Planning under Uncertainty with Weighted State Scenarios

Erwin Walraven
Delft University of Technology
Mekelweg 4, 2628 CD
Delft, The Netherlands

Matthijs T. J. Spaan
Delft University of Technology
Mekelweg 4, 2628 CD
Delft, The Netherlands

Abstract

In many planning domains external factors are hard to model using a compact Markovian state. However, long-term dependencies between consecutive states of an environment might exist, which can be exploited during planning. In this paper we propose a *scenario* representation which enables agents to reason about sequences of future states. We show how weights can be assigned to scenarios, representing the likelihood that scenarios predict future states. Furthermore, we present a model based on a Partially Observable Markov Decision Process (POMDP) to reason about state scenarios during planning. In experiments we show how scenarios and our POMDP model can be used in the context of smart grids and stock markets, and we show that our approach outperforms other methods for decision making in these domains.

1 INTRODUCTION

The Markov Decision Process (MDP) formalism is a mathematical framework for modeling agents interacting with their environment (Puterman, 1994). In many real-world planning domains, however, external factors can be difficult to predict, which makes it hard to obtain a Markovian model with the right state features and an appropriate level of detail (Witwicki et al., 2013). In such domains, it is hard to estimate probabilities for the occurrence of uncertain events, and therefore decision making can be a challenging task.

An example of a hard-to-model external factor is renewable energy supply such as generation of wind power. Research has shown that the most severe problems in electricity grids occur during peak-load hours when energy demand is high and wind power generation is interrupted (Moura and De Almeida, 2010), because then the supply of renewable electricity may not be sufficient to satisfy the demand of consumers. A potential solution is exploiting the flexibility of the loads of consumers, such that they can be supplied during off-peak hours. This solution requires reasoning about future wind speed, but many external factors influencing wind make it hard to define a compact Markovian state for wind. Additionally, methods to predict short-term wind power are affected by errors and may be inaccurate (Giebel et al., 2011).

In order to accommodate planning in domains with events that are difficult to predict and hard to model, we propose a framework that enables agents to reason about future states. This approach is based on the observation that there can be long-term dependencies between states, which can be exploited during planning, rather than explicitly defining a state transition model with appropriate features. In our framework, such long-term dependencies are modeled by scenarios, which are sequences of states. An advantage of using scenarios is illustrated in Figure 1, in which we compare wind predictions generated by a second-order Markov chain and actual wind scenarios that have been observed in practice. The lines in the figure visualize the 5th percentile, mean and 95th percentile of these predictions, and show us that scenarios provide information that is not sufficiently modeled by a second-order Markov chain.

In our work we assign weights to scenarios, corresponding to the likelihood that a scenario predicts future states accurately, and we use the Partially Observable Markov Decision Process framework (Kaelbling et al., 1998) to reason about scenarios during planning. We demonstrate the proposed Scenario-POMDP model in two domains. Besides wind scenarios in smart grids, we show how a Scenario-POMDP can be applied to financial stock markets. This domain has also been subject of study in the artificial intelligence community, since stock price is hard to model and depends on many external factors (Hassan and Nath, 2005). An experimental evaluation shows that our method outperforms other methods for decision making in both domains, indicating that scenarios are a valuable representation to model uncertainty regarding the future.

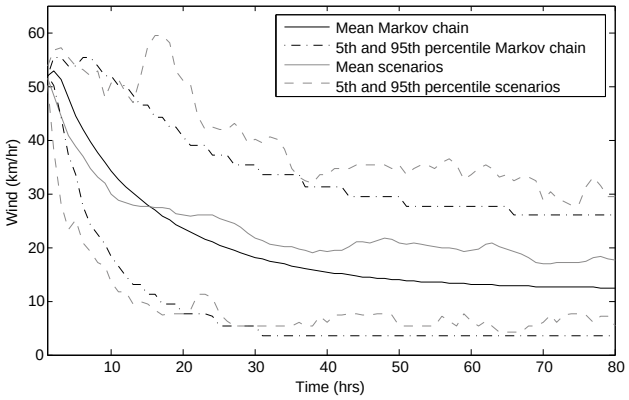

Figure 1: Comparison between Markov chain wind predictions and realistic wind scenarios starting from 51 km/hr.

The structure of this paper is as follows. Section 2 introduces planning under uncertainty. In Section 3 we introduce scenarios and related concepts. In Sections 4 and 5 we show how planning with scenarios can be applied to smart grids and stock markets. In the remaining sections we discuss related work, conclusions and future work. In the supplementary material we provide problem formulations and additional information regarding the problem domains.

