3E1-03

包摂アーキテクチャ上のパラメータ探索を用いた群ロボットによる適応的行動生成

Generation of Adapting Behavior by Multi-Robot using Parameter Searching on Subsumption Architecture

伊藤芳子[*1] 長谷川修[*2*3]
Ito Yoshiko Hasegawa Osamu

[*1]東京工業大学大学院 総合理工学研究科 知能システム科学専攻
Department of Computational Intelligence and System Science,Tokyo Institute of Technology

[*2]東京工業大学大学院 理工学研究科 像情報工学研究施設 / [*3]科学技術振興機構 さきがけ研究 21
Imaging Science and Engineering Lab.,Tokyo Institute of Thechnology / PRESTO, JST

In this research,we aim to have robots robustness to the change of environments by giving robots autonomous charging function of energy, searching and adaptability to the change of tasks and number of robots as multi-robot. To realize them,we use parameter searching on a subsumption architecture. The subsumption architecture control robots' fundamental behaviors and robots move continuously in principle. Besides,the ability of disposing tasks is developped by parameter searching and then robots adapt to the change of tasks and the number of robots.

1. はじめに

近年、複数台のロボットに分散・協調的に作業を行わせる研究が盛んであるが、それらの多くは「一定のタスク」を効率的に処理するための協調学習手法やアルゴリズムの研究に主眼が置かれており、「タスクやロボット数が変化する状況」にも対応可能な手法やアルゴリズムの研究はあまり見られない[平野 2000],[Mataric 1997]。

そこで本研究では、複数台のロボットの各々に障害物回避やエネルギーの自律的補給などの基本的な行動機能を与えるとともに、群ロボットとして様々なタスクに応じて適切な行動を探索する機能や、タスクやロボット台数の変化に対応する機能を持たせることを考える。

2. アルゴリズム概要

本研究では、作業フィールド内に生じる様々なタスクを確実かつ柔軟に処理するために、包摂アーキテクチャ上のパラメータ学習を用いる。包摂アーキテクチャは個々のロボットの基本的な挙動制御に利用しており、これによりロボットは原則としていかなる状況でも停止することなく動き続けることができる。

包摂アーキテクチャには「作業フィールドに出ずに通信のみする」、「平衡状態に向かう」というレベルを導入する。これらのレベルにより、ロボットは作業フィールド内で予め一定の台数がタスクの発生に備えて適切な配置に着くことができる。

各ロボットの作業フィールド内での位置は、各ロボットごとに、作業フィールドの壁から受ける反力と他のロボットから受ける反力の全てを勘案して決定される。これらの反力の間には、タスクの種類や作業フィールド内に出ているロボットの台数に応じたウエイトが存在する。ロボットは、作業フィールドに出動するロボットの台数、および反力間のウエイトの最適値を適宜探索し、ロボット全体のタスクに対する処理評価の向上を目指す。

さらに本研究では、最適パラメータの探索や記憶メモリの生成を主導するリーダーロボットを特定しないメカニズムを導入することによって、ロボットの故障やロボット台数の変化にも対応可能とした。

連絡先: 伊藤芳子:〒 226-8503 横浜市緑区長津田町 4259 東京工業大学 像情報工学研究施設 R2-52 長谷川研究室，045-924-(5180 Tel，5175 Fax)，ito@isl.titech.ac.jp

3. 問題設定

作業フィールド：ロボットは正方形の作業フィールド内で活動する。正方形の一辺には充電場を設定し、エネルギー残量が一定値を下回ったロボットは充電場でエネルギーを補給する。

ロボット：ロボットには以下の仮定をおく。

- ロボットはすべて均質で、個別のロボット番号を持つ
- 各ロボットはロボット間反力の計算のため周囲に一定のポテンシャルを持つ
- 各ロボットは移動距離や通信量に応じエネルギーを消費
- タスクの発生は全ロボットが認識可能
- ロボット同士は互いに通信可能

