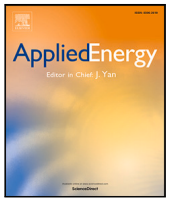
Applied Energy 302 (2021) 117519

Contents lists available at ScienceDirect

Applied Energy

journal homepage: www.elsevier.com/locate/apenergy



Applying reinforcement learning and tree search to the unit commitment problem

Patrick de Mars *, Aidan O'Sullivan

UCL Energy Institute, United Kingdom

ARTICLE INFO

Keywords:
Unit commitment
Reinforcement learning
Tree search
Deep learning
Power systems

ABSTRACT

Recent advances in artificial intelligence have demonstrated the capability of reinforcement learning (RL) methods to outperform the state of the art in decision-making problems under uncertainty. Day-ahead unit commitment (UC), scheduling power generation based on forecasts, is a complex power systems task that is becoming more challenging in light of increasing uncertainty. While RL is a promising framework for solving the UC problem, the space of possible actions from a given state is exponential in the number of generators and it is infeasible to apply existing RL methods in power systems larger than a few generators. Here we present a novel RL algorithm, guided tree search, which does not suffer from an exponential explosion in the action space with increasing number of generators. The method augments a tree search algorithm with a policy that intelligently reduces the branching factor. Using data from the GB power system, we demonstrate that guided tree search outperforms an unguided method in terms of computational complexity, while producing solutions that show no performance loss in terms of operating costs. We compare solutions against mixed-integer linear programming (MILP) and find that guided tree search outperforms a solution using reserve constraints, the current industry approach. The RL solutions exhibit complex behaviours that differ qualitatively from MILP, demonstrating its potential as a decision support tool for human operators.

1. Introduction

Unit commitment (UC) is a fundamental problem in power systems operation. It requires estimating the optimal schedule of generators to commit (turn on/off) to meet demand while minimising costs and maintain system stability. Generators must usually be committed hours or days in advance of delivery to account for generator startup and shutdown constraints and to allow the system operator time to evaluate security of the grid. Therefore, solutions to the UC problem must account for uncertainties arising from inaccurate forecasts of demand and renewables generation as well as other contingencies.

A reserve constraint is often enforced to manage uncertainties, and the resulting deterministic optimisation problem is solved by mixed-integer linear programming (MILP) [1]. However, these deterministic UC (DUC) approaches can be economically sub-optimal in high uncertainty power systems, due to the reliance on heuristic reserve constraints [2]. Research has shown that DUC can be improved upon by solving stochastic formulations of the problem that more rigorously account for uncertainties [3,4]. However, these methods have not seen widespread uptake in industry, due to their much higher computational requirements [2]. Nevertheless, the ubiquity and growing size of power systems means that even small relative efficiency improvements afforded by improved UC solution methods may yield significant economic and environmental benefits in absolute terms.

Recent advances in artificial intelligence (AI) have demonstrated promising results in learning optimal control strategies in complex domains by repeated trial-and-error in simulated environments. This approach of reinforcement learning (RL) has shown especially impressive results in games-playing domains [5,6], where AI has been able to exceed expert human performance without prior knowledge using self-play.

RL is a promising methodology for the UC problem, as it is capable of converging to optimal behaviour in complex stochastic domains, and much of the computational expense can be conducted 'offline' during training, in advance of the decision period. RL has shown some promising applications in power systems domains, such as for maintaining system security using remedial actions [7]. However, the curse of dimensionality in both state and action spaces means that applying RL methods to the UC problem 'out-of-the-box' is intractable for large problem systems. For this reason, existing applications of RL to the UC problem have considered small power systems of up to 12 generators [8].

* Corresponding author.
E-mail address: patrick.demars.14@ucl.ac.uk (P. de Mars).

https://doi.org/10.1016/j.apenergy.2021.117519
Received 15 March 2021; Received in revised form 21 July 2021; Accepted 29 July 2021
Available online 7 August 2021

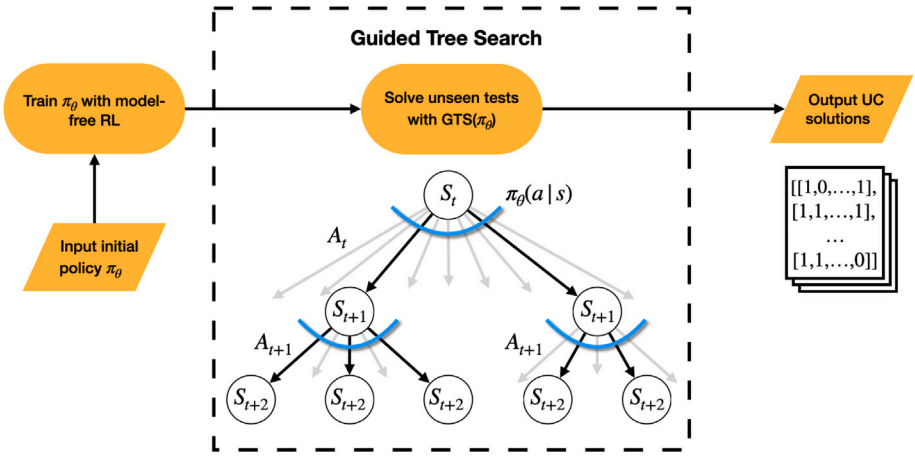


Fig. 1. Flowchart of guided tree search (GTS). The initial policy π_θ begins with random parameters θ, and is trained by model-free RL in the simulation environment. For unseen test problems, the trained policy π_θ calculates a distribution over actions, indicated by the blue line. Low probability actions are removed from the search tree. The reduced search tree (with branches indicated by black arrows) is then solved with uniform-cost search, outputting a UC schedule for each problem.

RL methods can be augmented by using tree search lookahead strategies which improve reliability in safety-critical contexts such as power systems operation. Inspired by similar approaches in other domains [9,6], in this paper we use a novel RL-aided tree search algorithm, 'guided tree search', to solve the UC problem with uncertain demand and wind generation. Policy gradient RL is used to learn a policy mapping states to a distribution over actions, which can be used to intelligently reduce the branching factor of a search tree. The policy is parametrised as a neural network. We use sample average approximation (SAA) and a simple tree search algorithm to find the least expected cost path through this reduced tree. Using SAA directly optimises for the expected costs, avoiding the use of heuristics to manage uncertainty. A schematic of our methodology, training with RL and testing with guided tree search, is shown in Fig. 1.

