UCT 探索を用いた大貧民クライアント

谷聖一　研究室　　佐藤 友啓
Tomohiro Sato

概要

電気通信大学が主催する UEC コンピュータ大貧民大会に参加することを目的として，大貧民クライアントプログラムを作成した．モンテカルロ法による乱数を使ったシミュレーションを基礎として，全合法手に対して決められた回数分シミュレーションするのではなく，シミュレーションの勝率が高く，かつ，シミュレーション回数が少ない合法手を優先的に選択するように，シミュレーション割り振り法を UCT 探索によって実装した．

1　はじめに

１９５０年代より人工知能 (AI) の研究が盛んに行われている．AI とは Artificial Intelligence の略であり，コンピュータに人間と同様の知能を実現させようという試みである．近年 AI はロボットやゲームなどの分野でも用いられている ([1])．ゲームにおける AI プログラムでは様々なアルゴリズムが提案されているが，その全てが必ずしも精度の高い解を出力するとは限らない．一般的に行動や状態に関する情報が全て与えられているゲームを完全情報ゲームと言う．例えば，オセロや将棋は完全情報ゲームである．AI プログラムを人間と戦わせることも盛んに行われており，将棋では２０１０年に AI プログラムが女流王将に勝利した実績 ([2]) がある．

完全情報ゲームに対する AI プログラムでは最善手を判断するために評価関数が用いられている．評価関数とは局面の優劣を評価するために用いられ，その精度が高いほど AI プログラムの性能は向上する．しかし精度の高い評価関数を作成するにはゲームに関する知識やその知識をプログラムとして表現する必要があり，良い評価関数の作成は困難であることが多い．このような問題に対応する手法にモンテカルロ法がある．モンテカルロ法とは乱数を用いたシミュレーション (プレイアウト) を複数回行い解を近似的に求める統計的な手法である ([3])．モンテカルロ法は評価関数を用いないため，様々なゲームに有効とされている．

不完全情報多人数ゲームにおいても研究が行われている ([4, 5])．不完全情報ゲームとは，お互いの情報が不明なゲームであり，例えば 7 並べや麻雀，大貧民などが挙げられる．戦略を考えるのが困難な上，運の要素も大きいという特徴がある．完全情報ゲームで有効とされている戦略が不完全情報多人数ゲームに対しても有効かが議論されている．

２００６年から電気通信大学では UEC コンピュータ大貧民大会 ([6]) が毎年開かれている．UEC コンピュータ大貧民大会ではサーバークライアントシステムが採用されている．サーバークライアントシステムとはクライアントはサーバーに処理を依頼して，サーバーはクライアントの依頼を受け結果を返信することである．この大会の場合，クライアントがカードを選択しサーバーに提出．サーバーは提出されたカードが正しいかどうかを判定して場の更新を行っている．

今年度で 7 回目であるが，第 4 回からモンテカルロ法を使用したクライアントが成果を上げている．この大会に参加するにおいて谷聖一研究室に所属している松藤，長谷川と 3 人共同でクライアントを実装した．表 1 では実際に UEC コンピュータ大貧民大会に出場した成績である．合計１０チームが参加しており，２ブロックに分かれて上位２チームと，両ブロックで得点が高い 3 位が決勝に行けるシステムになっている．結果は 5 位で予選敗退という結果に終わった．

1 位	2 位	3 位	4 位	5 位
1459	1276	1248	1009	1008

表 1 :予選 A ブロックでの戦い

この大会では 1 手当たりの制限時間が設けられているため，限られた時間の中でより良質な手を導きだすことがクライアントに要求される．一方，佐藤・松藤・長谷川 3 人共同で開発したモンテカルロ法クライアントは，制限時間内で良質な解を得るのに十分な回数プレイアウトを実行できなかった．そこで，探索空間が大きくなることを防ぎ，かつ効率的にプレイアウトをさせる手法として UCT(Upper bound Confidence for Tree) 探索を用いて UEC コンピュータ大貧民大会クライアントを実装した．

