Energy- and Cost-Efficient Pumping Station Control

Timon V. Kanters
Informatics Institute
University of Amsterdam
tvkanters@tvkdevelopment.com

Frans A. Oliehoek
Informatics Institute, Univ. of Amsterdam
Dept. of CS, University of Liverpool
fao@liverpool.ac.uk

Michael Kaisers
Centrum Wiskunde & Informatica
michael.kaisers@cwi.nl

Stan R. van den Bosch
Nelen & Schuurmans
stan.vandenbosch@nelen-schuurmans.nl

Joep Grispen
Nelen & Schuurmans
joep.grispen@nelen-schuurmans.nl

Jeroen Hermans
HH Hollands Noorderkwartier
j.hermans@hhnk.nl

Abstract

With renewable energy becoming more common, energy prices fluctuate more depending on environmental factors such as the weather. Consuming energy without taking volatile prices into consideration can not only become expensive, but may also increase the peak load, which requires energy providers to generate additional energy using less environment-friendly methods. In the Netherlands, pumping stations that maintain the water levels of polder canals are large energy consumers, but the controller software currently used in the industry does not take real-time energy availability into account. We investigate if existing AI planning techniques have the potential to improve upon the current solutions. In particular, we propose a light weight but realistic simulator and investigate if an online planning method (UCT) can utilise this simulator to improve the cost-efficiency of pumping station control policies. An empirical comparison with the current control algorithms indicates that substantial cost, and thus peak load, reduction can be attained.

Introduction

The Netherlands contains many areas that lie below sea level. To prevent these areas, called *polders*, from flooding, they are surrounded by dikes and allow their water levels to be controlled. This is done through mechanical devices such as windmills or, nowadays usually, pumping stations. When rain falls and the water levels in the polders rise too high, the water is pumped out into canals that act as a drainage system. The operation of the pumps in these systems falls under the responsibility of water boards and has evolved from manual control to the use of automatic controllers.

The design of pumping station controllers can be complex due to the need to reason about probabilities of rainfall and energy prices, and the way that the actions of different pumping stations will interact. While current state-of-the-art controllers of our industrial partner, Nelen & Schuurmans (N&S), do reason about projected amounts of rain, they do not explicitly consider the interaction of actions taken at different pumping stations, nor do they reason about energy prices and their uncertainty. This latter point is expected to be particularly problematic for the affordability of maintaining desired water levels in the future; operation of pumping stations is energy intensive and, with the advent of more renewable energy sources such as wind and solar, the amount of available energy and thus its price is expected to vary more [Würzburg, Labandeira, and Linares, 2013]. As with other applications that rely greatly upon energy availability [De Nijs, Spaan, and De Weerdt, 2015; Rogers, Ramchurn, and Jennings, 2012], controlling pumping stations without taking the availability into account will become very expensive and challenging. On the other hand, this yields an opportunity: optimising when to pump can save costs. Moreover, better scheduling may lead to reduced peak load and thus a positive effect on the entire energy network by reducing the amount of additional energy required to be generated [Ketter, Peters, and Collins, 2013].

While there has been considerable research into the conceptually related topic of controllers for irrigation networks [Cantoni et al., 2007], these networks rely on gravity rather than pumps, meaning that the energy efficiency question is much less pressing in this domain. In this paper, we present an initial investigation into whether AI planning techniques can be used to improve coordination and energy use in pumping station control. In particular, we present a detailed yet lightweight simulation of an existing polder system in The Netherlands, and discuss how a state-of-the-art AI planning technique, UCT [Kocsis and Szepesvári, 2006], can be applied to it. We validate the proposed simulation model by comparing it with more detailed industrial models, discuss domain-specific modifications of UCT, and report on an empirical evaluation that demonstrates that the resulting technique has the potential to significantly reduce costs when compared to the solutions currently used in practice.

The Problem Domain

The pumping station control problem is a sequential decision problem under uncertainty. The main difficulty is the requirement to coordinate between multiple pumping stations, while taking into account the uncertainty of how circumstances (such as rain and energy prices) will develop. In this paper, we focus on the polder system called *Vereenigde Raaksmaats- en Niedorperkoggeboezem* (VRNK) (Figure 1), which is located in The Netherlands and administered by the water board *Hoogheemraadschap Hollands Noorderkwartier* (HHNK). HHNK and N&S have identified this

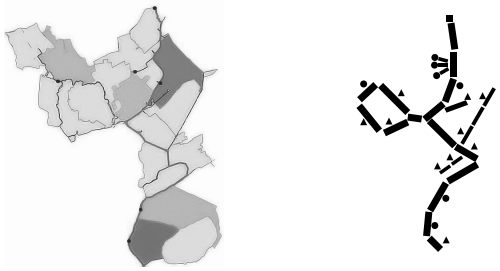

Figure 1: Illustrations of the VRNK polder system: on the left an overview with main (controllable) polders indicated by darkened areas and on the right the modelled VRNK polder system with lines as canal parts, circles as controllable pumping stations, triangles as uncontrollable pumping stations and a square as end drain.

system as suitable for experimentation in the form of testing new controllers and having a large potential for energy cost savings [Nelen & Schuurmans, 2013].

