Teaching a Robot the optimal operation of an Electrical Energy System with high integration of renewable energies

Ruben Chaer
Facultad de Ingeniería
Universidad de la República
Administración del Mercado Eléctrico
Montevideo, Uruguay
rchaer@fing.edu.uy

Vanina Camacho
Administración del Mercado Eléctrico
Montevideo, Uruguay
vcamacho@adme.com.uy

Ximena Caporale
Facultad de Ingeniería
Universidad de la República
Administración del Mercado Eléctrico
Montevideo, Uruguay
xcaporale@adme.com.uy

Juan Felipe Palacio
Administración del Mercado Eléctrico
Montevideo, Uruguay
fpalacio@adme.com.uy

Pablo Soubes
Administración del Mercado Eléctrico
Montevideo, Uruguay
psoubes@adme.com.uy

Damián Vallejo
Facultad de Ingeniería
Universidad de la República
Administración del Mercado Eléctrico
Montevideo, Uruguay
dvallejo@adme.com.uy

Ignacio Ramírez
Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
nacho@fing.edu.uy

I *Abstract*—**This work shows different strategies for a Robot to learn the optimal operation of a diverse electrical energy generation system including resources such as thermal, hydroelectric, wind, solar generators and energy accumulators. The large number of variables in these systems results in a huge state space. Thus, computing an explicit representation of the cost function over said space, which is at the heart of most current optimization methods, becomes infeasible. The strategies presented here aim at solving the aforementioned problem by learning an implicit representation of the cost function over the state space. Another key idea is to keep the complexity of the representation at a minimum, in order to obtain a solution which captures the most relevant characteristics of the cost-to-go of the system, with the least possible parameters.**

***Index Terms*—Energy, Optimization, Dispatch, Approximate Dynamic Programming, Optimal Policy Learning.**

II Introduction

The optimal dispatch of energy systems can be posed as a *dynamic stochastic optimization* problem. In principle, the optimal operation of such system can be obtained using the classical algorithm, known as *Bellman's Recursion* [1]. That work also introduces what is known as *Bellman's Curse of Dimensionality,* which establishes that the computational cost of the Bellman's Recursion algorithm increases exponentially with the dimension of the state space of the system, thus quickly becoming infeasible.

The advent of renewable energy sources and the diversification of electrical systems has resulted in a significant increase in the number of additional state space variables. As mentioned, this poses a serious problem to the current solutions in use by governments to optimize their power operation. The case of Uruguay is no different: its software, known as SimSEE (*Simulación de Sistemas de Energía Eléctrica*) [2], has been in use and in active development for several years by the state-run ADME (*Administración del Mercado Eléctrico*).

The current implementation of SimSEE already incorporates strategies to alleviate the computational burden. In particular, it is capable of solving the problem at three different time scales: long-term, to estimate the future balance between offer and demand; mid-term, to program the use of the water reservoir and the fuel import schedule, and short term, for the dispatch of the different units and the exportable energy blocks and their minimal prices, in a probabilistic manner, with daily and hourly details for the next 168 hours. SimSEE also includes the ability to approximate the state space with different number of dimensions by means of statistical methods (dimensionality reduction). This allows for state space representations of different dimensions for different time scales, so that the Curse of Dimensionality can be kept at bay.

However, even such techniques are insufficient for coping with the significant increase in state space variables due to the recent diversification of the Uruguayan power grid. This work presents the first results in an effort to overcome these difficulties by switching to new state space and cost function representation paradigms beyond linear dimensionality reduction and uniform quantization of state space variables. We have implemented these techniques directly into SimSEE and tested them on a simplified state space model; this simplification is done mainly so that the results can be easier to interpret and we can gain intuition on the properties of the new models.

The rest of the document is organized as follows. Section II defines the general problem and notation. Section III describes the experimental setting in detail, Section IV shows and discusses our preliminary results, and Section V draws some conclusions from them.

This work is based on the results of the research project "Planificación de inversiones con energías variables, restricciones de red y gestión de demanda", made in 2019-2020 with Uruguay's Fondo Sectorial de Energía of Agencia Nacional de Investigación e Innovación (ANII) funding.

III BACKGROUND

Below we provide a brief introduction to the problem of optimal operation and, in particular, to the basic methodology behind the resolution of said problems, known generally as Approximate Dynamic Programming. See [3,4,5] for more.

