組立/分解作業手順の学習制御 —離散事象システム論からのアプローチ—

鈴木 達也^{*1} 残間 忠直^{*1} 稲葉 昭夫^{*2} 大熊 繁^{*1}

Learning Control of Assembly/Disassembly Task Sequence —An Approach with Discrete Event System Theory—

Tatsuya Suzuki^{*1}, Tadanao Zama^{*1}, Akio Inaba^{*2} and Shigeru Okuma^{*1}

This paper studies on the learning control problem of assembly/disassembly task sequence. The proposed learning control system has two sub systems. One of them is a supervisory control system which can realize the implicit specifications such as equipment-related or task-independent specifications. The other one is a learning automata system which can realize the explicit specifications such as task-dependent specifications. This approach allows the flexible design to accommodate frequently changing manufacturing requirements. Some simulation results to verify the effectiveness of the proposed system are shown.

Key Words: Supervisory Control, Learning Automata, Assembly Sequence

1. はじめに

最近、ロボティクス・オートメーションの分野において、組立/分解作業の自動化に関する研究が盛んに行われている[1]～[12]。これは組立/分解作業が機構学や制御工学に代表される要素技術と離散事象システム理論などに代表されるシステム技術との融合から成り立つ作業であり、どちらの立場から見ても数々の興味深い問題を含んでいるためである。組立作業の自動化に関する研究は大きく分けて、1. 組付け・分解可能性判定問題、2. 可能な組立/分解手順の表現問題、3. 最適手順の探索問題、4. 手順の制御、実行問題、の四つのステージに分けられると考えられる。1.に関しては二次元の場合は解決済[13]で、三次元の場合は微小変位を仮定した場合は解決済[14]、一般の場合は未解決である。2.に関しては、3., 4.の基本的枠組を与えるものになるが、これまで、AND/OR グラフ[1]～[3]やペトリネット (Petri Net) を用いて表現したもの[8]～[11]、あるいは時間論理 (Temporal Logic) を用いた表現手法[12]などが提案されている。これらは、いずれも比較的コンパクトに可能なすべての組立/分解手順を表現できるという特長を有しているが、そのあとの計画や制御問題を考えると、代数的構造を有し、かつグラフィックユーザーインターフェースが可能なペトリネットを用いた表現が、計算機、および設計者との整合性という点で優位に立っていると考えられる。ペトリネットを用

いる場合、3.に関しては、各単位作業(タスク)に割り当てられた何らかの意味でのコストの総和を最小化するという問題が線形計画法に帰着できることが知られている[8]。また、4.に関しては、離散事象システムの分野における制御手法の一つであるスーパーバイザ制御の適用が検討されており[11]、セーフ性、排他性や公平性といった実時間の仕様に応えるための方法論が示されている。

上記の一連の研究において注意しておくべき重要な点は、3.において、各タスクを評価するコストにかなりの曖昧さがあるという点である。これらの値は、各タスクの難しさを定量的に表現したものともみなすことができるが、この場合、設計者の主観によって左右される場合が多い。また、対象物やロボットが有する機械的パラメータをもとに決める場合においても、三次元データが有する不確かさのため、やはり、かなりの曖昧さが残る。このような場合、単に確定的な線形計画問題として枠組を構成するには限界があり、何らかの不確かさをもった離散事象システムにおいて、学習的な動作を行わせることにより、ある意味で望ましい手順を自ら獲得する問題としてとらえる方がむしろ自然である。

以上の点を踏まえて本論文では、各タスクが持つ曖昧さを確率分布として表現し、学習オートマトンの考え方[24]に基づいた組立/分解作業手順の学習問題について検討する。学習オートマトンは確率オートマトンの一種で、複数個用意されたアクション(組立/分解作業の場合は各タスクに相当するので以後タスクと呼ぶ)の一つを生起確率に従って出力し、そのタスクに対する評価値をもとにタスクの生起確率を更新する。組立/分解手順の問題の場合は、手順、つまり一連のタスク系列が評価の

原稿受付 1995年8月7日

^{*1}名古屋大学工学部

^{*2}岐阜県金属試験場

^{*1}Nagoya University School of Engineering

^{*2}Gifu Prefectural Metal Research Institute

対象となるため、系列に含まれる各タスクのコストの総和で手順を評価し、その評価に基づいて、各タスクの生起確率を更新することにする。この場合、階層的ではあるが、静的な状況下での学習とみなすことができ、絶対適応性と呼ばれる数理的性質が保証される強化学習則が存在することが知られている[29]。

このように各タスクの総和に基づいた評価法を用いる場合、注意しなければならない点は、排他性、並行性や公平性などといった、作業環境などに起因するタスクの生起順序に関する仕様を考慮に入れることが困難であるという点である。このような仕様を満たすよう制御する問題は、離散事象システムにおけるスーパーバイザ制御問題として定式化されており[16]～[22]、文献[11]では、組立問題に適用した例も報告されている。そこで本論文では、上記のような仕様を満たすスーパーバイザと学習オートマトンを併用することによって、生起の順序に関する仕様を満たす範囲内で、学習オートマトンにより最適な手順を獲得する学習制御システムを提案する。提案システムの特長は、各タスクの情報に基づいて積極的に良い手順を探そうとするサブシステム（学習オートマトン）と作業環境などに起因するタスクの順序関係に関する仕様を満たしたり、明らかな禁止事項だけは行わせないある意味での消極的なサブシステム（スーパーバイザ）とが個別に実現されている点である。これによって、作業環境などが変更になって生起の順序に関する仕様などに変更があった場合や、評価規範を再設計するような場合でも、システム全体を変更する必要はなく、対応するサブシステムを変更するだけで対応できる。また、両サブシステムとも同じオートマトンの枠組内で構成されているので、互いに整合性が良く、計算機への実装が容易になるという特長を合わせ持つ。

なお、本論文では簡単のため、複数のタスクの同時実行は考えないが、簡単な拡張により、同時実行を許す場合に対しても適用可能となる。以下に準備として、本論文で用いる記号の定義を行う。