In order to determine whether the run time of guided tree search scales better in the number of generators than an exhaustive unguided tree search while producing UC solutions of similar operating cost, we simulate power systems of between 5–30 generators using demand and wind data based on the GB power system. We then evaluate whether guided tree search can provide operating costs that are competitive with industry-standard MILP solutions. To the best of our knowledge, this research is the largest simulation study using RL to solve the UC problem. In addition, it is the only study that uses unseen test profiles to evaluate performance. Given that RL policies often take hours or days to converge, the ability to generalise to unseen profiles is an important characteristic of an RL solution to the UC problem, allowing the training computation to be conducted in advance of the decision period.

Our results show that guided tree search does not exhibit the exponential time complexity in the number of generators faced by the unguided tree search algorithm. Guided tree search achieves lower operating costs as compared with MILP using a reserve constraint for systems of up to 30 generators. We find that the policy generalises from the problems seen in training to the unseen test problems and exhibits novel operational strategies in its solutions.

This paper contributes to the literature a novel approach for solving the UC problem with RL and tree search. Using RL we are able to overcome the curse of dimensionality, making our approach applicable to larger power systems. In addition, our method is applied in stochastic environment and to unseen problems.

The paper is structured as follows: Section 2 reviews literature on applications of RL to the UC problem. Section 3 presents our formulation of the UC problem as a Markov Decision Process (MDP). Section 4 describes our approach to solving the UC MDP. Section 5 describes the experimental setup for training and testing. Section 6 presents the results, comparing guided tree search, unguided tree search and MILP solutions. We discuss our results in Section 7. Section 8 concludes the paper and proposes future work.

2. Literature review

Unit commitment has been a major topic of research for decades and for a comprehensive review see [10,11]. Here we will review RL approaches to the problem.

Q-learning, a popular class of RL methods, has been applied to the UC problem in [12–14]. These papers have applied tabular Q-learning [12] and function approximation [13,14] with applications to problems of up to 10 generators. In [13], the results are validated against by comparison with Lagrangian Relaxation for a small deterministic problems while elsewhere Q-learning is not compared with the state of the art. The Q-learning methods applied so far are only applied to relatively small power systems and suffer from curses of dimensionality in the state space when tabular methods are used [12] and action space due to its combinatorial structure. Moreover, these methods do not consider testing on unseen problems, and the Q-learning agent is trained from scratch for each problem.

The cross-entropy method is used in [15] which solves a two-stage combined problem of day-ahead UC and real-time dispatch for the IEEE RTS-96 system. The action space is limited to choosing from a set of 20 generator commitments over the portfolio that is used for the entire day. There is therefore only one decision period for the UC component of this problem, representing a significant simplification of the UC problem that is likely to be sub-optimal.

Tree search methods are used to solve the UC problem in [8], which followed a similar approach to this paper. The authors investigate algorithms which choose greedily with respect to the least cost path from a given state to a fixed horizon. A tree search considering all available actions up to a fixed horizon produces 27% lower operating costs than a metaheuristic solution for a system of 12 generators. The authors apply a method for sub-sampling actions based on their distance in the action space to the current commitment, which decreases the execution time for longer time horizons. Compared with the approach used in this paper, the tree search in [8] does not exploit model-free RL or function approximation. In addition, the problems do not include stochastic demand or renewables generation, and no comparison is made with the state of the art.

The existing research on RL for UC has shown some promising results, but curses of dimensionality in both state and action spaces has limited these methods to small-scale problems. The only larger scale application [15] simplifies the problem by considering just a

single commitment decision, applied to all periods, chosen from a reduced action set of 20 actions which is determined by a heuristic. Moreover, there has been limited research on the generalisability of RL agents to unseen problems. In other domains, deep learning has been used to overcome the curse of dimensionality and train generalisable policies. Combined with Monte Carlo tree search (MCTS), this approach has seen success in the game of Go [6] and chemical synthesis [9]. However, while highly effective in problems formulated as game trees, MCTS is not well-suited nor designed for optimisation tasks such as the UC problem. Novel approaches are required in order to exploit the potential of RL and tree search methods for the UC problem, which we explore in this paper.

3. Unit commitment as a Markov decision process

In order to apply RL methods, we formulate the UC problem as a Markov Decision Process (MDP) with T decision periods, representing market settlement periods. MDPs consist of states, actions, rewards and a transition function determining the probability of moving from one state to the next following an action. A decision-making agent acts according to a policy which maps states to actions, and the task of RL is to determine a policy which maximises the long run expected return (sum of rewards) in the MDP [16].

In the UC MDP, the task is to maximise reward by minimising the operating cost of the power system. Demand and wind generation are stochastic, and the dispatchable thermal generators must meet the net demand (demand minus wind generation), or incur a lost load penalty. At each timestep t the agent observes the state of the power system S_t and takes an action A_t determining the commitment decision for generators at time $t+1$. The agent receives a reward R_{t+1} that reflects the operating cost of the grid and advances to a new state S_{t+1}, determined by a transition function $F(s, s', a) = \Pr(S_{t+1} = s' | S_t = s, A_t = a)$. In transitioning to a new state, the economic dispatch problem is solved giving the real power outputs of committed generators which is used to calculate the fuel costs. The objective is to maximise the expected return $\mathbb{E}[G_0] = \mathbb{E}[\sum_{t=0}^{T-1} \gamma^t R_{t+1}]$, equivalent to minimising the total operating cost of the grid. $\gamma \leq 1$ is the discount factor, which determines the present value of future rewards [16]. If $\gamma = 1$, the agent assigns as much credit to distant rewards as to immediate ones; if $\gamma = 0$, the agent only aims to maximise the immediate reward.

