PROCEEDINGS OF SCIENCE

ONLINE ICRC 2021
THE ASTROPARTICLE PHYSICS CONFERENCE
Berlin | Germany
37th International Cosmic Ray Conference
12–23 July 2021

Deep-learning applications to the multi-objective optimisation of IACT array layouts

Bernardo Fraga,[a,*] **Ulisses Barres de Almeida**[a] **and Clecio R. Bom**[a]

[a] *Centro Brasileiro de Pesquisas Físicas,*
Rua Dr. Xavier Sigaud 150, 22290-180 Rio de Janeiro, RJ, Brazil
E-mail: bernardo@cbpf.br

The relative disposition of individual telescopes in the ground is one of the important factors in optimising the performance of a stereoscopic array of imaging atmospheric Cherenkov telescopes (IACTs). Following previous attempts at an automated survey of the broad parameter space involved using evolutionary algorithms, in this paper we will present a novel approach to optimising the array geometry based on deep learning techniques. The focus of this initial work will be to test the algorithmic approach and will be based on a simplified toy model of the array. Despite being simplified, the model heuristics aims to capture the principal array performance features relevant for the layout optimisation. Our final goal is to create an algorithm capable of scanning the large parameter space involved in the design of a large stereoscopic array of IACTs to assist optimisation of the array geometry (in face of external constraints and multiple performance objectives). The use of simple heuristics precludes direct comparison to existing real-world experiments, but the analysis is internally consistent and gives insight as to the potential of the technique. Deep learning techniques are being increasingly applied to tackle a number of problems in the field of Gamma-ray Astronomy, and this work represents a novel, original application of this modern computational technique to the field.

37th International Cosmic Ray Conference (ICRC 2021)
July 12th – 23rd, 2021
Online – Berlin, Germany

*Presenter

PoS(ICRC2021)908

1. Introduction

The observation of gamma-rays with ground-based telescopes is done indirectly by detecting the cascade of secondary particles they induce when interacting with the Earth's atmosphere: the *atmospheric shower*. In the case of gamma-initiated, or leptonic air showers, the secondary particles are electron-positron pairs and secondary gamma-ray photons, which develop in a multiplicative process that continues for a few kilometers and is sustained by bremsstrahlung and pair production. Cherenkov photons are produced in the shower as the result of the superluminal velocity of the electron-positron pairs, and their detection can be used to reconstruct the energy and arrival direction of the primary gamma-rays.

The Whipple telescope was the first IACT (Imaging Atmospheric Cherenkov Telescope) to successfully use this technique to detect an astronomical source, the Crab Nebula in 1989 [1]. Since then, more than two hundred sources have been detected by the successive generations of instruments.

The performance of an IACT can be assessed by the array effective area and the quality of the shower reconstruction. Optimising the performance is done both at the individual telescope level, and for the configuration of the array, using Monte Carlo simulations. For small, homogeneous arrays, the telescopes are arranged in regular, symmetric geometries, where the relative spacing of the telescopes is a trade-off between a larger effective area (larger detection rate) and a better reconstruction of the primary. For hybrid or larger arrays, with more than 6 telescopes, the parameter space gets much more complex, and using Monte Carlo simulations for their optimisation is a challenging task. State of the art simulations which use CORSIKA [2] to calculate the shower development, and follow the detailed detection process by the telescopes, are necessary for a thorough analysis. Since the parameter space is highly complex for large arrays, the computational power and computing times required here are very large, and only pre-defined sectors of the full parameter space can be simulated for optimisation.

A previous work [3] has tried to circumvent some of these limitations on the phase-space exploration by proposing a heuristic model of the array and using Evolutionary Optimisation techniques to optimise the shape of an array based on relatively simple metrics. The goal of the work was to provide further indications of which regions of the large parameter space merit further exploration. In this work, we use the same heuristic model for the array but we apply a Deep Reinforcement Learning scheme for the optimisation, since Neural Networks have been proven as the state of the art in several such problems, and seem to be a more promising avenue to approach the problem.

