favorable growth than a centralized benchmark using CPLEX. While the presented abstraction methods have been successfully applied to a market-based scheduling approach [2], we hope that the presented algorithms and models also generalize well to other problem domains.

The paper focused on the modeling and optimization aspects of the overall hierarchical, multi-agent approach sketched in Sect. 1. Uncertainties arising from deviations of predictions are dealt with by means of robust optimization methods using trust-based scenarios as described in [3]. Cooperative, i.e., truthful behavior of the involved agents is incentivized using techniques inspired by mechanism design [2]. It remains to be analyzed if a truthful mechanism for revealing the supply automata (the "types" in the language of algorithmic mechanism design) is achievable and/or desirable.

Our experiments also revealed further directions of research: given that the quality of abstraction depends strongly on the number and location of sampling points (see Sect. 5.2), we plan to investigate better choices for a guided selection strategy for sampling points using active learning or response surfaces. First tests show significant potential for improvement [39]. Furthermore, Sect. 5.3 highlighted the influence of the hierarchy on solution quality and runtime. Current techniques [43] rely on local rules triggering reconfigurations. Assuming patterns in the relationship of, e.g., input size, type composition and runtime,

we hope to learn more about the hardness of subproblems given certain features using techniques from empirical algorithmics such as model-based algorithm configuration [22].

References

1. Abouelela, M., El-Darieby, M.: Multidomain hierarchical resource allocation for grid applications. J. Electr. Comput. Eng. **2012**, 8 (2012)
2. Anders, G., Schiendorfer, A., Siefert, F., Steghöfer, J.P., Reif, W.: Cooperative resource allocation in open systems of systems. ACM Trans. Auton. Adapt. Syst. **10**, 11 (2015)
3. Anders, G., Schiendorfer, A., Steghöfer, J.P., Reif, W.: Robust Scheduling in a Self-Organizing Hierarchy of Autonomous Virtual Power Plants. In: Stechele, W., Wild, T. (eds.) Proceedings of the 2nd International Workshop Self-optimisation in Organic and Autonomic Computing Systems (SAOS 2014), pp. 1–8 (2014)
4. Anders, G., Siefert, F., Reif, W.: A particle swarm optimizer for solving the set partitioning problem in the presence of partitioning constraints. In: Proceedings of the 7th International Conference on Agents and Artificial Intelligence (ICAART 2015), pp. 151–163. SciTePress (2015)
5. Arroyo, J.M., Conejo, A.J.: Modeling of start-up and shut-down power trajectories of thermal units. IEEE Trans. Power Syst. **19**(3), 1562–1568 (2004)
6. Boudjadar, A., David, A., Kim, J.H., Larsen, K.G., Mikučionis, M., Nyman, U., Skou, A.: Hierarchical scheduling framework based on compositional analysis using uppaal. In: Fiadeiro, J.L., Liu, Z., Xue, J. (eds.) FACS 2013. LNCS, vol. 8348, pp. 61–78. Springer, Heidelberg (2014)
7. Chevaleyre, Y., et al.: Issues in multiagent resource allocation. Informatica **30**(1), 3–31 (2006)
8. Choueiry, B.Y., Faltings, B., Noubir, G.: Abstraction methods for resource allocation. Technical report, Swiss Federal Institute of Technology in Lausanne (EPFL) (1994)
9. Cousot, P.: Abstract interpretation. ACM Comput. Surv. **28**(2), 324–328 (1996)
10. CPLEX: IBM ILOG CPLEX Optimizer, Dec 2013. online Resource. http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/. Accessed December 2013
11. Dash, R.K., Vytelingum, P., Rogers, A., David, E., Jennings, N.R.: Market-based task allocation mechanisms for limited-capacity suppliers. IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum. **37**(3), 391–405 (2007)
12. Deutsche Gesellschaft für Sonnenenergie e.V.: Energymap, Dec 2013. http://www.energymap.info/. Accessed December 2013
13. Frantz, F.: A taxonomy of model abstraction techniques. In: Proceedings of the Winter Simulation Conference 1995, pp. 1413–1420 (1995)
14. Frey, S., Diaconescu, A., Menga, D., Demeure, I.: A generic holonic control architecture for heterogeneous multi-scale and multi-objective smart micro-grids. ACM Trans. Auton. Adapt. Syst (2015)
15. Giunchiglia, F., Walsh, T.: A theory of abstraction. Artif. Intell. **57**(2), 323–389 (1992)
16. Götz, S., Wilke, C., Richly, S., Piechnick, C., Püschel, G., Assmann, U.: Model-driven self-optimization using integer linear programming and pseudo-boolean optimization. In: International Conference on Adaptive and Self-Adaptive Systems and Applications (ADAPTIVE 2013), pp. 55–64 (2013)

