博士（工学）論文

報酬に基づく創発的制御手法に関する研究
－計算生態学アプローチと強化学習の応用－

山崎　達志

大阪大学 大学院基礎工学研究科
システム人間系専攻 システム科学分野

2003年3月

概要

近年，環境への適応性・柔軟性を持った新たな制御手法の一つとして，創発的手法が注目を集めている．創発的手法には，遺伝的アルゴリズム，ニューラルネットワーク，免疫システム，強化学習，計算生態学アプローチなどがあり，様々な研究がなされている．本論文では，創発的制御手法の中から報酬に基づく創発的手法に着目し，これを用いた制御手法を提案する．報酬に基づく手法では，環境からなんらかの評価規範を報酬，あるいは利益として受け取ることで制御器が自身の制御方策を獲得していく．情報を報酬という形に集約して扱っているため，多くの問題に比較的容易に適用できる．

本論文は，用いた創発的手法から大きく二つに分けられる．前半では計算生態学アプローチを，後半では強化学習に基づいた制御手法について述べている．

前半においては，計算生態学アプローチに基づく制御手法と，ネットワークルーチングへの応用について述べる．計算生態学アプローチでは，計算生態学モデルと呼ばれる，Huberman と Hogg が提案したエージェント集団のマクロ的挙動を表す数理モデルを用いる．このモデルでは，一つのエージェント集団と複数の資源が存在し，集団内のエージェントがより高い利益を得られるように資源に対し競合する様を表現しているが，情報の不確かさや時間遅れによってはエージェント集団の振る舞いにカオスが生じることが報告されている．Hogg と Huberman は，このカオス現象を抑制するために，エージェントに多種の戦略を持たせ資源から得られる報酬に応じて採用する戦略を動的に変更する報酬メカニズムを導入した．これを Hogg-Huberman モデルと呼ぶ．本論文では，この Hogg-Huberman モデルを複数のエージェント集団に拡張することで，マルチエージェントシステムにおける資源配分を表すモデルとしての定式化を行う．これにより，本来，エージェント集団の振舞いを表すものとして生まれた計算生態学モデルを，制御のための手法として用いる枠組みを提供している．

そして，この定式化に基づき，計算生態学モデルを用いたネットワークルーチング方式を提案する．ネットワーク上の各ノードについてパケットをエージェント，次ノードへのリンクを資源とみなし，拡張した Hogg-Hubermann モデルを適用する．そして，各ノードに到着したパケットの各リンクの選択割合の決定を，このモデルに従って行うことでルーチングを行う．

論文の後半では，強化学習を用いた離散事象システムの最適スーパバイザ制御について述べる．強化学習では，環境から受け取る報酬をもとに，学習者はより良い行動政策を獲得するように学習を行う．また，離散事象システム理論におけるスーパバイザ制御では，制御仕様を満たすように，スーパバイザが，システムに生起を許容する事象の集合を制御パターンとして提示する．しかし，制御対象と仕様の言語あるいはオートマトンによる正確な記述が必要なことが一つの問題であった．本論文では，制御パターンの与え方をシステムから受け取る報酬をもとに強化学習を通じて行うことにより，不確かな環境や陽でない制御仕様のもとでも，事象の生起／禁止コストおよび仕様を考慮した最適スーパバイザを求める手法を提案する．このとき，いくつかの仮定のもとで，学習速度の向上を図っている．提案手法により，環境が不確かであったり変化する場合に対してもスーパバイザ制御を適用するための一つの方法論を示している．最初に，システムの状態がスーパバイザから完全に観測できる場合について述べ，次にシステム内部での事象の生起を部分的にしか観測できない場合へと拡張する．

以上のように，本論文は，創発的手法の制御への応用に関し，それぞれ計算生態学アプローチと強化学習という二つの方法論に基づいた，新たな方式を提案するものである．

謝辞

本研究を遂行するにあたり，大阪大学大学院基礎工学研究科 潮 俊光 教授には，終始適切なご指導ご鞭撻をいただきました．また先生には，私が研究室に配属されてからの長きに渡り，指導教官として適切なご助言と有益な議論を通じ，研究者としてのさまざまな教えを受けました．ここに深く感謝するとともに厚くお礼申し上げます．

本論文の副査として，査読のために貴重な時間を割いていただき，適切なご助言とご意見をいただきました，大阪大学大学院基礎工学研究科 田村 坦之 教授ならびに藤井 隆雄 教授に深く感謝いたします．

また，有益なご助言を賜りました大阪大学大学院基礎工学研究科 山本 茂 助教授，小林 啓吾 助手に感謝いたします．事務に関しては西脇 加代美 さんにお世話になりました．ここにお礼申し上げます．

また，ともに学生時代を過ごした潮研究室の皆様に深く感謝の意を表します．

現在の勤務先である関西学院大学理工学部情報科学科ならびに実験担当の皆様には，本論文の執筆にご理解賜りましたことに厚くお礼申し上げます．特に，情報科学科 北橋 忠宏 教授には，本論文に対するご助言をいただきましたことに感謝します．

最後に，多くの負担がかかるのにもかかわらず，大学，大学院と長きに渡って通わせていただき，暖かい励ましをいただいた両親に心より感謝します．

目次

第 1 章

緒論

1.1 研究の背景と目的

システムが大規模化複雑化していくに伴い，従来の制御手法をそのまま適用していったのでは，環境への適応性，柔軟性といった面で困難に直面する場合も少なくない．そのため，このようなシステムを扱う新たな手法が求められている．その一つとして，創発的手法を用いた様々な研究がなされている．創発的手法として，遺伝的アルゴリズム，ニューラルネットワーク，免疫システム，強化学習，計算生態学アプローチなど多くの手法が知られている．これらを用いた手法の特徴として，環境の変化に対しシステムの構成要素が相互に作用し，自律的に対応することで，全体として制御を実現している．主に人工知能の分野から生まれてきたこれらの手法を制御問題へと適用していくことは，新たな制御手法の開発に有益であると考えられる．

創発的手法の中には，行動に対する評価規範を報酬，あるいは利益という形で表現して受け取り利用するものがある．現実のシステムに当てはめて考えるときに，制御器となる部分と環境として扱う部分に分け，制御方策を報酬に基づいて動的に決定していくという構成は，汎用的で様々な対象への応用に適していると考えられる．個々の問題に応じ，望ましい特性を報酬の中に盛り込めばよい．また，環境から得られる情報を報酬として用いることができれば，環境そのものの詳細を制御器は知らなくても済む．このことは，容易にシステム全体を把握することができない大規模システムにおいては大きな利点となる．

報酬に基づく創発的手法の中に，計算生態学アプローチと強化学習がある．計算生態学アプローチは，計算生態学モデルと呼ばれる，Huberman と Hogg が提案した