An episode is defined by a forecast demand profile, $\bar{\boldsymbol{d}}_0$, forecast wind profile $\bar{\boldsymbol{w}}_0$ and initial generator up/down times $\boldsymbol{u}_0$ as well as parameters for the stochastic processes of wind and demand forecast errors. At each timestep, forecast errors for demand X_t and wind Y_t are sampled, determining the demand and wind realisations $D_t = \bar{d}_t + X_t$ and $W_t = \bar{w}_t + Y_t$. The forecast errors are modelled with autoregressive moving average (ARMA) processes. We will now detail the states, actions, rewards and transition function of the UC MDP:

States. The observable state vector S_t includes the current generator up/down times $\boldsymbol{u}_t = [u_1, u_2, \dots, u_N]$ (where $u_i \neq 0$), for N generators; the demand forecast $\bar{\boldsymbol{d}}_t = [\bar{d}_{t+1}, \bar{d}_{t+2}, \dots \bar{d}_T]$; the wind forecast $\bar{\boldsymbol{w}}_t = [\bar{w}_{t+1}, \bar{w}_{t+2}, \dots \bar{w}_T]$. The forecast errors X_t and Y_t are not observed in our problem, since these are not available when solving the day-ahead UC problem. A state is terminal when $t = T$.

Actions. An action is a commitment decision that determines the on/off statuses of generators for the next timestep. An action is defined as a binary array $\boldsymbol{a}_t = [a_{1,t}, a_{2,t} \dots, a_{N,t}]$, $a_{i,t} \in \{0, 1\}$ for N generators. In other words, the agent chooses to turn each generator on ($a_i = 1$) or off ($a_i = 0$) at the following timestep. Actions must obey minimum up/down time constraints of the generators. The total number of unique actions is 2^N, but from a given state is limited to a subset of legal actions that obey operating constraints.

Rewards. The reward function for testing is the negative total operating cost of the system:

$$R_t = -\left[V_L L_t + \sum_{i=1}^{N} (C_i^f(p_{i,t}(D_t - W_t)) + C_i^s(u_{i,t}, u_{i,t-1}))\right] \tag{1}$$

where $C_i^f(\cdot)$ and $C_i^s(\cdot)$ are the fuel and start cost functions respectively for generator i. $p_{i,t}(D_t - W_t)$ is the power output of generator i at *net demand* $D_t - W_t$, and is determined by solving the economic dispatch problem. This is a convex optimisation problem that is solved with the lambda iteration method [17]. $u_{i,t}$ is the generator up/down time in periods. L_t is the lost load (difference between supply and demand), and V_L is the value of lost load.

Since the reward depends on the realisations of demand and wind forecast errors which are not included in the state vector, R_t is a random variable with respect to the state vector.

Transitions. When the agent acts on the environment, the transition function causes the environment to advance to the next timestep. Recall that the observable state vector comprises generator up/down times and the demand and wind forecasts. In our problem, the generator up/down time updates are deterministic. Given a commitment decision $a_{i,t}$ and integer generator up/down time $u_{i,t}$ for generator i, the transition function for the generator up/down time is:

$$u_{i,t+1} = \begin{cases} u_{i,t} + 1, & \text{if } a_{i,t} = 1 \text{ and } u_{i,t} > 0 \\ 1, & \text{if } a_{i,t} = 1 \text{ and } u_{i,t} < 0 \\ -1, & \text{if } a_{i,t} = 0 \text{ and } u_{i,t} > 0 \\ u_{i,t} - 1, & \text{if } a_{i,t} = 0 \text{ and } u_{i,t} < 0 \end{cases} \tag{2}$$

Demand and wind forecasts roll forward one timestep with each transition, such that $\bar{d}_{k+1}$ becomes $\bar{d}_k$.

The demand and wind forecast errors are sampled from ARMA(p, q) processes. At timestep t, the demand forecast error is sampled from:

$$X_t = \sum_{i=1}^{p} \alpha_i X_{t-i} + \sum_{i=1}^{q} \beta_i \epsilon_{t-i} + \epsilon_t \tag{3}$$

where α_i and β_i are constant parameters and ϵ_t is a normally distributed random variable with mean 0 and standard deviation σ. Wind forecast errors are sampled from an ARMA process with different parameters.

Having formulated the UC problem as an MDP, the agent's objective is to select a sequence of actions (i.e. a UC schedule) that maximises the expected return from the initial state. In the next section we will describe our methodology for solving the MDP with RL and tree search.

4. Methodology

The UC MDP described in Section 3 is solved by a tree search algorithm which searches for the path of least expected cost through a search tree representing the MDP up to a fixed horizon. Our novel adaptation is guided tree search, which uses a policy to choose a subset of actions to consider. This reduces the branching factor of the tree, thereby making it more tractable in large action spaces with more generators. The policy is trained by model-free RL. In this section we will first describe the unguided tree search algorithm, then the RL-aided (guided) tree search. Finally, we will describe the process for training the policy with RL.

4.1. Tree search

The basis for the tree search algorithm is similar to that described in [8]. A search tree is used to represent the MDP, where nodes represent states and edges represent actions. Traversing an edge (a state–action pair) incurs a cost which is the negative reward in the MDP. We calculate the least cost path through the tree up to a limited depth, representing the lookahead horizon, to save computation. Uniform-cost search is used to determine the least cost path, which is a

simple, heuristic-free algorithm that is guaranteed to find the optimal solution [18].

Once the least cost path is found from a root node at state s_t up to search depth H, we greedily choose the first action a_t in the least cost path. We then solve again for the new root at $s_{t+1} \sim F(s_t, a_t)$, retaining the search tree.

In order to apply this approach to an environment with a stochastic reward function, we must consider the distribution of costs for traversing an edge on the tree. The expected cost of an edge on the search tree is evaluated by averaging over the operating costs for multiple scenarios (sample average approximation) for the demand and wind forecast errors. The scenarios are generated by sampling from the ARMA processes. This approximates the expected reward of a given (s, a) transition. Calculating the expected cost for N_s scenarios of a given (s, a) pair involves solving the convex economic dispatch problem N_s times, so has linear run time complexity in N_s.

In the limit of H and N_s, this approach will minimise the expected operating cost over all scenarios by exhaustively searching for the least expected cost path through the MDP. However, using uniform-cost search will, in the worst case, visit each node in the search tree exactly once [18]. It therefore has a time complexity of $\mathcal{O}(2^{NH})$ for N generators. The exponential time complexity in N makes this method unscalable to large power systems. In addition, it risks making short-sighted decisions if there are long time dependencies due to generator constraints.