2 PLANNING UNDER UNCERTAINTY

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 (Kaelbling et al., 1998).

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 (Puterman, 1994), 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

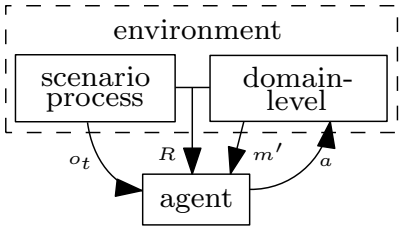

Figure 2: An agent executing action a, which causes a state transition to m', and the agent receives state observation o_t from the scenario process at time t and receives reward R.

in belief state b can be determined using Bayes' rule:

$$b_a^o(s') = \frac{P(o|a,s')}{P(o|a,b)} \sum_{s \in S} P(s'|s,a)b(s)$$

where $P(o|a,b) = \sum_{s' \in S} P(o|a,s') \sum_{s \in S} P(s'|s,a)b(s)$. The state space S, action space A and observation space O are assumed to be finite in this paper.

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. The optimal value function $V^*(b)$ is defined as:

$$V^*(b) = \max_{a \in A} \left[\sum_{s \in S} R(s,a)b(s) + \gamma \sum_{o \in O} P(o|a,b)V^*(b_a^o) \right]$$

where b_a^o is defined by Bayes' rule, and γ is a discount factor satisfying $0 \leq \gamma < 1$. Computing exact solutions to POMDPs is known to be intractable (Papadimitriou and Tsitsiklis, 1987), but many approximate methods exist based on point-based value iteration (Pineau et al., 2003; Spaan and Vlassis, 2005). In this paper we use POMCP (Silver and Veness, 2010), an online Monte-Carlo planning algorithm that is capable of dealing with a large number of states. We assume the planning horizon to be finite.

3 PLANNING WITH SCENARIOS

In this section we propose a scenario representation for planning under uncertainty. First we introduce the notion of scenarios and we explain how scenarios can be weighted based on previous observations. We also present a general POMDP model to reason about scenarios and future states during planning.

3.1 SCENARIOS AND WEIGHTS

We assume that an agent interacts with an environment as shown in Figure 2. The environment consists of a process

input : observation sequence $o_{1,t}$, scenario set X, threshold ρ
output: weights w

$X' \leftarrow \emptyset$
$d \leftarrow 0$
while $|X'| < \rho$ **do**
  $X' \leftarrow \{x \in X : W(x, o_{1,t}) \leq d\}$
  $d \leftarrow d + 1$
end
foreach $x \in X$ **do**
  if $x \in X'$ **then**
    $w_x \leftarrow 1 \ / \ (\varepsilon + W(x, o_{1,t}))$
  else
    $w_x \leftarrow 0$
  end
end
$w^* \leftarrow \sum_{x \in X} w_x$
foreach $x \in X$ **do**
  $w_x \leftarrow w_x \ / \ w^*$
end

Algorithm 1: WEIGHTS.

for which the domain-level state changes to m' after executing an action. For simplicity, in this paper we assume that the state of this process is observable, but our approach is not limited to this assumption. Additionally, there is another process for which a compact Markovian model does not exist, called the scenario process. We assume that an agent observes a numerical-valued state o_t of this process at time t, which we call a state observation, but there is no model available defining the state transitions. The actions executed by the agent do not influence the state transitions of the scenario process. We assume that the rewards received by the agent depend on the state m, as well as on the state of the scenario process, which means that the agent has to account for future states to optimize the long-term reward.

In order to be able to reason about future states, we propose a scenario representation below. A scenario is a sequence of states of the scenario process, and implicitly models the dependencies between multiple consecutive states.

Definition 1. *(Scenario). A scenario $x = (x_1, \ldots, x_T)$ is a sequence of states of the scenario process for T consecutive timesteps, where x_t is the state at time t. The sequence containing the first t states of scenario x is denoted by $x_{1,t}$.*

States of the scenario process are assumed to be directly observable, represented by a sequence $o_{1,t} = (o_1, o_2, \ldots, o_t)$, containing state observations from the first t timesteps. The sequence of state observations can be compared to a scenario, by comparing the individual state observations with states in the scenario. We illustrate this with a small example for the scenario $x = (x_1, x_2, x_3, x_4) = (8, 5, 3, 2)$. Suppose that the state observations are defined by the sequence $(o_1, o_2, o_3) = (8, 5, 4)$, then the first and second state observation are identical to the first two states defined by scenario x, and the third state observation is different.