エージェント集団のマクロ的挙動を表す数理モデルを用いる．この数理モデルでは，複数のエージェントが，より良い利益を得られるように共通の資源に対し競合する様子を表現している．そして，エージェントに多種の戦略を持たせ，得られる報酬により動的に戦略を変更する報酬メカニズムの導入により，各エージェントの受け取る利益が等しくなる資源の配分状態が達成される．また，環境の変化に対して自律的に資源の選択割合を変化させる．このことは，計算生態学モデルが資源の配分を決定する手法として有望であることを示していると考えられるが，計算生態学アプローチの工学的応用に関する研究は充分とはいえず，様々な視点からの研究が望まれる．

強化学習は，環境から受け取る報酬をもとに学習者がより良い行動指針を見つけ出す学習手法である．主に人工知能における機械学習の分野において長く研究されてきていたが，特に近年，動的計画法との関連から数学的基礎を与えられたこともあって，注目を集めている．一方，離散事象システムにおける制御の方法論として，Ramadge と Wonham によって提案されたスーパバイザ制御がある．スーパバイザ制御では，スーパバイザが，与えられた制御仕様に基づいてシステムの可制御な事象の生起／禁止を決定し，制御パターンとして提示することによって制御を行う．このとき，従来は形式言語あるいはオートマトンによる対象システムと制御仕様の正確な記述が必要であったが，一般にはそれは簡単なことではなく，現実の問題へ適用するときの一つの問題となっていた．そのため，スーパバイザ制御を容易に現実の問題に適用するための手法の開発は重要である．

本論文では，以上の観点に基づき，創発的手法の中から特に，環境から得られる情報を主に報酬という形で集約して表現するタイプの手法として，計算生態学アプローチと強化学習に着目した．そして，

1. 計算生態学アプローチをマルチエージェント系の資源配分問題へと適用し，ネットワークの経路選択制御へと応用すること
2. 離散事象システムにおけるスーパバイザ制御問題に対し，強化学習を用いた最適スーパバイザ制御手法を提案すること

を目的として，著者のこれまでの研究成果をまとめたものである．

1.2 論文の構成

以下では、本論文の構成について述べる．第 2 章および第 3 章では，計算生態学モデルと，これを用いた制御手法について述べる．第 2 章では，まず計算生態学モデルについて述べ，マルチエージェント系の制御問題への拡張と定式化を行う．第 3 章では，第 2 章で述べた定式化に基づき，ネットワークの経路選択問題への応用例を示す．第 4 章および第 5 章では，強化学習を用いた最適スーパバイザ制御について述べる．まず第 4 章において，スーパバイザ学習システムの基本的な構成と，スーパバイザが制御対象である離散事象システムの状態を完全に観測できる場合の制御法について述べる．そして第 5 章では，これを部分観測な場合に拡張する．最後に，第 6 章では，結論と今後の課題について述べる．また，付録 A として従来のスーパバイザ制御について，付録 B として強化学習について簡単に述べる．

各章の概要は以下のとおり．

■ 第 2 章：計算生態学アプローチに基づく制御手法

本章では，計算生態学モデルと，これに報酬メカニズムを導入した Hogg-Huberman モデルを紹介する．そして，複数のエージェント集団に対しこのモデルを拡張し，計算生態学アプローチに基づく制御手法を提案する．

最初に，2.2 節で計算生態学モデルについて述べる．計算生態学モデルは，エージェント集団のマクロ的挙動を表す数理モデルである．Huberman と Hogg によって提案されたこのモデルは，複数の同質のエージェントが共通の資源に対して競合しつつ資源選択を行っている様子を，1 階の差分方程式で表現している．次に，2.3 節で計算生態学モデルの拡張である Hogg-Huberman モデルについて述べる．Hogg と Huberman によって提案されたこのモデルでは，エージェントに多種の戦略を持たせ，受け取る利益に応じて戦略を変更する報酬メカニズムが導入されている．

以上のモデルでは，エージェント集団は一つであったが，一般のマルチエージェントシステムでは，異なる利益関数を持った複数のエージェント集団が存在すると考えられる．そこで 2.4 節で，Hogg-Huberman モデルを複数のエージェント集団が存在する場合に拡張する．そして，複数のエージェント集団が共通の資源に対し競合するという枠組みを，資源配分問題として捉えることで，この拡張したモデルを資源配分

の制御手法として用いることを提案する．また，2.5節で本章の内容に関するいくつかのシミュレーション結果を示す．

■ 第3章：計算生態学アプローチの経路選択問題への応用

本章では，2章で示した計算生態学アプローチに基づく制御手法を，ネットワークの経路選択問題に応用する．

最初に3.2節で，本章で対象とするネットワークモデルを示す．ネットワーク上の各ノードは送信用のバッファを持つリンクでつながれ，パケットが各ノード到着時に次に選択するリンクを決定していくことで，最終目的ノードまで運ばれる．次に3.3節で，2.4節で示した拡張したHogg-Hubermanモデルを用いて，各ノードにおけるリンクの選択を複数のエージェント集団間の資源配分問題としてとらえて定式化する．このとき，パケットを最終目的地別に分けたものをそれぞれ異なるエージェント集団として用いた．また，各エージェントの利益関数は，リンクの待ち時間，リンク以遠の混み具合，目的ノードまでの最短ホップ数の3つの評価値の重み付き和を用いた．各ノードに到着したパケットは，モデルにより決定されるリンクの選択割合に基づき次のリンクを選択する．ここで，Hogg-Hubermanモデルの働きにより，各エージェント集団が受ける利益が等しくなり，トラヒックの変化に対しても動的に資源の選択割合が調整されていく．そして，3.4節でコンピュータシミュレーションにより，提案手法の有効性を検証する．

■ 第4章：強化学習を用いた離散事象システムの最適スーパバイザ制御
－完全観測の場合－

本章では，強化学習を用いたスーパバイザの構成法を提案する．スーパバイザ制御では，制御対象である離散事象システムに対し，形式言語で与えられた制御仕様を満足するように，スーパバイザが生起を許容する事象の集合を制御パターンとして提示する．スーパバイザ構成のためには，制御対象と制御仕様の正確な記述が必要であったが，一般には簡単ではない．また，論理的な仕様のみを考慮し，事象のコストに対しての考慮はされていない．

まず4.2節でシステムの数理モデルをBellman最適方程式により示し，いくつかの仮定を導入する．そして4.3節では，離散事象システムに対する制御パターンの与え方を，強化学習を用いて学習させるアルゴリズムを提案する．このとき，仮定を利用

することで，学習速度の向上を図っている．提案アルゴリズムでは，不確かな環境や陽でない制御仕様の下でも学習を通じて，事象の生起/禁止コスト及び仕様を考慮した最適スーパバイザを求める．4.4 節で，例題として n 本腕バンディット問題と猫とねずみの迷路の問題を考え，計算機実験によって提案手法の有効性について調べる．

■ 第 5 章：強化学習を用いた離散事象システムの最適スーパバイザ制御
ー部分観測の場合ー

4 章で提案した，強化学習を用いた離散事象システムの最適スーパバイザ制御は，離散事象システムの状態がスーパバイザ側から完全に観測できる場合について考えていた．しかし一般には，様々な制約から完全観測の状況は成り立たない．5 章では，離散事象システム内部の生起事象が，スーパバイザにとって部分観測的にしか観測できない場合について，強化学習を用いたスーパバイザ制御法を提案する．