We represent the dynamics of a discrete-time as follows:

$$X_{k+1} = f\left(X_k, u_k, r_k, k\right), \qquad (1)$$

where k is the time step, X_k is the state variable at time step k, r_k represents the non-controllable inputs (e.g. rain, wind, etc.), and u_k is a control vector (e.g. power dispatched by each generation unit). We define the cost incurred at a given time step as:

$$ce_k = ce\left(X_k, r_k, u_k, k\right) \qquad (2)$$

Let $J\left(X_s, k+1\right)$ represent the expected value of the optimal future operation beginning at state X_s. The *Optimal Operation Policy* is the one that minimizes the expected value of the sum of (2) and $J\left(X_s, k+1\right)$

$$J\left(X,k\right) = \left\langle \begin{array}{l} \min\limits_{u} \left\{ ce\left(X_k, u_k, r_k, k\right) + J\left(X_s, k+1\right) \right\} \\ \qquad @ \left| \begin{array}{l} u \in \Omega\left(X_k, r_k, k\right) \\ X_s = f\left(X_k, u_k, r_k, k\right) \end{array} \right. \end{array} \right\rangle_{r_k} \qquad (3)$$

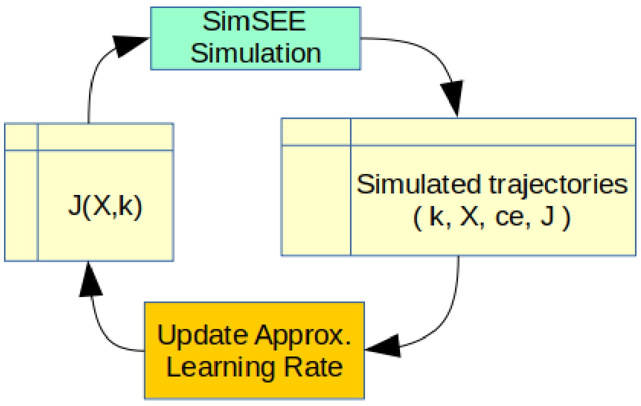
We assume that r_k is known at the beginning of the time step k, so that it is possible to solve the optimization problem of that time step and thus estimate the expected value by averaging the solutions obtained from pseudo-random ensembles of r_k.

In principle, the Bellman's Recursion allows for the calculation of $J\left(X_s, k+1\right)$ for any given state X_s and for all time steps k by repeatedly solving the minimization in (3). However, most often this minimization does not have an analytic form and must be solved numerically. Moreover, since the state space is usually continuous, only a finite number of points can be evaluated. The traditional approach is to uniformly quantize the state space. However, the number of points in the quantization grid grows exponentially as the dimension of the problem increases, turning strategies such as the one described above impractical.

Different techniques of simplification of the model have been developed to be able to solve the problem in an approximated manner [4,5,6]. These techniques include, for example, representing the problem at different time scales, and/or representing the state space with a reduced number of variables, as is currently done in SimSEE.

The method proposed in this work consists of building an implicit representation of the value function $J\left(X, k\right)$. More specifically, we train a Neural Network model on pairs of state-value samples obtained via simulations. The model is constructed iteratively from an initial representation of $J\left(X, k\right)$, which provides the dynamics for the simulation performed to train the next representation. The preceding iterations are performed until the representation converges. This procedure is depicted in Fig.1.

There are two important parameters that control the aforementioned algorithm: one is a *learning rate*, which defines the influence of past and new samples in a new estimation of $J(X,k)$, and the number of steps n over which the system is



SimSEE Simulation
Simulated trajectories (k, X, ce, J)
Update Approx. Learning Rate
J(X,k)

Fig. 1: The learning loop.

simulated. In the latter case, the algorithm is usually categorized as being one of Time Differences of order n, or TD(n).

For n=1, TD(1), only the evolution during one time step is considered subject to the system's dynamics. After that time step, it is supposed that the state of the system jumps to any new state following different exploration strategies not necessarily bound to (1). On the other extreme, for an horizon of N time steps, the TD(N) considers the whole chronicle of stats and values up to N according to the system's dynamics. Once an operation policy that is close to the optimum is attained, following through with the system dynamics has the advantage that the system states are visited according to their corresponding probability. On the other hand, at the beginning of the learning process, it is preferable to explore all the state space because the probabilities are not known in beforehand. Thus, a good strategy is to begin with TD(1), and to increase n as the iterations progress and the dynamics are more realistic.