T^* : T の要素から成るすべての有限事象列の集合を表す。

$\bar{L}$: 発火系列の集合 L のすべての要素に対して、その任意の接頭語から成る集合を表す。

!: 定義可能であることを表す。

$\wedge$: 論理積を表す。

A/B : 集合 A と集合 B の差集合 ($A - B$) を表す。

2. ペトリネットによる組立/分解作業手順の表現とその評価

この章では、Fig.1 に示すような Adjustable Clamp の分解作業を Fig.2 に示すような作業環境で行う場合を例にとってペトリネットを用いた組立/分解作業手順の表現方法とその評価関数を示す。

Fig. 1 Adjustable Clamp

2.1 ペトリネットによる表現

Fig.1 に示す Adjustable Clamp の可能なすべての分解作業手順は Fig.3 に示すペトリネットで表現される[9]。このペトリネットは Sanderson らの提案した部品の結合関係のみからなる AND/OR グラフをペトリネットに置き換え、文献[10]で提案されている詳細化アルゴリズムを用いて、Fig.2 に示す作業環境に関する情報を含めることによって得ることができる。ここで、Fig.3 のペトリネットを 5 項組の組立/分解ペトリネット N として次のように定義する。

$$N := (S, T, I, O, M_0) \quad (1)$$

ただし、

S : プレースの集合

T : トランジションの集合

$I: T \times S \rightarrow \{0, 1\}$ (プレースからトランジションへのアークの数)

$O: T \times S \rightarrow \{0, 1\}$ (トランジションからプレースへのアークの数)

M_0 : 初期マーキング

を表す。各プレースは半組立品の状態に対応し、各半組立品を構成する部品群、半組立品の位置、半組立品の向き、および半組立品のなかで固定治具で把持されている部品に関する情報を

Fig. 2 Production System

Fig. 3 Petri net for adjustable clamp

含んでいる。例えば、プレース 8 は部品 ABC から構成される半組立品が作業台 (Work Station) 上で部品 A が固定治具で X 方向に把持されている状態を表している。なお、作業台以外では、固定部品や把持方向に関する情報は考慮に入れないものとする。ここで、 s_p を完成品が製品置き場 (Product Station) に置かれた状態に対応するプレース (Fig.3 において、一番上にあるプレース) とすると、例えば、分解問題においては初期マーキングは次式で与えられる。

$$M_0(s_i) = 1, \quad \text{if } s_i = s_p \quad (2)$$

$$M_0(s_i) = 0, \quad \text{if } s_i \neq s_p \quad (3)$$

これに対し、目標マーキング M_m は部品が部品置き場 (Parts Station) に置かれた状態に相当する各プレース (一番下にあるプレース群) がトークンを持ち、他のプレースにはトークンが存在しない状態として与えられる。

また、各トランジションは Load (製品置き場または退避所 (Stack) にある一つの半組立品を作業台に設置する作業)、Unload (作業台に設置された一つの半組立品を部品置き場、または退避所へ移動させる作業)、Disassembly (作業台に設置された一つの半組立品からある一つの半組立品を分離し、それを部品置き場または退避所へ移動させる作業) のいずれかの作業を表している。例えばトランジション 1 は Load 作業で、製品置き場から ABCDE からなる半組立品を作業台に A という部品を固定治具で把持して X 方向に設置する作業を表している。

なお、このペトリネットはツリー状の構造をしていることから、これを状態方程式で表現すると、その基底解と実際の手順解が一对一に対応することが知られている [8]。したがって、各作業 (トランジション) に割り当てられた評価値の総和が最小になるような手順の探索問題が線形計画法により解けることが知られている。

2.2 各タスクの評価規範

この節では Fig.3 のペトリネットの各トランジションの評価方法の一例を示す。文献 [9] では、分解作業の難易度を示す尺度の一例として以下の三つの評価規範が示されている。

安定性	半組立品を構成する各部品の発生可能な速度ベクトルの総和で評価する。
総和が小さいほど安定性は高いとする。
操作性	マニピュレータの可操作性楕円体 [15] と把持部品の発生可能な速度ベクトルの集合との積集合で評価する。
積集合が大きいほど操作性は高いとする。
エネルギー	半組立品を構成する各部品の質量の総和で評価する。
質量が大きいほど大きなエネルギーが必要であるとする。

なお、これらの評価関数の詳細については文献 [9] を参照されたい。また、組立作業の場合はこれと異なる評価規範の導入も十分に考えられるが、この点についての詳細な検討は本論文の範囲外であるので、これ以上は言及しない。5 章では、これらの評価値に確率的な曖昧さを加味した評価値を考え、それを基に学習における評価規範を生成する。

3. スーパーバイザによる手順の制御

この章では、先行関係や排他性といった生起の順序に関する仕様を満たすスーパーバイザの構成可能性について検討し [11]、具体的な構成例を示す。

スーパーバイザの役割は、制御対象となるシステムが、ある部分発火系列のみを生成するように各トランジションの生起を許容あるいは禁止することである [16]~[21]。Fig.3 に示すような分解ペトリネットを制御対象とする場合、制御対象が様々な形式言語論的性質を有することから、スーパーバイザが有限の状態オートマトンで構成可能であることが以下のように示される。

3.1 組立/分解ペトリネットの形式言語論的性質

まずはじめに、 N の形式言語論的性質 [31] について考察するため、 N の可達グラフを次のようなオートマトン R としてみなす (4.1 節 Fig.7 参照)。

$$R := (Q_r, T, \delta_r, q_{r0}, q_{rm}) \quad (4)$$

ここで、

Q_r : 可達マーキングの集合

δ_r : トランジションのタスク系列に伴うマーキングの遷移を表す関数で

$T^* \times Q_r \rightarrow Q_r$ を表す部分関数

q_{r0} : 初期状態 ($= M_0$)

q_{rm} : 最終状態 ($= M_m$)

である。これらをもとに次式で表されるようなタスク系列の集合、すなわち言語を定義する。

$$L(N) := \{\sigma \in T^*; \delta_r(\sigma, q_{r0})!\} \quad (5)$$

$$L_m(N) := \{\sigma \in L(N); \delta_r(\sigma, q_{r0}) = q_{rm}\} \quad (6)$$

また、一般に $L_m(N) \subseteq L(N)$ である。

以上の定義をもとに、 N の形式言語論的性質に関する補題を以下に示す。

補題 1 $L_m(N)$ は正則言語である。

補題 2 $L_m(N)$ は *consistent* である。つまり、

$$\sigma_i \notin \{\bar{\sigma}_j\} \quad \forall \sigma_i, \sigma_j \in L_m(N) \quad (i \neq j) \quad (7)$$

