2H3-03

不完全な情報を用いたマルチエージェントシステムによる協調手法

Distributed Cooperation Method for Multi-Agent System by Using Incomplete Information

今福 啓[*1]
Kei Imafuku

[*1]獨協大学経済学部
Dokkyo University, Faculty of Economics

The objective of this paper is to move one system by the plural number of agents without communicating each other. In this work, each agent is assumed to recognize the limited part of the total system. Therefore the agent makes the local model in order to calculate the optimal action that satisfies the given purpose.

In the proposed method, each agent calculates the effect which is caused by the unrecognized part of the total system in real time, and defines the future act by using the real-time optimization method. The effectiveness of the proposed method is shown through the simulation.

1. はじめに

複数のエージェントが互いに通信を行わずに，与えられた目的を満たすために協調して行動することで，ひとつのシステムを稼働させる場合，各エージェントの行動は他のエージェントの挙動に大きな影響をおよぼす．したがって，各エージェントはその影響を予測しつつ，同時に目的を満たす最適な行動を決定しなければ，効率良く目的を達成することは難しい．

本研究では，エージェントが自己の周りの情報のみを得られる状況において，局所的なモデルを構築し，与えられた目的を達成するようにシステムを稼働させるための手法を提案する．提案手法では，各エージェントは他のエージェントからの影響の予測値を求め，その影響を考慮して，与えられた目的を最適とする行動を実時間で計算する．そして，提案手法によって，システムを目標位置まで移動させることができることを，シミュレーションにより示す．

2. 問題設定

本研究では，複数のエージェントが対象となるシステムに行動を加えて，目標位置までシステムを移動させる問題を扱う．その際，各エージェントは以下の条件のもとに行動を決定する．

条件 1 各エージェントは，他のエージェントと通信を行わずに，独立して離散時間 ΔT ごとに行動を決定する．

条件 2 各エージェントには，それぞれ目標値が与えられており，各エージェントがその目標値に到達した際，全システムの状態は唯一となるように設定されているものとする．

条件 3 各エージェントには，それぞれ評価関数が与えられれており，それを最適とするように行動する．

条件 4 各エージェントは，システムの全パラメータを正確に知ることができないものとする．

本論文では，エージェントが解決する具体的な問題として，図1に示す，棒の両端に質点が取り付けられたシステムを，障害物の配置された空間において，障害物に衝突しないように目標値まで移動させる，rod-in-maze 問題を扱う．質点 1,2 の質量をそれぞれ M_1, M_2 とする．X,Y は棒の重心位置，θ は X 軸から反時計回りを正とする姿勢角である．また，質点 1 にエージェント 1 が，質点 2 にエージェント 2 が力を加えて棒を指定された位置に移動させる．1 つのエージェントが集中的に図1のシステムの制御を行う場合，状態方程式は以下のようにあらわされる [中村 99].

$$\ddot{X} = (F_{1X} + F_{2X})/(M_1 + M_2) \quad (1)$$

$$\ddot{Y} = (F_{1Y} + F_{2Y})/(M_1 + M_2) \quad (2)$$

$$\ddot{\theta} = \{(M_2F_{1X} - M_1F_{2X})\sin\theta + (M_1F_{2Y} - M_2F_{1Y}) \cdot\cos\theta\}/\{M_1M_2(l_1 + l_2)\} \quad (3)$$

F_{1X}, F_{1Y} は，質点 1 に加える X 方向，Y 方向の力である．また F_{2X}, F_{2Y} は，質点 2 に加える X 方向，Y 方向の力である．本研究では，各エージェントは，図 1のシステムにおける，すべてのパラメータを知ることができないものとする．各エージェントは，近傍において知ることのできるパラメータを積極的に利用してモデル化をおこない，未知部分についてはモデルに力を加えて移動させつつ推定することで，すばやくモデル全体の挙動を獲得し，与えられた評価関数を最適とする行動を決定して実行する．