At this point, we note that, during operation, the system needs to solve not one, but many instances of the aforementioned algorithm, one after another. As the system progresses through time, it is to be expected that consecutive solutions do not differ very much. We can thus exploit this by allowing our method to use the previous solution as a starting point for running the next problem; this is called a *warm restart*, a widely used strategy in these situations.

IV THE EXAMPLE

The purpose of this work is to show the first experiments in learning techniques for a Robot intended to operate an electrical energy generation system in an optimal form. As first trial system, a simplification of the Uruguayan system was selected. The simplification includes the real model of the hydroelectric plant associated with the country's larger reservoir with a storage capacity of full generation of 3 months, Uruguay's electrical demand and fuel-fired thermal plants.

One of the state variables is the stored volume in this reservoir. The inflow hydraulic streams to the lake are modeled with the stochastic model used for the real operation. This model has a state variable which combines information from the state of the central's basin and the temperature in the surface of the Pacific Ocean which determines the El Niño and La Niña phenomena, which imply for Uruguay, bias in more or less rain respectively. The possibility of exporting surplus energy to the region for 25 USD/MWh was also modeled. For a description of the Uruguayan electrical system see [7].

This simple system has the virtue of having non-linearities represented as the one introduced by the variation of the effective jump of the hydroelectic plant for the effect of the discharged flow, as well as the non-convexity imposed by the type of spilling of the plant that starts at a given height of the dam being impossible to spill below said height.

Besides, because the system only has two state variables, it allows to graphically visualize the function $J(X,k)$ for each value of the time step k. Fig.2 shows an example of such visualization. As it can be appreciated, the value function $J(X,k)$ shows high values for low value of x_1 (bottom of the lake) and for low values of x_2 (more probability of low flows). Fig.2 shows that in this case the function $J(X,k)$ is not convex so the problem will not be properly handled techniques such as SDDP [3].