まず 5.2 節では，部分観測を導入するために，離散事象システムの生起事象に対してスーパバイザが観測する事象を考え，離散事象システムの状態をスーパバイザが推定する仕組みを導入する．そして，スーパバイザの観測に対する Bellman 最適方程式を示す．このとき，4 章と同様の仮定を導入する．5.3 節では，5.2 節で考えたシステムに対し，部分観測の場合のスーパバイザ学習のアルゴリズムを提案する．5.4 節で，例題として猫とねずみの迷路の問題に対する，計算機実験によって提案手法の有効性について調べる．

第2章

計算生態学アプローチに基づく制御手法

2.1 緒言

本章では，計算生態学モデルおよびその拡張について述べる．そしてマルチエージェントシステムにおける動的な資源配分問題としての定式化と，これを用いた制御方式について述べる．

近年，分散環境下での耐故障性，柔軟性をもつシステムの1つとしてマルチエージェントシステムが注目されている [12, 20]．マルチエージェントシステムは，さまざまな問題に応用されているが，そのひとつとして，人間の経済活動をモデル化することにより分散システムにおける資源配分を行う市場モデル [15] を利用した様々な資源配分問題への適用の試みがなされている [5, 14, 30, 32]．さらに，マルチエージェントシステムにおけるエージェント集団のマクロ的挙動を表す数理モデルとして計算生態学モデルが Huberman と Hogg によって提案されている [9]．このモデルでは1つのエージェント集団内のエージェントが複数の資源に対して競合し，資源選択を行うが，このとき情報の不確かさや時間遅れによってはエージェント集団の振る舞いにカオスが発生することが報告されている．Hogg と Huberman はカオス現象を平衡状態に抑制するために，各エージェントに多種の戦略を持たせ資源から得られる報酬に応じて採用する戦略を変更するという報酬メカニズムを導入した [8]．ここではこのモデルを Hogg-Huberman モデルと呼ぶことにする．Hogg-Huberman モデ

ルにおいては，一つのエージェント集団が報酬メカニズムのもとで，それぞれの受け取る利益が等しくなるように資源配分を行う．また，環境の変化に対しても適応して振舞うことが示されている．そのため，このモデルは環境に適応しつつ資源配分を行うモデルとして有効なものであるといえる．

Hogg-Huberman モデルについて現在，いくつかの研究がなされている．今森らは，離散時間 Hogg-Huberman モデルについて戦略としてバイアスを用いる場合の不動点の性質について解析を行っている [10]．潮らは，計算生態学モデルの応用例として生産システムのパーツ配分法を提案している [28]．これらの結果は 2 章，3 章の成果とともに解説記事としてまとめられている [27]．他に Youssefmir らは計算生態学モデルで生じるバーストと，Hogg-Huberman モデルによる抑制のメカニズムについて解析している [39]．また，柴田らは 2.4 節で示す拡張した Hogg-Huberman モデルを用いて，生産者の市場に対する好みを考慮した市場選択問題の定式化を行っている [23]．

以下，2.2 節では，Huberman と Hogg が提案した計算生態学モデルについて，2.3 節ではその拡張である Hogg-Huberman モデルについて述べる．次に 2.4 節で，Hogg-Huberman モデルを複数のエージェント集団に拡張して適用することにより，計算生態学アプローチに基づく制御手法を提案する．本章の内容についてのシミュレーションを 2.5 節で示す．最後に，2.6 節で本章のまとめを述べる．

2.2 計算生態学モデル

本節では，マルチエージェントシステムに対する数理モデルの一つである，計算生態学モデル (Computational ecology model) [9] について述べる．

2.2.1 計算生態学アプローチ

最初に簡単に概念的な説明をしておく．

現実においても計算機上でも，環境から得られる情報に時間遅れや不確かさが存在しないとは限らない．Huberman と Hogg は，そのような状況を扱える一つのモデルとして計算生態学モデルを提案した．

計算生態学の基本モデルは図 2.1 に示すように，複数の資源 (resource) と，多数の

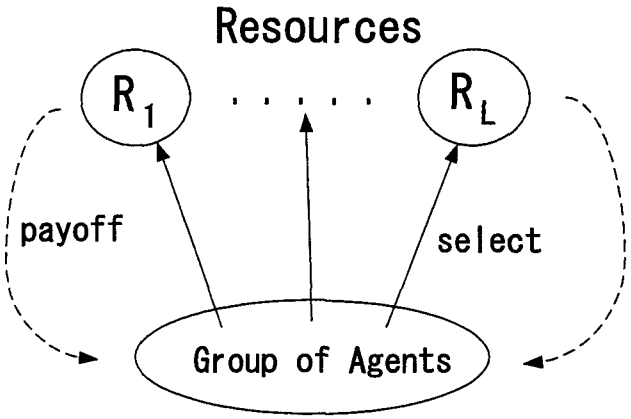


図 2.1: 計算生態学モデル

エージェント (agent) からなる一つのエージェント集団から構成されている.

この単純化されたマルチエージェントシステムにおいて, 各エージェントはそれぞれ独立に資源を選択し, 資源の配分状況に依存して資源から利益 (payoff) を受ける. このエージェントは, より受ける利益が多くなるように資源の選択割合を動的に変更する. この際, エージェントが受ける利益について情報の時間遅れや不確かさを持たせることにより, 不完全な情報を基にエージェントが意志決定を行うという状況を表現できる.

Huberman らはこのようなエージェントを数理モデルとして表現し, これよりエージェント集団全体としてのマクロ的挙動を表す数理モデルを導出した. これが計算生態学モデルである.

2.2.2 計算生態学モデル

もともとの計算生態学モデルは連続時間系において表された微分方程式からなる. しかし, このことがシステムの解析の大きな困難となる. また, 現実のシステムにこのモデルを適用する際には, 状態の変更は連続的ではなく一定時間ごとに離散的に行われることが多いと考えられる. それゆえここでは, 解析が比較的容易で, 現実のシステムにも容易に適用可能な離散時間計算生態学モデルを示し, 以後の議論においては, この式を基本に考えることにする [28].

離散時間計算生態学モデルのダイナミクスは, 次式で表される.

$$f_r(k+1) = f_r(k) + \alpha\left(\rho_r(k) - f_r(k)\right) \tag{2.1}$$

ここで，各パラメータの意味は以下の通りである．なお，以後も個々に明示はしていないが，確率や割合としたパラメータについては値域はその要件 (0 以上 1 以下，総和を取れるものについては総和が 1) の制約を受けている．

- k : 時刻 $(k = 0, 1, 2, \cdots\cdots)$.
- $f_r(k)$: 時刻 k でエージェントが資源 r を選択する割合.
- α : 資源の選択を判断し直す割合.
- $\rho_r(k)$: 時刻 k で資源 r を選択する確率.