である。

補題 3 $L(N)$ は *nonblocking* である。つまり、

$$\overline{L_m(N)} = L(N) \quad (8)$$

である。

これらの補題の証明はいずれも組立/分解ペトリネットがツリー構造をしていることから容易に導かれる。詳細は文献 [11] を参照されたい。

3.2 組立/分解制御ペトリネットの閉ループ制御

本節ではまず, N を次式で表されるような外部入力プレース付きペトリネット N_c に拡張する.

$$N_c := (S \cup S_c, T, I_c, O_c, M_{c0}) \quad (9)$$

ここで,

S_c : 外部入力プレースの集合

$I_c: T \times (S \cup S_c) \rightarrow \{0, 1\}$

$O_c: T \times (S \cup S_c) \rightarrow \{0, 1\}$

M_{c0} : 初期マーキング

である. また, $\forall s_c \in S_c$ に対して, $I_c(t, s_c)$ のとる値は 1 または 0 のいずれかであり, しかも $I_c(t, s_c) = 1$ を満たす t はたかだか一つであるとする. また, 外部入力プレースを導入することによって T をその発火が外部入力プレースによって制御されるトランジション (可制御トランジション) の集合 T_c と制御されないトランジション (非可制御トランジション) の集合 T_u に分けて考える. すなわち,

$$T_c := \{t \in T; I_c(t, s_{ci}) = 1, \exists i\} \quad (10)$$

$$T_u := T/T_c \quad (11)$$

とする. Fig. 4 にトランジションに対して外部入力プレースを付加した例を示す. 以上をもとに, 組立/分解制御ペトリネット G を次のように定義する.

$$G := (N_c, \Gamma) \quad (12)$$

ただし, $\Gamma = \{0, 1\}^{S_c}$ で, 各 $\gamma \in \Gamma$ は, $\gamma = (\gamma_1 \cdots \gamma_z)$, ただし $z = \|S_c\|$ であり, γ_i は i 番目の外部入力プレース内のトークンの有無を表す. つまり,

$$M_c(s_{ci}) = \gamma_i, \quad \forall s_{ci} \in S_c \quad (13)$$

である. このトークンの有無を制御することによって, 各トランジションの発火を制御できることになる. この意味で, γ は制御パターンと呼ばれている. また, $M_c(s_i) = M(s_i)$ ($\forall s_i \notin S_c$) である.

次に, G に対するスーパーバイザ C を以下のよう定義する.

$$C := (V, \Phi) \quad (14)$$

ただし, $V := (Q_v, T, \delta_v, q_{v0})$ で $\Phi: Q_v \rightarrow \Gamma$ は V の状態から制御パターンへの写像を表す. つまり, $q_v \in Q_v$ に対して

$$\gamma := \Phi(q_v) \in \Gamma \quad (15)$$

Fig. 4 External input place

である. 以上をもとに閉ループ系 $G|C$ (Fig. 5) を考える. Fig. 5 に示すように, スーパーバイザは制御ペトリネットに対して, 制御パターンを出力し, 制御ペトリネットは発火したトランジションをスーパーバイザにフィードバックする. このとき, 閉ループ系 $G|C$ の状態 x は次のように定義される.

$$x := (M_c, q_v) \quad (16)$$

つまり, 閉ループ系において G と C は同期して動くものとする. 組立/分解ペトリネット N に対して, 言語 $L(N)$, $L_m(N)$ を定義したように, $G|C$ に対して, 以下の言語を定義する.

$L(G|C)$: 初期状態 x_0 での可能なタスク系列の集合
(17)

$$L_m(G|C) := L(G|C) \cap L_m(N) \quad (18)$$

一般に, トランジション t が N において発火可能であるならば, スーパーバイザ C においても t は生起可能であることが閉ループ制御においては必要となる. このようなとき, スーパーバイザは完全であるといわれ, それは以下のように定義される.

定義 1 $G|C$ において, 次の条件が成立するとき, スーパーバイザ C は完全であるという.

任意の $\sigma \in L(G|C)$ と任意の $t \in \Sigma$ に対して, $x_0[\sigma > x$ とおくと, t は x で G において発火可能であるならば, $\delta_v(\sigma t, q_{v0})!$ である.

また, 次式が成り立つときスーパーバイザは *nonblocking* であるという.

$$\overline{L_m(G|C)} = L(G|C) \quad (19)$$

3.3 組立/分解制御ペトリネットに対するスーパーバイザの構成

ここでは, 組立/分解ペトリネット N に対して, 制御目的がある部分言語 K を用いて与えられたときに, その制御目的を実現するスーパーバイザが存在するための条件について考察する. はじめに準備として, 可制御性 [16] の定義を示す.

定義 2 部分言語 K が G に関して可制御であるとは, 次式が成り立つことをさす.

$$\overline{K}T_u \cap L(N) \subseteq \overline{K} \quad (20)$$

一般に非可制御なトランジションの発火はスーパーバイザによって制御できないが, 式 (20) は非可制御なトランジション

Fig. 5 Supervisory control system

が発火しても常に制御目的を満たしていることを表している。組立分解ペトリネットにおいては、次の定理が成り立つことが明らかにされている。

定理 1 組立/分解ペトリネットにおいて、制御目的を表す、ある空でないタスク系列の部分言語 $K \subseteq L_m(N)$ に対して、 $L_m(G|C) = K$ なる完全でブロックしないスーパーバイザ C が存在するための必要十分条件は K の可制御性のみである。

証明は文献[11]を参照されたい。また[11]では組立/分解制御ペトリネットに対するスーパーバイザの構成法についても述べられている。Fig.3中のすべてのトランジションが可制御、つまり外部入力プレースを有する場合は、その生起を任意に許容したり禁止したりできることになり、 $T_c = T$, $T_u = \emptyset$ となるので、任意の $K \subseteq L_m(N)$ に対してスーパーバイザが構成可能であることが示される。

