Monte-Carlo Tree Search vs. Model-Predictive Controller: A Track-Following Example

Anonymous Author(s)
Affiliation
Address
email

Abstract

Monte-Carlo Tree Search (MCTS) has achieved remarkable success in the game of Go. However, most success of MCTS is in games where actions are discrete. For automous driving, the vehicle action such as throttle and steering angle is continuous. To fill the gap, we propose an MCTS algorithm for continuous actions, and used it specially for a track-following scenerio. We compared MCTS with a standard Model Predictive Controller (MPC) on the Udacity simulator. Using the same cost function and system model, this MCTS algorithm achieves a much lower cost than MPC. MCTS drives with an adaptive speed, as well as exhibits a braking behavior in sharp turns. MPC drives a nearly constant speed regardless of the curvy track.

1 Introduction

Autonomous driving aims to make cars safer. Nearly 1.3 million people die in road crashes each year, on average 3,287 deaths a day. Road crashes cost USD $518 billion globally, costing individual countries from 1-2% of their annual GDP. [1] So far there are three major avenues for autonomous driving. **The classical approach** extracts perception and localization results from sensors, summarizing into geometry relationship of the car with its environment. Based on the *geometry representation* of the world, a controller is built. This approach is so far the most popular and widely adopted by industrial leaders such as Google, Uber and Baidu. **The learning-from-demonstration approach**, started from the simple full-connected neural networks [Pomerleau, 1989] in the old days to recent deep convolution layers by NVIDIA [Bojarski et al., 2016], regresses the steer angle given the camera view. This approach leads to a simpler architecture for autonomous driving. **The affordance approach**[Chen et al., 2015] predicts relevant geometry features (called “affordances”) from images. Based on the predicted features, a controller can be developed. This approach bears some similarity to Palovian control in which animals map predictions of events into behaviors[Modayil and Sutton, 2014].

Besides these exciting progress, it is interesting to bring reinforcement learning to autonomous driving. Reinforcement learning achieved remarkable success in Atari games [Mnih et al., 2015] and Go [Silver et al., 2016]. Recently, Mobileye proposed an interesting architecture for autonomous vehicles. Similar to the classical approach, their achitecture also has two layers. In particular, their high-level path planning is implemented using a recurrent neural network over the trajectory of the car [Shalev-Shwartz et al., 2016]. The low-level control is a model-based approach that learns a model for the state transition in response to the car’s action. Mobileye’s efforts stand for extending model-based reinforcement learning [Sutton et al., 2008, Yao and Szepesvári, 2012, Grünewälder et al., 2012] to autonomous driving. Modeling the state that the car sees next turns out to be very

[1] http://asirt.org/initiatives/informing-road-users/road-safety-facts/road-crash-statistics

Submitted to 30th Conference on Neural Information Processing Systems (NIPS 2016). Do not distribute.

important for cars although there has not been convincing applications published yet. However, considerable progress was made in video prediction [Zeng et al., 2017, Oh et al., 2015], which can be possibly used on cars. The reward function is also a fundamental issue to bring reinforcement learning to autonomous driving. In games, the reward signal is noise free since win or loss signals can be observed as a delayed but groundtruth reward. Although how to learn a reward function for autonomous driving is still an open problem, Hadfield-Menell et al. explored teaching a car to align with a human driver with his reward function. Brechtel et al. modeled the car's environment using an Markov Decision Process (MDP) in which the state space is equidistant cells of the coordinates on the road, and the transition probabilities are approximated using a Dynamic Bayesian Networks. Their empirical studies show that the car can coordinate well when to overtake according to oncoming traffic. Exploring in a driving environment is challenging because such exploration (normally practiced in reinforcement learning without constraint) must be safty guaranteed. Recently, Mnih et al. proposed an asynchronous reinforcement learning framework that lets a number of learning agents run in parallel aiming to explore different parts of the environment. Their algorithm on a simulated driving environment achieved near to a human driver with only 12 hours of training. It is interesting to see whether this new framework can solve the specific exploration constraint in autonomous driving.

In this paper, we study Monte-Carlo Tree Search (MCTS) for an autonous driving setting. MCTS is especially advantageous for large and complex decision making problems, as demonstrated in the competition of AlphoGo against Mr. Lee Sedol [2]. MCTS is well practised and relatively easy to implement. All the top Go programs have used MCTS for a decade, e.g., [Coulom, 2007, Silver, 2009, Enzenberger et al., 2010]. So far the success of MCTS is largely in board games where actions are discrete. However, in autonomous driving a car's actions like throttle, braking and steer angles are all continuous. We consider a simple motion planning setting where a car has been given a trajectory to follow, and its goal is to drive within track boundary. Note in our problem, motion planing is by no means to be realistic. Practical motion planning also considering avoiding obstacle, e.g., [Kuwata et al., 2008]. We aim to have an environment that renders a simple cost function and vehicle model under which comparing the performance of MCTS and MPC is easy.