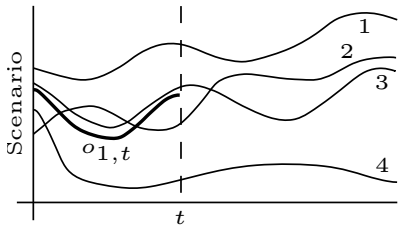

Figure 3: Scenarios and state observations until time t.

Weights can be assigned to scenarios, representing the likelihood that a scenario perfectly predicts the states that will be observed in the future. To reason about future states, we assume that a large scenario set X is given, containing sequences of states that can be observed in T consecutive timesteps. If a sequence of states $o_{1,t}$ is observed until time t, then the sequence can be used to assign weights to the scenarios in X. In Sections 4 and 5 we discuss how such a scenario set can be obtained in realistic domains.

An informal visual representation of scenarios is shown in Figure 3. It shows four scenarios, labeled 1 to 4, and the state observation sequence $o_{1,t}$. As can be seen in the figure, the state observation sequence does not correspond to any scenario until time t, but it is very similar to scenario 3. If X is an accurate set of scenarios, then it is probable that scenario 3 predicts future states. Therefore, the weight assigned to this scenario should be high in comparison to the weights assigned to other scenarios.

The weights assigned to scenarios are inversely proportional to the distance between scenarios and the state observation sequence. The distance between state observation sequence $o_{1,t} = (o_1, o_2, \ldots, o_t)$ and the first t states of a scenario $x \in X$ can be measured by computing the sum of squared errors. The function W below computes this distance for a given scenario $x \in X$ and state observation sequence $o_{1,t}$:

$$W(x, o_{1,t}) = \sum_{i=1}^{t} (o_i - x_i)^2.$$

We select scenarios up to a certain distance from the state sequence until time t, and we assign weights to the scenarios such that they sum to 1. The algorithm that we use to assign weights is shown in Algorithm 1. It selects a subset of scenarios X' containing at least ρ scenarios similar to $o_{1,t}$, based on the sum of squared errors. This step ensures that there is a sufficient number of scenarios with non-zero weight, to prevent that the future is predicted by only one or very few scenarios. A probability distribution is defined over the scenarios in X', in which the probabilities are inversely proportional to the computed distance. A nor-

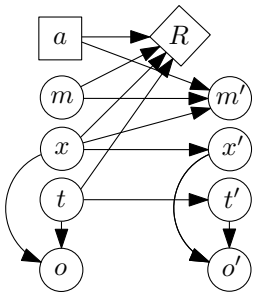

Figure 4: General dynamic Bayesian network of the Scenario-POMDP model.

malization step is performed to ensure that the sum of the probabilities equals 1. The parameter ρ can be adjusted to lower-bound the number of scenarios with non-zero probability.

3.2 SCENARIO-POMDP MODEL

In this section we present a POMDP model, which we can use to model any planning problem with the type of agent-environment interaction outlined in the previous section. The model can be used to reason about future states of a scenario process during planning, such that agents can reason about future states to optimize the long-term reward.

We present a POMDP model with factored states, as shown in the dynamic Bayesian network in Figure 4. Before explaining the Scenario-POMDP model in detail, we formalize it below.

Definition 2. *(Scenario-POMDP). A Scenario-POMDP is a POMDP in which each state $s \in S$ can be factored into a tuple $s = (m, x, t)$, where m is the domain-level state of the environment, $x \in X$ is a scenario and t is a time index. A Scenario-POMDP has the following properties:*

1. *The scenario state variable x is partially observable, and state variables t and m are observable*[1]*. In state $s = (m, x, t)$, observation x_t is observed with probability 1, determined by x and t.*
2. *The transitions of m are determined by a predefined transition function. State variable t is always incremented by 1 after executing an action. The scenario state variable x is fixed.*
3. *The reward received in state $s = (m, x, t)$ depends on x_t, defined by variable x and variable t, and depends on domain-level state m and action a.*

A Scenario-POMDP models problems in which the interaction with the environment is represented by an MDP defining the domain-level actions and states, but the rewards also depend on the state of the scenario process. Actions only affect the domain-level state m, and we assume that they do not influence the state transitions of the scenario process.

[1]The Scenario-POMDP model can also be used if m is partially observable, which requires factored observations.

input: initial domain-level state m_0, horizon T, scenario set X, threshold ρ

$m \leftarrow m_0$
for $t = 1, \ldots, T$ **do**
  $o_t \leftarrow$ state of scenario process
  $o_{1,t} \leftarrow (o_1, \ldots, o_t)$
  $w \leftarrow \text{WEIGHTS}(o_{1,t}, X, \rho)$
  $a \leftarrow \text{POMCP}(m, w, t)$
  execute action a
  $m \leftarrow$ state obtained after executing a in state m
end

Algorithm 2: Scenario-POMCP.