Next we will describe a method for using RL to improve the scalability of this tree search algorithm to larger power systems.

4.2. Guided tree search

Since the tree search algorithm previously described scales exponentially in the number of generators, it cannot be easily applied to large power systems. To make it tractable, we use an 'expansion policy' $\pi(a|s)$ to reduce the breadth of the search tree. The policy gives a probability distribution over actions for a given state, which can be used to intelligently reduce the branching factor of the tree. This probability distribution represents the relative value of different actions, with high probability actions being more promising than low probability ones. Instead of adding all available actions to the tree from state s, we instead only add those actions which satisfy $\pi(a|s) \geq \rho$, where $0 \leq \rho \leq 1$ is a branching threshold. The maximum number of nodes M that can be added to the tree is therefore limited to $M \leq \frac{1}{\rho}$, since $\sum_{a \in A} \pi(a|s) = 1$. In summary, if $A(s)$ is the complete set of available actions available from state s, the expansion policy is used to choose a subset of actions $A_\pi(s)$:

$$A_\pi(s) = \{a \in A(s) | \pi(a|s) \geq \rho\} \tag{4}$$

Using an expansion policy prevents the exponential time complexity of the tree search algorithm in the number of generators. We call this method *guided tree search*, and the algorithm described in Section 4.1 without an expansion policy *unguided tree search*.

The breadth and depth of the search tree are controlled by ρ and H respectively. There is a trade-off between these two parameters, as reducing ρ and increasing H both increase the run time of guided tree search. Setting ρ to be large results in a narrow search that can cause operating costs to increase. Similarly, a shallow search using a low setting of H can also degrade performance due to short-sighted decision-making. The optimal setting of H and ρ for a given computational budget can be determined experimentally, and depends on the MDP characteristics.

The expansion policy can be defined with expert rules, trained by supervised learning on expert data, or trained by RL as in this paper. Next we will describe our approach for training the expansion policy.

4.3. Expansion policy

During the training phase, we use policy gradient RL to learn a policy $\pi_\theta(a|s)$ to be used in guided tree search. Policy gradient methods learn a policy parameter θ to maximise the performance $J(\theta) = v_{\pi_\theta}(s_0)$, where $v_{\pi_\theta}(s)$ is the value function giving the expected return from state s when acting under policy π_θ [16]. In other words, we aim to learn the policy parameter θ that will result in the highest expected reward from the initial state s_0. Numerous algorithms have been designed to improve on the policy parameter with gradient ascent with respect to $J(\theta)$. For our research we use proximal policy optimisation (PPO) [19] with entropy regularisation. PPO uses a clipped objective that prevents large policy updates which can cause large temporary performance decreases resulting in insecure operational strategies. Entropy regularisation promotes more stochastic policies and is widely used in policy gradient RL to prevent premature convergence. In our case it also favours policies which will propose a diverse pool of actions satisfying $\pi(a|s) \geq \rho$ when used in guided tree search.

As is common in policy gradient RL methods, we simultaneously train a 'critic' to learn the state-value function $V(s_t) = \mathbb{E} \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$. Using a critic reduces the variance of policy gradient updates [16].

The policy is parametrised using a feed forward neural network. The size of the action space in the UC MDP is 2^N for N generators. Parametrising the neural network with 2^N output nodes is not feasible for even moderately large power systems. It also does not exploit symmetries in the action space, such as that base-load generators should be very rarely turned off. We design the neural network as a binary classifier which sequentially predicts each bit in the action sequence $\boldsymbol{a} = [a_1, a_2, \ldots, a_N]$, where $a_i \in \{0, 1\}$. Each a_i is a sub-action giving the commitment for generator i. The output of the classifier at each iteration is passed as an input into the next forward-pass through the network, thus maintaining the history of bits already classified. In addition, the input vector includes a one-hot encoding indicating the a_i being classified on each forward pass. This parametrisation preserves the inter-dependencies between generator commitment probabilities, while remaining scalable to larger numbers of generators.

The critic does not share neural network weights with the actor, due to the different dimensions of the input vectors of actor and critic networks.

It is not possible to analytically compute the entire distribution $\pi(\cdot|s)$ with this parametrisation. It is necessary to approximate the distribution in order to determine which actions meet the branching threshold ρ in the expansion phase of guided tree search (Eq. (4)). We use a Monte Carlo method to approximate the distribution, sampling many actions and observing their frequencies. In other words, we estimate that $\pi(a|s) = \frac{n_a}{N}$ where n_a is the number of times a was sampled and N is the total number of samples.

We will now describe the setup of the problem environments used for training the expansion policy with RL and testing guided tree search.

5. Experimental setup

The experiments will first compare guided tree search (which uses an expansion policy to reduce the search breadth) to unguided tree search (which does not use an expansion policy, and hence exhaustively searches for the least cost path through the tree). We aim to evaluate whether the run time of guided tree search scales better with the number of generators as compared with the unguided (exhaustive) search, without significant increase in operating costs. Second, guided tree search will be compared with MILP to compare whether the solutions are comparable in terms of operating costs with industry standard approaches. We will begin by describing the power systems used in our experiments.

Table 1
Generator specifications for the 10 generator problem [20].

$p_{\min}$	$p_{\max}$	u_0	a	b	c	$t^{\text{down}}_{\min}$	$t^{\text{up}}_{\min}$	c^s
150	455	16	0.00048	16.19	1000.0	16	16	4500
150	455	16	0.00031	17.26	970.0	16	16	5000
20	130	−10	0.00200	16.60	700.0	10	10	550
20	130	−10	0.00211	16.50	680.0	10	10	560
25	162	−12	0.00398	19.70	450.0	12	12	900
20	80	−6	0.00712	22.26	370.0	6	6	170
25	85	−6	0.00079	27.74	480.0	6	6	260
10	55	−2	0.00413	25.92	660.0	2	2	30
10	55	−2	0.00222	27.27	665.0	2	2	30
10	55	−2	0.00173	27.79	670.0	2	2	30

5.1. Power system setup

Comparing guided and unguided tree search, we consider power systems of between 5–10 generators. Comparing guided tree search and MILP, we consider 10, 20 and 30 generator power systems. Each day-ahead UC problem is comprised of 48 settlement periods of 30 min each. The generator data is from [20], a widely-used UC benchmark problem, which specifies generator data for 10 generators. This data is displayed in Table 1, where the coefficients a, b, c specify quadratic fuel cost curves of the form:

$$C^f(p) = xT(a(p)^2 + b(p) + c) \tag{5}$$

for each generator operating at power output p in MW. T is the dispatch frequency in hours and $x \in \{0, 1\}$ is the generator commitment (off or on). Each generator has minimum up/down time constraints $t^{\text{up}}_{\min}$ and $t^{\text{down}}_{\min}$, and a startup cost c^s. The initial status u_0 is only used for the test problems; in training the u_0 are randomly initialised to promote exploration of the state space.