2 Background

The classical approach is so far the most practiced and mature. A two-level architecture for autonomous vehicle is often used: path planning at a high level and vehicle control (with a target path and speed) at a low level [Paden et al., 2016, Berntorp, 2017]. There are a spectrum of methods for each of the problems. For example, Rapid-exploring Random Trees finds feasible tracjectories for robots with high degress with freedom [Lavalle, 1998, Kuwata et al., 2008]. MPC is classical control method [Garcia et al., 1989], and has been used for motion planning in a short time horizon [Paden et al., 2016, Kim et al., 2014, Omar et al., 1998, Yim and Oh, 2004, Raffo et al., 2009, Ng et al., 2003, Bakker et al., 1987, Kong et al., 2015, Rajamani, 2011, Besselmann and Morari, 2009, Levinson et al., 2011, Urmson et al., 2007]. MPC is a major research field on its own and this section provides the application context of MPC for autonomous driving, especially lane following.

2.1 The Model and the Problem

The car's dynamics is represented by a practical model, often referred to as the *Kinematic model*. In this model, two car wheels connected by a rigid link. The state of the car is given by $[x, y, \psi, v]$, where x, y are the x-y coordinates of the car, ψ and v are the orientation and speed of the car. The model can be expressed by,

$$\begin{aligned} \dot{x} &= vcos(\psi) \\ \dot{y} &= vsin(\psi) \\ \dot{\psi} &= \frac{a[steer]v}{L_f} \\ \dot{v} &= a[throttle], \end{aligned} \tag{1}$$

where L_f is the distance between the two front wheels. This is descretized using Euler method in practice. In our problem, at each time step, an agent (MPC or MCTS) receives a number of reference

[2]https://deepmind.com/research/alphago/

coordinate points. These points are often provided by a high-level trajectory planner. The agent is also given the a distance measures δ, the distance of the car's center to the track axis; an angle deviation measure, ω, the difference between the car's heading angle (ψ) and the track axis direction. In practice, both δ and ω are computed by first regressing a polynomial line from the reference points. The goal of the agents is to drive close a target speed v^* within the track. Specificially at each time step k, the agent selects an action a. Afterwards it receives a cost signal that is computed from the following equation:

$$r(s_k, a) = w_{tr}\delta_{k+1}(a)^2 + w_{ang}\omega_{k+1}(a)^2 + w_v(v_{k+1}(a) - v^*)^2 + w_{st}a[\text{steer}]^2 + w_{thr}a[\text{throttle}]^2 \\ + w_{steerd}(a[\text{steer}] - a_{k-1}[\text{steer}])^2 + w_{throtd}(a[\text{throttle}] - a_{k-1}[\text{throttle}])^2 \quad (2)$$

2.2 Model Predictive Controller

MPC assumes the knowledge of the cost function in equation 2. It defines a cost function that considers N steps ahead. This cost function is essentially the undiscounted, N-step truncated return. MPC produces a sequence of N actions to maximize the return

$$a_{0:N-1} = \arg\max_{a_{0:N-1}} R = \arg\max_{a_{0:N-1}} \sum_{t=0}^{N-1} r(\tilde{s}_t, a),$$

where $\tilde{s}_0$ was set to the current state s_k. Common practice of MPC is to use the interior point optimizer [Paden et al., 2016].

2.3 Monte-Carlo Tree Search

MCTS is a special policy search algorithm. Comparing to other reinforcement learning methods, policy search algorithms can find global optima [Valko et al., 2013, Munos, 2014]. MCTS algorithms are designed for descrete actions. For example, the "pure" MCTS algorithm works by playing a number of random games to the end; and the moves that achieves the best game scores are chosen. In the Upper Confidence Tree (UCT) algorithm [Kocsis and Szepesvári, 2006], the actions are treated as the arms in a multi-armed bandit problem and the frequency of actions is used to measure the knowledge of the actions according to which the exploration term in selecting the action is determined. Practice and theory of MCTS for continuous actions is largely a gap. A number of recent advances aims to generalize across actions. In particular, [Couetoux et al., 2011, Yee et al., 2016] explored generalizing in actions from already exploited actions. HOOT replaces the UCB algorithm in UCT with a continuous action slection procedure [Mansley et al., 2011]. In this short paper, we aimed to first extend pure MCTS for autonomous driving.