The state variable m represents the domain-level state of the environment, and the actual definition of m is dependent on the problem domain, as we will show later. The state variable x represents a scenario, where the scenario is a sequence of states as introduced in Definition 1. A scenario defines the state observations to be made in future timesteps, and therefore the scenario x is considered to be partially observable. A scenario cannot be fully observable, because this would imply that there is prior knowledge regarding future states of the scenario process. In this paper x denotes both a scenario and scenario state variable, but this makes the explanations more intuitive and clear. The state variable t is a variable representing the current timestep, and is fully observable. The Scenario-POMDP model can be considered as a MOMDP, which is a subclass of POMDPs for problems with mixed observability (Ong et al., 2010).

The observations in a Scenario-POMDP are exclusively determined by the factored state variables x and t. For scenario variable x and time variable t, the observation received by the agent is the observation at time t in scenario x. Suppose that the scenario x is $(6, 9, 12)$ and the time state variable equals 2, then the agent will receive observation 9. This explains how the observations in Figure 4 depend on the scenario variable x and time variable t.

An action a, represented by the square in the figure, only affects the domain-level state m and does not influence the scenario x. For each action, the reward R is dependent on the domain-level state of the environment, as well as the observed state of the scenario process, defined by scenario state variable x and time variable t.

3.3 PLANNING FOR SCENARIO-POMDPs

We present a general planning algorithm, called Scenario-POMCP, for planning problems that can be formulated as a Scenario-POMDP. The description of the algorithm is

Table 1: Task Scheduling State Variables.

VARIABLE	DESCRIPTION
m_s^i	state of task i, for $i = 1, \ldots, n$
$m_a \in \{1, \ldots, n\}$	agent owning the token
$x \in X$	scenario
$t \in \{1, \ldots, T\}$	time

shown in Algorithm 2. The initial domain-level state m_0, time horizon T, scenario set X and threshold ρ are given as input. The state observations o_t are used to define $o_{1,t}$ and Algorithm 1 is used as a subroutine to assign weights to scenarios in X. The POMCP algorithm (Silver and Veness, 2010) is used as planning algorithm, which is an online planning algorithm for POMDPs based on Monte-Carlo tree search. This algorithm receives m, w and t as input, and samples scenarios from X with a probability proportional to their weight in the vector w. We use POMCP because this algorithm can be adapted to sample scenarios based on weights, rather than sampling states from a belief state, and the algorithm is able to deal with a large number of states. The latter is relevant since the number of states of a Scenario-POMDP may grow very large. Eventually our algorithm executes the resulting action a before proceeding to the next timestep. At any timestep a new POMCP search tree is created, because there is no explicit link between the weights of consecutive timesteps. More implementation details are provided in the supplementary material.

4 SCENARIOS IN SMART GRIDS

In this section we formulate matching of demand with renewable supply in smart grids as a Scenario-POMDP, and we run simulations to compare the performance with other methods.

4.1 BACKGROUND

We consider n power demanding tasks, denoted by $J = \{j_1, \ldots, j_n\}$, 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, must be finished by the deadline and cannot be preempted. 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 $l_i \cdot p_i$.

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

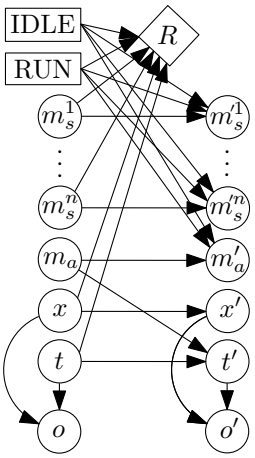

Figure 5: Dynamic Bayesian network of the Scenario-POMDP for task scheduling.

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

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

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 $D(S, t)$ of a schedule $S = (h_1, \ldots, h_n)$ at time t is defined as $D(S, t) = \sum_{i=1}^{n} I_S(i, t) \cdot p_i$, 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 $U_{S,x}$ for schedule S and fixed scenario $x = (x_1, x_2, \ldots, x_T)$ can be computed as follows:

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

The cost function $c(U_{S,x})$ is used as an objective function to be minimized, in order to match demand and the renewable supply defined by the scenario.

