Proceedings of the 2014 Federated Conference on Computer Science and Information Systems pp. 1529–1538
DOI: 10.15439/2014F49
ACSIS, Vol. 2

Synthesised Constraint Models for Distributed Energy Management

Alexander Schiendorfer, Jan-Philipp Steghöfer, Wolfgang Reif
Institute for Software & Systems Engineering, Augsburg University, Germany
Email: {alexander.schiendorfer, steghoefer, reif}@informatik.uni-augsburg.de

***Abstract*—Resource allocation is a task frequently encountered in energy management systems such as the coordination of power generators in a virtual power plant (unit commitment). Standard solutions require fixed parametrised optimisation models that the participants have to stick to without leaving room for tailored behaviour or individual preferences. We present a modelling methodology that allows organisations to specify optimisation goals independently of concrete participants and participants to craft more detailed models and state individual preferences. While considerable efforts have been spent on devising efficient control algorithms and detailed physical models in power management systems, practical aspects of unifying several heterogeneous models for optimisation have been widely ignored – a gap we aim to close. As a by-product, we give a formulation of warm and cold start-up times for power plants that improves existing power plant models. The concepts are detailed with the load-distribution problem faced in virtual power plants and evaluated on several random instances where we observe that a significant number of soft constraints of individual actors can be satisfied if considered.**

I. Constraint Optimisation Problems in Power Systems

RESOURCE allocation and scheduling are difficult problems that occur frequently in energy systems, be it the coordination of power generation [1], demand-side management, or building control software. In a producer-based view, supply needs to meet the demand as accurately as possible in order to guarantee stability and avoid costs incurred by corrective measures. Similarly, consumers may try to find cost-minimising schedules for processes required throughout a day with respect to time-dependent energy prices. Current initiatives[1] are based on the assumption that *groups* of prosumers (i.e., energy producers and/or consumers) can form and team up to achieve better prices or production rates for their participants. We also adopt the notion of *agents*, indicating that the prosumers are in principle autonomous entities, even if they surrender the decision about their power output to the group.

A straightforward solution (see, e.g., [2], [3], [4], [5]) to this resource allocation problem is to model the decision making process (e.g., distributing the load in a virtual power plant (VPP) or scheduling energy-consuming domestic processes in a consumer coalition) as a mathematical optimisation problem such as a mixed integer program (MIP), a linear program (LP) or as a constraint satisfaction and optimisation problem (CSOP) as done by industrial distributed energy management tools such as Siemens DEMS [6] or PLEXOS Integrated Energy Model [7]. DEMS is used, e.g., by the municipal utility of the city of Munich for controlling a VPP [8]. In essence, the problem is specified in terms of (decision) variables, their associated domains, and constraints that regulate which assignments are valid. The task accomplished by the respective solvers is then to assign values to all variables such that no constraint is violated and an optimisation objective is minimised (or maximised).

Typically, such tools (DEMS in particular) offer a predefined range of agent types such as energy generators, storages, or controllable loads. Users may then specify the topology of their energy system to calculate optimized power schedules. A concrete power generator is thus essentially represented by *one tuple* in a data repository containing the parameters defining its behaviour. Consequently, the provided models constitute a static *one-for-all* solution that needs to encompass all supported characteristics of power generators, including, e.g., time-dependent properties such as inertia.

Clearly, power generators show varying characteristics such as change rates, cool or warm start-up times or power boundaries depending on, e.g., the power plant type or manufacturer. *Parametrised models* as described above cannot support this variety. At some point the model has to be fixed for all participants and individual variables necessary to model a certain constraint cannot be added. To overcome this limitation, we suggest to synthesise an optimisation problem from several *individual models*. Such *synthesised models* allow for individual preferences (typically in the form of knowledge acquired by power plant operators such as economically optimal production ranges or limited ramp-up or -down of a generator) and separate modelling of the organisational optimisation problem and physical models of individual participants – properties that are attractive for organisations as more clients can be served as well as for individual participants as they can influence the assigned plans. This methodology is not only nice to have in multi-agent systems, where optimisation problems result from a combination of several sub-problems – it is *necessary*.

Our contribution leads to a methodology that offers:

1) support for heterogeneous prosumers requiring specific sets of variables;
2) isolated modelling of physical components;
3) clean separation of the organisational aspects such as

[1] cf. https://www.energiekosten-stop.at/ for consumer alliances or http://www.swm.de/geschaeftskunden/effizienz-umwelt/virtuelles-kraftwerk.html for virtual power plants

978-83-60810-58-3/$25.00

objectives or fairness constraints from physical models;
4) incorporation of individual preferences into the optimisation routine of a coalition to increase and incentivise the participation.

We exemplify model synthesis with the problem of creating schedules in a virtual power plant and show how to integrate custom behaviour in the form of cold and warm start-up times that are specific to certain power plant types as well as individual economical preferences. While we demonstrate the modelling and synthesis approach for a single organisation, these concepts can be incorporated to solve hierarchical resource allocation problems as described in [9], which focused on the abstraction of constraint models.

The paper is structured as follows: Sect. II introduces a formalism to express preferences with the help of constraint relationships while Sect. III shows the general approach to power plant scheduling within a virtual power plant. Both approaches are used in Sect. IV to synthesise individual models within a group of power plants. Sect. V then instantiates the general framework for use with IBM ILOG CPLEX. Our experimental results and the findings drawn from them are discussed in Sect. VI. We conclude the paper with a discussion of future research directions.

II. Constraint Programming with Constraint Relationships

As we suggest a modelling methodology for constraint satisfaction and optimization problems (CSOPs, see, e.g., [10]) to solve resource allocation problems, we briefly revisit the core model elements as well as the definition of constraint relationships [11] that we use to denote individual preferences of single agents. A CSOP $\zeta = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}, f \rangle$ consists of a set of decision variables $\mathcal{X}$ that take values from the domain $\mathcal{D}$ where consistent assignments $\theta \in (\mathcal{X} \to \mathcal{D})$ are regulated by the set of constraints $\mathcal{C}$. We write $\theta \models c$ if the constraint c is satisfied by an assignment θ. The objective function $f : (\mathcal{X} \to \mathcal{D}) \to \mathbb{R}$ measures the quality of a solution and effectively imposes an ordering over the assignments where we seek the best one.

However, not all constraints need to be hard requirements and some may also be violated if no assignment simultaneously satisfies all constraints [12]. We call these constraints *soft* and denote them by $\mathcal{C}_s$ as opposed to hard constraints $\mathcal{C}_h$ with $\mathcal{C}_h \cup \mathcal{C}_s = \mathcal{C}$, $\mathcal{C}_h \cap \mathcal{C}_s = \emptyset$. A set of *constraint relationships* for the soft constraints C_s of a CSOP ζ is given by a binary asymmetric relation $\mathcal{R} \subseteq \mathcal{C}_s \times \mathcal{C}_s$ whose transitive closure $\mathcal{R}^+$ is a partial order relation. We write $c' \prec_{\mathcal{R}} c$ or $c \succ_{\mathcal{R}} c'$ iff $(c, c') \in \mathcal{R}$ to define c to be *more important* than c', analogously for $\mathcal{R}^+$. If $c' \prec_{\mathcal{R}} c$ we call c' a *direct predecessor*, if $c' \prec_{\mathcal{R}^+} c$ a *transitive predecessor* of c. Moreover, we refer to the *constraint relationship graph* as the directed graph spanned by $\langle \mathcal{C}_s, \mathcal{R} \rangle$. Figure 1 shows a toy example of a CSOP with constraint relationships.