3 A New MCTS Algorithm

Algorithm 1 shows an extension of the descrete-action Pure MCTS to continuous actions. This MCTS algorithm generates a number of paths expanded from continuous actions. In expanding the subtrees from a state, we enforce the continuity in the actions going down a tree. The inpiration of the algorithm is that in driving the actions do not change abruptively. This small trick reduces the search space significantly.

4 Experiments

In the experiment, we used the Udacity simulator [3]. The simulator is developed by Unity to support self-driving car development. Both algorithms used the same model in equation 1 and the same cost function in equation 2. Simulation for both algorithms was run with lookahead depth of 8. The weights for the reward function are, $w_{tr} = 10.0, w_{ang} = 50.0, w_v = 1.0, w_{st} = 10.0, w_{thr} = 3000.0, w_{steerd} = 10.0, w_{throtd} = 3000.0$.

For both MPC and MCTS, only the first action a_0 was used although N actions were produced at a single time step. In the experiment, the target speed was set to 70 km/h.

[3] `https://github.com/udacity/self-driving-car-sim`

input : An action model A and a return function that considers N steps of future rewards.
output : A policy that maximizes the return function.
Initialize the state s_0 and the action a_0.
for $t = 0, 1, \ldots$ **do**
    Observe state s_t
    Receive a number of reference points, and fit a polynomial line
    /* Search over N_p paths*/
    for $p = 0, \ldots, N_p$ **do**
        Set $\tilde{s}_0 = s_t, a' = a_t$ /* each path starts with the current state and action */
        Set $R(p) = 0$
        /* planning into future N steps */
        for $k = 0, \ldots, N$ **do**
            Sample a from a distribution $u(a')$
            Predict the next state, $\tilde{s}_{k+1} = A(\tilde{s}_k, a)$
            Compute the reward r according to $\tilde{s}_{k+1}$, a, a', and deviation from the reference line
            Update $R(p) = \gamma R(p) + r$
            Set $a' = a$
        end
    end
    Select the best path (with highest return R)
    Set a_t to the first action in the best path.
    Take action a_t
end

Algorithm 1: Continuity-preserved (Monte-Carlo) Tree Search.

As shown in Figure 1 (left plot), MCTS achieved a much smaller cost than MPC. The cost function is a linear combination of seven cost components. MCTS achieved both a smaller velecity cost and a smaller trackPos cost (deviation from the track center) as shown in the second plot. The third plot in Figure 1 shows the velocity on the track.

MCTS's speed is more adaptive due to that the track has quite a few sharp turns. MPC, on the other hand, drives at a nearly constant speed. In particular, after the beginning acceleration period, MPC drove between 57.8 km/h and 58.6 km/h with an average speed of 58.4 km/h. MCTS drove between 42.8 km/h and 67.5 km/h, averaging at 62.3 km/h. Interestingly, MCTS shows braking behavior (continually negative throttles) ahead of sharp turns (Figure 2) although it was never explictly trained to do so. In contrast, MPC never braked. For MCTS, 10000 paths were generated with random samples of action (both steer angle and throttle). In generating the actions, uniform samples in the small range of last steer angle and last throttle were independently drawn.

We produced a video of MCTS driving:

`https://youtu.be/YP7qPJSJAVU`

MPC driving:

`https://youtu.be/SLl50wMenyY`

References

Egbert Bakker, Lars Nyborg, and Hans B Pacejka. Tyre modelling for use in vehicle dynamics studies. Technical report, SAE Technical Paper, 1987.

Karl Berntorp. Path planning and integrated collision avoidance for autonomous vehicles. In *American Control Conference (ACC)*, pages 4023–4028, 05 2017.

Thomas Besselmann and M Morari. Autonomous vehicle steering using explicit lpv-mpc. pages 2628–2633, 08 2009.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao,

