1L1-R-7-6

マルチエージェントシステムにおける 実時間タスクプランニングに関する研究

Research on Real Time Task Planning for Multi-Agent System

伊藤 友貴[*1] 赤石 美奈[*2] 堀 浩一[*1]
Tomotaka ITO Mina AKAISHI Koichi HORI

[*1]東京大学大学院工学系研究科
School of Engineering, The University of Tokyo

[*2]法政大学情報科学部
Faculty of Computer and Information Sciences, Hosei University

In this study, we focus on real time task planning as a method to improve the performance of multi-agent systems. Real time task planning is online planning which modifies tasks to suit the changing environment from time to time. So, multi-agent system which consists of agents performing real time task planning is expected to be applied to real multi-robot systems, such as rescue robots for disaster and planetary exploration rovers. For contributing to realization of these systems, we aim to build the generic methodology to construct the algorithm of real time task planning in multi-agent system.

As a first step, we solve a distributed constraint optimization problem to which the parameter of time is added. In order to verify the utility of real time task planning in multiple agents, we apply the solution which combined the algorithm of real time task planning and DSA to the problem.

1. 研究背景

1.1 実時間タスクプランニング

タスクプランニングとは, あるミッションにおいて, 目標の達成に必要なタスクの順番や行動の結果について予測し, タスク列を導出する推論技術である. 近年, ロボットや人工エージェント等の自律システム実現のために, タスクプランニングは主に人工知能の分野で盛んに研究されている. 特に, 現実に存在する問題を扱うことを目指し, 事前にオフラインで問題を解くのではなく, 時々刻々と変化する環境に合わせてタスクを組み替えていくオンラインの実時間プランニングが注目されてきている. このような技術は, 災害救助用ロボットや惑星探査自律型ロボットなどへの応用が期待される.

実時間タスクプランニングに関しては, 長らく個々の問題に特化した方法論を議論した研究が多くなされてきたが,2006 年に船瀬が「計算時間の扱い方」に関する統一的方法論の提案 [船瀬 06] を行っている. これは, 個々の問題に対するタスクプランニングのアルゴリズム設計指針を示しており, それまでより広い範囲の問題を扱うことのできるものであった. しかし, この手法で用いられる「解の効用」や「予測モデル」を具体的に構築するには, さらに個々の問題に依存した議論をしなければならない. 現実での応用を目指すためには, 個々の問題の性質の深い分析やさらなる洗練された方法論の展開が求められる.

1.2 マルチエージェントシステム

エージェントとは, 受容器を用いて環境を知覚し, 効果器を通して自分の意志により行動するものである. 知能ロボットだけではなく, ソフトウェアの概念, 動物や人などがエージェントに分類されることもある. このようなエージェントが複数存在し, 互いに相互作用を及ぼしながら全体が機能を持つシステムがマルチエージェントシステムである. このシステムの利点には次のようなものがある.

- 問題解決能力

  1 体のエージェントでは実行不可能なことを, 複数のエージェントが集まることによって効率的に実現できる可能性がある.

- 適応能力

  新たにエージェントを追加したり, エージェント間の相互作用を調節することで, 対処できる問題の規模を拡大できる可能性がある.

- ロバスト性

  あるエージェントが故障等で使えなくなっても, 他のエージェントがその役割を代わることで, システム全体の機能への影響を小さくすることができる.

このような様々なメリットを持つシステムとしてマルチエージェントシステムは注目され, 近年盛んに研究されている.

上記した様々な利点を効果的に活かすためには, システム全体の能力を向上させるような何らかの機能を個々のエージェントが持つ必要がある. その 1 つの方法として, よりよい行動を判断し選択する知的能力を獲得させるものがある. 個々のエージェントを賢く動かすことにより, システム全体の性能を向上させようとする方法である. ただし, システムを適用する問題によって必要な知的能力は異なるので, 問題に応じた適切な機能を設定する必要がある.