The binary relation over soft constraints needs to be lifted to sets of soft constraints that are violated by an assignment. Such a *violation set* is denoted by capitalizing the letter used for the assignment; i.e., for some assignment $t \in (\mathcal{X} \to \mathcal{D})$ its violation set is $T = \{c \in C_s \mid t \not\models c\}$. We propose different dominance properties $p \in \{\text{SPD}, \text{TPD}\}$ where SPD (single-predecessor-dominance) indicates that one constraint may only dominate a single predecessor and TPD (transitive-predecessor-dominance) defines that a single constraint is more important than *all* of its predecessors (corresponding to the relative importance among constraints) – we refer to [11] for more details. We write $T \longrightarrow_R^p U$ to denote that the violation set T *worsens* to U with dominance level p and use the following rules:

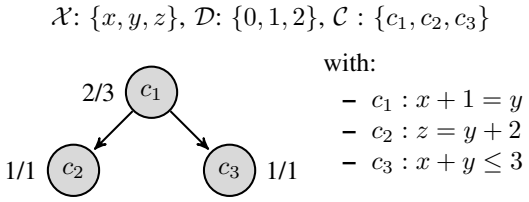

Fig. 1. Not all three constraints can be satisfied simultaneously, e.g. c_2 forces z to be 2 and y to be 0, conflicting with c_1, We can choose between solutions satisfying $\{c_1, c_3\}$ or $\{c_2, c_3\}$. Weights are given in the form "SPD / TPD".

$$T \longrightarrow_R^p T \uplus \{c\} \tag{W1}$$

$$\frac{T_1 \longrightarrow_R^p U_1 \quad T_2 \longrightarrow_R^p U_2}{T_1 \uplus T_2 \longrightarrow_R^p U_1 \uplus U_2} \tag{W2}$$

$$T \uplus \{c\} \longrightarrow_R^{\text{SPD}} T \uplus \{c'\} \quad \text{if } c \prec_R c' \tag{SPD}$$

$$T \uplus \{c_1, \ldots, c_k\} \longrightarrow_R^{\text{TPD}} T \uplus \{c'\} \quad \text{if } \forall c \in \{c_1, \ldots, c_k\} : c \prec_{R^+} c' \tag{TPD}$$

Finally, we define a partial order over solutions, denoted by $t >_R^p u$ and to be read as "t is better than u", using $T \ (\longrightarrow_R^p)^+ \ U$ (meaning repeated sequential application of the rules) for the selected $p \in \{\text{SPD}, \text{TPD}\}$. To use them in a CSOP, we calculate weights for the constraints that respect the partial order $>_R^p$ where we summarise the weights of violated constraints as the *penalty* of a solution:

$$w_R^{\text{SPD}}(c) = 1 + \max\{w_R^{\text{SPD}}(c') \mid c' \in \mathcal{C} : c \succ_R c'\}$$
$$w_R^{\text{TPD}}(c) = 1 + \sum_{c' \in \mathcal{C}_s : c \succ_R c'} (2 \cdot w_R^{\text{TPD}}(c') - 1)$$

Consequently, we define an objective based on $p : (\mathcal{X} \to \mathcal{D}) \to \mathbb{N}$ as:

$$\underset{\theta}{\text{minimize}} \ \ p(\theta) = \sum_{c \in \mathcal{C}_s, \theta \not\models c} w(c) \tag{1}$$

III. Scheduling Power Plants within a Virtual Power Plant

Our approach to synthesise individual models is exemplified with the problem of finding schedules for power plants in a virtual power plant that we described in [9] and [13]. This problem is also known as economic load dispatch (ELD) [14] or unit commitment (UC) [15]. In essence, the task is to

```
int lastSimStep = 10;
range TIMERANGE = 0..lastSimStep;
{string} PowerPlants = ...;

tuple PowerPlantData {
  float minimal; float maximal; float fixedRamp;
};

PowerPlantData plants[PowerPlants] = ...;
float demand[TIMERANGE] = ...;

dvar float+ production[PowerPlants][TIMERANGE];
dexpr float totalProduction[t in TIMERANGE] =
            (sum( p in PowerPlants ) (production[p][t]));

minimize
  sum( t in TIMERANGE )
    abs(demand[t] - totalProduction[t]);

subject to {
  forall (p in PowerPlants, t in 0 .. (lastSimStep - 1)) {
    production[p][t] >= plants[p].minimal;
    production[p][t] <= plants[p].maximal;
    abs(production[p][t] - production[p][t+1])
      <= plants[p].fixedRamp;
  }
}
```

Listing 1. A minimalistic, *parametrised* model of a load distribution problem

distribute a given load (for a certain time window) to a set of power generators in such a way that their capacities as well as inertia between consecutive time steps are respected. We present this basic scheduling problem as a CSOP (which we will refine to accommodate additional model aspects):

$$\begin{aligned} \underset{P_t^a}{\text{minimize}} \quad & \sum_{t\in\mathcal{W}} |P_t - \mathcal{D}_t|\,,\ P_t = \sum_{a\in\mathcal{A}} P_t^a \\ \text{subject to} \quad & \forall a \in \mathcal{A}, \forall t \in \mathcal{W} : P^a_{\min} \le P^a_t \le P^a_{\max}, \\ & v^a_{\min}\left(P^a_{t-1}\right) \le P^a_t \le v^a_{\max}\left(P^a_{t-1}\right) \end{aligned} \tag{2}$$

where P_t^a are the decision variables representing the production of plant $a \in \mathcal{A}$ at time step $t \in \mathcal{W}$, the scheduling window. The demand is given as the vector $\mathcal{D}$ and basic properties about the constraints of each power plant include minimal and maximal production values, $P_{\min}$ and $P_{\max}$ and functions denoting minimal and maximal production given the current output $v^a_{\min}, v^a_{\max} : \mathbb{R} \to \mathbb{R}$ to incorporate inertia and model the ramping behaviour of power plants [16]. To make this example more concrete, we assume that a power plant is described by its production boundaries as well as a fixed ramp up rate between two consecutive time steps (taking the role of $v_{\min}$ and $v_{\max}$) depending on the nameplate capacity as, e.g., given in [17]. We can then formulate the optimisation problem from Eq. 2 in IBM's optimisation programming language (OPL) as in Listing 1. OPL is used by IBM ILOG CPLEX [18] which in turn can be employed by both DEMS and PLEXOS.

The listing shows the shortcomings of a *parametrised* model: All power plants are described by the *same* set of parameters and constraints are defined uniformly for all power plants and time steps. Hence, the possible variety is severely limited.