(2.1) 式によって，エージェント集団の選択する資源の割合の変化が，ρ_r によって表される資源の選択確率と実際の選択割合 f_r の差をもとに決められるというダイナミクスが表されている．これより ρ_r は f_r の目標値となっているといえるが，この ρ_r そのものを現在の環境から推定したエージェント集団の利益をもとに決定することにより，ρ_r をエージェント集団にとっての，不確かさを含んだ推定目標値とみなすことができる．またこのことにより，計算生態学モデルにおいては通常の制御と異なり，目標値そのものも所与のものでない不確かさのなかで，エージェント集団が目標値そのものも推定しつつより良い状態を目指す挙動を示しているととらえることができる．

資源が 2 つの場合の ρ_1 の一例を次に示す．

$$\rho_1(k) = \frac{1}{2}\left(1 + \mathrm{erf}\left(\frac{G_1(f_1(k-\tau)) - G_2(f_1(k-\tau))}{\sqrt{2}\sigma}\right)\right) \tag{2.2}$$

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt \tag{2.3}$$

但し，G_1 は，資源 1 を，G_2 は資源 2 をそれぞれ選択したときの利益，τ は情報の時間遅れ，σ は情報の不確かさの量，erf は誤差関数を示す．誤差関数は図 2.2 に示すような関数で，$+1$ ，-1 が漸近線となるという特徴を持つ．また，資源は 2 個であるから明らかに $\rho_2 = 1 - \rho_1$ であり，f_2 の挙動は f_1 と対になるものであるから，ここでは f_1 のみを問題としている．この ρ_1 では，エージェントは利益の大きい資源

をより高い確率で選択し，また，資源から受ける利益が等しいとき，ρ_1 と ρ_2 は等しくなる．

Huberman と Hogg は ρ_r に情報の時間遅れや不確かさといった要素を持たせることにより，システムの持つ情報の不確実性や遅延を表現した．そして，情報の時間遅れ τ を増加させた場合や，情報の不確かさが小さい場合に，f_r の平衡点が不安定になり，f_r の挙動にカオスが発生することを示した．

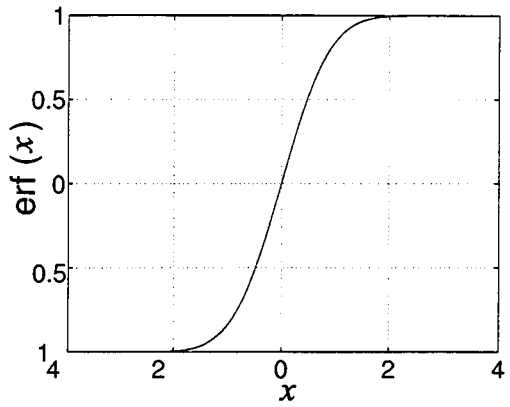


図 2.2: 誤差関数 (erf(x))

2.3 Hogg-Huberman モデル

本節では，計算生態学モデルの拡張である Hogg-Huberman モデル [8] について述べる．

2.3.1 報酬メカニズム

Hogg と Huberman は，(2.1) 式で表される計算生態学モデルにおいて生じるカオス現象の安定化のため，エージェントに多種の戦略を持たせ，資源選択により得られる利益に応じてエージェントの取る戦略を動的に変更させるという報酬メカニズムを導入した．以後，(2.1) 式の計算生態学モデルに報酬メカニズムを持たせたモデルを，Hogg-Huberman モデルと呼ぶことにする．

Hogg-Huberman モデルの概念図を図 2.3 に示す．このように一つのエージェント集団がさらにその採用する戦略によって複数のグループに分かれ，それぞれの戦略に従って資源を選択している．ある戦略を取るエージェントがより利益を多く得ること

ができるとき，報酬メカニズムによりその戦略をとるエージェントの割合が増加する．これにより，エージェント集団のより良い戦略の分布へと自動的に向かうメカニズムが提供されている．エージェント集団全体として，各エージェントの受ける利益が等しくなるような資源配分状態と，システムの挙動の安定化を目指す．

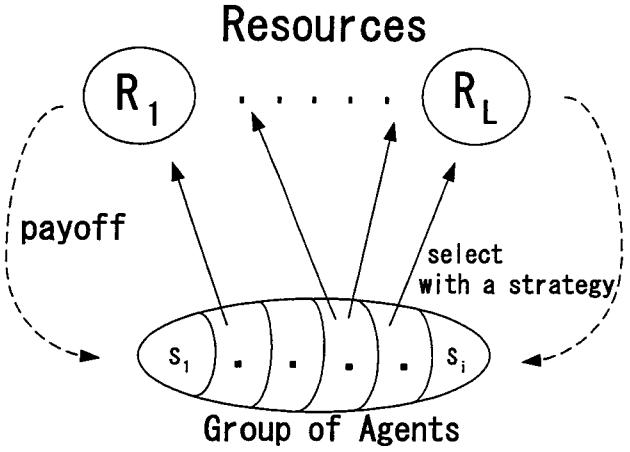


図 2.3: Hogg-Huberman モデル

この Hogg-Huberman モデルのダイナミクスは，次式で表される．

$$f_{rs}(k+1) = f_{rs}(k) + \alpha\left(f_s^{str}(k)\rho_{rs}(k) - f_{rs}(k)\right) + \gamma\left(f_r^{res}(k)\eta_s(k) - f_{rs}(k)\right) \quad (2.4)$$

$$f_r^{res}(k) = \sum_s f_{rs}(k), \quad f_s^{str}(k) = \sum_r f_{rs}(k)$$

$$\eta_s(k) = \frac{\sum_r f_{rs}(k)G_r(k)}{\sum_r f_r^{res}(k)G_r(k)}$$

各パラメータの意味は以下の通りである．

- k : 時刻 $(k = 0, 1, 2, \cdots\cdots)$.
- $f_{rs}(k)$: 時刻 k で資源 r を使い，戦略 s をとるエージェントの割合.
- $f_r^{res}(k)$: 時刻 k で資源 r を使うエージェントの割合．(2.1) 式における f_r に相当する.
- $f_s^{str}(\mathrm{k})$: 時刻 k で戦略 s をとるエージェントの割合.
- $\rho_{rs}(\mathrm{k})$: 時刻 k で戦略 s をとるエージェントが資源 r を選択する確率.
- α : エージェントが資源の選択を判断し直す割合.

- γ : エージェントの行いが報われる割合.
- $\eta_s(k)$ 時刻 k で新しいエージェントが戦略 s をとる確率.
- G_r (k): 時刻 k で資源 r を選択したときの利益.

