修 士 論 文 の 和 文 要 旨

研究科・専攻	大学院　情報理工　　学研究科　　情報・通信工学　　専攻　博士前期課程		
氏　　名	武藤　孝輔	学籍番号	1531102
論 文 題 目	ターン制戦略ゲーム AI の棋力向上		

要　旨

ターン制戦略ゲームは，探索空間が広いため探索の効率化が重要な問題の 1 つである．これは複数着手性という同一手番で複数の駒を動かせる性質が原因であり，また，囲碁将棋やリアルタイムストラテジーゲームほど盛んに学術的な研究が行われていなかったため，人間プレイヤに匹敵する強い AI プログラムが存在しなかった．ターン制戦略ゲームの具体的な研究対象として，ターン制戦略ゲームの研究用プラットフォームである TUBSTAP を用い，強い AI プログラムのアルゴリズムについて検討を行った．

本研究では，M-UCT と F-UCT という 2 つの手法を提案し，それらのアルゴリズムを実装した AI プログラムを作成し，対戦実験から性能評価を行った．

M-UCT は，ユニット行動木という 1 つの駒の行動ごとに分解されたゲーム木を構成し，UCT 探索により行動決定を行う手法である．ユニット行動木を用いることにより，駒を動かす順番を考慮した探索が可能となった．複数着手性のある問題に対して，着手の順番は重要な要素であるため，この特性が棋力の向上に大きな影響を与えた．既存の AI プログラムと対戦実験を行った結果，全ての対戦プログラムに勝ち越しその強さを確認した．また，Game AI Tournaments2016 で開催された第 2 回 TUBSTAP 大会で優勝した．

F-UCT は M-UCT にファジィ評価を加えて拡張した手法である．人間プレイヤが CPU プレイヤに対して遥かに強いという現状から，人間プレイヤの直感評価をファジィ集合でモデル化し，探索に用いることで効率化を図った．M-UCT との自己対戦の結果，ファジィ集合生成のための学習に使用したマップ，およびそのマップに特徴が類似したマップでは M-UCT に勝ち越し，ファジィ集合による人間の直感のモデル化とそれによる評価の有効性を示した．

電気通信大学情報理工学研究科
情報・通信工学専攻情報数理工学コース修士論文

ターン制戦略ゲーム AI の棋力向上

平成 29 年 1 月 30 日

情報数理工学コース

学籍番号 1531102

武藤 孝輔

指導教員 緒方 秀教
村松 正和
助言 西野 順二

目 次

第1章　序論

1.1　研究の目的と背景

ある問題に対して取り得る全ての解の空間を探索空間と呼び，探索空間が大きいほど扱いづらい問題であると見なせる．本研究は，探索空間が膨大な問題に対する 2 つの解法を提案し，ターン制戦略ゲームを対象に実験的にその効果を示すことを目的とする．

対戦型ゲームにおいて，プレイの上達やプレイを楽しいものに出来るかは，対戦相手に拠るところが大きい．対戦相手として人間プレイヤを常に求めることは難しいため，非人間プレイヤ，つまりコンピュータプレイヤの存在がどのゲームにも必要とされる．

今日，ゲーム AI の研究は大衆向けボードゲームの分野において盛んに行われており，多くのゲームプログラムが存在する．プロ棋士レベルの強さを持ったプログラムさえも存在し，チェスでは 1997 年に Deep Blue[1] が世界チャンピオンのガルリ・カスパロフに勝利，将棋では 2013 年に ponanza[2] が佐藤慎一四段に平手で勝利，そして囲碁でも 2016 年に AlphaGo[3] がプロ囲碁棋士のイ・セドルに互先で勝利した．このため，これらの分野のゲームにおいて，コンピュータプログラムは一般的な人間プレイヤの相手としてほぼ十分である．

一方で，ターン制戦略ゲーム [4] は，前述のボードゲームの要素を多く含んでおり，位置づけはそれらを拡張したゲームといえるが，研究が盛んに行われるようになってから間もなく，まだ十分な結果は得られていない．これは，ターン制戦略ゲームには同一ターンで複数のユニット (駒) を動かせる複数着手性という特徴があるため探索が困難であるということと，ベンチマークが存在しなかったことが理由として考えられる．ここで，ターンとは手番のことを，ユニットとは将棋やチェスでいう駒のことを指し，ターン制戦略ゲームではそれぞれターン，ユニットと呼ぶことが一般的であるため，以後ターン，ユニットで統一する．

実際にターン制戦略ゲームの AI は人間プレイヤよりも弱く，対等な条件でのゲームなら，プレイ経験の浅いプレイヤでもそのゲームのルールと基本的な定石を覚えることで AI に対して安定して勝つことができる．初心者でも分かる有効な手を，AI は選択できていないことが考えられる．このようなターン制戦略ゲームの AI を強くし，人間プレイヤの相手が務まる AI を実装することが求められている．

1.2　研究の概要

前項で述べた複数着手性のため，ターン制戦略ゲームにおいて探索の効率化は重要な課題の 1 つである．多数の合法手の中には人間プレイヤから見て明らかな悪手も多く，これにより探索性能を上げる余地がある．また，人間プレイヤを対戦相手として想定した場合，精神的負担を考慮して 1 ターンの思考時間を平均的な人間プレイヤが 1 ターンにかける時間と揃えるのが

望ましい．

本研究では，探索の効率化として，ターン単位での合法手，すなわち全てのユニットの行動の組み合わせ全体を探索するのではなく，1 つのユニットの行動ごとに分解しゲーム木を構成して探索する手法を提案する．こうして再構築したゲーム木をユニット行動木と，このユニット行動木の上で UCT 探索を行う手法を M-UCT と呼ぶことにする．

M-UCT を行なっても，なお探索木の分岐数はその他のゲームと比べて多いため，更なる効率化の方法として人間の知識を用いた行動決定を行うことを考える．人間は膨大な局面空間を直感的に評価して有効な着手を選択することができ，人の知見を探索および評価利用することは囲碁 [5] や将棋 [6] においても棋譜学習を通じて行われ有効性が示されている．

戦略ゲームでは囲碁や将棋と異なり，上級者の棋譜データがほとんど入手できない．このため人間の直感評価の少量データをモデリングする手法として，多次元空間でのファジィ集合 [7][8] を用いる．探索中の着手について，曖昧な「良い手」のファジィ集合に含まれる度合いを，UCT 探索の優先度に使用する手法を F-UCT と呼ぶことにする．

本研究ではターン制戦略ゲームの具体的な対象として TUBSTAP[9] を用いる．TUBSTAP のルール，特徴や既存手法について第 2 章で解説する．本研究の提案手法である M-UCT の解説，実験結果を第 3 章に，M-UCT の改良版である F-UCT を第 4 章に記載する．最後に，結論と今後の展望について第 5 章に記載する．

第2章　ターン制戦略ゲーム

2.1　ターン制戦略ゲームの特徴

ターン制戦略ゲームは，大戦略やファミコンウォーズなど，コンピュータゲーム市場で人気を博しているゲームジャンルの 1 つであり，ある特定のゲームを指す言葉ではない．個々のゲームによって正確なルールは異なるが，大半のターン制戦略ゲームに共通し，ターン制戦略を特徴付ける最も大きな要素は複数着手性である．複数着手性とは同一ターンで複数のユニットをプレイヤが動かせる性質のことであり，これにより探索空間が膨大になる．1 ターンの合法手数の平均が，チェスは約 40 手，将棋は約 80 手程度であるのに対し，ターン制戦略ゲームでは，例えば移動先が 10 通りあるユニットが 3 つだけある局面でも，$10^3 = 1000$ 通りの手が存在し，合法手の多いゲームとして扱われる囲碁よりも更に多い．

前章でターン制戦略ゲームは将棋や囲碁の拡張ゲームの位置づけにあるとしたが，ターン制戦略ゲームの拡張ゲームとしてリアルタイム戦略ゲーム (RTS) がある．RTS は，非ターン制であるため，各プレイヤ交互に手を指すのではなく，任意のタイミングで各々が手を指せる．RTS は研究が盛んであり，AI プログラムのコンテストも頻繁に行われている．

ターン制戦略ゲームは，囲碁将棋と RTS の中間的な存在であったため，これまで研究対象として注目を集めることが少なかった．探索が困難であり，研究も盛んに行われていないという理由から，人間プレイヤに匹敵する優れた CPU プレイヤが存在しない．このような背景から建ち上げられたターン制戦略ゲームの研究プラットフォームが TUBSTAP である．TUBSTAP について，次節より解説する．

2.2　研究プラットフォーム TUBSTAP

TUBSTAP は北陸先端科学技術大学院大学 (JAIST) がターン制戦略ゲームの思考アルゴリズムの研究・開発の為に立ち上げたプロジェクト，およびそのゲームのことを指し，図 2.1 に示すように 2 人のプレイヤがそれぞれの軍を持ち，ユニットを動かして倒し合いをするゲームである．

2013 年にプロジェクトが開始され，2016 年までに 3 回大会が開催された．第 1 回はゲームプログラミングワークショップ 2015 で，第 2 回は Game AI Tournaments2016 で，第 3 回はゲームプログラミングワークショップ 2016 で開催され，参加者は自作の AI プログラムを持ち込み，プログラム同士の対戦からその棋力を競い合った．

ユーザはルールや GUI は与えられたものを用いて思考ルーチンの開発のみに専念でき，ユーザ同士統一の環境で比較が行える．ゲームのルールは簡略化されており，多くの戦略ゲームに共通する基本的な要素だけを持っている．本研究ではこのゲームにおける AI の棋力向上を狙う．

