卒 業 研 究 報 告 書

題　　目

深層強化学習に基づく
トレーディングカードゲーム環境の構築

研 究 グ ル ー プ　　第 1 研究グループ

指 導 教 員　　森 直樹 教授

令　和　4　年（　　2022　年）度 卒 業

（No.　　1201201100　　）　　西村 昭賢

大阪府立大学工学域電気電子系学類情報工学課程

目 次

図 目 次

表 目 次

1　はじめに

近年, 人工知能に関する研究分野は目覚ましい発展を遂げており様々な分野に応用されている. その中でも人間の学習プロセスに近いとされる強化学習と深層学習を融合した深層強化学習は自動運転やロボット, 推薦システム等の実生活の問題解決への応用例が数多く報告されている [2][3][4]. 実世界の問題解決への応用だけでなく, 深層強化学習はゲームへの応用も盛んである. 特に将棋や囲碁といった, プレイヤーが意思決定をする段階でそれ以前の意思決定の過程がすべて把握可能な完全情報ゲームへの応用においては AlphaGo [5], AlphaZero [6] を筆頭に現役のプロプレイヤーを圧倒する性能を残しており成果が顕著である. 最近では麻雀やポーカーのような, プレイヤーに与えられる情報が部分的である不完全情報ゲームへの応用も注目されている.

本研究では不完全情報ゲームであるトレーディングカードゲーム環境への深層強化学習の適用と深層強化学習を用いたゲームバランス調整手法を提案し, 独自に構築したトレーディングカードゲーム環境を用いて数値実験することでその有効性を示す.

以下に本論文の構成を示す. まず, 2 章では本研究で用いる要素技術について, 3 章では類似研究と提案手法について説明する. 4 章では本環境で独自に構築したトレーディングカードゲーム環境の概要を示す. 5 章で実験方法の説明をし, 6 章で実験結果と考察を示す. そして, 7 章に本研究のまとめ及び今後の課題について述べる.

2　要素技術

2.1　OpenAI Gym

OpenAI Gym [7] は非営利企業 OpenAI が提供する強化学習のシミュレーション用ライブラリであり, 強化学習の環境として図 2.1 に示すように多くのゲーム, シミュレータが登録されている. さらには提供されているインターフェースに沿って, エージェントの行動空間や状態空間, 報酬などを定義, 実装することで自作の強化学習環境を構築し利用することができる. 様々な強化学習用ライブラリに対応しているため比較的容易に強化学習を試すことができる.