A subset of the generators is used to create the smaller test systems and for larger systems we duplicate the 10 generators. Each episode consists of a demand forecast and a wind forecast, while the parameters of the ARMA processes are the same between episodes. For demand forecasts we use National Grid Demand Data [21] from 2016–2019. For each problem setting (10, 20 or 30 generators) we linearly scaled the demand to be between 40%–100% of the total capacity of the generation mix. For wind forecasts, we used openly available data for Whitelee onshore wind farm [22], chosen as a relatively large wind farm that operated continuously between 2016–2019. The wind data was scaled to be between 0%–40% of the total capacity of the generation mix. For both cases we used data from 2016–2019 inclusive for training and randomly sampled 20 days to withhold for testing. The average wind penetration (wind generation as proportion of demand) in training was 17.2%, with a maximum daily wind penetration of 58.1%. We set the value of lost load (VOLL) to be $10,000 per MWh for both training and testing, set to represent the approximate VOLL for a range of customer types [23].

The demand and wind forecast errors are sampled from ARMA(5, 5) processes, as described in Section 3, and we scaled the standard deviation of the white noise ϵ_t (Eq. (3)) linearly with the number of generators.

The RL environment used for training and testing is available open-source as a Python package, using the common OpenAI Gym API for RL research.[1]

5.2. Training details

We trained a policy using RL for each power system. During training, the RL agent samples a random day from the training data for each episode and the initial generator up/down times u_0 are set randomly to encourage exploration of the state space. The agent operates the grid 'online', and observes a single sample of the reward after each action, rather than the sample average approximate expected cost used by the tree search algorithms. The episode ends if the agent encounters lost load. In addition, we set the discount factor $\gamma = 0.95$, defined in Section 3.

Each policy was trained over 8 parallel workers, with weights of the actor and critic neural networks periodically updated and synchronised [24] after every epoch of 2000 policy evaluations. Note that at each timestep, the policy is evaluated multiple times corresponding to each generator sub-action, due to the sequential parametrisation of the policy (see Section 4.3). For the 5–9 generator problems, we trained for 25,000 epochs (approx. 2 h). For 10, 20 and 30 generator problems we trained for 100,000 (6 h), 200,000 (8 h) and 300,000 epochs (13 h), respectively. A grid search approach was used to determine the best performing combination of neural network architectures for actor and critic networks, testing combinations of the three architectures used in [25]: (64, 64), (100, 50, 25) and (400, 300). We determined the (64, 64) and (100, 50, 25) architectures to be the best performing and used these in our experiments.

Table 2
Summary of the 20 unseen profiles used as test problems for the 10 generator power system.

Date	DOTW	Wind (%)	$D_{\min}$ (MW)	$D_{\max}$ (MW)
2016-01-12	Tuesday	11.0	990.9	1593.1
2016-07-25	Monday	21.7	790.2	1118.5
2016-11-21	Monday	18.1	972.5	1545.3
2017-03-18	Saturday	21.3	827.5	1253.9
2017-04-07	Friday	7.4	880.2	1214.2
2017-05-12	Friday	14.4	867.5	1176.9
2017-05-26	Friday	14.8	817.4	1058.7
2017-06-25	Sunday	2.8	700.3	954.1
2017-12-18	Monday	8.1	1011.6	1599.9
2017-12-30	Saturday	18.4	916.4	1318.7
2018-01-15	Monday	15.0	893.3	1513.2
2018-03-08	Thursday	7.0	1024.1	1509.8
2018-03-18	Sunday	36.8	943.6	1432.6
2018-05-30	Wednesday	16.5	782.9	1177.1
2018-09-11	Tuesday	30.8	799.6	1183.8
2018-11-13	Tuesday	31.1	852.6	1409.8
2019-04-03	Wednesday	15.6	901.0	1306.3
2019-05-22	Wednesday	10.1	819.0	1104.8
2019-10-30	Wednesday	9.5	938.8	1414.2
2019-11-09	Saturday	8.9	918.5	1356.7

5.3. Testing details

We held out 20 days from the training data to use as test problems. A summary of test problems for the 10 generator system are shown in Table 2 showing date, day of the week (DOTW), wind penetration, minimum demand $D_{\min}$ and maximum demand $D_{\max}$. The tree search is used to solve each day based on the demand and wind forecasts. We then used a Monte Carlo method to estimate the expected operating costs over forecast error scenarios for each solution. 1000 realisations of demand and wind forecast errors were sampled from the ARMA processes, and the economic dispatch problem was solved for each, returning a distribution of operating costs for each solution.

To determine whether the performance of guided tree search is competitive with the industry methods, we compared with mixed-integer linear programming (MILP) solving a deterministic formulation of the UC problem. This approach is widely used in industry [2]. The DUC problem is described in [26], and we used the Power Grid Lib software package[2] to solve it within a 1% optimality bound. Two MILP benchmarks are used MILP(4σ) and MILP(perfect). The first uses a

[1] https://github.com/pwdemars/rl4uc

[2] https://github.com/power-grid-lib/pglib-uc

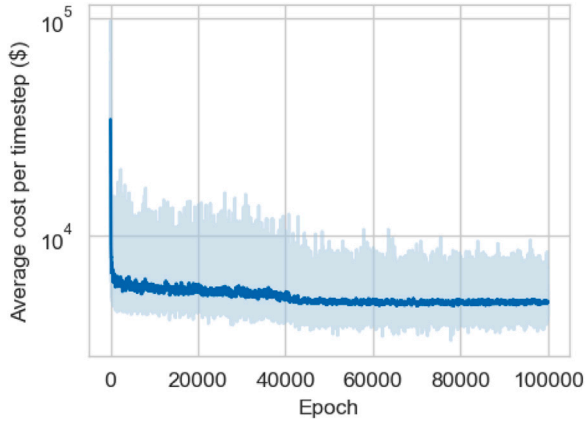


Fig. 2. Average operating cost per timestep for 10 generator policy training. The dark blue line shows a rolling average over 100 epochs (1 epoch = 2000 policy evaluations); lighter blue shows the data for each epoch.