TABLE I
DIFFERENT COLD AND HOT START-UP TIMES FOR POWER PLANT TYPES. A COLD START OCCURS IF A PLANT IS DOWN FOR MORE THAN 48H, A HOT START IF IT IS DOWN FOR LESS THAN 8H. TAKEN FROM: [17], [21]

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

In order to achieve feasible schedules, a number of different types in constraints are usually employed in power plant models. While the models presented within the scope of this work are far from complete and much more detailed models exist (see, e.g., [14], [15], [16]) these types are common and therefore representative. In addition, the more complex and heterogeneous models found in the literature only emphasise the need for a methodology to incorporate these diverse descriptions of physical and economic limitations with organisational aspects. The types of constraints considered here are:

Minimal up/down times: a generator is required to run (or be switched off) for a certain number of steps before being switched off (or turned on) [15].

Ramp up/down rates: usually, a fixed amount of production change between two consecutive time steps is assumed [19], especially in thermal power plants in which physical boundaries for heating and cooling the system have to be captured (as used in Listing 1). Alternatively, the possible change can be specified as a relative quantity denoting the percentage of the current output that a power plant can adapt.

Cold/warm start-up times: a power plant may need a certain number of time steps to ramp up from 0 to its minimal production level as modelled by [16]. However, for some plants, this start-up duration depends on the downtime as "cold starts" differ from "warm starts" (see Table I for sample values). We show how to formalise these start-up times in a MIP-framework using the transition system in Fig. 2. We consider the actual duration of a start-up as opposed to the costs which can be approximated with exponential functions [20].

Please bear in mind that *not all* power plants feature the *same* constraints. Consider, e.g., a power plant where the ramp up/down rates are high enough to regulate from minimal to maximal production in just one time step (say 15 minutes). Then, the model may not contain such a constraint [19]. If we want to consider this constraint in a parametrised model, *all* power plants would have to include model elements (e.g., ramp up rates). The problem becomes even more obvious when we consider minimal off times. As we demonstrate in Sect. IV-A, we need *additional* decision variables for the current off time

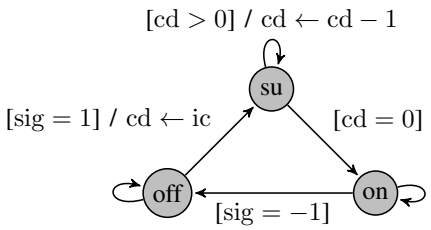

Fig. 2. Transition system to model adaptive start-up times depending on down time. The signal (sig) takes values from $\{-1, 0, 1\}$ where 1 shows a start-up, 0 is the default, and -1 triggers a shut-down. A countdown is initialised with ic as a result of a function of the down time and has the plant stay in the state *su* (start-up) for ic steps. A plant can only contribute in the state *on*. Expressions in brackets contain guards and other annotations at the transitions denote actions manipulating local variables.

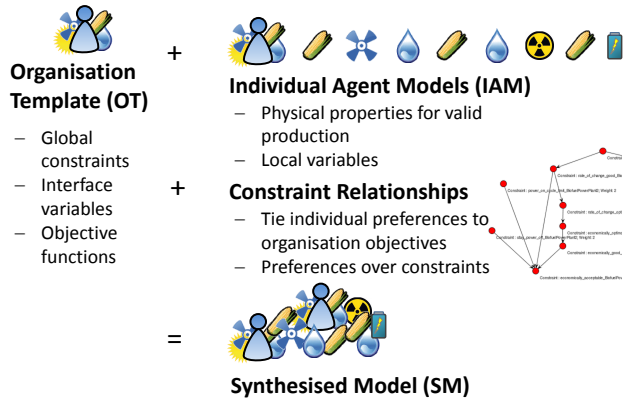

Fig. 3. Overview of the synthesis process

to adequately model these constraints. A parametrised model would again introduce these variables for *all* power plants including those that do not show minimal off times. For them, dummy values allowing to ignore certain constraints (e.g., a fixed ramp up rate of $P_{\max}$) would be required.

From an engineering perspective, it thus makes sense to model these aspects separately. Variants of these constraints can also be specified as preferences, if, e.g., a ramp up of 15 % is technically possible but it is more desirable to limit ramp up to 10 % to save expenses and material. We will use these exemplary constraints to create individual, heterogeneous power plant models to be used in synthesised models.

IV. Synthesis of Coalition Models

In light of the presented problems we can distinguish two aspects that are intermingled in Eq. 2:

Individual Agent Models (IAM) describe the properties of *one* agent representing a physical entity in terms of constraints for the available production. Constraints can be formulated depending on the internal state (being on/off, production levels etc.) independent of other agents. This model needs to be provided by the agent designer, e.g., the power plant manufacturer, possibly with customisations by its operator. An IAM defines the feasible production or consumption range of an agent but also regulates possible transitions between different time steps. Moreover, preferences (such as those avoiding high ramp-up rates) can be specified with constraint relationships to further constrain feasible schedules.

Organisational Templates (OT) represent the specific goals of an organisation or a coalition formed by the organisation. We consider organisations to be entities that exist independently of their specific agents and are therefore modelled separately [22]. The template captures the optimisation criterion and provides so-called *interface variables* that each IAM needs to incorporate. Additional (soft) constraints can impose policies of the organisation such as "prefer agents of type X" or "distribute resources in a fair manner".

Model synthesis is concerned with creating a CSOP from a set of agent models including their individual preferences and an organisational template as Fig. 3 shows. Aside from the generally required parameters of the individual agent models introduced in Listing 1 such as possible contributions, the power plant models exhibit varying characteristics regarding feasible schedules given by the constraints presented in Sect. III. Recall that to formalise these properties, we might need additional decision variables such as number of time steps a power plant has been switched off to capture the down time. As only *some* power plants may need these for modelling their start-up behaviour, we call them *local variables*. IAMs constitute a possible refinement of *option models* in the context of *Energy Agents*, a unifying framework for agent-based energy systems [23].

In our VPP example, the interface *decision* variables are the scheduled production values of all power plants over all time steps. Similarly, $P^a_{\min}$ and $P^a_{\max}$ need to be specified by each power plant a to calculate load percentages. Thus, the interface variables model the *homogeneous* parts of all considered agents as opposed to the *heterogeneity* introduced by custom behaviour in local variables: To accommodate minimal up/down times, we need variables $\top_t$ and $\bot_t$ that represent for how many time steps a power plant has been up or down, respectively, at time step t. For convenience, we further define $\nu_t \leftrightarrow P_t > 0$. A minimal up time of $\top^{\min}$ is expressed by the following constraint:

$$\forall t \in \mathcal{W} : \nu_t \wedge \neg\nu_{t+1} \rightarrow \top_t \geq \top^{\min} \tag{3}$$

For start-up times that depend on down times we need a countdown variable that regulates the time steps the power plant has to be in the start-up state shown in Fig. 2. We will further discuss how to formulate this constraint but first describe the general steps required for model synthesis.