タスク：タスクは作業フィールド内の充電場以外の場所に発生する。発生時間間隔は一定で、発生場所はランダムまたは特定の 1 or 2 箇所とする。タスクが発生するとその場所に指定された台数が集まり、設定されたエネルギーを消費する。

4. 動作計画手法

4.1 包摂アーキテクチャ

包摂アーキテクチャはシンプルな処理メカニズムを多層に重ね、高レベルの層が低レベルの層を包摂する機構である。本研究では５層のレベルを設定する。

- レベル０：障害物回避
  距離センサの入力値が一定値以下なら回避行動をとる。
- レベル１：充電場で充電する
  エネルギー残量が一定値以下なら充電場で充電する。
- レベル２：出動せずに通信のみをする
  共有メモリ１を参照し、作業フィールド内に存在するロボット数が決まった数以上の場合は出動せずに通信のみをし、タスクに備えるロボット台数を一定に保つ。最適ロボット数は探索により求める。

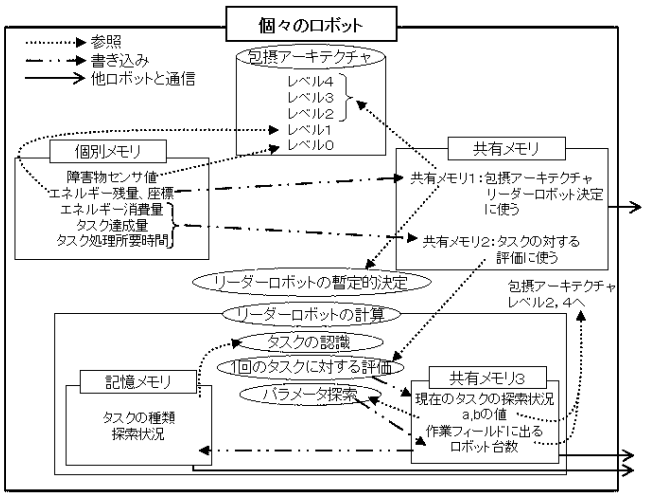

図 1: 個々のロボットのメカニズム

- レベル３：タスクの処理
  共有メモリ 1 を参照してタスク発生場所までの距離を計算し、距離が近い順にタスクの処理に向かう。
- レベル４：平衡状態に向かう
  壁からの反力、他のロボットからの反力・摩擦力を受け、力学的に平衡に達するように移動する。

以上により個々のロボットは原則としていかなる状況でも停止することなく動き続ける。次にレベル４について説明する。

レベル４：平衡状態に向かう

このレベルでは、タスクを迅速に処理するために作業フィールドに出るロボットの最適な配置を行う。これを平衡状態と呼び、以下のように計算する。

- 壁からの反力
  ロボットの左右上下の壁からの距離をそれぞれ d_{left}、d_{right}、d_{up}、d_{down} とすると、壁から受ける反力の x 成分 f_{wall_x}、y 成分 f_{wall_y} は
  $$f_{wall_x} = d_{right} - d_{left}$$
  $$f_{wall_y} = d_{up} - d_{down}$$
- 他のロボットからの反力
  ロボット i がロボット j となす角を θ_j、距離を d_j とすると、ロボット i がロボット j から受ける反力の x 成分 $f_{robot_{jx}}$、y 成分 $f_{robot_{jy}}$ は以下とする。
  $$f_{robot_{jx}} = \alpha \times \exp\left(\frac{-d_j}{\alpha^2\sigma^2}\right) \times \cos\theta_j$$
  $$f_{robot_{jy}} = \alpha \times \exp\left(\frac{-d_j}{\alpha^2\sigma^2}\right) \times \sin\theta_j$$
  ここで α,σ は実験的に定めた定数である。これを自身以外のすべてのロボットについて求め、合計したものを他のロボットからの反力とする。

以上の反力に摩擦力 f_{react} を加え、ロボット i に働く力の x 成分 f_{i_x}、y 成分 f_{i_y} を以下のように求める。

$$f_{i_{(x,y)}} = a \times f_{wall_{(x,y)}} + b \times \sum_{j \neq i}^{j} f_{robot_{j(x,y)}} - f_{react_{(x,y)}} \quad (1)$$