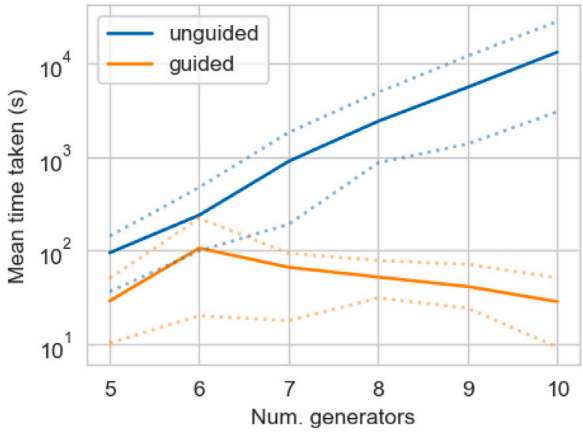


Fig. 3. Mean computation time for guided and unguided tree search from 5–10 generators. Dotted lines show the maximum and minimum time taken for a single problem. Unguided search run time increases exponentially with the number of generators for a fixed search depth, while guided search shows no significant increase in run time.

reserve constraint which is fixed to be 4 times the long-run standard deviation of the net demand forecast errors, an industry method described in [27]. The second assumes perfect foresight of demand and wind generation, and includes no reserve constraint. As with the tree search methods, a Monte Carlo method is used to evaluate the operating costs for 1000 realisations of demand and wind for the MILP(4σ) approach. Since MILP(perfect) is solved only for the point forecast, we do not apply the Monte Carlo evaluation method to this solution.

In the first experiment, comparing guided and unguided tree search, we set $H = 2$. Due to the exponential time complexity of unguided tree search with respect to H, higher settings of H were not feasible in practical run times. In the second experiment, comparing guided tree search with MILP, we set $H = 4$. We chose $\rho = 0.05$ as the most suitable setting for all our experiments using guided tree search, limiting the branching factor to 20 actions. Lower settings of ρ were more expensive to compute as the branching factor is inversely proportional to ρ, and performed only slightly better. Higher settings of ρ had faster run times but yielded higher operating costs as a result of narrower search.

6. Results

In this section we will present the results of our experiments. First we present the convergence of the expansion policies during training with RL. Then we will compare the performance of the RL-aided guided tree search with that of unguided tree search. Finally, for larger problems we will compare our approach to the two MILP benchmarks described in Section 5.3.

6.1. Training performance

For each problem setting (number of generators) we trained a policy using PPO as described in Section 5.2. In this setup, the agent interacts with the environment 'online' observing states and rewards as it plays through each day.

All policies converged to stable levels of average reward within the training epochs. Fig. 2 shows the convergence of the 10 generator policy in terms of operating cost per timestep (negative reward per timestep). The policy improves rapidly as it learns to avoid lost load events, before converging after around 40,000 epochs. There is still significant variability between epochs, shown in the faded blue of Fig. 2 which shows that the RL agent's operation of the grid is quite insecure. We used the policies to solve the test problems without tree search, and found it had a high loss load probability (4.0% for the 10 generator problem).

6.2. Guided vs. unguided tree search

Having trained the expansion policy, we will now apply it in guided tree search. To evaluate whether our method is capable of overcoming the curse of dimensionality in the number of generators without increasing operating costs, we will compare the performance to unguided tree search. As described in Section 4, unguided tree search calculates the lowest expected cost path over the full search tree to a fixed depth H while guided search considers a reduced set of actions from each state by using an expansion policy (Eq. (4)).

For the 5 and 10 generator problems, the comparison between guided tree search with $H = 2, \rho = 0.05$ and unguided search performing on 20 test problems with 1000 realisations of demand and wind generation is presented in Table 3. The guided tree search solutions produce nearly identical mean operating costs in both cases. Loss of load probability (LOLP) in the 10 generator case is significantly lower using guided tree search. The guided search is roughly 3 times faster to compute for 5 generators and around 500 times faster for 10 generators due to the reduced branching factor.

Fig. 3 shows that the computation time of the unguided search rises exponentially with the number of generators N. The run time for guided tree search remains comparatively stable, with a maximum run time of 220 s.

6.3. Guided tree search vs. MILP

Having shown that guided tree search overcomes the curse of dimensionality faced by unguided tree search without increasing operating costs, we will now compare its performance with industry-standard deterministic UC for power systems of 10, 20 and 30 generators. We compare with MILP(4σ), which enforces a reserve constraint equal to 4 times the standard deviation of the net forecast demand error (4σ). We also compare with MILP(perfect), which assumes the forecast is perfect and has no reserve constraint. Each solution is again evaluated over 1000 realisations of demand and wind forecast errors.

The results comparing MILP and guided tree search are shown in Table 4. All guided tree searches were conducted with $H = 4$ and $\rho = 0.05$. Guided tree search reduces costs as compared with MILP(4σ) constraint by 0.33%, 0.87% and 0.45% for the 10, 20 and 30 generator problems respectively. The LOLP is 28%, 56% and 51% lower than MILP(4σ) for 10, 20 and 30 generator problems. Guided tree search is 6.19%, 6.56% and 7.99% more expensive respectively than the MILP(perfect) solution with no reserve constraint and perfect forecast.

Table 3
Comparison of guided and unguided search for 5 and 10 generator problems.

