GVG-AIのためのMonte Carlo Tree Searchの改善に関する研究

Oh HyunWoo[1,a)]　金子 知適[2,3,b)]

概要：General Game Playing は，未知で多様なゲームをプレイできるゲーム人工知能の構築を目的とする研究分野である．General Video Game-AI (GVG-AI) は，その中で，ビデオゲームを対象としているものである．本研究では GVG-AI によく用いられている Monte Carlo Tree Search の改善のために，2 点の新しい変更を提案する．1 点目は，未来の小さい報酬を発見する速度の向上のために追加報酬を利用する GreedyUCB1 に mixmax backups を適用して，GreedyUCB1 より広い範囲に正確な探索をするようにしたもので，MixMax-Greedy-UCT と呼ぶ．2 点目は，エージェントの行動の頻度によって行動にペナルティーを付与することで，新たな行動の実行を誘導する効果をねらったもので，Novelty of Action based Penalty (NAP) と呼ぶ．既存の plain-UCT を適用したエージェントと，MixMax-Greedy-UCT を適用したエージェント，NAP を適用したエージェント，MixMax-Greedy-UCT と NAP を一緒に適用したエージェントを GECCO の 2015 年のゲームセットで評価した結果，NAP でエージェントの性能を向上させることが示された．

Enhancements of Monte Carlo Tree Search for GVG-AI

Oh HyunWoo[1,a)]　Tomoyuki Kaneko[2,3,b)]

Abstract: General Game Playing(GGP) is aimed to develop game AI agents that can play diverse games without pretraining. General Video Game-AI (GVG-AI) is specialized for GGP in video games. In this research, we propose two enhancements for the Monte Carlo Tree Search which is commonly used for GVG-AI. The first one is to apply mixmax backups to GreedyUCB1 which gives additional rewards to improve the speed of finding small rewards of the future, to make an accurate search for wider range than GreedyUCB1. We call this method as MixMax-Greedy-UCT. The second one is for exploration of new states by giving a penalty to frequent actions. We call this method as Novelty of Action based Penalty (NAP). Our experiments comparing agent applying plain-UCT, agent applying MixMax-Greedy-UCT, agent applying NAP, agent applying MixMax-Greedy-UCT and NAP with GECCO 2015's game set showed that agent with NAP make better performance than the other agents.

1. Introduction

最近の人工知能分野には輝かしい発展が続いている．しかしながら，人の知能が様々な分野に置いて自ら適応できることと比較すると，最近までのの人工知能は次のような二つの課題があると考えられる．(1) 事前に設定や学習を済ませておく必要があり，環境に合わせて臨機応変に対応することができない．(2) 特定分野に専門的に特化した人工知能エージェント/プログラムは，適用範囲がその分野に限定される．囲碁のために開発された人工知能プログラムがチェスをプレイすることはできない．課題 (1) に対しては，人工知能が自ら学習することのできる強化学習技法の発展により，かなり研究が進んでいる．しかし，課題 (2)

1 東京大学大学院 情報学環・学際情報学府
Graduate School of Interdisciplinary Information Studies, the University of Tokyo
2 東京大学大学院情報学環
Interfaculty Initiative in Information Studies, the University of Tokyo
3 国立研究開発法人科学技術振興機構さきがけ
JST, PRESTO
a) hyunwoo-oh@g.ecc.u-tokyo.ac.jp
b) kaneko@acm.org

を解決するための研究については，まだ初期段階にすぎない．

一方，「実社会」という環境は複雑で，動的で，データ収集に対する難しさがあるため，実社会のための人工知能を現時点で開発することは難しい．「ゲーム」は，実社会環境の複雑さに制限を加え，人工知能開発を容易にしたテスト環境という役割を果たしている．つまり，ゲーム人工知能の開発は，実社会環境に適用できる人工知能の開発の一歩だと言える．

General Game Playing (GGP) は，「ゲーム」環境を用いて汎用性を持つ人工知能を研究をする分野である．スタンフォード大学から始まった GGP の目的は，未知で多様なゲームをプレイすることができるゲーム人工知能を構築することである．初期の GGP はボードゲームを主に扱ったが，ビデオゲームを扱う新しいプラットホームである General Video Game Playing (GVGP) が，最近は活発に研究されている．

GVGP には二つのメインフレームワークがある．まず，Atari 2600 というゲームをプレイするエージェントの開発を目的にする，Arcade Learning Environment (ALE) が作られた．次に，GVG-AI という，Video Game Description Language で書かれたビデオゲームをプレイすることができるエージェントを開発することを目的にする分野が続いた．[9]