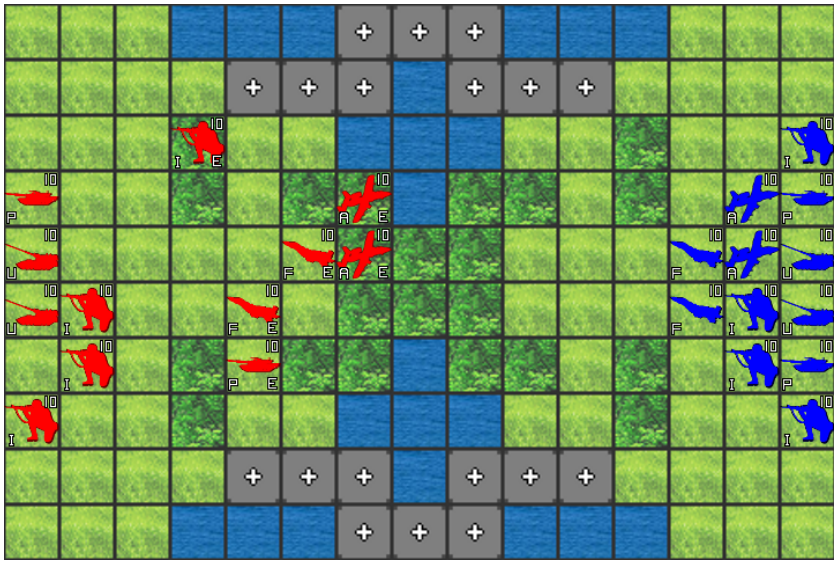

図 2.1: TUBSTAP のゲーム画面

2.2.1 TUBSTAP のルール

TUBSTAP は，上下左右 4 方に隣接したマス目からなる盤面に，複数のユニットを配置し，各プレイヤのターンでユニットを行動させ，ゲームを進める．

ユニット

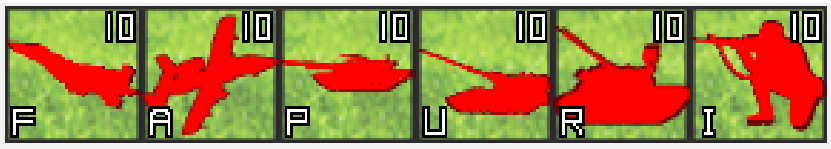

図 2.2: ユニット一覧

ユニットには HP，攻撃力，移動力，射程の 4 つのパラメータが与えられている．この内，攻撃力，移動力，射程はユニットの種類毎に固定の値であり，HP は対戦開始時にそれぞれのユニットに 1〜10 の値が設定される．HP はユニットの耐久力を表し，後述の攻撃行動により減少し，0 になるとそのユニットは盤面から取り除かれる．攻撃力は相手ユニットを攻撃した時のダメージ計算時に関わり，ユニット間の相性を表す．移動力は 1 ターン内に移動できる基本的な距離である．ここで，基本的な距離としているのは，実際に移動できる距離は後述の地形効果と相まって決まるからである．射程は攻撃可能な相手ユニットとの距離である．

ユニットは図 2.2 に示す 6 種類がある．

- 戦闘機 (F):対空航空ユニット．全ユニットで移動力が最も高い．
- 攻撃機 (A):対地航空ユニット．対空戦車を除く地上ユニットには反撃を受けずに攻撃可能．
- 戦車 (P):対地地上ユニット．地上ユニット全てに対して有利を取れる．
- 自走砲 (U):対地地上ユニット．隣接する上下左右 1 マスに攻撃できず，移動後攻撃が不可能．そのかわりにマンハッタン距離 2〜3 のマスへの遠距離攻撃が可能．
- 対空戦車 (R):地上ユニット．全ユニットに攻撃可能．
- 歩兵 (I):対地地上ユニット．攻撃力が低く弱い．

  各ユニットのパラメータを表 2.1 に示す．射程は，相手ユニットを攻撃対象に選択できるマンハッタン距離，攻撃力は，攻撃を行う際に使用し攻撃対象の種類によって変化するパラメータ，移動力は，移動を行う際に使用するパラメータである．これらのパラメータは，後述のダメージ計算 (式 (2.1)) に用いられる．

表 2.1: ユニットのパラメータ

ユニット	射程	攻撃力						移動力
		F	A	P	U	R	I	
戦闘機 (F)	1	55	65	0	0	0	0	9
攻撃機 (A)	1	0	0	105	105	85	115	7
戦車 (P)	1	0	0	55	70	75	75	6
自走砲 (U)	2-3	0	0	60	75	65	90	5
対空戦車 (R)	1	70	70	15	50	45	115	6
歩兵 (I)	1	0	0	5	10	3	55	3

盤面と地形

ターン制戦略ゲームは将棋やチェスのような決められた初期盤面が存在しない．盤面の大きさ，初期ユニットの種類・数が異なる様々な初期盤面を定義できる．

TUBSTAP には標準問題として 5 つの盤面 (マップ) が用意されている．最新版の TUBSTAP には，直近の大会で使用されたマップも同梱されている．これらをベンチマークとして用い，対戦実験から AI の性能評価を行うことができる．

マップの各マスには地形が割り当てられ，地形には移動コストと地形効果というパラメータが与えられている．移動コストは，ユニットがそのマス上を移動する際にそのユニットが持つ移動力が消費される値であり，地形効果は，地上ユニット (P，U，R，I) がそのマス上にいる時に相手のユニットから攻撃を受けた時のダメージ計算時に影響する値である．

地形は図 2.3 に示す 7 種類がある．

図 2.3: 地形一覧 (左から草原，森，山，海，道路，陣地，進入禁止)

- 草原:移動に支障無し．防御に適さない．
- 森:車両ユニット (P，U，R) は移動に支障有り．防御に比較的適している．
- 山:地上ユニットは歩兵のみ進入可能．防御に適している．
- 海:航空ユニットのみ進入可能．
- 道路:移動に支障無し．防御に全く適さない．
- 陣地:移動に支障無し．防御に適している．
- 進入禁止:全てのユニットが移動できない．

各地形のパラメータを表 2.2 に示す．移動コストは，ユニットがその地形上を移動する際に使用するパラメータ，地形効果は，ユニットがその地形上で戦闘を行う際に使用するパラメータである．

表 2.2: 地形のパラメータ

地形	移動コスト						地形効果
	F	A	P	U	R	I	
草原	1	1	1	1	1	1	1
森	1	1	2	2	2	1	3
山	1	1	-	-	-	2	4
海	1	1	-	-	-	-	0
道路	1	1	1	1	1	1	0
陣地	1	1	1	1	1	1	4
進入禁止	-	-	-	-	-	-	-

2.2.2 行動 (移動と攻撃)

行動とは，移動と攻撃の組み合わせであり，移動のみ，攻撃のみ，移動後に移動先で攻撃，何もしない，の 4 種がある．ただし，厳密には何もしないは，現在いるマスに移動する移動のみの行動として扱う．プレイヤは，各ユニット毎に 4 種の行動のうち 1 つを選択することができる．複数着手性のため，プレイヤは自分のターンで，全ユニットについて行動を決めることができる．同じターンで，各ユニットは 1 度だけ行動できる．行動を行うユニットの順番はプレイヤの任意でありターン毎に異なっても構わない．

例えば，航空ユニット (F，A) は移動力の分だけ移動することが可能である．戦闘機 (F) は移動力 9 であるため，マンハッタン距離が 9 離れたマスまで移動可能である．地上ユニット (P，U，R，I) は移動力と移動先の地形の移動コストに応じて移動できる範囲が変わる．戦車 (P) は移動力 6 であるが，移動中に森のマスを通る場合，戦車の移動コストは 2 であるため，移動できる距離が 1 短くなる．また，移動する際に自ユニットがいるマスは通り抜け可能であるが，最終的な移動先としては選べない．相手ユニットがいるマスは通り抜けることも不可能である．

攻撃は，上下左右 1 マスへの隣接攻撃が基本となるが，表 2.1 に示したように自走砲のみマンハッタン距離 2〜3 のマスへの遠距離攻撃が可能である．そのかわり，自走砲は移動後攻撃が不可能である．隣接攻撃は，被攻撃ユニットが戦闘後も生存した場合，攻撃ユニットに対して反撃を行う．

2.2.3 ダメージ計算

ダメージとは，攻撃行動の際に発生する値である．ユニットはダメージを受けると，その値だけ HP が減少する．

ユニット u_1 が u_2 に攻撃した時与えるダメージ $d(u_1, u_2)$ は，式 (2.1) で定まり，式 (2.2) で u_2 の HP が減少する．

$$d(u_1, u_2) = \frac{p_{u_1} h_{u_1} + 70}{100 + q h_{u_2}} \tag{2.1}$$

$$\mathrm{HP}_{\mathrm{u_2}} \leftarrow \mathrm{HP}_{\mathrm{u_2}} - d(u_1, u_2) \tag{2.2}$$

ただし，

$$\begin{aligned} p_u &: \text{ユニット } u \text{ の攻撃力} \\ h_u &: \text{ユニット } u \text{ の HP} \\ q &: \text{地形効果} \\ \mathrm{HP}_u &: \text{ユニット } u \text{ の HP} \end{aligned}$$

である．攻撃側の HP が高いほど攻撃対象に与えるダメージも高くなるが，被攻撃側も HP が高いほど受けるダメージを軽減される．また，ランダム要素も含まず確定値である．

2.2.4 勝敗決定

相手のユニットを全て倒すと勝利となる．しかし，一方のプレイヤの生存ユニットが戦闘機 (対空ユニット) だけ，もう一方のプレイヤの生存ユニットが戦車 (対地ユニット) だけといった局面になったとき，互いに攻撃不可能となり決着がつかなくなるため，予め決められたターン数を超えても勝敗がついていない場合，各プレイヤの生存ユニットの HP の総和の差によって勝敗の決定，または引き分けとなる．

勝敗決定の式を (2.3) に示す．

$$\begin{cases} win & \text{if } |U'| == 0 \\ lose & \text{if } |U| == 0 \\ judge & \begin{cases} win & \text{if } \sum_{u \in U} \mathrm{HP}_u - \sum_{u \in U'} \mathrm{HP}_u \geq T \\ lose & \text{if } \sum_{u \in U'} \mathrm{HP}_u - \sum_{u \in U} \mathrm{HP}_u \geq T \\ draw & \text{otherwise} \end{cases} \end{cases} \tag{2.3}$$

ただし，

$$\begin{aligned} U &: \text{自分生存ユニットの集合} \\ U' &: \text{相手生存ユニットの集合} \\ \mathrm{HP}_u &: \text{ユニット } u \text{ の HP} \\ T &: \text{勝敗決定閾値} \end{aligned}$$