Num. gens	Version	Mean cost (mln. $)	Std. cost	Mean time (s)	Max. time	Min. time	LOLP (%)
5	Guided	4.63	0.32	27.7	52.5	8.9	0.077
5	Unguided	4.63	0.31	94.1	141.8	35.9	0.074
10	Unguided	9.43	1.11	13249.9	28272.4	3037.9	0.177
10	Guided	9.44	0.83	28.3	50.6	9.2	0.152

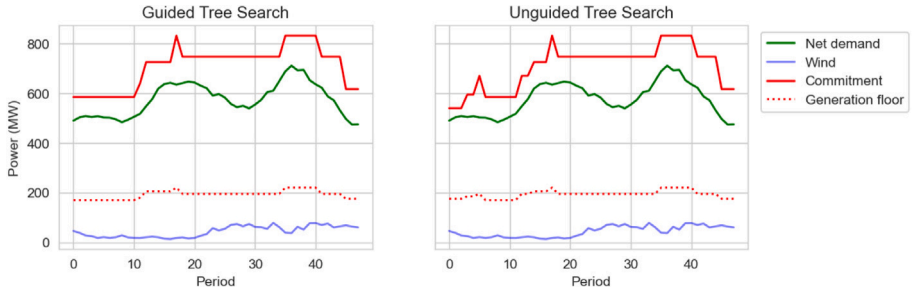


Fig. 4. Comparison of schedules produced by guided and unguided tree search, for the day 2018-03-08 with 5 generators. The generation floor is the sum of minimum operating outputs $p_{\min}$ of committed generators. The unguided tree search makes more frequent commitment changes and operates tighter reserve margins.

Table 4
Comparison of MILP and guided tree search solutions for 10, 20 and 30 generator problems.

Num. gens	Version	Mean cost (mln. $)	Std. cost	LOLP (%)
10	MILP(perfect)	8.82	0.00	0.000
10	Guided	9.36	0.76	0.128
10	MILP(4σ)	9.40	1.02	0.180
20	MILP(perfect)	17.58	0.00	0.000
20	Guided	18.73	1.41	0.107
20	MILP(4σ)	18.90	2.92	0.244
30	MILP(perfect)	26.31	0.00	0.000
30	Guided	28.41	2.04	0.142
30	MILP(4σ)	28.53	4.94	0.291

Our results show that guided tree search results in reduced operating costs and lower LOLP as compared with the MILP(4σ) solution with a reserve constraint for our problem instance.

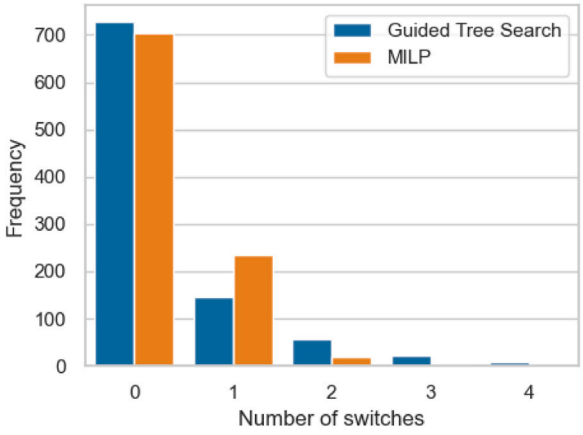


Fig. 5. Frequency of actions changing the commitment of multiple generators for the 10 generator problem. The guided tree search solution makes more frequent use of actions changing the status of 2 or more generators.

7. Discussion

Our results found that guided tree search exhibited no significant increase in run time when increasing from systems of 5 to 10 generators, and costs were slightly lower than those produced by the unguided tree search. Comparing with MILP, guided tree search outperformed the reserve constrained MILP(4σ) approach for all problem sizes.

There are differences in the operation patterns of guided and unguided tree searches. Guided tree search used 11% and 15% fewer startups than unguided startups for 5 and 10 generator problems, respectively. Unguided tree search uses more frequent commitment changes and operates tighter reserve margins. Fig. 4 illustrates the more inconsistent operation of unguided tree search. Unguided search, with a horizon of 2 timesteps, decommits generation at the beginning of the day, then performs restarts shortly after. By contrast, guided tree search avoids these short-sighted actions, suggesting that these are not proposed by the expansion policy which has learned to avoid such actions during training. In addition to the preferable run time complexity in the number of generators, the avoidance of short-sighted actions is another advantage of guided tree search over unguided tree search, which may explain its lower LOLP.

There are also some qualitative differences between the schedules produced by guided tree search as compared with MILP(4σ). Actions taken by guided tree search are more concentrated towards those which change the commitment of multiple generators at once, as shown in Fig. 5 for the 10 generator problem. At the same time, guided tree search makes more frequent use of the action which keeps all commitments the same. Fig. 6 shows example solutions for guided tree search and MILP(4σ) for the 20 generator problem. Compared with the MILP solution, guided tree search is characterised by longer periods of no commitment changes, and larger reserve margins at the end of the day when forecast errors are more likely to be large. The RL agent has learned complex behaviours, and guided tree search takes actions that would be difficult for a human operator to identify. As a result, this method shows promise for application as a decision support tool.

Some test problems were solved more quickly than others by guided tree search. In cases where the expansion policy evaluates states with a high degree of certainty, the tree search can complete very quickly as the branching factor is small. When the policy produces a relatively flat distribution, representing higher uncertainty, guided tree search took longer to complete. The guided tree search therefore adaptively spends more computation on more challenging states.

In [8], which is the closest research to our own, the tree search algorithm is shown to outperform a meta-heuristic method for a deterministic problem with up to 12 generators and 24 settlement periods. Through our experiments on 20 and 30 generator problems with 48 decision periods, we have shown that RL-aided tree search methods can be applied to larger problems using an expansion policy.

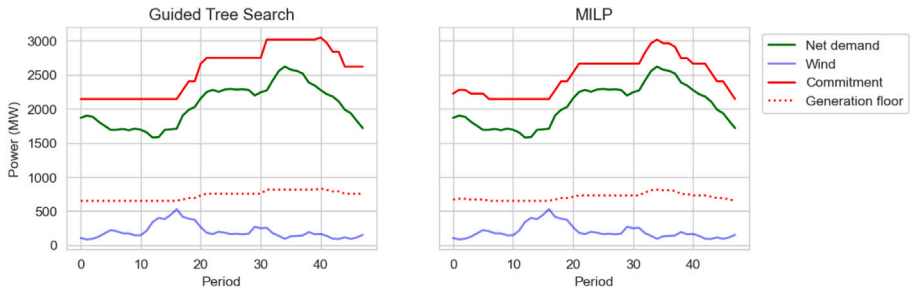


Fig. 6. Guided tree search and MILP(4σ) solutions for the 20 generator profile 2019-11-09. Guided tree search makes more frequent use of actions making 0 switches, thereby avoiding startup costs. The guided tree search solution also employs larger reserve margins at the end of the day when forecast errors can be larger.