Monte Carlo Tree Search (MCTS) は，優れた性能のため，General Video Game AI (GVG-AI) 分野だけではなく，ゲーム人工知能全般で幅広く活用されている．[2] MCTS は現在のゲームの状態から未来のより良い状態につながる行動を探索することを目的に，ランダムシミュレーションを利用して行動の良さを推定する．GVG-AI の場合，ゲームの事前知識が全くない状態からプレイを行わなければならないため，ゲームの知識なしに動作する MCTS が適している．

本研究では，GVG-AI フレームワークを活用して，既存の MCTS を基盤とした人工知能エージェントを改善するための方法を提案する．

2. Background

第 2 章では，GVG-AI と，強化学習の代表的な問題である Multi-Armed Bandit Problem, それを解決するための政策である Upper Confidence Bound について説明する．

2.1 GVG-AI Competition

GVG-AI Competition は，2014 年から始まった [5]． 製作したエージェントを GVG-AI サイトに登録すると，エージェント制作者には未知のゲームセットを用いてエージェントの性能が検証される．その結果で，Competition に参加した他のエージェントと競争する．GVG-AI Competition では，同じ環境でエージェントを製作するようにフレームワークを提供している．また，テストのための 92 個のゲームを提供している．GVG-AI フレームワークでエージェントは，ゲームの現在の状態と可能な行動を得る．ゲームをプレイする時，エージェントの行動によってゲームの状態がアップデートされる．

2.2 Multi-Armed Bandit Problem

1952 年 Robbins により始めて提案された MAB 問題は，複数のスロットマシンをプレイする時に，引く腕を選択して，合計の報酬を最大化する問題である．多くの状況で我々は目の前にある最も多くの報酬を得られる選択をするが，その選択で絶対に最大の報酬を得られるとは限らない．つまり，目の前の最も高い利益を期待して選択するか，未来のより良い報酬を期待して現在の利益を犠牲するかのトレードオフがあり，多くの研究が活発に行われている．

2.3 Upper Confidence Bound

MAB 問題を解決するために，Upper Confidence Bound を用いることが有用であると知られている．最も簡単な UCB は文献 [1] が提案した UCB1 で，各ステップで最も有望な選択をするための公式である．

$$\mathrm{UCB1} = \overline{X_J} + \sqrt{\frac{\ln N}{N_J}} \qquad (1)$$

UCB1 は Exploitation term と Exploration term の組み合わせで構成されている．Exploitation term は腕 J の平均報酬を意味する $\overline{X_J}$ から構成されており，現在の最も高い利益を期待して選択することを意味する．Exploration term の N_J は腕 j が引かれた回数であり，N は今までプレイした回数である．このタームは，引かれた回数がより少ない腕を引くように誘導する役割を持っている．つまり，未来のより良い報酬を期待して，現在の利益を犠牲にする選択も考慮される．

3. Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) は，確率的シミュレーションを用いた最良優先探索手法であり，有限な長さのどのゲームにも適用できる．[3] さらに，ゲームに関する事前知識を必要としていないため，GVG-AI フレームワークに適する．

本章では，MCTS のアルゴリズムと，その特徴を説明する．

3.1 Algorithm

MCTS アルゴリズムで探索木の節点はゲーム状態を，辺はエージェントの行動を意味する．MCTS アルゴリズムは図 1 のように Selection, Expansion, Simulation, Back-propagation の 4 つの過程を繰り返して実行される．