The VRNK polder system has a water surface of 163 ha and discharges in the north. It contains 20 polders with a typical surface between 300 ha and 700 ha. We consider the pumping stations of seven polders to be controllable by our method, which have been selected for real-world pilots as they have the largest pumping capacity and thus the highest expected cost reduction.

Currently, the inlet to and discharge from the VRNK canals is fully automated using the controller software *Control NEXT* (CN) [Deltares, 2015a]. CN is based on expert-knowledge and the results of a repeatedly running hydrological simulation model, and has proven itself in practice through its utilisation by various companies. CN bases its pumping actions on the current water levels and the expected amount of excess water in the future. This gives a good indication of how much pumping capacity is needed and over what time span. However, CN does not reason about energy prices or interactions between pumping stations, which offers great potential for improvement. Pseudocode for CN is described by Kanters [2015].

To produce the energy required by the pumping stations, water boards rely on purchasing energy through one of the available markets. Like many European electricity markets, the Dutch electricity market is held in three eponymous stages: the Day-Ahead Auction E_a, the Intraday market E_r and the imbalance market E_i [Triple, 2015]. The Day-Ahead Auction is held one day before delivery. The Intraday market allows purchases at hourly intervals as well as freely definable block orders up to 5 minutes prior to delivery. While these two markets are two-sided and settle demand and supply between different electricity traders, the imbalance market typically employs a one-sided auction where up- or downward regulation power (or reserve capacity) is offered to the *transmission system operator* as single buyer.

All of these markets are characterised by stochastic price developments, with decreasing transaction volume v, increasing *average* prices μ and increasing price volatility σ, such that $v_{E_a} > v_{E_r} > v_{E_i}$, $\mu_{E_a} \leq \mu_{E_r} \leq \mu_{E_i}$ and $\sigma_{E_a} < \sigma_{E_r} < \sigma_{E_i}$. We have selected the imbalance market as a challenging test environment. As it is notoriously difficult to predict, we expect results to generalise to other market price signals, given that they may be more predictable and thus less challenging to plan for. The experiments in this article thus use historical data of imbalance prices [TenneT, 2015].

Online AI planning

Since current pumping station controllers do not take energy prices into account, the expected cost for maintaining water levels within desired limits is going to increase if energy is obtained from volatile short term markets with higher average prices. In this paper we investigate the ability of online planning methods, and particularly Monte Carlo Tree Search (MCTS) [Browne et al., 2012], to exploit the flexibility in when to pump to benefit from the changing prices. Like model predictive control (MPC) [Qin and Badgwell, 2003], such methods employ models of the environment to determine which action to take while interacting with the environment. However, in contrast to typical MPC approaches, MCTS methods reason about different possible execution paths that might occur (due to stochastic noise, unpredictable events, etc.), as well as the optimal actions to take if those paths occur.

Conceptually, these methods work by treating their environment as a *Markov decision process (MDP)* [Puterman, 1994]; at every time step, or *decision epoch*, the environment is in a particular state s, which is affected by the action a of the controller, or *agent*, leading to a next state s'. A full MDP model specifies both the probabilities of transitions $P(s'|s,a)$, as well as the corresponding immediate rewards $R(s,a,s')$ that specify the task. The agent's goal is (typically) to maximise the expected sum of rewards.

MCTS methods, however, are sample-based planning methods that do not need access to a full MDP model. Instead they only need a *generative model*, or *simulator*, $\mathcal{G}$ from which transitions and rewards can be sampled $s', r \sim \mathcal{G}(s,a)$. They work by sampling the effects of actions and creating a tree structure based on the results. While building the tree, MCTS uses a *rollout policy* to select actions in states that are not part of the tree yet, to swiftly sample the quality of the state. This quality is then used to update the tree, allowing it to estimate the quality of each state in it. MCTS navigates through its existing tree structure by selecting actions that it believes are worth sampling. This selection greatly affects the performance of MCTS. One of the most successful ways of doing this is through *Upper Confidence Bounds for Trees* (UCT) [Kocsis and Szepesvári, 2006]. Using the algorithm UCB1 [Auer, Cesa-Bianchi, and Fischer, 2002], actions are selected based on their optimistic potential value.

