A Scenario State Representation for Scheduling Deferrable Loads under Wind Uncertainty

Erwin Walraven
Delft University of Technology
Mekelweg 4, 2628 CD
Delft, The Netherlands
e.m.p.walraven@tudelft.nl

Matthijs T. J. Spaan
Delft University of Technology
Mekelweg 4, 2628 CD
Delft, The Netherlands
m.t.j.spaan@tudelft.nl

ABSTRACT

Integration of renewable energy in power systems is a potential source of uncertainty, because renewable generation is variable and may depend on changing and highly uncertain weather conditions. In this paper we present and evaluate a new method to schedule power-demanding tasks with release times and deadlines under uncertainty, in order to balance demand and uncertain supply. The problem is considered as a multiagent sequential decision making problem where agents have to deal with uncertainty. Our main contribution is a scenario state representation and an algorithm that computes a belief over future scenarios, rather than states. The algorithm is used to recompute the belief when new information becomes available. Experiments show that our method matches demand and uncertain supply to reduce grid power consumption, and outperforms an existing online consensus scheduling algorithm.

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]: Scheduling; G.3 [**Probability and Statistics**]: Probabilistic algorithms; I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Algorithms, Performance

Keywords

Planning under Uncertainty, Scheduling, Smart Grids

1. INTRODUCTION

Handling uncertainty of renewable generation is an important challenge in the development of reliable smart grids. There is a worldwide growth of distributed renewable energy generation and governments want to accelerate this to address problems related to, for example, climate change [8]. An example is power generated by wind turbines. Although renewable wind energy is clean and cheap, it may be intermittent and its availability is uncertain and difficult to predict. For instance, the prediction of short-term wind power has a root mean square error of approximately 18 percent with a prediction horizon of 24 hours [5], which shows that decision making methods have to deal with uncertain information regarding the short-term future.

Research has shown that the most severe problems occur during peak-load hours if energy demand is high and wind power generation is interrupted, because the system may not have sufficient resources to compensate for the lack of wind power [9]. Demand-side management is a potential solution to deal with this problem, which intends to encourage consumers to adapt their behavior in terms of demand, timing and flexibility, as an alternative to improving renewable energy availability on the supply side [7]. To reduce peak power consumption and to mitigate the effects of uncertain renewable power supply, loads can be deferred in time, such that they can be executed during off-peak hours. An example of such a deferrable load is the charging process of an electrical vehicle, which often does not have to be charged immediately, as long as the available power in the battery is sufficient to reach a destination.

In this paper we present a new algorithm to schedule deferrable loads, which takes uncertain information regarding renewable supply into account. Our algorithm schedules deferrable loads with a release time, deadline and a fixed demand profile, and assumes that there are two energy sources: conventional generation and uncertain renewable supply. The demand is assumed to be fixed and known, i.e., there are no online arriving tasks. We formulate the problem as a sequential decision making problem with multiple agents and we apply planning to make online scheduling decisions. In particular, we formulate the problem as a special case of a Partially Observable Markov Decision Process (POMDP) [6, 17]. We model each deferrable load as an agent, which occurs naturally in practice if, for example, multiple electrical vehicles have to coordinate their energy consumption. Our algorithm belongs to a more general class of task scheduling algorithms, and therefore we use the terms load and task interchangeably.

A key idea in our work is that we do not only keep track of the current availability of renewable energy at an arbitrary timestep t. Instead, we compute a probability distribution over the entire set of scenarios, given the renewable supply observed until timestep t. In planning terminology, rather than maintaining a belief over states, we compute a belief over scenarios. We illustrate this with an example, also shown in Figure 1. The available renewable energy at time t (i.e., the state) is approximately the same for each scenario, but the trajectories until time t are different. If the available supply is very low until time t (i.e., the bottom trajectory is observed), then it is most likely that the sce-

Appears in: *The 10th Annual Workshop on Multiagent Sequential Decision-Making Under Uncertainty* (*MSDM-2015*), held in conjunction with *AAMAS*, May 2015, Istanbul, Turkey.

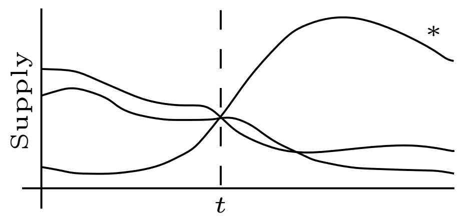

Figure 1: Scenarios and long-term correlations.

nario labeled $*$ predicts the future and supply is expected to increase after time t. If we only look at the supply at time t, which is identical for each scenario, we would not be able to derive this prediction because the renewable supply is not a Markovian signal, and is not easily modeled as such. This minimalistic example illustrates that scenarios account for long-term correlations in available supply.

Our main contributions can be summarized as follows. We consider an online task scheduling problem to balance demand and uncertain supply. For this problem we provide a planning formulation involving scenarios defining renewable supply, as illustrated above. In experiments we show that our method outperforms an existing online task scheduling algorithm in case of high supply uncertainty. As a concrete example of uncertain renewable energy, we include data from a real wind farm in our evaluation.

First we provide some background regarding planning under uncertainty in Section 2. Then we define an online task scheduling problem in Section 3, and we discuss two recently proposed algorithms for this problem. In Section 4 we show how the problem can be defined as multiagent decision making problem using the POMDP framework. Section 5 describes our experiments, in which we compare our algorithm with existing methods. In the remaining parts of the paper we discuss related work and we summarize our conclusions.

