社団法人 人工知能学会
Japanese Society for
Artificial Intelligence

人工知能学会研究会資料
JSAI Technical Report
SIG-Challenge-B101-6 (5/4)

アクション連鎖探索によるオンライン戦術プランニング

Online Tactics Planning using Action Sequence Search

秋山英久
Hidehisa AKIYAMA
福岡大学工学部
Faculty of Engineering, Fukuoka University
akym@fukuoka-u.ac.jp

Abstract

In this paper, we propose an framework to search action sequences in order to enable online tactics planning in multiagent systems. It was difficult to apply a search tree methodology to tasks that the space is continuous and requires realtimeness. However, it has become possible to apply such an approach since the computational resources became more powerful today. We applied a search tree method to the RoboCup soccer simulation environment and analyzed its effectiveness by evaluating the team's performance.

1 はじめに

サッカーのようにチームで対戦するゲームでは，複数のプレイヤが協調して連携動作する戦術的な振る舞いを実現しなければ，チームとしてのパフォーマンスを向上させることは難しい．戦術的な振る舞いを実現するには，ある目標状態に向けて複数のプレイヤのアクションの連鎖をプランニングする必要がある．プランニングの実行には，適切なアクションの列をオンラインで探索しなければならない．しかしながら，空間が連続なだけでなく意思決定に実時間性が求められるタスクでは，ゲーム木探索のような計算機の能力を駆使するアプローチの適用は困難であった．この問題に対して，近年の計算機の能力向上に伴い，従来のような探索に基づいたプランニングを実時間性を損なうこと無く実行することが可能となってきた．

本稿では，連続空間における複数エージェントによる戦術的振る舞いをオンラインで探索するアプローチを取り，探索による戦術プランニングの実現を試みた．さらに，オンライン戦術プランニングの有効性を示すために，RoboCup サッカー 2D シミュレータを用いた評価実験を行う．実験では，オンライン戦術プランニングを実装したチームを用いて実際に試合を行い，探索の導入によるチームパフォーマンスの変化を測る．

2 従来研究

RoboCup サッカー 2D シミュレーション[1]はマルチエージェントシステムのテストベッドとして知られており，2D サッカーシミュレータは，さまざまな機械学習手法の適用だけでなく，戦略，戦術の枠組みに関する研究のためのプラットフォームとしても利用されている．

従来の取り組みでは，特に強化学習研究に関連して，敵対するエージェントが存在する中でプレイヤエージェントの協調的な意思決定あるいは最適な個体制御を獲得させる研究が多く進められている．Stone らは強化学習のテストベッドとして Keepaway というサッカーのサブタスクを提案している[5]．また，Gabel らは敵エージェントの行動を妨害するタスクを設定し，実用的な性能を持った個体制御を強化学習によって獲得することに成功している[7]．強化学習の枠組みにおいては，エージェントが意思決定する機会において最良の行動選択を行う能力の獲得が期待できる．しかしながら，強化学習のようなボトムアップのアプローチでは，目標となる状態を複数のエージェントで共有した上での，戦略· 戦術的な意思決定能力の獲得は想定されていない．

戦術や戦略に基づいた集団制御を実現するために，2D サッカーシミュレーションにおいてはトップダウンなアプローチが取られている．トップダウンなアプローチとして，Situation Based Strategic Position[3]や Locker Room Agreement[2]やなどの，エージェント間で戦略や戦術を事前知識として共有しておく手法が効果的であることが知られている．Situation Based Strategic Position は，戦略のデザインをチームのフォーメーションとして捉え，状況に応じたエージェントの配置を事前に決めておくアプローチである．Locker Room Agreement では，特定の条

件下で実行すべき戦術を事前知識として静的に共有しておくことで，環境の状態が条件に合致した際に固定的なプランが遂行される．これらの手法によって，事前の知識共有に基づいた複数エージェントの協調的な振る舞いを見かけ上は実現できるものの，実行されるアクション連鎖や目標状態が固定的であるために柔軟性に乏しいといった問題がある．

トップダウンなアプローチとしては，環境を俯瞰して観察できるコーチエージェントを用意し，部分情報しか観測できないプレイヤエージェントへコーチエージェントからアドバイスを与えるという取り組みもなされている。サッカーにおける戦略･戦術記述言語として，Reis らによる Coach Unilang が提案されている [4]．Coach Unilang は後に 2D サッカーシミュレータの公式コーチ言語に仕様の一部が採用されている．しかしながら，コーチエージェントを利用した戦略や戦術のアドバイスが有効に機能している事例はまだ十分に報告されていない．これは，コーチエージェントからのアドバイスが適切であったとしても，それを反映する能力をプレイヤエージェントが持たないためであろう．これは，プレイヤエージェントの意思決定におけるプランニング能力に十分な柔軟性を持たせられていないことが原因である．従来のアプローチの問題を解決するには，オンラインでの戦術プランニングを実現し，動的な環境へより柔軟に対応する必要がある．