2. Reinforcement Learning

Reinforcement learning (RL) is a general framework where an algorithm (called agent) continuously interacts with an environment, performing actions that change it and being positively or negatively rewarded. The state of the environment is relayed back (fully or partially) to the agent at every step, so that it can choose its next action. One example of a possible application of RL is in the game of tic-tac-toe: the players (agents) perform actions (putting X's and O's) on the environment (the board), changing it at every step. The optimal move of the player will depend on the state of the

board at each turn, and one can imagine a simple reward scheme where a move leading to victory is given a positive reward, a move leading to defeat a negative reward, and the others a zero reward.

More generally, at each time step the agent perform an action a_t guided by its *policy*, $\pi(a_t|s_t)$, taking into account the current observation of the environment s_t, and receives a reward r_{t+1} and the next observation s_{t+1}. The goal is to improve the policy so as to maximize the sum of rewards. More formally, it tries to approximate the optimal action-value function

$$Q(s_t, a_t) = \max_{\pi} \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots | s_t, a_t, \pi\right], \tag{1}$$

which is the maximum sum of rewards discounted by γ. This discount can make the agent select an immediate worse action if this would lead to better rewards in the future.

In the case of tic-tac-toe, the space of all board states is small enough so that a procedure can be built to play optimally. In more complex games such as Chess or Go, however, the space of possible states and moves is large enough for such methods to be not only impractical but unfeasible. In such cases, an agent that can learn from playing the game (much like one of us would) is one possible solution. AlphaGo Zero, which combines Deep Reinforcement Learning and a tree-search algorithm was able to defeat human champions of Go [4]. Its successor AlphaZero, a general reinforcement learning algorithm, was able to defeat other champion algorithms in Chess, Shogi (japanese chess) and AlphaGo Zero itself in Go [5].

RL is not limited to gaming and has been succesfully applied to several areas such as self-driving vehicles [6], robotics [7], industry automation [8, 9] and others, showing that it is a powerful technique to solve different kinds of tasks.

2.1 Deep Q-Learning

Deep Neural Networks (DNNs) have emerged as the state of the art algorithm for several different tasks such as computer vision, natural language processing, time series analysis. DNNs have been used in RL as an approximator of the action-value function with good results (REF).

However, when nonlinear function approximators such as DNNs are used to approximate the action-value function, RL is known to be unstable or even diverge. Deep Q-learning [10] tries to address these issues with two main ideas:

- The agent's experience (the state of the environment, action, reward and subsequent state) are stored at each time step in what is called the Experience Replay. During learning, the inputs to the network are uniformly drawn from the stored samples.

- An interactive update, where the action values are adjusted towards the target values only periodically. This in practice is done by cloning the base network every fixed number of steps and using the cloned network (called the target network) to generate further experience for updating the weights of the base network.

Using an experience replay can break the correlations that would arise by learning from consecutive samples; since the inputs are drawn randomly from the stored experience, the variance between updates is reduced. Furthermore, each experience can be used in multiple weight updates,

allowing for greater data efficiency. And finally, when sampling randomly from stored experience, the behaviour distribution is averaged over several past states, which avoids oscillations and divergences in the parameters.

By generating the new experience steps using older values for the weights (since the target network is only updated every few steps), a delay is caused between an update to the base network and the effect this update would have in collecting new experiences, thus avoiding instabilities and divergences.

3. Metrics

Optimising an array with a number of different telescope types is a complex problem, with a very large parameter space to be searched while also involving the simulation of cosmic showers. Here, we present a toy model of an IACT array with relatively simple metrics to be optimised, following [3]. This is an initial step to a more complete approach to the problem.

The first metric we will use for the optimisation is the effective area. For one telescope, it is given by

$$\int f \, dA \approx \sum_i f(r_i) \Delta A_i \tag{2}$$

where f is the probability of detection, ΔA_i is the area element in a discrete grid containing the telescope and r_i is the radial distance between the telescope position and the area element. The probability of detection for the different telescope types was calculated by [3], and we use their results here.