この力がつり合うところが平衡状態であり、各ロボットはその場所で停止する。ここでパラメータ値 a,b の最適値はタスクの種類や作業フィールドに出るロボット台数に依存するので、パラメータ探索により最適な値を求める。

4.2 リーダーロボットの暫定的決定

ロボット台数の変化にも対応できるよう、計算や行動の決定を行なうリーダーロボットを限定しない。タスクが発生するごとに共有メモリ 1 内を参照して、ロボット番号が小さい順にエネルギー残量が一定値を上回っているロボット 1 台をリーダーロボットとし、そのロボットが次から述べるタスクの認識、パラメータ探索、記憶メモリの生成を行う。またリーダーロボットが計算途中に故障すると他ロボットはリーダーロボットが故障したことを認識し、同様の手順で再決定を行う。

4.3 タスクの認識

タスクの種類の変化に対応できるよう、現在どのようなタスクが発生しているかはリーダーロボットによって認識される。過去 10 回分のタスクの記録を蓄積し、それらのタスク発生場所、タスク処理に必要な台数が一定であるかどうかを判定し、一定であればその一定の発生場所、必要台数でのタスクであると決定する。一定でない場合は一定であると認識できるまでタスクが発生するごとにタスクの記録を更新し続ける。

4.4 パラメータ探索

タスクの種類が認識できると、ロボット全体のタスク処理の評価がよくなるよう、作業フィールドに出るロボット台数と式 1 のパラメータ a,b (以下これら 3 つのパラメータの組をパラメータ状態と呼ぶ) を探索により決定する。タスクが認識できない間はパラメータ探索は行なわず、事前に決められたパラメータ状態でタスクの処理を行う。

タスク処理に対する評価

タスク処理に対するパラメータ状態の評価は、一回のタスクをこなすごとに式 2 で計算する。

一回のタスク処理に対する評価 =

$$\frac{\text{タスク達成量}}{\text{エネルギー消費量}} \times \alpha + \left(\frac{\text{タスク達成量}}{\text{タスク処理所要時間}} \times \gamma - \delta\right) \times \beta \quad (2)$$

ここで式 2 の第一項はいかにエネルギーの消費量を少なくしてタスク処理を行うか、第二項はいかに所要時間を短くしてタスク処理を行うかを評価する項である。第二項の γ, δ は第一項の値との調整のための実験的に定めた定数である。また、α, β は 2 種類の評価をどの程度考慮に入れるかを決定する値であり、それぞれ 0 以上 1 以下、かつ 2 値の和が 1 以下である。

一つのパラメータ状態に対し、タスク 100 回分の平均評価をそのパラメータ状態に対する評価値とする。100 回こなすと評価が安定することを実験で確かめた。

パラメータ探索の探索範囲決定

タスクの認識が出来ると、それに応じてパラメータ探索の探索範囲を決定する。作業フィールドに出るロボット台数は (タスク処理に必要な台数) ～ (タスク処理に必要な台数) + 2 を探索範囲とし、それぞれの台数において a,b の探索範囲は 0 ～ 8 の 2 次元平面とする。このように探索範囲を決定したのは、作業フィールドに出るロボット台数をこれ以上増加させても、エネルギー効率が悪化し評価に改善が見られなかったためである。また a,b の値の探索範囲については、これ以上大きくするとロボットに働く反力がつり合わずにロボットが平衡状態に達せず、ロボットの行動の評価に改善が見られなかったためである。

パラメータ探索

パラメータ探索の初期探索地点は $(a,b) = (2,2), (6,2), (4,4), (2,6), (6,6)$ である。初期探索地点をこのように探索範囲に均等に散らばらせたのは、局所解に陥るの極力防ぐためである。