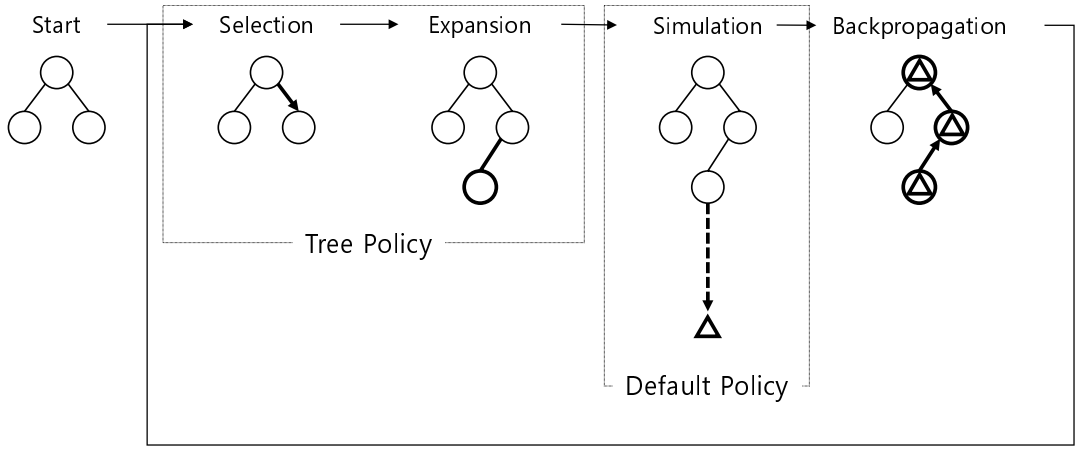

図 1 One iteration of the general MCTS approach.

(1) Selection: root node から始めて，子ノードを選択する政策 (tree policy) を，拡張ができるノードまでツリーを下りながら反復的に実行する．ここで，ノードは non-terminal 状態であって訪問していない子ノードがあると拡張できるとする．
(2) Expansion: その状態での実行できる行動に対応して，一個または複数の子ノードを追加し，ツリーを拡張する．
(3) Simulation: 新たに生成されたノードで default policy によって Simulation を実行し，報酬を得る．
(4) Backpropagation: Simulation の結果をツリーを登って back up する．

ここで，エージェントの行動を決定する政策 (Policy) があって，Selection と Expansion で使うものを Tree Policy，Simulation で使うものを Default Policy と呼ぶ．

(1) Tree Policy: 探索木に存在しているノードからエージェントの行動を決定する．
(2) Default Policy: シミュレーションにおいて，ドメインの non-terminal 状態からエージェントの行動を決定する．

3.2 Upper Confidence Bound for Tree

UCB を MCTS に適用するために Exploration term の影響力を調整する部分において，式 (2) のように修正して用いる．これを Upper Confidence Bound for Tree (UCT) と呼ぶ．

$$\mathrm{UCT} = \overline{X_J} + C_P\sqrt{\frac{\ln N}{N_J}} \qquad (2)$$

MCTS アルゴリズムで UCT は Tree Policy に用いられ，最も有望だと推測されるノードを選択する．

3.3 Algorithm's Characteristics

MCTS の最も主要な長所は，ドメインの事前知識が不必要であることである．[2] MCTS は様々な選択肢を simulation しながら最も有望な行動を探して行く．そのため，(1) ビデオゲームの予測不可能という性質に対応しやすい．(2) 悪い選択をする可能性が低い．また，どの時点でもアルゴリズムを中断して結果を得られるため [2]，(3) 計算的な側面で効率が良い．[10]

4. Related Research

本章では，GVG-AI フレームワークを基盤とした MCTS を改善する研究について説明する．

4.1 GreedyUCB1

GVG-AI Competition では，すべてのゲームステップにおいて，40 ms の内に行動を決める必要がある．従って時間的に効率の良い探索が重要である．しかし，GVG-AI フレームワークの一部のゲームにあるように，報酬の発生頻度が非常に少ない場合，または，何段階もの未来に小さい報酬を見つける必要がある場合，UCT では，この報酬情報が無意味になる程縮小されてしまうので，報酬を得る方向に十分な探索を行うことができない．このような問題を解決するために，報酬が発生するゲームの行動に追加報酬 t_J を与える GreedyUCB1 が提案されている．[8]

$$\overline{X_J} + C_P\sqrt{\frac{\ln N}{N_J}} + t_J \qquad (3)$$

t_J は，行動 a_J に対する報酬の累積値である．MCTS の各ステップを繰り返しながら子ノードが得た報酬を親ノードに 累積する．GreedyUCB1 は，最初には t_J の値が小さいため既存の MCTS と類似した探索を行うが，報酬が発生する状態を見つけ t_J の値が大きくなると，X_J の値が少なくても t_J の値が大きい方向に探索を行う．[8]

4.2 MixMax Backups

Mixmax backups は UCT の Exploitation term に適用する手法で，式 (4) のようになる．

$$Q \times \mathrm{Maxscore} + (1-Q) \times \overline{X_J} \qquad (4)$$

Q は 0 以上 1 以下の値であり，Maxscore と $\overline{X_J}$ の重み付き平均に用いられる．Q の値が大きいほど Exploitation term を構成する Maxscore の比率が上がり，Exploitation term の全体の値が高くなる．つまり，エージェントは報酬を得るためにリスクを許容する行動を選択する．この手法は MCTS を Super Mario Bros に適用した時，「Mario が穴の前で，ジャンプで越えることができてもジャンプせずに，怖がる」という問題を解決するために提案された．[6]