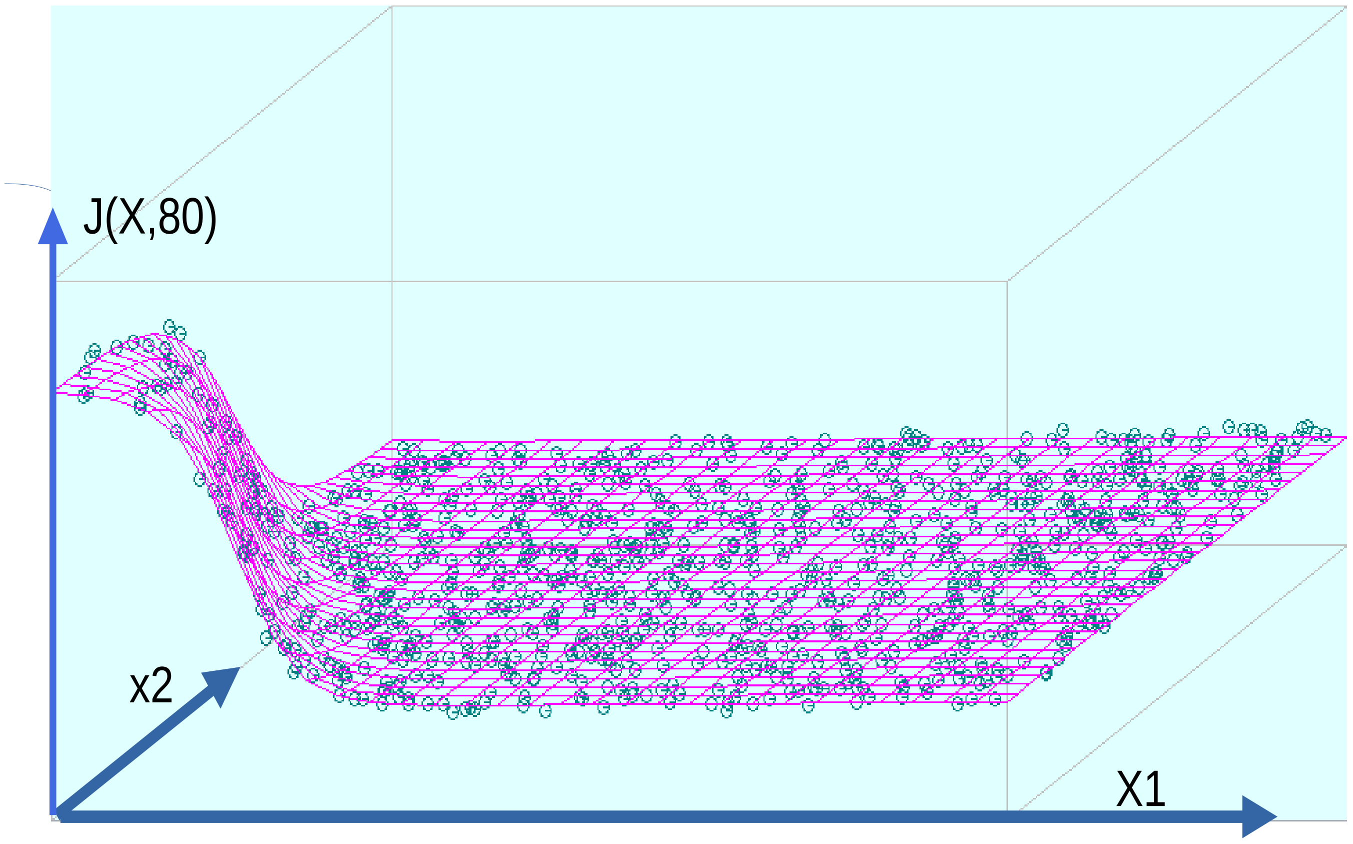


Fig. 2: Value function approximation of the model 2thPSmax at iter. Nro 200 for k = 80.

Given the system, 100 iterations were executed, with 1000 trajectories each one, of the learning loop using TD(1) and with the following types of structure for the Neural Network:

- 2th1PSmax. It is made up from two neurons with a non linear function tanh in layer 1 and one neuron in the output layer of the type Ponder Soft Max, which corresponds to the linear combination of the inputs pondered by the coefficients that the Softmax operation returns with the same inputs.
- 2th1th. It is made up from two neurons with tanh saturation function in layer 1 and one neuron with tanh as well in the output layer.
- 4th1th. It is made up from 4 neurons with tanh saturation function in layer 1 and one neuron with tanh as well in the output layer.

In all of the cases, the inputs are the two state variables explained before and the output is the estimated value of $J(X,k)$.

The system solved with Bellman's Recursion is done using a 10 value discretization of the state variable x_1 (reservoir's volume) and 5 values of the state varible x_2 (hydrological state variable). This discretization carries a spreadsheet of $J(X,k)$ of $10\times 5=50$ values for each time step k. Table 1 sums up the complexity of each structure of approximation of $J(X,k)$ as well as the amount of parameters to train on each time step.

Model	**Complexity**
Classic	50
2th1PSmax	8
2th1th	11
4th1th	17

Table 1 -Number of parameters to train according to model per time step

The models of networks have two additional parameters to calibrate the offset and scale of the output.

V Results

Fig.3 shows the cost-to-go of the first 100 iterations of the learning loop of each trainable model and the horizontal dashed line next to the 10 MUS$ corresponds to the result of the operation with Bellman's Recursion. This value can be considered as the minimum reachable, even though lower values could be obtained due to the system's discretization of the state space for the resolution of Bellman's Recursion it is an approximation to the problem.

Once the models with 100 iterations of 1000 independent realizations of the stochastic processes each are trained, a 100 realizations simulation was performed starting the system from a given initial state and evolving in each one of the 100 realizations respecting the system's dynamics in a way that the obtained value for the sum of the costs is comparable between the different models and with the classic solution.

The training begins in all models with the initial value $J(X,k)=0; \forall(X,k)$ which leads to give a null value to the stored water given that the value for the future of the stored water is given by $-\frac{\partial}{\partial x_1} J(X,k)=0$.

This leads to that in the first iteration, water is used before the thermal plants which ends up emptying the lake and imposing then that the most expensive thermal plants are dispatched, this could be avoided saving water for critical moments. The cost of this operation is 350 MUS$ in the two simulated years.