初期探索地点の評価値がすべて求まると、その評価値の上位3つの探索地点を次の探索の始点とし、それぞれの始点周囲の8点を探索する。次にすべての探索地点の中で評価が最大の地点を次の探索の始点とし、同様に探索して評価が最大の地点を探す。この操作を繰り返し始点の周囲の8点を探索しても評価に改善が見られなくなったらその始点を現在の探索台数における最終地点とする。すべての作業フィールドに出るロボット台数の探索範囲で同様の探索を行い、すべての探索地点の中で最大の地点を現在起こっているタスクに対する探索の最終地点とし、その時の作業フィールドに出るロボット台数、および式1における a, b の値でタスクの処理を行う。

4.5 記憶メモリの生成

タスクの変化に対応出来るよう、記憶メモリの生成を行う。

記憶メモリにはタスクの種類と探索の進捗状況を組にして書き込む。リーダーロボットがタスクを認識すると、記憶メモリを参照して経験済みタスクかを判断する。経験済みでなければ新しく記憶メモリの形成を行い、初めからパラメータ探索を進める。経験済みならば、記憶メモリの中から同じ種類のタスクの探索の進捗状況を取り出し、続きから探索する。一つのパラメータ状態に対する評価が決定したら記憶メモリに書き込む。

記憶メモリは更新されるごとにリーダーロボットが一定のルールに基づき特定のロボットに通信し、他ロボットは通信を受け取ると同じく特定のロボットに通信する。よってすべてのロボットは同じ経験を持つ。

4.6 個別メモリ、共有メモリ

以上のアルゴリズムを実現するために、個別メモリと共有メモリ3種類を作成する（図1参照）。

個別メモリ

個別メモリは自身の障害物センサー値、エネルギー残量、座標値、タスク処理間のエネルギー消費量、タスク達成量、タスク処理所要時間を書き込み、他のロボットと通信は行わない。

共有メモリ1

個々のロボットは個別メモリ内のエネルギー残量と座標を共有メモリ1に書き込み、また自分が作業フィールドに出るかを判断し、結果を書き込む。これにより個々のロボットは他のすべてのロボットのエネルギー残量、座標、作業フィールドに出るかの判断を知り、包摂アーキテクチャの実現とリーダーロボットの暫定的決定に用いる。個々のロボットが値を更新するごとに共有メモリ1に結果を書き込み、メモリ内の情報が変更になるたびに決まったロボットに通信する。これによりすべてのロボットはこのメモリを共有する。

共有メモリ2

共有メモリ2には個々のロボットが個別メモリからエネルギーの消費量、タスク達成量、タスク処理所要時間を書き込み、リーダーロボットがタスクに対する評価を行う際に使用する。タスクの処理が終了するごとに個別メモリから共有メモリ2に書き込み、共有メモリ1と同様の方法で他ロボットと通信を行い、このメモリもすべてのロボットで共有する。

共有メモリ3

共有メモリ3には現在のタスクの探索状況とパラメータ状態、作業フィールドに出るロボット台数をリーダーロボットが書き込む。タスクの探索状況とパラメータ状態はリーダーロボットがパラメータ探索をする際に使用し、パラメータ状態と作業フィールドに出るロボット台数は個々のロボットが包摂アーキテクチャで動作する際のレベル2，4で参照する。リーダーロボットがメモリ内容を変更するたびに特定ロボットに通信し、他ロボットはこのメモリ内容を受け取ると特定ロボットに通信する。これによりこのメモリもすべてのロボットで共有する。

5. シミュレーション実験の結果と考察

以上のアルゴリズムの有用性を確かめるために、[Khepera Simulator version 2.0] によりシミュレーションを行った。初期ロボット台数は5台、タスクは表1の9種類を設定した。タスクは発生場所が1箇所の場合は $(500, 410)$ の位置に、2箇所の場合は $(200, 200)(800, 200)$ の位置に発生するとした。

表 1: タスクの種類

タスクの種類	必要台数	発生場所
Task1	1	Random
Task2	2	Random
Task3	3	Random
Task4	1	1 箇所
Task5	2	1 箇所
Task6	3	1 箇所
Task7	1	2 箇所
Task8	2	2 箇所
Task9	3	2 箇所