4.2 SCENARIO-POMDP FORMULATION

We formulate the problem to start and defer tasks as a planning problem, in which we assume that an agent is associated with each task. A scenario $x = (x_1, \ldots, x_{24})$ is a sequence of wind state observations, where each x_i corresponds to the wind speed measured during hour i. The time

horizon T is equal to 24. Available renewable supply generated by wind is observed during the day, and the cost of running a task depends on the supply from wind and the decisions made for other tasks.

The problem is formulated as a Scenario-POMDP, using the factored state variables in Table 1. The state variables $m_{s,1}, \ldots, m_{s,n}$ define the states of the individual tasks, which encode properties such as release time, duration and deadline. The state variable m_a represents which agent is allowed to make a decision, which is used to reduce the size of the action space. More details are provided in the supplement. The factored state description is also visualized as a dynamic Bayesian network in Figure 5, which shows the correspondence with the Scenario-POMDP model in Figure 4.

An agent is able to execute two different actions: RUN and IDLE, corresponding to either running a task at a certain timeslot or doing nothing. The rewards are equal to the cost of running a task, multiplied by -1, which leads to a higher penalty if costly conventional generation is used. The observations automatically follow from the Scenario-POMDP model, as explained in Section 3.

4.3 EXPERIMENTS

In the experiments we run simulations to compare the proposed Scenario-POMDP formulation with other methods. We obtained historical hourly wind data from the Sotavento wind farm located in Galicia, Spain for 1708 days in the period from October 2008 until May 2013.[2] We consider the dataset as a long vector defining wind for 1708 consecutive days of 24 hours each, in which wind is measured in km/hr. For each subsequence of 24 hours, we define a scenario $x = (x_1, \ldots, x_{24})$, which yields 40969 scenarios in total. In order to discretize the observation space, we round wind speed values to the nearest integer. 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 (Robu et al., 2012). 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.

We evaluate our planning algorithm on 200 task scheduling instances. Each instance consists of a set containing 6 tasks: $J = \{j_1, j_2, \ldots, j_6\}$. We assign a duration between 3 and 7 to each task j_i, and a release time between 8 and 12, both sampled uniformly at random. The release times represent that tasks are released between 8AM and noon. The deadline d_i is set after 24 hours, to ensure that tasks have finished by the end of the day. The power demand p_i equals 10 for each task, such that running tasks require 10 units at each timestep. The cost of consuming one unit from the grid is assumed to be 1. To define the renewable supply that is available during the day in each experiment, we sample an observation sequence from the scenario set. Days in which the renewable supply is relatively flat contain limited uncertainty. Therefore, we select scenarios 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 the renewable supply is unstable and varies during the day.

[2]Consult `www.sotaventogalicia.com` for details.

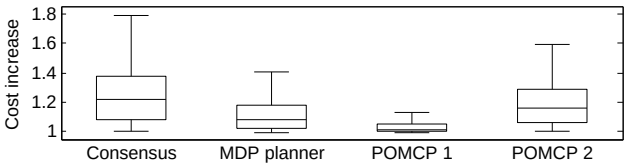

Figure 6: Performance comparison between planning with scenarios and other methods, without outliers.

Table 2: Statistics for Smart Grids Experiment 1.

	CONS.	**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

In our experiments we aim to compare the performance of Scenario-POMCP with the cost of optimal omniscient schedules. These assume that the supply throughout the day is known, which is a lower bound on the performance. We use mixed-integer programming with a 1 percent MIP gap to compute this for each instance. The scenario-based POMCP planner runs 200 iterations with an ε-greedy exploration strategy, in which the probability to select random actions decreases linearly from 1 to 0, and threshold ρ in the weight computation equals 10. Additionally, we run an MDP planner that is based on Monte-Carlo tree search, and we also compare with a consensus task scheduling algorithm (Ströhle et al., 2014). More implementation details, and additional information regarding the consensus algorithm, are provided in the supplement.

The results of our comparison are shown in Figure 6, in which we show the performance relative to the cost of the optimal omniscient schedules. The cost of the optimal omniscient schedules is represented by 1, and the distributions show the performance relative to this cost. For example, if the cost is 1.2, this means that the cost is 20 percent higher than the cost of an optimal omniscient schedule. For readability reasons outliers have been omitted in the figure, and therefore additional statistics are provided in Table 2.

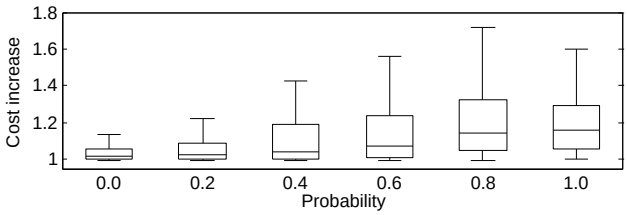

Figure 7: Cost increase for increasing probability to exclude the state observation sequence from X.