2. 研究目的

マルチエージェントシステムの応用先として期待される多くの問題では, ある時間制約が課されている状況下で, ミッションを達成することが求められる. 例えば, 大規模自然災害時の救助エージェントロボット群は, 生命の危機が増大する前になるべく早く被災者を救助する必要があり, どのような救助行動を行うかを迅速に判断することが重要になる. また, 惑星探査をミッションとするようなローバー群は, 故障してしまう可能性が高い厳しい環境下で活動し, 限られたエネルギーを用いて

連絡先: 伊藤 友貴
東京大学大学院工学系研究科航空宇宙工学専攻
〒 113-0033 東京都文京区本郷 7-3-1
t-ito@ailab.t.u-tokyo.ac.jp

動くため, なるべく時間をかけずに地図作製などのミッションを達成する必要がある.

これらのような時間制約が厳しいミッションにおいては, 環境や内部状態の変化に合わせて時々刻々とタスクを組み替えていく実時間タスクプランニングの適用が効果的だと考えられる. そこで本研究は, マルチエージェントシステムを構成する個々のエージェントに実時間タスクプランニングの能力を獲得させることによって, システム全体の機能を向上させることを目的とする. また, 実時間タスクプランニングの議論をさらに深め, マルチエージェントシステムにおける実時間タスクプランニングのアルゴリズムを構築するための統一的方法論の提案を目指す.

3. 実時間タスクプランニングの統一的手法

本研究では, 船瀬が提案した実時間タスクプランニングの統一的方法論 [船瀬 06] を出発点とし, マルチエージェントシステムへの拡張を目指す. ここでは, 前提となる統一的手法について述べる.

3.1 解の効用

実時間タスクプランニングでは,「ある解（行動）をある状況で実行した結果, どの程度目標を達成すると期待されるか」を示す尺度が必要になってくる. 具体的には, 探索時間を費やして得たある解 r をある状況 s, 時刻 t で実行した結果の良さを評価する量として, 解の効用 U を定義する. ここで, 効用 U は解 r, 状況 s, 時刻 t の関数である.

図 1 は, その時刻における解の効用 U の最大値が時間とともにどのように変化していくかを表している. 図から読み取れるように, 効用の最大値は始め時間とともに増加し, ある時刻でピークを迎えてその後減少していく. このように変化するのは, 解を得ようと探索時間を費やすことにメリットとデメリットの両方の側面があるからである.

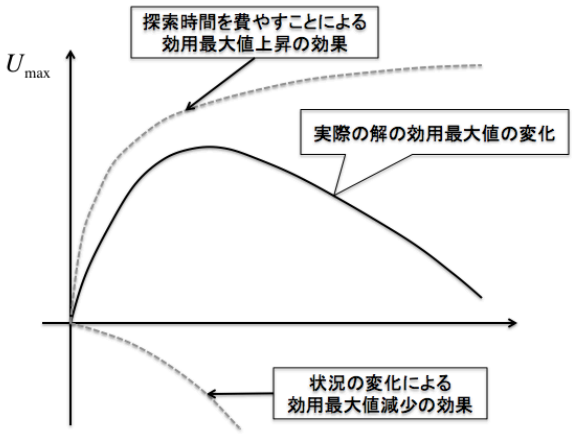

図 1: 解の効用のイメージ

- 解探索時間のメリット

  解を探索する時間を費やすことのメリットは,「時間をかけて探索すればするほどよりよい解を見つけ出すことができる」ということから発生している. 探索時間を長くとれば多くのタスクについて考慮することができ, よりよい結果を期待できる解を発見する可能性が大きくなるのである.

- 解探索時間のデメリット

  解を探索する時間を費やすことのデメリットは,「時間の経過とともに周りの状況が変化する」ということから発生している. 探索時間を長くかけてよい解を見つけ出したとしても, 実行に移そうとする時には既にその解は最良でなくなっていたり, そもそも実行不可能な状況になってしまっていたりする. よってオンラインのタスクプランニングでは, 無尽蔵に探索時間を費やすわけにはいかず, ある時点でそれまでに得られた最良の解を実行に移す判断をする必要がある.

