Deep Reinforcement Learning for Plan Selection

Thomas Asikis*, Georgios Touloupas† and Christos Sakaridis‡
repos
ETH Zürich
Zürich, Switzerland
*Email: *asikist@ethz.ch, †gtouloup@student.ethz.ch, ‡csakarid@vision.ee.ethz.ch*

***Abstract*—Cooperation and collaboration between technical systems is a driving factor for research and technological advancement. Several problems pertaining to such systems, such as supply/demand in smart grids or smart city planning, are formulated as combinatorial optimization problems, in which each component of the system, referred to as agent, selects from an individual set of plans, with the goal to optimize a global objective. This is referred to as the *plan selection problem*. This project demonstrates a novel approach for the approximation of near-optimal solutions in such problems by utilizing recent advances in deep reinforcement learning (DRL).**

I. Introduction

Technological advances increase the complexity of everyday life in modern societies. Internet of Things sensors and smart devices keep increasing in numbers daily. By combining such devices and synchronizing their functionalities, several complex problems can now be addressed in a more efficient manner. Such problems are related with the placement of electric vehicle stations in smart city planning [1], sharing economies [2], [3], satisfaction of energy demands and prevention of black-outs in smart grids [3] or cell placement in telecommunications [4]. Several of the aforementioned problems are addressed via the optimization of a global common objective function L, which evaluates the selection of each agent's u plan options matrix $\boldsymbol{C}_u$. An agent is often considered as a real or conceptual autonomous unit, such as a household in smart grids, a community in smart cities or a geographical area in telecommunication cell planning.

There are several application domains, where the cooperation of agents is required to address specific goals. Such systems are called multi-agent systems. The agents are expected to learn and cooperate in a collaborative fashion to optimize a global objective function that satisfies those goals. There are several algorithms that illustrate how cooperation between agents can lead to efficient selection of plans [5], [6], [7], [3]. Such algorithms involve learning or optimization and operate in a decentralized manner. Moreover, in addition to multi-agent systems, there are several algorithms that operate in a centralized manner, where the plan selection problem is formalized as an integer programming problem [8], [9]. Integer programming problems are addressed by a diversity of algorithms, such as dynamic programming [10] and meta-heuristics, e.g. genetic algorithms [4] or particle swarm optimization [11].

A. Problem Formulation

The problem addressed in this project is the optimization, i.e. minimization, of the global objective function $L : \mathbb{R}^{|\mathbb{U}|\times M} \rightarrow \mathbb{R}$ defined as $L(\boldsymbol{C}^*)$, where $\boldsymbol{C}^*$ is a *plan matrix* that contains the selected plan $\boldsymbol{c}_u^*$ for each agent in the form of a real vector. The dimensionality of this matrix is $|\mathbb{U}| \times M$, where $|\mathbb{U}|$ is the total number of agents, M is the (common) dimensionality of the plans and each agent's plan constitutes a row of the matrix. Minimization is achieved when an optimal plan $\widehat{\boldsymbol{c}_u}$ is selected for each agent. A plan matrix $\boldsymbol{C}^*$ is considered optimal, which is denoted as $\boldsymbol{C}^* = \widehat{\boldsymbol{C}}$, when

$$L(\boldsymbol{C}^*) = \min_{\boldsymbol{C}}\{L(\boldsymbol{C})\}. \tag{1}$$

The total number of possible plan matrices is $n = \prod_u |\boldsymbol{C}_{u:,0}|^{|\mathbb{U}|}$. This proves that this problem is NP-Hard, so the global optimum cannot be determined in polynomial time.

This project addresses the above problem by proposing a DRL architecture called *Deep Centralized Plan Optimizer* (DCPO), which identifies a near-optimal solution to the above plan selection problem. There is prior work where deep (reinforcement) learning is utilized for the solution of other combinatorial problems [12] such as optimal graph computation [13], [14], [15] and prediction of popular discussion threads [16]. For planning specific scenarios, there has been recent work that focuses on a Reinforcement Learning (RL) architecture for each agent [17] or a single plan selection [18]. This work, on the other hand, focuses

on distributing DRL architectures over agents. Our main contributions are: (i) formulation of the problem described in *Equation 1* in the RL framework, (ii) the design and evaluation of two novel DRL architectures to solve the problem and finally (iii) a comparison of the proposed DRL architectures with state-of-the art methods, evidencing competitive performance for both of them.