である．

2.3 TUBSTAP の性質

以下では，複数着手性により囲碁将棋に比べて非常に探索空間が大きくなることを，例を示して述べる．

図 2.4 にマップの例を示す．これは TUBSTAP に同梱されたうち最も小さいマップである．6×6 の盤面に各チーム 6 ユニットずつ合計 12 ユニットで戦う．

先手は赤チームで，左上の歩兵から行動することを考えた時，行動パターンの数はその場に留まるか移動先候補 4 マスのどこかに移動する 5 である．この時の行動によってその後のユニット達の行動パターン数は変わりうるが，今回はその場に留まる行動を選択したことにする．次に，右の自走砲の行動パターン数は，12 である．こちらも同様にその場に留まる行動を選択したことにする．残り 4 つのユニットも全てその場に留まる行動を順に選択していくと，それぞれ行動パターン数は，7，18，9，21 である．そして，合法手はそれぞれのユニットの行動の組み合わせとなるので，合法手の数は

$$5 \times 12 \times 7 \times 18 \times 9 \times 21 = 1,428,840$$

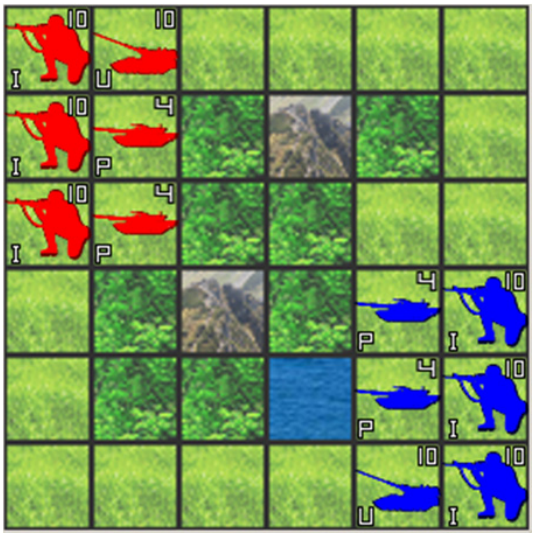

図 2.4: TUBSTAP 付属のマップ (map01)

となる．前述したように，あるユニットの行動によって他のユニットの行動パターン数は増減するため，この合法手数は固定の値ではないが，1 番小さいマップでも最初の合法手数は 100 万以上になることが分かる．同梱の 1 番大きいマップは，最初のターンは戦闘機と攻撃機が行動できない設定になっているため最初のターンの合法手数は約 300,000 であるが，次のターンには 300,000,000 以上になる．

複数着手性は，原理的にユニットの数の冪乗で合法手が増加していく．そして，膨大な数の合法手の中には，明らかな悪手も数多く含まれている．このため，探索に工夫を加える必要がある．

2.4 既存手法

2.4.1 行動評価関数

行動評価関数とは，攻撃ダメージの評価値であり，行動の指針とすることが多い．後述の最良単独行動アルゴリズム，行動ペアの合計評価アルゴリズムで用いる関数である．攻撃行動にのみ適用され，ユニット u_1 がユニット u_2 を攻撃する時，行動評価関数 $f(u_1, u_2)$ は式 (2.4) で定まる．

$$f(u_1, u_2) = d(u_1, u_2)m_{u_2} - r(u_1, u_2)m_{u_1} \tag{2.4}$$

ただし，

$$\begin{aligned} d(u_1, u_2) &: u_1\text{が } u_2\text{に与えるダメージ} \\ r(u_1, u_2) &: \text{その時に受ける反撃ダメージ} \\ m_u &: \text{ユニット } u \text{ の価値} \end{aligned}$$

である．式 (2.4) によって返される評価値が高いほど良い攻撃行動であるとみなす．

また，m_{u_1}，m_{u_2} は式 (2.5)，(2.6) で定まる．

$$m_{u_1} = 10 + h_{u_1} + w_{u_1} \tag{2.5}$$

$$m_{u_2} = 10 + \max_{u \in U'} d(u_2, u) + w_{u_2} \tag{2.6}$$

ただし，

h_u ： ユニット u の残り HP
w_u ： ユニット u の種類によって定まる価値
U' ： 既にその手番内で行動した u_1 チームのユニットの集合

である．攻撃ユニットは残り HP が多いほど価値が高くなり，被攻撃ユニットは既に行動済の移動できない攻撃側ユニットに対して大きなダメージを与えられる脅威度の高いものほど価値が高くなる．例えばユニットの種類によって定まる価値は，最良単独行動アルゴリズムの場合は全てのユニットが 0 である．

2.4.2 最良単独行動

最良単独行動とは，TUBSTAP の標準搭載 AI で用いられている行動評価関数が最大となる行動を選択するアルゴリズムである．

以下の手順により行動決定を行う．

1. 相手ユニットに攻撃行動を行えるユニットがある時，行動評価関数が最大となる最善評価値の行動を選択して実行する．
2. 攻撃行動を行えるユニットがなければ，ランダムにユニットを 1 つ選び移動ルーチンに従う．

最良単独行動のアルゴリズムを Algorithm1 に示す．

Algorithm 1 最良単独行動

Require: 現在の局面 s
Ensure: 行動 a^*
  U':攻撃行動を行える状態にあるユニットの集合
  if $|U'| > 0$ **then**
    $A'(s, U')$:局面 s における全攻撃行動の集合
    $a^* \leftarrow \mathrm{argmax}_{a \in A'(s)} f(s, a)$
  else
    移動ルーチンに従う
  end if
  return a^*

移動ルーチンは，攻撃対象に取れるユニットが相手チームに存在する場合，行動評価関数から最も価値の高い攻撃になる相手ユニットに向かって移動する．攻撃対象に取れるユニットが相手チームに存在しない場合，地形効果が高い地形に留まるように移動する．

2.4.3 行動ペアの合計評価

行動ペアの合計評価による行動決定は，村山らによって提案されたアルゴリズムである [4]．ユニットを動かす順番の重要性に着目し，最良単独行動を基に提案された．行動ペアとは，1 番目と 2 番目に選ぶ攻撃行動の組み合わせのことである．最良単独行動に比べ，ターン全体で見た時のそれぞれの行動の評価値の合計が高くなりうる．これは，最良単独行動ではユニットを動かす順番を考慮した行動決定ができないからである．以下の手順により行動決定を行う．

1. 攻撃行動を行えるユニットが 1 以下の場合は最良単独行動に従う．
2. 1 番目に攻撃行動を行うユニット u とその行動 $a' \in A(u)$ を選び，行動評価関数 $f(a')$ より評価する．
3. 行動 a' の実行後の局面を s'，s' における攻撃行動の集合を $A(s')$ として，s' における評価値が最大の行動 $\max_{a \in A(s')} f(a)$ を求め，2 の評価値と合算したものを $v(u, a')$ とする．
4. $v(u, a')$ が最大となる u を選び，行動 a' を選択する．

行動ペアの合計評価のアルゴリズムを Algorithm2 に示す．

Algorithm 2 行動ペアの合計評価

Require: 現在の局面 s
Ensure: 行動 a^*

```
U':攻撃行動を行える状態にあるユニットの集合
if |U'| ≤ 1 then
  return maxAct(s) // maxAct 関数は最良単独行動
else
  for Each u ∈ U' do
    for Each a' ∈ A(u) do
      s':a' 実行後の局面
      v(u, a') ← f(a') + max_{a∈A(s')} f(a)
      if maxv < v(u, a') then
        maxv ← v(u, a')
        a* ← a'
      end if
    end for
  end for
end if
return a*
```

2.4.4 深さ限定モンテカルロ法

深さ限定モンテカルロ方は藤木らによって提案されたアルゴリズムである [10][11]．原始モンテカルロ法では終局までランダムシミュレーションを行うのに対し，深さ限定モンテカルロ法は限られた深さ (ターン数) までランダムシミュレーションを行う．そのためランダムシミュレーションが返す報酬は勝敗ではなくランダムシミュレーションを止めた時点での局面の評価値となる．終局までランダムシミュレーションを行わない分計算コストが少なくなるため，時間あたりのシミュレーション数を増やすことが出来る．なお，シミュレーションを行い評価値を計算する一連の流れをプレイアウトと言う．この手法におけるゲーム木では，枝 1 つが行動の組み合わせにあたり，深さ 1 が 1 ターン先を意味する．

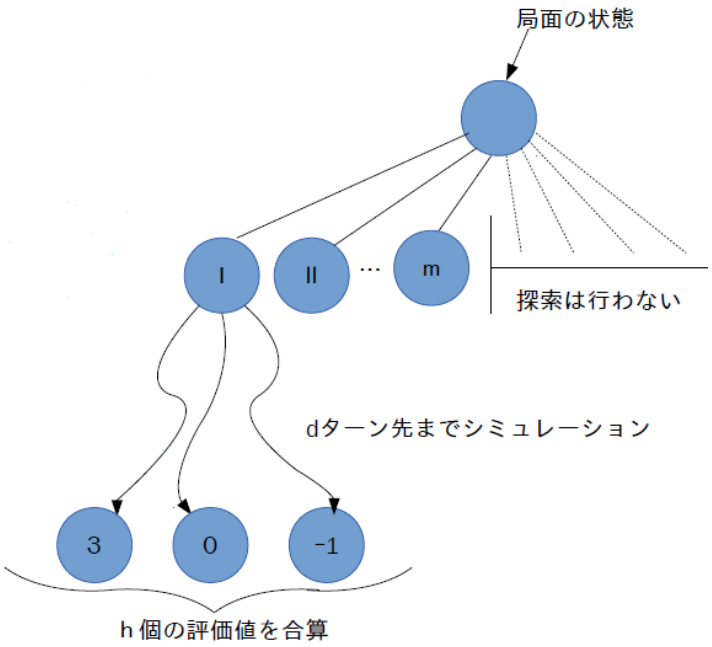

図 2.5: 深さ限定モンテカルロ法の概念図 ([10] の図より)

深さ限定モンテカルロ法のアルゴリズムを Algorithm3 に示す.

