MAP-Elites を用いた MountainCar のシミュレーション

清水川　七海　(指導教員：オベル加藤ナタナエル)

1　はじめに

MAP-Elites のような Quality Diversity アルゴリズム（以下 QD アルゴリズム）は従来の単目的最適化方法の代わりとして考案されたアルゴリズムである [1]。良質な複数の異なる解に注目することで発散と収束の間の適正なバランスをとることができる。このアルゴリズムは元々迷路問題などの進化ロボティクスの問題に応用されていた。この研究では Map-Elites を利用して Mountain car というベンチマークのシミュレーションを行った。

1.1　Map-Elites について

MAP-Elites はその単純性と、進化ロボティクスの分野で優れた結果を残したことで QD アルゴリズムの中で最も有名なアルゴリズムである。連続空間に離散化された格子を設定し、各格子に最も優れた個体を保有させる。このようにすることで一度に多様な優良結果を保持するこができる [1]。

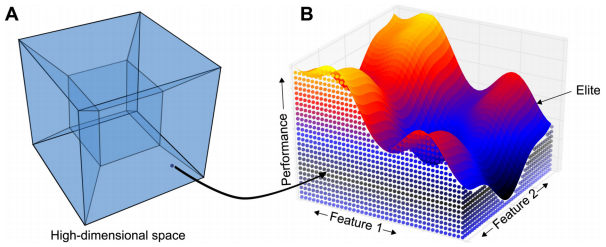
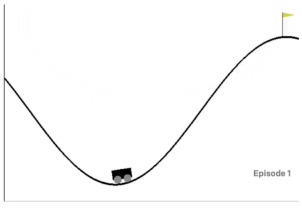


図 1: Map-Elites の概略

1.2　MountainCar について

MontainCar 問題は強化学習の分野の基本的な OpenAIgym のベンチマークである。空間には 2 つの山があり、その谷部に車が初速度 0 で設置されている。本テストのゴールは車が右の山を登り切ることだ。車のエンジンが弱いため、山を登る唯一の方法は左右の山を往復して助走をつけることである。また、左の山を登ることに対してのペナルティはなく、各エピソードは右の山頂に達した時か 200 回の反復を繰り返した時に終了する。入力は速度と位置の 2 つで出力は左に押す、右に押す、どちらにも押さないの 3 つである。



Num	Action
0	push left
1	no push
2	push right

Num	Observation	Min	Max
0	position	-1.2	0.6
1	velocity	-0.07	0.07

図 2: MountainCar の概略 [2]

2　研究方法

この研究では MAP-Elites の性能を調べるにあたって、QD アルゴリズムを実行するフレームワークである QDpy[3] を使用して主に 3 つの実験を行った。

実験 1：レイヤーの大きさを変える

入力と出力の間のレイヤーのサイズを変えて結果の性能がどのように異なるかを観察する。レイヤーの大きさはそれぞれ 1,5,10,15,100,[10,10] に設定した。

実験 2：MountainCarContinuous-v0 環境 [4] で実験する

実験 1 と実験 3 は MountainCar-v0 環境 [5] で実行されている。2 つの環境は同じ山と車で構成されているが、この新しい環境では山頂に達するまでのエネルギーが小さい程結果の値が良くなる。

実験 3：特徴の組み合わせを変える

実験で使った特徴は x 軸、y 軸、距離、車の速度の 4 つである。よって全部で 6 種類の組み合わせを比較した。

3　シミュレーション結果

実験 1

この実験では横軸に距離、縦軸に速度を用いた。

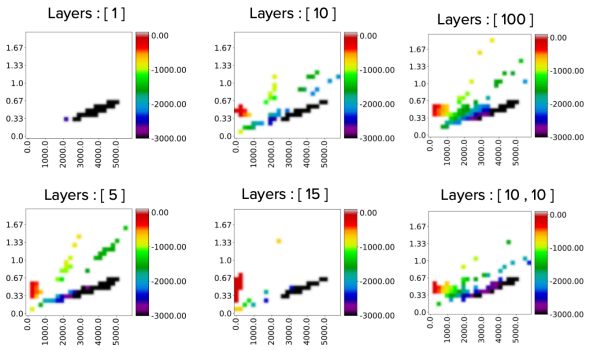


図 3: 実験 1 の結果

レイヤーが 1 つしかない時は結果の値が低く、範囲も狭いことがわかる（図 3(左上)）。しかしレイヤーのサイズが 5 以上の実験では概ね良い結果となった。どれも同じような分布になっていて、全体的に左下の部分に良い結果が集中していることがわかる。

実験 2

図 4 は特徴の組み合わせとその範囲が実験 1 と同条件で、環境のみ変えた場合の結果である。全体的に精度が落ちたように見えるが、分布は実験 1 と似た形となった。また縦軸の速度の範囲を [0,2.0] から [0,1.5] に変更して同様に実験を行った。その結果が図 5 である。

速度の範囲を狭めた方が結果の精度は良くなった。ただ分布の範囲には概ね変化が無かった。これらの結果より環境が変わっても実験結果にはそれほど影響しないことがわかった。