II. Models and Methods

A. Formulation as a Reinforcement Learning Problem

Under the assumption that each user can select among the same number of plans[1] or the plan matrices are padded with NaN-valued plan vectors[2] to be of equal dimensions, then $\boldsymbol{C}_u = \boldsymbol{c}_{u,i,:}$ is a slice of a three dimensional[3] plan selection tensor $\mathbf{C}$.

To redefine the problem stated in 1 in , first some basic concepts related to RL need to be specified. In most agent-based approaches [3], [11], [4], this or similar problems are solved in an iterative manner. Each agent selects a selected plan $\boldsymbol{c}_u^*$ and then changes the selection over the course of time based on the logic of the algorithm. In terms of RL, the selected plan at a given iteration or time step is the state of the agent $s_{u,t} = c_{u,t}^*$. The transition from one state to another is defined as an action scalar a. In the models presented below, each action of an agent at a time denotes the index of the next selected plan $a_{u,t} = c_{u,t+1}^*$. The number of possible actions per agent is equal to the number of agent plans $|\mathbb{A}_u| = |\mathbb{C}_u|$. The aim of the proposed DRL architectures is to learn a near-optimal policy function $\pi : \mathbb{S}_u \rightarrow \mathbb{A}_u$ that selects the action for the next time step $\pi\left(s_{u,t}\right) = a_{u,t+1}$ while trying to indirectly[4] minimize L by maximizing a reward function $rL(\boldsymbol{C}^*)$. Since the network architecture is centralized, the proposed architectures attempt to learn all agent policies at the same time, approximating the optimal actions and states $\boldsymbol{\pi}\left(\boldsymbol{s_t}\right) \rightarrow \widehat{\boldsymbol{a}}_t = \widehat{\boldsymbol{s}}_t$ for all agents.

B. Objective Function and Reward

In the examined setting, we are interested in minimizing the *sample variance* of the set of elements of the vector that constitutes the sum of plans over all agents. More formally, for a given plan matrix $\mathbf{C}$ which comprises agents' plans $\mathbf{c}_u$ for $u \in \{1, \ldots, |\mathbb{U}|\}$, the objective function is expressed as follows:

$$L(\mathbf{C}) = \mathrm{Var}\left(\sum_u \mathbf{c}_u\right). \tag{2}$$

The reward in our RL models is defined based on the value of the objective function in two alternative ways, following the principle that decreasing values of the objective function should result in an increase of the reward. Using subscript t to distinguish iterations, the reward r for the model at iteration t is defined either as

$$r_t = L_{t-1} - L_t \tag{3}$$

or as

$$r_t = \min_{\tau \in \{1,\, \ldots,\, t-1\}} \{L_\tau\} - L_t. \tag{4}$$

C. Deep Reinforcement Learning Models

In this project we tested the variants of two state-of-the-art DRL architectures, the Deep Q-Learning (DQL) algorithm [19] and the Asynchronous Actor-Critic Agents (A3C) model [20]. A brief explanation of each architecture and each variant we evaluated is provided next.

1) Deep Centralized Plan Optimizer: Deep Q-Learning (DCPO: DQN): : The input of this architecture is a vector of plan indexes. The output layer of the DQN is referred to as Q-Layer and is of dimensions $|\mathbb{U}| \times |\mathbb{A}|$. Each $q_{u,a}$ value of the Q-layer is the corresponding q-value for an action-agent pair. The next action for each agent at iteration t is determined as $a_{u,t+1} = \arg\max_a \{q_{u,a,t}\}$.

The loss function of the model is defined as

$$J(\mathbf{Q}; \theta) = \sum_{u,a} \left(q_{u,a,t} - q_{u,a,t}^{\text{target}}\right)^2, \tag{5}$$

where q^{target} is an estimate of the future reward after taking the corresponding action and is computed (based on the next q-values after a subsequent feed-forward pass) as

$$q_{u,a,t}^{\text{target}} = \begin{cases} r_t + \gamma q_{u,a,t+1}, & \text{if } a = \arg\max_a\{q_{u,a,t+1}\} \\ q_{u,a,t}, & \text{otherwise.} \end{cases} \tag{6}$$

Our fully-fledged deep neural network involves a sequence of convolutional layers in the front-end to obtain informative representations of the initial plan vectors, followed by a selection of plans based on the current state and a final bidirectional LSTM module before the Q-layer, as shown in Figure 1.