このように探索時間のメリットとデメリットにはトレードオフの関係があるため, 適当な探索時間を選択するためにはそのどちらの影響も考慮した概念が必要になってくる. よって, ある状況で最良解を実行した結果を定量的に評価する量として「解の効用」を定義し, これを用いて実際のプランニングを行っていくことになる.

3.2 解の効用を用いたプランニングの概要

実時間タスクプランニングの肝は,「どの時点で探索をやめ, タスクを行動に移すか」を判断することである. その判断基準として解の効用を用いる.

時刻 t に得られた解 r_t を状況 s_t において実行した場合の効用 $U(r_t, s_t, t)$ と, さらに探索を行って時刻 $t' = t + \Delta t$ において解 $r_{t'}$ を状況 $s_{t'}$ において実行したときの効用 $U(r_{t'}, s_{t'}, t')$ の差を考え,

$$U(r_{t'}, s_{t'}, t') - U(r_t, s_t, t) > 0$$

であれば探索を継続し, そうでなければ探索を中止して解を実行する. この Δt を小さくすることで, 効用が極大となる時刻において最良と期待される解を実行することになる.

ここで, 時刻 t' での解 $r_{t'}$ や状況 $s_{t'}$ は未来の値なので, 予測をする必要がある. ただし, 時刻 t の時点で解 r_t は求められている最良の解であり, それ以上良いと思われる解をプラナは把握していない. よって実際は解そのものは予測せず, 現在の解 r_t, 状況 s_t, 時刻 t と予測が可能な状況 $s_{t'}$ をもとに効用を予測させることを考える.

そもそも時刻 t における状況 s_t や効用は, 取得することができる限られた情報から推測することにより得られるものであり,「どういう状況である確率が高いか」というような確率分布の形で考えることが多い. 値を決定する必要がある場合は, 期待値を用いる.Δt 後の状況 $s_{t'}$ や効用についても, 事前情報や取得情報の履歴等から確率分布や期待値を求めていくことになる.

3.3 探索時間制御アルゴリズム

具体的な探索時間制御アルゴリズムを構築するに当たって, 以下の関数を定義する.

- GetPossibleComputation 関数

  時刻 t, 状況 s_t, その時に得られている解 r_t を引数に取り, その状況における Possible Computational Action C_p とそのサイズ n_p を出力として返す関数.

- EstimateUtility 関数

  現時点での解, 状況, 時刻をもとに, 探索行動 C_{p_i} を実行した場合に得られる解の効用を予測する関数. これをいかに実装するかが, アルゴリズムの核となる.

- ExecuteComputation 関数

  選ばれた探索行動を実行する関数.U_{best} が U_{now} を超える時に限り実行する. 実行結果に基づいて解, 状況, 時刻を更新する.

- ExecuteSolution 関数

  探索をストップした時点での解 r_t を実行する関数.

ここで,Possible Computational Action とは, ある時刻 t, 状況 s_t において解 r_t が得られているときの取り得る解探索行動の集合であり, 以下のように表す.

$$C_p(r_t, s_t, t) = \{C_{p_1}, \cdots, C_{p_i}, \cdots\}; \quad 1 \leq i \leq n_p$$

探索時間を制御するメタ推論においては, 解探索を行うこと自体を行動と見なし, 上のように定義している.

