GRIDA: INTRODUCING A SELF-LEARNING ARTIFICIAL INTELLIGENCE FOR AUTONOMOUS NETWORK PLANNING

Age VAN DER MEI
Duinn – The Netherlands
Age.vandermei[at]duinn.nl

Todor KOSTOV
Duinn – The Netherlands
Todor.Kostov[at]duinn.nl

Jan-Peter DOOMERNIK
Enexis – The Netherlands
Jan-Peter.Doomernik[at]enexis.nl

ABSTRACT

This paper presents a self-learning artificial intelligence for grid planning. The self-learning means that no expert input data is required, which is a large advantage in electricity distribution planning with little expert data for optimal grids. The performance of the self-learning algorithm, nicknamed Grida, is comparable on small planning problems and slightly better on larger problems compared to the widely used ORtools however it does not (yet) provide optimal solutions.

INTRODUCTION

The **subject** addressed is the application of self-learning artificial intelligence (AI) in planning. The advanced self-learning artificial intelligence is a novel self-reinforcement neural network. A self-learning neural network that can learn how to plan a network, while taking into account relevant contextual factors, would benefit planning and especially the planning in the face of high complexity, such as multi-energy networks.

The **problem** is the planning of reliable and affordable networks that facilitate the energy transition and can incorporate emerging technologies, while facing uncertainty. Bi-directional, multi-energy networks with both decentral and central production, and home and grid storage present enormous planning complexity. The planning complexity and problem size means it is not solvable by a provable, mathematical optimal method. In mathmetical terminology, at least two problems which hare NP-hard problems have to be solve jointly for an optimal solution. New methods and tools are required to assist human planners. We seek to address the problem by building, testing and introducing a first self-learning artificial intelligence: Grida.

RESEARH QUESTION

The **research question** is stated in three steps: what is the task environment for electric distribution that is facing the artificial intelligence? **How can an artificial intelligence neural network learn how to design networks without expert input examples? What design strategies would a self-learning artificial intelligence come up with given the competing objectives for cost, efficiency and reliability?** The first research questions is seen as valid and relevant as no application of self-learning artificial intelligence is yet available or published for grid planning.

The second question is relevant as it allows for comparison to different design strategies: radial, loop and network systems, messed or not etc, and compare this to what a self-learning AI comes up with.

METHOD

The method of addressing this problem is the application of **self-learning neural network**. More precisely, a self-reinforcement, unsupervised, context-dependent deep neural network. Currently, such an approach represents state-of-the-art AI in advanced problem solving under uncertainty, now beginning to be applied in games [1], visioning and unsupervised training of robots. The presented self-learning 'learns' to solve for the location and sizing of producers and connecting producers to consumers. The performance of the self-learning network is compared to a heuristic approach consisting of two widely used algorithms: the Agglomerative Hierarchical Clustering (AHC) to identify the subgrids, and (2) the ORTool Library from Google to solve the traveling salesman problem in each of the resulting grids.

Artificial intelligence entails 'the designing and building of intelligent agents that receive input and instructions from the environment and take actions that affect that environment'. 'An intelligent agent is one that acts so as to achieve the best outcome or the best expected outcome in case of uncertainty' [2]. Authors endeavour here to apply artificial intelligence to the task of network planning, architecture and design, without the need for labelled expert data. This paper applies a rational agent to the problem of designing a microgrid.

In **reinforcement learning** the algorithm learn from a series of reinforcements, being rewards or punishments. **Unsupervised** means that no prior or historical data is required for the algorithm to learn how to master the activity, in this case the planning of energy networks. This is an advantage as consistent and reliable network-state data is difficult to obtain and apply in a useful manner for training a supervised algorithm.

The **context-dependent** implies that the algorithm can learn how to take into account contextual factors that are related to input variables and constraints. This is deemed important as there are large amounts of tacit knowledge involved in real-life planning.

The **problem definition** of network planning is broken down into to parts: the production and the connection. Both better known as the travelling salesman problem (TSP) and a metric facility location problem (MFLP).

The **objective function** is to minimise the total investment of production, transport and distribution while providing a minimum redundancy score. This is achieved by finding the subgrids with shortest TSP connecting them. Then we move to position the producers closest to the supplied grids, but this is calculated outside of the network. The objective function is defined in detail on the next page in yellow.

Learning is performed on 100 thousand simulated grids, which resemble real-life electricity demand. We modelled the electricity demand by fitting a gaussian mixture model to real world data from the municipality of Groningen. Then we sampled random points from the 20x20 grid. The smallest grid contains 7 points and the largest contains 25 points.

Testing and benchmarking of the self-learning algorithm, is performed on 20 real-life demand situations and compared to the real-life situation as well as ORtools.

RESULTS

The **results** are presented in three steps: (1) the definition of the task environment that is relevant for electric distribution, (2) the structure of the self-learning agent, (3) visualizing the results, (4) benchmarking of the the self-learning agent.