2. PLANNING UNDER UNCERTAINTY

In this section we provide background information about planning under uncertainty and sequential decision making. Planning under uncertainty involves agents that interact with their environment by executing actions, and observing effects caused by these actions. This is a challenging problem if agents are uncertain about the outcome of their action execution, and if they cannot fully observe the environment they are acting in. The Partially Observable Markov Decision Process (POMDP) formalism provides a framework to plan in such uncertain environments [6, 17]. In a POMDP, it is assumed that the environment is in a state $s \in S$. After executing an action $a \in A$ in state s, the state of the environment transitions to another state $s' \in S$ according to probability distribution $P(s'|s,a)$ and a reward $R(s,a)$ is received from the environment. A state transition from s to s' is only conditionally dependent on state s and action a, which is called the Markov property. In contrast to MDPs with full observability [13], the agent does not directly perceive the state of the environment in a POMDP. It receives an observation $o \in O$ that can be used to reason about the underlying MDP state of the environment, using a probability distribution $P(o|a,s')$. Since states are not directly observable in a POMDP, agents maintain a belief state, denoted b, which represents a probability distribution over states. The resulting belief state b_a^o after executing action a and observing o in belief state b can be determined using Bayes' rule. To act in a partially observable environment, agents use a policy $\pi(b)$, which maps belief states to actions. A policy $\pi(b)$ is characterized by a value function $V^{\pi}(b)$ defining the expected discounted reward collected by the agent when executing policy π from belief state b. Computing exact solutions to POMDPs is known to be intractable [11], but many approximate methods exist (see, e.g., [12, 18]). In this paper we use POMCP [16], an online Monte-Carlo planning algorithm that is capable of dealing with a large number of states. We use POMCP because it does not need explicit transition and observation probability distributions. It only requires a black-box simulator of the POMDP during planning. Another reason for using POMCP is that it can be used to sample scenarios, rather than sampling states from a belief state.

3. TASK SCHEDULING

Before presenting our planning algorithm to schedule tasks, we introduce the model of the task scheduling problem under consideration in this section. Several algorithms have been proposed to solve this task scheduling problem. Therefore, we also discuss an offline scheduling algorithm using a simple greedy heuristic, and an online consensus algorithm. The overview serves as an introduction to existing work, and we use the algorithms in our experiments to compare the performance of our new algorithm with existing methods.

3.1 Task Scheduling Model

In this section we formally define the online task scheduling problem, and we introduce the notion of scenarios that we exploit in our solution.

We consider a set of n power-demanding tasks, denoted by $J = \{j_1, \dots, j_n\}$, where each task j_i is parameterized by a duration l_i, release time r_i, deadline d_i and power demand p_i. Hence, we define each task j_i as a tuple $j_i = (l_i, r_i, d_i, p_i)$. A task is not allowed to start before its release time and must be finished by the deadline. The power demand p_i of task j_i represents the demand per timestep, which means that the total power consumption of task j_i equals $l_i \cdot p_i$. Tasks cannot be preempted during execution once a task has been started. We assume a finite time horizon T, defining the discrete timesteps $1, 2, \dots, T$.

There are two energy sources available: renewable energy derived from wind and conventional generation from the electricity grid. The availability of conventional generation is infinite and there is a cost function $c(u)$ defining the cost of consuming u units from the grid. We assume that renewable energy per timestep is finite, has zero cost and cannot be stored to be used in subsequent timesteps. The amount of renewable energy available is represented by a scenario $x = (x_1, x_2, \dots, x_T)$ defining the the number of units available at each timestep.

A schedule $S = (h_1, \dots, h_n)$ defines, for each task j_i a starting time h_i satisfying the following conditions:

$$h_i \geq r_i \qquad h_i + l_i - 1 \leq d_i \qquad (i = 1, \dots, n).$$

The first condition states that task j_i cannot start before its release time r_i, and the second condition defines that task j_i cannot run after the deadline d_i. The total demand $w(S,t)$

```
input  : tasks $J$ and scenario $x$
output: schedule $S$
1  sort tasks in $J$ by decreasing length
2  $S \leftarrow$ empty schedule
3  for $t = 1, \ldots, T$ do
4      $u_t \leftarrow$ units available at time $t$ in scenario $x$
5  end
6  foreach $j_i \in J$ do
7      $h_i \leftarrow$ minimum cost starting time of $j_i$ given $u$
8      $S \leftarrow S \cup \{h_i\}$
9      deduct renewable units consumed by $j_i$ from $u$
10 end
```

Algorithm 1: Offline greedy.

of a schedule $S = (h_1, \ldots, h_n)$ at time t is defined as follows:

$$w(S,t) = \sum_{i=1}^{n} I_S(i,t) \cdot p_i,$$

where $I_S(i,t)$ is an indicator function that equals 1 if task j_i runs at time t in schedule S, and equals 0 otherwise. The number of required grid units $U_{S,x}$ corresponding to schedule S and fixed scenario $x = (x_1, x_2, \ldots, x_T)$ can be computed as shown below:

$$U_{S,x} = \sum_{t=1}^{T} \max\left(w(S,t) - x_t,\ 0\right).$$

The cost $c(U_{S,x})$ is used as an objective function to be minimized, in order to match demand and the supply defined by the scenario. If the scenario is known prior to the first timestep, then an optimal solution can be computed using mixed-integer programming. However, in this paper we consider the case in which the scenario is revealed online, reflecting the uncertainty in renewable energy. This means that x_t becomes available and known to our algorithm at time t, and scheduling decisions have to be made online without certain information regarding the future. The partial scenario $(x_1, x_2, \ldots, x_t)$ revealed until time t is called a realization, denoted $q_{1,t} = (q_1, q_2, \ldots, q_t)$.