A. Components of Model Synthesis

We first formally illustrate the ingredient models of our synthesis approach and give an implemented example using OPL in Sect. V. For doing so, we assume an organisation λ that controls a set of agents, $\mathcal{A}^\lambda$, along with their IAMs that consist of the interface variables $\mathcal{X}^\lambda$ and local variables

$\widehat{\mathcal{X}}^a$ for each agent a. In our example, modelling downtime limitations, $\mathcal{X}^\lambda = \{P_{\max}\} \cup \{P_t \mid t \in \mathcal{W}\}$ and $\widehat{\mathcal{X}}^a = \{P^*_{\min}, P^*_{\max}\} \cup \{\top_t, \bot_t, \nu_t \mid t \in \mathcal{W}\}$ would be a suitable choice. We need $P_{\max}$ to specify load factors later on and can technically distinguish $P_{\max}$ from P_t since the former are indeed constants and not decision variables as their value is fixed for a power generator. However, for ease of presentation we assume $P_{\max}$ to be just a decision variable with a domain consisting of only one value and treat all variables alike. The same holds for $P^*_{\min}$ and $P^*_{\max}$ that locally specify preferred ranges of operation. Then, the valid scope for all constraints within a single IAM of an agent a is $\mathcal{X}^\lambda \cup \widehat{\mathcal{X}}^a = \mathcal{X}^a$. The set of individual constraints $\mathcal{C}^a$ consists of both hard and soft constraints, $\mathcal{C}^a_h$ and $\mathcal{C}^a_s$ where $\mathcal{R}^a$ represents the constraint relationships defined over $\mathcal{C}^a_s$. We illustrate how to tie the auxiliary local variables in $\widehat{\mathcal{X}}^a$ to the interface variables with the vector denoting whether a plant is running:

$$\forall t \in \mathcal{W} : \nu_t \leftrightarrow P^0_t > 0$$

This expression helps in defining additional constraints and can be implemented by a decision expression in OPL directly. An individual preference for a certain production subrange $[P^*_{\min} P^*_{\max}]$ is specified by the soft constraint:

$$\forall t \in \mathcal{W} : P^*_{\min} \leq P^a_t \leq P^*_{\max}$$

To summarize, IAMs are specified as constraint satisfaction problems and represented as $\langle \mathcal{X}^a, \mathcal{D}^a, \mathcal{C}^a, \mathcal{R}^a \rangle$ with suitable choices of domains.

Next, consider the organisational template (OT). It provides at least the interface variables for *all* its subordinate agents – i.e., in contrast to a single IAM which specifies in terms of variables $x \in \mathcal{X}^\lambda$, the OT consists of a set of interface variables $\mathcal{X}^{\mathcal{A}^\lambda} = \{x^a \mid x \in \mathcal{X}^\lambda, a \in \mathcal{A}^\lambda\}$ for every agent a. These variables have to be provided in every IAM so the OT can use them for defining the organisational goal. Note that this would certainly not be true for local variables. It is not possible to specify, e.g., $\top^a_t$ indexed by a as it does not have to be defined in every model – contrary to a parametrised scheme. For unit commitment, the objective is taken from Eq. 2:

$$f = \underset{P^a_t}{\text{minimise}} \sum_{t \in \mathcal{W}} \left| P^\lambda_t - \mathcal{D}_t \right|,\ P^\lambda_t = \sum_{a \in \mathcal{A}} P^a_t$$

where $\mathcal{D}_t$ are constants representing the demand at time steps t and P^λ_t can either be seen as an additional decision variable in OT restricted by the constraint to be the sum of all agent productions or directly as a decision expression that is fully determined by the value of its associated decision variables (P^a_t). A soft fairness constraint for a VPP could be that no power plant should produce less than 40 % of its nominal capacity, $P_{\max}$, if possible. We call this organisational constraint oc1:

$$\text{oc1} : \forall a \in \mathcal{A}^\lambda, t \in \mathcal{W} : P^a_t \geq 0.4 \cdot P^a_{\max} \tag{4}$$

Similarly, we could impose a soft upper bound (oc2) for the load factor to prefer schedules that do not rely on few power generators providing all energy. All together, an organisation template is given by $\langle \mathcal{X}^{\mathcal{A}^\lambda}, \mathcal{D}^\lambda, \mathcal{C}^\lambda, \mathcal{R}^\lambda, f \rangle$ – the interface variables instantiated for each agent and their domains, a set of organisational hard and soft constraints with optional constraint relationships and an optimisation function f.

B. Steps of Model Synthesis

Synthesis takes a set of IAMs $\langle \mathcal{X}^a, \mathcal{C}^a \rangle$ for agents $a \in \mathcal{A}^\lambda$ and an organisation template to create the synthesised model (SM) for λ, a CSOP $\zeta = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{R}, f \rangle$, the construction of which we discuss step-by-step.

Step 1: First we make sure that local variables do not clash when combining several agents by renaming them to a unique identifier. A substitution operator $\{x_1 \mapsto x_2\}$ replaces all occurrences of the variable x_1 by the variable x_2 in a constraint or optimisation function. We then write x^a for a variable $x \in \mathcal{X}^a$. Note that this intentionally connects the interface variables $\mathcal{X}^\lambda$ used in an IAM with the counterpart in the OT, $\mathcal{X}^{\mathcal{A}^\lambda}$. The set of variables of ζ is then defined as $\mathcal{X} = \mathcal{X}^{\mathcal{A}^\lambda} \bigcup_{a \in \mathcal{A}^\lambda} \{x^a \mid x \in \widehat{\mathcal{X}}^a\}$ (furthermore, constraints keep their original domains to form $\mathcal{D}$).

The constraints $\mathcal{C}$ then consist of all hard and soft constraints of the organisational template and the constraints in all IAMs after substituting variables by their labelled counterparts. Hence:

$$\mathcal{C} = \mathcal{C}^\lambda \cup \bigcup_{a \in \mathcal{A}^\lambda} \{c\{x \mapsto x^a, \forall x \in \mathcal{X}^a\} | c \in \mathcal{C}^a\}$$

Constraint relationships for the synthesised model can be formed under the assumption that organisational constraints have a higher priority than individual preferences.

Step 2: As Fig. 4 shows, we then impose artificial edges that specify that all constraints in the OT are more important than all soft constraints of IAMs that do not have constraints that are deemed more important. However, this is a design decision as it might not be desirable for all problems to strictly prioritise organisational constraints. It should be noted further, that constraint relationships of different IAMs all have a disjoint scope – the respective $\mathcal{X}^a$ sets – and that the constraint relationships are subject to the variable renaming as well.

$$\mathcal{R} = \mathcal{R}^\lambda \cup \bigcup_{a \in \mathcal{A}^\lambda} \mathcal{R}^a \cup \{(x, y) \mid x \in \mathcal{C}^\lambda_s, y \in \mathcal{C}^a_s, \nexists z : z \succ_{\mathcal{R}^a} y\}$$

Step 3: Finally, as the optimisation function is defined in terms of the instantiated interface variables, $\mathcal{X}^{\mathcal{A}^\lambda}$, we can keep f as the objective of ζ. This yields two possibly conflicting objectives regarding the satisfaction of individual and organisational soft constraints versus meeting the original objective f. We could then use an existing multi-objective optimisation technique such as utopia search [24] but propose an alternative three- stage optimisation to strongly favour the organisational objective f while still optimising p if possible.