2) Deep Centralized Plan Optimizer: Asynchronous Actor Critic Agents (DCPO: A3C-DNN): This model was built based on the the work of Mnih et al. [20] with the addition

[1] In this case the number of combinations is maximum: $n = \prod_u |\boldsymbol{C}_{u:,0}|^{|\mathbb{U}|}$

[2] This creates the constraint the users will not be able to select the NaN-valued plans

[3] The third dimension of the tensor can be a discrete or continuous vector representation of a plan. There are cases, where a plan tensor may not have 3 dimensions, because each plan is represented in more than one dimension or a scalar. The current project presents the most common case. Nevertheless, the abstraction to more or less dimensions can be derived with simple modifications to the presented case.

[4] The meaning of indirectly is further discussed in Section IV

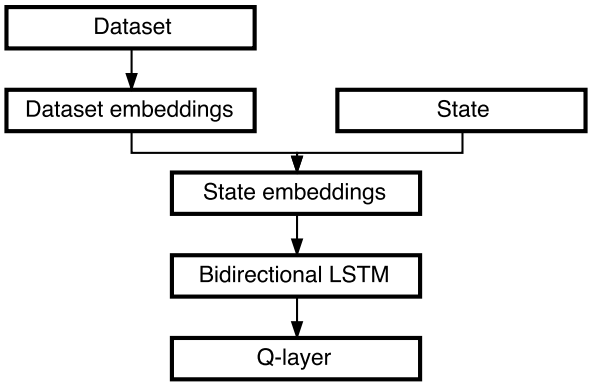

Figure 1. Schematic overview of the architecture of our DCPO-DQN model with LSTM back-end.

of an extra agent dimension in the $P(s)$ layers. The input of the network are the selected plan vectors per agent. The selection of actions based on the probabilities outputted in the final layer was done by sampling a probability distribution per agent. The architecture of the network was 3 convolutional layers that export output channel per agent selected plan convolving on the plan times steps. Increasing the number of convolution layers seemed to increase performance. The advantage calculation is done based Schulman et al. [21]. An LSTM layer is used to capture the temporal difference, whereas the BiLSTM layer is used to re-evaluate the outputs before the policy layers. Both LSTM layers are of dimensionality $[|\mathbb{U}|, 30]$, where the first dimension is the sequence size and the second the number of output features from the convolutions. The architecture is summarized in Figure 2.

D. Baselines

The proposed architecture against was compared against the following baselines:

1) I-EPOS: [3]. I-EPOS is an iterative optimization algorithm that arranges the agents in a tree structure, where each agent is assigned a n number of children agents. The I-EPOS iteration consists of two phases: (i) the forward phase, where each agent choses the selected plan and propagates it along with the selected plans of its children. The agents with children, select the plan that cancels out the variance of selections of the plans received from their children by calculating the total plan variance. (ii) The backward phase in which the parent informs its children whether to keep their plans or change them. The backward phase creates a learning effect in the network.

2) Genetic Algorithm: : An implementation of a genetic algorithm inspired by Jeffrey et all [4], where each gene is an integer mapped to a plan index. The chromosome has

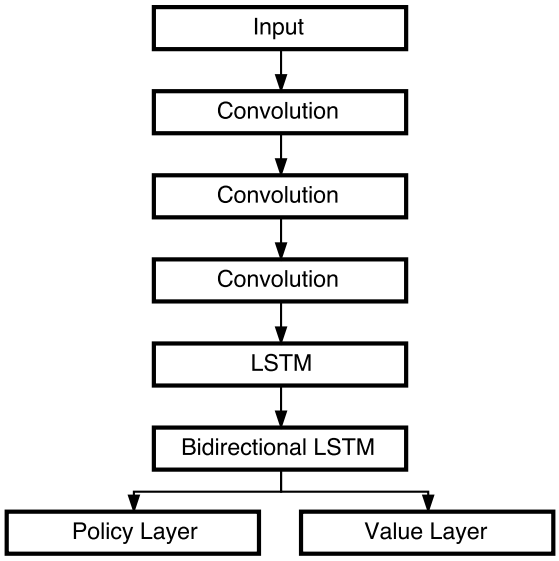

Figure 2. The a3c architecture. The last two layers are fully connected to the output of the BiLSTM.

length $|\mathbb{U}|$. Each gene shows the plan selection of a user and denotes a state $\boldsymbol{s}$ for all the users. The genetic performs crossovers over the chromosomes of the population and via 3 tournament rounds and then selects the chromosome with the minimum variance according to *Equation 2*.