When MCTS is applied in the real world, it uses the simulator for planning. In this paper, however, we evaluate the proposed method in simulation. This gives rise to two different simulators: the *real simulator* $\mathcal{M}_s$ as stand-in for the real world, and the *planning simulator* $\mathcal{M}_p$ as generative model given to the agent for planning purposes. In many cases, these two simulators can be identical, but we will need to treat them differently in some aspects as discussed later.

Polder System Simulation

This section proposes a generative model for the pumping station domain. As sample-based techniques often require a large number of samples that indicate effects of actions, it is important that the simulation runs fast enough to facilitate this. Water level simulators currently used in practice, such as SOBEK [Deltares, 2015b], focus on accuracy more than speed and thus are not fit for our purposes. In this section we propose our own simulator which has a more suitable trade-off between speed and accuracy.

The model we use for the polder system represents its canals as a graph with canal parts as nodes. Each canal part can have a polder or an end drain connected to it, where polders can pump water into the canal part and end drains pump water out of the canal part. Though our controller only selects actions for a number of pumping stations, as mentioned earlier, all pumping stations are modelled in our simulation, and by default controlled through CN. Canal parts and polders both have a number of fixed properties:

- a target (or goal) water level margin L_g in mNAP[1];
- a bottom water level margin L_b in mNAP;
- a top water level margin L_t in mNAP;
- a maximum water level L_m in mNAP (exceeding this causes flooding);
- a surface area A in m^2.

Polders and end drains have one additional fixed property:

- a pumping station capacity S_c in m^3/s.

State Definition Apart from fixed properties, canal parts and polders also have variable properties, which are defined in the *state*. The state also contains variable properties for the energy prices and weather. We formally define the state as $s = \langle L_c, H_{C_e}(t), H_{R_f}(t) \rangle$ where L_c is a vector containing the current water level for each canal part and polder in mNAP, H_{C_e} is the history of energy prices C_e in euros up to decision epoch t and H_{R_f} is the history of rainfall R_f in m up to decision epoch t. The histories are used by our simulator for transitioning states. In practice, we found that, in order to prevent histories from indefinitely increasing in size over time in practice, limiting them so that only the ten most recent observations are stored (i.e., 2.5 hours) is sufficient for planning.

Action Definition The controller selects an action $a = \langle a_1, ..., a_{N_a} \rangle$ where a_n is the selected action value for controllable pumping station n and N_a is the amount of controllable pumping stations. The value of a_n linearly scales the pumping station's capacity to determine the pumping capacity used for the transition. The possible values for a_n may differ per controller. For our proposed controller, we found the available action values per pumping station $a_n \in \{0, 1\}$ to be suitable. This discrete action space gives us a size of 2^{N_a}. If in practice it proves useful to have additional options, this action space can easily be extended to include intermediate values. Control NEXT does require more actions than on or off, and as such has access to action values $a_n \in [0, 1]$.

[1]mNAP stands for 'meter boven Normaal Amsterdams Peil', or 'metres above Amsterdam Ordnance Datum', and is used as a unit for water level height.

Effects of Actions Pumping stations with an action value $a_n > 0$ will alter the water levels. In case the pumping station is connected to a polder, the polder water level lowers and the water level in the connected canal part rises. In case the pumping station is an end drain and therefore not connected to a polder, the water level in the connected canal part lowers. The water levels are adjusted by $D_l = a_n S_c T / A$ where D_l is the water level difference in m and T is the time spent pumping in s (the decision epoch length).

Water Flow After water levels change due to pumping, water from one canal part can flow to its neighbours. A standard way to model this is via the *Gauckler–Manning formula* (GMF) [Manning et al., 1891] which estimates the velocity of liquid in an open channel, the details of which are described by Kanters [2015]. As the pumping stations can process an action every 15 minutes, we use this as our decision epoch. Between each decision epoch, we must calculate the new water levels. However, 15 minutes is a too coarse granularity to apply GMF in our model. To remedy this, each decision epoch is subdivided into multiple *GMF steps* that each calculate intermediate water levels. This allows water movement to span multiple canal parts in one decision epoch. As additional GMF steps do come at the cost of additional computation, we aim for a good balance between realism and performance. In our experiments, we found ten GMF steps per decision epoch to be suitable.