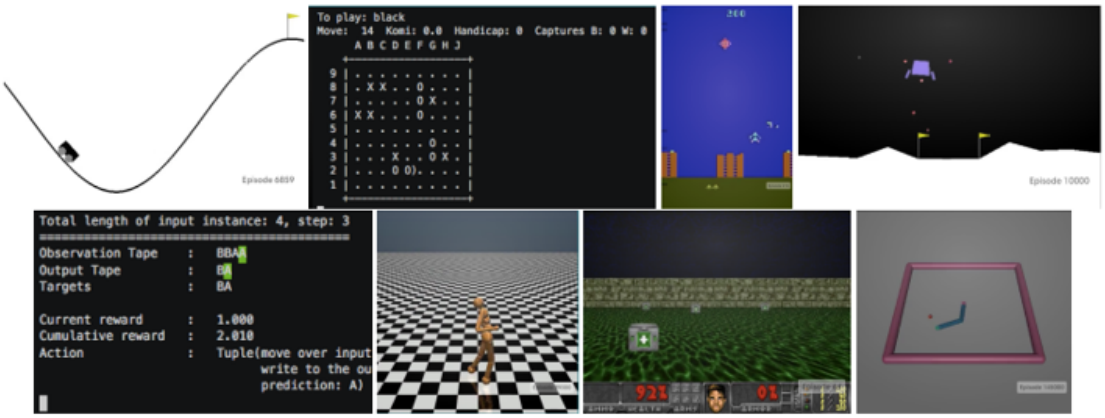

図 2.1: OpenAI Gym が提供している様々な環境

2.2　Q 学習

強化学習では, エージェントが行動することで環境における状態が変化し報酬を得る. 強化学習における行動はその直後に獲得する報酬の大きさではなく, 未来に渡っての報酬の総和を見積もった値である「価値」の最大化に繋がるかという観点で評価される. 価値の最大化を目指す場合にはある状態 s において行動 a をとった時の価値が分かればよい. この価値のことを Q 値, あるいは行動価値関数と呼ぶ. この Q 値を基に行動を選択していく価値ベースの強化学習手法において代表的な手法が Q 学習である. Q 学習ではエージェントの 1

ステップごとに (2.1) 式に示す更新式で Q 値を更新する.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \mathrm{max}_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \tag{2.1}$$

なお t は時間, r は報酬, α は Q 値の更新度合いを表す学習率, γ は将来の価値の割引度合いを表す割引率である.

また, Q 学習に代表される強化学習においては環境の調査を目的とする探索と, 探索により得られた良い報酬を得ることができる経験の活用をそれぞれどの程度にすればよいかといういわゆる「探索と活用のトレードオフ」という問題が発生する. これを解決する手法として一般的なものが ϵ – greedy 法である. ϵ – greedy 法では確率 ϵ でランダムに行動し探索, それ以外では経験を活用して価値が最も高い行動を選択することで探索と活用のバランスをとっている.

2.3 Deep Q Network

Q 学習を実際に実装する場合, 状態と行動をインデックスとした Q 値のテーブルを作成する. しかし状態空間や行動空間が高次元である, あるいは状態や行動が離散値ではなく連続値で表現される場合には Q テーブルのメモリ量は爆発してしまう. この問題を解決した技術が Deep Q Network (DQN) [8] である. DQN ではニューラルネットワークを用いて, ある状態における行動ごとの Q 値を推定することでたとえ状態が連続値であっても学習可能としている. DQN では, エージェントが経験した過去の体験を Replay Memory に一定期間保存しておき, 過去の経験をランダムにサンプリングして学習する Experience Replay や行動を決定する Q 値のネットワークと Q 値を学習するネットワークを分けることで Q 値の過大評価を防ぐ Fixed Target Network といった工夫により安定した学習を可能としている.

Algorithm 1 deep Q-learning with experience replay

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
for episode = 1, M **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \mathrm{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1} and preprocess \phi t + 1 = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, w_{j+1})$ from D

$$\mathrm{Set} y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \mathrm{max}_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$$

    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  end for
end for

図 2.2: DQN のアルゴリズムの疑似コード [1]

2.4 Genetic Algorithm

Genetic Alogorithm (GA) とは, 生物の進化と進化の過程を模した最適化手法であり, 主に組み合わせ最適化問題に対して適用される. GA では 1 つの解を 1 つの個体として表現し, 多数の個体からなる個体群を用いて解空間の多点

を同時に探索する. 各個体はどの程度良い解であるかという指標として適用度を持つ. また, 各個体は染色体と呼ばれる配列で表される. この染色体を構成する要素を遺伝子, 染色体上の遺伝子が収まる座標を遺伝子座と呼ぶ. 探索においては, 個体群に対して選択, 交叉, 突然変異と呼ばれる 3 種類の遺伝演算子を適用させ世代と呼ばれる探索ステップを進めていく.

選択では, 探索において適用度が良好な個体が存在する部分を重点化するように現在の個体群から個体を選び, 次世代の個体群を生成する. 選択方法としては, 個体の適応度に比例する確率で個体を選択するルーレット戦略, 個体群からランダムに数個個体を抽出しその中で最も良好な個体を選択するトーナメント戦略がある. またこれらと併用される戦略として各世代の最良個体を保存するエリート保存戦略がある. エリート保存戦略により探索で発見した良い個体が失われることを防ぐことができる. 交叉では, 2 つの個体からそれらの形質を受け継いだ新たな個体を生成する. 交叉においても染色体のランダムな 1 点で染色体を切断し部分列を染色体同士で交換する 1 点交叉, ランダムな 2 点で切断する 2 点交叉, 遺伝子座ごとランダムに交換する一様交叉など様々な方法がある. 突然変異では, 各遺伝子座の遺伝子を許容された範囲の遺伝子の内容に置換する. 交叉や突然変異にはそれぞれ確率が設定されていることが一般的である.