4.3 Reversal Penalty

従来の MCTS エージェントは，ある隣接した場所に行動しては，元の場所に戻るなど，振動のような動きをする場合がある．この問題は，ゲーム中の各時刻で MCTS が起動される度に，少ない回数のシミュレーションが実行され，その中で偶然に大きい報酬を持ったいくつかの行動が UCT に大きな影響を与えることで，発生する可能性が高い．[5] このような問題を解決するために文献 [5] では Reversal Penalty が提案された．Reversal Penalty は，エージェントの最近の 5 つの行動をセットとして保持して置いて，また行動を選択する時に UCT により選択された行動が保持した行動セットにある場合，UCT 値に微細なペナルティーを付与する方法である．該当論文では 0.95 を掛ける方式で用いた．

5. Enhancements of MCTS

既存の MCTS を GVG-AI に適用するために次のような改善手法を提案する．

5.1 MixMax-Greedy-UCT

GreedyUCB1 は追加報酬を得やすいゲームに対しては良い探索が可能であるため，既存の MCTS より良い性能を示したが，高速な探索より正確な探索が必要なゲームに対しては既存の MCTS より低い性能となった．[8] これを改善するために mixmax backups を導入し GreedyUCB1 を，式 (5) のように変更する．

$$Q \times \text{Maxscore} + (1-Q) \times \{\overline{W_J} + t_J\} + C_P \sqrt{\frac{\ln N}{N_J}} \quad (5)$$

式 (3) と比べると，mixmax backups の Q による調整が追加されている．Q の調整によって，GreedyUCB1 より広い範囲に正確な探索ができると期待される．

5.2 Novelty of Action Based Penalty

既存のペナルティーの手法には文献 [5] の Reversal Penalty がある．Reversal Penalty で 0.95 という 微細なペナルティーを付与した理由は，保持した行動セットの行動と一致しない他の行動には大きい影響を与えないためである．しかし，もし，エージェントの可能な行動の数が少ない場合，同じ行動が多数のペナルティーを得て，意図より高いペナルティーを付与される場合がある．この問題を解決するために，最近の 5 つの行動を保持する方式ではなく，エージェントの選択可能な行動の数により，可変的に保持する数を変更する方法が提案されている．[7]

文献 [5] と文献 [7] の方法はいつも同じ値のペナルティーを付与する．これは，すでに多く実行した行動にも同じペナルティーを付与し，新たな行動を実行するまでの時間がかかる問題がある．

新しい状態の探索を誘導する方法の一つとして，文献 [11] では，状態の新しさを定義して新しくない状態を pruning することで，新たな状態を探す方法が提案されている．本研究では，新しさというアイデアを継承して，行動の新しさを定義し，新しくない行動にペナルティーを付与することで，新しい行動を選択する確率を上げる方法を提案する．新たに実行した行動が保持した行動セットに含まれている頻度によってペナルティーを付与することで，新たな行動をより早く選択するようにする．

$$\text{Penalty} = 1 - \frac{\text{n}}{\text{size}} \times \text{penalty rate} \quad (6)$$

$$\text{NAP} = \left[\overline{X_J} + C_P \sqrt{\frac{\ln N}{N_J}} \right] \times \text{Penalty} \quad (7)$$

式 (7) のように UCT の値に Penalty をかけることでペナルティーを付与する．ここで n は，新しく実行した行動が保持した行動セットに一致している回数で，size は保持行動セットの大きさを調整する定数である．penalty rate はペナルティーの大きさを調整する定数である．

6. Experiments

6.1 GVG-AI Game

本研究では，http://www.gvg-ai.net に公開されているゲームセットを用いる．まず，トレーニングゲームセットを用いてエージェントを学習させ，テストゲームセットを用いて性能を評価する．本研究のトレーニングゲームセットは，GECCO 2015 の training set であり，テストゲームセットは GECCO 2015 の validation set である．二つのゲームセットは両方 10 個のゲームで構成されている．

6.2 Results of Training Set

本研究で提案した MixMax-Greedy-UCT と，Novelty of Actions Based Penalty を用いるエージェントをトレーニングゲームセットを用いて学習させた．MixMax-Greedy-UCT の Q の値は，0 から 1 まで，0.05 ずつ変更しながら最適値を探した．Novelty of Action Based Penalty は size を 1 から 5 まで 1 ずつ，penalty rate を 0.05 から 0.95 まで 0.05 ずつ変更しながら最適値を探した．異なる定数を持つ各エージェントを，10 個のゲームを用いて，各ゲームご