C. Multi-objective Optimisation with Constraint Relationships

During synthesis, constraint relationships are responsible for combining soft constraint priorities as well as establishing a

connection to organisational constraints. Fig. 4 shows how agents have different preferences on soft constraints that only affect their individual performance, e.g., *economically optimal* states that prescribe that production should better be in this range whereas other values are technically feasible. Optimizing the SM $\zeta = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{R}, f \rangle$ with respect to the original objective *and* soft constraint satisfaction (as defined by the penalty function p in Eq. 1) can be achieved as follows (w. l. o. g. we restrict ourselves to minimization problems):

1) Let $\hat{f}$ be the optimal result of ζ (with the original objective f).
2) Find an upper/lower bound f^* for the objective such that $f^* = \delta_f \times \hat{f}$ for some x, where $\delta_f \geq 1.0$ for a minimisation problem. Note that for a non-negative minimisation objective, $\delta_f \times \hat{f} = 0$ so the bound collapses to a single point, 0 such that any valid solution has to be optimal if at least one is.
3) Impose a constraint to restrict $f(\theta)$, $\theta \in (\mathcal{X} \rightarrow \mathcal{D})$. Let ζ' be the problem that minimises the violation of penalties with respect to the bound f^*: $\zeta' = \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup \{c' : (f(\theta) \leq f^*\}, \mathcal{R}, p \rangle$.
4) Solve ζ' and let $\hat{p}$ be the minimal sum of penalties of violated constraints.
5) Impose a restriction on the sum of penalties to be less than or equal to $\hat{p} \times \delta_p$ and solve for the original objective: $\zeta'' = \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup \{c'' : (p(\theta) \leq \hat{p}\}, \mathcal{R}, f \rangle$. This step is necessary as otherwise a solver might just lie within the tolerance of f^* even if better solutions in terms of f (having the same penalty sum) exist.

Appropriate choices for δ_f and δ_p have to be found specific to a problem but so do weights of a single combined objective function commonly employed in a global criterion method [24]. This approach can certainly be costly if the optimisation problems themselves are hard and time-consuming to solve – but if the coalitions are sufficiently small and organised (as, e.g., in a hierarchical system [9]) the benefits of finding solutions that are attractive to the individual participants outweigh the expensive optimisation runs. Moreover, the solutions found in previous steps can be used as starting points in subsequent runs to speed up the optimisation and other multi-objective optimisation approaches such as utopia search require several optimisation runs as well.

V. Implementing Synthesis in CPLEX

We exemplify the synthesis process with an example of a VPP consisting of three power plants with heterogeneous constraints modelling the requirements presented in Sect. III. For each power generator, we formulate the individual constraints and explain the usage of local and interface variables. The problem is comparable to the parametrised model presented in Listing 1. The models are directly presented in the Optimisation Programming Language (OPL) used in CPLEX [18] to provide a prototype even though our concepts do not rely on any specific CPLEX features such that the models could equivalently have been presented in a pseudo-code for CSOPs. Extensions to OPL regarding soft constraints are therefore implemented using the keyword *SOFT-CONSTRAINTS* in comments. We do not describe entire models (which are provided online[2]) but rather highlight the most important aspects.

A. Organisational Template

We start with the organisational template as it contains the core optimisation problem to be tackled. The first section indicates the identifier of the set consisting of the agents' identifiers (*plants* in our case) as well as generally needed constants such as the time series and load curve.

```
{string} plants = ...;
range TIMERANGE = 0..5;
float loadCurve[TIMERANGE] =
  [200.0, 250.0, 230.0, 247.0, 349.0, 551.0];
```

The set of *plants* needs to be filled with the identifiers of actual power plant models. *TIMERANGE* and *loadCurve* are furthermore examples of interface variables that individual agents may use to define their local variables and constraints. Additional decision variables and expressions common to all agents are described in the next section. *Production* for different time steps are the decision variables in this example and we require each agent to provide its maximal output (nameplate capacity) such that we are able to define the load factor of each plant. Note that these decision variables are indexed by the set of child agents ($\mathcal{X}^{\mathcal{A}^\lambda}$). Decision expressions primarily are syntactic tools to facilitate the formulation of optimization functions and constraints. They have to be fully determined by the value of the decision variables but offer to aggregate a set of decision variables using the *sum*, *min*, or *max* constructs. The presented expressions serve to provide a formulation of the constraint presented in Eq. 4.

```
float P_max[plants] = ...;
dvar float+ production[plants][TIMERANGE];
dexpr float totalProduction[t in TIMERANGE] =
  sum(p in plants) production[p][t];
dexpr float loadFactor[p in plants][t in TIMERANGE] =
  production[p][t] / P_max[p];
dexpr float minLoadFact[t in TIMERANGE] =
  min(p in plants) loadFactor[p][t];
dexpr float maxLoadFact[t in TIMERANGE] =
  max(p in plants) loadFactor[p][t];
```

Violation takes the sum of the absolute values of the deviation between the aggregated production and the load curve. It is the quantity that we aim to minimize.

```
dexpr float violation = sum(t in TIMERANGE)
  abs(totalProduction[t]-loadCurve[t]);

minimize violation;
```

We can explicitly request that all productions are below the nameplate capacity as a *hard* constraint. Furthermore, this organisational template provides for two soft constraints regarding distributions of load — denoted by $\mathbf{oc1}$ and $\mathbf{oc2}$. They indicate that the load factor of the individual agents should not

[2]Please refer to footnote 4

vary too much if possible thereby avoiding a highly skewed resource allocation. In particular, power plant operators might expect to have their generator contribute at least to a certain extent to generate revenue. Indifference between **oc1** and **oc2** is expressed by not modelling a relationship.

```
subject to {
  forall (t in TIMERANGE) {
    oc1: minLoadFact[t] >= 0.4;
    oc2: maxLoadFact[t] <= 0.6;
    forall (p in plants) {
      production[p][t] <= P_max[p];
    }
  }
};

/* SOFT-CONSTRAINTS
oc1
oc2 */
```

Compared to Listing 1, we only require a subset of the homogeneous aspects (the maximal power). However no restriction in terms of rates of change and other, inertia-based constraints are imposed as they are part of the individual agent models.

B. Individual Agent Models

We describe three types of power plant models consisting of the constraints presented in Sect. III. These are implemented independently from the organisational template but also contain definitions for the interface variables to be tested in advance. The first type, **A**, does not implement a warm and cold start-up and does not foresee minimal up and down times. Thus, it corresponds to a power plant that can be started up fast enough for the considered time step durations [19]:

```
float P_min = 50.0; float P_max = 100.0;

float rateOfChange = 0.15;
dvar float production[TIMERANGE];
dexpr int running[t in TIMERANGE] = !(production[t] == 0);

subject to {
  forall (t in 0..TIMERANGE) {
    running[t] => production[t] >= P_min;
    rate_of_change: (running[t] == 1) && (running[t+1] == 1)
     => abs(production[t] - production[t+1])
          <= production[t] * rateOfChange;
    c1: (running[t] == 1) && (running[t+1] == 1)
     => abs(production[t] - production[t+1])
          <= production[t] * 0.07;
    c2: (running[t] == 1) && (running[t+1] == 1)
     => abs(production[t] - production[t+1])
          <= production[t] * 0.10;
  }
};
/* SOFT-CONSTRAINTS
 c1 >> c2 */
```

During synthesis, the system automatically distinguishes between *P_max* being an interface variable and *P_min* being a local variable. Some power generators such as hydro power plants could be throttled down to no production at all, whereas others can enforce a minimal operation production as modelled in this example. The decision expression *running* helps syntactically to distinguish between these states. Note that here, *production* is not indexed over any agent set, so an individual agent model only has access to its own decision variables. We furthermore have two constraints reflecting a preference for small rates of change which aim for operational stability.