3　オンライン戦術プランニング

本稿では，従来はあまり扱われてこなかった連続空間における実時間オンライン戦術プランニングを実現するために，アクション連鎖探索フレームワークを提案する．提案するフレームワークでは，探索木によって有効なアクションの連鎖を探索することで戦術のプランニングを実現する．

3.1　アクション連鎖探索フレームワーク

提案するフレームワークは，自分と他者を含めた複数のエージェントによって実行されうるアクション（パス，ドリブル，シュートなど）を生成し，探索木にノードとして格納していくことで有効なアクション連鎖の探索を実行する．図 1 にアクション連鎖のイメージ図を示す．この図では，10 番のプレイヤによって発見されたボールを扱う 4 つのアクションの連鎖を表している．

提案するフレームワークには，アクションの列とそれらアクションを実行後の状態を生成，評価するメカニズムが用意されている．このメカニズムは，以下の 3 つのクラスによって実現される．

- CooperativeAction クラス
- ActionGenerator クラス

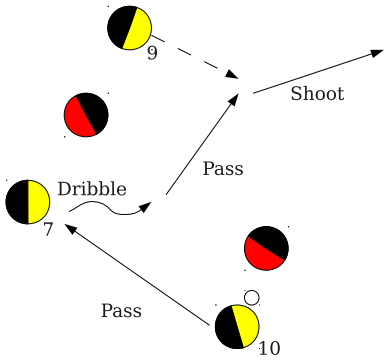

Figure 1: アクション連鎖のイメージ図．10 番から 7 番へのパス，7 番によるドリブル，7 番から 9 番へのパス，9 番によるシュート，というアクション連鎖を表す．

- FieldEvaluator クラス

これらクラスは，オブジェクト指向における抽象クラスを意味する．フレームワークを利用する場合，これらクラスから派生した具象クラスを，要求されるタスクに応じて実装しなければならない．

3.1.1　CooperativeAction クラス

CooperativeAction クラスは，探索実行時に最小単位となるアクションを表す抽象クラスである．サッカーの場合であれば，パス，ドリブル，シュートなどの，サッカープレイヤとしてある程度意味のある行動を具象クラスとして実装することになる．

このクラスは，アクションのタイプ，目標状態，目標状態に到達するまでに要する時間などの情報を保持し，エージェントが参照することができる．エージェントは，これらの情報を参照することで最終的な自身の体の制御を行う．

3.1.2　ActionGenerator クラス

ActionGenerator は，エージェントが観測あるいは予測した環境の状態を入力とし，その状態において取りうる *CooperativeAction* を生成する．通常，ある入力状態に対して複数の *CooperativeAction* の候補を生成することができる．例えば，ボールを所有するプレイヤがパスのアクションを実行しようとする場合，無数のパスコース候補が存在する．

成功と予測される *CooperativeAction* が生成されると，その結果の予測状態が同時に生成され，*ActionStatePair* が作られる．*ActionStatePair* は，*CooperativeAction* と予測状態とを単純に組にしたものである．

ActionGenerator は，ノードとして既に探索木に入れられている *ActionStatePair* を入力として，再帰的に *ActionStatePair* を生成することができる．首尾良く生成された *ActionStatePair* は新しいノードとして探索木に加え

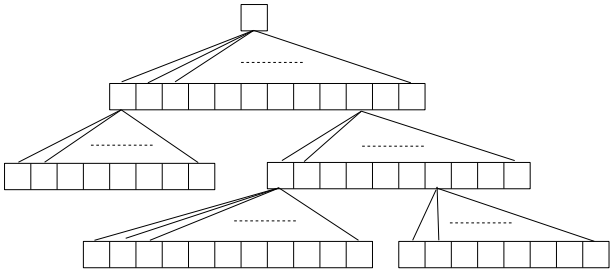

Figure 2: フレームワークで採用する探索木の模式図．多分探索木を採用する．各セルは *ActionStatePair* のノードを表す．各ノードが持つ予測状態が子ノードへの入力となる．