The **task environment** is the problem description that faces the agent. For electric distribution the authors define the task environment as: fully observable, single-agent, deterministic, episodic, sequential, static, continuous states, discrete actions and a finite time horizon. Using this problem description, the next step is to describe the representation of the problem. The geographical area involved, including the location and size demand and restrictions are projected on a linear rectangular grid. This simplified representation and reduces computational resources required. This builds upon earlier creation of AI for network planning [3].

Self-learning artificial intelligence for network planning

The **structure** of the approach can be summarized as following steps:

1. Determine demand (load, geographic location)
2. Set performance function (efficiency, costs, reliability)
3. Determine input variables (relevant investment, costs and redundancy metrics)
4. Generation of task environments for learning (sample a grid from the demand distribution that we approximated using a mixture of gaussians)
5. Creating of self-learning agent using deep neural network and a policy network
6. Simulate random problem instances on a 20 by 20 grid for training purposes.
7. Training is being done using an reinforcement algorithm to minimize expected cost (using Monte Carlo sampling of the cost function).
8. Reinforcement of the neural network solutions that yield lower cost of the solution (i.e. these solutions are reinforced most in and used for further training).
9. Train the self-learning agent by letting it compete against other (similar) agents
10. Evaluate the training results and checking validity (using reinforcement, i.e. reward and punishment, to allow it to lean better actions to solve the problem. The algorithm has a build functions to evaluate the results of the two agents playing)
11. Benchmark the self-learning algorithm versus current widely used tools.
12. Apply the trained agent to solve a real-life case (testing is done on known problems and mathematical challenges for which the solution is known).
13. Visualize the results.
14. Evaluate approach and re-iterate.

The chosen **structure** of the self-learning agent is a self-reinforcement, unsupervised, context-dependent deep neural network. The self-learning agent is integrated, meaning it solves for the two problem definition in one step. The deep neural network is trained using policy gradient to produce multinomial distributions (positions of the distributors) and k binomial distributions. The training is done using gradient ascent, maximizing the reward, in this case the number of wins the agent scores against a competing agent. The policy gradients learn a policy that produces an action based on the current state (the observed network). The training of the intelligent agent starts with training against a static agent in order to produce viable

results. When the agent is sufficiently trained (wins 100%), it then commences to start training against a second agent and train them jointly.

The objective function of the self-learning agent is defined as: The objective function is defined as:

Expectation $(\text{cost}(P_1, P_2, \text{TSP}(P_1), \text{TSP}(P_2)) = \textit{producercost}(\text{SUM}(d_i) + \text{SUM}(d_y)) * \alpha_{\text{producer}} + \text{const}_{\text{producer}} * 2 + (|P_1| + 1) * \text{const}_{\text{connectionfactor}} + \text{SUM}(d_i) * \alpha_{\text{connection}} + (|P_2| + 1) * \text{const}_{\text{connectionfactor}} + \text{SUM}(d_y) * \alpha_{\text{connection}})$.

for d_i being the corresponding demands for points in P_1 and d_y being the corresponding demands for points in P_2 . The main aim is to minimise the objective function, which will be denoted as *cost*(.). This implies that even for small instances to find optimal solution the number of possible combinations is too large to be handled efficiently:

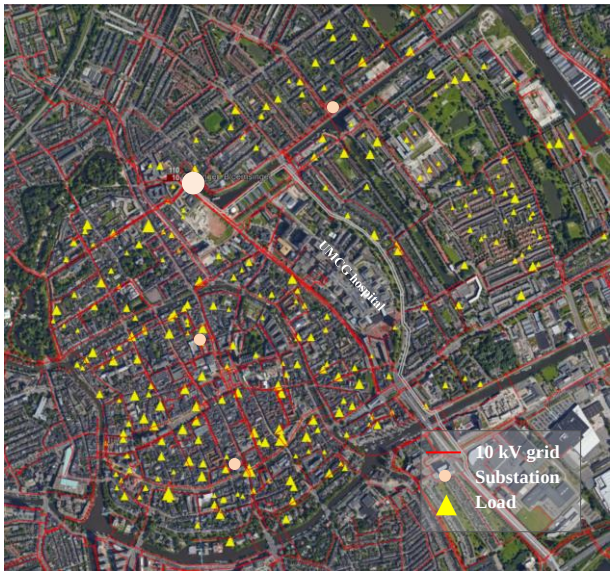
Possible partitions $|partitions| = 2^{n-1}$. We can model the problem as a game, where an agent A makes a series of decisions (first partition decisions) and later based on the chosen partitions it takes actions to perform the connection.

The proposed model uses two neural reinforcement networks to produce an approximately optimal solution to the given problem. The first network is used to produce a partition (left side of the image), while the second network is used to calculate an approximate optimal circle (approximation of the TSP problem) within the predicted partitions (see orange box on the right part of the image).