3.2 Offline Greedy Scheduling

A greedy algorithm to schedule the tasks in the case without uncertainty is shown in Algorithm 1. Initially the tasks are sorted by decreasing length, and then it greedily assigns starting times to tasks by computing the starting time that leads to minimum cost, given the starting times of the previously scheduled tasks and the remaining renewable energy units. The renewable energy units in the data structure u are used to keep track of the renewable supply available after scheduling a task. The intuition behind starting with the longest task is that it has the highest demand, and it would be more difficult to find a low-cost starting time if several smaller tasks have been scheduled already.

Algorithm 1 assumes that no tasks have been started already, and computes a starting time for each task in J. If a partial schedule exists and some tasks in J already have a starting time assigned, then the same algorithm can be used for the remaining tasks. However, it has to account for the renewable energy units consumed by the scheduled tasks before assigning starting times to the remaining tasks.

3.3 Online Consensus Scheduling

Ströhle et al. [19] present a multi-machine consensus algorithm to schedule multiple tasks under wind uncertainty in case of uncertain supply and demand. The problem we study is similar, without demand uncertainty. Since our problem is a special case of the problem they consider, we can directly apply their consensus algorithm to schedule tasks with fixed demand. In the remainder of this paper consensus refers to the m-consensus algorithm from [19], which we briefly describe below. The notation has been adapted in order to be consistent with the problem we defined in Section 3.1.

Consensus operates on a set of scenarios, denoted X, containing several possible scenarios defining the renewable supply at each timestep. Note that these scenarios are not necessarily the same as the scenario that is used for online scheduling. The algorithm starts with an empty realization $q_0 = \emptyset$, since no prior information is known. Subsequently, it incrementally builds the realization defining the renewable supply known until the current timestep.

At any timestep t, Algorithm 2 is called with schedule S as input, containing the starting times of tasks that have been started already, and the scenario set X and realization $q_{1,t}$. In the definition of the algorithm, OfflineGreedy represents a call to Algorithm 1, where the greedy algorithm takes partial schedule S into account when scheduling the remaining tasks. The function $\mathcal{L}(x|q)$ represents the likelihood that x predicts future renewable energy given the realization until the current timestep q. In our experimental setup we discuss how such a likelihood can be computed. The symbol $\perp$ denotes the decision to schedule no additional task. The consensus algorithm solves an offline scheduling problem for each scenario in X, and weights its decisions with the likelihood of the scenario. Based on the decisions, it selects one additional task to be started at time t, and the procedure repeats until no more tasks are started.

4. TASK SCHEDULING AS PLANNING

Given the uncertainty that is present when matching demand to uncertain supply, it is natural to consider this problem as a sequential decision making problem under uncertainty involving multiple cooperative agents. Coordination among agents is required to ensure that tasks do not start all at the same time when renewable energy suddenly becomes available. In this section we formulate the online task scheduling problem as a special case of a Partially Observable Markov Decision Process (POMDP), which allows us to apply online planning algorithms to make decisions under uncertainty. We argue how the online task scheduling problem can be formulated as a POMDP, and eventually we introduce a new online task scheduling algorithm for this problem based on POMCP, which is an online algorithm for solving POMDPs.

4.1 Inferring Beliefs over Scenarios

In this section we discuss beliefs over scenarios, and how we can infer the belief from observed renewable supply. In contrast to a standard state representation for renewable supply, a scenario accounts for long-term correlations and relates renewable supply observed in the past to renewable supply in the future. As we discussed before, the available renewable energy at time t in scenario $x = (x_1, x_2, \ldots, x_T)$ is defined by variable x_t, which becomes known at time t. Therefore, it is a natural assumption to let the set of POMDP

```
input  : partial schedule $S$, set of tasks $J$, realization $q$,
         scenario set $X$ and current timestep $t$
output: set of tasks $J_t$ starting at time $t$
$J_t = \emptyset$
do
    $f_i \leftarrow 0 \quad (i = 1, \dots, n)$
    $f_\perp \leftarrow 0$
    foreach $x \in X$ do
        $S_x \leftarrow$ OfflineGreedy$(J, x, S)$
        $p \leftarrow$ set of tasks starting at time $t$ in $S_x$
        if $J_t = p$ then
            $f_\perp \leftarrow f_\perp + \mathcal{L}(x|q)$
        else
            foreach $j_i \in J - J_t$ do
                if $j_i$ starts at time $t$ in $p$ then
                    $f_i \leftarrow f_i + \mathcal{L}(x|q)$
                end
            end
        end
        $k \leftarrow \arg\max_{i \in \{1,\dots,n\}} f_i$
        if $j_\perp > f_k$ then
            $j^* \leftarrow \perp$
        else
            $j^* \leftarrow j_k$
            $J_t \leftarrow J_t \cup \{j_k\}$
            add task $j_k$ to $S$ with starting time $t$
        end
    end
while $j^* \neq \perp$;
```

Algorithm 2: Consensus.

```
input  : realization $q_{1,t}$, scenario set $X$, threshold $\rho$
output: belief $b$
$X' \leftarrow \emptyset$
$d \leftarrow 0$
while $|X'| < \rho$ do
    $X' \leftarrow \{x \in X : \sum_{i=1}^{t} (q_i - x_i)^2 \leq d\}$
    $d \leftarrow d + 1$
end
foreach $x \in X$ do
    if $x \in X'$ then
        $b_x \leftarrow 1 \ / \ (\varepsilon + \sum_{i=1}^{t} (q_i - x_i)^2)$
    else
        $b_x \leftarrow 0$
    end
end
normalize $b$
```

Algorithm 3: Inferring a scenario belief.