Algorithm 3 深さ限定モンテカルロ法

Require: 現在の局面 s
Ensure: 行動の組み合わせ $\mathbf{c}^*$
  $C = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_m\}$: 行動の組み合わせをランダムに m 個サンプル
  for $i = 1$ to m **do**
    s':$\mathbf{c}_i$ 実行後の局面
    for $j = 1$ to h **do**
      sum $\leftarrow$ sum + eval(sim(s', d))
    end for
    if maxsum $<$ sum **then**
      maxsum $\leftarrow$ sum
      $\mathbf{c}^* \leftarrow \mathbf{c}'$
    end if
  end for

sim は局面 s' から d ターン先までランダムシミュレーションを行い，シミュレーション後の局面を返す関数であり，eval は局面を引数に取る，後述の状態評価関数である．

深さ限定モンテカルロ法の状態評価関数

深さ限定モンテカルロ法でプレイアウト後の局面に使用する評価関数．ユニット u の HP を l_u とすると，そのユニットの評価値 $v(u)$ は式 () で定まる．

$$v(u) = \begin{cases} l_u \times 0.2 & \text{if ユニット } u \text{ は歩兵 (I)} \\ l_u & \text{else} \end{cases} \tag{2.7}$$

自分チームのユニットの集合を M，相手チームのユニットの集合を E とすると，状態評価関数 $h(s)$ は式 (2.8) で定まる．

$$h(s) = (\sum_{u \in M} v(u) - \sum_{u \in E} v(u)) \times B \tag{2.8}$$

ここで，B は式 (2.9) で定まる．

$$B = \begin{cases} 2 & \text{if } \sum_{u \in M} v(u) = 0 \text{ or } \sum_{u \in E} v(u) = 0 \\ 1 & \text{else} \end{cases} \tag{2.9}$$

評価値は，自分チームのユニットの HP の総和が大きいほど高くなり，相手チームのユニットの HP の総和が大きいほど低くなる．ただし，歩兵の HP のみ補正がかかっているのは，歩兵が他のユニットに比べて戦力として期待できないからである．また，ボーナス係数 B より勝敗が決した状態は評価値が 2 倍される．

既存手法の相互の強さとしては，最良単独行動は，TUBSTAP の標準搭載 AI でもあり，上記の中で最も棋力が低い．行動ペアの合計評価は，対戦マップに依るところもあるが，最良単

独行動よりも強くなる．深さ限定モンテカルロ法は，これら 2 つの手法よりも有意に強いということが分かっている．

第3章　M-UCT

提案手法 M-UCT[12][13][14] のアルゴリズムおよびその AI プログラムについて述べる．M-UCT の M は Multi unit を意味し，UCT は次節で解説する UCT 探索のことである．Game AI Tournaments2016 で開催された第 2 回の大会で優勝したため，2016 年 1 月現在で最新版の TUBSTAP(ver1.08) に M-UCT の AI プログラムが同梱されている．

UCT 探索は，囲碁 AI の分野で特に効果を出している探索手法であるが，これは囲碁が局面の評価が難しく，合法手が多いからである．ターン制戦略ゲームにもこの特徴は当てはまるため，棋力向上が見込める．また探索木の構造を，1 行動ごとに分解されたユニット行動木にすることで，分岐因子を削減する．

TUBSTAP の AI プログラムは，局面の情報を受け取り，1 つの行動を返す関数から成り立つ．M-UCT は，受け取った局面情報を含むデータ構造を定義し，それを基にユニット行動木を構成し UCT 探索を行って，最終的に 1 つの行動を決定する．

3.1　UCT 探索

UCT(Upper Confidence bound applied to Trees) 探索とは，モンテカルロ木探索の 1 種である [15]．モンテカルロ木探索は，前章で解説した原始モンテカルロ法を木探索に拡張したものである．つまり，あるノードのプレイアウト数が閾値を超えたらそこから子ノードを 1 段展開し，プレイアウトを重ねるほど探索木が深くなっていき先の手を考慮した探索が行える．UCT 探索は，探索する子ノード決定の指標に UCB 値を用い，UCB 値は式 (3.1) で与えられる．

$$\mathrm{UCB} = \overline{X_i} + c\sqrt{\frac{\log n}{n_i}} \tag{3.1}$$

$\overline{X_i}$ ： 選択肢 i の報酬の期待値

n ： それまでの全プレイアウト数の合計値

n_i ： 選択肢 i を選びプレイアウトを行なった回数

c ： UCB 値の性質を決める定数

である．UCB 値は，報酬の期待値，つまりゲームにおいてはその手を選んだ結果の勝率が高ければ単純に高くなり，またその手を選びプレイアウトを行なった回数が少ない場合でも高くなる．したがって，UCB 値は単純に勝率の高い手の他に，まだプレイアウトを十分に行なっていない調べる価値がある手に対しても高くなる．そのため，有望な手の探索が効率的に行えるこ

とが考えられる．c の値を大きくすることで，プレイアウト数の少ない手の探索を優先するように，小さくすることで，グリーディに勝率の高い手の探索を優先するようになる．

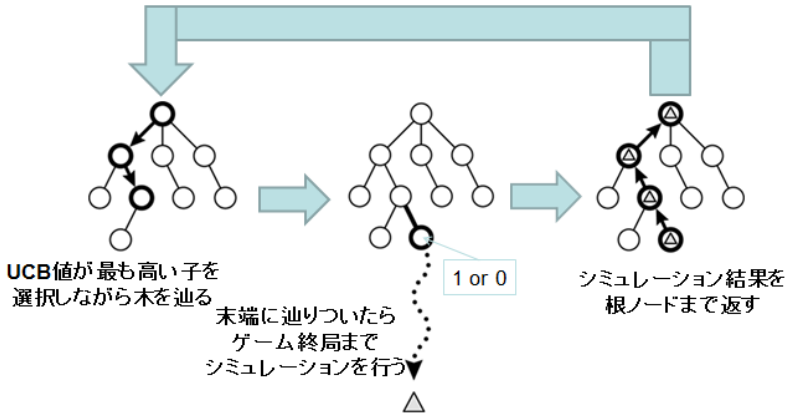

図 3.1: UCT 探索の流れ

3.2 ユニット行動木のデータ構造

ユニット行動木のノードのデータ構造は以下のようになっている．

```
class Game_Tree
{
    public Map board; // 局面情報
    public int depth; // そのノードの深さ
    public int simnum; // そのノードを訪問した回数
    public double lastscore; // 前回のシミュレーション結果
    public double totalscore; // これまでのシミュレーション結果の総和
    public double housyuu; // 勝率
    public Action act; // 直前に取った行動
    public List<Game_Tree> next; // 子ノードのリスト
}
```

3.3 M-UCT のアルゴリズム

複数着手性のあるゲームの探索木は，ユニット 1 つの行動を 1 つの枝に割り当てる縦長の木と，全てのユニットの行動の組み合わせを 1 つの枝に割り当てる横長の木が考えられる [16]．ユニット行動木は前者に該当する．

それぞれの探索木の利点は，まず 1 つの行動を 1 つの枝に割り当てる縦長の探索木は，分岐因子が比較的少ないため，あるプレイアウト数に対する探索木の平均深さが深くなる．このため，より先の手の探索が見込める．横長の探索木は，探索木で用いる局面の状態が，ターンプレイヤが全ての行動を終えた状態であるので，局面の状況をより詳細に表せている．

M-UCT の探索の流れは，前述の UCT 探索準拠であり，UCB 値が最大の子ノードを辿っていき，到達した葉ノードの訪問回数が閾値以上ならばそこから子ノードを展開し，閾値未満ならゲーム終了までランダムシミュレーションを行い，その結果を辿ってきたノードに反映する．

また，この探索の流れを具体例を用いて図 3.2〜3.7 に示す．

まず，未行動のユニットが 3 つあり，それぞれ 2 通りの行動ができる局面の状態が与えられたとする (図 3.2)．

全行動に対し UCB 値を計算した結果，ユニット u1 の第 2 行動が最も高く，その枝を辿る．到達したノードは末端であり，そのノードを訪問した回数は閾値以下であったため，ゲーム終了までランダムシミュレーションを行う (図 3.3)．

シミュレーション結果が得られたら，辿ってきたノードにその結果を反映する (図 3.4)．

この流れを繰り返し，ある時 u1 の第 2 行動先のノードを訪問したら，訪問回数が閾値を超えていたため，そこから子ノードの展開を行う．そのノードの局面の状態は u1 が行動を終了した後の状態であるため，まだ未行動の u2 と u3 の 2 つの行動で子ノードの展開を行う (図 3.5)．

その後もプレイアウトを繰り返していき，子ノードは未行動のユニットの行動で展開する (図 3.6)．

自ユニットに未行動のユニットが存在しなくなった場合，相手のユニットの行動で子ノードを展開する．相手ユニットが 2 つ存在し，そのノードの状態から，それぞれ 2 通りの行動ができる場合，それら 4 つの行動で展開される (図 3.7)．

この例の場合，深さ 4 まで探索木を展開して，初めて相手の手の探索にたどり着く．

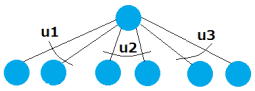

図 3.2: M-UCT の探索の様子 1

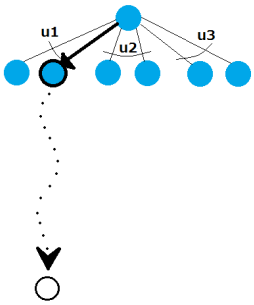

図 3.3: M-UCT の探索の様子 2

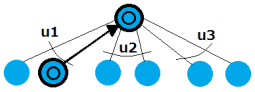

図 3.4: M-UCT の探索の様子 3

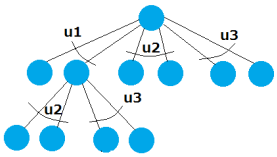

図 3.5: M-UCT の探索の様子 4

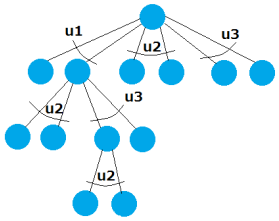

図 3.6: M-UCT の探索の様子 5

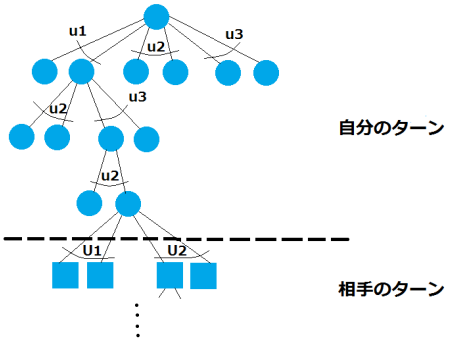

図 3.7: M-UCT の探索の様子 6

M-UCT のアルゴリズムを Algorithm4 に示す.