2.5 Nondominated Sorting Genetic Algorithm II

Nondominated Sorting Genetic Algorithm II (NSGA-II) とは,

3 提案手法

本研究では, トレーディングカードゲーム環境を作成し環境におけるデッキ間の勝率を最適化した. 最適化の過程において深層強化学習と GA を組み合わせて, 調整するカードの枚数を最小化するような最適化手法を提案する.

3.1 トレーディングカードゲーム環境

本研究において構築した環境は, Magic : The Gathering [1] に代表されるトレーディングカードゲーム (Trading Card Game : TCG) を参考にした. TCG は 2 人のプレイヤーからなるゲームである. 囲碁や将棋のように, プレイヤーは先攻と後攻に分かれてターン制で進行していく. TCG の大きな特徴として, 囲碁や将棋のように各プレイヤーが同じユニット群を持つのではなく事前に各プレイヤーの選択による異なるユニットからなるデッキを構築することが挙げられる. また, TCG はゲームタイトルごとに細かいルールは異なるが, 一般的に相手プレイヤーのカードの 1 部分はプレイヤーから観測できない不完全情報ゲームである. 以下, 実装したカードゲームのルールと用語を説明する.

3.1.1 プレイヤー

一般的な TCG と同様に, ゲームは 2 人のプレイヤーからなり, プレイヤーは複数のカードからなるデッキを持つ. プレイヤーは手札, 盤面と呼ばれるカードを保有する領域を持ち, ドローと呼ばれる操作でカードをデッキから手札に加える. また, プレイと呼ばれる操作でカードを手札から盤面に出す. また, デッキからカードが無くなった状態をデッキ切れと呼ぶ. また, プレイヤー自身が HP, マナという 2 つの整数値パラメータを持つ. プレイヤー自身の HP が 0 となればゲーム敗北となり, 相手のカードからの攻撃などで減少していく. マナはカードをプレイする際, 後述するカードのコスト分減少する. プレイヤーは残りマナを超えるコストを持つカードをプレイすることができない. 今回の環境ではプレイヤーの HP の最大値は 20, マナの上限値は初期値が 1 で最大値を 5 と設定した.

[1] https://magic.wizards.com

3.1.2 カード

カードはそれぞれ攻撃力と HP とコストの 3 つの整数値パラメータを持つ. 盤面にあるカードは対戦相手の盤面にあるカード, あるいは相手プレイヤーに攻撃することができる. カードが攻撃する際には, 相手盤面に存在する攻撃対象のカードの HP, あるいは相手プレイヤーの HP へとカードの持つ攻撃力分ダメージを与える. またカードへと攻撃する際には攻撃対象のカードが持つ攻撃力分, 攻撃するカードもダメージを受ける. ただし, 攻撃が可能となるのはカードが盤面にプレイされたターンの次のターンからになる. カードの HP が 0 になった, あるいは後述する手札と盤面の枚数制限を超えて盤面にプレイされた時はカードは破壊される. 破壊されたカードはゲームから取り除かれる. また, カードによっては以下に示す特殊効果を持つものもある.

召喚： 盤面に出したら (攻撃力 , HP) = (1 , 1) のユニットを追加で盤面に出す

治癒： 盤面に出したら自プレイヤーの HP を 2 回復する

攻撃： 盤面に出したら敵プレイヤーの HP を 2 減らす

取得： 盤面に出したら自プレイヤーは 1 枚カードをドローする

速攻： 盤面に出たターンに攻撃できる

3.1.3 ゲームフロー

以下, ゲームの流れを説明する.