本演習では完全情報２人ゲームでは有効とされているモンテカルロ法を不完全情報多人数ゲームに応用させ，UCT 探索を用いてのプレイアウトの効率化する．UEC

コンピュータ大貧民大会に出場することを目標として不完全情報多人数ゲームにおける強いクライアントを開発することを目標にした．

本論文は以下のような構成になっている．２節では作成したクライアントについて述べる．３節では作成したクライアントを用いて実験を行う．４節では今後の課題について述べる．

2　作成クライアント

本節では完全情報ゲームで採用さている代表的な手法を説明する．

2.1　モンテカルロ法

大貧民におけるモンテカルロ法 ([3]) とは，ある局面の全合法手に対して決められた回数プログラム内でプレイアウトして勝率を求め，最も勝率が高かった合法手を実際に選択する手法である．合法手 i を選択してプレイアウトを行った回数を s_i，プレイアウトによって得られた報酬 (ポイント) の和を X_i をとする．この時の勝率 $\overline{X}_i$ は

$$\overline{X}_i = \frac{X_i}{s_i}$$

によって与えられる．本演習で作成したクライアントでは，報酬を表２のように定めた．

大富豪	富豪	平民を	貧民	大貧民
1	0.75	0.5	0.25	0

表２:プレイアウトの報酬の割り振り

プレイアウトでは現在の局面からゲームの終わりまでランダムに合法手を選ぶことで局面の展開を行い手札もランダムに配分した．この時貧民や大貧民はジョーカーを持っていないこと，交換したカードは仕様によりわかっているため，それら以外をランダムに配分した．またプレイアウト中の合法手の選択を３回に２回の確率で大会側が配布しているデフォルトクライアントの提出方法に従い，残った確率ではランダムに提出する．これは少しでもゲームの流れに近づけるためである．しかしこの方法では，合法手の数が増加するとプレイアウト回数も増加して処理時間が長くなるという問題が生じる．この問題を改善するために UCT 探索を適用する．

2.2　UCT 探索

本小節ではモンテカルロ法に多腕バンディット問題のアルゴリズムを利用した UCT 探索 ([7]) について説明する．

2.2.1　多腕バンディット問題

多腕バンディット問題 ([8]) とは，多くの腕を持つスロットマシンをプレイするギャンブラーに基づく機械学習問題である．

スロットマシンをプレイすると，ギャンブラーは各腕のある確率分布に基づいた報酬を得ることができる．ギャンブラーはスロットマシンを繰り返しプレイして，その報酬の合計を最大化することを目的とする．ギャンブラーはスロットマシンに関する知識を持たず，プレイを繰り返すことで報酬を高く得られる腕を見つけ出し，集中的にプレイすることができるようになるとされている．

多腕バンディット問題は，既に得られている知識に基づいて報酬を最大化することと，他の腕をプレイして，その腕に関する知識を増やすこととのバランスを取る問題であり，強化学習では「収穫と探検のジレンマ」として知られている．

モンテカルロ法の場合に当てはめると，それぞれの合法手を多腕バンディット問題におけるそれぞれの腕として，プレイアウトの勝敗の結果が得られる報酬となる．

2.2.2　UCT 探索

UCT 探索ではプレイアウトを行う合法手に制御を与え，結果的に選択されそうな手に対して多くのプレイアウト回数を行う手法である．この制御のために多腕バンディット問題に対する解法として用いられている UCB1 アルゴリズム ([9]) を利用しており，合法手の選択は UCB(Upper Confidence Bound，信頼上限) 値によって選択を行う．$\overline{X}_i$ を合法手 i の勝率，s_i を探索中に合法手 i が選択された回数，n をその時までに行った総プレイアウト回数とする．合法手 i の UCB 値は以下のように定義される．

$$\mathrm{UCB}(i) = \overline{X}_i + \sqrt{c\frac{\log n}{s_i}}$$

UCB 値が最も高い合法手を探索 (プレイアウト) する．UCB 値の第１項により勝率の高い合法手を，第２項により探索回数の少なさを数値で表し，探索回数が少ないほど選択されやすくする．第１項，第２項により勝率が高く探索回数が少ない合法手が選択されやすくなる．c は探索回数による影響の大きさを与える定数で $c = 2$ とした．また制作したプログラムの処理上，合計探索回数を８００回にすることにした．