これらの関数や記号を用いて, 解探索時間制御アルゴリズムは図 2 のように定式化することができる.

```
r ← φ //「何もしない」という初期解
  s ← s_0
  t ← 0
  while true do
    [C_p,n_p] ← GetPossibleComputation(r,s,t)
    U_now ← U(r,s,t)

    U_best ← −∞
    for i=1 to n_p do
      t_i ← C_{p_i}.t
      s_i ← s_{t_i}
      U'_i ← EstimateUtility(C_{p_i},r,s_i,t_i)
      if U_best ≤ U'_i then
        i_best ← i
        U_best ← U'_i
      end if
    end for

    if U_best > U_now then
      (r,s,t) ← ExecuteComputation(C_{p_{i_best}},r,s,t)
    else
      break;
    end if
  end while

  ExecuteSolution(r)
```

図 2: 解探索時間制御アルゴリズム

4. 考慮する問題と手法の適用

本研究では, 実時間タスクプランニングの理論の妥当性を確認するため, 分散制約最適化問題 (Distributed Constraint Optimization Problem) に時間に関するパラメータを追加し, 本手法を適用することにした.

4.1 分散制約最適化問題

分散制約最適化問題とは, マルチエージェントシステムにおける協調問題解決の基本的な枠組みである. この問題では, 各エージェントは一部の他のエージェントと制約で繋がっており, 自身の内部変数とその値を決定する権限をもつ. 各エージェントは制約で繋がったエージェントと情報交換を行いながら, 利得の総和を最大化するように変数への割当てを決定する.

図 3 に簡単な例を示す. 左は 5 つのノードからなる制約グラフであり, エージェント間のリンクの利得が右の利得表で定義されている. エージェントは利得の和が最大になるようにそれぞれの変数（状態）を決定する.

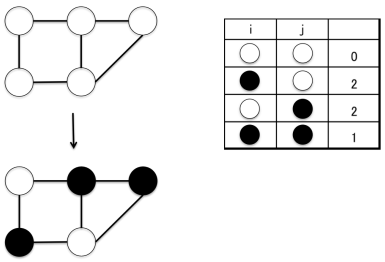

図 3: 分散制約最適化問題

分散制約最適化問題の解法は, 厳密解法と非厳密解法に大別される. 厳密解法は, Pseudo-tree にもとづく ADOPT, ADOPT-ing, DPOP や, 仲介者エージェントを用いる OptAPO などが提案されている. また非厳密解法は, 山登り方や確率的解法にもとづく DSA[Zhang 05], Max-Sum などが提案されている. 本研究は厳密解を求めることが目的ではなく, 時間とともに変化する環境の中でどのように妥当な行動を選択するかに焦点を当てているため, 非厳密解法に実時間タスクプランニングの手法を適用していくことにした.

4.2 問題設定

分散制約最適化問題への時間変化パラメータの導入のしかたとして, 以下のようなものが考えられる.

(1) リンクから得られる利得が時間経過とともに減少する

(2) グラフ構造が時間経過とともに変化する

(3) エージェントの取り得る状態が時間とともに制限される

これらを群ロボットの協調システムに当てはめて考えると,(1) は環境変化によって行動を行った後の結果の良さが変化してしまうことに対応し,(2) はロボットの移動によって通信できる他のロボットが変わることに対応する. また,(3) はロボット自身の劣化や環境変化によってロボットの取り得る行動が制限されることに対応する. 始めは, 利得が減少していく (1) の時間変化のみ導入し, グラフ構造と取り得る状態は変化しない問題を考慮することにした.

またグラフ構造としては, 以下のようなものがよく用いられる.

- グリッド

  格子点上にエージェントを配置し, 縦と横のエージェントがリンクで繋がれているようなグラフ構造.

- ランダムグラフ

  n 個のノード（エージェント）からなる集合における 2 個のノードの組それぞれに対して, その間に確率 p の辺を張ることで生成されるグラフ構造.

まずは比較的エージェント間の制約が均一なグリッド構造を対象に実験を行うことにした.

4.3 探索時間制御アルゴリズムの適用

基本的には, 図 2 の解探索時間制御アルゴリズムの流れをそのまま適用していく. ただし, 各エージェントは Distributed Stochastic search Algorithm (DSA)[Zhang 05] を簡単化したアルゴリズムにより自らの状態を変化させていくこととする. 具体的には, リンクされた四方のエージェントから変数などの情報を得た結果, 自らの状態変化によりリンクの利得和が向上すると予測される場合は, ある確率 p で状態を変化させる. この操作を, 各エージェントが毎時間ステップごとに行う.

