GPCC報告(2017年)

Games and Puzzles Competitions on Computers

http://hp.vector.co.jp/authors/VA003988/gpcc/gpcc.htm

藤波順久*

1　2017年の課題

2017年のGPCCでは、以下の課題を取り上げた。

タワーチェス　二人で行うボードゲームである。6×6の盤で、一人につきポーン6、ルーク2、ビショップ2、クイーン1、キング1を使う(ナイトはない)。駒の動かし方はチェスに似ているが、動かした先で自分または相手の駒の上に乗せることができる(飛び越すことはできない)。相手のキングの上に乗せると勝ちとなる。自分のキングの上に乗せることはできない。動かせるのは一番上の駒だけである。

ポーンの動きは特殊で、最初の位置からは2マス進んでもよい。また、駒に乗るときは斜め前に進まなければならない。キャスリングなど他の特殊ルールはない。

TANTRIX　いくつか遊び方があるが、ここでとりあげるのはタントリックスストラテジーと呼ばれるもので、2〜4人で行うゲームである。使用するのは56枚の6角形のタイルで、赤黄緑青の中から3色の線が描かれている。線は各色1本ずつ、2辺をつなぐように描かれている。3本とも向かい合う辺を結ぶものを除く、すべての組み合わせ(回転して同じになるものを除いて56通り)がある。

タイルを置くときは、接する辺の色が合うようにする。置いていくとできることがある、タイル3枚で囲まれたスペースを、ゴブルと呼ぶ。

ゲームを始めるには、タイルをすべて袋に入れ、各プレーヤに6枚ずつ、袋から取り出して表向きで配る。プレーヤは自分の担当する色を決める。

プレーヤは自分の番で以下を行い、1枚置くたびに袋からタイルを1枚取り出す(表向き)。

- ゴブルがあれば置ける限りタイルを置き続ける
- 1辺または2辺で接する場所にタイルを1枚置く
- ゴブルがあれば置ける限りタイルを置き続ける

終盤(=袋のタイルがなくなる)になるまでは、置き方にさらに制約がある。

*株式会社ソニー・インタラクティブエンタテインメント、GPCC chair

- 3辺が同じ色のゴブルを作ってはいけない
- タイル4枚で囲まれたスペースを作ってはいけない
- ゴブルを埋めていくといずれはタイル4枚で囲まれたスペースができるような位置に置いてはいけない

タイルを全部置き終わったら、プレーヤの担当する色の最長のラインまたはループを探す。ラインなら構成するタイル数、ループならその2倍が得点となる。

ガイスター　二人で行うボードゲームである。6×6の盤にそれぞれ8個の駒を以下のように配置する。

←▼▼▼▼→ 矢印は出口
＊▼▼▼▼＊
＊＊＊＊＊＊
＊＊＊＊＊＊
＊▲▲▲▲＊
←▲▲▲▲→

駒は赤4個、青4個で、相手の駒の色は色は見えない(駒を取ると見られる)。初期配置でどこに赤と青を置くかは自由に決められる。自分の番では駒を前後左右に1マスずつ動かす。相手の駒と同じマスに動かすと、取ることができる。以下のどれかを満たすと勝ちである。

- 自分の赤い駒を全て相手に取らせる
- 相手の青い駒を全て取る
- 敵陣の出口に青い駒を置いた状態で自分の番になる(脱出する)

千日手を防ぐため、先手後手を合わせて254手目に後手が指して上の条件を満たさなかった場合は、引き分けとする。

2　2017年の進展

タワーチェス については八木原勇太さんがプログラムを作成中とのことである。

TANTRIX については、明星大学の長江恭英さんと丸山一貴さんが、第59回プログラミング・シンポジウムのポスター発表で、オンライン対戦システムのフレームワークを利用したAIプレイヤーをベースに、戦略を改善しようという試みを報告する予定である。詳細は同シンポジウム予稿集の「タントリックスストラテジーにおける序盤の妨害要素を追加したAIの実装」を参照してほしい。

ガイスターについては、サイボウズ・ラボの西尾泰和さんがプログラムを作成し、第22回ゲームプログラミングワークショップ(GPW-17)中に開催されたガイスターAI大会[1]で準優勝した。そのアルゴリズムの解説を3節に示す。

[1] http://www2.matsue-ct.ac.jp/home/hashimoto/geister/

3　部分観測モンテカルロ計画法を用いたガイスターAI(西尾泰和さん)

3.1　本稿の目的

2017年11月のGPWでのガイスターAI大会に提出し準優勝となったAIの中身を簡単に解説することで、観測できない情報の推測が重要な状況でのAIの作り方に関する研究を促進する。

3.2　離散的な論理で考えることの落とし穴

たとえばジャンケンで、グーで買った場合は1点、チョキとパーで買った場合には2点得られるというゲームがあるとしよう。このゲームの強いAIを作るにはどうしたらいいだろうか?