The second type, **B**, is used to express minimal uptimes [15]. We need additional decision variables to capture the number of time steps a plant is running consecutively at a particular time step. Based on these, we can decide whether a transition from on to off is feasible. And for the sake of the argument, assume that no relative rate of change but rather a fixed rate of change at every production level is given (we omit variables already discussed). We also assume three ranges of different economical preference that can be expressed with soft constraints:

```
int minUpTime = 2;
float fixedChange = 20;

dvar int+ consRunning[TIMERANGE];
 [...]
forall (t in TIMERANGE) {
  c1: production[t] >= 22 && production[t] <= 25;
  c2: production[t] >= 20 && production[t] <= 30;
  c3: production[t] >= 18 && production[t] <= 33;

  fixed_change: (running[t] == 1 && running[t+1] == 1) =>
    abs(production[t] - production[t+1]) <= fixedChange;
  cons_run: (running[t+1] == 1 &&
              consRunning[t+1] == (1 + consRunning[t])) ||
            (running[t+1] == 0 &&
              consRunning[t+1] == 0);

  min_up_time: (running[t] == 1 && running[t+1] == 0) =>
                (consRunning[t] - minUpTime) >= 0;
}
/* SOFT-CONSTRAINTS
 c1 >> c2
 c2 >> c3 */
```

Note that it becomes apparent that a parametrised model would now be severely limited. To support minimal up times within a MIP framework, we need those variables (or implement custom constraints and propagators for a constraint solver) but would have to offer these variables for *all* plants.

Finally, our third type, **C**, incorporates hot and cold start-up times based on the down time (with down time being defined analogously to up time in the previous model). In essence, we provide a MIP formulation for the transition system presented in Fig. 2 implementing start-up times that respect the data shown in Table I. We use a stepwise function that returns 2 time steps duration for down times of less than 3 and 4 time steps for longer down times. A decision variable *signal* stores when to initiate a start-up process and we need to enforce that those signals are only sent when a plant is in the appropriate state (e.g., sending a start-up signal only when in the *idle* state). Additionally, we also have three soft constraints (c1, c2, c3) regarding economical ranges and rates of change.

```
int IDLE = 0;
int STARTING = 1;
```

```
int STOPPING = 2;
int UP = 3;
stepFunction startUp = stepwise{ 2->3; 4 };

dvar int+ countdown[TIMERANGE];
dvar int+ powerPlantState[TIMERANGE] in 0..3;
dvar int+ consStopping[TIMERANGE];
dvar int+ consRunning[TIMERANGE];
dvar int+ state[TIMERANGE] in 0..3;
dvar int signal[TIMERANGE] in -1 .. 1;
 [...]
forall (t in TIMERANGE) {
 c1: production[t] >= 300.0 && production[t] <= 350.0;
 c2: production[t] >= 280.0 && production[t] <= 370.0;
 c3: abs(production[t] - production[t+1]) <= 20;

  signal_states : signal[t] == 1 => state[t] == IDLE &&
                  signal[t] == -1 => state[t] == UP;
 state[t] == IDLE =>
        ( state[t+1] == IDLE ||
        ( state[t+1] == STARTING &&
          signal[t] == 1 &&
         (countdown[t+1] == startUp(consStopping[t ]))));

 ( state[t] == STARTING && countdown[t] >= 1) =>
        ( state[t+1] == STARTING &&
          countdown[t+1] == countdown[t] - 1);
 ( state[t] == STARTING && countdown[t] == 0) =>
          state[t+1] == UP;
 state[t] == UP => (state[t+1] == UP ||
       ( state[t+1] == IDLE && signal[t] == -1));

}
/* SOFT-CONSTRAINTS
 c1 >> c2
 c1 >> c3 */
```

This model is significantly more detailed than the previous ones and indicates how future realistic models could be integrated to achieve more accurate schedules.

C. Synthesised Model

Combining these three individual agent models with the organisational template results in one CPLEX model where constraints, local variables and interface variables are replaced as described earlier. In contrast to the formal definition (where x would just be replaced by x^a), we can however distinguish interface from local variables in the synthesised model. This is due to the fact that it can come in handy to have interface variables indexed by plant identifiers (e.g., for defining the expression `totalProduction`). As we cannot assume local variables to be available for every individual agent, we add the plant identifier to the variable name, such `rateOfChange` would become `rateOfChange_a` for a plant `a`. The individual constraint relationship graphs are combined with the organisational template yielding a synthesised graph as depicted in Fig. 4. We give some snippets that show parts of the synthesised model.

```
{string} plants = {"b", "c", "a"};
float P_max[plants] = [35.0, 400.0, 100.0];
dexpr float totalProduction[t in TIMERANGE] =
  sum( p in plants ) production[p][t];
 [...]
```

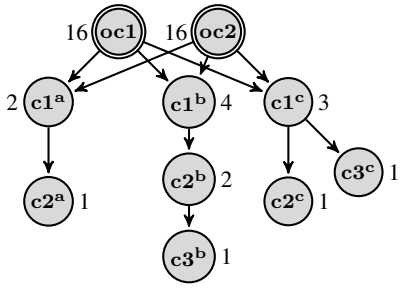

Fig. 4. Synthesised Constraint Relationship Graph with weights presented for TPD

```
float rateOfChange_a = 0.15;
float fixedChange_b = 5;
dvar int+ consStopping_b[TIMERANGE];
 [...]
subject to {
  forall (t in TIMERANGE) {
   c1_b: production["b"][t] >= 22 && production["b"][t] <= 25;
  }
   [...]
};
/* SOFT-CONSTRAINTS
oc1
oc2
oc1 >> c1_b;
oc1 >> c1_a;
oc2 >> c1_a; */
```

The last step to acquire a solvable model is then to transform the constraint relationships into weights and add decision variables and expressions to incorporate these weights as penalties to come to a scalar objective function. Essentially, all soft constraints are collected and used to index a vector of penalties and a reformulation is performed for each soft constraint $c \in \mathcal{C}_s$:

$$c' : (c \wedge p_c = 0) \vee (\neg c \wedge p_c = w(c))$$

which leads to the following changes in OPL:

```
{string} softConstraints = {"c1_b", "c1_c" ,[...]};
dvar int+ penalties[softConstraints][TIMERANGE];
dexpr float penaltySum = sum(t in TIMERANGE,
                             c in softConstraints) penalties[c][t];
 [...]
c1_b: (production["b"][t] >= 22 && production["b"][t] <= 25
       && penalties["c1_b"][t] == 0) ||
      (!( production["b"][t] >= 22 && production["b"][t] <= 25)
       && penalties["c1_b"][t] == 4);
```

We then have `penaltySum` as the expression to minimize the violation of soft constraints.