5.1 タスクごとの最適パラメータ状態の探索

タスクの種類は一定で式2における α, β の値を (1.0,0.0), (0.0,1.0),(0.5,0.5) の3種類に変化させて実験を行った。

Task8 におけるそれぞれの α, β の組に対する評価の変化を図2に示す。横軸は1台のロボットが1回動作する時間間隔を 1pas とし、縦軸は評価を示す。

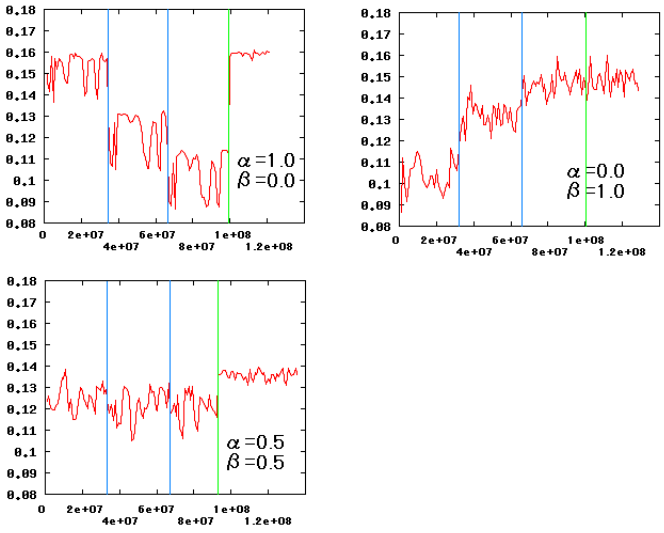

図 2: Task8 における評価の遷移 (探索が終了すると (図中の緑線) 一番良い評価を与えるパラメータ状態で行動)

図中の緑線は探索終了線を示し、青線は探索する作業フィールドに出るロボット台数が1台増加する時間を示す。どの評価においても、緑線以降は探索が終了しているので、それまでの評価の中で一番良い評価に近い値が出るパラメータ状態で行動を行っていることがわかる。

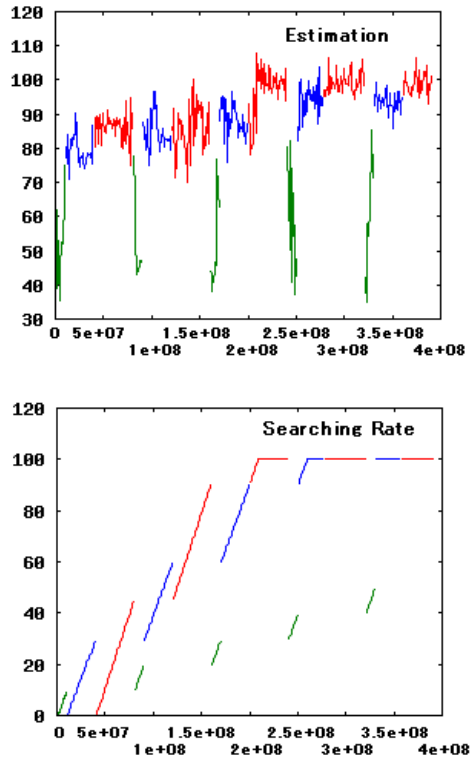

図 3: Task6→Task2→Task7 と変化させたときの評価と探索率の遷移 (上：評価の遷移、それぞれのタスクで一番良い評価を与えるパラメータ状態で行動, 下：探索率の遷移、タスクが変化すると途中から探索を再開)