3.4 スーパーバイザの構成例

制御目的として、次の制御目的を考える。

制御目的1: 作業台に二つ以上の半組立品を設置しない。

この制御目的を実現するスーパーバイザは、Fig.6で表されるオートマトン V と Table 1 で与えられる制御パターン γ より構成される。3.2節で述べたように、スーパーバイザはそれぞれの状態で Table 1 で示されるような制御パターンを出力し、一方、組立/分解制御ペトリネットは、スーパーバイザが出力した制御パターンのもとで発火可能なトランジションを発火させる。そして、発火したトランジション（実行したタスク）をスーパーバイザにフィードバックする。このとき、スーパーバイザと制御対象（組立/分解制御ペトリネット）は同期して状

態、およびトークンの遷移を行う。また、初期状態においてはスーパーバイザ、組立/分解制御ペトリネット共に、初期状態、および初期マーキングにあるとする。例えば初期状態において、スーパーバイザの出力する制御パターンは $\gamma = (1, 1, \dots, 1)$ であるから、すべてのトランジションの発火が許容されている、組立/分解制御ペトリネットは初期マーキングで発火可能な t_1 と t_2 のどちらかを発火させ（つまり、どちらかのタスクを実行する）、発火したトランジションをスーパーバイザにフィードバックする。これを繰り返し、例えば t_1, t_3, t_8 の順で発火したとすると、組立/分解制御ペトリネットにおいては s_9, s_{20} と s_{27} にトークンが存在する状態に、また、スーパーバイザにおいては状態 10 に到達する。この状態で組立/分解制御ペトリネットにおいて t_{17} が発火すると作業台上に二つの半組立品が設置されることになり、制御目的に反する。したがって、スーパーバイザは Table 1 に示すように、状態 10 で t_{17} の発火を禁止するような制御パターンを出力し、その発火を禁止している。

4. 学習オートマトンによる手順の学習

前章ではスーパーバイザを導入して手順の順序関係に関する制御が行えることを示した。しかしながら、スーパーバイザそれ自体は、望ましくない事象の生起を禁止するのみで、積極的に何らかの意味で手順を最適化しようとする役割を果たすものではない。そこで本章では、各タスクのコストに曖昧さがあるために線形計画法による最適手順の探索が可能でない場合について考察するため、学習オートマトンの考え方[24]を導入し、手順の学習を行う方法論を示す。

4.1 学習オートマトンに関する諸定義

組立/分解を表すオートマトンは3.1節で述べた可達グラフより生成される（ $= R$ ）。このとき R は Fig.7 に示すような階層的な構造となる。今、 R に対して一般性を失うことなく、Fig.7 に示すように階層レベルを与え、各タスク（トランジション）に対して以下のように新たにラベル付けを行い、それをタスク指令とみなす。ただし、Fig.7 では、一部のラベルは省略してある。

$$\alpha_{i_1 i_2 \dots i_N} : \text{レベル } N \text{ で選択されるタスク} \quad (21)$$

また下層レベルほど添字が長くなることを避けるために、

$$i_1 i_2 \dots i_n := i_n \quad (22)$$

と記述する。ただし、 R においては、階層レベルを越えた結合も存在するが、Fig.7 においては、そのような場合、ダミー状態を新たに付加して（点線の丸で表された状態）、階層レベル

Fig.6 Supervisory V for example

Table 1 Control pattern γ for example

State q_v	10	11	14	16	others
t_{17}	0	0	1	1	1
t_{18}	0	0	1	1	1
t_{24}	1	1	0	0	1
t_{25}	1	1	0	0	1
others	1	1	1	1	1

Fig. 7 Reachability graph

Fig. 8 Learning automaton system

を越えた結合は存在しない形となっている点に注意されたい。なお、このダミー状態の存在は後述する学習に対して何らの影響も及ぼさない。以上の記号を用いて、学習オートマトンの動作について説明する (Fig. 8)。

学習を行う前は、各状態において次に出力するタスク指令の生起確率は均等に割り振られているものとする。例えば状態 1 でタスク指令 α_1 と α_2 の生起確率は $1/2$ ずつとする。学習オートマトンは、それぞれのタスクに割り当てられた生起確率に従って、各状態で次に実行すべきタスク指令 α を確率的に選択し、出力する。出力されたタスク指令は制御パターン生成部で制御パターン γ に変換される。この変換によって、出力されたタスクに相当するトランジションに付加された外部入力プレースのみにトークンを一つ与え、他の外部入力プレースにはトークンを与えないようなパターンが得られる。出力されたタスク指令が実行されたら、制御ペトリネットは実行したタスク t を学習オートマトンにフィードバックし、学習オートマトンと制御ペトリネットは同期して状態、およびトークンを遷移

させる。以後、これを繰り返し、最終状態に到達したら、実行したタスク系列 σ を評価し、その評価に基づいて各タスク指令の生起確率を更新する。

4.2 環境モデル

学習オートマトンにとって環境とは、 n 回目の試行においてオートマトンが選択したタスク系列を評価し、評価値 $\beta(n)$ ($0 \leq \beta(n) \leq 1$) をある確率分布に従って出力するシステムと言い換えられる。学習オートマトンの枠組においては、この $\beta(n)$ の取り得る値によって環境モデルが次の三つのモデルに分類される。

- (1) $0, 1$ の 2 値に対してのみ確率分布が与えられる：
P モデル
- (2) 区間 $[0, 1]$ で有限の離散的な確率分布が与えられる：
Q モデル
- (3) 区間 $[0, 1]$ で連続的な確率分布が与えられる：
S モデル

ここで、いずれのモデルにおいても $\beta(n)$ はペナルティーの大きさを表し、その値が小さいほど望ましいとする。直観的には P, Q, S モデルの順に、より細かな評価が可能になるということがいえる。

4.3 学習の収束性

強化学習則を示す前に収束性を評価するための尺度を定義する。今、 n 回目の試行におけるタスク系列自体を一つのタスクとみなし、 $\sigma(n)$ とする。このとき、 $\sigma(n) \in L_m(N)$ で、 $L_m(N)$ の各要素を σ_i と表記し、 n 回目の試行におけるそれぞれの等価生起確率を $o_i(n)$ とする。また、等価生起確率ベクトル $o(n)$ を次のように定義する。

$$o(n) = \{o_1(n), o_2(n), \dots, o_r(n)\} \quad (23)$$

ただし、 r はすべての可能なタスク系列の数を表す。このとき次式で与えられる尺度を考える。

$$M(n) \equiv E[\beta(n)|o(n)] \quad (24)$$

$$= \sum_{i=1}^r u_i o_i(n) \quad (25)$$

ここで、

$$u_i = E[\beta(n)|\sigma(n) = \sigma_i] \quad (26)$$

である。 $M(n)$ は n 回目の試行において、実行したタスク系列に対して環境から与えられるペナルティーの期待値を表す。このとき、次式が成り立つならば、学習オートマトンは環境に対して絶対適応するという [29]。

$$E[M(n+1)] < E[M(n)] \quad (27)$$

また強化学習則が絶対適応性を満たす場合、次式も成り立つ。

$$E[o_{opt}(n+1)] > E[o_{opt}(n)] \quad (28)$$

ここで、 o_{opt} は最も望ましいタスク系列、つまり最小の u_i を与えるタスク系列の生起確率を表す。ただし、このことは試行を重ねるに従って必ず最適なタスク系列に収束することを意味