Rainfall and Energy Price Transitions Apart from the deterministic effects of water flow and pumping station actions, the state transition is influenced by rainfall and energy price development. It is imaginable that there could be correlations between these [Panagopoulos, Chalkiadakis, and Jennings, 2015], which an analysis of 10 years of historical data confirms [Kanters, 2015]. We therefore propose a method of simulation that preserves these correlations. However, we must make a discrimination between the *real simulator* $\mathcal{M}_s$ and the *planning simulator* $\mathcal{M}_p$ as defined earlier.

For the *real simulator* we propose to take a historical data approach to simulation. In particular, we use synchronised historical data in the form of time series. When the initial state of the real simulator is generated, a random time point in this data is selected as the first rainfall level and energy price (i.e., this time point is the same in both data sets). At each state transition, the next time point in the data is used to determine the rainfall and energy price for the next state. While this limits the simulation of rain and energy price to be the same as those observed in our data set, the advantage is that it respects their correlation. This could be important; ignoring the correlation may lead to our estimations being too positive since high prices and rain are likely correlated.

For the *planning simulator* it is not possible to use such an approach. As real transitions bear uncertainty, it is important that the controller's planning reflects this. Here, we take different approaches for rainfall and energy price simulation.

For the model of rainfall, we propose the utilisation of external tools to circumvent this prediction problem. When a UCT-based controller would be deployed in practice, it could have access to commercially available weather forecasts that predict the amount of rain that will fall in the area

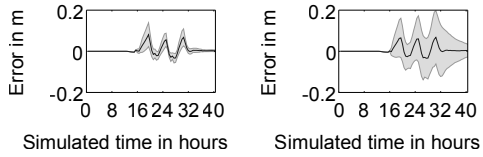

Figure 2: The mean and standard deviation of all water level differences between $\mathcal{M}_s$ and $\mathcal{M}_b$ with $G_c = 0.015$ (left) and $G_c = 0.03$ (right).

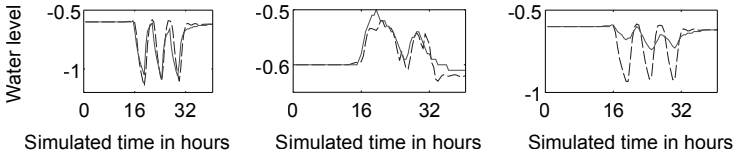

Figure 3: Example water levels from different canal parts in simulation $\mathcal{M}_s$ (solid line) and $\mathcal{M}_b$ (dashed line).

of interest in the coming hours. We expect that these forecasts are much more accurate than any learned model of rainfall transitions up to the point that we can argue that the agent knows what the future rainfall will be. Therefore, in our planning simulator, the basis for the sampled rain levels is the forecast. In the current simulations, we use historical data from *Meteobase* [STOWA, 2015], but for real world usage, forecast services like *Buienradar* [Buienradar, 2015] may be used. Of course, forecasts are not flawless. Taking this into into account, we add Gaussian noise with a standard deviation of a third of the value predicted by the forecast.

For energy price data, however, no such prediction is available. Instead, we use a technique for *inferring beliefs over scenarios* [Walraven and Spaan, 2014]. This technique attempts to find the best fits of the last few observed energy prices in the available historical data. The *scenarios* in the historical data that match the observed data best are then used for the energy price transition, where a scenario is randomly selected based on their probability. To prevent this technique from being reduced to a look-up of an exact match, we supply it with a different data set than the one that the actual simulation uses. We build upon this technique by utilising the correlation between rainfall and energy prices to improve its performance in our problem; we select scenarios not only based on the energy price fit, but on the best fit of both the energy costs and rainfall.

Evaluation of Simulation Realism

Further into this paper, we report on experiments regarding the performance of our proposed techniques. In order to assess the implications for real-world deployment, however, it is important to evaluate the accuracy of the *real simulator*. This evaluation focuses on the water transitions only. Though we also simulate energy price transitions, in the *real simulator* only historical data is used, which is realistic by definition. Energy price transitions in the *planning simulator* affect UCT performance, but not the representation of our results for real-world deployment. Its accuracy is described by Kanters [2015].

Sufficient historical data for water flow is not available, and thus in order to evaluate the realism of our *real simulator* $\mathcal{M}_s$, we use one of the state-of-the-art simulators as baseline $\mathcal{M}_b$: SOBEK [Deltares, 2015b]. SOBEK is a physics-based modelling suite used by water authorities and water management consultancies for water flow simulation, and is sufficiently accurate to be considered ground truth. As it does not handle pumping station control, the user must specify when pumping stations activate and the amount of rainfall. With the VRNK model that is used by HHNK to determine suitable control policies, we can compare the water behaviour of $\mathcal{M}_s$ and $\mathcal{M}_b$. In our experiment, we run a typical simulation in SOBEK where the water levels start at their target and are raised by heavy rain. Once the water levels reach a certain point, pumping stations activate and water flows through the canals. Using the same rainfall and pumping station actions, we mimic this simulation in our own simulator.