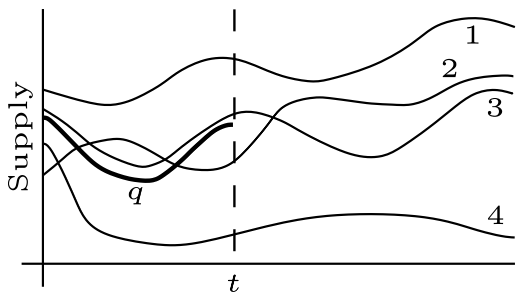

Figure 2: Scenarios and realization until time t.

observations correspond to the set of values these variables can take, such that x_t is observed at time t with probability 1 if x is the scenario. However, if we would apply a belief state update based on Bayes' rule, this would lead to problems if the realization $q_{1,t}$ does not correspond to the first t units of at least one scenario in X. In the remainder of this section we discuss an alternative method to compute a scenario belief, which infers the belief from the partial scenario observed so far.

Algorithm 3 computes a belief b over scenarios, given the realization until time t, scenario set X and a threshold ρ. The main idea behind the algorithm is that the future can be predicted by looking at scenarios similar to the realization. The similarity between a scenario x and the realization until time t can be measured by computing the sum of squared errors. Therefore, the algorithm constructs a set X' containing at least ρ scenarios similar to realization $q_{1,t}$, based on the sum of squared errors. Then it defines a probability distribution over the scenarios in the set X', where the probabilities are inversely proportional to the computed errors. The probability 0 is assigned to scenarios that are not in the set X'. A normalization step is performed on line 14 to ensure that the probabilities sum to 1. The symbol ε represents a very small non-zero constant to avoid division by zero.

Selecting scenarios based on an error metric ensures that probabilities are assigned to scenarios similar to the realization. An informal visual representation of this approach is shown in Figure 2, which contains four scenarios defining renewable supply. Timestep t is represented by the vertical dashed line, and the bold line labeled q represents the realization $q_{1,t}$ until this timestep. The supply and timesteps are depicted as being continuous, but it is also possible to apply the same techniques for discrete supply and discrete timesteps. As can be seen in the figure, the realization does not match any scenario until time t, but is very similar to scenario 3. The algorithm will assign a high probability to this scenario, because it has a small error. The remaining scenarios will get a lower probability, and the threshold ρ can be used to filter out scenarios for which the error is too high. If the set X contains scenarios that are sufficiently representative, then the procedure computes a probability distribution over scenarios defining future renewable energy availability starting from timestep $t+1$, given the realization until time t.

4.2 POMDP Task Scheduling Model

In this section we discuss an online task scheduling method based on POMDPs, in which each task is represented by an agent deciding whether it should start at the current timestep or not, given the decisions made by other agents and a belief regarding the scenario defining available renewable energy. We define n agents, where each agent corresponds to a task. Agents are able to start a task if they own a token, which is initially owned by agent 1. Once agent i has made a decision, it gives the token to agent $i+1$ $(1 \leq i < n)$, and the token returns to agent 1 once agent n has made a decision. In this approach, n actions are executed within one real-world timestep, before proceeding to the next timestep.

The rotating token allows us to reduce the size of the action space from exponential to constant. When considering all combinations of actions, however, the joint action space remains exponential in the number of agents. The same approach to reduce the size of the action space is used in [15]. The token approach requires that we store the decisions made by agents in a separate state variable, which will become clear in the remainder of this section. We model the multiagent problem as a POMDP, where agents are cooperative and we assume the problem to be centralized. We define a factored state space using the following state variables:

$s_x \in X$	scenario
$s_t \in \{1, \dots, T\}$	current timestep
$s_a \in \{1, \dots, n\}$	agent owning the token
$s_{r,i} \in \{1, \dots, T\}$	release time of task i $(i = 1, \dots, n)$
$s_{d,i} \in \{0, \dots, T\}$	delay steps of task i $(i = 1, \dots, n)$
$s_{s,i} \in \{0, \dots, T\}$	remaining steps of task i $(i = 1, \dots, n)$
$s_{c,i} \in \{\mathrm{R}, \mathrm{I}\}$	decision of agent i $(i = 1, \dots, n)$

Now we explain the definition of some of the state variables, and how they relate to the task scheduling problem from Section 3.1. The variable s_x denotes the scenario that describes the amount of renewable energy from time 1 to T. This state variable cannot be observed directly since it has to be derived from observations regarding, for instance, wind speed. For each agent i there is a variable $s_{r,i}$ to encode the release time r_i of the task. The variable $s_{d,i}$ represents the maximum number of timesteps that task i can be postponed, which can be used to encode the deadline d_i. To encode the length l_i of task i, the variable $s_{s,i}$ represents the number of timeslots task i still has to be running. The state also contains information regarding the last decision made by agent i for task i, represented by $s_{c,i}$, which can be either running (R) or idle (I). In total this state representation comprises $4n + 3$ state variables.