2.2.3　UCB1-Tuned

UCB1-Tuned([10]) は UCB1 アルゴリズムを改善したものである．UCB1 アルゴリズムで UCB 値を求める際

に使用した c を定数として与えるのではなく以下の式によって動的に与える．この時 σ_i^2 は報酬の分散である．

$$V_i = \sigma_i^2 + \sqrt{\frac{2\log n}{s_i}}$$
$$c = min(\frac{1}{4}, V_i)$$

この式を与えることにより分散が考慮され，探索が進んだ時に極端に勝率の低い合法手が探索される回数が減少する．

2.2.4 Progressive Pruning

Progressive Pruning とは ([11]) 勝率と報酬の標準偏差から勝率が取り得る値の区間を推定する．その結果，選ばれる見込みの無い合法手を求め，探索を行わないことでモンテカルロ探索の効率を改善する．合法手 i の勝率を $\overline{X}_i$，報酬の標準偏差を σ_i，プレイアウト回数を s_i とした時に，区間推定を用いて以下のようにそれぞれの合法手の勝率が取り得る値の区間を求める．

$$\overline{X}_{Li} = \overline{X}_i - r\frac{\sigma_i}{\sqrt{s_i}}$$
$$\overline{X}_{Ri} = \overline{X}_i + r\frac{\sigma_i}{\sqrt{s_i}}$$

$\overline{X}_{Li}$ では合法手 i がどの程度まで勝率が下がる可能性があるかを推定した値であり，$\overline{X}_{Ri}$ では合法手 i がどの程度まで上がる可能性があるかを推定した値である．つまり合法手 i の勝率は $[\overline{X}_{Li}, \overline{X}_{Ri}]$ となる可能性が高い．ここで r は信頼係数である．合法手 i と j が存在して，$\overline{X}_{Ri} < \overline{X}_{Lj}$ である場合，合法手 i の勝率が合法手 j の勝率より高くなる可能性は低い，よって合法手 i が選択させる可能性は低いために，これ以上探索しても探索結果に影響を与える可能性は低く，枝狩りすることができる．しかし今回制作したクライアントにこの戦略を加えても強くはならなかった．理由としてプレイアウト回数が足らない，UCB1 アルゴリズムとの相性の関係性，プログラムに問題があることが考えられる．

2.3 必勝手順アルゴリズム

ゲーム AI プログラムを作成するにおいて最も強い戦略とは全探索することである．しかし全探索には膨大な時間を要するため，制限時間内での探索は困難である．一方，大貧民では特定の状況の中では必ず上がれる手順が存在し，それを容易に実装することができる．特定の状況とは以下のことを言う．ある局面において自分のターンから始まる場合，持っている手札の中で切り札以外のカードが 1 手以下ならば必勝手順が存在する．ここで切り札とはそのカードを出せば他のプレイヤーが必ずパスになるカードのことを指す．また場に他プレイヤーのカードが存在した場合，そのカードの上に切り札を提出し，残りの手札が上記の「自分のターンから始まる場合」に当てはまるならば必勝手順が存在する．

この戦略を用いることでプレイアウトをする必要がなくなるために時間が短縮され，特定の状況の中で必ず上がることができる．

3 実験

作成したクライアントが強くなるか，試合回数を４００回として実験する．対戦相手は第５回優勝クライアントである snowl([12]) と大会側から配布されているデフォルトのクライアントを２つ採用する．デフォルトのクライアントとは最低限ゲームが動作するクライアントであり，カードを小さい順に提出するクライアントである．デフォルトのクライアント以外のクライアントの実行時間は全て統一して実験を行った．1 試合毎に貰えるポイントの割り振りは表３の通りである．