Algorithm 4 M-UCT

Require: ノード n
  $n' \leftarrow$maxUCB(n) // maxUCB(n) は n の子ノードから UCB 最大のものを選択する関数
  if n' の訪問回数が閾値超過 **then**
    if n' が葉ノード **then**
      dev(n')　// dev(n') は n' から子ノードを展開する関数
    end if
    M-UCT(n') // 再帰
    update() // update() はシミュレーション結果をノードに反映させる関数
  else
    randSim(n) // randSim(n) は n からゲーム終了までランダムシミュレーションを行う関数
    update()
  end if

3.4　対戦実験によるアルゴリズムの評価

M-UCT アルゴリズムの性能を AI 同士の対戦実験によって評価した．TUBSTAP には標準でオートバトル機能が実装されており，この機能を利用して対戦実験を行なった．オートバトル機能は，ユーザが対戦回数と対戦プログラムを指定することで，自動で対戦実験を行う機能である．

実験で使用したマップは，標準で用意されているものから選んだ.

3.4.1　ベースライン対戦比較実験

TUBSTAP 標準搭載 AI が用いる最良単独行動をベースラインアルゴリズムとして対戦させ，行動ペアの合計評価，深さ限定モンテカルロ法，M-UCT の性能評価を行った.

実験設定

図 3.8，3.9，3.10 に示す 3 つのマップで実験を行なった．map01 は 2 章でも示した同梱マップの中で最も小さく，最もスタンダードなマップである．map03 は遠距離攻撃を行える 2 両の自走砲の使い道が重要となるマップである．map05 は全兵科が揃った，取れる戦略の幅が広いマップである.

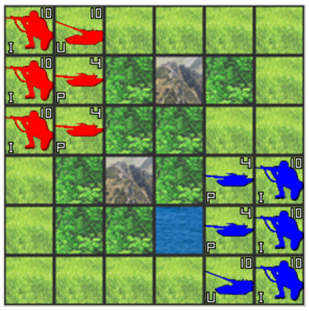

図 3.8: map01

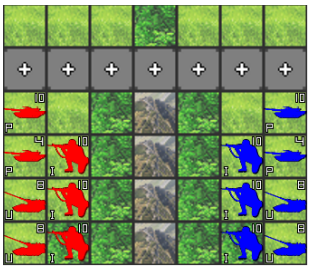

図 3.9: map03

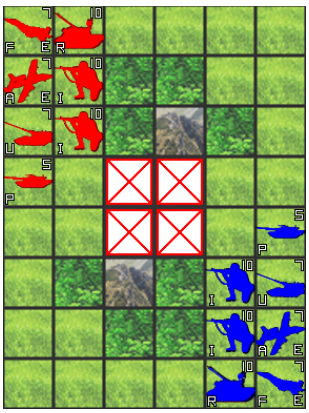

図 3.10: map05

比較したアルゴリズムと，その設定を以下に示す.

- 行動ペアの合計評価
  - 対戦回数:各マップ 1000 回
  - ターン上限数:全マップとも 20，上限数を超えた時の残りユニットの合計 HP が 10 以上なら合計 HP が多いチームの勝利とし，それ以外は引き分け
  - 行動評価関数で用いるユニットの価値: 戦闘機:2，攻撃機:5，戦車:3，自走砲:3，対空戦車:3，歩兵:0
- 深さ限定モンテカルロ法
  - 対戦回数:各マップ 100 回
  - ターン上限数:map01 は 30，map03，map05 は 20，上限数を超えた時の残りユニットの合計 HP が 10 以上なら合計 HP が多いチームの勝利とし，それ以外は引き分け
  - 行動の組み合わせのサンプル数:100
  - ランダムシミュレーションの深さ: 6(相手の手，自分の手を各 3，合計 6 ターン先まで)
  - 各サンプルのシミュレーション回数:80
- M-UCT
  - 対戦回数:各マップ 100 回
  - ターン上限数:map01 は 30，map03，map05 は 20，上限数を超えた時の残りユニットの合計 HP が 10 以上なら合計 HP が多いチームの勝利とし，それ以外は引き分け
  - 行動の組み合わせのサンプル数:100
  - 1 行動決定のプレイアウト数:2000
  - 各ノードの展開閾値:20
  - UCB 値の式の定数 c の値:0.15

実験結果

各実験結果を表 3.1，3.2，3.3 に示す．勝率の計算は引き分けを 0.5 として計算した．

表 3.1: 行動ペアの合計評価 vs 最良単独行動 (各 1000 戦)

	勝	負	引分	勝率
map01	509	417	74	0.546
map03	567	387	46	0.614
map05	577	360	63	0.609

表 3.2: 深さ限定モンテカルロ法 vs 最良単独行動 (各 100 戦)

	勝	負	引分	勝率
map01	71	23	6	0.74
map03	56	15	29	0.61
map05	46	20	34	0.63

表 3.3: M-UCT vs 最良単独行動 (各 100 戦)

	勝	負	引分	勝率
map01	76	24	0	0.76
map03	79	9	12	0.85
map05	87	5	8	0.91

考察

比較した 3 種のアルゴリズムの全てが最良単独行動に勝ち越し，標準搭載 AI よりも強くなった．M-UCT はその中でも最大 91%の勝率で勝ち越しており，特に高い成績を残した．M-UCT は map01 から map05 にかけてマップが広くなるほど，勝率が 0.76，0.85，0.91 と増加しており，マップサイズやユニット種類の増加により戦略の幅が広がるほど強くなることが示された．

3.4.2 深さ限定モンテカルロ法対戦

深さ限定モンテカルロ法に対し直接対戦を行い，M-UCT の強さを調べた．

実験設定

図 3.8，3.9，3.10 に示す map01，map03，map05 で実験を行なった．
対戦，各アルゴリズムの設定を以下に示す．

- 対戦設定
  - 対戦回数:各マップ 100 回
  - ターン上限数:map01 は 30，map03，map05 は 20，上限数を超えた時の残りユニットの合計 HP が 10 以上なら合計 HP が多いチームの勝利とし，それ以外は引き分け
- M-UCT
  - 1 行動決定のプレイアウト数:2000
  - 各ノードの展開閾値:20
  - UCB 値の式の定数 c の値:0.15
- 深さ限定モンテカルロ法
  - 行動の組み合わせのサンプル数:100
  - ランダムシミュレーションの深さ: 6(相手の手，自分の手を各 3，合計 6 ターン先まで)
  - 各サンプルのシミュレーション回数:80

実験結果

実験結果を表 3.4 に示す．勝率の計算は引き分けを 0.5 として計算した．

表 3.4: M-UCT vs 深さ限定モンテカルロ法 (各 100 戦)

	勝	負	引分	勝率
map01	51	40	9	0.56
map03	58	13	29	0.73
map05	73	8	19	0.83

考察

全てのマップにおいて M-UCT が勝ち越しており，また，対戦実験 1 に続きこちらでもマップ 1 からマップ 5 にかけて勝率が増加した．戦略の幅が広がるということは，合法手の数も増加するということであり，探索木の分岐因子も増加するため，時間あたりに探索できる深さは浅くなる．それにも関わらず勝率がすることから，プレイアウトの結果を保証するのに求められる探索木の深さは浅く，広く浅く探索することが効果的であった．

3.4.3 ランダム M-UCT 対戦

M-UCT の棋力が何に依るかを調べる実験を行なった．ユニット行動木は，分岐因子の数が少ないということが利点として挙げていたが，この他にユニットを動かす順番を考慮した探索ができることも利点であると考えられる．そこで，通常の M-UCT とユニットを動かす順番をランダムに決定する M-UCT(以降ランダム M-UCT と呼ぶ) を対戦させた．

通常の M-UCT とランダム M-UCT のユニット行動木の例を，図 3.11，3.12 に示す．ユニットが 3 つあり，それぞれ 2 通りの行動が取れる場合，通常の M-UCT のユニット行動木は 6 つ全ての行動を展開するが，ランダム M-UCT のユニット行動木はユニットを 1 つランダムに選択し，そのユニットの行動だけを展開する．

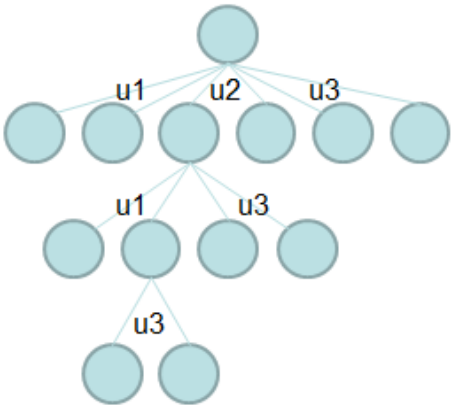

図 3.11: 通常の M-UCT のユニット行動木

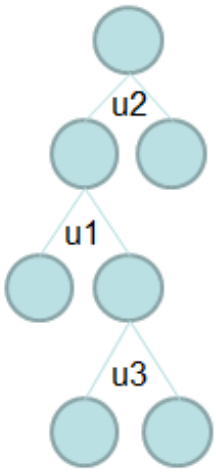

図 3.12: ランダム M-UCT のユニット行動木

実験設定

図 3.8 の map01 で実験を行なった.
対戦，アルゴリズムの設定を以下に示す.

- 対戦設定
  - 対戦回数:1000 回
  - ターン上限数:30，上限数を超えた時の残りユニットの合計 HP が 10 以上なら合計 HP が多いチームの勝利とし，それ以外は引き分け
- アルゴリズムの設定
  - 1 行動決定のプレイアウト数:2000
  - 各ノードの展開閾値:10
  - UCB 値の式の定数 c の値:0.15

実験結果

実験結果を表 3.5 に示す．勝率の計算は引き分けを 0.5 として計算した.