The action space represents the actions individual agents can take to start or postpone a task at a given timestep. Since we are dealing with multiple agents, a joint action space representation would scale exponentially in the number of agents. Our rotating token concept allows for a constant size action space. We define two actions RUN and IDLE, which correspond to the decisions for the agent owning the token. The factored state variables can be used to determine which actions are feasible to execute given the current state. If s_a equals j (i.e., agent j owns the token), then action RUN is feasible if $s_{s,j} > 0$ and $s_t \geq s_{r,j}$. Thus, action RUN can be executed if task j has been released and has not been completed yet. For agent j, action RUN decrements $s_{s,j}$ by one, sets $s_{d,j}$ to 0 and sets $s_{c,j}$ to R. This ensures that task j can never be postponed in the remaining timesteps. Similarly, if s_a equals j, then action IDLE is feasible if $s_{d,j} > 0$. It decrements $s_{d,j}$ by one if $s_{d,j} > 0$ and sets $s_{c,j}$ to I. IDLE is also executed if the task has been completed (i.e., $s_{s,j}$ equals 0). After each action execution, the token variable s_a is updated such that the next agent receives the token. If s_a equals n, then s_t is also incremented by one to proceed to the next timestep.

The rewards are negative, representing the cost of scheduling tasks, and depend on the decisions made by the agents and the available renewable supply defined by the scenario. For example, if two agents decide to run at time k and they both require 5 units, then the reward is determined by the

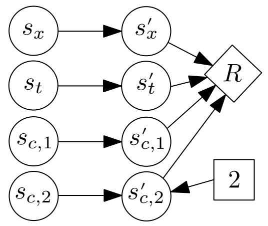

Figure 3: DBN defining the dependencies between state variables and rewards for 2 agents.

available renewable energy at time k in scenario s_x. If there are 8 units renewable energy are available, this results in reward $-c(2)$, since two units have to be consumed from the grid. The dependencies between the relevant factored state variables and the reward are shown in Figure 3, for a task scheduling problem with two agents. Recall that within one real-world timestep, n actions are executed before the state variable s_t is incremented. When the second agent decides to run a task, it sets its decision in the corresponding state variable, indicated by the square labeled 2 and the arrow to $s'_{c,2}$. The reward is determined by the scenario, the current timestep, and decisions made by other agents within the current timesteps. In the example in Figure 3, the decision of agent 1 is represented by $s'_{c,1}$, and this information is required to compute the remaining renewable supply to determine the reward of agent 2. With a joint action space we would not need to have separate state variables to keep track of decisions made by agents, but with our fixed-size action space it is required because rewards depend on previously executed actions.

Our state space has some special characteristics that we can exploit to plan more efficiently. All state variables, except scenario variable s_x, are fully observable and behave as a deterministic state machine where any transition probability is either 0 or 1. These fully observable variables represent the states of the tasks. The variable s_x represents the scenario and cannot be fully observed, but this state variable does never change and therefore its transitions are not dependent on other state variables. Instead of maintaining a belief over the entire state space, we infer a belief over state variable s_x, and together with the known state of the scheduler (i.e., the remaining state variables) it defines a belief state.

Our POMDP state representation can also easily be converted to a regular MDP, by discarding the scenario variable s_x and introducing an additional factored state variable that represents the available renewable energy at time s_t. In contrast to a scenario, this state variable represents the available units at an individual timestep, and its transitions can be defined by a Markov chain. In our experiments we compare both formulations of the problem.

4.3 Task Scheduling using POMCP

To create an online task scheduling algorithm, we use an adapted version of POMCP [16], which is an online planning algorithm that relies on Monte-Carlo tree search to decide which action to execute. POMCP has shown to be able to deal with POMDPs having a large state space, and does not require full enumeration of the state space. This is relevant because our POMDP formulation may grow very large

```
input : set of tasks $J$, scenario set $X$, threshold $\rho$
1  $q_{1,0} \leftarrow \emptyset$
2  $s \leftarrow$ initial scheduler state
3  $D \leftarrow \emptyset$
4  for $t = 1, \ldots, T$ do
5      $o \leftarrow$ renewable units observed at time $t$
6      $q_{1,t} \leftarrow q_{1,t-1} \cup \{o\}$
7      $b \leftarrow \text{BELIEF}(q_{1,t}, X, \rho)$
8      for $i = 1, \ldots, n$ do
9          $a \leftarrow \text{POMCP}(s, b, X)$
10         if $a = \text{RUN}$ and $j_i \notin D$ then
11             start agent $i$ at time $t$
12             $D \leftarrow D \cup \{j_i\}$
13         end
14         $s \leftarrow$ state obtained after executing $a$ in state $s$
15     end
16 end
```

Algorithm 4: Online Scenario POMCP.

depending on the time horizon and the number of agents. Additionally, it does not perform a standard belief state update using Bayes' rule.

The high-level description of our algorithm is shown in Algorithm 4. The algorithm starts with defining an initial empty realization, representing that no prior knowledge regarding renewable energy is available. On line 2 the initial state of the scheduler is defined, which consists of all POMDP state variables, except scenario variable s_x. At each timestep, the algorithm observes the number of renewable energy units available and updates the realization (line 5 and 6). Then a belief over scenarios is computed using Algorithm 3. For each agent, the algorithm performs a new POMCP search starting from the current scheduler state s and scenario belief b to decide whether tasks corresponding to the agents should start or not. After choosing an action, the new scheduler state is computed on line 14, which is possible because these state variables are fully observable and their transitions are deterministic (see Section 4.2). Notice that we do not have to implement the rotating token explicitly, because the state transitions ensure that the corresponding state variable is updated correctly.

We use the POMCP algorithm [16] with three modifications. In the simulate procedure we use an ε-greedy action selection heuristic, where the probability to select randomly decreases over time. The reason is that an ε-greedy action selection heuristic gives more stable performance in this domain in comparison to UCB [2]. Instead of performing a random rollout when the search leaves the existing tree, we compute an offline greedy schedule for the remaining timesteps using Algorithm 1, consistent with decisions made previously, and we compute its cost. For each agent a new POMCP search tree is created, instead of pruning the search tree after executing an action and making an observation.