られる．このとき，新規に生成されたノードは，入力となった親ノードの子ノードとなる．

3.1.3 FieldEvaluator クラス

FieldEvaluator は，新規作成されたノードからルートノードへ繋がるアクション連鎖全体を入力とし，生成された *ActionStatePair* を評価する．*FieldEvaluator* は評価値として実数を返し，各ノードはこの値を保持する．探索木の走査終了後，プレイヤエージェントは，得られた評価値に基づいて，生成された葉ノードの中からもっとも高評価のものを選択する．葉ノードが選択されると，ルートノードまでたどることによって現在状態からのアクション連鎖が得られる．

3.2 探索木の構造と走査アルゴリズム

提案するフレームワークで採用する探索木の模式図を図 2 に示す．図で示すように，フレームワークでは多分探索木によって探索処理を実行する機能が提供されている．図中のセルは木のノードを表し，それぞれ一つの *ActionStatePair* を格納する．木のルートノードから葉ノードまでのノード列をつなげると，あるアクション連鎖が得られる．

探索木の走査アルゴリズムとして，本稿では単純な最良優先探索を用いる．*ActionStatePair* によるノードが生成される際に，*FieldEvaluator* によってノードの評価値が得られる．この評価値は，最良優先探索のためのヒューリスティック値としても利用される．新規ノードが追加されるごとに，評価値に基づいてノードを格納する優先順位付きキューが更新される．ノードの走査は優先順位付きキューでの格納順に実行される．

探索木の走査では，葉ノードに到達した時点で探索の終了条件を満たしているかどうかを確認する．本稿では，以下を終了条件とする．

- 走査した全ノード数があらかじめ設定した最大数を越えた．

また，葉ノードにおいて以下の条件のいずれかを満たす場合は，その葉ノードでの新規子ノードの生成は行われない．

- 深さがあらかじめ設定した木の深さの最大数を越えた．
- 入力された予測状態から *CooperativeAction* を生成できなかった．
- アクション連鎖の終了と設定されている *CooperativeAction*(例: シュート) が生成された．

4 評価実験

実験環境として，RoboCup サッカー 2D シミュレータ[1] を用いる．評価用の実行チームとして RoboCup2010 で優勝した HELIOS2010 を用い，探索木の設定変更によるチームのパフォーマンス変化を調査する．

4.1 実装

本稿では，プレイヤエージェントがボールを蹴るアクションの連鎖のみを扱う．実験で動作させるプレイヤエージェントには，以下のような *CooperativeAction* の具象クラスを実装した．

- Clear：可能な限り敵プレイヤに取られない方向または位置へボールを蹴り出すアクション．
- Dribble：ボールを蹴った後に，再び自分がボール所有者になるアクション．
- Pass：ボールを蹴った後に，自分以外がボール所有者になるアクション．
- Shoot：敵ゴールへボールを蹴り入れるアクション．

上記アクションクラスに対して，以下のような *ActionGenerator* の具象クラスを実装した．ただし，計算量の問題から，探索木におけるルートノードとそれ以外のノードとで使用する *ActionGenerator* を切り替えることとする．以下のリストのうち，†がついたものはルートノードでの *CooperativeAction* 生成に使用されるもの，‡のついたものはルートノード以外で使用されるものを意味する．

- Shoot‡：簡易予測計算を行うシュート
- Cross†：厳密な予測計算を行うパス（ 敵ゴール前）
- StrictCheckPass†：厳密な予測計算を行うパス
- DirectPass‡：簡易予測計算を行うパス（ 味方の足元を目標位置とするパス）
- VoronoiPass‡：簡易予測計算を行うパス（ 敵の配置に基づくボロノイ図を利用して目標位置をあらかじめ制限するパス）

[1] http://sourceforge.net/projects/sserver/

Figure 3: 探索によって実際に発見されたアクション連鎖の例．9 番から 10 番へのパス，10 番から 6 番へのパス，6 番によるシュートまでのプランが 9 番のプレイヤによって生成されている．図中の緑色または青色の円は，9 番のプレイヤよって観測された他のプレイヤの位置を表す．

- ShortDribble†：厳密な予測計算を行うドリブル
- SelfPass†：厳密な予測計算を行うドリブル
- SimpleDribble‡：簡易予測計算を行うドリブル

*FieldEvaluator*に関しては，いくつかのルールを人手で作り込んだものを用意し，すべてのプレイヤエージェントで共通に使用する．実装した *FieldEvaluator* は評価関数として必ずしも最適化されていない．ただし，RoboCup2010 で優勝した HELIOS2010 が使用したものと同じ設定であるため，現在の競技レベルで実用できる性能は保証されている．