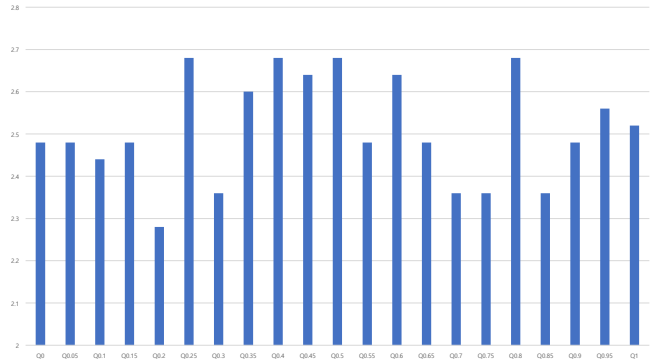

図 2　Q の調整による MixMax-Greedy-UCT の性能変化

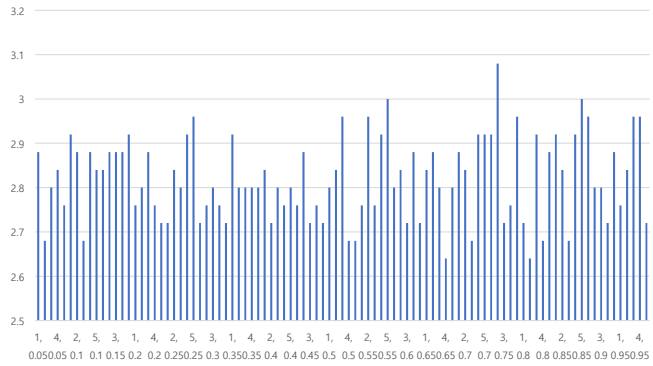

図 3　size, penalty rate の調整による NAP の性能変化

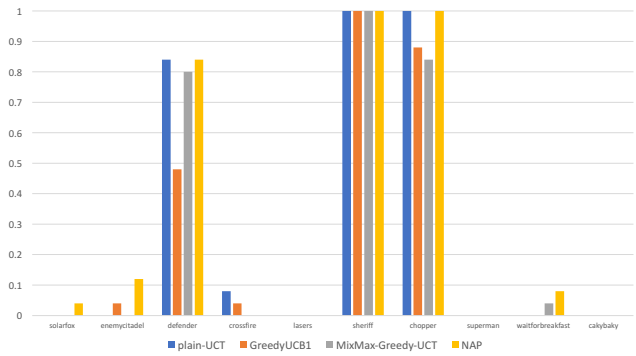

図 4　Results of training Set

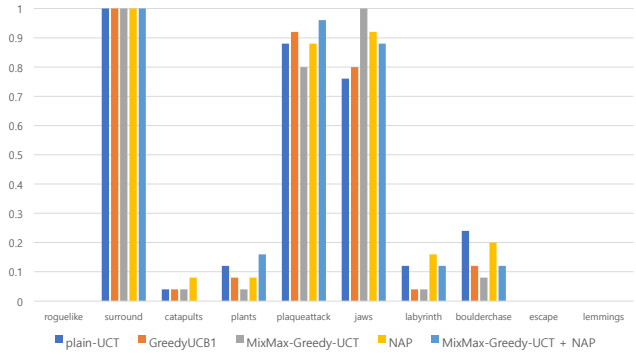

図 5　Results of Test Set

と 5 段階のレベル × 5 回 で，合計 25 回プレイした．勝率の合計が最も高くなる Q，size，penalty rate の値を最適値だと判断し，その値を用いてテストゲームセットで性能を評価した．勝率の最も高いエージェントが複数の時には，ゲームスコアの合計が最も高いものを最適だと判定した．

Q の調整により MixMax-Greedy-UCT の性能を評価した結果を，図 2 に示す．最も勝率が高い Q の値は 0.25, 0.4, 0.5, 0.8 であった．その中でゲームスコアの合計が最も高いものは 0.25 であったので，Q の最適値は 0.25 だとする．size, penalty rate の調整により NAP の性能を評価した結果は，図 3 である．size の最適値は 2，penalty rate の最適値は 0.75 であった．実験結果で，ゲームごとの勝率は図 4 と表 1 に示す．