表 3.5: 通常 M-UCT vs ランダム M-UCT(1000 戦)

勝	負	引分	勝率
843	127	30	0.858

考察

通常の M-UCT がランダム M-UCT に大差をつけて勝ち越した．図 3.11，3.12 に示すように，両アルゴリズムは探索木の構造が異なり，ユニットを動かす順番の探索の有無以外にも差異が厳密にはある．ただし，ランダム M-UCT のユニット行動木は，通常よりも分岐因子が減少している分，あるプレイアウト数に対する探索木の平均深さは，通常よりも深くなっている．つまり，通常よりも深い先の手の探索ができるということであり，その観点からいえばランダム M-UCT のユニット行動木の方が有利な条件での実験であった．それでも，通常の M-UCT よりも明らかに弱くなるということから，ユニットを動かす順番の考慮が棋力に大きな影響を与えているということが言える.

第4章　F-UCT

第二の提案手法 F-UCT[12][13][14] について述べる．F-UCT は M-UCT にファジィ評価を加えた手法である．

4.1　背景と目的

分岐因子が大きく探索が困難な戦略ゲームに対しては，モンテカルロ探索の有効性も限定的である．その一方で人間プレイヤはそれなりに強く，コンピュータプレイヤに対しても勝ち越す現状である．このとき，人間プレイヤがコンピュータより深く広い探索を行なっていることは考えづらい，人間が強いのは直感的な局面判断に優れ，大幅な探索の枝刈りが出来ているためと考えられる．

この，人が直感的に行うゲーム局面の大局的な評価をファジィ集合を介して行うファジィ評価を提案する．

ファジィ集合は人の持つ概念をある要素がどれだけその集合に含まれるかという帰属度 (メンバーシップ値) によって特徴付けて，あいまいな集合として定式化したものである．ゲーム局面の評価においては，人の知識に基づいて良い盤面という曖昧な集合を定義し，ゲーム中，新たに得られた局面の良い盤面集合への帰属度を参照することで人の直感に基づいたその局面の良さを評価できる．

4.2　ファジィ集合

ファジィ集合とはファジィ理論の根幹をなす概念で,「大きな数」のような境界の曖昧な集合である．Zadeh はこのようなファジィ集合 A を全体集合 X の上で，式 4.1 に示すメンバーシップ関数 μ_A を定義することで特徴付けた．

$$\mu_A : X \to [0,\, 1] \tag{4.1}$$

たとえばある要素 $x \in X$ があいまいな集合 A に含まれる度合いは $\mu_A(x) = 0.8$ などとして表現される．

4.3　ファジィ集合の同定と生成

ファジィ集合を構築することはメンバーシップ関数を定義することと同値である．メンバーシップ関数の同定にはいくつかの方法が提案されており，たとえば以下のようなものがある．

- 関数形を人が試行錯誤を経て直接生成する
- ファジィ数を基礎として生成する
- データに基づき生成する

ゲーム局面の状態空間は多変数で離散的なため，人手によったり，一次元のファジィ数をもとにすることは困難である．ある局面がどのように分類されるかについては，人の直感的な判断を得る事ができるので，データに基づいたファジィ集合の生成を行う．

表 4.1 に示す特徴量を持つ特徴量ベクトル $\mathbf{v}_i$ の空間 V 上で,「良い局面」というファジィ集合 G を生成することを考える．

以下の手順で生成することができる [17].

まず，サンプルの獲得を行う．

1. あるゲーム局面 s_i を一つ生成する
2. 人間プレイヤが G に含まれると言えるかどうか判断し $h(s_i) \in \{0, 1\}$ として含まれるかどうかを記録する
3. 十分なサンプル集合 $S = \{s_1, s_2, \cdots s_n\}$ が集まるまで 1. へ戻る

獲得したサンプルからファジィ集合を生成する．

1. 得られたサンプル s_i を特徴量ベクトル $\mathbf{v}_i = \mathbf{v}(s_i)$ に変換する
2. 表 4.1 に示す特徴量からなる特徴量空間でのデータサンプルの相対密度を各サンプル点で計測する．各サンプル点特徴量 $\mathbf{v}_i$ について標準距離 D_ε 以内の点集合 $N_i = \{\mathbf{v} | d(\mathbf{v}_i, \mathbf{v}) \leq D_\varepsilon\}$ の要素数を数え，その最大値 $\mathrm{Max}_N = \max_i(|N_i|)$ に対する割合 $\mu_G(\mathbf{v}_i) = |N_i|/\mathrm{Max}_N$ を $\mathbf{v}_i$ の帰属度とする．ここで $d(\mathbf{x}, \mathbf{y})$ は $\mathbf{x}$ と $\mathbf{y}$ のユークリッド距離である．
3. すべての $\mathbf{v_i} \in V$ についての帰属度を表 T_G として保持する

実装上は，ここで生成された帰属度の表 T_G がファジィ集合 G のメンバーシップ関数の実体である．表に含まれない一般の特徴量ベクトル $\mathbf{v} \in V$ については値の定まった要素から式 (4.2) に示す k 近傍集合 K からの重み付き平均補完によって求める．

$$\mu_G(\mathbf{v}) = \frac{\sum_{\mathbf{v}_i \in K} w_i \mu(\mathbf{v}_i)}{\sum_{\mathbf{v}_i \in K} w_i} \tag{4.2}$$

ここで $\mathbf{v}$ の k 近傍は, T_G 中で $\mathbf{v}$ への距離が近い順に k 個の $\mathbf{v}_i$ からなる集合であり, w_i は距離の逆数 $w_i = 1/d(\mathbf{v}_i, \mathbf{v})$ である．

4.4 F-UCT のアルゴリズム

F-UCT は M-UCT にファジィ判断の評価による探索優先度の順位付けを加える．ユニットの種類毎に生成されたファジィ集合には表 4.1 により分類される各行動ベクトルにメンバーシッ

表 4.1: 行動の特徴量

特徴量	要素数
行動ユニットの HP	10
攻撃対象ユニットの種類	7
攻撃ダメージ	11
次のターンに受けうる合計ダメージ	11
上下左右マスの塞がり度	5
周辺味方ユニット数	15
周辺敵ユニット数	15
ターン内の最終行動かどうか	2
総 HP 和の差	30
残りターン数	40
味方戦闘機数	10
味方攻撃機数	10
味方戦車数	10
味方自走砲数	10
味方対空戦車数	10
味方歩兵数	10
敵戦闘機数	10
敵攻撃機数	10
敵戦車数	10
敵自走砲数	10
敵対空戦車数	10
敵歩兵数	10
その行動が良いか悪いか	3

プ値が与えられている．このメンバーシップ値を探索優先度の重みに用い，M-UCT の探索木において新しく展開された子ノードの中からメンバーシップ値の高い行動を優先して探索する．

F-UCT の探索の流れは以下の通りである．

1. 局面の状態 s を根ノードの盤面の状態とする
2. s における未行動のユニットの行動の集合 $A(s)$ に含まれる a による次局面 $s'(s,a)$ が，s の子ノードとなる
3. ファジィ集合から $s'(s,a)$ のメンバーシップ値 $\mu_{G(a)}$ を求める
4. $s'_i = s'(s,a_i)$ に対し，s'_i の訪問回数 n_i を $n_i = \mu_{G(a)} \times 10$，報酬 X_i を 1 とする
5. M-UCT アルゴリズムに従う

M-UCT からの変更点として，展開された時点で各子ノードが既に訪問およびプレイアウトを数回行なった扱いになっており，プレイアウトの結果は全て自分チームが勝利したことにして初期報酬値を 1 としている．これにより，メンバーシップ値の高い行動，つまり良い行動として学習された行動の子ノードほど初期訪問回数が多いため優先して展開される．

4.5　対戦実験によるアルゴリズムの評価

F-UCT アルゴリズムの性能を AI 同士の対戦実験によって評価した．実験の進め方は M-UCT の性能評価と同様である．

4.5.1　F-UCT のプレイアウト数に対する棋力変化

M-UCT と F-UCT のプレイアウト数による棋力の変化を調べた．ベースラインアルゴリズムはプレイアウト回数 2000 回の M-UCT とした．

実験設定

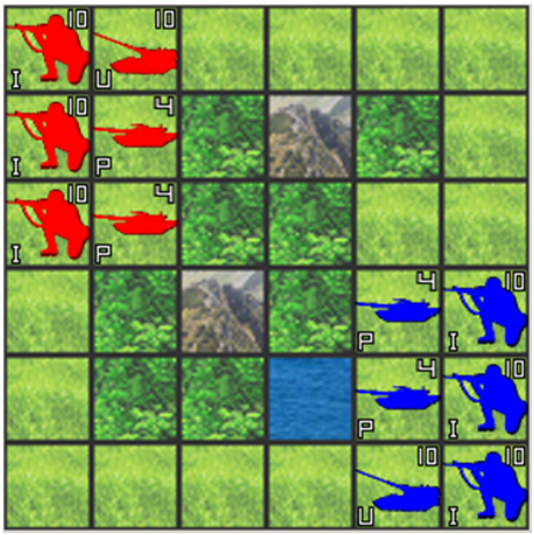

図 4.1: map01

図 4.1 に示す map01 で実験を行なった．このマップは，F-UCT がファジィ集合を生成するための学習データをサンプルする際に用いたマップである．

対戦，各アルゴリズムの設定を以下に示す．

- 対戦設定
  - 対戦回数:各 1000 回
  - ターン上限数:30，上限数を超えた時の残りユニットの合計 HP が 10 以上なら合計 HP が多いチームの勝利とし，それ以外は引き分け
- M-UCT および F-UCT のアルゴリズムの設定
  - 1 行動決定のプレイアウト数:500，1000，2000，4000
  - 各ノードの展開閾値:10
  - UCB 値の式の定数 c の値:0.15

実験結果

各実験結果を表 4.2，4.3 に示す．また，プレイアウト数による勝率の遷移を図 4.2 に示す．PO 数とは，プレイアウト数のことを指す．

表 4.2: PO 数 n 回 M-UCT vs PO 数 2000 回 M-UCT(各 1000 戦)

n	勝	負	引分	勝率
500	169	735	96	0.22
1000	349	563	88	0.39
2000	477	441	82	0.52
4000	532	414	54	0.56

表 4.3: PO 数 n 回 F-UCT vs PO 数 2000 回 M-UCT(各 1000 戦)

n	勝	負	引分	勝率
500	145	809	46	0.17
1000	313	594	93	0.36
2000	555	343	102	0.61
4000	716	201	83	0.76