1. ゲーム開始時に各プレイヤーは自身のデッキをシャッフル.
2. デッキから初期手札としてカードを 5 枚ドロー.
3. 先攻プレイヤーは 1 ターン目のドローステップをスキップし行動.
4. 後攻プレイヤーはカードを 1 枚ドローして行動.
5. 2 ターン目以降は先攻プレイヤーもカードを 1 枚ドローしてから行動.

6. 4 , 5 の繰り返し. なお, ターンプレイヤーは行動前にマナを上限値まで回復. このときマナの上限値が 5 でなければ上限値を 1 増やしてから回復.
7. プレイヤーがデッキ切れになっている状態でカードをドローしようとした, あるいはプレイヤー自身の HP が 0 となった場合はそのプレイヤーが敗北となりゲーム終了.

本構築環境では一般的な TCG と同様にカードがプレイされた次のターンから行動可能となるため, 先攻プレイヤーがカードの行動が早くなり有利となる. そのため, 先攻の 1 ターン目のドローステップをスキップしている.

3.2 関連研究

トレーディングカードゲーム環境におけるバランス調整に関して, Fernando らは HearthStone [2] 環境内においてデッキ間の勝率が 50 % となるように, 3 つのデッキの計 64 枚のカードにおける 180 個の調整可能なパラメータを 180 個の要素を持つ 1 次元配列として GA, NSGA-II を適用した [9]. GA を用いた場合, デッキ間の勝率をほぼ 50 % に近づけるという目標は達成したが, 180 個の調整可能パラメータの総変更量は 402 となり元のデッキの原型が無くなった. NSGA-II において勝率とパラメータの総変更量の 2 つの目的関数を最適化するように設定した場合, 勝率は GA を用いた場合とほぼ変わらず, パラメータの総変更量は 402 から 154 へと減少した.

しかし, NSGA-II を用いることで GA に比べてカードのパラメータの総変更量は減少したが, 変更を及ぼす領域は調整対象のカード全体に及んでいる. 本研究における提案手法は, 遺伝的アルゴリズムの解空間の次元を深層強化学習によるシミュレーションを参考に削減することで, 調整を施すカードの数を減らすことを目的としている.

[2] https://hearthstone.blizzard.com

3.3 提案手法：深層強化学習によるデッキ内のカードパワーの測定

TCG において, デッキ内の各カードのカードパワーを測る指標としてはそのカードがプレイされたときの勝率 (Win Rate when Play : WRP) などが考えられる. しかし, カードにコストが存在する場合においては低コストのカードと高コストのカードについてプレイされる回数に偏りが生じてしまう. このため, 得られる結果はカードのコストの影響が反映されてしまい意味のある結果とはいえない.

本研究では, 同デッキ同戦略のプレイヤーを先攻後攻両方に配置してカードを 1 種類ずつデッキから除いた場合の勝率をデッキ内のすべてのカードについて計算することで戦略下におけるデッキ内のカードパワーを測定する手法を提案する.

3.4 提案手法：GA における解空間の次元を削減した最適化

新たにデッキをトレーディングカードゲーム環境へと追加する際, デッキ間の勝率は $50 \pm 5\%$ になることが環境として好ましい. 3.2 節で述べたようにトレーディングカードゲーム環境におけるデッキ間の勝率の最適化において GA の有用性が知られている. 本研究では, 深層強化学習を用いたシミュレーションによってデッキ内のカードにおいて調整するべきかどうか優先順位を定義し, 変更するカードを明確にしたまま GA の解空間を削減したトレーディングカードゲーム環境の最適化手法を提案する.

4　実験方法

4.1　実験 1：深層強化学習によるエージェント構築

構築したトレーディングカードゲーム環境へ深層強化学習を適用する. 深層強化学習手法として DQN を用いて構築環境において後攻のプレイヤーとして学習し, 学習済のエージェントで 10000 回ゲームを実行して勝率を計算した. また, 学習が進み具合を把握するため学習時の獲得報酬の推移を記録した. さらに学習済エージェントを用いて 50000 回の対戦を実行し, エージェントが選択した行動の総数, 各カードごとのプレイされた総数を記録し学習済エージェントがどのような戦略を構築したか, それが人間から見て合理的な戦略かどうか考察した.