We determine the accuracy at each decision epoch t by using the difference of water levels $\mathcal{M}_e(t) = \mathcal{M}_s(t) - \mathcal{M}_b(t)$ as predicted by simulators $\mathcal{M}_s$ and $\mathcal{M}_b$. Figure 2 shows the mean and standard deviation of the water level differences of the different canal parts. The first 16 hours do not involve pumping and only contain a slight rain build-up. After this first period, pumps activate, which allows differences in water levels between the simulations to become apparent.

Typical Gauckler-Manning coefficient G_c references suggest $G_c \approx 0.03$ for natural canals [Mott and Wagenaar, 2009; Te Chow, 1959; Edwards, 2000]. However, we find that $G_c = 0.015$, which is typically used for smoother man-made canals, gives us a much more realistic water flow. Comparing the left and right plots of Figure 2, we see that the means of both coefficients are similar, but $G_c = 0.03$ gives a much higher standard deviation than $G_c = 0.015$, indicating that there is a lot more simulation error there. As such, we set $G_c = 0.015$ in our simulation and experiments.

Examples of water levels in individual canal parts are seen in Figure 3. The left plot shows water levels close to the end drain, which activates three times causing the water levels to drop sharply and rise again. The middle plot relates to water levels south in the VRNK polder system where pumping stations of polders activate twice. These first two plots are examples of areas where $\mathcal{M}_s$ has very similar results to $\mathcal{M}_b$. Figure 2 shows that there are also parts where higher error in prediction occurs. One of these canal parts is seen in the right plot. This shows the water levels of a branching canal in the centre of the VRNK polder system and illustrates that our model's accuracy in this area can still be improved. Before using our simulation in a real world pilot, looking into these model discrepancies together with N&S or HHNK can likely reduce these water level differences.

Overall, our simulation $\mathcal{M}_s$ is sufficiently accurate and generally keeps water levels close to those of $\mathcal{M}_b$. The error seen in Figure 2, which ranges from 0.05 m below the baseline to 0.1 m above it, could be considered a risk. We can minimise this risk in practice by simply adjusting the targets; placing the top target 0.05 m lower and the bottom target 0.1 m higher will cause the controller to select safer actions when water levels are near their target boundaries, preventing damages from simulation error.

Controlling Pumping Stations through UCT

Having specified our simulator, we look into the controller itself. In this section we describe how UCT can be applied to the pumping station domain. We use a slightly modified version of UCT that fits our model and consider the effects of different parameters and rollout policies.

Action Quality and Rewards

The immediate quality of an action is based on the cost of the pumping stations that were active and the resulting state. The penalties for pumping costs, water levels and overflow combined determine the total reward $r = -\sum_n P_c(n) - \sum_l P_l(l) - P_o$ where r is the reward in euros, $P_c(n)$ is the pumping cost in euros for pumping station n, $P_l(l)$ is the water level penalty in euros for canal part or polder l and P_o is the overflow penalty in euros.

Penalty for Pumping Using data from HHNK, we found a correlation between energy consumption relative to the polder's surface and the water level difference between the polder's target L_g and the canal's target. This is shown in Figure 4 where each point represents a pumping station. The line represents the fit that we use for the simulation. This gives us a cost per pumping station of $P_c(n) = C_c D_l(n) A(n) C_e$ where C_c is a constant with value 8.1×10^{-6}, $D_l(n)$ is the water level difference in m and C_e the energy price per MWh in euros.

Penalty for Water Levels Water levels deviating from the target levels incur costs. Each polder and canal part have a bottom and top target water level. When the current water level is outside of this target, a penalty is given according to

$$P_l(l) = \begin{cases} |L_t(l) - L_c(l)|^2 A(l) C_t, & \text{if } L_c(l) > L_t(l) \\ |L_b(l) - L_c(l)|^2 A(l) C_b, & \text{if } L_c(l) < L_b(l) \\ 0, & \text{otherwise} \end{cases}$$

where C_t is the cost per squared target level excess per m^2 surface in euros and C_b is the cost per squared target level deficit per m^2 surface in euros.