17. Henzinger, T.A., Ho, P.H., Wong-Toi, H.: Algorithmic analysis of nonlinear hybrid systems. IEEE Trans. Autom. Control **43**(4), 540–554 (1998)
18. Heuck, K., Dettmann, K.D., Schulz, D.: Elektrische Energieversorgung. Vieweg+Teubner (2010). (in German)
19. Hladik, P.E., Cambazard, H., Déplanche, A.M., Jussien, N.: Solving a real-time allocation problem with constraint programming. J. Syst. Softw. **81**(1), 132–149 (2008)
20. Horling, B., Lesser, V.: A survey of multi-agent organizational paradigms. Knowl. Eng. Rev. **19**(4), 281–316 (2004)
21. Hundt, M., Barth, R., Sun, N., Wissel, S., Voß, A.: Verträglichkeit von erneuerbaren Energien und Kernenergie im Erzeugungsportfolio. Technisch-ökonomische Aspekte. Studie des Instituts für Energiewirtschaft und rationelle Energieanwendung (IER) im Auftrag der E. ON Energie AG. Stuttgart (2009). (in German)
22. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Coello, C.A.C. (ed.) LION 2011. LNCS, vol. 6683, pp. 507–523. Springer, Heidelberg (2011)
23. Jarass, L., Obermair, G.M.: Welchen Netzumbau erfordert die Energiewende?: Unter Berücksichtigung des Netzentwicklungsplans Strom 2012. MV-Wissenschaft, Monsenstein und Vannerdat (2012). (in German)
24. Karl, J.: Dezentrale Energiesysteme: Neue Technologien im liberalisierten Energiemarkt. Oldenbourg (2012). (in German)
25. Knapp, A., Schiendorfer, A., Reif, W.: Quality over quantity in soft constraints. In: Proceedings of the 26th International Conference on Tools with Artificial Intelligence (ICTAI 2014), pp. 453–460 (2014)
26. Kost, C., Mayer, J.N., Thomsen, J., Hartmann, N., Senkpiel, C., Philipps, S., Nold, S., Lude, S., Saad, N., Schlegl, T.: Levelized cost of electricity- renewable energy technologies. Techno-Economic Assessment of Energy Technologies, Fraunhofer ISE (2013)
27. Lee, C.Y.: Inventory replenishment model: lot sizing versus just-in-time delivery. Oper. Res. Lett. **32**(6), 581–590 (2004)
28. Lee, K., Fishwick, P.A.: A methodology for dynamic model abstraction. SCS Tran. Simul. **13**(4), 217–229 (1996)
29. LEW Verteilnetz GmbH: LEW Netzdaten, December 2013. http://www.lew-verteilnetz.de/. Accessed December 2013
30. Mayer, J.N., Kreifels, N., Burger, B.: Kohleverstromung zu Zeiten niedriger Börsenstrompreise, August 2013. http://www.ise.fraunhofer.de/de/downloads/pdf-files/aktuelles/kohleverstromung-zu-zeiten-niedriger-boersenstrompreise.pdf/view
31. Meseguer, P., Rossi, F., Schiex, T.: Soft constraints. In: Rossi, F., van Beek, P., Walsh, T. (eds.) Handbook of Constraint Programming, Chap. 9. Elsevier (2006)
32. Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G.R.: MiniZinc: towards a standard CP modelling language. In: Bessière, C. (ed.) CP 2007. LNCS, vol. 4741, pp. 529–543. Springer, Heidelberg (2007)
33. Okhrin, I., Richter, K.: The linear dynamic lot size problem with minimum order quantity. Int. J. Prod. Econ. **133**(2), 688–693 (2011). towards High Performance Manufacturing
34. Padhy, N.P.: Unit commitment- a bibliographical survey. IEEE Trans. Power Syst. **19**(2), 1196–1205 (2004)
35. PLEXOS Integrated Energy Model, January 2014. http://energyexemplar.com/software/plexos-desktop-edition/. Accessed 09 January 2014