4.1.1　対戦相手の行動ルーチンと対応するデッキ

学習時, また学習後の対戦は共通して同一の対戦相手を設定した. 対戦相手の戦略として, TCG における代表的な戦略として挙げられるアグロとコントロールの 2 種類を設定した. アグロとコントロールは様々な解釈がありデッキのカードの構成として定義されることもあるが, 本研究では以下のようにカードの攻撃の際の戦略として定義する. 詳細な疑似コードは,

アグロ： 敵プレイヤーへの攻撃を優先しなるべく早くゲームエンドに持ち込む.

コントロール： 相手の盤面のカードの処理を優先し, 長期戦に持ち込む.

4.1.2　状態空間, 行動空間, 報酬の定義

強化学習では, エージェントの取りうる行動と観測できる状態の空間, 報酬を定義する必要がある. TCG ではドローやプレイ, カードの攻撃による破壊といった行動で盤面や手札の枚数が変化する場合があり, 各ステップ時点でそれぞれプレイヤーの取りうる行動の総数が変化し得る. そのため, ステップごとに行動空間の次元数が変化し上手く学習できない問題が生じる.

Algorithm 2 対戦相手の行動ルーチン (アグロ)

```
1: 手札から盤面にカードを出せるだけプレイ (ドロー順が古い方から)
2: for 自盤面のカード (プレイ順が古い方から) do
3:     if 敵の盤面にカードが無い then
4:         敵プレイヤーを攻撃
5:     else
6:         if 自身の HP が 12 以上 then
7:             敵プレイヤーを攻撃
8:         else
9:             if 敵盤面に破壊できるカードがある then
10:                そのカードを攻撃
11:            else
12:                敵プレイヤーを攻撃
13:            end if
14:        end if
15:    end if
16: end for
17: ターンを終了
```

図 4.1: アグロの疑似コード

そのため本研究では図 4.3 に示すように予め手札と盤面の枚数の上限をそれぞれ 9 枚, 5 枚と定め, 手札と盤面に存在するカードに自盤面 1 というように番号をつけ, カードが存在しない場合は状態を 0 とすることで状態空間と行動空間を定義した. 表 4.1, 4.2 に状態空間, 行動空間の定義を示す. なお, ドローやプレイといった操作でカードを追加し枚数の上限を超える場合には追加しようとしたカードを破壊する.

また, 報酬は以下のように設定した.

- 1 ステップ終了後

$$\mathrm{r} = 0.0$$

- 1 エピソード終了後

$$\mathrm{r} = \begin{cases} 1.0 & \text{(学習プレイヤーの勝利)} \\ -1.0 & \text{(敵プレイヤーの勝利)} \end{cases}$$

表 4.1: 定義した状態空間

状態説明	次元数	最小値	最大値
各プレイヤーの HP	2	0	20
各プレイヤーの マナ	2	0	5
手札 1 ～ 9 の HP , 攻撃力, コスト, 特殊効果	36	0	5
自盤面 1 ～ 5 の HP と攻撃力	10	0	5
敵盤面 1 ～ 5 の HP と攻撃力	10	0	5
自盤面 1 ～ 5 が 攻撃可能かどうか	5	0	1
お互いのデッキの 残り枚数	2	0	30

表 4.2: 定義した行動空間

行動説明	次元数
手札 1 ～ 9 を自盤面に出す	9
自盤面 1 が敵盤面 1 ～ 5 に攻撃 or 敵プレイヤーに攻撃	6
自盤面 2 が敵盤面 1 ～ 5 に攻撃 or 敵プレイヤーに攻撃	6
自盤面 3 が敵盤面 1 ～ 5 に攻撃 or 敵プレイヤーに攻撃	6
自盤面 4 が敵盤面 1 ～ 5 に攻撃 or 敵プレイヤーに攻撃	6
自盤面 5 が敵盤面 1 ～ 5 に攻撃 or 敵プレイヤーに攻撃	6
ターンエンド	1