6.3　Results of Test Set

トレーニングゲームセットで求めた定数を用いて既存手法の plain-UCT と GreedyUCB1，提案手法の MixMax-Greedy-UCT, NAP, MixMax-Greedy-UCT + NAP のエージェントの性能をテストゲームセットで評価した．図 4 と表 2 はその結果である．結果としては NAP のみ適用したエージェントが最も高い勝率を見せた．GreedyUCB1 を改善するための手法である MixMax-Greedy-UCT は，トレーニングゲームセットでは改善があるように見えたが，テストゲームセットで評価した結果，改善が明確ではなかった．Novelty of Action based Penalty は，plain-UCT と MixMax-Greedy-UCT に適用した結果，明確な改善があった．しかし，よくプレイできるゲームの傾向には NAP の有無で違いはあまりなかった．特に，パズル係のゲームのように，最初の選択がゲームの勝敗に大きな影響を与えるゲームにおいては，勝率がとても低かった．

7.　Conclusions and Future Works

本研究は，GVG-AI のために既存の MCTS の拡張手法である GreedyUCB1 を改善するために，mixmax-backups を適用する MixMax-Greedy-uct と，エージェントが新たな行動をするようにペナルティーを付与する Novelty of Action based Penalty を提案した．実験の結果，MixMax-Greedy-uct は明確な改善がなかったが，Novelty of Action based Penalty では明確な改善が認められた．

しかし，図 4 と，図 5 を見ると性能が高いゲームの種類には制限があることが確認できた．これはつまり，提案した全てのエージェントが特定の種類のゲームのみをうまくプレイできたことを意味する．トレーニングゲームセットでの MixMax-Greedy-UCT の結果である図 6 でも，Q の値とは関係なく，得意なゲームと不得意なゲームが決まっていることがわかる．また，同じくトレーニングゲームセットでの NAP の結果である図 7 でも，size, penalty rate の値とは関係なく，得意なゲームと不得意なゲームが決まっていることがわかる．GreedyUCB1 手法は追加報酬を利用して，未来の小さい報酬を探索する速度の向上を図る．[8]