8. Conclusion

In this paper we presented a novel guided tree search method for solving the unit commitment (UC) problem with uncertain renewables generation and demand using reinforcement learning (RL) and tree search. Guided tree search was shown to scale better than unguided tree search with increasing number of generators, while at the same time producing solutions with similar operating costs. Existing research either considered small power systems when applying RL [8,12–14], or simplified the problem to achieve tractability [15]. Our approach used RL to train a policy to reduce the action space, allowing for application to larger problem sizes, and showed that operating costs did not increase as a result. We applied our approach to a power system of 30 generators which, to the best of our knowledge, is the largest application of RL to the UC problem in the literature.

We also compared our results with the industry standard, mixed-integer linear programming solving a deterministic UC problem, and found that guided tree search reduces operating costs and loss of load probability. The application of RL generalising to unseen test problems is a novel contribution to the UC literature which, to the best of our knowledge, has not been attempted before now. Our results showed that this approach is effective, and the RL agent is capable of learning complex behaviours in a domain with high dimensional state and action spaces.

CRediT authorship contribution statement

Patrick de Mars: Conceptualisation, Methodology, Software, Validation, Investigation, Data curation, Visualisation, Writing – original draft. **Aidan O'Sullivan:** Conceptualisation, Methodology, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors acknowledge the use of UCL's Myriad High Performance Computing cluster for this research. This research was supported by an Engineering and Physical Sciences Research Council, United Kingdom research studentship (grant number: EP/R512400/1).

References

[1] Carrión M, Arroyo JM. A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem. IEEE Trans Power Syst 2006;21(3):1371–8.
[2] Bertsimas D, Litvinov E, Sun XA, Zhao J, Zheng T. Adaptive robust optimization for the security constrained unit commitment problem. IEEE Trans Power Syst 2012;28(1):52–63.
[3] Ruiz PA, Philbrick CR, Zak E, Cheung KW, Sauer PW. Uncertainty management in the unit commitment problem. IEEE Trans Power Syst 2009;24(2):642–51.
[4] Quan H, Srinivasan D, Khambadkone AM, Khosravi A. A computational framework for uncertainty integration in stochastic unit commitment with intermittent renewable energy sources. Appl Energy 2015;152:71–82.
[5] Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, et al. Playing atari with deep reinforcement learning. 2013, arXiv preprint arXiv:1312.5602.
[6] Silver D, Hubert T, Schrittwieser J, Antonoglou I, Lai M, Guez A, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science 2018;362(6419):1140–4.
[7] Marot A, Donnot B, Romero C, Donon B, Lerousseau M, Veyrin-Forrer L, et al. Learning to run a power network challenge for training topology controllers. Electr Power Syst Res 2020;189:106635.
[8] Dalal G, Mannor S. Reinforcement learning for the unit commitment problem. In: 2015 IEEE Eindhoven PowerTech. IEEE; 2015, p. 1–6.
[9] Segler MH, Preuss M, Waller MP. Planning chemical syntheses with deep neural networks and symbolic AI. Nature 2018;555(7698):604–10.
[10] Padhy NP. Unit commitment—A bibliographical survey. IEEE Trans Power Syst 2004;19(2):1196–205.
[11] Zheng QP, Wang J, Liu AL. Stochastic optimization for unit commitment—A review. IEEE Trans Power Syst 2014;30(4):1913–24.
[12] Jasmin E, Ahamed T. Reinforcement learning solution for unit commitment problem through pursuit method. In: 2009 International conference on advances in computing, control, and telecommunication technologies. IEEE; 2009, p. 324–7.
[13] Jasmin E, Ahamed T, Remani T. A function approximation approach to reinforcement learning for solving unit commitment problem with photo voltaic sources. In: 2016 IEEE international conference on power electronics, drives and energy systems. IEEE; 2016, p. 1–6.
[14] Li F, Qin J, Zheng WX. Distributed Q-learning-based online optimization algorithm for unit commitment and dispatch in smart grid. IEEE Trans Cybern 2019;50(9):4146–56.
[15] Dalal G, Gilboa E, Mannor S. Hierarchical decision making in electricity grid management. In: International conference on machine learning. PMLR; 2016, p. 2197–206.
[16] Sutton RS, Barto AG. Reinforcement learning: An introduction. MIT press; 2018.
[17] Wood AJ, Wollenberg BF, Sheblé GB. Power Generation, Operation, and Control. John Wiley & Sons; 2013.
[18] Russell SJ, Norvig P. Artificial Intelligence: A modern approach. 3rd ed.. Pearson; 2009.
[19] Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. 2017, arXiv preprint arXiv:1707.06347.
[20] Kazarlis SA, Bakirtzis A, Petridis V. A genetic algorithm solution to the unit commitment problem. IEEE Trans Power Syst 1996;11(1):83–92.
[21] National Grid Demand Data; https://www.nationalgrideso.com/data-explorer.
[22] Balancing Mechanism Reporting Service; https://www.bmreports.com.
[23] Schröder T, Kuckshinrichs W. Value of lost load: An efficient economic indicator for power supply security? A literature review. Front Eng Res 2015;3:55.

[24] Mnih V, Badia AP, Mirza M, Graves A, Lillicrap T, Harley T, et al. Asynchronous methods for deep reinforcement learning. In: International conference on machine learning. PMLR; 2016, p. 1928–37.

[25] Henderson P, Islam R, Bachman P, Pineau J, Precup D, Meger D. Deep reinforcement learning that matters. 2017, arXiv preprint arXiv:1709.06560.

[26] Knueven B, Ostrowski J, Watson J-P. On mixed-integer programming formulations for the unit commitment problem. INFORMS J Comput 2020;32(4):857–76.

[27] Holttinen H, Milligan M, Kirby B, Acker T, Neimane V, Molinski T. Using standard deviation as a measure of increased operational reserve requirement for wind power. Wind Eng 2008;32(4):355–77.