The amount of metres off target is squared in order to penalise larger deviation. We set $C_t = C_b = 1$ in consultation with N&S. This is a rough estimation of the true cost based on real world scenarios, but fulfils its purpose for our experiments. More in-depth research into these costs can be done to improve real world performance.

Penalty for Overflow Finally, there is an extra cost in case of flooding, which adds $P_o = OC_o$ where O is the amount of overflowed water in m^3 and C_o the cost in euros per m^3. As advised by N&S, the damages reported during a flooding of Texel, a Dutch isle, are roughly representative for those in the VRNK polder system. We therefore set $C_o = 3$ based on data from that event [Nationaal Watertraineeship, 2015].

Domain-Specific UCT Extensions

In order to effectively apply UCT to the pumping station domain, we make a number of domain-specific design choices.

First, we introduce *binning* of state variables only within the UCT search tree. I.e., for the purpose of creating nodes in the search tree, we group the water levels and energy prices for the UCT tree structure in bins of 1 cm and € 10 respectively based on our experiments [Kanters, 2015].

Second, we consider a number of alternative rollout policies. Without domain knowledge, it is standard practice to select a rollout policy which selects random actions. The downside of this is that it causes UCT value actions highly when they lead to a good outcome if a random policy takes over later on. I.e., it does not necessarily optimise towards states that are good when more sensible policies take over.

Using domain knowledge, it may be possible to construct better rollout policies. One of our considered rollout policies is CN. Inspired by CN, which looks at future states that follow after doing nothing, we also consider a *do nothing* rollout policy which keeps all pumping stations inactive.

Finally, we consider a rollout policy that learns from UCT's sampling to determine if actions look promising in the future: *Predicate-Average Sampling Technique* (PAST) [Finnsson and Björnsson, 2010]. PAST selects actions based on the average rewards they received since the start of the program. The average rewards are stored per *predicate*, which is based on certain features of the state that the action was taken in. The action distribution used in PAST is based on the predicate with the best value. We find this too optimistic in our situation however. E.g., a predicate for low energy prices might be selected rather than a predicate indicating imminent flooding. Instead of using different predicates for individual state features, we therefore have predicates defined by all state features.

Empirical Evaluation

Here, we empirically investigate if our proposed application of UCT can improve pumping station control over Control NEXT, which currently controls the VRNK polder system, and an alternative method commonly used in practice.

Experimental Setup One requirement of a pumping station controller is to be able to perform well during a long time span. To accommodate this, we let the experiment last two (simulated) weeks such that the *horizon* $h = 1344$ decision epochs. To ensure that the controller's task is challenging, we make the experiments difficult in two ways.

First, the initial state is generated such that water levels are selected uniformly randomly from $L_c \in [L_t - 0.2, L_t]$, causing them to be near their top target. Second, we select a point in the historical data where rain is just starting. This results in a scenario where the water levels are high (sometimes dangerously so) and still rising. We generate 8 such initial states and report the mean average results over those.

UCT has a number of parameters that can be set. Based on our experiments [Kanters, 2015], we found $65,000$ planning simulations (which takes about about 5 seconds on a typical work station) and a search depth of 32 decision epochs (which equates to 8 hours) to be good settings.

Rollout Policies UCT's rollout policy uses greatly affects its performance. As such, we first determine which policy yields the best results in our problem. We compare all earlier described rollout policies: random actions, Control NEXT,

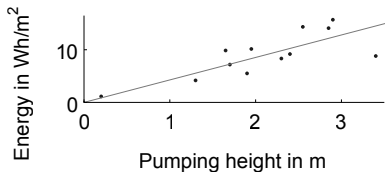

Figure 4: The relative energy consumption per pumping station (points) and the fit we use (line).

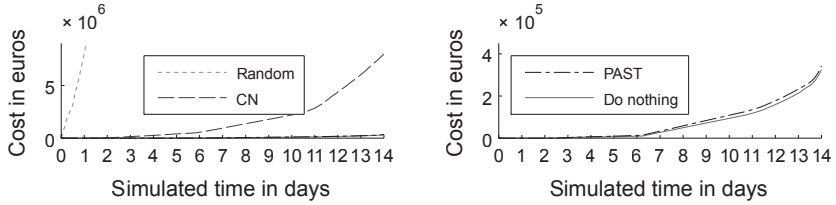

Figure 5: The mean cost (and standard error) of different UCT rollout policies over time. All tested policies are shown on the left, and the two best are zoomed in on the right.

doing nothing and PAST. As seen in Figure 5, a random rollout policy quickly leads to very bad results. The other rollout policies are more feasible, with PAST and the *do nothing* rollout policy performing much better than CN as rollout policy. Though close, *do nothing* achieved slightly lower costs than PAST. As is it also computationally lighter, we use the *do nothing* rollout policy during other experiments.