なお，本研究では，$F_{1X}, F_{1Y}, F_{2X}, F_{2Y}$ には，以下のような上下限があるものとする．

$$F_{1\,\mathrm{min}} \le F_{1X}, F_{1Y} \le F_{1\,\mathrm{max}} \quad (4)$$

$$F_{2\,\mathrm{min}} \le F_{2X}, F_{2Y} \le F_{2\,\mathrm{max}} \quad (5)$$

図 1のシステムを，エージェント 1 がどのように認識するのかを，図 2 に示す．エージェント 1 は，質点 1 にのみ入力 F_{1X}, F_{1Y} を加え，質点 1 の質量とその位置 X_1, Y_1 のみを正確に知るものとする．また，質点 2 が存在し，それによって重心位置が棒上のどこかに存在することは認識しているが，その正確なパラメータは知らないものとする．以上の点を考慮して図 2 をモデル化すると，次の状態方程式が得られる．エージェント 2 についても，同様にしてモデル化するものとする．

$$\ddot{X_1} = F_{1X}/M_1 + w_1^1 \quad (6)$$

$$\ddot{Y_1} = F_{1Y}/M_1 + w_2^1 \quad (7)$$

$$\ddot{\theta_1} = (F_{1X}\sin\theta_1 - F_{1Y}\cos\theta_1)/(M_1l_1') + w_3^1 \quad (8)$$

連絡先: 今福 啓，獨協大学経済学部，〒 340–0042 草加市学園町 1-1，k03082@dokkyo.ac.jp

$w1^1, w2^1, w3^1$ は，エージェント 1 において，式 (6)-(8) であらわされる状態方程式では表現されない未知パラメータの影響により生じる外乱である．また，式 (8) において，l'_1 はエージェント 1 が重心と考える位置までの距離である．エージェント 1 は，質点 2 の正確な質量を把握していないことから，l'_1 の正しい値を設定できない．そこで，適当な位置を重心位置とみなしたモデルを作成し，事前に知ることのできない部分については，未知パラメータの影響として，システムに入力 F_{1X}, F_{1Y} を加えて稼働させつつ，w_3 として推定する．X_1, Y_1 も，エージェント 2 が加える力の影響を受けるために，同様にシステムを稼働させつつ推定する．

各エージェントは，次のような評価関数を内部に持つ．

$$J^i(x^i, u^i, w^i, t) = J^i_{cost}(x^i, u^i, w^i, N, t) + J^i_{obs}(x^i, u^i, w^i, N, t) \tag{9}$$

$i = 1, 2$ は，エージェントの番号であり，$x^i = [X_i, Y_i, \theta_i], u^i = [F_{iX}, F_{iY}], w^i = [w^i_1, w^i_2, w^i_3]$ である．J_{cost} では x^i の大きさを評価して，システムの目標位置を設定する．J_{obs} では障害物との距離を評価することで，障害物の回避を行うように設定する．N は，どの程度先までのステップ数を評価するのかを決定するパラメータである．各エージェントは，時刻 t において，式 (9) を最小とする入力 $u^i_o(t)$ を決定する際に，外乱 $w^i_o(t)$ を同時に推定しなければならない．

次章では，エージェント 1 が w^1_o を推定する手法と，入力 u^1_o の計算方法について述べる．なお，エージェント 2 についても，エージェント 1 と同様にして w^2_o および u^2_o を求めることができるため，決定手順については省略する．

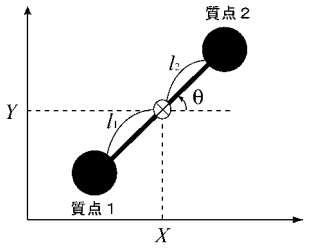

図 1: rod-in-maze 問題

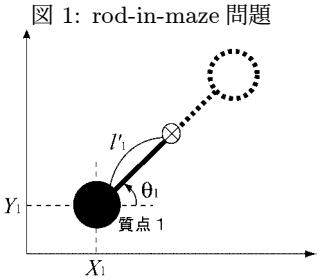

図 2: エージェント 1 が認識するシステム

3. エージェントの行動決定手法

この章では，エージェント 1 が，時刻 t において式 (9) を最小とする行動 $u^1_o(t)$ および外乱 $w^1_o(t)$ を求める手法を提案する．

