Learning Self-Game-Play Agents for Combinatorial Optimization Problems

Extended Abstract

Ruiyang Xu
Northeastern University
Boston, Massachusetts
ruiyang@ccs.neu.edu

Karl Lieberherr
Northeastern University
Boston, Massachusetts
lieber@ccs.neu.edu

ABSTRACT

Recent progress in reinforcement learning (RL) using self-game-play has shown remarkable performance on several board games as well as video games (e.g., Atari games and Dota2). DeepMind researchers have already implemented model-free RL to play Go and Chess at a superhuman level using neural Monte-Carlo-Tree-Search (neural MCTS). Therefore, it is plausible to consider that RL, starting from zero knowledge, can be applied to other problems which can be converted into games. We try to leverage the computational power of neural MCTS to solve a class of combinatorial optimization problems. Following the idea of Hintikka's Game-Theoretical Semantics, we propose the Zermelo Gamification (ZG) to transform specific combinatorial optimization problems into Zermelo games whose winning strategies correspond to the solutions of the original optimization problem. The ZG also provides a specially designed neural MCTS. We use a combinatorial planning problem for which the ground-truth policy is efficiently computable to demonstrate that ZG is promising.

KEYWORDS

Reinforcement Learning; Neural MCTS; Self-game-play; Combinatorial Optimization; Model-free

ACM Reference Format:
Ruiyang Xu and Karl Lieberherr. 2019. Learning Self-Game-Play Agents for Combinatorial Optimization Problems. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 3 pages.

1 INTRODUCTION

We transform a certain form of combinatorial optimization problems (e.g. the HSR problem, described in section 3) into games so that a game-play agent can be leveraged to play the game and solve the original problem on a specific instance. In Fig. 1 one can see how two competitive agents, called P and OP, gradually, but with setbacks (as in AlphaZero [9] and [8]), improve and jointly arrive at the winning strategy. Model-Free learning converges and solves a non-trivial problem although the underlying game is fundamentally different from Go and Chess. Related works on ML and combinatorial problems can be found in [1], [4], [5], [6] and [10].

We make three main contributions: (1.) We introduce the Zermelo Gamification which consists of two contributions (1.a) a way to generalize combinatorial problems to more comprehensive combinatorial problems in the form of Zermelo games following the approach in Hintikka's Game-Theoretical Semantics [3]; (1.b) we implemented a variant of the neural MCTS algorithm ([2] and [7]) specifically designed for those Zermelo games; 2. Evaluation: we evaluate our algorithm on a problem (i.e., *HSR*) for which a Bernoulli's triangle shows the winning strategy efficiently. Our result shows that, for problems under a certain size, neural MCTS does find the optimal strategy, hence solving the original optimization problem in a tabula-rasa (model-free) style. 3. Indications: We show how the winning strategy for both players of the Zermelo game provides indications that a given problem instance does (not) have a solution. Those indications are made possible through the generalization mentioned in contribution 1 and they are more useful then the simple answer: there is no solution. A complete description of this study can be found in [11].

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada.

2 ZERMELO GAMIFICATION

The combinatorial optimization problems studied in this paper can be described with the following logic statement MQ2: $\exists n\{G(n) \wedge (\forall n' > n\ \neg G(n'))\}$, $G(n) := \forall x\ \exists y : \{F(x, y; n)\}$ or $G(n) := \exists y\ \forall x : \{F(x, y; n)\}$. In this statement, n is a natural number and x, y can be any instances depending on the concrete problem. F is a predicate on x, y, n. Hence the logic statement above essentially means that there is a maximum number n such that for all x, some y can be found so that the predicate $F(x, y; n)$ is true. Formulating those problems as interpreted logic statements are crucial to transforming them into Zermelo games. A Zermelo game is defined to be a two-player, finite, and perfect information game with only one winner and loser, and during the game, players move alternately (i.e., no simultaneous move). Leveraging the logic statement above, the Zermelo game is built on the Game-Theoretical Semantic approach (by Hintikka [3]) with two phases and two players (the Proponent (P), who claims that the statement is true, and the Opponent (OP), who argues that the statement is false. The original problem can be solved if and only if the P is able to propose some optimal number n so that a perfect OP cannot refute it).