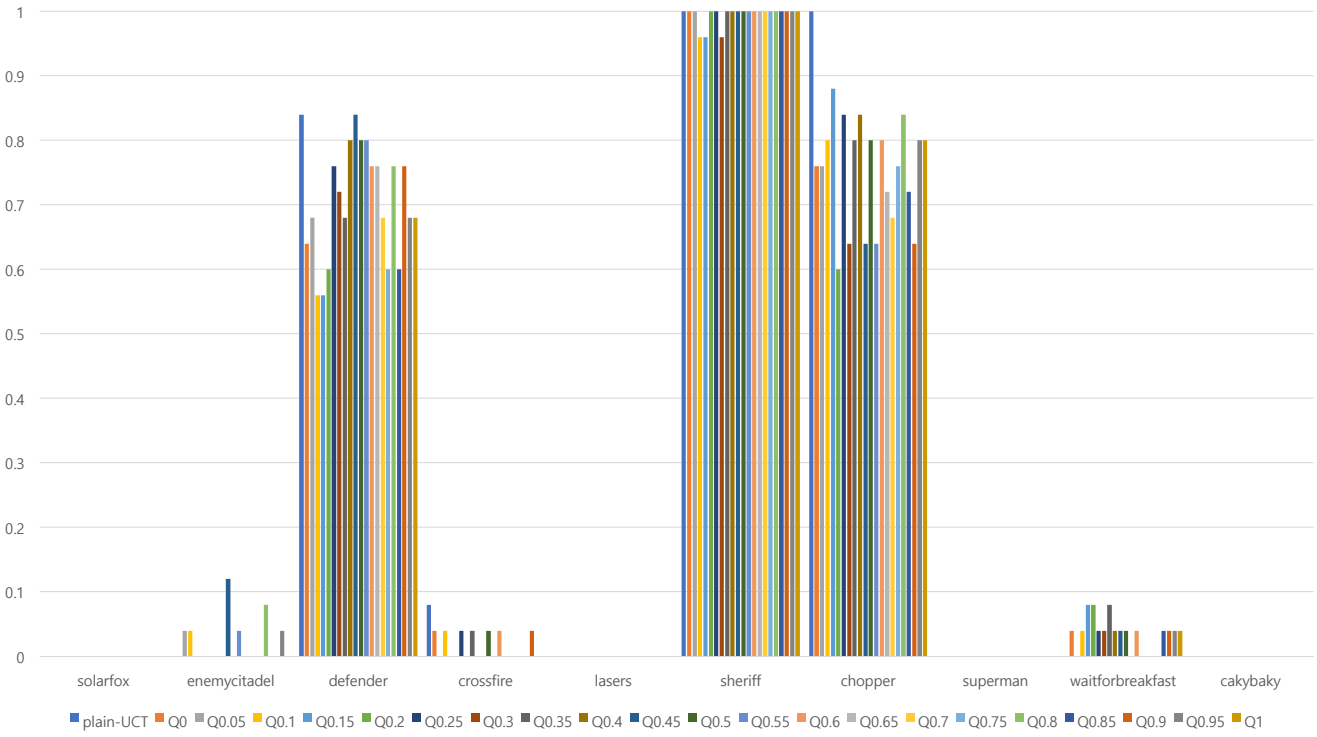

図 6　トレーニングゲームセットでの Q の調整による MixMax-Greedy-UCT の性能変化

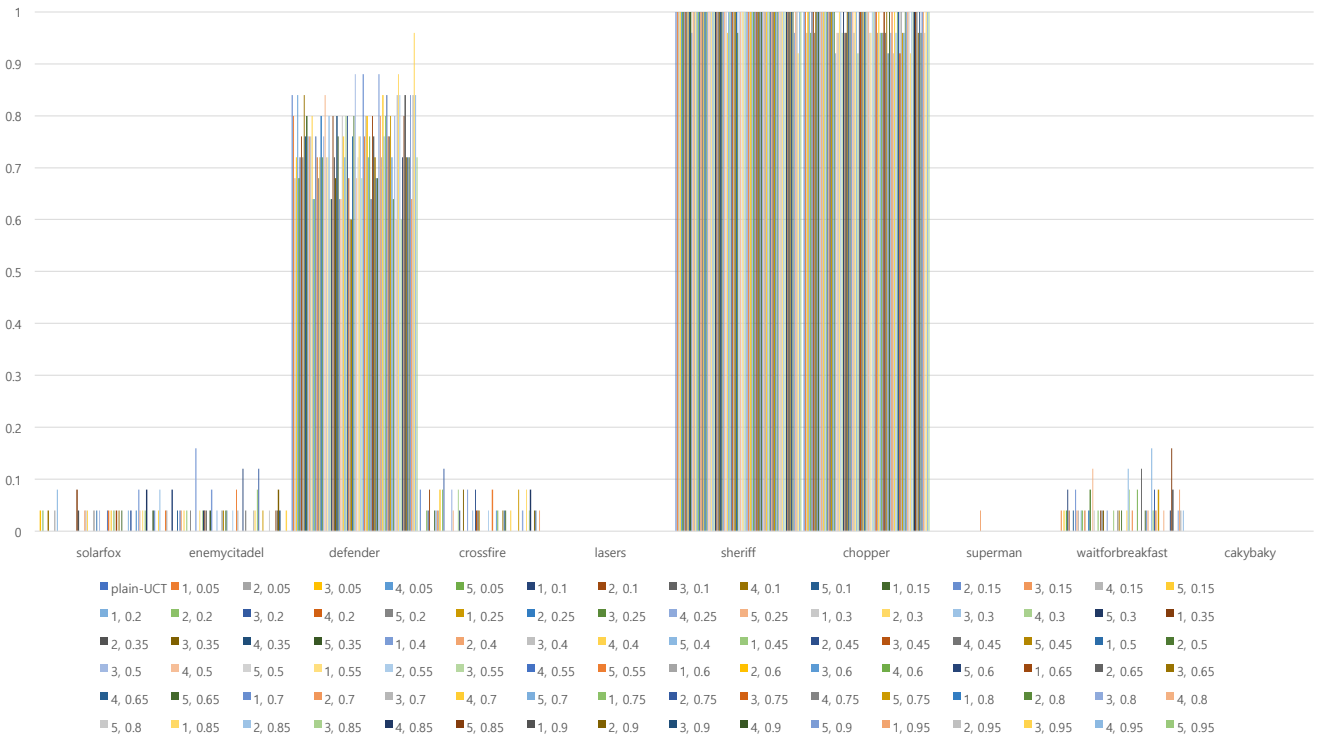

図 7　トレーニングゲームセットでの size, penalty rate の調整による NAP の性能変化

表 1 Results of Training Set

game \agent	GreedyUCB1	MixMax-Greedy-UCT	NAP
solar fox	0.0	0.0	**0.04**
enemy citadel	0.04	0.0	**0.12**
defender	0.48	0.76	**0.84**
crossfire	**0.04**	**0.04**	0.0
lasers	0.0	0.0	0.0
sheriff	**1.0**	**1.0**	**1.0**
chopper	0.88	0.84	**1.0**
superman	0.0	0.0	0.0
waitforbreakfast	0.0	0.4	**0.08**
cakybaky	0.0	0.0	0.0
sum	2.44	2.68	**3.08**