VI. Evaluation

Evaluating our approach at this stage is not straightforward as to the authors' knowledge, no public benchmark library containing unit commitment models is available. In addition, the described synthesis is primarily a methodology to incorporate heterogeneous models, not a concrete algorithm, thus performance issues are not yet in the focus of development. To

still get a quantitative (and as objective as possible) evaluation about how well individual preferences can be considered, we performed synthetic randomised experiments based on the constraints presented in the paper and power plant data[3].

The source code of both the experiments and the example discussed in Sect. V are available online[4] to facilitate replication of our experiments.

A. Experimental Design

We first generate a pool of 400 power plant models based on the types presented in Sect. V taking into account existing data of gas turbines and biomass power plants to capture plant-specific properties. We then perform a number of optimisation runs by picking a set out of these power plants randomly, synthesising them into one model and running the three-stage-optimisation with the *synthesised model*. The following parameters characterise one experiment:

- n the number of power plants
- δ_v the delta applied to the first optimum in the three-stage-optimisation as a tolerance
- δ_p the delta applied to the penalty in the third step
- d the dominance level SPD or TPD
- r the number of runs that are conducted

Several measurements are taken to evaluate properties of the synthesis and averaged over the r runs:

- v_1 – v_3 the violation (discrepancy between demand and production) for the three optimisation steps
- p_1 – p_3 the same holds for the aggregated penalties
- $\overrightarrow{v}$, $\overrightarrow{q}$ measures the relative improvement or worsening of v_1 to v_3 and p_1 to p_3 to show how the objectives change when considering penalties: $\overrightarrow{v} = \frac{v_1}{v_3}$, $\overrightarrow{p} = \frac{p_1}{p_3}$
- **#pred** is the average number of predecessors per violated constraint as a measure for how more important constraints are treated
- **#vc** sums up the total number of violated soft constraints relative to the number of all soft constraints (with $\overrightarrow{\#\text{vc}}$ denoting the relative improvement from the first optimisation step to the third)

Note that #pred and #vc are evaluated for the final result returned by the three-stage-optimisation. Some runs failed to provide a correct solution within a threshold of 3 minutes. We excluded those results from the evaluation as our focus is not yet on performance and robustness issues.

B. Experimental Results

We examine questions of interest and present the results of the experiment runs.

a) Solution quality: How does taking into account individual preferences affect the solution quality? We want to obtain an impression about how the choice of the parameters δ_v and δ_p affect the solution quality. For all choices of parameters, we found that allowing a small tolerance regarding the mismatch of demand and production allowed for comparable, substantial improvements in the satisfaction of soft constraints as Table II shows. If a violation tolerance of 10% was imposed, the solver still managed to reduce the number of violated soft constraints by half from the optimal solution to the final one while staying within a range of 1 % optimality. As expected, increasing the violation tolerance leads to better reductions in terms of penalties and soft constraints.

TABLE II
COMPARISON OF DIFFERENT DELTAS REGARDING THE SOLUTION QUALITY. MEASUREMENTS REPRESENT AVERAGES OVER 50 RUNS FOR 9 TIME STEPS WITH $n = 5, d = \text{SPD}$ AND STANDARD DEVIATIONS.

(δ_v/δ_p)	1.1/1.2	1.2/1.2	1.2/1.1	1.3/1.2
$\overrightarrow{v}$	1.0039	1.0161	1.056	1.022
	(0.0076)	(0.0286)	(0.044)	(0.034)
$\overrightarrow{p}$	0.599	0.546	0.534	0.527
	(0.116)	(0.142)	(0.142)	(0.148)
$\overrightarrow{\#\text{vc}}$	**0.555**	**0.502**	**0.486**	**0.487**
	(0.132)	(0.152)	(0.181)	(0.163)

TABLE III
COMPARISON OF DIFFERENT DOMINANCE LEVELS. MEASUREMENTS REPRESENT AVERAGES OVER 50 RUNS FOR 9 TIME STEPS WITH $\delta_v = \delta_p = 20\%$ AND STANDARD DEVIATIONS.

(n/d)	5/SPD	5/TPD	10/SPD	10/TPD
$\overrightarrow{v}$	1.022	1.0001	1.027	1.0
	(0.027)	(0.0009)	(0.038)	(0.0005)
$\overrightarrow{p}$	0.595	0.7484	0.45	0.774
	(0.169)	(0.15)	(0.085)	(0.168)
#pred	**2.905**	**2.541**	**3.23**	**2.76**
	(0.406)	(0.2569)	(0.41)	(0.281)
#vc	**0.34**	**0.468**	**0.283**	**0.462**
	(0.12)	(0.138)	(0.064)	(0.123)

b) Influence of dominance property: How does the selected dominance property affect the number of violated soft constraints? The dominance property influences how much more important a single constraint is with respect to its dominated constraints. In the case of single predecessor dominance, a constraint is only more important than one of its predecessors, not a whole set. Therefore the weights lie more closely to each other and no strong "hierarchical" difference is imposed. Choosing the property, however, is not straightforward as this substantially influences the number of soft constraints that are "dropped" by a solver in favour of more important ones. We observe this behaviour in Table III: The percentage of violated soft constraints is significantly higher when using TPD than SPD for both 5 and 10 power plants while the average number of predecessors per violated constraint (measuring its importance) is lower when using TPD. TPD semantics lead to an average dissatisfaction of about 40% of all soft constraints, whereas SPD only dissatisfies 30% albeit returning slightly worse solutions (about 2% higher demand violations when 20% higher were allowed). It is thus a relevant question for preference elicitation and requirements engineering to find out whether the system's constraints are more hierarchical or egalitarian. Constraint hierarchies correspond to TPD semantics [9] and we argue

[3]see http://www.energymap.info/ and http://www.lew-verteilnetz.de/
[4]https://github.com/Alexander-Schiendorfer/synthesisenergycoalitions

that there are certainly circumstances where having more soft constraints fulfilled is more desirable than just satisfying the most important ones and no others.

VII. Conclusion and Outlook

Although there is a vast body of literature on efficient models and algorithms for unit commitment problems (see, e.g., [20], [14], [19], [15] for a modest selection), to the best of our knowledge there is no approach that consolidates these various types of models and addresses the domain's inherent heterogeneity. We proposed an approach that attempts to leverage existing models and to simplify the engineering of optimisation problems with a well-defined modelling methodology. In addition, we showed how start-up behaviour of thermal power generators presented in [16] or [19] can be extended to account for start-up times that depend on the previous down time using transition systems. We showed a MIP formulation for this adaptive start-up problem and used heterogeneous models employing different start-up behaviour as the case study for our approach. Moreover, the modelling and synthesis strategy presented allows unit operators to express individual preferences that could not be considered before. Our first experiments indicate that this may increase the willingness of single power generators to collaborate in collective schemes where autonomy is sacrificed for potential economic benefits as we could halve the number of violated soft constraints. As the process is fully automated, we plan to combine it with models of the resource-levelling problem faced in the scheduling of (short-lived) coalitions of energy consumers to achieve better prices.