Baseline Comparison Finally, we compare UCT's performance to current industry baselines: Control NEXT and a *two-threshold* controller (TTH). TTH is a simple *on-off* controller with hysteresis [Aström and Murray, 2010], which despite (of because of) its simplicity is frequently used in practice [Taylor et al., 2000; Driankov, Hellendoorn, and Reinfrank, 2013]. In our implementation, we enable pumps when the water level reaches a certain threshold, and disable them again when the water level has lowered to another threshold [Kanters, 2015].

As indicated in Figure 6, TTH incurs much higher costs than CN, which in turn is significantly outperformed by UCT. Pumping while energy is expensive can have great effects, as is shown by the large jumps in costs at certain points in the experiments. Comparing UCT's behaviour to that CN and TTH, the latter tend to pump at sudden moments while UCT spreads this pumping time more broadly. This allows UCT to select the cheapest moments to pump in advance, preventing it from having to pump when energy is costly.

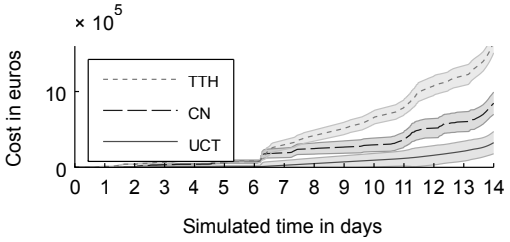

Figure 6: The mean cost and standard error of UCT, Control NEXT and TTH over time.

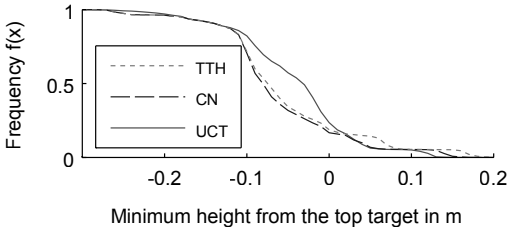

Figure 7: Frequency $f(x)$ with which water levels throughout the experiments exceeded their top target L_t by x m.

The costs mainly consist of pumping costs, as all methods are able to keep the water levels either within margin or close to it, even under the harsh experiment conditions. This is shown in Figure 7, which illustrates how frequent water levels throughout the experiments rose above certain levels relative to their top target L_t. In particular, it shows $f(x) = \frac{1}{N_l \times h} \sum_{l=1}^{N_l} \sum_{t=0}^{h-1} I\{L_c(l,t) > L_t(l) + x\}$, where N_l is the number of canal parts and polders, and $I\{\cdot\}$ is the indicator function with value 1 if $\{\cdot\}$ is true and 0 otherwise. We also see a behavioural difference between UCT and the baselines; UCT keeps more water levels near their target, which shows that it utilises the given bounds to save costs.

Conclusions and Future Work

The control of pumping stations is a critical task that requires large amounts of power. With the expected increase of price volatility due to an increased mix of renewable energy sources, current controllers will become costly to operate. This paper investigated the potential of AI planning techniques to improve cost effectiveness of pumping station control by allowing the controller to reason about uncertainty in energy prices, thereby also contributing to lower peak loads for the energy network. The paper detailed how an online planning algorithm, UCT, can be applied to this domain, which involves the formulation of a generative model, as well as a number of domain-specific extensions of UCT. We performed an empirical evaluation using the VRNK polder system in the Netherlands. The proposed generative model was compared to the industry standard and is found to be sufficiently accurate for purposes of online planning. Moreover, our proposed application of UCT shows a marked improvement over the current industry standard algorithms.

While our evaluation suggests that large savings could be possible, it is only performed in a simulated environment. In future research we intend to run a field evaluation where our controller is used for the actual management of the VRNK polder system. When applying the proposed UCT solution to a larger number of pumping stations in a polder system, the action space may become too large to handle properly. In this case, decentralising the problem to a multiagent setting allows it to be more scalable. An approach such as FV-POMCP [Amato and Oliehoek, 2015] promises favourable results and can be investigated further [Kanters, 2015].

Acknowledgements

Frans Oliehoek is funded by NWO Innovational Research Incentives Scheme Veni #639.021.336.

References

Amato, C., and Oliehoek, F. A. 2015. Scalable Planning and Learning for Multiagent POMDPs. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, Texas, USA.*, 1995–2002.

Aström, K. J., and Murray, R. M. 2010. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press.