For multiple telescopes, the probability of detection by at least m telescopes out of N, for each area element, is given by

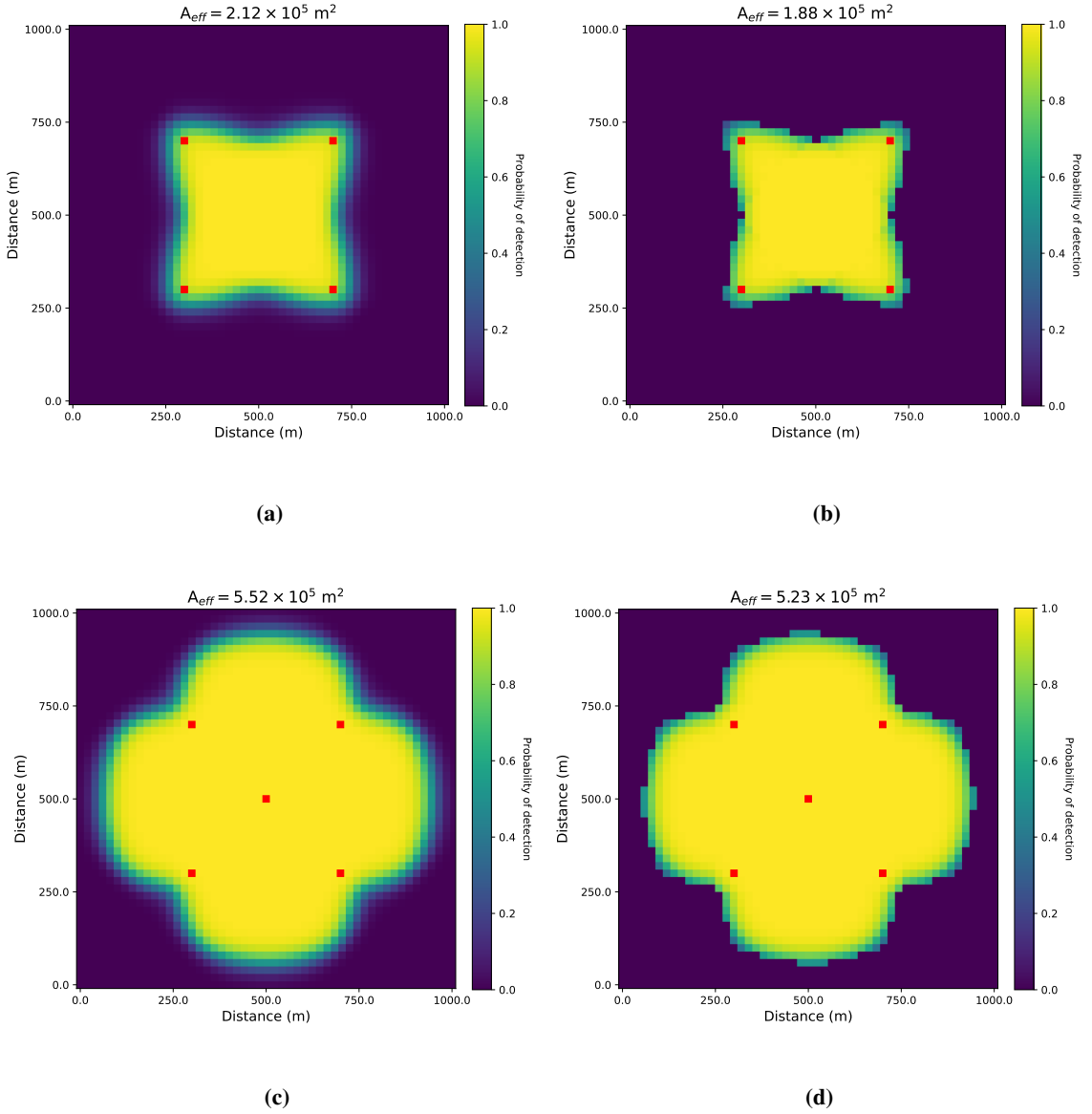
$$P_i^m = \sum_{i=m}^{N} k_\nu^m C_\nu^m, \tag{3}$$

where m is called the *multiplicity* and C_ν^m is the sum of probabilities of all possible combinations of N telescopes taken ν at a time. k_ν^m are the elements of the m-th column of Pascal's Triangle with alternating plus and minus signs (see references in [3]).

One problem with this approach is the increasing computing time needed with increasing number of telescopes due to the factorial in the combinatorics. The training needs to calculate the area twice at each step, and the computing time becomes prohibitively large (of the order of minutes for arrays with ten telescopes), for seven telescopes or more. To speed up the calculation, we make a number of simplifications and calculate the effective area the following way:

1. Suppose that a shower with given energy hits every cell of the grid;
2. For each cell, compute the probability of detecting the shower for each telescope based on the distance of the telescope position and the shower position
3. If the probability is higher than a the mean of 1,000 numbers randomly taken from a uniform distribution $\in [0, 1]$, we consider the shower detected by that telescope.
4. Calculate the probability of detection in that area element with Equation 3, taking only the telescopes that detected the shower for the combinations.