最初の一歩として、相手が1/3の確率で手を選ぶとしよう。その場合、自分はチョキを出すのが最も得られる点差の期待値が高い。自分が勝ったら2点、相手が勝ったら1点だからだ。

ならば「チョキを出す戦略」が最善の戦略か？お分かりの通りその戦略は「グーを出す戦略」に負ける。その戦略は「パーを出す戦略」に負ける。そしてその戦略は「チョキを出す戦略」に負ける。論理的に考えているつもりが、思考が堂々巡りになって、AI作りを諦めてしまう。

このゲームにおいて、取りうる行動の選択肢は「グー、チョキ、パー」の3通り。このゲームの戦略を「状況を引数にとって、3つの行動のどれかを返す関数」と捉えていると上記の堂々巡りにはまる。そうではなく「状況を引数にとって、3次元の実数値ベクトルを返す関数」に拡張する必要がある。この関数の返り値は総和が1とする。離散の確率分布である。この枠組みの中では「グーを出す戦略」は返り値が(1, 0, 0)であり、「1/3の確率で各選択肢を選ぶ戦略」は(1/3, 1/3, 1/3)である。

行動の選択肢が確率的に選ばれる、という枠組みは、ゲーム理論の言葉で言えば「混合戦略ゲーム」である。混合戦略ゲームでは少なくとも1つのナッシュ均衡が存在することがナッシュの定理で証明されている。ここでナッシュ均衡とは「どのプレーヤーも自分の戦略を変更することによって、より高い利得を得ることができない戦略の組み合わせ」である。ガイスターAIが十分強くなれば、このナッシュ均衡に到達するだろう。

3.3　部分観測マルコフ決定過程

ある盤面状態sで、自分が行動aをとった場合、次に自分の手番になった時の盤面状態s'はどう表現できるだろうか。

強化学習の分野では、これをマルコフ決定過程(MDP)だと考える。つまりある状態sから別の状態s'へ、遷移確率$P(s, a, s')$に従って確率的に遷移するというわけだ。

この「盤面状態」は、自分に見えている盤面ではない。相手にしか見えない隠れ情報も含んだものである。なぜなら、相手が各行動を選択する確率は、相手にしか見えない情報によって影響を受けるからである。つまり、ガイスターはマルコフ決定過程だが、片方のプレイヤー

は状態のすべてを観測することができない。これを部分観測マルコフ決定過程と呼ぶ。強化学習の分野でよく研究されている問題設定である。

片方のプレイヤーに観測できない情報とは「相手の駒の初期配置が${}_8C_4$の70通りのうちのどれであったか」である。どれであったと思うかを0〜1の実数で表現すると70次元の実数値ベクトルになる。これを相手の初期配置に対する「信念」と呼ぶ。このベクトルは総和が1で、離散の確率分布と考えられる。

マルコフ決定過程の「状態」を離散な集合であると勘違いしている人も多い。ロボットの姿勢制御などをイメージして頂ければ、一般に状態は連続値を取りうることがわかるだろう。というわけで70次元実数値の信念ベクトルを状態に含めよう。部分観測マルコフ決定過程において、観測できない情報に対する信念を確率分布として状態に含めることで、新たな完全観測のマルコフ決定過程を作ることができる。これをbelief MDPという。

ガイスターにおける「相手の色を推定する」という部分問題は、信念という事前分布を相手の行動情報を観測することで更新していくベイズ推定のプロセスだと考えられる。具体的な例として「相手の駒がゴール直前にいるのに、相手の手番で相手はゴールしなかった」と観測した場合を考えてみよう。自分が相手の立場ならどうするだろうか、と考える。事前には赤か青かわからなかったので両方50%の確率とする。駒が青なら自分なら25%ゴールしないとする。0%だろと思うかもしれないが説明の都合だ。赤ならゴールできないから100%ゴールしない。観測事実は「ゴールしない」だった。ということは $50\% \times 25\% + 50\% \times 100\%$ の確率で観測事実が再現する。そのうち $50\% \times 25\%$ のケースが青である。つまりこの観測によってこのコマに対する信念は青50% 赤50%から青20% 赤80%へと更新される。

ゴールインの観測は確率が極端なので、論理で考えている人でも同様に「このコマは赤だな」と推論できただろう。実際には「コマの交換を迫ったら逃げた」のような、青であるとも確定しないが、無情報とも言い切れない、曖昧な観測が観測される。この観測から少しずつ相手の秘密情報をかすめ取り、有利な立場を築いていくのがガイスターというゲームである。