今回探索時間制御アルゴリズムを適用するにあたり, 図 2 の各ステップの操作を以下のように対応させることにした.

- GetPossibleComputation 関数

  その状況において,1 ステップ後に取り得ることができるエージェントの状態変数の候補を取得する.

- 状況 s_{t_i} の取得 ($s_i \leftarrow s_{t_i}$)

  四方のエージェントから情報を得, 現時点までの情報の履歴等から解探索後の状況を予測する.

- EstimateUtility 関数

  状態変化後の利得を予測する. 実際に取得できる利得情報は四方のリンクのみであるので, 基本的には四方のリンクの利得の総和を予測することになるが, 問題設定によりそれ意外のリンクの予測も可能であるような場合は, 予測範囲を拡大することも考える. この関数をどのように実装するかが探索時間制御アルゴリズムの核であり, 事前の実験により効用の表れ方の傾向を調べてモデルを構築したり, 効用の確率分布を仮定して推定したりすることになる.

- ExecuteComputation 関数

  選ばれた状態変化をある確率 p で実行する. 実行結果に基づいて状況, 時刻を更新する.

- ExecuteSolution 関数

  状態変化をストップさせ, エージェントの状態を決定する. 一度この関数を実行したら, 再度状態変化は行わない.

全てのエージェントが ExecuteSolution 関数により自らの状態を決定したら, そのステップ（時刻）でのリンクの利得の総和を計算し, 最終的に得られた利得とする. これを, 探索時間制御アルゴリズムを用いない場合の同時刻の利得と比較し, アルゴリズムの評価を行いたいと考えている.

5. 今後の予定

今回は, 実時間タスクプランニングをマルチエージェントシステムに適用することについて議論し, システムの機能向上の可能性を示唆した. また, 探索時間制御アルゴリズムを適用する分散制約最適化問題について述べ, 問題への時間パラメータの導入とアルゴリズムの適用方法について論じた.

今後は, 利得が減少する場合の他, グラフ構造の変化やエージェントの取り得る状態の制限などの様々な条件を導入し, アルゴリズムのシミュレーションを行っていく予定である. 分散制約最適化問題で実時間タスクプランニングの有用性を確認後, 様々な問題に対応できるようなアルゴリズム構築の統一的方法論について議論していきたい.

参考文献

[Zhang 05] Weixiong Zhang, Guandong Wang, Zhao Xing, Lars Wittenburg: ”Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks”, Artificial Intelligence, No.161, pp. 55-87 (2005).

[Balaji 10] P. G. Balaji and D. Srinivasan: ”Multi-Agent System in Urban Traffic Signal Control”, Computation Intelligence Magazine, IEEE Volume:5 Issue:4, pp. 43-51 (2010).

[Russell 09] Stuart Russell and Peter Norvig: ”Artificial Intelligence: A Modern Approach (Third edition)”, Prentice Hall (2009).

[船瀬 06] 船瀬龍: ”解探索の時間管理手法を導入した実時間タスクプランニングに関する研究”, 東京大学大学院工学系研究科航空宇宙工学専攻博士論文 (2006).

[沖本 11] 沖本天太, ジョ ヨンジュン, 岩崎敦, 横尾真: ”疑似木に基づく分散制約最適化問題の制度保証付き非厳密解法の提案”, 第 25 回人工知能学会全国大会 (2011).

[松井 09] 松井俊浩, 松尾啓志: ”非同期分散協調アルゴリズムのためのプログラミング言語に関する一考察”, 第 23 回人工知能学会全国大会 (2009).

[尾田 07] 尾田十八・三木光範 共著, 上田完次 編著: ”創発とマルチエージェントシステム”, 培風館 (2007).