(2.4) 式において，右辺第 2 項が元の計算生態学モデルに対応し，第 3 項が報酬メカニズムに対応している．η_s の作用により，それぞれの戦略に対し，得ている利益の大きさに比例してエージェントに報酬が与えられ，より有効な戦略を採用するエージェントの割合が増加していく．

(2.4) 式を戦略 s と，資源 r でそれぞれ和をとると，(2.5)，(2.6) 式になり，それぞれが，エージェントの資源に関するダイナミクスと，戦略に関するダイナミクスを表している．この結果からも，Hogg-Huberman モデルがもとの計算生態学モデルの枠組みでそれぞれの資源を選択するエージェントを，さらに内部的に戦略というもので複数に分割していることが分かる．また，(2.6) 式は資源 r に依存しないので戦略の分布は報酬メカニズムによってのみ変更されることが分かる．

$$f_r^{res}(k+1) = f_r^{res}(k) + \alpha\left(\sum_s f_s^{str}(k)\rho_{rs}(k) - f_r^{res}(k)\right) \tag{2.5}$$

$$f_s^{str}(k+1) = f_s^{str}(k) + \gamma\left(\eta_s(k) - f_s^{str}(k)\right) \tag{2.6}$$

Huberman と Hogg は，戦略 s の実現方法として，s を ρ_r の中の情報の時間遅れに対応させたものと，バイアス値に対応させたものの 2 つを示している．資源が 2 つときの s の与え方を次に示す．

- 情報の時間遅れに対応させた場合

$$\rho_{1s}(k) = \frac{1}{2}\left(1 + \mathrm{erf}\left(\frac{G_1(f_1^{res}(k-\tau+s)) - G_2(f_1^{res}(k-\tau+s))}{\sqrt{2}\sigma}\right)\right) \tag{2.7}$$

- バイアス値に対応させた場合

$$\rho_{1s}(k) = \frac{1}{2}\left(1 + \mathrm{erf}\left(\frac{G_1(f_1^{res}(k-\tau)) + s - G_2(f_1^{res}(k-\tau))}{\sqrt{2}\sigma}\right)\right) \tag{2.8}$$

戦略を情報の時間遅れに対応させた場合，各戦略は，どれぐらい前の時刻の情報をもとに ρ_r を決定するかという点での異なる基準に対応する．また，バイアス値に対

応させた場合では，戦略は，資源から受け取る利益に対して，各戦略がどのように修正して見積もるかという点で異なる基準を与えている．

2.4 計算生態学アプローチに基づく制御手法

前節で示した Hogg-Huerman モデルでは，一つのエージェント集団に属する複数のエージェントが共通の資源に対し競合している様を表現していた．そこでは，いくつかの異なる戦略で分けられているものの，受け取る利益をどのように評価するかの基準はそれぞれのエージェントで共通であった．しかし，一般には，エージェントの持つ評価基準は同一でない．この状況を扱えるように，エージェント集団として，利益に対する評価基準が同一であるエージェント毎に，複数のエージェント集団 $A_i(i = 1, 2, \cdots\cdots, N)$ を用意する．そして，(2.4) 式で表現されている Hogg-Huerman モデルを以下のように拡張する．また，図 2.4 は，複数のエージェント集団が，共通の資源に対して競合する様子を概念的に示している．

$$f_{irs}(k+1) = f_{irs}(k) + \alpha\left(f_{is}^{str}(k)\rho_{irs}(k) - f_{irs}(k)\right) + \gamma\left(f_{ir}^{res}(k)\eta_{is}(k) - f_{irs}(k)\right) \tag{2.9}$$

$$f_{ir}^{res}(k) = \sum_s f_{irs}(k), \quad f_{is}^{str}(k) = \sum_r f_{irs}(k)$$

$$\eta_{is}(k) = \frac{\sum_r f_{irs}(k)G_{ir}(k)}{\sum_r f_{ir}^{res}(k)G_{ir}(k)}$$

ここで，各変数の意味は以下の通りである．

- k : 時刻 $(k = 0, 1, 2, \cdots\cdots)$.
- A_i: 異なるエージェント集団を表すラベル $(i = 1, 2, \cdots\cdots, N)$.
- $f_{irs}(k)$: 時刻 k で，エージェント集団 A_i における，資源 r を使う戦略 s のエージェントの割合.
- $\rho_{irs}(k)$: 時刻 k で，A_i における，戦略 s のエージェントについて資源 r を選択する確率.
- $f_{ir}^{res}(k)$: 時刻 k で，A_i における，資源 r を使うエージェントの割合.
- $f_{is}^{str}(k)$: 時刻 k で，A_i における，戦略 s のエージェントの割合.

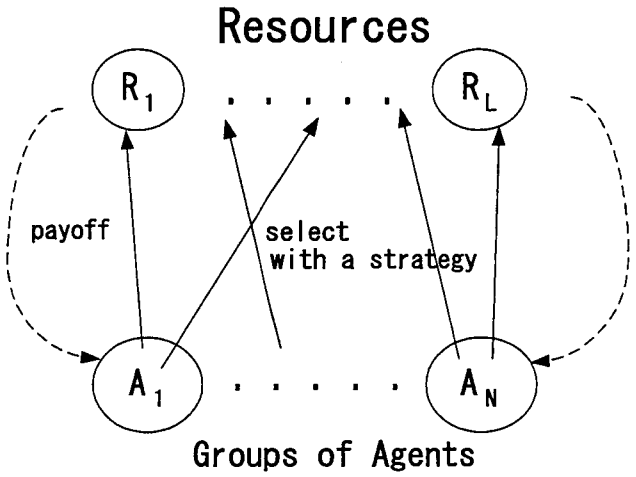


図 2.4: エージェント集団間で競合のある計算生態学モデル

- α : エージェントが資源の選択を判断し直す割合.
- γ : エージェントの行いが報われる割合.
- $\eta_{is}(k)$: 時刻 k で, A_i における, 新しいエージェントが戦略 s になる確率.
- $G_{ir}(k)$: 時刻 k で, A_i について, 資源 r を選択したときの利益関数.

この拡張により, それぞれの価値基準を持った複数のエージェント集団が共通の資源に対し競合を行っている様子を表現している. これによって, マルチエージェントシステムにおける資源配分問題としての一般的な枠組みが提供されたといえるが, これはまた, 一般の資源配分問題として通用する枠組みであるといえる. そのため, (2.9) 式による定式化は, 広く一般の資源配分問題に適用できると考えられる.

次に, (2.9) 式を用いた, 計算生態学アプローチに基づく制御手法を示す. まず, 資源の選択を行う主体をエージェントとしてとらえ, それぞれの持つ価値基準に応じて複数のエージェント集団 A_i に分割する. また, 利益関数 G_{ir} を適切に設定する. そして, 各 A_i の資源の選択を, A_i における資源 r の選択割合 f_{ir}^{res} に基づき決定し, 同時に資源選択から得られる利益をもとに, f_{irs} を更新していくことで, 動的に資源配分の決定を行っていく. ここでは, 本来, エージェント集団の振舞いの解析の結果得られたモデルを逆に, エージェントの振舞いを制御するための手法として積極的に用いようとしている.