5. EXPERIMENTS

We conducted several experiments to evaluate our algorithm and to compare its performance with the algorithms from Section 3. We also compare the performance with an offline optimal algorithm which assumes that there is no uncertainty and future available renewable power is known. As a concrete example of renewable energy integration with uncertainty, we have chosen wind power and we use data from a real wind farm. First we introduce the setup of our experiments and the configuration of the algorithms involved, which also shows how a scenario set can be built in a more realistic setting.

5.1 Wind Scenarios from Real Data

As mentioned before, in our experiments we demonstrate our scenario-based task scheduling algorithm using scenarios derived from a real wind farm. We obtained historical hourly wind data from the Sotavento wind farm located in Galicia, Spain for 1708 consecutive days.[1] For each sequence of 24 hours, we define a scenario $x = (x_1, \ldots, x_{24})$, where each x_i corresponds to the wind speed measured during hour i in meters per second. This yields 40969 scenarios in total, where each scenario consists of 24 hours. We round the wind speed values to the nearest integer, to discretize the observations. The generated power $Z(x,t)$ at time t in scenario x can be derived using a sigmoid power curve:

$$Z(x,t) = C \cdot (1 + e^{6 - \frac{2}{3}x_t})^{-1},$$

where C is a variable to define the capacity of the generator [14]. For each task scheduling instance, we choose a scalar C such that $Z(x,t) = \sum_{i=1}^{n} l_i \cdot p_i$, which ensures that the total demand equals the available renewable supply during the day. The same approach to model uncertain supply is used by Ströhle et al. [19].

5.2 Task Scheduling Instances

In each experiment we evaluate task scheduling algorithms on 200 instances. Each instance consists of a set containing 6 tasks, $J = \{j_1, j_2, \ldots, j_6\}$, which implies that the tasks that have to be scheduled by 6 agents. We assign a duration l_i between 3 and 7 to each task j_i, and a release time r_i between 8 and 12, both sampled uniformly at random. The release times represent that tasks are released between 8AM and noon during the day. The deadline d_i is set in such a way that tasks have finished by the end of the day. The power demand p_i equals 10 for each task, so a task requires 10 units at each timestep that the task is running. The cost of consuming one unit from the grid is assumed to be 1 (i.e., $c(u) = 1 \; \forall u \in \mathbb{N}$). To define the renewable supply that is available during the day in each experiment, we sample a realization $q_{1,24}$ from the scenario set. When evaluating offline algorithms, there is no uncertainty and the realization is assumed to be known throughout the day. In our online algorithms, however, the realization is revealed online during the day. Days in which the realization is relatively flat contain limited uncertainty, so we selected realizations where the renewable supply from time 1 to 6 and from time 13 to 18 is higher than the supply during the remaining hours. This guarantees that in any instance, the renewable supply is unstable and varies during the day.

5.3 Algorithm Configurations

Several algorithms are involved in our evaluation for comparison. We use mixed-integer programming to compute offline schedules, and we run an MDP planner, our scenario-based POMCP algorithm and consensus. In this section we briefly describe their setup and parameters.

[1]Consult `www.sotaventogalicia.com` for more information.

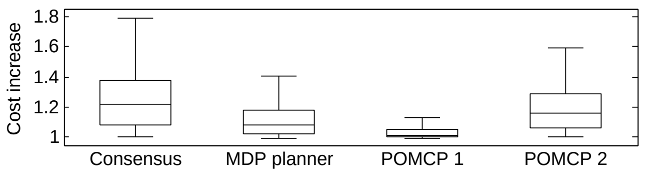

Figure 4: Cost increase for each algorithm, without outliers.

	Consensus	MDP	POMCP 1	POMCP 2
Mean	1.33	1.15	1.05	1.23
Std	0.44	0.21	0.12	0.25
Max	4.22	2.55	2.07	2.84

Table 1: Experiment statistics.

To compute offline schedules, where supply is known and certain, we use Gurobi[2] with a 1 percent MIP gap. The resulting schedules are useful to compare the performance of online algorithms with offline schedules without uncertainty, because the cost of an offline schedule is a lower bound on the cost of any online schedule for the same task scheduling instance.

Unless stated otherwise, the scenario-based POMCP algorithm runs 200 iterations, and follows an ε-greedy exploration strategy. In the first 100 iterations, the probability to select random actions decreases linearly from 1 to 0, and the last 100 iterations are fully greedy. To infer the scenario belief, we use Algorithm 3 with a threshold parameter ρ equal to 10.

We also implemented an MDP planner with the same state representation, but instead of having scenarios, we use a Markov chain to model the renewable supply (see Section 4.2 for details). To find actions with our MDP model, we apply 5000 search iterations of POMCP.

The consensus algorithm [19] has been implemented as shown in Algorithm 2, and we use a Hidden Markov Model to compute the likelihood of scenarios. We use the Baum-Welch algorithm [3] to learn a Hidden Markov Model with 10 hidden states from the Sotavento wind data, and we run the forward-backward algorithm to calculate observation probabilities.[3] For more details about Hidden Markov Models to determine the likelihood of scenarios, we refer to the paper by Ströhle et al. [19].