表 2 Results of Test Set

game \agent	plain-UCT	GreedyUCB1	MixMax-Greedy-UCT	NAP	MixMax-Greedy-UCT + NAP
roguelike	0.0	0.0	0.0	0.0	0.0
surround	1.0	1.0	1.0	1.0	1.0
catapults	0.04	0.04	0.04	**0.08**	0.0
plants	0.12	0.08	0.04	0.08	**0.16**
plaque attack	0.88	0.92	0.8	0.88	**0.96**
jaws	0.76	0.8	**1.0**	0.92	0.88
labyrinth	0.12	0.04	0.04	**0.16**	0.12
boulder chase	**0.24**	0.12	0.08	0.2	0.12
escape	0.0	0.0	0.0	0.0	0.0
lemmings	0.0	0.0	0.0	0.0	0.0
sum	3.0	3.16	3.0	**3.32**	3.24

また，Novelty of Action based Penalty は，ある行動の実行回数によってペナルティーを付与することで，より早く新たな行動を実行することを図る．しかし，パズルなどのような一部のゲームでは迅速な探索より正確な探索の方が重要であるので，提案した手法が低い性能を見せたと考えられる．

今後の課題として，性能が低かったゲームでもよくプレイできるように改善する必要がある．ランダムシミュレーションで選ぶ行動の確率を，スコアを上げるように変更する手法で，文献 [4] の手法がある．これを応用して，同じ場所を避けて新たな場所に行くようにする研究が考えられている．また，Novelty of Action based Penalty は，新しい行動を実行するように誘導する手法であるが，文献 [11] に提案されている Novelty Based Pruning は新たなゲーム状態に進むように誘導する手法である．Novelty Based Pruning を参考して，pruning ではなく，Penalty を付与する方法に関する研究を考えている．また，GVG-AI の研究には，MCTS に関する研究以外にも組み合わせ特徴の自動生成や，GVG-AI のためのゲーム作り研究など，様々な研究テーマがある．特に，文献 [12] の研究は，どうぶつしょうぎなどの GGP における価値関数の学習の際に自動的に組み合わせ特徴を生成する手法を提案した．これを参考にして，GVG-AI における価値関数の学習の際に自動的に組み合わせ特徴を生成する研究も有力と考えられる．

謝辞

この研究の一部は，JSPS 科研費 16H02927 と JST さきがけの支援を受けています．

参考文献

[1] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[2] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[3] G. Chaslot, S. Bakkes, I. Szita, and P. Spronck. Monte-carlo tree search: A new framework for game ai. In *AIIDE*, 2008.

[4] C.-Y. Chu, S. Ito, T. Harada, and R. Thawonmas. Position-based reinforcement learning biased mcts for general video game playing. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*, pages 1–8. IEEE, 2016.

[5] F. Frydenberg, K. R. Andersen, S. Risi, and J. Togelius. Investigating mcts modifications in general video game playing. In *Computational Intelligence and Games (CIG), 2015 IEEE Conference on*, pages 107–113. IEEE, 2015.

[6] E. J. Jacobsen, R. Greve, and J. Togelius. Monte mario:

platforming with mcts. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 293–300. ACM, 2014.

[7] P. Oh, J.-M. Kim, S.-J. Kim, and S. Hong. Enhanced mcts algorithm for generating ai agents in general video games. In Korean. *The Journal of Information Systems*, 25(4):23–36, 2016.

[8] H. Park, H. Kim, and K. Kim. Greedyucb1 based monte-carlo tree search for general video game playing artificial intelligence. In Korean. *KIISE Transactions on Computing Practices*, 21(8):572–577, 2015.

[9] D. Perez-Liebana, S. Samothrakis, J. Togelius, S. M. Lucas, and T. Schaul. General video game ai: Competition, challenges and opportunities. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[10] B. Ross. General video game playing with goal orientation, 2014.

[11] D. J. Soemers, C. F. Sironi, T. Schuster, and M. H. Winands. Enhancements for real-time monte-carlo tree search in general video game playing. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*, pages 1–8. IEEE, 2016.

[12] 藤田康博, 鶴岡慶雅, 伊庭斉志, et al. General game playing のための組み合わせ特徴の自動生成. **ゲームプログラミングワークショップ** *2014* **論文集**, 2014:180–187, 2014.