Hogg-Huberman モデルの特性として, 各々のエージェントは資源への競合を通じて, エージェント間での直接的な交渉無しで, 資源から得られる利益が等しくなるよ

うな資源の配分状況を全体として構成する．これは資源配分として望ましい解を求めていると考えられる．また，このモデルは環境の変更に対し，自律的に資源の配分状況を変更していくという適応性も有している．

2.5 シミュレーション

2.5.1 単一のエージェント集団に対するシミュレーション

最初に，エージェント集団が一つの場合について，計算生態学モデルによるカオスの発生と，Hogg-Huberman モデルによる安定化の一例をコンピュータシミュレーションにより示す．

2 個の資源を持ち，次のような利益関数を持つシステムを考える．

- 資源 1 の利益関数 $G_1(f_1) = 4 + 7f_1 - 5.33f_1^2$
- 資源 2 の利益関数 $G_2(f_1) = 4 + 3f_1$

このシステムに対し，以下のパラメータを用いて計算生態学モデルの挙動のコンピュータシミュレーションを行った．

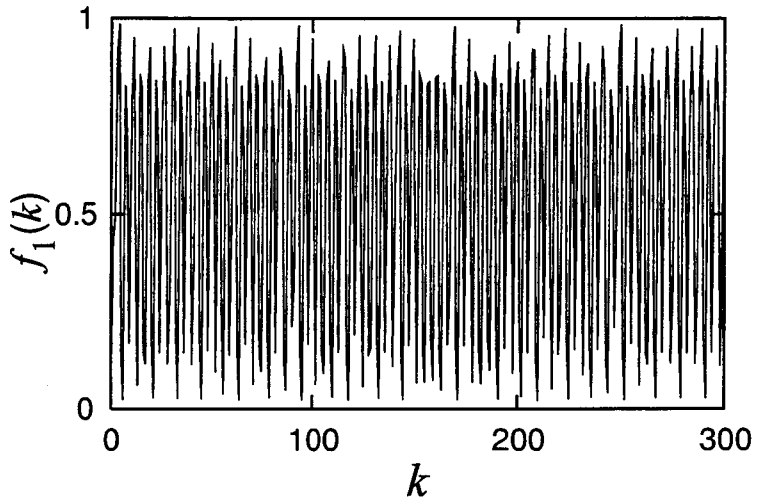
- $\alpha = 0.85 \quad \sigma = 0.25 \quad \tau = 1$
- 初期値 $f_1(0) = 0$
- ρ_r …… (2.3) 式を用いる．

このとき，システムの振舞いは図 2.5 (a) のようになり，カオスが発生していることが分かる．

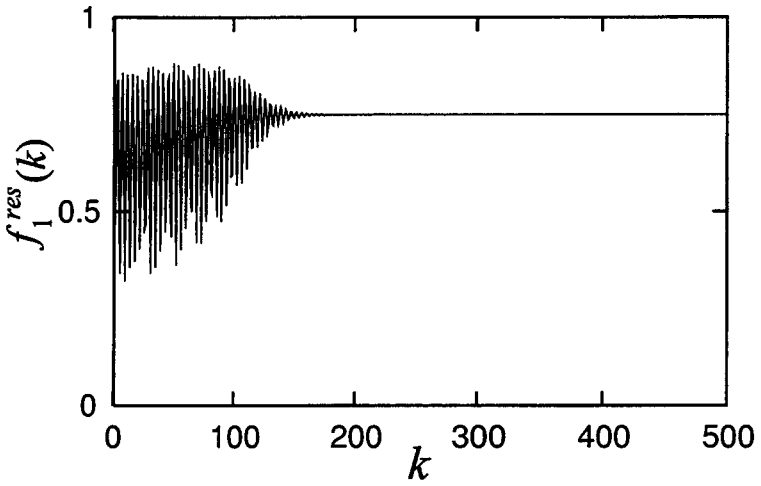
次に，このシステムに対し，Hogg-Huberman モデルを適用してコンピュータシミュレーションを行った．このとき，以下のパラメータを用いた．

- $\alpha = 0.85 \quad \sigma = 0.25 \quad \tau = 1 \quad \gamma = 1$
- ρ_{rs} …… (2.8) 式を用いる．
- 戦略 s として -1 から 1 まで 0.1 きざみの合計 21 戦略を用意する．
- 戦略の初期分布は一様に与える．

このとき，システムの振舞いは図 2.5 (b) のようになり，報酬メカニズムの導入によってシステムが安定になっていることが分かる．



(a) 計算生態学モデルによる時系列



(b) Hogg-Huberman モデルによる時系列

図 2.5: 各モデルのシミュレーション（単一のエージェント集団の場合）

2.5.2 複数のエージェント集団が存在する場合のシミュレーション

次に，複数のエージェント集団が存在する場合についても同様にコンピュータシミュレーションを行った．2 個の資源に対し，2 つのエージェント集団が競合しているシステムを考える．利益関数は，

- エージェント集団 A_1 の資源 1 に対する利益関数 $G_{11}(f_{11}) = 4 + 7f - 5.33f^2$

- エージェント集団 A_1 の資源 2 に対する利益関数 $G_{12}(f_{11}) = 4 + 3f$
- エージェント集団 A_2 の資源 1 に対する利益関数 $G_{21}(f_{21}) = 7 + 5f - 3.55f^2$
- エージェント集団 A_2 の資源 2 に対する利益関数 $G_{22}(f_{21}) = 5 + 6f$

とした．ここで，$f = 0.5(f_{11} + f_{21})$ である．この利益関数により，各エージェントの利益は，自分の所属するエージェント集団だけでなく，他のエージェント集団の資源選択割合にも影響を受けることになる．

このシステムに対し，以下のパラメータを用いて戦略を用いない場合の挙動のシミュレーションを行った．

- $\alpha = 0.5,\ \sigma = 0.35,\ \gamma = 1$
- 初期値 $f_{11}(0) = 0.5,\ f_{21}(0) = 0.5$
- ρ_{ir} の与え方:

$$\rho_{i1} = \frac{1}{2}\left(1 + \mathrm{erf}\left(\frac{G_{i1}(f_{i1}(k-\tau)) - G_{i2}(f_{i1}(k-\tau))}{\sqrt{2}\sigma}\right)\right)$$

計算生態学モデルによるエージェント集団 A_1 の資源 1 の選択割合 f_{11} の時系列を図 2.6(a) に，f_{11}，f_{21} の位相平面を図 2.6(b) に示す．両図より，システムの挙動が安定していないことが分かる．次に，戦略を導入した場合のシミュレーション結果を示す．このとき，以下のパラメータを用いた．