考察

プレイアウト数を増やすことで，M-UCT，F-UCT ともに勝率の向上見られる．プレイアウト数 2000 回の M-UCT に対し，M-UCT はプレイアウト数を 500 回まで減らすと勝率が 0.22 まで減少するのに対して，4000 回まで増やしても勝率は 0.56 までしか増加しない．一方で F-UCT はプレイアウト数を 500 回まで減らすと勝率は更に減少し 0.17 となるが，4000 回で M-UCT を大きく上回る 0.76 となる．

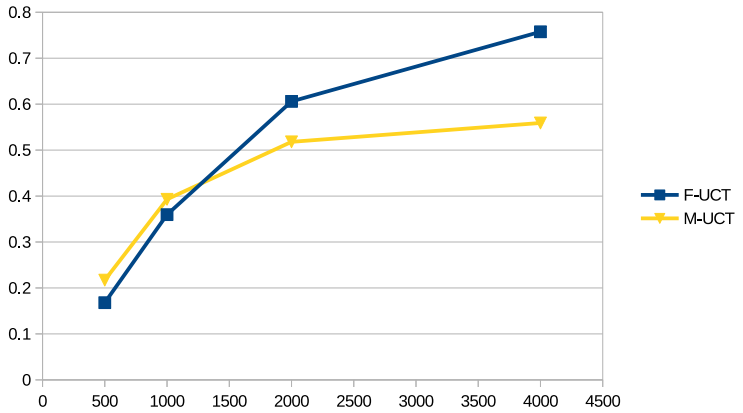

図 4.2: PO 数 2000 回の M-UCT に対する勝率の遷移

プレイアウト数が少ないと F-UCT は効果が薄く M-UCT よりも弱くなるが，回数を増やしていくにつれて効果が大きくなっていき M-UCT より強くなると考えられる．これは，プレイアウト数を増やすことで，F-UCT は探索優先度の重み付けが付けられている分，より深い探索ができているためだと考えられる．プレイアウト数が減少すると勝率が M-UCT よりも減少する理由としては，探索木の展開のバランスがうまく取れていないということが考えられる．F-UCT による探索優先度の重み付けは，必ずしも本当に良い手に重みを加えているとは言えず，少ないプレイアウト数では重みを加えた手が本当に良い手であったか分かる前に探索を終了してしまう可能性がある．F-UCT が特に効果を発揮するには，探索木をバランスよく展開するための一定のプレイアウト数が必要であると考えられる．

4.5.2 F-UCT の汎用性の検討

F-UCT が学習データをサンプルする際に用いたマップとは異なるマップでの対戦実験を行なった．学習マップだけでなく，他のマップにも対応できるかを確認する．ベースラインアルゴリズムはプレイアウト回数 2000 回の M-UCT とした．

実験設定

図 4.3 に示す map05 で実験を行なった．このマップは，TUBSTAP 同梱のマップで最も大きなマップであり，学習マップ (map01) には存在しなかった戦闘機，攻撃機，対空戦車が存在する．

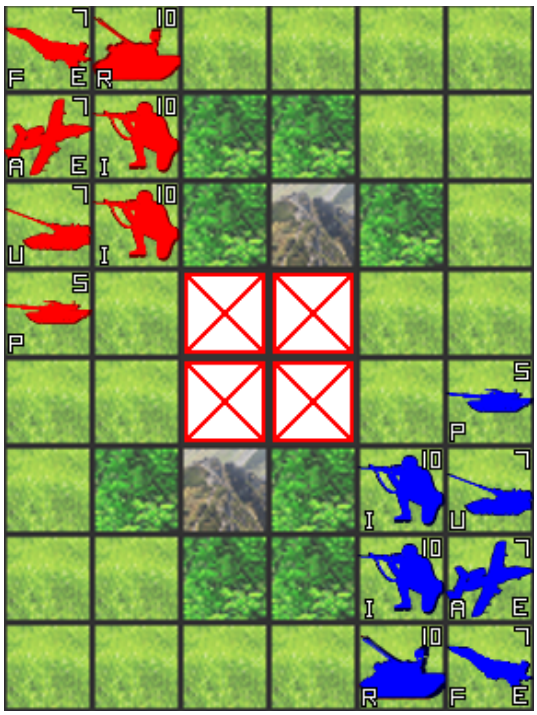

図 4.3: map05

対戦，アルゴリズムの設定を以下に示す.

- 対戦設定
  - 対戦回数:1000 回
  - ターン上限数:20，上限数を超えた時の残りユニットの合計 HP が 10 以上なら合計 HP が多いチームの勝利とし，それ以外は引き分け
- F-UCT のアルゴリズムの設定
  - 1 行動決定のプレイアウト数:2000
  - 各ノードの展開閾値:10
  - UCB 値の式の定数 c の値:0.15

実験結果

実験結果を表 4.4 に示す.

表 4.4: PO 数 2000 回 F-UCT(非学習マップ) vs PO 数 2000 回 M-UCT(1000 戦)

勝	負	引分	勝率
381	385	234	0.50

考察

プレイアウト数を M-UCT，F-UCT ともに 2000 回で固定しており，対等な条件での対戦だったが，結果はほぼ互角の強さとなり，探索優先度の重み付けによる効果が見られなかった．このことから 1 つのマップから得られる学習データだけでは，他のマップにも応用が効く汎用的な特徴を学習しきれていないことが考えられる．複数のマップから学習データを集めることで，特定のマップに依らないゲーム全体を表す特徴の学習が見込める．

4.5.3 F-UCT の類似マップへの汎用性

人間プレイヤの目線から，学習マップと似たマップでの対戦実験を行なった．対戦実験 2 のマップ 5 は，学習マップのマップ 1 と明らかに特徴が異なるマップであったが，ある程度似た特徴を持つと人間プレイヤが思えるマップならば学習マップと完全に同じマップでなくても効果が得られると考えられる．ベースラインアルゴリズムは対戦実験 2 に続きプレイアウト回数 2000 回の M-UCT とした．

実験設定

図 4.4〜4.7 に示す 4 つのマップで実験を行なった．これらのマップは TUBSTAP 同梱のマップではなく，マップ 1 に似た特徴を持つように今回の実験のために作成したものである．

類似マップ a は，縦横を 1 マスずつ小さくし，戦車と歩兵の数を減らしつつも，マップ 1 の初期配置の面影を残すように作成したマップである．類似マップ b は，4 つの類似マップの中で一番類似しているものと考えられ，マップ 1 に歩兵が 1 つ追加されただけである．類似マップ c は，そこから更に自走砲が 1 つ追加されたマップである．最後に類似マップ d は，縦横 1 マスずつ大きくなり，初期配置ユニットは類似マップ c を用い，地形から海を取り除き点対称マップにした．

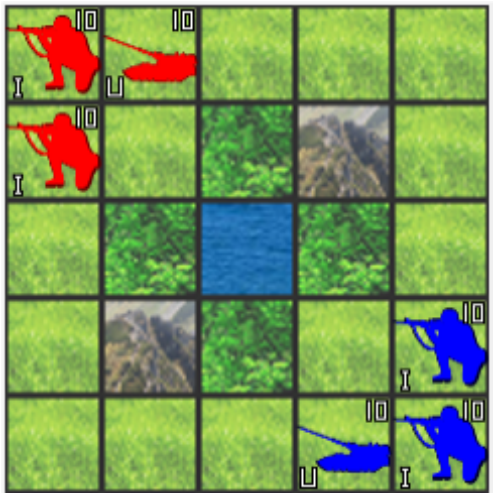

図 4.4: 類似マップ a

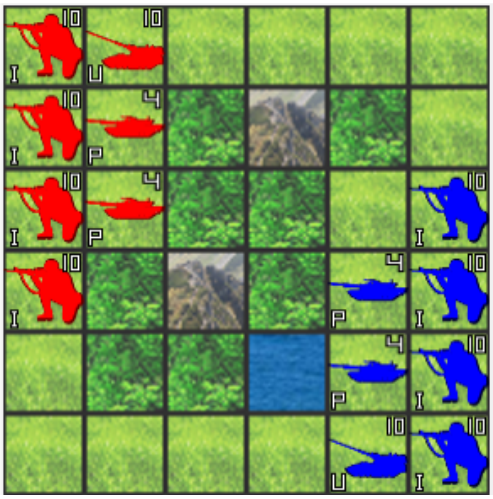

図 4.5: 類似マップ b

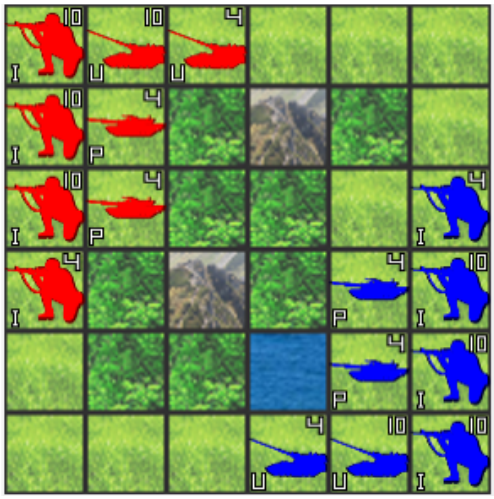

図 4.6: 類似マップ c

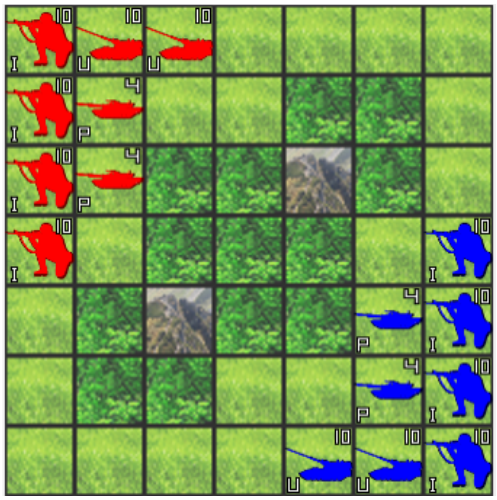

図 4.7: 類似マップ d

対戦設定を以下に示す．なお，F-UCT のアルゴリズムの設定は対戦実験 2 と同様である．

- 対戦設定
  - 対戦回数:各 1000 回
  - ターン上限数:a から順に，20，30，20，30 上限数を超えた時の残りユニットの合計 HP が 10 以上なら合計 HP が多いチームの勝利とし，それ以外は引き分け

実験結果

実験結果を表 4.5〜4.8 に示す.