するものではなく、式 (28) は、あくまでも最適なタスク系列を選択する期待値が単調増加するということを示している。その裏付けとして、絶対適応性が満たされる場合、次式が成り立つ。

$$\lim_{n \rightarrow \infty} o_i(n) = 1 \quad \exists_i \quad (29)$$

これは、最終的にはどれかのタスク系列に収束する確率が1であることを示しており、収束するタスク系列が必ずしも最適な系列とは限らない点に注意されたい。しかしながら、学習ゲイン $\lambda(p(n))$ を小さくすることによって、次式もまた成り立つことが知られている。

$$\lim_{\lambda(p(n)) \rightarrow 0} \Pr\{\lim_{n \rightarrow \infty} o_{opt}(n) = 1\} = 1 \quad (30)$$

すなわち、 $\lambda(p(n))$ を十分小さくすることにより、最適なタスク系列を選択する確率が最終的に1となる確率を限りなく1に近づけることが可能となる。この性質は ϵ -最適性と呼ばれている [24]。

4.4 強化学習則

この節では、Narendra らによって提案された階層型のオートマトンに対する P モデル環境下での強化学習則 [24] を基にした、Q モデル環境下での強化学習則について述べる。

第1レベルの更新

第1レベルで $\alpha(n) = \alpha_i$ が選択されたとき、

$$\begin{cases} p_i(n+1) = p_i(n) + \lambda(p(n))(1-\beta(n))(1-p_i(n)) \\ p_j(n+1) = p_j(n)(1-\lambda(p(n))(1-\beta(n))) \quad (j \neq i) \end{cases} \quad (31)$$

第 L レベルの更新

第 L レベルで $\alpha(n) = \alpha_{i_L}$ が選択されたとき、

$$\begin{cases} p_{i_L}(n+1) = p_{i_L}(n) + \lambda_{i_{L-1}}(\hat{p}(n))(1-\beta(n))(1-p_{i_L}(n)) \\ p_{i_{L-1}j_L}(n+1) = p_{i_{L-1}j_L}(n)(1-\lambda_{i_{L-1}}(\hat{p}(n))(1-\beta(n))) \\ p_{j_L}(n+1) = p_{j_L}(n) \quad (\text{for others}) \end{cases} \quad (j \neq i) \quad (32)$$

ここで $\lambda(p(n))$ は、第1レベルにおける学習ゲインであり、 $0 < \lambda(p(n)) < 1$ を満足するものとする。 $\lambda(p(n))$ においては、 n 回目の試行におけるタスクの生起確率に依存しているので、 $p(n)$ が明記されている。また、各レベルにおける学習ゲイン $\lambda_{i_L}(\hat{p}(n))$ は次式のように再帰的に与えられる。

$$\begin{aligned} \lambda_{i_L}(\hat{p}(n)) &= \frac{\lambda_{i_{L-1}}(\hat{p}(n))}{p_{i_L}(n+1)} \\ &= \frac{\lambda(p(n))}{\prod_{k=1}^{L-1} p_{i_k}(n+1)} \end{aligned} \quad (33)$$

いま、 $0 < \lambda_{i_L}(\hat{p}(n)) < 1$ は成り立たないが、 $0 \leq \lambda_{i_L}(\hat{p}(n))(1-\beta(n)) < 1$ は成り立つ点に注意されたい。ま

た、この強化学習則が絶対適応性を満足することは容易に示すことができ、 $\beta(n) = 0$ または $\beta(n) = 1$ を上式に代入することにより文献 [24] で示された、P モデル環境下での学習強化則とまったく同じ式が得られることに注意されたい。このとき、 $0 < \lambda_{i_L}(\hat{p}(n)) < 1$ が満たされることは容易に示すことができ、またこの学習強化則を用いることによって、未知の、つまり、環境の確率分布に関する情報が一切分からない定常ランダム環境に対して前節で述べた絶対適応性が保証されることが証明されている。上記の学習則は、 $\beta(n) = 0$ の場合は確率を更新するが (Reward)、 $\beta(n) = 1$ の場合は何もしない (Inaction) ことから L_{R-I} 学習則と呼ばれている。

4.5 学習制御システムと手順の評価

この節では、前章で示したスーパーバイザと学習オートマトンをリンクした学習制御システムの構成について述べる。また、2章で示した評価規範に基づいた手順の評価法についても述べる。

4.5.1 システム構成

提案する学習制御システムの構成図を Fig. 9 に示す。Fig. 9 において、学習オートマトン、スーパーバイザと制御ペトリネットはそれぞれ同期して動く。Fig. 9 は Fig. 5 と Fig. 8 を組み合わせた構成となっており、制御ペトリネットはスーパーバイザから受けとる制御パターン γ_I と学習オートマトンから受けとる制御パターン γ_E の2種類の制御パターンの論理積をとった制御パターンを入力とすることになる。提案するシステムの動作について以下に述べる。まず初期状態において、学習オートマトンが初期状態でのタスク生起確率に従って、あるタスク指令を出力する。出力されたタスク指令を制御パターン生成部で制御パターン γ_E に変換する。変換された制御パターンは、出力されたタスクに相当するトランジションに付加された外部入力プレースのみにトークンを一つ与え、他の外部入力プレースにはトークンを与えないようなパターンとする。同様にスーパーバイザは初期状態での制御パターン γ_I を出力する。制御ペトリネットは $\gamma = \gamma_E \wedge \gamma_I$ を入力として (このとき γ のなかの1の数とはたかだか一つである)、 $\gamma \neq 0$ ならば発火可能なトランジションを発火させ (つまりタスクを実行し)、学習オートマトン、スーパーバイザと制御ペトリネットはその発火に伴って、それぞれ同期して状態、あるいはトークンを選