We use the mean of 1,000 numbers as a cutoff to alleviate the effects of using randomness to check for a detection. Compared to the full calculation, in step 3 we exclude a a certain number of telescopes that did not detect the shower, lowering the number of combinations in step 4 and speeding up the process.



(a) **(b)**

(c) **(d)**

Figure 1: Examples of the detection probability for arrangements in regular polygons. On the left (a, c) we show the full calculation, and on the right (b, d) we show the simplified one. The telescopes are shown as red dots and placed on a square with a 400 meter side. We used a shower with energy of 1 TeV and a multiplicity of 3.

Figure 1 shows the probability of detection for a grid with 51 square cells of 20 meters side for some regular arrangements of 10 meter diameter telescopes, using the full calculation and the simplified one shown above. It can be seen that the figures and the calculated effective areas are

quite similar, but the simplified way is five times faster for 4-5 telescopes and can be a thousand times faster for 14-15 due to the reduced number of possible combinations.

4. Training the agent

As an unsupervised algorithm, training an RL agent does not follow the traditional machine learning approach. First of all, we need to fill the experience replay with experiences so the agent can have some options to start learning. This is usually done by using a random policy for a given number of steps. After that, at each step, the training proceeds in three phases:

1. Collect trajectories with the agent's policy in the training environment, saving it in the experience replay; this increases the number of actions given as input to the agent and, as training progresses, these actions are refined;
2. Train the agent using samples collected in the previous step as input. Update the target network if in the correct step;
3. Run the agent in the evaluation environment for some steps and record the metrics.

The training is done using a ϵ-greedy policy: at every move, the agent has an ϵ chance of choosing a random action and a $1 - \epsilon$ chance of choosing the action that maximizes sum of rewards. This allows the agent to explore a larger set of available actions and learn a better policy. Typical values of ϵ are around 0.1.

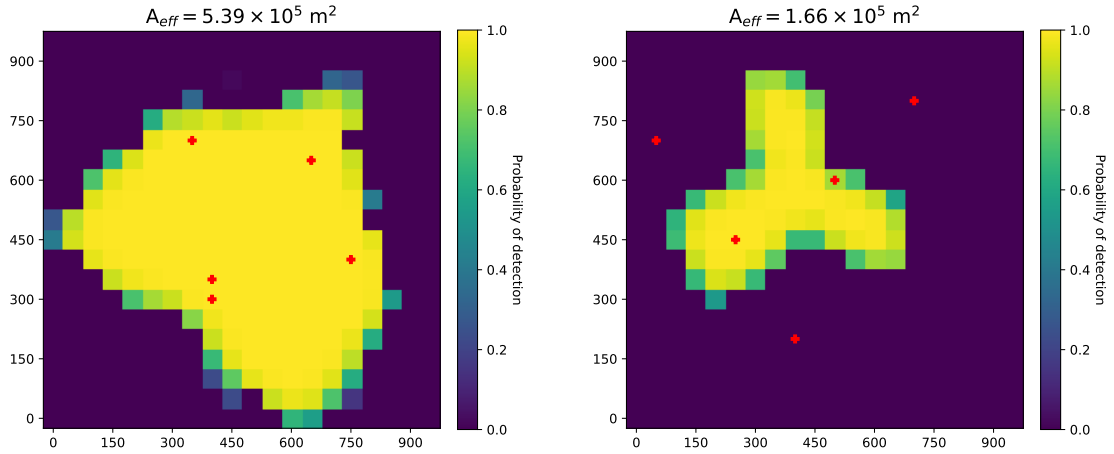
In the same way data is divided into training and validation subsets, we create two different environments, one for collecting data and training (steps 1 and 2), and the other to evaluate the agent (step 3). The evaluation is done after a certain number of steps to check the rewards and possibly other metrics.

5. Preliminary Results

Since one single telescope already has a large area ($\approx 10^5$ km^2), stacking multiple telescopes will cause the shower to be detected more easily (because of the multiplicity) without decreasing the effective area too much; this can cause the agent to position the telescopes close together. To avoid this, we take into account the internal area of the configuration together with the effective area.