3) Particle Swarm Optimization (PSO): inspired by Laskari et al [11]. In this algorithm, a particle swarm is a group of actors that search the solution space. In this algorithm each particle calculates a solution vector of size equal to $|\mathbb{U}|$. Each element of the solution vector corresponds to the plan selection of an agent. The solution variance is evaluated and at every iteration iteration then each particle changes part of the solution to move closer to the optimal solution. Some particles move randomly to encourage exploration over exploitation.

III. Results

In order to evaluate our proposed DRL models against the existing state-of-the-art approaches, we use the real-world energy consumption dataset *PNW-evening*[5]. Each plan consists of a sequence of values, each of which encodes the amount of energy that is consumed by the respective agent during a fixed-size time interval. The number of time intervals is $M = 144$. *PNW-evening* includes 100 agents and four plans per agent, which renders the computation of the optimal plans *infeasible* on the full dataset. For this reason,

[5]Available upon request at http://www.pnwsmartgrid.org/participants.asp (last accessed: March 2017). We have signed an NDA and so we cannot provide the used data for reproduction of the results. Please contact epournaras@ethz.ch for confirmation.

Table I
COMPARISON ON THE REAL-WORLD *PNW-evening* ENERGY CONSUMPTION DATASET OF THE OBJECTIVE VALUE FOR PLAN SELECTION (IN THIS CASE THE VARIANCE OF AGGREGATED PLANS OVER TIME) THAT IS INDUCED BY VARIOUS METHODS. WE COMPARE ON THE FULL DATASET WHICH INCLUDES 100 AGENTS (*PNW-evening-100*) AND ON A SUBSET OF IT WHERE ONLY 10 OF THESE AGENTS ARE KEPT (*PNW-evening-10*).

	PNW-evening-100	*PNW-evening-10*
Global Optimum	n/a	0.0386
Random Search	1.341	0.0386
Genetic	1.068	0.0386
I-EPOS [3]	1.104	0.0410
PSO	1.288	0.0386
DCPO-DQN-LSTM	1.258	0.0386
DCPO-A3C-deep	1.253	0.0386

in the comparison that we perform in Table I, apart from the full dataset (denoted by *PNW-evening-100*) we further use a smaller subset of it (denoted by *PNW-evening-10*) which contains only 10 agents and hence is amenable to global optimization, so that the resulting optimal solution serves as a reference point for the solutions of our approaches as well as the rest of the methods. Table I reports performance of the state-of-the-art I-EPOS framework [3], a genetic algorithm ("Genetic") and a particle swarm optimization approach ("PSO"). Moreover, we have implemented a soft baseline ("Random Search"), which randomly selects a fixed number of plan matrices from the solution space and performs an exhaustive search over these matrices to find that with minimal induced objective value. We use 10^5 randomly selected plan matrices for *PNW-evening-100* and 2×10^4 for *PNW-evening-10*.

On *PNW-evening-100*, both our proposed models A3C-deep and DQN-LSTM outperform PSO and Random Search significantly. Even though both the state-of-the-art I-EPOS [3] approach and Genetic compare favorably to our DRL models, we would like to note that the networks run for a short period of time. The model architectures and hyper parameters are not tuned, so a proper model parametrization and runtime is expected to yield better results.

IV. DISCUSSION

In this project we formalized the plan selection problem as a reinforcement learning problem and attempted to solve it with the usage of state of the art techniques. During the implementation and design of the project, we came across several ideas and considerations, which we share below.

A. Centralized vs Decentralized Solutions

We proposed a centralized architecture with the intuition that the the agent synchronization is optimal when a neural network manages the agent choices based on the available data. Centralized solutions are easier to scale out computationally centrally and often are more robust, still they might prove to be privacy intrusive and also may be difficult to scale up to a large number of users [2], [3]. Experiments have proven that the DCPO architectures can be adapted to a smaller scale of agents, but they still have to prove that their end-to-end architecture is able to learn meaningful representation, especially for the PNW dataset.

B. Parametrization and Cross Validation

The run-times for each DCPO model is approximately 8 hours. All the other algorithms finished in less than an hour. Reevaluating the networks in the proposed architectures and scaling them in GPU clusters may solve this problem.

C. Future Work

The models can be used on other datasets as well with different loss functions. Cross validation and hyperparameter tuning could be performed to yield better performance.