以上の実装に基づいて，実際にオンラインで利用可能となるように探索木の設定を調整する．HELIOS2010 では，探索木の深さの最大数を 4，ノード走査の最大数を 500 としたものを実用した．この設定のもとで，図 3 に示すようなアクション連鎖をオンラインで発見できることが確認されている．図中で示されている例では，アクションの連鎖は 3 段階となっている．このように，*FieldEvaluator* が返す評価値によっては，探索木の深さの最大数よりも小さい深さで発見されたアクション連鎖が最良の結果として使用される．

4.2 実験設定

探索木の設定変更によるチームのパフォーマンス変化を調査するために，以下の 2 項目の全組み合わせで試合を 10 試合ずつ行う．

- 探索木の深さの最大数：1, 2, 3, 4, 5, 6, 7
- ノード走査の最大数：10, 100, 500, 1000, 10000

ただし，木の深さやノード走査の最大数が極端に大きい場合，実時間での計算が間に合わない．そこで，今回の実験では，2D サッカーシミュレータに用意されている同期モードを用いることで，計算時間を無視する．

対戦相手には，agent2d-3.0.0[2] [6]を用いる．チームのパフォーマンスの指標として，一試合中のパスの成功回数と得点を用いる．

4.3 実験結果

表 2 にパス成功回数の結果を，表 1 に得点の結果を示す．値はすべて 10 試合の平均値である．

パスの成功回数に関しては，ノードの走査数が 10 と極端に少ない場合に増加していることが分かる．これ以外の組み合わせでは，木の深さの最大数が 2 から 3 の間でパスの成功回数が増加している．

得点に関しては，ノード走査の最大数がより大きく，また，木の深さの最大数がより大きい方が得点が増えていることが伺える．この結果をグラフ化したものを図 4 に示す．グラフからも，ノード走査の最大数が 10 の場合を除いて，木の深さの最大数が大きくなるにつれて得点数が増える傾向にあることが分かる．また，ノード走査の最大数が大きくなると，深さの最大数の変化の影響が小さくなっている．

4.4 考察

走査するノードの最大数が極端に小さい場合，ほぼすべての状況において深さ 1 の探索となってしまうため，生成される *CooperativeAction* の順序によってチームの特徴が決定してしまう．今回の実験では，パスアクションが最初に生成されるように実装されていたため，ノード走査の最大数が 10 の場合にはほとんどパスしか生成されず，パスの成功回数が増えるという結果になった．

一方で，走査するノードの最大数が一定の値を越えると，チームのパフォーマンスが安定してくることが分かる．これは，より多くのノードを走査することで，局所解に陥りにくくなるためであると予想される．

より深い探索によってチームのパフォーマンスが向上する可能性が示されたが，そのためには，走査するノードの最大数を一定数確保しなければ探索によるパフォーマンスの改善を得られにくくなると言えるだろう．ノード走査の最大数の最適な値は，*ActionGenerator* によって生成されうる *CooperativeAction* の数，また，探索の深さの最大数によっても異なる．さらに，探索木の設定は実時間での実行可能性も考慮して決定しなければならないため，エージェントを実行する計算機の性能によっても最適な設定は異なると予想される．さまざまな環境下でより多くの試合を実行し，パフォーマンスを測定することで探索木の設定を最適化することが必要である．

[2] http://sourceforge.jp/projects/rctools/ より入手可能

Table 1: 得失点差(10 試合の平均) .

最大ノード 数＼最大深さ	1	2	3	4	5	6	7
10	1.5	2.3	1.7	2	2.1	2.3	1.3
100	2.9	2.8	1.9	2.4	1.9	3.2	3
500	2.6	3	3.4	1.5	2.9	2.4	2.7
1000	2.6	2.8	2.7	3	3.5	3.5	3.4
10000	2.2	2.7	2.7	3	2.7	3.1	3.6

Table 2: パス成功回数(10 試合の平均) .