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

Browne, C.; Powley, E. J.; Whitehouse, D.; Lucas, S. M.; Cowling, P. I.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; and Colton, S. 2012. A Survey of Monte Carlo Tree Search Methods. *IEEE Trans. Comput. Intellig. and AI in Games* 4(1):1–43.

Buienradar. 2015. Buienradar. `http://buienradar.nl`. [Accessed 2015-09-15].

Cantoni, M.; Weyer, E.; Li, Y.; Ooi, S. K.; Mareels, I.; and Ryan, M. 2007. Control of Large-scale Irrigation Networks. *Proceedings of the IEEE* 95(1):75–91.

De Nijs, F.; Spaan, M. T. J.; and De Weerdt, M. 2015. Best-Response Planning of Thermostatically Controlled Loads under Power Constraints. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, Texas, USA.*, 615–621.

Deltares. 2015a. Control NEXT. `https://publicwiki.deltares.nl/display/CTRLN/Control+NEXT+home`. [Accessed 2015-09-15].

Deltares. 2015b. SOBEK. `https://www.deltares.nl/en/software/sobek/`.

Driankov, D.; Hellendoorn, H.; and Reinfrank, M. 2013. *An Introduction to Fuzzy Control*. Springer Science & Business Media.

Edwards, K. 2000. Manning's n Coefficients for Open Channel Flow. `http://www.lmnoeng.com/manningn.htm`. [Accessed 2015-09-15].

Finnsson, H., and Björnsson, Y. 2010. Learning Simulation Control in General Game-Playing Agents. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, Atlanta, Georgia, USA*.

Kanters, T. V. 2015. Energy- and cost-efficient pumping station control. Master's thesis, Informatics Institute, University of Amsterdam.

Ketter, W.; Peters, M.; and Collins, J. 2013. Autonomous Agents in Future Energy Markets: The 2012 Power Trading Agent Competition. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, Bellevue, Washington, USA*.

Kocsis, L., and Szepesvári, C. 2006. Bandit Based Monte-Carlo Planning. In *Machine Learning: ECML 2006, 17th European Conference on Machine Learning, Berlin, Germany, Proceedings*, 282–293.

Manning, R.; Griffith, J. P.; Pigot, T.; and Vernon-Harcourt, L. F. 1891. On the flow of water in open channels and pipes. *Transactions of the Institution of Civil Engineers of Ireland* 20:161–207.

Mott, R., and Wagenaar, J. 2009. *Toegepaste stromingsleer*. Pearson Education.

Nationaal Watertraineeship. 2015. Texel Waterveilig Schade Schattingen. `https://sites.google.com/site/texelwaterveilig/resultaten`. [Accessed 2015-09-15].

Nelen & Schuurmans. 2013. EneryApp: Slimmer pompen op duurzame energie.

Panagopoulos, A. A.; Chalkiadakis, G.; and Jennings, N. R. 2015. Towards Optimal Solar Tracking: A Dynamic Programming Approach. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, Texas, USA.*, 695–701.

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

Qin, S. J., and Badgwell, T. A. 2003. A Survey of Industrial Model Predictive Control Technology. *Control Engineering Practice* 11(7):733–764.

Rogers, A.; Ramchurn, S. D.; and Jennings, N. R. 2012. Delivering the Smart Grid: Challenges for Autonomous Agents and Multi-Agent Systems Research. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, Toronto, Ontario, Canada.*

STOWA. 2015. Meteobase. `http://meteobase.nl`. [Accessed 2015-09-15].

Taylor, C. W.; Venkatasubramanian, M. V.; Chen, Y.; et al. 2000. Wide-area Stability and Voltage Control. In *Proc. VII Symp. Specialties Electr. Oper. Expansion Planning*, 21–26.

Te Chow, V. 1959. *Open Channel Hydraulics*. McGraw-Hill Book Company, Inc; New York.

TenneT. 2015. TenneT Imbalance Prices. `http://www.tennet.org/bedrijfsvoering/Systeemgegevens_afhandeling/verrekenprijzen/`. [Accessed 2015-09-15].

Triple. 2015. The Balance of Power - Flexibility Options for the Dutch Electricity Market.

Walraven, E., and Spaan, M. T. 2014. A Scenario State Representation for Scheduling Deferrable Loads under Wind Uncertainty. In *Annual Workshop on Multiagent Sequential Decision Making Under Uncertainty*, volume 46, 1273–1274.

Würzburg, K.; Labandeira, X.; and Linares, P. 2013. Renewable Generation and Electricity prices: Taking Stock and New Evidence for Germany and Austria. *Energy Economics* 40:S159–S171.