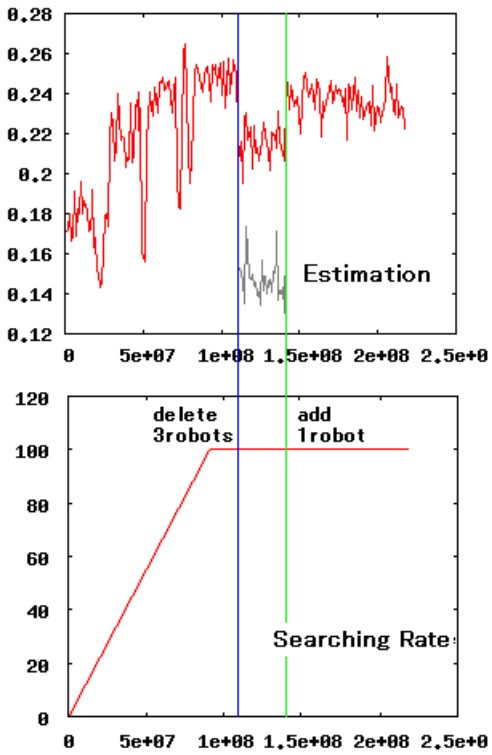

図 4: 探索終了後にロボット台数を変化させたときの評価と探索率の遷移 (上：評価の遷移、台数が減少してもその台数で一番良い評価を与えるように行動, 下：探索率の遷移、探索が終了すると 100 になりそれ以上探索しない)

5.2 タスクの種類が変化する場合の探索

一つのタスクに対して探索している最中にタスクの種類が変化する場合の動作を確かめる。

ここでタスクの種類を 3 種類設定し、Task6, 1000 回 →Task2, 3000 回 →Task7, 4000 回と繰り返し変化させた。結果を 3 に示す。緑線が Task6、青線が Task2、赤線が Task7 である。上のグラフが $(\alpha, \beta) = (0.5, 0.5)$ のときの評価で、探索中の最高値を 100 として正規化して示している。下のグラフは探索率を示す。最終的な探索地点の数を 100 として正規化し、探索地点の数により、探索の進行状況を示した。

図に示されるように、新しいタスクが発生すると探索率が一旦 0 に落ち、記憶メモリを新規生成して初めから探索している。また過去に経験したことのあるタスクが再び発生すると、経験済みタスクと認識して探索率が前回タスクが変わった時点の値から再開している。これらよりタスクごとに記憶メモリを作り、記憶を経験として有効に活用できていることがわかる。

5.3 ロボット台数が変化する場合の探索

探索が終了してからロボットが壊れたり追加したりしてロボット台数が変化しても、リーダーロボットをタスク発生ごとに決定して行動し、破綻することなく動き続けるかどうかをシミュレーションにより確かめる。

Task1 を発生させ、探索終了後にロボット 3 台 (ロボット番号 0, 1, 2) を消去し、しばらくしてから 1 台 (ロボット番号 5) を追加した。結果を図 4 に示す。青線で示した地点がロボットを削除した時間、緑線で示した地点がロボットを追加した時間である。上のグラフが評価の遷移で、下のグラフが探索終了時点の探索地点の数を 100 とした探索率の遷移である。探索率は探索は終了しているのでロボット台数が変化しても変わらず 100 のままである。

本実験の場合はタスク処理に必要な台数が 1 台なので、作業フィールドに出るロボット台数の探索範囲は 1 台から 3 台である。2 台に減少すると、探索範囲 1 台、2 台の中で一番良い評価を与えるパラメータ状態を取り出して行動するが、5 台存在していた時より評価は悪くなる。これはエネルギーの補給などを行っているとき、常に作業フィールドに出るロボット台数を一定保てないためだと考えられる。ランダムなパラメータでロボットが行動したときの評価である灰色線より評価は良くなっており、探索の成果が現れていると考えられる。

またロボット台数が 3 台に増えても記憶メモリを参照して探索状況を読み出し、その台数で一番よい評価を与えるパラメータ状態で行動する。ロボット台数が 3 台になると評価は改善され、ほぼ前回の評価を取り戻すことが出来る。これは台数が 3 台になると前述のような悪影響が減少するためと考えられる。

参考文献

[平野 2000] 平野智一, 他：未知環境における移動ロボット群の経路学習, 機論 (C 編),pp168-175(2000)

[Mataric 1997] Mataric,M.J. : "Reward Functions for Accelerated Learning", The 11th Int'l Conf on Machine Learning,pp181-189(1997)