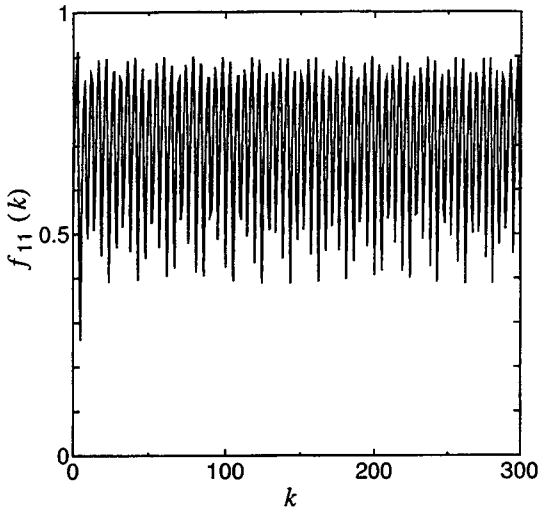
- $\alpha = 0.5,\ \sigma = 0.35,\ \gamma = 1$
- ρ_{irs} 与え方:

$$\rho_{i1s} = \frac{1}{2}\left(1 + \mathrm{erf}\left(\frac{G_{i1}(f_{i1}(k-\tau)) + s - G_{i2}(f_{i1}(k-\tau))}{\sqrt{2}\sigma}\right)\right)$$

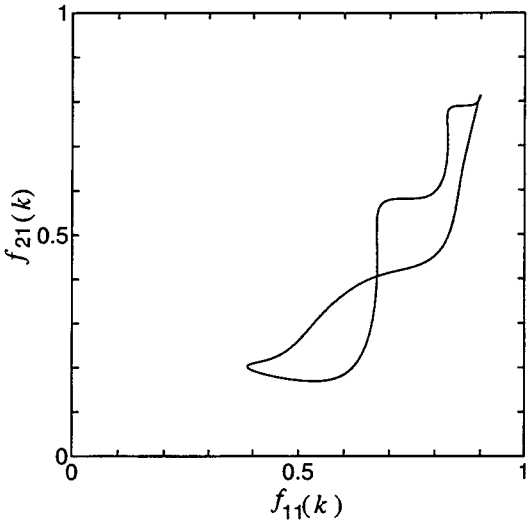
すなわち，バイアス値の与え方を戦略に用いてる．

- 戦略 s として，-1 から 1 まで 0.1 きざみの合計 21 戦略を用意する．
- 戦略の初期分布は一様に与える．

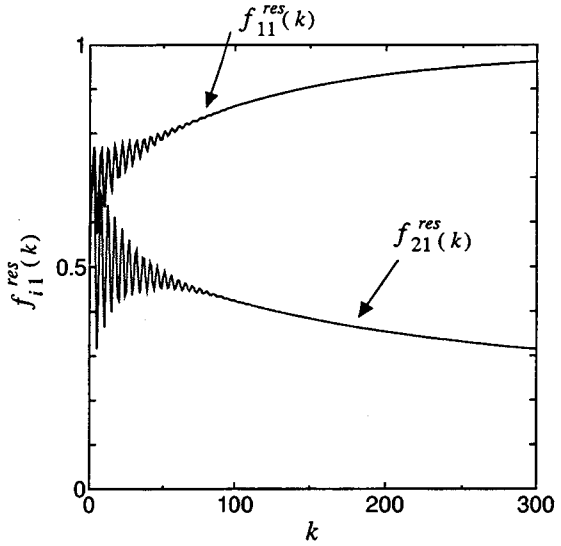
エージェント集団 A_1，A_2 それぞれの，資源 1 の選択割合 f_{11}^{res}，f_{21}^{res} は，図 2.6(c) のようになり，複数のエージェント集団が存在する場合も，報酬メカニズムによりシステムが安定になっていることがわかる．



(a) 計算生態学モデルによる時系列



(b) 計算生態学モデルによる位相平面



(c) Hogg-Huberman モデルによる時系列

図 2.6: 各モデルのシミュレーション (エージェント集団が 2 個の場合)

2.6 結言

本章では，計算生態学モデルおよび，これに生じるカオス現象を抑制するための報酬メカニズムを導入したHogg-Hubermanモデルを紹介し，このモデルを複数のエージェント集団が存在するように拡張したHogg-Hubermanモデルに基づく制御手法の枠組みを示した．提案方式では，自律的に環境に適応するモデルのもとで，エージェント間でのより良い資源配分を動的に決定していく．次章では，本章で示した制御手法を用いての実際の適用例として，ネットワークの経路選択問題への応用を示す．

第 3 章

計算生態学アプローチの経路選択問題への応用

3.1 緒言

近年発展が目覚ましい通信ネットワークにおいては，技術的な発達とともに，インターネットへの接続人口の爆発的な増加もあって，そのトラヒックは飛躍的に増大している [1, 2, 11]．それゆえ，ネットワーク上を流れる様々な種類のトラヒックに対してネットワーク資源の効率的な利用がよりいっそう求められている．ネットワークは一般に分散非同期の大規模システムであり，集中管理的な運用は決して簡単ではない．そうしたシステムの特性をふまえ，環境の様々な変化に対しても自律分散的に振舞うことのできる制御系設計が必要である．その解決策としてマルチエージェント型のネットワークルーチング方式 [16] や QoS 制御 [34] も提案されている．

ネットワークに関する諸問題の中でもルーチングは広く一分野を形成しているが [18, 26, 35]，本章ではネットワークの中から今日広く使われているパケット網を取り上げ，これに Hogg-Huberman モデルを拡張して適用することによりパケット網ルーチングの一手法を提案する [36, 37]．ここではそれぞれのノードにおいて，各パケットを目的地別に異なったエージェント集団とし，リンクを資源と考えることにより，マルチエージェントシステムにおける資源配分問題としてルーチングをとらえる．そして，前章で示した複数のエージェント集団が存在するように拡張した Hogg-Huberman モデルを適用することにより，パケットのリンクの選択割合を決定

する．この選択割合にしたがってパケットをリンクに割り振ることによりルーチングを達成する．利益関数に基づいた効率的な資源配分を実現することでネットワーク全体としての性能向上を図っている．それぞれのノードは独立にルーチングを行うが，Hogg-Huberman モデルの利用により環境への適応性を持たせ，また，隣接ノードのみとの情報交換による少ない制御用トラヒックの下で，分散環境下での耐故障性の向上も図る．

以下，3.2 節では本章で対象とするパケット網モデルを示す．これに対し 3.3 節で，2.4 節で示した手法に基づき，拡張した Hogg-Huberman モデルを用いた新たなパケット網ルーチング方式を提案する．3.4 節では，3.3 節で提案した方式について 3 つの視点からコンピュータシミュレーションを行い，本方式の性能を調べる．3.5 節で本章の結論を述べる．

3.2 ネットワークモデル

本章で対象とするネットワークは，個々のデータをパケットと呼ばれる一定長のデータ列に分解してネットワーク上を送信させるパケット網とする [2, 19, 25]．具体的には，ノード，リンク，バッファからなる抽象化されたネットワークを考える．ノード間はリンクで接続されており，パケットはリンクを通ってノード間を移動し，パケット発生時に設定された最終的な目的ノードへと伝送される．リンク選択の概念図を図 3.1 に示す．このように，前段のノードから運ばれてきたパケットが，それぞれのノードで次に進むリンクを選択し，対応するバッファに入れられる．以下の条件を仮定する．