Acknowledgment

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

References

[1] S. D. Ramchurn, P. Vytelingum, A. Rogers, and N. R. Jennings, "Putting the 'smarts' into the smart grid: a grand challenge for artificial intelligence," *Commun. ACM*, vol. 55, no. 4, pp. 86–97, Apr. 2012. doi: 10.1145/2133806.2133825. [Online]. Available: http://doi.acm.org/10.1145/2133806.2133825

[2] H. Morais, P. Kádár, P. Faria, Z. A. Vale, and H. Khodr, "Optimal scheduling of a renewable micro-grid in an isolated load area using mixed-integer linear programming," *Renewable Energy*, vol. 35, no. 1, pp. 151 – 156, 2010. doi: http://dx.doi.org/10.1016/j.renene.2009.02.031. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0960148109001001

[3] P. Arcuri, G. Florio, and P. Fragiacomo, "A mixed integer programming model for optimal design of trigeneration in a hospital complex," *Energy*, vol. 32, no. 8, pp. 1430 – 1447, 2007. doi: http://dx.doi.org/10.1016/j.energy.2006.10.023. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360544206003197

[4] L. G. Fishbone and H. Abilock, "Markal, a linear-programming model for energy systems analysis: Technical description of the bnl version," *International Journal of Energy Research*, vol. 5, no. 4, pp. 353–375, 1981. doi: 10.1002/er.4440050406. [Online]. Available: http://dx.doi.org/10.1002/er.4440050406

[5] S. Thiébaux, C. Coffrin, H. Hijazi, and J. Slaney, "Planning with MIP for supply restoration in power distribution systems," in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, ser. IJCAI'13. AAAI Press, 2013. ISBN 978-1-57735-633-2 pp. 2900–2907. [Online]. Available: http://dl.acm.org/citation.cfm?id=2540128.2540546

[6] T. Werner, "DEMS – The Decentralized Energy Management System." http://w3.siemens.com/smartgrid/global/en/smart-grid-world/experts-talk/pages/dems.aspx, 2013, [Online; accessed 07-January-2014].

[7] "PLEXOS Integrated Energy Model," http://energyexemplar.com/software/plexos-desktop-edition/, January 2014, [Online; accessed 09-January-2014].

[8] C. Tesche, "Stadtwerke München und Siemens nehmen ein virtuelles Kraftwerk in Betrieb (in German)," http://www.energie-und-technik.de/smart-grid-smart-metering/news/article/87399/0/Stadtwerke_Muenchen_und_Siemens_nehmen_ein_virtuelles_Kraftwerk_in_Betrieb/, 2012, [Online; accessed 07-January-2014].

[9] A. Schiendorfer, J.-P. Steghöfer, and W. Reif, "Synthesis and Abstraction of Constraint Models for Hierarchical Resource Allocation Problems," in *Proc. of the 6th International Conference on Agents and Artificial Intelligence (ICAART)*, vol. 2. SciTePress, March 2014. doi: 10.5220/0004757700150027. [Online]. Available: http://dx.doi.org/10.5220/0004757700150027

[10] E. Tsang, *Foundations of constraint satisfaction*. Academic press London, 1993, vol. 289.

[11] A. Schiendorfer, J.-P. Steghöfer, A. Knapp, F. Nafz, and W. Reif, "Constraint relationships for soft constraints," in *Research and Development in Intelligent Systems XXX*, M. Bramer and M. Petridis, Eds. Springer International Publishing, 2013, pp. 241–255. ISBN 978-3-319-02620-6. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-02621-3_17

[12] P. Meseguer, F. Rossi, and T. Schiex, "Soft Constraints," in *Handbook of Constraint Programming*, F. Rossi, P. van Beek, and T. Walsh, Eds. Elsevier, 2006, ch. 9.

[13] G. Anders, A. Schiendorfer, J.-P. Steghöfer, and W. Reif, "Robust Scheduling in a Self-Organizing Hierarchy of Autonomous Virtual Power Plants," in *Proc. of the 2nd International Workshop on „Self-optimisation in Organic and Autonomic Computing Systems" (SAOS14) in conjunction with ARCS 2014*, vol. 2, February 2014.

[14] L. Rani, M. Mam, and S. Kumar, "Economic load dispatch in thermal power plant taking real time efficiency as an additional constraints," *International Journal of Engineering Research & Technology (IJERT)*, vol. 2, no. 7, 2013. [Online]. Available: www.ijert.org

[15] N. Padhy, "Unit commitment-a bibliographical survey," *Power Systems, IEEE Transactions on*, vol. 19, no. 2, pp. 1196–1205, 2004. doi: 10.1109/TPWRS.2003.821611

[16] J. Arroyo and A. Conejo, "Modeling of start-up and shut-down power trajectories of thermal units," *Power Systems, IEEE Transactions on*, vol. 19, no. 3, pp. 1562–1568, 2004. doi: 10.1109/TPWRS.2004.831654

[17] L. Jarass and G. Obermair, *Welchen Netzumbau erfordert die Energiewende?: Unter Berücksichtigung des Netzentwicklungsplans Strom 2012 (in German)*, ser. MV-Wissenschaft. Monsenstein und Vannerdat, 2012. ISBN 9783869916415. [Online]. Available: http://books.google.de/books?id=jID5HQcICLkC

[18] "IBM ILOG CPLEX Optimizer," http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/, December 2013.

[19] J. Šumbera, "Modelling generator constraints for the self-scheduling problem," in *Vědecký seminář doktorandů FIS – únor 2012*, Prague, Czech Republic, Feb 2012. ISBN 978-80-245-1862-6

[20] C. Rajan, "An evolutionary programming based tabu search method for unit commitment problem with cooling-banking constraints," in *IEEE Power India Conference, 2006*, 2006. doi: 10.1109/POWERI.2006.1632557 p. 8.

[21] J. N. Mayer, N. Kreifels, and B. Burger, "Kohleverstromung zu Zeiten niedriger Börsenstrompreise," Aug. 2013, http://www.ise.fraunhofer.de/de/downloads/pdf-files/aktuelles/kohleverstromung-zu-zeiten-niedriger-boersenstrompreise.pdf/view.

[22] V. Dignum and J. Padget, "Multiagent organizations," *Multiagent Systems, G. Weiss, ed., MIT Press,*, 2013.

[23] C. Derksen, T. Linnenberg, R. Unland, and A. Fay, "Unified energy agents as a base for the systematic development of future energy grids," in *Multiagent System Technologies*, ser. Lecture Notes in Computer Science, M. Klusch, M. Thimm, and M. Paprzycki, Eds., vol. 8076. Springer Berlin Heidelberg, 2013. doi: 10.1007/978-3-642-40776-5_21. ISBN 978-3-642-40775-8 pp. 236–249. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-40776-5_21

[24] R. Marler and J. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and Multidisciplinary Optimization*, vol. 26, no. 6, pp. 369–395, 2004. doi: 10.1007/s00158-003-0368-6. [Online]. Available: http://dx.doi.org/10.1007/s00158-003-0368-6