Table 3: Statistics for Smart Grids Experiment 2.

	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

From the results we can conclude that the MDP planner performs slightly better than the consensus planner. For Scenario-POMCP we ran the algorithm with two different configurations. The column labeled POMCP 1 represents the case in which the observed state scenario is already present in the scenario set X, and then it performs much better than other methods. The column labeled POMCP 2 shows the results for the experiment in which the observed state scenario is never present in the set X. In those cases the performance is slightly worse, but still competitive with other methods.

In practice it can be expected that accurate scenario sets can be obtained from large historical datasets, and it is unlikely that the observed state scenario is never present in X. Therefore, we did an additional experiment in which the observed state scenario is excluded from X with a certain probability. This means that the algorithm sometimes encounters known scenarios, and in other cases the observed scenario is new. The results are shown in Figure 7 and Table 3, from which we can conclude that the performance of Scenario-POMCP is better if it is more likely that the observed state sequence is already part of the set. Therefore, better performance can be obtained by having a scenario set that covers all possible sequences of states accurately.

Our experiments make clear that the Scenario-POMDP model can be used for matching demand and uncertain supply in the smart grids domain. Additional results can be found in previous work (Walraven and Spaan, 2015).

5 SCENARIOS IN OPTION TRADING

In this section we show how scenarios can be used in financial stock markets.

5.1 BACKGROUND

A popular type of financial option is the European call option. This option gives the holder the right, but not the obligation, to buy a share at a prescribed point in time for a prescribed price regardless of the stock price. A European call option is parameterized by a strike price E and expiry date H, giving the holder the right to pay E for a share at time H. The value of the European call option at time H is $\max(S(H) - E, 0)$, where $S(H)$ denotes the stock price at time H. If the strike price is lower than the stock price at expiry, the option holder can earn money by buying a share and selling it immediately on the market. If the strike price is higher, then the trader cannot gain anything. The value of a European call option can be determined using the Black-Scholes equation (Black and Scholes, 1973), for which a description is provided in the supplement. Reasoning about the future is necessary during trading, because if the stock price drops an option may become worthless.

5.2 SCENARIO-POMDP FORMULATION

We formulate buying and selling call options as a single-agent planning problem. A scenario $x = (x_1, \ldots, x_t)$ is defined as a sequence of stock price values, where x_i is the stock price observed at time i. We assume that there is an agent observing the market, and depending on the stock price it may decide to buy a call option, and if it owns a call option the agent may decide to sell the option to make profit. There is one type of call options the agent can buy. If the current stock price is j, the agent can buy a call option that expires after 10 days, with strike j.

The planning problem can be defined as a Scenario-POMDP, and we formulate the problem using the factored state variables in Table 4. The scenario variable x and time variable t are identical to the state variables x and t in a Scenario-POMDP, and they define the observations. The state variables m_o, m_e and m_t represent the current state of the option portfolio of the agent. The variable m_o can be either true (T) or false (F), representing whether the agent currently owns a call option or not. If the agent owns a call option, the strike price of the option is represented by state variable m_e, and state variable m_t represents the number of days until expiry. The factored variables m_o, m_e and m_t together define the state m of the option portfolio. The factored state description is also visualized as a dynamic Bayesian network in Figure 8, which shows the correspondence with the Scenario-POMDP model in Figure 4.

The agent can execute three different actions: BUY, SELL and NOOP. The action NOOP represents doing nothing, and the actions BUY and SELL correspond to buying and selling an option. The agent must sell once the option has expired. The rewards correspond to either paying a certain amount of money, or receiving a certain amount of money. The agent always pays the Black-Scholes value of the op-

Table 4: Option Trading State Variables.

VARIABLE	DESCRIPTION
$m_o \in \{\mathrm{T}, \mathrm{F}\}$	represents whether agent owns a call
$m_e \in \mathbb{N}$	strike price of the call
$m_t \in \{0, \ldots, 9\}$	time to expiry
$x \in X$	scenario
$t \in \{1, \ldots, T\}$	time

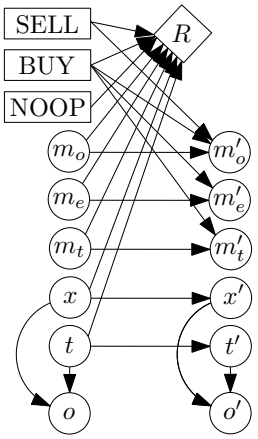

Figure 8: Dynamic Bayesian network of the Scenario-POMDP for option trading.

tion when buying an option, and receives the Black-Scholes value of the option when selling the option. The observations automatically follow from the Scenario-POMDP model, as explained in Section 3. A full description of the state transitions and rewards can be found in the supplement.