36. Ramchurn, S.D., Vytelingum, P., Rogers, A., Jennings, N.R.: Putting the 'Smarts' into the smart grid: a grand challenge for artificial intelligence. Commun. ACM **55**(4), 86–97 (2012)
37. Santos, C., Zhu, X., Crowder, H.: A mathematical optimization approach for resource allocation in large scale data centers. Technical report, HPL-2002-64, HP Labs, March 2002
38. Schiendorfer, A., Knapp, A., Steghöfer, J.-P., Anders, G., Siefert, F., Reif, W.: Partial valuation structures for qualitative soft constraints. In: Nicola, R., Hennicker, R. (eds.) Wirsing Festschrift. LNCS, vol. 8950, pp. 115–133. Springer, Heidelberg (2015)
39. Schiendorfer, A., Lassner, C., Anders, G., Lienhart, R., Reif, W.: Active learning for abstract models of collectives. In: Proceedings of the 3rd International Workshop Self-optimisation in Organic and Autonomic Computing Systems (SAOS15), March 2015
40. Schiendorfer, A., Steghöfer, J.P., Knapp, A., Nafz, F., Reif, W.: Constraint relationships for soft constraints. In: Bramer, M., Petridis, M. (eds.) Proceedings of the 33rd SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence (AI 2013), pp. 241–255. Springer (2013)
41. Schiendorfer, A., Steghöfer, J.P., Reif, W.: 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), vol. 2, pp. 15–27. SciTePress (2014)
42. Schiendorfer, A., Steghöfer, J.P., Reif, W.: Synthesised constraint models for distributed energy management. In: Proceedings of the 3rd International Workshop Smart Energy Networks & Multi-Agent Systems (SEN-MAS 2014), pp. 1529–1538 (2014)
43. Steghöfer, J.P., Behrmann, P., Anders, G., Siefert, F., Reif, W.: HiSPADA: Self-organising hierarchies for large-scale multi-agent systems. In: Proceedings of the IARIA International Conference on Autonomic and Autonomous Systems (ICAS) 2013, IARIA (2013)
44. Ströhle, P., Gerding, E.H., de Weerdt, M.M., Stein, S., Robu, V.: Online mechanism design for scheduling non-preemptive jobs under uncertain supply and demand. In: Proceedings of the International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2014), pp. 437–444. International Foundation for Autonomous Agents and Multiagent Systems, Richland (2014)
45. Wang, C., Shahidehpour, S.: Effects of ramp-rate limits on unit commitment and economic dispatch. IEEE Trans. Power Syst. **8**(3), 1341–1350 (1993)
46. Weld, D.S., Addanki, S.: Task-driven model abstraction. In: 4th International Workshop on Qualitative Physics, pp. 16–30 (1990)
47. Werner, T.: DEMS - The Decentralized Energy Management System (2013). http://w3.siemens.com/smartgrid/global/en/smart-grid-world/experts-talk/pages/dems.aspx. Accessed 07 January 2014
48. Yingvivatanapong, C., Lee, W.J., Liu, E.: Multi-area power generation dispatch in competitive markets. IEEE Trans. Power Syst. **23**(1), 196–203 (2008)