提案手法では，最初に，過去 N ステップの外乱のうち，絶対値が最大となる値を求める．そして，その範囲内で，w および u をどちらも入力とみなして（実際には w は外乱のために自由に決められないが，まずは決められるものとして）式 (9) を最小化して，N ステップ先までを最適とすると予測される u, w である，$\hat{u}^1_N = [\hat{u}^1(t), \cdots, \hat{u}^1(t + (N-1)\Delta T)], \hat{w}^1_N = [\hat{w}^1(t), \cdots, \hat{w}^1(t + (N-1)\Delta T)]$ を求める．

次に，時刻 $t - \Delta T$ において最適となるように計算した J^1_{cost}, J^1_{obs} の値が，それぞれ実際の値とはどの程度の誤差の割合であったのかを，$\kappa^1_{cost}, \kappa^1_{obs}$ として求める．これらの値は，後に，評価関数の予測値が，より実際の値に近づくように補正する際に用いる．

そして，J^1_{cost}, J^1_{obs} に，それぞれ $\kappa^1_{cost}, \kappa^1_{obs}$ をかけて補正した評価関数を用いて，最適な入力を計算する．最後に，求めた $\hat{u}^1_N$ の最初の要素 $\hat{u}^1(t)$ を入力として加える．

以上の手順を具体的な計算式を用いて述べると，以下の手順となる．なお，初期時刻では，$\hat{u}^1_N, \hat{w}^1_N$ を，すべて 0 で初期化しておく．

Step 1 時刻 $t - \Delta T$ における外乱 w^1 の実際の値 $w^1_{real} = [w^1_{1real}, w^1_{2real}, w^1_{3real}]$ を，以下の式により計算する*1．

$$\begin{aligned} & w^1_{1real}(t - \Delta T) \\ &= \frac{2}{\Delta T^2}\left\{X_1(t) - X_1(t-\Delta T) - \dot{X}_1(t-\Delta T)\cdot\Delta T\right\} \\ &- \frac{1}{M_1}F_{1X}(t-\Delta T) \end{aligned} \tag{10}$$

$$\begin{aligned} & w^1_{2real}(t - \Delta T) \\ &= \frac{2}{\Delta T^2}\left\{Y_1(t) - Y_1(t-\Delta T) - \dot{Y}_1(t-\Delta T)\cdot\Delta T\right\} \\ &- \frac{1}{M_1}F_{1Y}(t-\Delta T) \end{aligned} \tag{11}$$

$$\begin{aligned} & w^1_{3real}(t - \Delta T) \\ &= \frac{2}{\Delta T^2}\left\{\theta_1(t) - \theta_1(t-\Delta T) - \dot{\theta}_1(t-\Delta T)\cdot\Delta T\right\} \\ &- \frac{1}{M_1 l'_1}\{F_{1X}(t-\Delta T)\cdot\sin\theta_1(t-\Delta T) \\ &\quad -F_{1Y}(t-\Delta T)\cos\theta(t-\Delta T)\} \end{aligned} \tag{12}$$

Step 2 過去 N ステップのうち，$w^1_{1real}, w^1_{2real}, w^1_{3real}$ の絶対値が最大のものを，それぞれ $w_{1\max}, w_{2\max}, w_{3\max}$ とする．

Step 3 式 (9) を最小化する $\hat{u}^1_N, \hat{w}^1_N$ を，最適化手法（付録 A を参照）により求める．

Step 4 以下の式から，$\kappa^1_{cost}, \kappa^1_{obs}$ を計算する．

$$\begin{aligned} \kappa^1_{cost}(t) &= \frac{1}{\lambda+1}\left\{\sum_{i=0}^{\lambda}\kappa_{cost}(i) \right. \\ &\quad \left. + \frac{J^1_{cost}(x, \hat{u}^1_N, \hat{w}^{\,0}_N{}^1_N, N, t-1)}{J^1_{cost}(x, \hat{u}^1_N, \hat{w}^1_N, N, t-1)}\right\} \end{aligned} \tag{13}$$

$$\kappa^1_{obs}(t) = \frac{1}{\lambda+1}\left\{\sum_{i=0}^{\lambda}\kappa_{obs}(i)\right.$$