5.4 Scheduling Known Realizations

We ran the algorithms on 200 task scheduling instances and Figure 4 shows their performance in comparison to offline schedules. For each instance, we computed an offline schedule and we computed the cost increase of the resulting online schedules relative to the cost of the offline schedule. In the figure, the offline cost is represented by 1, and the boxplots show the distributions of the increased cost for each algorithm, relative to offline. For example, a cost increase of 1.2 represents that the cost of an online schedule is 20 percent higher than the cost of the offline schedule for the same instance. For readability reasons we removed the outliers from the figure, and we included additional statistics in Table 1. The statistics for the scenario POMCP algorithm are shown in the columns labeled POMCP 1. We can conclude that both MDP planning and the scenario-based POMCP planner outperform consensus on this set of instances with high supply uncertainty.

In this experiment, the realization $q_{1,T}$ is also one of the 40969 scenarios in the set X, so after having received several observations, Algorithm 3 automatically identifies the right scenario in X. This shows that if the algorithm encounters a known realization (i.e., $q_{1,T} \in X$), it performs better than both consensus and a standard MDP planner.

5.5 Scheduling Unseen Realizations

In our first experiment we concluded that the scenario-based planner performs well for realizations that already exist in the scenario set. If the scenario set is accurate and representative, it will rarely happen that new realizations are encountered. However, it is also interesting to study the performance on unseen realizations, that have not been encountered before. We repeated the experiment above, except that we excluded the realization from the scenario set X for each run of the algorithm. This situation represents that a new realization is encountered that does not exist in X. We also increased the number of POMCP search iterations to 500. The results are shown in Figure 4 and Table 1, in the columns labeled POMCP 2. We conclude that our scenario-based POMCP planner still performs well if it encounters new realizations. The algorithm still performs better than consensus, and performs slightly worse than the MDP planner. In this experiment the belief over scenarios may not always be an accurate representation of future supply, which explains that a naïve MDP formulation may perform better on some task scheduling instances. However, in practice it is unlikely that realizations are always new, and it can be expected that the performance of the POMCP planner improves if it encounters known scenarios more often, which we study in the next section.

5.6 Known and Unseen Realizations

As explained above, in practice it is unlikely that a realization is always new. Therefore, we ran the scenario POMCP planner on the same set of instances, but we excluded the realization from the scenario set with a certain probability. This represents a situation in which a realization is sometimes known (i.e., $q_{1,T} \in X$), and in other cases a new realization is encountered ($q_{1,T} \notin X$). The results of the experiment are shown in Figure 5 and Table 2, where the probability corresponds to the probability to exclude the actual realization from the set X, and the distributions represent the cost increase relative to offline scheduling without uncertainty. From the results in Table 1 and Table 2 we conclude that the scenario POMCP planner always performs better than consensus, and if it becomes more likely to encounter known realizations, then performance increases.

5.7 Discussion

To summarize, we briefly reformulate the important outcomes and conclusions from our experiments. Our main findings can be described as follows. If a known scenario is encountered (i.e., $q_{1,T} \in X$), then our scenario-based algorithm recognizes this existing scenario and performs better

[2]Further information about Gurobi Optimizer is available on `www.gurobi.com`.

[3]For the HMM implementations we use the Jahmm library, which can be found here: `code.google.com/p/jahmm`.

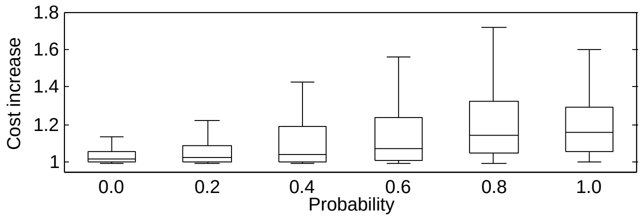

Figure 5: Cost increase for increasing probability to exclude the realization from X, without outliers.

Probability	0.0	0.2	0.4	0.6	0.8	1.0
Mean	1.05	1.10	1.17	1.20	1.27	1.23
Std	0.12	0.21	0.35	0.37	0.38	0.25
Max	2.07	2.77	4.19	4.19	4.19	2.84

Table 2: Experiment statistics for increasing probability to exclude the realization from X.

than the standard MDP planner and consensus. If the realization has never been encountered before (i.e., $q_{1,T} \notin X$), then our scenario-based algorithm still performs better than consensus, but performs slightly worse than the MDP planner. If the probability to encounter known realizations increases, the performance of the scenario-based algorithm, in terms of grid power cost, becomes better. Our experiments show that the scenario representation is a valuable state representation, which turns out to be useful if an accurate scenario set exists. As explained in the setup of the experiments, it is expected that an accurate scenario set X can be built using historical data (e.g., wind data). If a large and accurate scenario set exists, then it pays off to use beliefs over long-term scenarios.

6. RELATED WORK

Most notably, our work relates to the results from Ströhle et al. [19]. The authors present a consensus algorithm to match uncertain demand to uncertain supply in an online setting with multiple agents. The consensus algorithm can directly be applied to our problem, and our algorithm has shown to outperform consensus in case of high supply uncertainty. The role of consensus in online stochastic scheduling in general is discussed in work by Bent and Hentenryck [4]. Similar to our work, there is a notion of scenarios, but we implemented such scenarios in POMDP-based decision making which, to the best of our knowledge, has not been addressed in existing work. The authors identify the need to generalize consensus to multiple machines, which is addressed by Ströhle et al. [19].