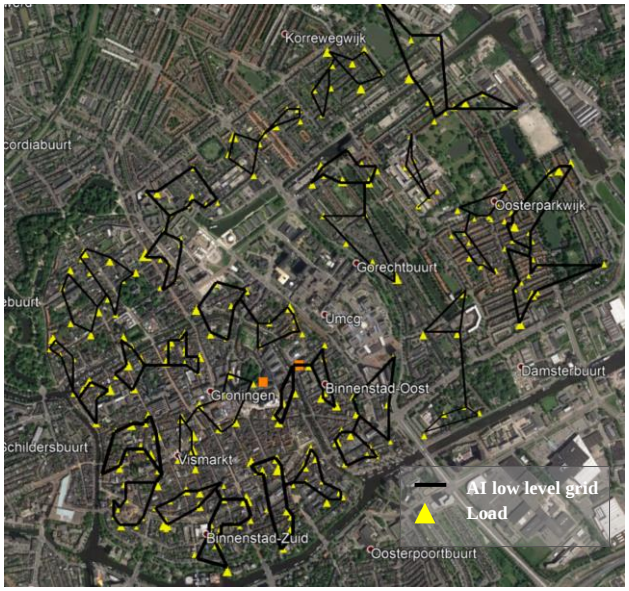
In the following sections we describe the architecture of the first model. The second model is based on Neural Combinatorial Optimization with Reinforcement Learning, by Bello et al. The current architecture replaces the LSTM encoder and decoder of Bello at all and replaces them with the encoder and decoder from Attention is all you need

As in state-of-the-art networks that are used to approximate NP-Hard combinatorial problems, we modelled our network using a encoder-decoder structure, which is described in the following sections.

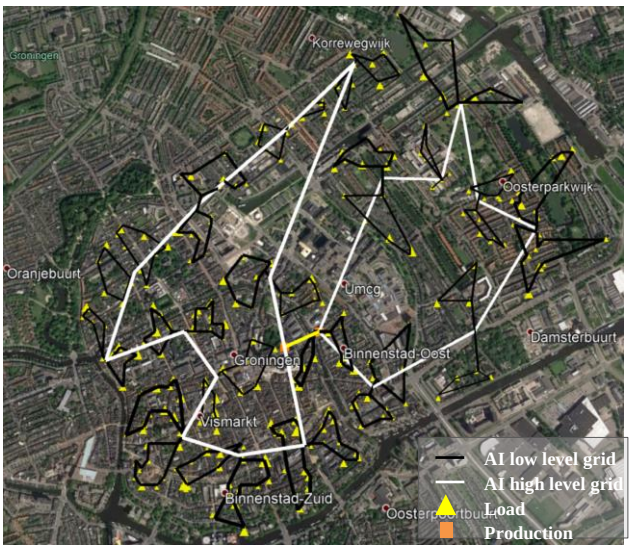
Schematic 1: current grid in Groningen (in red)



Schematic 2: artificial intelligence generated design (lower-level grid design)



Schematic 3: artificial intelligence generated design (lower level and higher-level grid design)



Comparison of self-learning algoritm

Comparing the self-learning algorithm, Grida, with the benchmark Agglomerative Hierarchical Clustering (AHC) and the ORTool Library from Google yield comparable results, and in some cases slightly better results for grid length and costs.

Table 1: benchmark algorithms versus Grida

Grid ID	Grida	Benchmark AHC + ORtools	Grida + Bello net to refine TSP
1	3223.56	2953.92	3152.25
2	3935.81	3879.59	3880.57
3	7391.66	7286.81	7286.81
4	5225.30	5158.85	5213.71
5	3464.81	3464.81	3464.81
6	10561.52	10558.62	10558.62
7	10824.24	10317.95	10792.04
8	6145.91	6190.39	6142.09
9	1360.43	1364.45	1355.45
10	2500.07	2500.07	2500.07
11	3112.50	3106.06	3108.20
12	7900.34	7814.89	7819.36
13	3528.47	3528.47	3528.47
14	1314.10	1314.10	1314.10
15	10806.72	10806.72	10806.72
16	4878.05	4996.38	4878.05
17	2984.81	2965.99	2984.81
18	2792.24	2792.24	2792.24
19	1795.70	1789.35	1794.59
Second LVL Agent Input	239871.77	224827.97	232615.89
Second LVL benchmark input	236440.99	229826.76	230305.06

CONCLUSIONS

This paper demonstrates that applicability of self-learning algorithms to network planning is feasible. The algorithm nicknamed Grida provides proof-of-principle and showcases that is can provide comparable performance to current widely used tools. The performance is comparable and better in a number of subgrids.

DISCUSSIONS

Authors hope that the paper will spark a broader application of self-learning algorithms for use in the electricity sector and are small step towards intelligent and self-organising networks.

Acknowledgments

Authors would like to acknowledge the assistance and financial support by Enexis.

REFERENCES

[1] Silver, D. et al, 2017, Mastering the game of Go without human knowledge, Nature, vol. 550, p. 354-359

[2] P. Norvig, S. Russell, 2016, *Artificial Intelligence a Modern Approach*, Pearson, Harlow, England.

[3] Van der Mei, A.J., Doomernik, J.P., *Artificial Intelligence for Microgrid Planning and Operation*, paper 0303, April 18

[4] Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, Samy Bengio, *Neural Combinatorial Optimization with Reinforcement Learning*, 12 Jan 2017

[5] Vaswani, A. et al, 2017, Attention is all you need, *31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA*, 6 dec 2017.