*1 式 (12) は，時刻 $t - \Delta T$ から時刻 t までの θ が一定であるとして，近似的に求めた．

$$+ \quad \frac{J^1_{obs}(x,\hat{u}^1_N,\hat{w}^{\theta 1}_N,N,t-1)}{J^1_{obs}(x,\hat{u}^1_N,\hat{w}^1_N,N,t-1)}\Bigg\} \quad (14)$$

$$\lambda = t/\Delta T \quad (15)$$

$\hat{w}^{\theta 1}_N$ は，$\hat{w}^1_N$ の最初の要素に，時刻 $t-\Delta T$ の外乱を代入したものである．

Step 5 $\kappa^1_{cost}, \kappa^1_{obs}$ を用いて，より実際の評価値に近い値を予測するために補正した評価関数を，次式のように構成する．

$$J^{1^\theta} = k^1_{cost}(t)J_{cost} + k^1_{obs}(t)J_{obs} \quad (16)$$

そして，上式に $\hat{w}^1_N$ を代入し，$\hat{u}^1_N$ についてのみ最適化（付録 A参照）を行う．

Step 6 得られた $\hat{u}^1_N$ の最初の要素 $\hat{u}^1(t)$ を，入力として加える．

4. シミュレーション

提案手法の有効性を検証するために，シミュレーションを行った．LAN で接続された 2 台の計算機を使用し，計算機 1(CPU:Athlon XP 2500+, 周波数:1.84GHz, OS:Windows XP) ではエージェント 1 の行動を，計算機 2(CPU:Pentium M, 周波数:1.3GHz, OS:Window XP) ではエージェント 2 の行動を計算した．また，計算機 1 では，エージェント 1 の行動計算と同時に，式 (1)-(3) であらわされる状態変化を Runge-Kutta 法により計算した．プログラムは Borland C++ Builder 5 で作成した．

4.1 評価関数

J_{cost}, J_{obs} は，次式のように設定した．

$$\begin{aligned} & J_{cost}(x,u,w,N,t) \\ &= \sum_{k=0}^{N}(|X(t+k\Delta T)| + |Y(t+k\Delta T)| \\ &\quad +0.1|\theta(t+k\Delta T)|)\Delta T \\ &+ X^2(t+N\Delta T) + 0.5\dot{X}^2(t+N\Delta T) \\ &+ Y^2(t+N\Delta T) + +0.5\dot{Y}^2(t+N\Delta T) \\ &+ 0.1\theta^2(t+N\Delta T) + 0.05\dot{\theta}^2(t+N\Delta T) \quad (17) \\ & J_{obs}(x,u,w,N,t) \\ &= \sum_{k=0}^{N}(\Lambda(X(t+N\Delta T),Y(t+N\Delta T)) \\ &+ \Gamma^2(X(t+N\Delta T),Y(t+N\Delta T)) \quad (18) \end{aligned}$$

Λ は時刻 t から $t+N\Delta T$ までにシステムが障害物に衝突しないように設定するペナルティ関数であり

$$\Lambda(X,Y) = \begin{cases} \infty & ((X,Y) \in \text{Obstacle}) \\ 0 & ((X,Y) \notin \text{Obstacle}) \end{cases} \quad (19)$$

とした．また，$\Gamma(X,Y)$ は，式 (9) の終端時刻である，時刻 $t+N\Delta T$ において，システムの XY 座標が障害物に近づくほど値が大きくなり，障害物から遠ざかるように設定する，障害物に対するペナルティ関数である．本研究では，簡単のために障害物を矩形状の物体に限定し，障害物の位置に応じて，以下の 2 つの場合に分ける．

1. 障害物が Y 軸上に存在する場合：

$$\Gamma(X,Y) = \begin{cases} 0 \quad (Y > Y_{o\max} + D) \\ \frac{\left|\frac{X_{o\min}+X_{o\max}}{2}\right| - |X|}{Y - Y_{o\max} + \epsilon} \\ \quad ((X_{o\min} < X < X_{o\max}) \\ \quad \cap (Y_{o\min} < Y < Y_{o\max})) \\ \frac{Y - Y_{o\min}}{|X| - |X_{o\max}| + \epsilon} \\ \quad ((X_{o\max} < X < X_{o\max} + D) \\ \quad \cap (Y_{o\min} < Y < Y_{o\max})) \\ \frac{Y - Y_{o\min}}{|X| - |X_{o\min}| + \epsilon} \\ \quad ((X_{o\min} - D < X < X_{o\min}) \\ \quad \cap (Y_{o\min} < Y < Y_{o\max})) \\ \infty \quad ((X,Y) \in \text{Obstacle}) \\ 0 \quad (\text{Otherwise}) \end{cases} \quad (20)$$