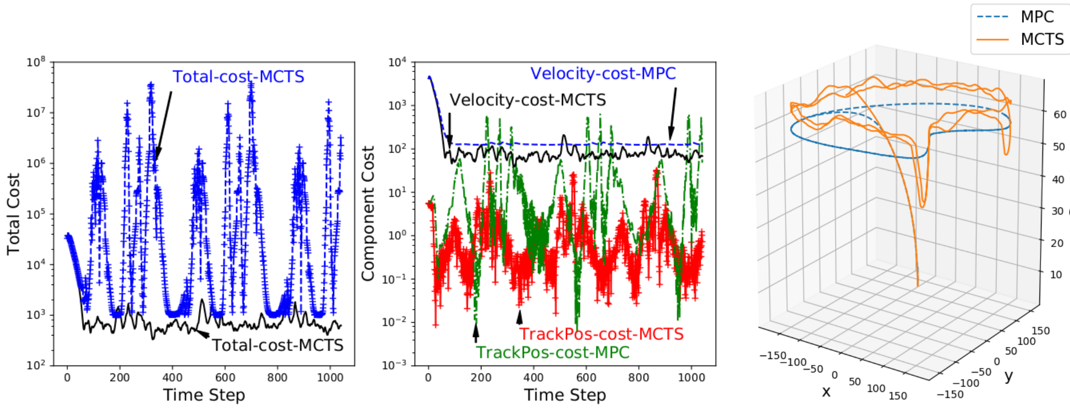

Figure 1: MCTS vs. MPC. The left plot shows the total cost over the next 8 time steps of the two algorithms. The middle plot shows the cost of the velocity component and the trackPos (distance to the center of the lane) component in the same run as the left plot. The right plot shows the velocity on the track, which shows that (a) MCTS accelerates faster in the beginning (the ascending curves from bottom); (b) MCTS drives closer to the target speed (70 km/h) than MPC most of the time; (c) MCTS's control is more adaptive to curvature in the track. In particular, at sharp turns we see speed dip in the orange line while MPC drives at almost a constant speed (58 km/h).

Figure 2: MCTS braking in front of a sharp turn. MPC never shows braking behavior in the experiment.

and Karol Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016. URL `http://arxiv.org/abs/1604.07316`.

S. Brechtel, T. Gindele, and R. Dillmann. Probabilistic mdp-behavior planning for cars. In *14th International Conference on Intelligent Transportation Systems, IEEE*, page 1537–1542, 2011.

Chenyi Chen, Ari Seff, Alain L. Kornhauser, and Jianxiong Xiao. DeepDriving: Learning affordance for direct perception in autonomous driving. In *ICCV*, 2015.

Adrien Couetoux, Jean-Baptiste Hoock, Nataliya Sokolovska, Olivier Teytaud, and Nicolas Bonnard. Continuous Upper Confidence Trees. In *LION'11: Proceedings of the 5th International Conference on Learning and Intelligent OptimizatioN*, page TBA, Italy, January 2011. URL `https://hal.archives-ouvertes.fr/hal-00542673`.

Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *Proceedings of the 5th International Conference on Computers and Games*, CG'06, pages 72–83, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 3-540-75537-3, 978-3-540-75537-1. URL `http://dl.acm.org/citation.cfm?id=1777826.1777833`.

Markus Enzenberger, Martin Müller, B Arneson, and Richard Segal. Fuego - an open-source framework for board games and go engine based on monte carlo tree search. 2:259–270, 01 2010.

C. E. Garcia, D. M. Prett, and M. Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25(3):335–348, May 1989. ISSN 0005-1098. doi: 10.1016/0005-1098(89)90002-2. URL `http://dx.doi.org/10.1016/0005-1098(89)90002-2`.

Steffen Grünewälder, Guy Lever, Luca Baldassarre, Massimilano Pontil, and Arthur Gretton. Modelling transition dynamics in mdps with rkhs embeddings. In *ICML*, pages 1603–1610, USA, 2012. Omnipress. ISBN 978-1-4503-1285-1.

Dylan Hadfield-Menell, Anca D. Dragan, Pieter Abbeel, and Stuart J. Russell. Cooperative inverse reinforcement learning. *CoRR*, abs/1606.03137, 2016. URL `http://arxiv.org/abs/1606.03137`.

E Kim, J Kim, and Myoungho Sunwoo. Model predictive control strategy for smooth path tracking of autonomous vehicles with steering actuator dynamics. 15:1155–1164, 12 2014.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Proceedings of the 17th European Conference on Machine Learning*, ECML'06, pages 282–293, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-45375-X, 978-3-540-45375-8. doi: 10.1007/11871842_29. URL `http://dx.doi.org/10.1007/11871842_29`.

Jason Kong, Mark Pfeiffer, Georg Schildbach, and Francesco Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *Intelligent Vehicles Symposium (IV), 2015 IEEE*, pages 1094–1099. IEEE, 2015.

Yoshiaki Kuwata, Gaston Fiore, Justin Teo, Emilio Frazzoli, and Jonathan How. Motion planning for urban driving using rrt. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pages 1681–1686, 09 2008.

Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, Iowa State University, 1998.