Fig. 9 Proposed learning control system

移させる。また、入力 $\gamma = 0$ の場合は、発火可能なトランジションは存在しないため、状態遷移は行わず、学習オートマトンに実行不可能であることを伝える。学習オートマトンは、出力したタスクが実行不可能として reject された場合は、そのタスクの生起確率を強制的に 0 とし、その状態でのそれ以外のタスクの生起確率を均等に上げる。つまり、第 L レベルで $\alpha(n) = \alpha_{i_L} (= \alpha_{i_1 \dots i_L})$ が選択され、それが実行不可能として reject されたとき、

$$\begin{cases} p_{i_L}(n+1) = 0 \\ p_{i_{L-1}j_L}(n+1) = p_{i_{L-1}j_L}(n) + \frac{1}{r_L - 1} p_{i_L}(n) \quad (j \neq i) \end{cases} \quad (34)$$

とし、第 L レベルから再学習を行う。ここで、 r_L は α_{i_L} を出力した状態での実行可能なタスクの数である。この確率更新則を用いる場合、前節で述べた絶対適応性は厳密な意味では保証されないことになるが、上記の確率更新則は生起順序に関する仕様を満たさないタスクを単に取り除くことと等価であるため、収束性に大きな影響を与えることはない。

このシステムの特長は、各タスクの情報に基づいて積極的に良い手順を探そうとするサブシステム（学習オートマトン）と作業環境等に起因するタスクの順序関係に関する仕様を満たしたり、明らかな禁止事項だけは行わせないある意味での消極的なサブシステム（スーパーバイザ）とが個別に実現されている点である。これによって、作業環境などが変更になって生起の順序に関する仕様などに変更があった場合や、評価規範を再設計するような場合でも、システム全体を変更する必要はなく、対応するサブシステムを変更するだけで対応できる。また、両サブシステムとも同じオートマトンの枠組内で構成されているので、互いに整合性が良く、計算機への実装が容易になるという特長を合わせ持つ。

4.5.2 手順の評価システム

本論文では、作業手順の評価値は、ロボットがその手順の実行後、自ら計算するものとする。なお、ここで用いられる評価値としては通常人間が何気なく行っている評価を定量化したものを用いることが望ましいが、現実にはそれは容易ではない。そこで、ここでは評価値を単に各タスクのコストの総和として定めるものとする。その際に、2章で示した評価規範に従ってロボットは実行した作業手順の評価を行うことにする。また、ロボットが行う各タスクのコストにはセンサの精度などに起因する不確かさが存在するとし、本論文では、これらの不確かさを評価値に確率分布を持たせることで表現する。つまり、ロボット自身は、前もって作業手順の評価値は知らないが、その作業の実行後は、ある程度の精度でその評価を行うことが可能であるとする。そして、その評価値に基づいて各タスクの生起確率を強化学習アルゴリズムに従って更新する。なお、コストがこのような不確かさを持った場合、通常の線形計画法は適用不可能である点に注意されたい。

また、学習オートマトンに対して評価値をフィードバックする際に、その値は $0 \leq \beta(n) \leq 1$ とする必要がある。そのためには、各タスクのコストの総和の最大値、および最小値を知る必要がある。そこで、これらの値をより正確に推定するために

ロボットは始めの数十回の試行においては、学習を行わず、初期確率のままシステマティックに可能なすべての分解手順を2度ずつ実行し、そのデータから、各タスクのコストの総和の最大値と最小値を推定し、評価値を正規化することにする。環境がQモデルの場合は、計算された値をそのままフィードバックし、Pモデルの場合は、0.5以上であれば1をフィードバックし、それ以外の場合は0をフィードバックすることにする。なお、環境がSモデルの場合は環境の出力する値の確率分布が区間 $[0, 1]$ で連続的に与えられる必要があるが、本論文ではこのモデルは検討対象外とする。

5. シミュレーション結果

今、Fig.10 に示すようなロボット ($L_1 = 30, L_2 = 120, L_3 = 120, L_4 = 10$) を用いて Fig.2 の作業環境で作業を実行した場合の各タスクのコストの平均値を Table 2 に示す。ここでは、簡単のため、エネルギーに関するコストは省略してある。また、各タスクのコストの確率分布は安定性、操作性に関してそれぞれ分散 σ^2 が 0.01, 0.25 の正規分布に従うものとする。なお、Table 2 では、各コストの単位が明示されていないが、学習オートマトンにフィードバックする際は区間 $[0, 1]$ に正規化するので単位は問題とならない。各コストの計算の詳細は文献[9]を参照されたい。Fig.11 に操作性のみに着目したときの $\alpha_2, \alpha_{22}, \alpha_{221}, \alpha_{2121}, \alpha_{21211}, \alpha_{111231}$ の各タスクの生起確率の履歴を示す。環境はQモデルとした。また、スーパーバイザとしては、Fig.6 と Table 1 の組合わせを採用した。Fig.11 において最初の数十回は、評価値の最大値と最小値を探すために、学習は行っていない。ただし、その数十回のうちにスーパーバイザの禁止するタスクが選択された場合は、式(34)に従って、そのタスクの生起確率を0にし、他の選択可能なタスクの生起確率を上げる。Fig.11 では、15回目の試行で α_{21211} の生起確率が1となっているのがその現れである。また、このときの学習ゲインは $\lambda(p(n)) = 0.05$ とした。操作性のみに着目する場合、これらのタスク系列は最適な系列となり、スーパーバイザの許可する範囲内で学習がうまく行われたことが分かる。また、操作性と、安定性の両方に着目したときの $\alpha_2, \alpha_{21}, \alpha_{211}, \alpha_{2111}, \alpha_{111231}$ の各タスクの生起確率の履歴を Fig.12 に示す。このときも環境はQモデルとし、同じスーパーバイザを採用した。このとき評価値としては正規化された操作性の評価値と正規化された安定性の評価値の平均値をとった。操作性と安定性の両方に着目する場合は、このタスク系列