2. 障害物が Y 軸上に存在しない場合 ($X_{o\min} > 0$ のとき)：

$$\Gamma(X,Y) = \begin{cases} 0 \quad (Y > Y_{o\max} + D) \\ \frac{X - X_{o\min}}{Y - Y_{o\max} + \epsilon} \\ \quad ((X_{o\min} < X < X_{o\max}) \\ \quad \cap (Y_{o\max} < Y < Y_{o\max} + D)) \\ \frac{Y - Y_{o\min}}{X - X_{o\min} + \epsilon} \\ \quad ((0 < X < X_{o\min}) \\ \quad \cap (Y_{o\min} < Y < Y_{o\max})) \\ \frac{Y - Y_{o\min}}{X - X_{o\max} + \epsilon} \\ \quad ((X_{o\max} + D < X) \\ \quad \cap (Y_{o\min} < Y < Y_{o\max})) \\ \infty \quad ((X,Y) \in \text{Obstacle}) \\ 0 \quad (\text{Otherwise}) \end{cases} \quad (21)$$

本研究では，簡単のために式 (19),(20),(21) における X, Y を，エージェント 1 の場合は X_1, Y_1 およびエージェント 1 が重心とする位置（質点 1 から l'_1 だけ離れた位置）のうち，障害物に近いほうとした（エージェント 2 も同様とした）．また，これらの点が，障害物の周囲 D 以内に入った時点で計算をおこなった．

4.2 エージェントのパラメータ

各エージェントが使用するパラメータは，表 1のとおりである[*2]．また，エージェントが動かす空間内に，表 2に示す障害物が置かれているものとする．各エージェントは，図 1のシステムを，障害物に衝突しないように，各エージェントが設定された位置まで移動させなければならない．

初期値として，図 1の座標 $(X,Y,\theta) = (0.0, 1.0, 0.0)$ とし，目標値を

$$\begin{aligned} &\text{エージェント 1} \qquad (X,Y,\theta) = (-0.1, 0.0, 0.0) \\ &\text{エージェント 2} \qquad (X,Y,\theta) = (0.1, 0.0, 0.0) \end{aligned}$$

となるように設定した．

4.3 シミュレーション結果

得られた軌道のひとつを，図 3に示す．各エージェントは，相手のエージェントがどのような入力を加えているのかを把握していないにもかかわらず，障害物を回避して目標位置まで移動することに成功している．

*2 l'_2 は，エージェント 2 が正確に知ることのできない重心位置とみなした点までの長さである．

表 1: シミュレーションに使用したパラメータ

パラメータ	値
M_1, M_2	1.0[Kg], 1.0[Kg]
l_1, l_2, l_1', l_2'	0.1[m], 0.1[m], 0.15[m], 0.12[m]
$F_{1\,\mathrm{min}}, F_{1\,\mathrm{max}}$	-1.0[m/s^2], 1.0[m/s^2]
$F_{2\,\mathrm{min}}, F_{2\,\mathrm{max}}$	-10.0[m/s^2], 10.0[m/s^2]
D	0.02[m]
$N, \Delta T$	10, 50[ms]
シミュレーション時間	25[s]

表 2: 障害物

障害物 No.	障害物の範囲
1	$X \in [-0.3, 0.3], Y \in [0.2, 0.3]$
2	$X \in [0.4, 0.6], Y \in [0.1, 0.2]$
3	$X \in [-0.6, -0.4], Y \in [0.1, 0.2]$
4	$X \in [0.2, 0.6], Y \in [0.5, 0.7]$
5	$X \in [-0.6, -0.2], Y \in [0.5, 0.7]$