For these first results, we chose a square grid with 20 cells on each side and a cell size length of 50 meters. The reward depends on the difference in effective and internal areas before and after the move. We used a Deep Q-Learning algorithm with the DNN agent composed of a series of fully connected layers. The environment and the agent were implemented using TF-Agents [11], a RL library from TensorFlow. We used telescopes with a 10 meter diameter and fixed the shower energy to 1 TeV. The agent was trained for 10,00 iterations, collecting new data for 20 steps at the beginning of each.

We explored some of the parameter space such as the update time for the target network, different activation functions and different values for ϵ in the policy. The best results were found for a large update lag between the training and target network and for a value of $\epsilon = 0.05$.



(a) Multiplicity 3.

(b) Multiplicity 4.

Figure 2: Examples of configurations attained by a trained agent with five and six telescopes, each marked as a red cross. The multiplicity was fixed to three in both cases.

Figure 2 show some of the configurations a trained agent reached for five telescopes. In the case of lower multiplicity, some telescopes tend to be arranged together to increase the effective area, since those would always be able to detect the shower, with little decrease in the effective area. When the multiplicity is larger, this behaviour vanishes, since it is no longer favourable to have two telescopes close. In general, a trained agent tended to a solution similar to these two.

6. Future work

While here we concerned ourselves with a small number of telescopes and just two metrics, a more realistic and useful approach would be to include other metrics such as the angular resolution, energy resolution, gamma/hadron separation and others that are relevant for the array optimisation. Increasing the number of telescopes as well as including different types could provide a more general framework to be used for more realistic purposes.

While Deep Q-Learning has been successfully used in other tasks, a more powerful method might give better results here due to the large number of possible actions.

Detailed shower and detection simulations need to be used as input to calculate the metrics in order to turn the problem more realistic as well. These can be used with a Reinforcement Learning framework to investigate the parameter space for array configuration of ground-based observatories including trade-offs with different metrics, providing a novel approach to aid experiment design and optimisation.

References

[1] T.C. Weekes, M.F. Cawley, D.J. Fegan, K.G. Gibbs, A.M. Hillas, P.W. Kowk et al., *Observation of TeV Gamma Rays from the Crab Nebula Using the Atmospheric Cerenkov Imaging Technique*, *ApJ* **342** (1989) 379.

[2] D. Heck, J. Knapp, J.N. Capdevielle, G. Schatz and T. Thouw, *CORSIKA: a Monte Carlo code to simulate extensive air showers.* (1998).

[3] B.F. Souto and U. Barres de Almeida, *Studies of the performance of an array of Cherenkov telescopes by means of multi-objective evolutionary optimisation*, in *36th International Cosmic Ray Conference (ICRC2019)*, vol. 36 of *International Cosmic Ray Conference*, p. 628, July, 2019.

[4] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. van den Driessche et al., *Mastering the game of Go with deep neural networks and tree search*, *Nature* **529** (2016) 484.

[5] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez et al., *A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play*, *Science* **362** (2018) 1140.

[6] B.R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A.A. Al Sallab, S. Yogamani et al., *Deep Reinforcement Learning for Autonomous Driving: A Survey*, *arXiv e-prints* (2020) arXiv:2002.00444 [2002.00444].

[7] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang et al., *QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation*, *arXiv e-prints* (2018) arXiv:1806.10293 [1806.10293].

[8] J. Luo, O. Sushkov, R. Pevceviciute, W. Lian, C. Su, M. Vecerik et al., *Robust Multi-Modal Policies for Industrial Assembly via Reinforcement Learning and Demonstrations: A Large-Scale Study*, *arXiv e-prints* (2021) arXiv:2103.11512 [2103.11512].

[9] R. Evans and J. Gao, "Deepmind ai reduces google data centre cooling bill by 40%." https://deepmind.com/blog/article/deepmind-ai-reduces-google-data-centre-cooling-bill-40, 2016.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare et al., *Human-level control through deep reinforcement learning*, *Nature* **518** (2015) 529.

[11] S. Guadarrama, A. Korattikara, O. Ramirez, P. Castro, E. Holly, S. Fishman et al., "TF-Agents: A library for reinforcement learning in tensorflow." https://github.com/tensorflow/agents, 2018.