2.1 Proposal Phase

In the initial phase of the Zermelo game player P will propose a number n. Then the player OP will decide whether to accept this n, or reject it. OP will make his decision based on the logic statement: $A \wedge B$, $A := G(n)$, $B := \forall n' > n\ \neg G(n')$. Specifically, the OP tries to refute the P by attacking either on the statement A or B. The OP will accept n proposed by the P if she confirms $A = False$. The OP will reject n if she is unable to confirm $A = False$. In this case, the OP treats n as non-optimal, and proposes a new $n' > n$ which makes $B = False$. The rejection can be regarded as a role-flip between the two players.

2.2 Refutation Phase

This is the phase where the two players actually search for evidence and construct strategies to attack each other or defend themselves. Generally speaking, regardless of the role-flip, the P claims $G(n)$ holds for some n, the OP will refute this claim by giving some instances of x (for existential quantifier) so that $\neg G(n)$ holds. If the P successfully figures out the exceptional y (for universal quantifier) which makes $F(x, y; n)$ hold, the OP loses the game, otherwise, the P loses.

3 HSR PROBLEM AND HSR GAME

The HSR problem serves as a proof-of-concept in our research: consider throwing jars from a specific rung of a ladder, the jars could either break or not. One has k identical jars and q test chances to throw those jars, can she find the maximum height of the ladder so that any highest safe rung can be located with the given resources k, q? We observed the recursive structure in this problem and formulated the HSR problem as a logic statement as described in section 2, where:

$$G_{k,q}(n) = \exists m \leq n \; \forall a \in \{\text{“break”}, \text{“not break”}\} : \{F(m, a; n)\}$$

$$F(m, a; n) = \begin{cases} \text{True}, & \text{if } n = 0 \\ \text{False}, & \text{if } n > 0 \wedge (k = 0 \vee q = 0) \\ G_{k-1,q-1}(m-1), & \text{if } a = \text{“break”} \\ G_{k,q-1}(n-m), & \text{if } a = \text{“not break”} \end{cases}$$

In the logic statement, $G_{k,q}(n)$ means one can use k jars and q tests to locate any highest safe rung in a ladder with n rungs. The problem is defined recursively, m is the optimal testing point (if there is one) so that no matter the jar breaks or not, the rest of the resources can still be used to locate the highest safe rung.

With $G_{k,q}(n)$, we define the HSR game where the P and OP move alternately. In the proposal phase, the P will propose a number n and the OP will decide whether to accept it or reject it, as described in section 2. During the refutation phase, the P will pick a testing point m, then the OP will reply “break” or “not break”. Specifically, in each round, the P claims $\exists m \leq n : \{G_{k-1,q-1}(m-1) \wedge G_{k,q-1}(n-m)\}$ holds. Then the OP refutes the claim by either refuting $G_{k-1,q-1}(m-1)$ or $G_{k,q-1}(n-m)$. In this way, both the testing policy and the highest safe rung are learned implicitly.

4 EXPERIMENT

Two independent neural networks are applied to learn the proposal game and refutation game respectively. During each iteration of the learning process: 1. 100 episodes of self-play will be executed through a neural MCTS using the current neural network. Data generated during self-play will be stored. 2. the neural networks will be trained with the data in the replay buffer. And 3. the newly trained neural network and the previous old neural network are put into a competition. We collect the correctness data for both of the neural networks during each iteration. Fig. 1 show the process

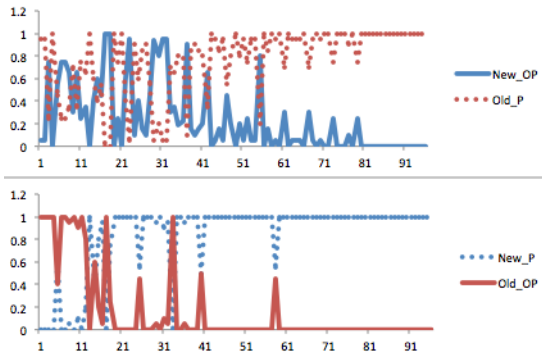

Figure 1: Correctness ratio measured on $k = 7, q = 7$. Where New_OP means the newly trained neural network plays as an OP; Old_P means the previously trained neural network plays as a P. Similar for New_P and Old_OP.