3.4 部分観測モンテカルロ計画法

ガイスターは部分観測マルコフ決定過程であるだけでなく、状態遷移確率が明示的に与えられていない厄介な問題である。そういう状況で使えるのが部分観測モンテカルロ計画法(POMCP)だ。これは状態遷移確率の代わりに、繰り返し実行できるブラックボックスシミュレータを与え、モンテカルロで問題を解く。

部分観測モンテカルロ計画法はパーティクルフィルタ(またの名を逐次モンテカルロ)とモンテカルロ木探索の組み合わせである。パーティクルフィルタによって信念分布を更新する。具体的には信念分布から状態をサンプリングし、その状態だと仮定してシミュレータに相手手番を1手進めさせる。その相手の着手が実際の相手の着手と一致したものだけ残して残りのサンプルを捨てると、結果として相手の手の観測によって更新された新しい信念分布が手に入る。

モンテカルロ木探索部分は、信念分布からサンプリングされた状態を仮定し、その状態から適当なRollout Policyに従って手を選び対戦することで、どの手の勝率が高いかの情報を得る。この情報を木構造でためていき、ある程度情報が集まっている場合には別のTree Policy(有

名なのはUCB1)で手を選択する。

今回のコンテストで筆者が実装したのはこの部分観測モンテカルロ計画法である。後述するParticle Reinvigorationは時間が足りず実装していない。またシミュレータの実装の中には、相手方AIのモデルが必要である。このモデルが「とにかく青でゴールインを目指す」というアルゴリズムFastestであるバージョン(POMCP-Fastest)と、西尾の感性にもとづいてif文を組み合わせ職人技で行動価値関数をくみ上げた、一手も先読みしないアルゴリズムIchiであるバージョン(POMCP-Ichi)とで出場した。

3.5 対戦結果と考察

筆者は残念ながら、大会が事前申し込みの必要なワークショップの一部として開催されることを理解しておらず、気づいたときには参加申し込みが締め切られていたため、プログラムだけを送って対戦して頂いた。全AIの総当たり戦を行い、筆者のPOMCP-Ichiと、川上直人氏のなおっちMINMAXが、他の参加者に対して全勝、互いの対戦で1勝1敗で、同点となった。そのため優勝者決定のためにこの2つのAIで先手後手交代して2回戦うプレーオフが行なわれ、POMCP-Ichiが2敗し準優勝となった。

名前から推察するになおっちMINMAXはMinMax法を使うのであろう。MinMax法は「最悪ケースが一番マシな手を選ぶ」という方法であり、とてもリスク回避的で慎重な性格である。一方で、筆者のPOMCPは手抜きでParticle Reinvigorationを実装していない。考えてみよう。パーティクルフィルタでは信念状態からサンプリングし、その一部を棄却することで信念を更新するのだった。たとえ棄却部分に差がなかったとしても、サンプリングの運で信念の強さがランダムウォークする。そして運悪くパーティクルが0になると、その信念は二度とサンプリングされない。「きっとXじゃないに違いない」と思い込むと、二度とXである可能性を検討しない、思い込みの強い性格なのである。これを緩和するのがParticle Reinvigorationである。

実はこれは既知の問題であった。時間の関係でただ一度だけ行なわれたユーザーテストにおいてPOMCP-Fastestは対戦者である竹内郁雄氏を青赤1個ずつの二択の賭けにまで追い込んだが、ゴール前で両者のコマがにらみ合っているうちに「進んでこないこのコマはきっと赤に違いない」と思い込んでゴール前から離れあっさり負けた。それがFastestよりまともなIchiを大急ぎで用意した理由でもあった。

3.6 まとめ

ガイスターのAIを考える上での有用な概念である「信念」と、それを推定する有用なアルゴリズムである部分観測モンテカルロ計画法について解説した。この情報によって今後のガイスターAI大会がより面白くなることを期待している。

筆者は次回はParticle Reinvigorationを実装した上でPOMCP-MinMaxで出場しようかと考えている。また有益なAIは可能なAI全体の空間のうちのごく小さな部分空間を占めるだろうと予測しており、赤コマの初期配置を推定するのと同様に、対戦相手のアルゴリズムを推定することも可能だろうと考えている。人間が対戦相手である場合には、思考時間から情報を盗み取るサイドチャネル攻撃が有効であると、筆者自身の対人戦経験から確信している。

また現在のPOMCPは、自分の行動からの情報リークを減らそうという最適化は入っていない。今回この記事を公開したことで対戦相手にPOMCPが来る確率が上がったので、それに対策するPOMCP-POMCPを作る手もある。そしてPOMCP-(POMCP-POMCP)も。これを無限に繰り返すとたぶんナッシュ均衡にたどり着くのであろう。