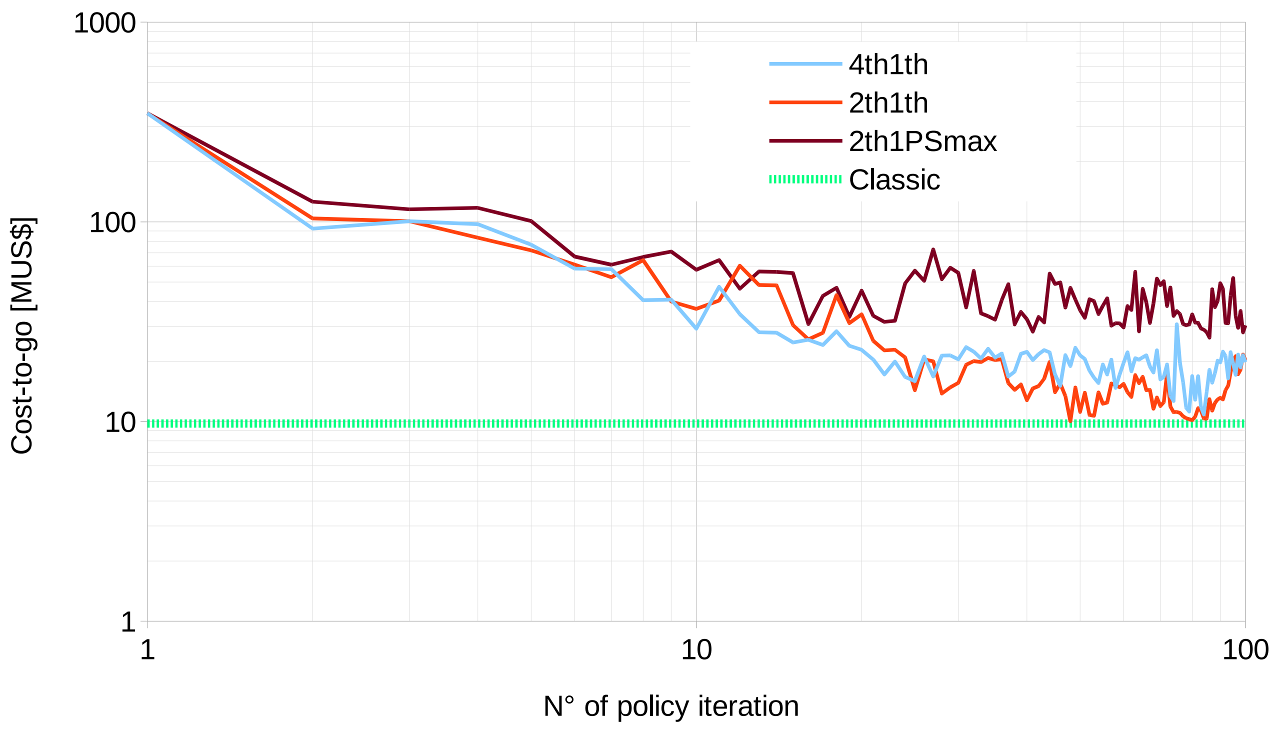


Fig. 3: Convergence of the first 100 iterations of the learning loop for each model.

As it can be observed the model with less parameters 2th1PSmax leads to an operation that in the best points of the training gets to values inferior to 30 MUS$. Models 2th1th y 2th4th both eventually get in some point of the training to the 10 MUS$.

The trainings of Fig.3 are realized with a 0.7 Learning-Rate. This means that the new information on each iteration of the loop of Fig.3 is pondered with a factor of 0.7 and the previous approximation $J(X,k)$ with the weight 0.3.

It can be observed that in the training, after the 20 iterations the convergence is slow and noisy. To try the effects of the Learning Rate, the approximation of iteration 80 of the 2th1th model was considered and 100 more iterations were performed for the same model but with 0.3, 0.7 and 0.95 Learning Rates. Obtaining the results that Fig.4 which shows that the Learning Rate should decrease as the learning process advances filtering the noise which is inherent to the stochastic processes with which the next information of each simulation group is obtained.