Jesse Levinson, Jake Askeland, Jan Becker, Jennifer Dolson, David Held, Soeren Kammel, J Zico Kolter, Dirk Langer, Oliver Pink, Vaughan Pratt, et al. Towards fully autonomous driving: Systems and algorithms. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 163–168. IEEE, 2011.

Christopher R. Mansley, Ari Weinstein, and Michael L. Littman. Sample-based planning for continuous action markov decision processes. In *21st International Conference on Auto- mated Planning and Scheduling, ICAPS*, 2011.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 02 2015. URL `http://dx.doi.org/10.1038/nature14236`.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783, 2016. URL `http://arxiv.org/abs/1602.01783`.

Joseph Modayil and Richard S. Sutton. Prediction driven behavior: Learning predictions that drive fixed responses. In *AAAI Workshop on AI and Robotics*, 2014.

Rémi Munos. From bandits to monte-carlo tree search: The optimistic principle applied to optimization and planning. *Foundations and Trends in Machine Learning*, 7(1):1–129, 2014. ISSN 1935-8237. doi: 10.1561/2200000038. URL `http://dx.doi.org/10.1561/2200000038`.

Andrew Y Ng, H Jin Kim, Michael I Jordan, Shankar Sastry, and Shiv Ballianda. Autonomous helicopter flight via reinforcement learning. In *NIPS*, volume 16, 2003.

Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-Conditional Video Prediction using Deep Networks in Atari Games. In C Cortes, N D Lawrence, D D Lee, M Sugiyama, R Garnett, and R Garnett, editors, *NIPS 28*, pages 2845–2853. Curran Associates, Inc., 2015.

T Omar, A Eskandarian, and N Bedewi. Vehicle crash modelling using recurrent neural networks. *Mathematical and computer Modelling*, 28(9):31–42, 1998.

Brian Paden, Michal Cáp, Sze Zheng Yong, Dmitry S. Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *CoRR*, abs/1604.07446, 2016. URL `http://arxiv.org/abs/1604.07446`.

Dean A. Pomerleau. Alvinn, an autonomous land vehicle in a neural network. Technical report, Carnegie Mellon University, 1989. URL `http://repository.cmu.edu/cgi/viewcontent.cgi?article=2874&context=compsci`.

G.V. Raffo, Guilherme K. Gomes, Julio Normey-Rico, Christian R. Kelber, and Leandro B. Becker. A predictive controller for autonomous vehicle path tracking. 10:92 – 102, 04 2009.

Rajesh Rajamani. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

Shai Shalev-Shwartz, Nir Ben-Zrihem, Aviad Cohen, and Amnon Shashua. Long-term planning by short-term prediction. *CoRR*, abs/1602.01580, 2016. URL `http://arxiv.org/abs/1602.01580`.

David Silver. *Reinforcement Learning and Simulation-Based Search in Computer Go*. PhD thesis, 2009.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. doi: 10.1038/nature16961. URL `http://dx.doi.org/10.1038/nature16961`.

Richard Sutton, Csaba Szepesvari, Alborz Geramifard, and Michael Bowling. Dyna-style planning with linear function approximation and prioritized sweeping. In *UAI*, pages 528–536, 2008.

Chris Urmson, J Andrew Bagnell, Christopher R Baker, Martial Hebert, Alonzo Kelly, Raj Rajkumar, Paul E Rybski, Sebastian Scherer, Reid Simmons, Sanjiv Singh, et al. Tartan racing: A multi-modal approach to the darpa urban challenge. 2007.

Michal Valko, Alexandra Carpentier, and Rémi Munos. Stochastic Simultaneous Optimistic Optimization. In *International Conference on Machine Learning*, Atlanta, United States, June 2013. URL `https://hal.inria.fr/hal-00789606`.

Hengshuai Yao and Csaba Szepesvári. Approximate policy iteration with linear action models. In *AAAI*, pages 1212–1217, 2012.

Timothy Yee, Viliam Lisy, and Michael Bowling. Monte carlo tree search in continuous action spaces with execution uncertainty. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 690–696. AAAI Press, 2016. ISBN 978-1-57735-770-4. URL `http://dl.acm.org/citation.cfm?id=3060621.3060718`.

Young Uk Yim and Se-Young Oh. Modeling of vehicle dynamics from real vehicle measurements using a neural network with two-stage hybrid learning for accurate long-term prediction. *IEEE Transactions on Vehicular Technology*, 53(4):1076–1084, 2004.

Kuo-Hao Zeng, William B. Shen, De-An Huang, Min Sun, and Juan Carlos Niebles. Visual forecasting by imitating dynamics in natural sequences. *CoRR*, abs/1708.05827, 2017. URL `http://arxiv.org/abs/1708.05827`.