5.3 EXPERIMENTS

We did several experiments to evaluate the performance of the option trading agent on realistic data. To be able to reason about the stock price in the future, we obtained the historical daily close price values to build realistic scenario sets for shares in companies A and B. For company A, we use the data from January 2, 2001 to December 8, 2010, and for company B we use the data from September 27, 2000 to September 12, 2008. The stock price values are discretized, such that each price is a natural number. We enumerated subsequences of length 40 from the datasets to create a scenario set for each company.

To evaluate the performance, we simulate the stock market for each company for a period of at least 1000 days, with historical data that is more recent than the datasets we used to create the scenario set, which ensures that both datasets are distinct. For company A we simulate the stock

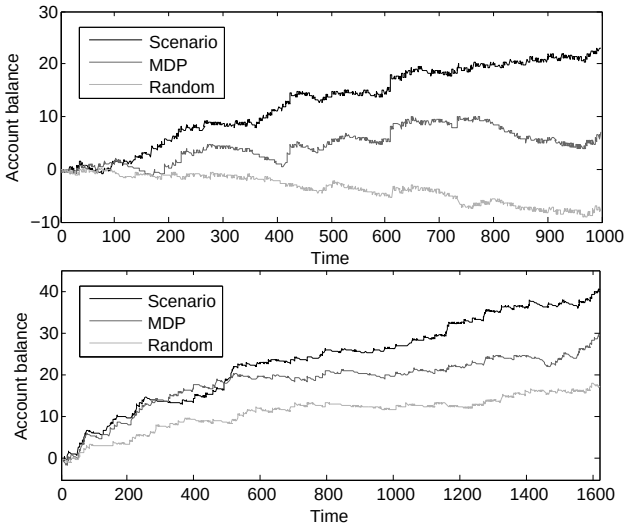

Figure 9: Account balance during simulations for company A (top) and company B (bottom).

price from December 9, 2010 to December 1, 2014. For company B we simulate the stock price from September 15, 2008 to February 20, 2015. We measured the average historical volatility using an Exponential Weighted Moving Average approach (Higham, 2008), and based on that we assume that the volatility of the market equals 0.4. The annual interest rate is assumed to be 0.03.

Our scenario-based agent uses Scenario-POMCP to select actions, which is implemented with a rolling time horizon. In our experiment the number of POMCP search iterations is set to 200, and the parameter ρ is equal to 10. POMCP uses UCB (Auer et al., 2002) as action selection heuristic with parameter 100 for company A and parameter 40 for company B.

In our experiments we aim to show that our method based on scenarios trades better than other methods. We can do this by keeping track of the account balance during the simulations, and we expect that an agent using the scenario planner earns more money than other methods. We also implemented an agent that is trading randomly, and an agent using an MDP formulation of the problem, which models the stock price as a Markov chain. The MDP-based agent uses Monte-Carlo tree search to select actions.

To compare the performance of the methods involved, we compare the trajectories defining the account balance during the simulations, as shown in Figure 9. For each method it shows the balance of the bank account for either 1000 or 1600 days. For each company, the scenario-based agent earns consistently more money in comparison to the other methods and, as expected, the random agent performs poorly. Based on our experiments we conclude that planning with scenarios has shown to perform well in this domain.

6 RELATED WORK

Ströhle et al. (2014) present a method to schedule tasks, where the uncertainty is also represented by weighted scenarios. Their method solves the problem for each scenario to reach consensus. A similarity with our work is that the method can be used to balance demand and supply, but our Scenario-POMDP formulation has been shown to outperform the consensus algorithm in case of high uncertainty. An important difference between the work by Ströhle et al. and this paper is that we define scheduling of tasks as a planning problem.

The problem to assign weights to expert opinions regarding the future is studied by Carvalho and Larson (2013). The method proposed is similar to our work because the similarity between opinions is measured by a distance function based on squared errors, which we also use to assign weights to scenarios, and our scenarios can also be considered as opinions regarding uncertain future events. In our work we always recompute weights associated with scenarios, whereas the work on opinion pools allows experts to update weights when new opinions become available.

Optimization using scenarios has been studied in the context of stochastic programming. Multi-stage stochastic programming problems can be considered as a subclass of Markov Decision Processes with a finite horizon (Defourny et al., 2011), and in this formalism the uncertainty is also represented by future scenarios, which is similar to our representation of state scenarios.