VI Conclusions

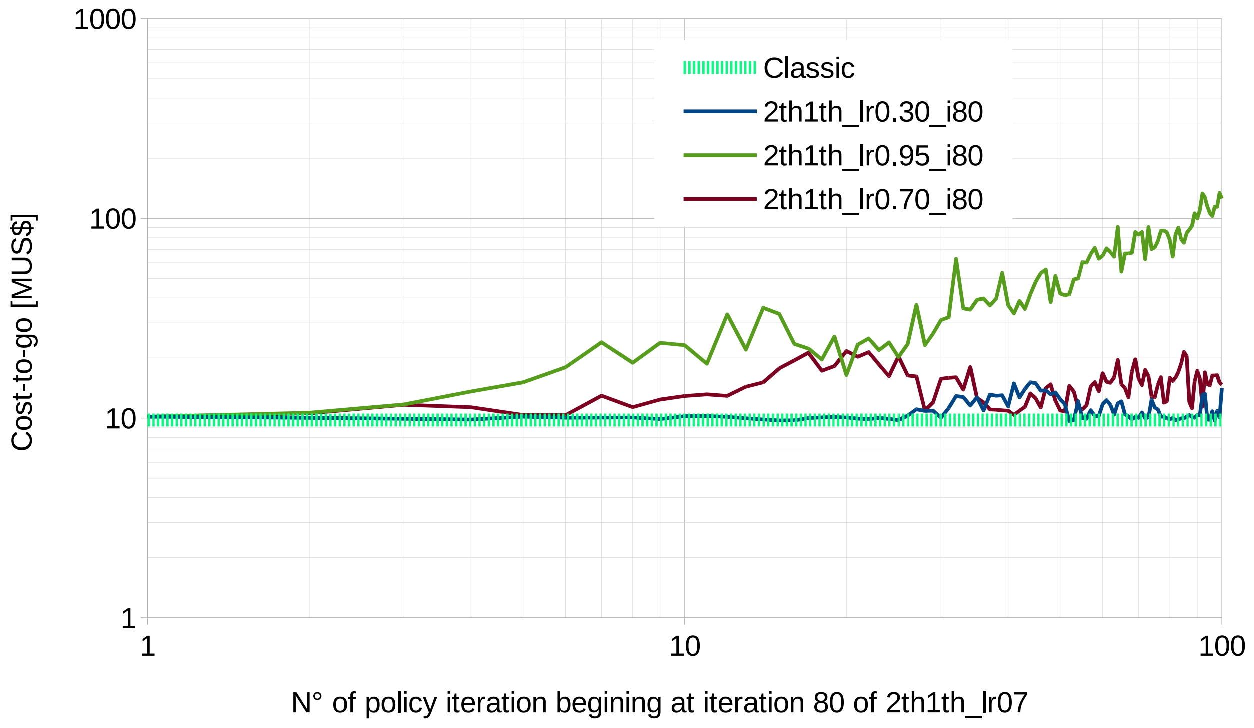


Fig. 4: Effect of the learning-rate after the iteration N°80

We have presented preliminary results of the incorporation of a new learning strategy into the SimSEE platform, applied to a simplified version of the Uruguayan electro-energetic system. Under this scenario, we were able to compare our initial results with those obtained using the classic approach already in production in SimSEE, based on Bellman's Recursion. In particular, we were able to obtain results comparable to the traditional approach (using 50 parameters) with as few as 11 model parameters. It should also be noted that critical parts of the implemented algorithm are parallelizable and easily reusable in the real operation process, both greatly simplifying the deployment of our proposed solution on a real-time operation of the system.

VII Disclaimer

The content of this article is entirely the responsibility of its authors, and does not necessarily reflect the position of the institutions of which they are part of.

References

1 DYNAMIC PROGRAMMING Bellman , Princenton University Press. 1957.

2 Open Source software platform for Simulation of Systems of Electrical Energy (SimSEE). https://simsee.org

3 Multi-stage stochastic optimization applied to energy planning.pdf Mathematical Programming. M. V. F. PereiraL. M. V. G. Pinto May 1991, Volume 52, Issue 1–3, pp 359–375

4 Warren B. Powell. Approximate Dynamic Programming. WILEY 2011.

5 Dimitri P. Bertsekas, John N. Tsitsiklis. Neuro-Dynamic Programming. Athena Scientific 1996.

6 Reinforcement Learning: An Introduction. February 28, 2018 Richard S. Sutton and Andrew G. Barto.

7 R. Chaer et al. Handling the intermittence of wind and solar energy resources, from planning to operation. Uruguay's success. 36th USAEE/IAEE North American Conference, Washington, DC, September 23 - 26, 2018.