4.1.3　DQN

4.2　実験 2 :

Algorithm 3 対戦相手の行動ルーチン (コントロール)

```
1: 手札から盤面にカードを出せるだけプレイ (ドロー順が古い方から)
2: for 自盤面のカード (プレイ順が古い方から) do
3:   if 敵の盤面にカードが無い then
4:     敵プレイヤーを攻撃
5:   else
6:     if 自盤面の行動可能なカード全てで敵プレイヤーに攻撃すれば勝利 then
7:       敵プレイヤーを攻撃
8:     end if
9:     if (敵盤面カードの総攻撃力) × 2.0 > (自盤面カードの総 HP) then
10:      敵プレイヤーを攻撃
11:    end if
12:    if 敵盤面に HP が残したまま破壊できるカードがある then
13:      そのカードを攻撃
14:    end if
15:    if 敵盤面に破壊できるカードがある then
16:      そのカードを攻撃
17:    else
18:      if 敵盤面カードの総攻撃力 > 自分の HP then
19:        敵盤面の最も攻撃力が高いカードを攻撃
20:      else
21:        敵盤面の最も HP の低いカードを攻撃
22:      end if
23:    end if
24:  end if
25: end for
26: ターンを終了
```

図 4.2: コントロールの疑似コード

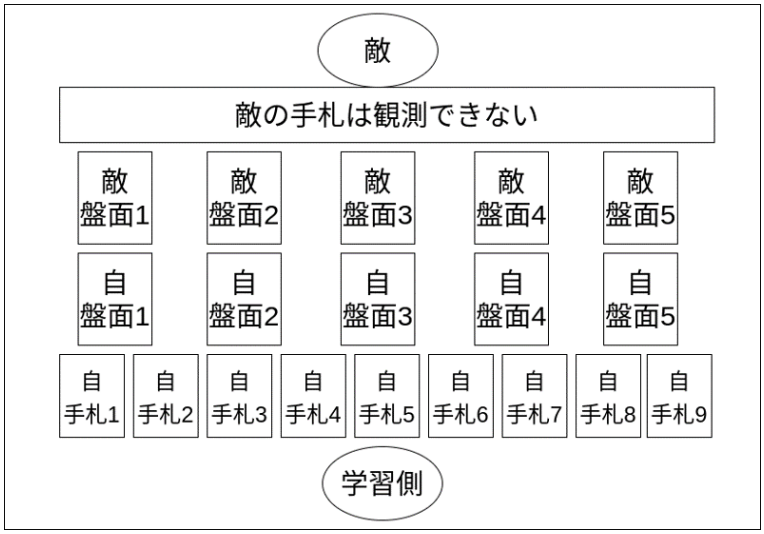

図 4.3: 深層強化学習における諸定義イメージ

5　結果と考察

6　まとめと今後の課題

謝辞

年 月 日

参考文献

[1] 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*, Vol. 518, No. 7540, pp. 529–533, February 2015.

[2] Andreas Folkers, Matthias Rick, and Christof Büskens. Controlling an autonomous vehicle with deep reinforcement learning. *CoRR*, Vol. abs/1909.12153, , 2019.

[3] Shixiang Gu*, Ethan Holly*, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Piscataway, NJ, USA, May 2017. IEEE. *equal contribution.

[4] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. Drn: A deep reinforcement learning framework for news recommendation. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, p. 167–176, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee.

[5] 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. Mastering the game of go without human knowledge. *Nature*, Vol. 550, pp. 354–, October 2017.

[6] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, Vol. 362, No. 6419, pp. 1140–1144, 2018.

[7] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv e-prints*, p. arXiv:1606.01540, June 2016.

[8] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. 2013. cite arxiv:1312.5602Comment: NIPS Deep Learning Workshop 2013.

[9] Fernando de Mesentier Silva, Rodrigo Canaan, Scott Lee, Matthew C. Fontaine, Julian Togelius, and Amy K. Hoover. Evolving the hearthstone meta. *CoRR*, Vol. abs/1907.01623, , 2019.