表 4.5: PO 数 2000 回 F-UCT(類似マップ a) vs PO 数 2000 回 M-UCT(1000 戦)

勝	負	引分	勝率
533	210	257	0.662

表 4.6: PO 数 2000 回 F-UCT(類似マップ b) vs PO 数 2000 回 M-UCT(1000 戦)

勝	負	引分	勝率
574	323	103	0.626

表 4.7: PO 数 2000 回 F-UCT(類似マップ c) vs PO 数 2000 回 M-UCT(1000 戦)

勝	負	引分	勝率
432	320	248	0.556

表 4.8: PO 数 2000 回 F-UCT(類似マップ d) vs PO 数 2000 回 M-UCT(1000 戦)

勝	負	引分	勝率
429	476	95	0.474

考察

類似マップ a の勝率が最も高くなった．マップサイズが異なるため，類似マップ b や c よりも特徴が離れているようにも見えるが，学習マップの部分問題マップと見なすこともできるため，学習の効果が大きく現れたのだと考えられる．類似マップ b でも F-UCT が勝ち越し，歩兵が 1 つ増える程度なら学習結果が応用できることが分かる．しかし，類似マップ c の，自走砲も追加される段階まで来ると，学習できない局面が増えてくるのか，勝率の減少が見られる．類似マップ d では，僅かに M-UCT に負け越してしまい，学習効果が出ていないと考えられる．

これらの結果から，ユニット数やマップサイズの増減から，マップの特徴は確かに変わっていき，学習すべき局面も増減することが考えられる．F-UCT は，学習マップと完全に一致するマップでなくても，ある程度特徴が似たマップでは効果を発揮し，棋力が向上することが考えられる．

第5章　結論と今後の展望

5.1　結論

本研究では，探索が困難な問題の1つであるターン制戦略ゲームからTUBSTAPを研究対象として用い，2つの手法M-UCTとF-UCTを提案し，実験からその効果を示した．

M-UCTは，複数着手性のあるゲームにおいて，1つのユニットの行動ごとに分解されたゲーム木であるユニット行動木を構築し，その上でUCT探索を行なって行動決定する．ユニット行動木を用いた探索は，ユニットを動かす順番を考慮でき，この特性が棋力の向上に大きく影響した．既存の手法に対してもM-UCTは対戦実験で勝ち越し，棋力の高いAIプログラムを作ることができたと言える．

F-UCTはM-UCTにファジィ評価を加えた手法で，人間の直感評価をモデル化し，探索の更なる効率化を図った．学習に使用したマップ，および学習マップに特徴が類似したマップにおいて，F-UCTはM-UCTよりも強くなり，ファジィ評価の有効性を示した．

5.2　今後の課題

F-UCTの効果が限定的であったため，多様なマップに対応するためのポイントが何かを発見することが今後の課題となる．今回ファジィ集合を生成するために用いた特徴量は22個であったが，ここから特徴量を増減させることでどのように性能に影響するか調べる必要がある，

そして，M-UCT，F-UCTともに効果を示したため，これらで用いたアルゴリズム，つまりユニット行動木の探索やファジィ評価を他の分野に適用していくことが，1番の課題となる．

参考文献

[1] Murray Campbell,A. Joseph Hoane Jr.,Feng-hsiung Hsu, ”Deep Blue ”,Artificial Intelligence 134 (2002)

[2] Takenobu TAKIZAWA, ”Contemporary Computer Shogi ”,IPSJ SIG Technical Report 2013-6-28

[3] Silver D,Huang A,Maddison C J,Guez A,Sifre L, van den Driessche G,Schrittwieser J,Antonoglou I,Panneershelvam V, Lanctot M,Dieleman S,Grewe D, Nham J,Kalchbrenner N,Sutskever I,Lillicrap T, Leach M,Kavukcuoglu K,Graepel T, Hassabis D, ”Mastering the game of Go with deep neural networks and tree search ”, Nature 2016 529(7587)

[4] 村山公志朗, 藤木翼, 池田心, ”学術研究用プラットフォームとしての大戦略系ゲームのルール提案 ”,IPSJ-GPW 2013-11-01

[5] David Silver,Gerald Tesauro, ”Monte-carlo simulation balancing ”, Proceedings of the 26th Annual International Conference on Machine Learning 2009

[6] 保木邦仁ほか, ”局面評価の学習を目指した探索結果の最適制御 ”, 第 11 回ゲームプログラミングワークショップ　2006

[7] Zadeh,A. Lotfi, ”Fuzzy sets ”, Information and control Vol.8 No.3 (1965)

[8] Zadeh,A. Lotfi, ”The concept of a linguistic variable and its application to approximate reasoning ”, Information sciences Vol.8 No.3 (1975)

[9] ターン制戦略ゲーム 学術用基板プロジェクト TUBSTAP

http://www.jaist.ac.jp/is/labs/ikeda-lab/tbs/

[10] 藤木翼, 村山公志朗, 池田心, ”ターン制ストラテジーのための状態評価型深さ限定モンテカルロ法 ”, エンターテイメントと認知科学研究ステーション第 8 回シンポジウム

[11] 藤木翼, 村山公志朗, 池田心, ”ターン制ストラテジーのための状態評価型深さ限定モンテカルロ法における消極的行動の抑制 ”, 第 19 回ゲームプログラミングワークショップ,pp.32-39,2014

[12] 武藤孝輔, 西野順二, ”ターン制戦略ゲームにおける UCT とファジィ評価の適用 ”,IPSJ-GPW 2015-11-06

[13] 武藤孝輔, 西野順二, ”ターン制戦略ゲームにおけるファジィ大局的評価 ”, 第 32 回ファジィシステムシンポジウム,2016-8-31

[14] Kosuke Muto,Junji Nishino, ”Fuzzy Evaluation of Macroscopic Situation for Turn based Strategic Games ”,SCIS&ISIS2016 August 25-28 2016

[15] 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 ”, Computational Intelligence and AI in Games,IEEE Transactions on,Vol. 4,No. 1,pp. 1 43, march 2012

[16] 加藤千裕, 三輪誠, 鶴岡慶雅, 近山隆, ”ターン制ストラテジーゲームにおける戦術決定のための UCT 探索とその効率化 ”,IPSJ-GPW 2013-11-01

[17] 西野順二, ”このへんファジィとそのあたり ”, 知能と情報 (日本知能情報ファジィ学会誌) Vol.20 No.5 pp776-784 (2008)

謝辞

本研究に際して，日頃より様々なご指導を頂いた西野順二助教に心より御礼申し上げます．また，様々な手続きでお世話になった主任指導教員の緒方秀教教授，合同ゲームゼミで貴重なご意見を頂いた村松正和教授，保木邦仁准教授，荒木信夫氏，そして研究生活の中で多くの助言を頂いた研究室の皆様に深く感謝致します．

付 録A　大会記録

これまでに行われた過去 3 回の TUBSTAP の大会の記録を載せる．これらの記録は，公式ホームページ [9] からも閲覧可能である．

A.1　対戦会 in GPW2015

公式で開催された初の大会である．ゲームプログラミングワークショップ (GPW) では，GPW 杯というゲーム AI の大会が開催され，2015 年から TUBSTAP も追加された．

参加者は自作の AI プログラムとマップを持参して参加した．戦略シミュレーションゲームとは本来同じマップでゲームを行うことが少なく，両対戦チームが完全に知らないマップで戦うことが公平である．そのため，その対戦に関わらない第三者が用意したマップで対戦を行えるように，参加者は AI プログラムだけでなく，他の参加者に対戦してもらうためのマップを用意した．

対戦会の規定は以下の通りであった．

- AI は 1 ターンを 10 秒で終わらせる
- マップは常に対戦に関わらない者が用意したものを用いる
- マップの学習時間などは用意しない
- スイス式 (参加者数によっては総当り)
- C#で書かれた開発用フレームワークをベースとして開発する
- マップサイズは最大で 8×8，ユニット数は 1 チーム 8 ユニットまで

A.1.1 結果

大会結果を表 A.1 に示す．対戦は総当りで先手後手 1 戦づつ行なった．

表 A.1: GPW2015 大会結果

番号	名前	1	2	3	4	5	勝ち数	順位
1	MilitaryStaffSystem		○×	○×	○○	○○	6	1
2	Fenrir	×○		○×	×○	○○	5	2
3	DLMC+AAC	×○	○×		×○	××	3	3
4	AI_Sato	××	○×	○×		○×	3	3
5	AI_Muto		××	××	○○	×○	3	3

A.2 GAT2016 TUBSTAP 大会

第二回の大会は，Game AI Tournaments 2016 で開催された．

参加者は前回同様に，AI プログラムとマップを持参し，対戦に関わらない第三者が用意したマップで対戦を行うように進行した．

大会の規定は前回と同じである．

A.2.1 結果

大会結果を表 A.2 に示す．まず予選として，総当りで先手後手 2 戦づつ行い，勝利点の高い 2 つのプログラムが決勝戦を行なった．決勝戦は 2 マップ使用し，先手後手各 1 戦 (合計 4 戦) 行なった．

表 A.2: GPW2015 最終順位

作者名	AI 名	順位
Muto	M_UCT	1
Fujiki	M3Lee	2
Ishitobi	MilitaryStaffSystem	3
Sato	AI_Sato	4
Masuda	Fortress_Formation	5
Takahashi	AI_Takahashi	6
Kimura	AI_Kimura	7

A.3　対戦会 in GPW2016

第三回の大会は，第 21 回ゲームプログラミングワークショップ (2016) で開催された．
大会の進行方法，規定は前回と同じである．

A.3.1　結果

大会結果を表 A.3 に示す．まず予選として，総当りで先手後手 2 戦づつ行い，勝利点の高い 2 つのプログラムが決勝戦を行なった．決勝戦は先手後手 2 戦を，引き分け (1 勝 1 敗または 2 回とも引き分け) ではなくなるまで様々なマップで対戦した．

表 A.3: GPW2016 最終順位

作者名	AI 名	順位
Sato	SatD2	1
Muto	M_UCT	2
Sagehashi	M_UCT+MaxActEval	3