Subramanian et al. [20] discuss online scheduling of deferrable loads with supply uncertainty, and the authors propose a method for predictive control of tasks. In contrast to our work, the predictions are a single valued prediction of the total future renewable energy availability, whereas our method and Ströhle et al. [19] use multi-valued predictions (i.e., scenarios) corresponding to multiple timesteps. Neely et al. [10] describe a method for scheduling flexible deferrable loads, which also tries to minimize the cost of consuming grid power. In addition to minimizing grid power consumption, the paper also presents an optimization problem where renewable sources dynamically set a price level of their service, which we did not study in our work.

Exploiting factored structures in the POMCP algorithm has been studied by Amato and Oliehoek [1]. They propose a variant of POMCP that does not assume a factored model, but it uses factored value functions, which reduces the number of joint actions and joint histories in the multiagent setting. Value functions are factored based on local effects of actions, which is a technique that can also be combined with our multiagent model to schedule tasks.

7. CONCLUSIONS AND FUTURE WORK

In this paper we presented a new online task scheduling algorithm to match demand to uncertain supply. This is especially relevant in the context of power systems, where renewable energy has to be integrated in the future smart grid. We consider deferrable loads as tasks, and assume that there are two energy sources: conventional generation and uncertain renewable energy. Tasks should be scheduled in such a way that renewable energy is used as much as possible. We defined the problem as a multiagent sequential decision making problem under uncertainty, and we use planning to make scheduling decisions. Our algorithm relates closely to the Partially Observable Markov Decision Process formalism. Rather than maintaining a belief over states, we infer a belief over scenarios defining the supply, and the belief is recomputed when new information becomes available. We conducted an evaluation study with uncertain renewable wind power, using data from a real wind farm. In the experiments we found that our algorithm outperforms an existing consensus algorithm in case of high supply uncertainty.

In future work we aim to study how a scenario belief representation can be implemented in a rolling horizon fashion, where a scenario predicts k future timesteps and a belief is constructed based on the last k observations. In our current work we use a centralized approach, but we also want to create a cooperative decentralized approach that corresponds to tightly connected network structures of the electricity grid. Another interesting direction for future research is generalizing scenarios. A scenario representation relates previous state observations to future state observations and we will study whether this can be exploited in other planning domains.

Acknowledgements

The work presented in this paper is funded by the Netherlands Organisation for Scientific Research (NWO), as part of the Uncertainty Reduction in Smart Energy Systems program. We would like to thank Mathijs de Weerdt for sharing the Sotavento wind farm dataset.

REFERENCES

[1] C. Amato and F. A. Oliehoek. Scalable Planning and Learning for Multiagent POMDPs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2015.

[2] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, 47(2-3):235–256, 2002.

[3] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A Maximization Technique Occurring in the Statistical

Analysis of Probabilistic Functions of Markov Chains. *Annals of Mathematical Statistics*, 41:164–171, 1970.

[4] R. Bent and P. V. Hentenryck. The Value of Consensus in Online Stochastic Scheduling. In *Proceedings of the International Conference on Automated Planning and Scheduling*, pages 219–226, 2004.

[5] G. Giebel, R. Brownsword, G. Kariniotakis, M. Denhard, and C. Draxl. The State-Of-The-Art in Short-Term Prediction of Wind Power. Technical report, ANEMOS. plus, 2011.

[6] L. Kaelbling, M. Littman, and A. Cassandra. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, 101(1):99–134, 1998.

[7] I. Koutsopoulos and L. Tassiulas. Challenges in Demand Load Control for the Smart Grid. *IEEE Network*, 25(5):16–21, 2011.

[8] J. Lopes, N. Hatziargyriou, J. Mutale, P. Djapic, and N. Jenkins. Integrating distributed generation into electric power systems: A review of drivers, challenges and opportunities. *Electric Power Systems Research*, 77:1189–1203, 2007.

[9] P. S. Moura and A. T. de Almeida. The role of demand-side management in the grid integration of wind power. *Applied Energy*, 87(8):2581–2588, 2010.

[10] M. J. Neely, A. S. Tehrani, and A. G. Dimakis. Efficient Algorithms for Renewable Energy Allocation to Delay Tolerant Consumers. In *Proceedings of the IEEE International Conference on Smart Grid Communications*, pages 549–554, 2010.

[11] C. Papadimitriou and J. N. Tsitsiklis. The complexity of markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[12] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for pomdps. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1025–1030, 2003.

[13] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1st edition, 1994.

[14] V. Robu, R. Kota, G. Chalkiadakis, A. Rogers, and N. Jennings. Cooperative Virtual Power Plant Formation Using Scoring Rules. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 370–376, 2012.

[15] J. Scharpff, M. T. J. Spaan, L. Volker, and M. de Weerdt. Planning under Uncertainty for Coordinating Infrastructural Maintenance. In *Proceedings of the International Conference on Automated Planning and Scheduling*, pages 425–433, 2013.

[16] D. Silver and J. Veness. Monte-Carlo Planning in Large POMDPs. In *Advances in Neural Information Processing Systems*, pages 2164–2172, 2010.

[17] M. T. J. Spaan. Partially Observable Markov Decision Processes. In M. Wiering and M. van Otterlo, editors, *Reinforcement Learning: State-of-the-Art*, pages 387–414. Springer Verlag, 2012.

[18] M. T. J. Spaan and N. Vlassis. Perseus: Randomized Point-based Value Iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220, 2005.

[19] P. Ströhle, E. Gerding, M. de Weerdt, S. Stein, and V. Robu. 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*, pages 437–444, 2014.

[20] A. Subramanian, M. Garcia, A. Dominguez-Garcia, D. Callaway, K. Poolla, and P. Varaiya. Real-time Scheduling of Deferrable Electric Loads. In *Proceedings of the American Control Conference*, pages 3643–3650, 2012.