大富豪	富豪	平民	貧民	大貧民
5	4	3	2	1

表３:報酬の割り振り

表４では必勝手順の有無を対戦させた．

snowl	必勝有	必勝無	def	def
1544	1387	1323	914	832

表４:必勝手順の有無で対戦させた結果

表５では UCT 探索の有無で対戦させた．

snowl	UCT 有	UCT 無	def	def
1529	1349	1277	935	910

表５:UCT 探索の有無で対戦させた結果

表６ではプレイアウトの回数を変更して対戦させた．

snowl	5000 回	800 回	def	def
1552	1443	1321	844	840

表６:プレイアウト回数を変えて対戦させた結果

表７では UCB1 アルゴリズムと UCB1 アルゴリズムを改善した UCB1-Tuned を対戦させた．

snowl	UCB1-Tuned	UCB1	def	def
1506	1429	1396	870	799

表７:UCB1-Tuned と UCB1 を対戦させた結果

表４から，必勝手順が有効である結果になった．しかしポイントの差が小さいのでプレイアウトの精度が高いことが考えられる．表５では UCT 探索の有効性が見受けられる．表６では勝率が探索回数に依存していること

がわかる．snowl に勝てないのは，プレイアウトの精度が snowl の方が高いこと，根本的な問題として作成したクライアントにバグが存在する可能性がある．表 7 より UCB1-Tuned が強くなることがわかるが，差が小さいので思った以上の結果を残すことができなかった．

4 おわりに

本演習では，モンテカルロ法によるプレイアウト中の手や手札をランダム選択することに加えて，知り得る情報を考慮した選択手法を提案し，プレイアウト割り振り法を UCT 探索に基づいて実装した．製作中での試行では snowl には及ばないものの，得点を向上することができ，これらの手法が有効であると確認できた．しかし 2011 年に開催された第 7 回優勝クライアントを検証した結果，UCT 探索と UCB1-Tuned は既に実装されていることがわかった．今後の課題は，制作したクライアントにバグが存在するか見直すこと．またその他に不完全情報多人数ゲームで有効なアルゴリズムを実装することが課題である．

参考文献

[1] 日本ロボット学会. http://www.rsj.or.jp/. アクセス日 2013.02.2 20:00.

[2] 清水女流王将 VS あから 2010, http://blog.livedoor.jp/hana180sx/archives/51628748.html, アクセス日 2013.01.21 18:00.

[3] B.Brügmann. Monte Carlo Go. Technical report, Physics Department, Syracuse University, 1993.

[4] J. Schäfer, M. Buro, and K. Hartmann. The UCT Algorithm Applied to Games with Imperfect Informaiton. Diploma thesis. Otto-von-Guericke-Universität Magdeburg, 2008.

[5] N.R. Strurtevant. An Analysis of UCT in Multi-player Games. Proceedings of the 6th inter-national conference on Computers ad Games, pp. 37-49, 2008.

[6] 電気通信大学, UEC コンピュータ大貧民大会, http://uecda.nishino-lab.jp/, 2006 年 ～.

[7] L. Kocsis, C. Szepesväri. Bandit based monte-carlo planning. In Enropean Conference on Machine Learning, pp. 282 - 293, 2006.

[8] B.Bouzy and T.Cazenave. Computer go: An AI oriented survey. Artificial Intelle-gence, 132(1): 39-103, 2001.

[9] P.Auer, N. Cesa-Bianchi, P.Fischer. Finite-time analysis of the multiarmed bandit problem. machine Learning, 47 (2/3), pp. 235 - 256, 2002.

[10] B. Bouzy and G. Chaslot. Bayesian generation and integration of k-nearest-neighbor patterns for 19x19 go. In G. Kendall and Simon Lucas, editors, IEEE 2005 Sympo-sium on Computational.

[11] B.Bouzy.Move-Pruning Techniques for Monte-Carlo Go.CG 2005 LNCS,vol.4250,pp.104-119,SpringerHeidelberg,2006.

[12] 須藤 弥, 成澤 和志, 篠原 歩, UEC コンピュータ大貧民大会向けクライアント「snowl」の開発, http://uecda.nishino-lab.jp/sympo/images/Suto.pdf, 2007 年.