4.4 計算時間

本研究において提案した手法を実システムに適用する場合，次のステップに加える入力の計算が，ΔT 以内に終了しなければならない．構成したシステムでは，25[s] のシミュレーションを，最大で 10.75[s] で計算可能であった．また，1 ステップごとの計算時間は，平均 21.5[ms]，最大 32[ms] となった．設定した $\Delta T = 50[ms]$ よりも小さい値であることから，提案手法は実システムにおいても有効に機能することが予想される．

5. おわりに

複数のエージェントが，他のエージェントと通信せずに，独立して 1 つのシステムに入力を加えて目標値まで移動させる問題を扱った．各エージェントは，近傍の局所的なパラメータのみを用いたモデルを構築し，他のパラメータについては未知パラメータとして行動しつつ獲得する手法を提案した．提案手法を用いたシミュレーションにより，各エージェントが与えられた目的を達成するようにシステムを動かすことのできることを示した．

本研究の応用として，さらに多くのエージェントが介在するシステムにたいして，提案手法を適用したシミュレーションをおこない，幅広い問題に対する有効性について検討する必要性がある．

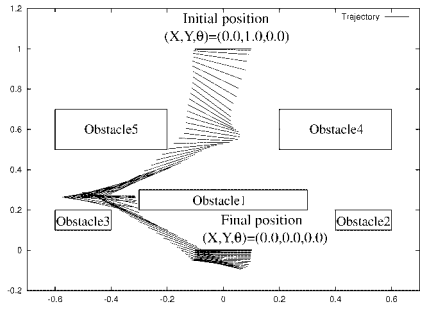

図 3: シミュレーション結果

A 最適化手法

[今福 02] で提案した最適化手法を，本研究に適用する方法を述べる．考え方や，各パラメータについての説明は参考文献に記述されているため，省略する．

Step 1 $u_o = 0, w_o = 0$ とする．

Step 2 使用するパラメータを，$R_{\mathrm{min}} = 0.01, R_{\mathrm{max}} = 1.0, N_{\mathrm{min}} = 10, N_{\mathrm{max}} = 190, l_{\mathrm{min}} = -30, l_{\mathrm{max}} = 190, P_{dr} = 0.1, P_{ir} = 0.01$ と設定する．

Step 3 次式により，$K(P, l)$ を計算する．

$$K(P,l) = R_{\mathrm{min}} + R_{\mathrm{max}}(P+\epsilon)\exp\left(\frac{-l^2}{N_{\mathrm{min}} + N_{\mathrm{max}} \cdot P}\right) \quad (22)$$

そして，現在の u_o, w_o を中心とする探索ベクトル $\Delta u, \Delta w$ を，次のように求める．

$$\Delta u = S \cdot Rand(-K, K) \quad (23)$$

Step 4 $u = u_o + \Delta u, w = w_o + \Delta w$ として式 (9) に代入して計算する．

Step 5 探索成功の場合：

$$u_o := u_o + \Delta u\ ,\ w_o := w_o + \Delta w \quad (24)$$

として，次式により P を更新する．

$$P := P(1 - P_{dr})\left(1 - \frac{K}{R_{\mathrm{min}} + R_{\mathrm{max}}(P+\epsilon)}\right) \quad (25)$$

探索失敗の場合：

$$l := l + 1 \quad (26)$$
$$\text{if } l > l_{\mathrm{min}} \text{ then } l := l_{\mathrm{max}}$$

として，次式により P を更新する．

$$P := (1 + P_{ir})P \quad (27)$$
$$\text{if } P > 1 \text{ then } P := 1$$

Step 6 Step 2〜Step 5 を設定した回数（本研究では 500 回）くり返す．

参考文献

[中村 99] 中村，佐野，沢田: 行動分割型マルチエージェントによる強化学習の高速化，信学技報，NC99-34, pp.41–48 (1999).

[今福 02] 今福，山下，西谷: 非線形 Receding–Horizon 制御のための実時間最適化手法，vol.20，no.8，pp.759–770 (2002).

[小林 01] 小林，木村，小野: 生物的適応システム〜進化・学習のアルゴリズムと創発システム論〜，計測と制御，vol.40，no.10，pp.752–757 (2001).