Fig. 10 6 D.O.F. robot

Table 2 Mean cost value for each task

Task	Stability	Manipulability
1	0.0	0.0
2	0.0	0.0
3	0.5	-7.76425
4	1.0	-4.69400
5	0.5	-6.42850
6	1.5	-6.08125
7	1.5	-4.69400
8	0.0	-5.77600
9	1.5	-7.76425
10	0.0	-17.10450
11	0.0	-13.08225
12	1.5	-6.08125
13	1.5	-6.42825
14	0.0	-5.00850
15	1.0	-6.18050
16	0.5	-17.10450
17	0.0	0.0
18	0.0	0.0
19	0.0	-7.76425
20	0.0	-13.08225
21	1.0	-5.00825
22	0.0	-6.42825
23	1.0	-6.18050
24	0.0	0.0
25	0.0	0.0
26	0.0	-5.48300
27	0.0	-13.33450
28	0.0	-5.7755
29	0.0	-13.84150
30	0.0	-5.10800
31	0.0	0.0
32	0.0	0.0
33	0.0	0.0
34	0.0	0.0

Fig. 11 Convergence of probability I

Fig. 12 Convergence of probability II

Fig. 13 Absolute expediency

が最適となり、この場合も学習がうまく行われていることが分かる。また、この場合操作性のみによる評価と比べて、収束に要する試行回数が増加している。これは、安定性の評価を付け加えることによって、より最適性の強い手順が存在することになったためである。操作性に関するコストの分散 σ^2 を 0.25, 1.0 および 2.0 とし、操作性と安定性の両方に着目した同じシミュレーションを何度か繰り返し、縦軸に $E[M(n)]$ をとった場合の履歴を Fig. 13 に示す（ここでは、500 回の学習シミュレーションの平均をとった）。これを見ると、分散が大きくなるにつれて、つまりタスクの評価の曖昧度が増すにつれて収束速度が劣化するものの、いずれの場合も 4.3 節で述べた絶対適応性が確認できる。ただし、4.3 節でも述べたようにこれらの結果は常に最適なタスク系列が得られることを保証するものではない。 $p_i(n)$ 自身が確率変数となるため、Fig. 11 や Fig. 12 は常に同じ結果が得られるとは限らず、式 (29) が示すように違うタスク系列に収束する場合もある。工学的にはどれぐらいの確率で最適なタスク系列に収束するかという問題は、非常に興味深い問題であるが、その解析的な値を求めることは困難である。

Table 3 $o_{opt}(\infty)$ and time needed for convergence

$\lambda(p(n))$	$o_{opt}(\infty)$	Time needed for convergence
0.01	0.70	2,500
0.05	0.58	600
0.1	0.37	400

そこでここでは、操作性に関するコストの分散を $\sigma^2 = 0.25$ としたときのシミュレーションを通して最適なタスク系列に収束する確率を求めることにする。ただし、操作性と安定性の両方に着目した評価を考え、環境は Q モデルとする。Table 3 に学習ゲインを様々な値にして、最終的に最適なタスクに収束した確率と、収束に必要な平均試行回数を示す。なお、Table 3 中の確率値や平均試行回数は 100 回の学習シミュレーションから求めた。Table 3 から、学習ゲインが小さいほど収束に要する時間は長くなるが、最適なタスク系列に収束する確率は高くなることが分かり、式 (30) で表される ϵ -最適性が確認できる。ただし、これらの値は評価規範と密接な関係があり、例えば、明らかに最適性の強いタスク系列が存在する場合は、Table 3 に示す値よりもより少ない試行回数で、しかもより高い確率で最適なタスク系列に収束する。

なお、環境を P モデルとした場合でもほぼ同様な結果が得られた。

6. ま と め

本論文では、ペトリネットを用いてモデリングされた組立/分解作業手順の学習制御法について検討した。提案した学習制御システムは、タスクの生起順序に関する仕様を満足するスーパーバイザと確率的なあいまいさを持つ評価規範のもとで最適な手順を探索する学習オートマトンとから構成されている。また、操作性と安定性という評価規範のもとで、提案手法の有効性をシミュレーションにより検証した。提案システムの特長は、役割を分担して個別に実現することによって、生起順序に関する仕様が変更になった場合や、評価規範を再設計するような場合、システム全体を再構成する必要はなく、対応するサブシステムを修正するだけで良いという点である。今回提案したシステムは、組立/分解作業に限らず、階層構造を有する他の離散事象システムに対しても適用可能であると考えられ、様々な応用を考えること、および収束速度を速めるための強化学習則 [26] の改良が今後の課題である。