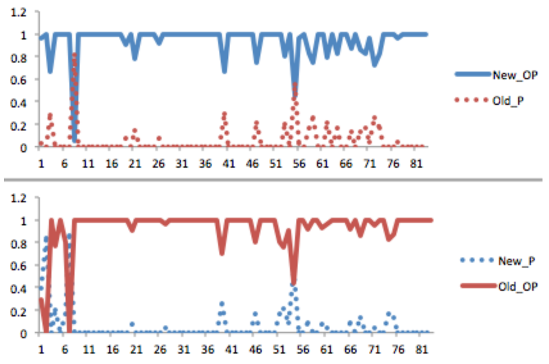

Figure 2: Refutation phase on $k = 7, q = 7$ but n is set to 129 so there is no solution.

where the two players continuously compete with each other until finally converging to a solution where the P's policy can always keep her winning position. The OP is in a dilemma when P is perfect, i.e., P chooses only the optimal n and m. On the other hand, for a problem without any solution, the opposite will happen (see Fig. 2).

5 CONCLUSION

We introduce MQ2 and the MQ2 Zermelo Gamification. We formulate the optimization problem using predicate logic (where the types of the variables are not "too" complex) and then we use the corresponding Zermelo game which we give to the adapted neural MCTS algorithm. For our proof-of-concept specialization of MQ2, HSR Zermelo Gamification, we notice that the adapted neural MCTS algorithm converges on small instances that can be handled by our hardware and finds the winning strategy. Our evaluation counts all correct/incorrect moves of the players, based on ground-truth We hope our research sheds some light on the potential for neural MCTS to solve interesting gamified problems.

REFERENCES

[1] Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. 2016. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940* (2016).

[2] Cameron Browne, Edward Jack Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez Liebana, Spyridon Samothrakis, and Simon Colton. 2012. A Survey of Monte Carlo Tree Search Methods. *IEEE Trans. Comput. Intellig. and AI in Games* 4, 1 (2012), 1–43.

[3] Jaakko Hintikka. 1982. Game-theoretical semantics: insights and prospects. *Notre Dame J. Formal Logic* 23, 2 (04 1982), 219–241.

[4] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. 2017. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*. 6348–6358.

[5] Alexandre Laterre, Yunguan Fu, Mohamed Khalil Jabri, Alain-Sam Cohen, David Kas, Karl Hajjar, Torbjorn S Dahl, Amine Kerkeni, and Karim Beguir. 2018. Ranked Reward: Enabling Self-Play Reinforcement Learning for Combinatorial Optimization. *arXiv preprint arXiv:1807.01672* (2018).

[6] Daniel Selsam, Matthew Lamm, Benedikt Bunz, Percy Liang, Leonardo de Moura, and David L Dill. 2018. Learning a SAT Solver from Single-Bit Supervision. *arXiv preprint arXiv:1802.03685* (2018).

[7] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529 (Jan. 2016), 484.

[8] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. 2018. A general reinforcement learning algorithm that masters Chess, Shogi, and Go through self-play. *Science* 362, 6419 (2018), 1140–1144.

[9] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. 2017. Mastering the game of Go without human knowledge. *Nature* 550 (Oct. 2017), 354.

[10] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer Networks. In *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.). Curran Associates, Inc., 2692–2700.

[11] Ruiyang Xu and Karl Lieberherr. 2019. Learning Self-Game-Play Agents for Combinatorial Optimization Problems. *arXiv preprint arXiv:1903.03674* (2019).