Exploiting factored structures in the POMCP algorithm has been studied by Amato and Oliehoek (2015). 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 multi-agent setting. A similarity is that we use a factored representation in each node of the POMCP search tree, which we can exploit to sample scenarios rather than states, but factored value functions have not been considered in our work.

7 CONCLUSIONS

In this paper we proposed a scenario-based approach to predict external factors that are difficult to model using a compact Markovian state. Scenarios represent sequences of states, and we have shown how a scenario can be weighted based on a sequence of states that occurred in the past, corresponding to the likelihood that a scenario perfectly predicts future states. In order to use the scenario representation in planning problems, we proposed a model called Scenario-POMDP, which enables agents to reason about future states during planning. To demonstrate the proposed model, we formulated matching of demand with renewable supply in smart grids as a Scenario-POMDP, and we have shown that our model can also be used to automatically trade financial options. In both cases our Scenario-POMDP model performs better than other methods for decision making in these domains.

In future work we aim to study metrics for computing the distance between scenarios in which states are not represented by a single numerical value. Another direction for further research is defining a belief update for scenarios based on Bayes' rule, to replace the Monte-Carlo backups in our planner. Moreover, in our current work we assume that the domain-level state of the environment is observable, but we also want to study planning with scenarios for problems in which this part of the environment is partially observable, which requires factored observations.

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 pointing out the dataset of the Sotavento wind farm.

References

Amato, C. and Oliehoek, F. A. (2015). Scalable Planning and Learning for Multiagent POMDPs. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 1995–2002.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time Analysis of the Multiarmed Bandit Problem. In: *Machine Learning* 47.2-3, pp. 235–256.

Black, F. and Scholes, M. (1973). The Pricing of Options and Corporate Liabilities. In: *The Journal of Political Economy* 81.3, pp. 637–654.

Carvalho, A. and Larson, K. (2013). A Consensual Linear Opinion Pool. In: *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 2518–2524.

Defourny, B., Ernst, D., and Wehenkel, L. (2011). Multi-stage Stochastic Programming: A Scenario Tree Based Approach to Planning under Uncertainty. In: *Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions*. Ed. by L. Sucar, E. Morales, and J. Hoey. Information Science Publishing.

Giebel, G, Brownsword, R., Kariniotakis, G., Denhard, M., and Draxl, C. (2011). *The State-Of-The-Art in Short-Term Prediction of Wind Power*. Deliverable of the European research project ANEMOS.plus.

Hassan, M. R. and Nath, B. (2005). Stock Market Forecasting Using Hidden Markov Model: A New Approach. In: *Proceedings of the 5th International Conference on Intelligent Systems Design and Applications*, pp. 192–196.

Higham, D. J. (2008). *An Introduction to Financial Option Valuation*. Cambridge University Press.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. In: *Artificial Intelligence* 101.1, pp. 99–134.

Moura, P. S. and De Almeida, A. T. (2010). The role of demand-side management in the grid integration of wind power. In: *Applied Energy* 87.8, pp. 2581–2588.

Ong, S. C. W., Png, S. W., Hsu, D., and Lee, W. S. (2010). Planning under Uncertainty for Robotic Tasks with Mixed Observability. In: *The International Journal of Robotics Research* 29.8, pp. 1053–1068.

Papadimitriou, C. H. and Tsitsiklis, J. N. (1987). The complexity of Markov decision processes. In: *Mathematics of Operations Research* 12.3, pp. 441–450.

Pineau, J., Gordon, G., and Thrun, S. (2003). Point-based value iteration: An anytime algorithm for POMDPs. In: *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1025–1030.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc.

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

Silver, D. and Veness, J. (2010). Monte-Carlo Planning in Large POMDPs. In: *Advances in Neural Information Processing Systems*, pp. 2164–2172.

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

Ströhle, P., Gerding, E. H., De Weerdt, M. M., Stein, S., and Robu, V. (2014). Online Mechanism Design for Scheduling Non-Preemptive Jobs under Uncertain Supply and Demand. In: *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pp. 437–444.

Walraven, E. and Spaan, M. T. J. (2015). A Scenario State Representation for Scheduling Deferrable Loads under Wind Uncertainty. In: *The 10th Annual Workshop on Multiagent Sequential Decision Making under Uncertainty*.

Witwicki, S., Melo, F. S., Capitán, J., and Spaan, M. T. J. (2013). A Flexible Approach to Modeling Unpredictable Events in MDPs. In: *Proceedings of the International Conference on Automated Planning and Scheduling*, pp. 260–268.