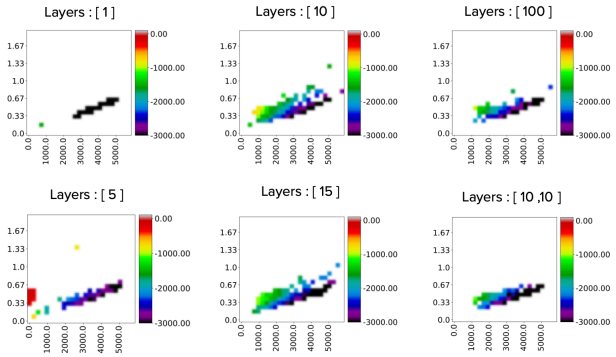


図 4: 実験 2 の結果 (速度範囲 [0,2.0])

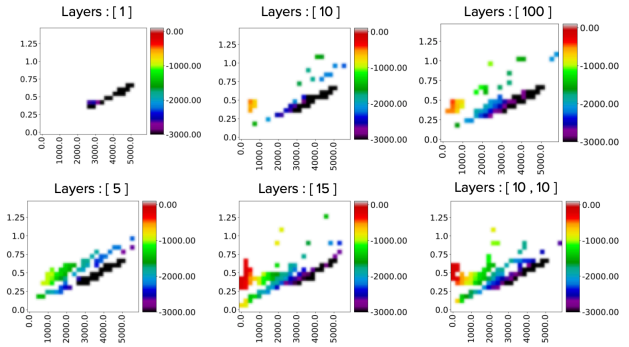


図 5: 実験 2 の結果 (速度範囲 [0,1.5])

実験 3

実験 1 よりレイヤーの大きさが 5 でも十分な結果が得られることがわかったので、まずレイヤーのサイズを 5 に設定して実験 3 を行った。図 6 の上段の縦軸は全て車の速度であり横軸は左から x 軸、y 軸、距離である。また下段の左 2 つの縦軸は y 軸、横軸が左から x 軸、距離、右下の縦軸が x 軸、横軸が距離である。この時車の速度の範囲は [-0.07,0.07] に設定されている。

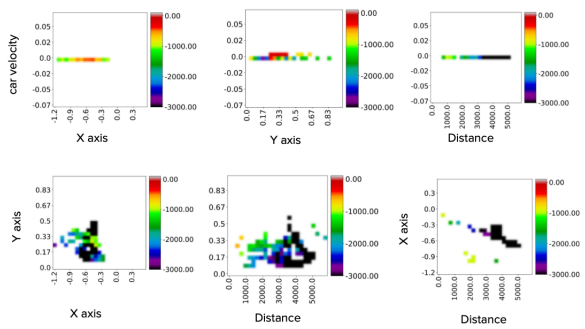


図 6: 実験 3 の結果 (レイヤー [5], 車の速度範囲 [-0.07,0.07])

取り扱う特徴が変わったので分布の形が既存の実験と明らかに変わった。また図 6 の上段の結果が直線のような形になっているが、これは車が常に前後を行き来する運動をしているので山頂に到達した時は正の数、それ以外の時は 0 を返すからである。これらを踏まえて、レイヤーのサイズを 100 に増やして車の速度範囲を [0,0.07] に変更し、その他の条件は変えずに同じ実験を行った。その結果が図 7 である。上段は範囲が狭められたこともあり結果が全体的に良くなったが、下段左 2 つはレイヤーを大きくしたことで逆に結果が悪くなった。

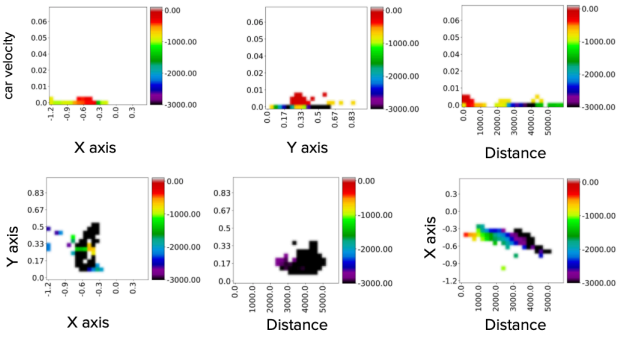


図 7: 実験 3 の結果 (レイヤー [100], 車の速度範囲 [0,0.07])

4 まとめと今後の課題

これらの実験を経て MAP-Elites の性能を知ることができた。しかし QD アルゴリズムの利点が良質で多様な解を保有することであるのに対し、MountainCar の結果では解が一部に偏ってしまい幅広い解を得ることができなかった。今後は QD アルゴリズムの他のアルゴリズムを同じベンチマークに適用することでそれぞれのアルゴリズムの性能を比較し、より幅広い解が求まるものがあるかを調べていきたい。

参考文献

[1] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. 2015.

[2] Mountaincar v0. `https://github.com/openai/gym/wiki/MountainCar-v0`.

[3] Leo Cazenille. Qdpy. `https://gitlab.com/leo.cazenille/qdpy`.

[4] Source of mountaincarcontinuous-v0. `https://github.com/openai/gym/blob/master/gym/envs/classic_control/continuous_mountain_car.py`.

[5] Source of mountaincar-v0. `https://github.com/openai/gym/blob/master/gym/envs/classic_control/mountain_car.py`.