REFERENCES

[1] A. Y. S. Lam, Y. Leung, and X. Chu, "Electric vehicle charging station placement: Formulation, complexity, and solutions," *CoRR*, vol. abs/1310.6925, 2013. [Online]. Available: http://arxiv.org/abs/1310.6925

[2] E. Pournaras, M. Yao, and D. Helbing, "Self-regulating supply–demand systems," *Future Generation Computer Systems*, vol. 76, pp. 73 – 91, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X16303946

[3] P. Pilgerstorfer and E. Pournaras, "Self-adaptive learning in decentralized combinatorial optimization - a design paradigm for sharing economies," in *2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, May 2017, pp. 54–64.

[4] J. A. JOINES, C. T. CULBRETH, and R. E. KING, "Manufacturing cell design: an integer programming model employing genetic algorithms," *IIE Transactions*, vol. 28, no. 1, pp. 69–85, 1996. [Online]. Available: https://doi.org/10.1080/07408179608966253

[5] A. Torreño, E. Onaindia, A. Komenda, and M. Stolba, "Cooperative multi-agent planning: A survey," *CoRR*, vol. abs/1711.09057, 2017. [Online]. Available: http://arxiv.org/abs/1711.09057

[6] A. Torreño, E. Onaindia, and O. Sapena, "FMAP: distributed cooperative multi-agent planning," *CoRR*, vol. abs/1501.07250, 2015. [Online]. Available: http://arxiv.org/abs/1501.07250

[7] M. P. Georgeff, "Distributed artificial intelligence," A. H. Bond and L. Gasser, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988, ch. Communication and Interaction in Multi-agent Planning, pp. 200–204. [Online]. Available: http://dl.acm.org/citation.cfm?id=60204.60217

[8] D. Bienstock and G. Nemhauser, Eds., *Integer Programming and Combinatorial Optimization*, ser. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, vol. 3064. [Online]. Available: http://link.springer.com/10.1007/b97946

[9] L. A. Wolsey, Wolsey, and L. A., "Mixed Integer Programming," in *Wiley Encyclopedia of Computer Science and Engineering*. Hoboken, NJ, USA: John Wiley & Sons, Inc., sep 2008. [Online]. Available: http://doi.wiley.com/10.1002/9780470050118.ecse244

[10] J. F. Shapiro, "Dynamic programming algorithms for the integer programming problem—i: The integer programming problem viewed as a knapsack type problem," *Operations Research*, vol. 16, no. 1, pp. 103–121, 1968. [Online]. Available: https://doi.org/10.1287/opre.16.1.103

[11] E. C. Laskari, K. E. Parsopoulos, and M. N. Vrahatis, "Particle swarm optimization for integer programming," in *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, vol. 2, 2002, pp. 1582–1587.

[12] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," *CoRR*, vol. abs/1611.09940, 2016. [Online]. Available: http://arxiv.org/abs/1611.09940

[13] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," *CoRR*, vol. abs/1704.01665, 2017. [Online]. Available: http://arxiv.org/abs/1704.01665

[14] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, A. P. Badia, K. M. Hermann, Y. Zwols, G. Ostrovski, A. Cain, H. King, C. Summerfield, P. Blunsom, K. Kavukcuoglu, and D. Hassabis, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, oct 2016. [Online]. Available: http://www.nature.com/articles/nature20101

[15] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *CoRR*, vol. abs/1410.5401, 2014. [Online]. Available: http://arxiv.org/abs/1410.5401

[16] J. He, M. Ostendorf, X. He, J. Chen, J. Gao, L. Li, and L. Deng, "Deep reinforcement learning with a combinatorial action space for predicting and tracking popular discussion threads," *CoRR*, vol. abs/1606.03667, 2016. [Online]. Available: http://arxiv.org/abs/1606.03667

[17] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," *CoRR*, vol. abs/1703.08862, 2017. [Online]. Available: http://arxiv.org/abs/1703.08862

[18] G. Farquhar, T. Rocktäschel, M. Igl, and S. Whiteson, "Treeqn and atreec: Differentiable tree planning for deep reinforcement learning," *CoRR*, vol. abs/1710.11417, 2017. [Online]. Available: http://arxiv.org/abs/1710.11417

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, p. 529, feb 2015. [Online]. Available: http://dx.doi.org/10.1038/nature14236http://10.0.4.14/nature14236https://www.nature.com/articles/nature14236{#}supplementary-information

[20] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," *CoRR*, vol. abs/1602.01783, 2016. [Online]. Available: http://arxiv.org/abs/1602.01783

[21] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *CoRR*, vol. abs/1506.02438, 2015. [Online]. Available: http://arxiv.org/abs/1506.02438