なお、本研究は平成 7 年度科学研究費補助金（奨励研究（A）課題番号：07750489）の援助を受けて行われた。ここに感謝の意を記す。

参 考 文 献

- [1] L.S. Homen de Mello and A.C. Sanderson: "AND/OR Graph Representation of Assembly Plans," IEEE Trans. on Robo. and Auto., vol.6, no.2, pp.188-199, 1990.
- [2] L.S. Homen de Mello and A.C. Sanderson: "Representations of Mechanical Assembly Sequences," IEEE Trans. on Robo. and Auto., vol.7, no.2, pp.211-227, 1991.
- [3] L.S. Homen de Mello and A.C. Sanderson: "A Correct and Complete Algorithm for the Generation of Mechanical Assembly Sequences," IEEE Trans. on Robo. and Auto., vol.7, no.2, pp.228-240, 1991.
- [4] A. Agnetis, C. Arbib, et. al: "Part Routing in Flexible Assembly Systems," IEEE Trans. on Robo. and Auto., vol.6, no.6, pp.697-705, 1990.
- [5] S. Lee and Y.G. Shin: "Assembly Planning Based on Subassembly Extraction," IEEE Proc. Inter. Conf. on Robo. and Auto., pp.1606-1611, 1990.
- [6] T. Cao and A.C. Sanderson: "Task Sequence Planning in a Robot Workcell Using AND/OR Nets," IEEE Proc. Inter. Sympo. on Intell. Cont., pp.239-244, 1991.
- [7] T. Cao, A.C. Sanderson: "Sensor-based Error Recovery for Robotic Task Sequence Using Fuzzy Petri Nets," IEEE Proc. Inter. Conf. on Robo. and Auto., pp.1063-1069, 1992.
- [8] 包原, 稲葉他: "ペトリネットを用いた組立作業計画-線形計画法による探索-", 計測自動制御学会論文集, vol.29, no.4, pp.461-469, 1993.
- [9] 稲葉, 包原他: "組立・分解作業のネットによる表現とその評価関数の一設定法", 日本ロボット学会誌, vol.12, no.2, pp.251-261, 1994.
- [10] 川瀬, 包原他: "自動組立手順生成のためのペトリネットの段階的詳細化", 日本機械学会論文集 C, vol.60, no.571-C, pp.1105-1112, 1994.
- [11] 鈴木, 大熊: "自動組立システムのスーパーバイザ制御", 計測自動制御学会論文集, vol.30, no.5, pp.584-593, 1994.
- [12] K.T. Seow and R. Devanathan: "A Temporal Framework for Assembly Sequence Representation and Analysis," IEEE Trans. on Robo. and Auto., vol.10, no.2, pp.220-229, 1994.
- [13] F. Avnaim, J.D. Boissonnat and B. Faverjon: "A Practical Exact Motion Planning Algorithm for Polygonal Objects amidst Polygonal Obstacles," IEEE Proc. Inter. Conf. on Robo. and Auto., pp.1656-1661, 1988.
- [14] R.H. Wilson: On Geometric Assembly Planning. Ph.D. Thesis, Stanford University, 1992.
- [15] 吉川: "ロボットアームの可操程度", 日本ロボット学会誌, vol.2, no.2, pp.63-67, 1984.
- [16] P.J. Ramadge and W.M. Wonham: "Supervisory Control of A Class of Discrete Event Processes," SIAM J. Cont. & Opt., vol.25, no.1, pp.206-230, 1987.
- [17] W.M. Wonham and P.J. Ramadge: "On the Supremal Controllable Language of a Given Language," SIAM J. Cont. & Opt., vol.25, no.3, pp.637-659, 1987.
- [18] P.J. Ramadge and W.M. Wonham: "Modular Feedback Logic for Discrete Event Systems," SIAM J. Cont. & Opt., vol.25, no.5, pp.1202-1218, 1987.
- [19] A.F. Vaz and W.M. Wonham: "On Supervisor Reduction in Discrete-Event Systems," Int. J. Cont., vol.44, no.2, pp.475-491, 1986.
- [20] 潮, 松本: "離散事象駆動型システムにおける可制御発火系列", 計測自動制御学会論文集, vol.24, no.2, pp.156-161, 1988.
- [21] 潮, 松本: "外部入力プレース付きペトリネットにおける論理型制御と状態フィードバック", 計測自動制御学会論文集, vol.24, no.11, pp.1195-1200, 1988.
- [22] L.E. Holloway and B.H. Krogh: "Synthesis of Feedback Control Logic for a Class of Controlled Petri Nets," IEEE Trans. on Auto. Cont., vol.35, no.5, pp.514-523, 1990.
- [23] 市川, 横山他: "事象駆動型システムの制御-無競合ペトリネットの可到達性と制御-", 計測自動制御学会論文集, vol.21, no.4, pp.324-330, 1985.
- [24] K.S. Narendra and M.A.L. Thathachar: "LEARNING AUTOMATA," Prentice-Hall, 1989.
- [25] R. Simha and J.F. Kurose: "Relative Reward Strength Algorithms for Learning Automata," IEEE SMC, vol.19, no.2, pp.388-398, 1989.
- [26] 最上, 馬場 他: "非定常環境中で動作する階層構造学習オートマトンの新しいアルゴリズム", 計測自動制御学会論文集, vol.30, no.8, pp.953-958, 1994.
- [27] S. Lakshmivarahan and M.A.L. Thathachar: "Absolutely Expedient Learning Algorithms for Stochastic Automata," IEEE SMC, vol.3, no.3, pp.281-286, 1973.
- [28] N. Baba: "On the Learning Behavior of the $SLR-I$ Reinforce-

ment Scheme for Stochastic Automata," IEEE SMC, vol.6, no.8, pp.580-582, 1976.

- [29] S. Lakshmivarahan and M.A.L. Thathachar: "Absolute Expediency of Q-model and S-model Learning Algorithms," IEEE SMC,

vol.6, no.3, pp.222-226, 1976.

- [30] 村田：ペトリネットの解析と応用. 近代科学社, 1992.

- [31] S. Eilenberg: Automata, Languages, and Machines, Vol.A. Academic Press, 1974.

鈴木達也 (Tatsuya Suzuki)

1964年1月3日生。1991年名古屋大学大学院博士課程後期課程電子機械工学専攻修了。同年名古屋大学工学部電気学科助手。1995年同講師，現在に至る。ロバスト制御，学習制御を用いたモーションコントロール，離散事象システム理論とその応用，確率オートマトンに基づいた知的システムに関する研究に従事（工学博士）。1994年システム制御情報学会奨励賞，1995年電気学会論文賞をそれぞれ受賞。電気学会，計測自動制御学会，システム制御情報学会，IEEEの各会員。（日本ロボット学会正会員）

稲葉昭夫 (Akio Inaba)

1961年9月26日生。1984年岐阜大学工学部電子工学科卒業。同年岐阜県工業技術センター勤務。1991年岐阜県金属試験場勤務，現在に至る。CAD，CG，メカトロニクス，タスクプランニングに関する研究に従事。情報処理学会，電子情報通信学会の各会員。（日本ロボット学会正会員）

残間忠直 (Tadanao Zanna)

1972年5月2日生。1995年名古屋大学工学部電気学科卒業。同年同大学大学院工学研究科修士課程進学，現在に至る。離散事象システムに関する研究に従事。電気学会，計測自動制御学会の各会員。

大熊 繁 (Shigeru Okuma)

1948年2月16日生。1972年名古屋大学大学院修士課程電気工学専攻修了。1974年ケース・ウェスタン・リザーブ大学大学院修士課程システム工学専攻修了。1977年名古屋大学大学院博士課程電気工学専攻修了。同年，名古屋大学工学部電気学科助手，講師，助教授を経て1990年同教授，現在に至る。主としてロボティクス，パワーエレクトロニクス，創発型ソフトコンピュータに関する研究に従事（工学博士）。1992年IECON Best Paper Award，1994年精密工学会論文賞，1995年電気学会論文賞をそれぞれ受賞。電気学会，システム制御情報学会，日本機械学会，IEEEの各会員。（日本ロボット学会正会員）