最大ノード 数＼最大深さ	1	2	3	4	5	6	7
10	205.9	219	212	210.4	219.3	215.3	219
100	154.1	175.1	159.2	157.7	158.8	156.4	166.3
500	145.1	200.1	200.2	168.2	160.8	154.9	154.1
1000	171.7	193.2	210.3	179.3	168.5	163.5	166.9
10000	145.6	196.8	229.1	198.3	182.6	167.3	163.6

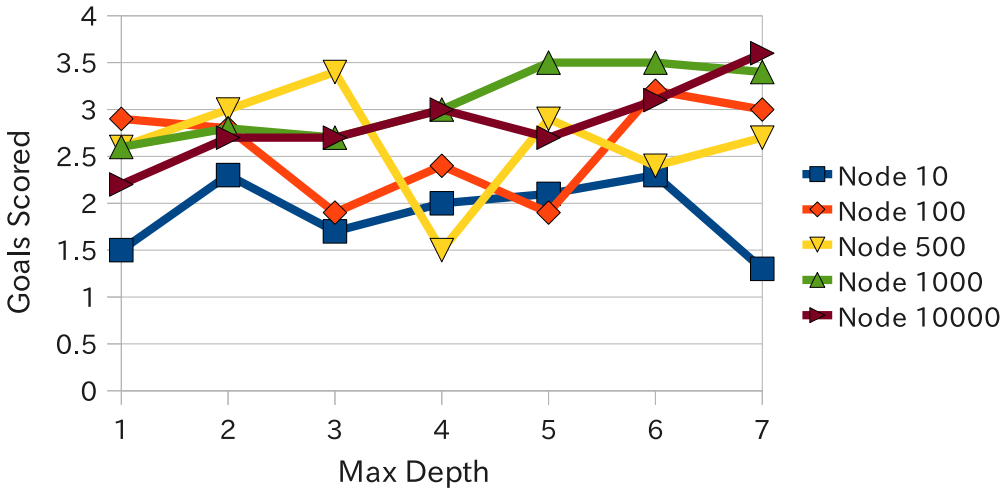

Figure 4: 探索木の設定変更によるチームパフォーマンスの変化．ノード 走査の最大数ごとにグラフ化している．横軸: 探索木の深さの最大数．縦軸:10 試合の得点数の平均値．

5 まとめと今後の課題

本稿では，探索木によりアクション連鎖を探索するフレームワークを提案し，これを用いることでオンラインでの戦術プランニングの実現を試みた．RoboCup サッカー 2D シミュレーション環境において，フレームワークを実装したエージェントを用い，探索木の深さと走査するノード数の変更によってチームのパフォーマンスに現れる影響を測った．

今後は，探索木の設定を最適化するためのデータ収集や，より多様な評価基準に基づいてチームの戦術的な振る舞いに現れる変化を分析することが必要である．また，本稿で提案するフレームワークで生成される行動連鎖はすべて直列であり，並列に実行されるアクションの集合を生成することはできない．そのため，ボールを持たないプレイヤエージェントの移動動作まで含めたアクション連鎖を扱えていない．並列に進むアクション連鎖を実現できるようにフレームワークを拡張することが今後の重要な課題である．

参考文献

[1] Itsuki Noda and Hitoshi Matsubara: Soccer Server and Researches on Multi-Agent Systems, Proc. of IROS-96 Workshop on RoboCup, pp. 1-7, (1996)

[2] Peter Stone and Manuela Veloso: Task Decomposition, Dynamic Role Assignment, and Low-Bandwidth Communication for Real-Time Strategic Teamwork, Artificial Intelligence, 110(2), pp.241-273, (1999)

[3] Luis Paulo Reis, Nuno Lau and Eugenio C. Oliveira: Situation Based Strategic Positioning for Coordinating a Team of Homogeneous Agents in Markus Hannebauer, Jan Wendler and Enrico Pagello Editors, Balancing Reactivity and Social Deliberation in Multi-Agent System From RoboCup to RealWorld Applications, Springer LNAI, Vol. 2103, pp. 175-197, (2001)

[4] Luis Paulo Reis and Nuno Lau: COACH UNILANG - A Standard Language for Coaching a (Robo) Soccer Team, RoboCup-2001: Robot Soccer World Cup V, Springer Verlag LNAI, Vol. 2377, pp. 183-192, Berlin, (2002)

[5] Peter Stone, Richard S. Sutton and Gregory Kuhlmann: Reinforcement Learning for RoboCup-Soccer Keepaway, Adaptive Behavior, 13(3), pp. 165-188, (2005)

[6] 秋山英久: ロボカップサッカーシミュレーション 2D リーグ必勝ガイド , 秀和システム, (2006)

[7] Thomas Gabel, Martin Riedmiller and Florian Trost: A Case Study on Improving Defense Behavior in Soccer Simulation 2D: The NeuroHassle Approach. RobopCup 2008: Robot Soccer World Cup XII. pp. 61–72, (2008).