- 各ノードは，そのノード内のトラヒック状態についてのみリアルタイムに把握しているとする．
- 各ノードはネットワークのトポロジー情報をあらかじめ保持しているとする．
- パケットのサイズは一定とする．
- 各接続リンク毎に一つの送信バッファを持つとする．各バッファ内のパケットは，FIFO 制御で順次送信されていく．

一般に通信においては，その開始時に送信側と受信側との間に情報を送信するための通信路（チャネル）をつくっておくコネクション型通信と，あらかじめ送信側と受

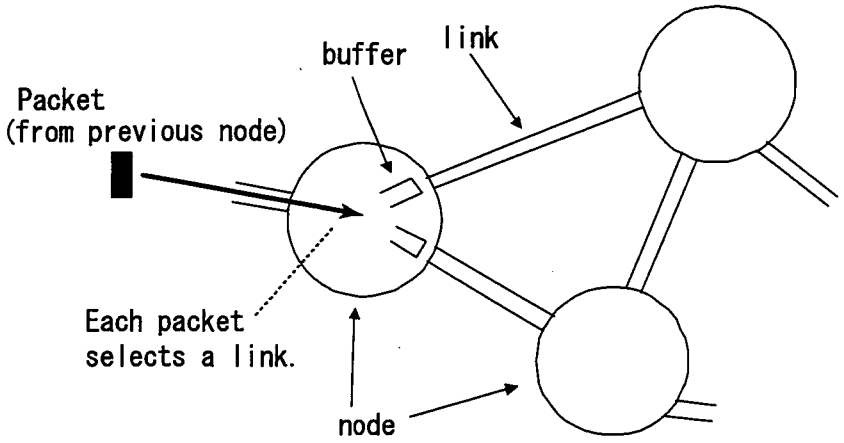


図 3.1: リンク選択の概念図

信側とで通信路を確保することなしにいきなり目的地へとデータの送信を始めるコネクションレス型通信の 2 つが存在する．コネクション型通信は回線をあらかじめ確保する分，高信頼型の通信が可能であり，電話網等で用いられている．一方，コネクションレス型通信はベストエフォート型の通信となるが比較的安価で，拡張性が高く，インターネット等で用いられている [11]．本論文ではコネクションレス型の通信を考える．

N 個のノードを持つネットワークに関して次のように記号を定義する．なお本論文では分散環境下でのルーチングを行うため，その動作についてはノードごとに独立に行われるとしている．そのため，簡単のためにノード番号に関する添字は省略する．ただし，実際においてはこれらのパラメータは，それぞれのノードにおいて異なっているので注意されたい．

まず，各ノードごとに以下のパラメータを持つとする

- l_r : ノードから出ているリンクのラベル $(r = 1, \cdots, L)$．L はノードからでているリンクの総数である．
- C_r : l_r の転送速度．ある単位時間に転送できるパケットの個数で表す．
- B_r : リンク l_r に対応するバッファに入っているパケット数．
- B_{MAXr} : バッファ B_r の容量．

次に，各パケットはそれぞれ以下のパラメータを持つとする．

- W : 許容遅延時間．パケットの発生時に設定される．

- t：経過遅延時間

また，P_i によって，パケットをその最終目的地別にラベル付けしておく．パケットは，$W < t$ となったとき，あるいはバッファの空き容量が無いときに廃棄されるとする．

3.3 システムの定式化

本節では，前章で述べた拡張した Hogg-Huberman モデルをネットワークモデルに適用することにより，パケット網ルーチング方式を提案する [36, 37, 38]．

ルーチングの基本方針

最初に，提案するパケットルーチングのアルゴリズムの要約を述べる．

パケットをその目的地ごとに異なったエージェント集団に分ける．資源を，そのノードから隣接するノードへと伸びているリンクと考える．ここで，各パケットはいずれかのリンクを選択する必要があるが，このとき，複数のエージェント集団が共通の資源であるリンクを取り合うという競合を持つモデルとなる．この枠組みは，マルチエージェントシステムにおける資源配分問題としてとらえることができる．そして，この競合の解消に Hogg-Huberman モデルを適用する．

各ノードにおいて，到着したパケットは 1 個のエージェントとして，それぞれのエージェント集団ごとに Hogg-Huberman モデルに従って決定され保持している各リンクへの選択割合を基に確率的にいずれかのリンクを選択する．リンクが選択されるとそのパケットは選択されたリンクに対応するバッファの最後尾に追加され，順次 FIFO で次ノードへと送られる．また，パケットによるリンクの選択は Hogg-Huberman モデルに影響を与えるので，この相互作用のもとで，Hogg-Huberman モデルに基づいたより良い資源配分の実現を目指すことになる．これらの処理を，それぞれのノードにおいて独立に行い，個々にルーチングを行う．

ここでは，Hogg-Huberman モデルによって，各エージェント（パケット）が資源（リンク）から得る利益が均衡するように振舞うことにより，結果的にネットワーク全体のパフォーマンスの向上が期待される．また，トラヒックの変化に対しても Hogg-Huberman モデルが自律的に資源の選択割合を調整することにより柔軟に対応

できると考える.

3.3.1 離散時間 Hogg-Huberman モデルの適用

前節, (2.9) 式を用いて, Hogg-Huberman モデルによるルーチングを定式化する. あるノードに関して, パケットをその最終目的地ごとにいくつかのエージェント集団に分けて, Hogg-Huberman モデルを複数のエージェント集団がある場合に拡張すると次式が得られる. 各パラメータの意味は, 本来の Hogg-Huberman モデルでの意味に代えて, ネットワークルーチングとしての解釈を行っている. このモデルのそれぞれの λ_{ir}^{res} に基づいて各ノードにおいてパケットのリンクへの振り分けを行い, ルーチングを実現することになる.

$$\lambda_{irs}(k+1) = \lambda_{irs}(k) + \alpha\left(\lambda_{is}^{str}(k)\rho_{irs}(k) - \lambda_{irs}(k)\right) + \gamma\left(\lambda_{ir}^{res}(k)\eta_{is}(k) - \lambda_{irs}(k)\right) \tag{3.1}$$

$$\lambda_{ir}^{res}(k) = \sum_{s}\lambda_{irs}(k), \quad \lambda_{is}^{str}(k) = \sum_{r}\lambda_{irs}(k)$$

$$\eta_{is}(k) = \frac{\sum_r \lambda_{irs}(k)G_{ir}(k)}{\sum_r \lambda_{ir}^{res}(k)G_{ir}(k)}$$

ここで, 各変数の意味は以下の通りである.

- $\lambda_{irs}(k)$: 時刻 k で, エージェント集団 P_i における, 戦略 s でリンク l_r を使うエージェントの割合.
- $\lambda_{ir}^{res}(k)$: 時刻 k で, P_i における, リンク l_r を使うエージェントの割合.
- $\lambda_{is}^{str}(k)$: 時刻 k で, P_i における, 戦略 s を採るエージェントの割合.
- α : エージェントが資源の選択を判断し直す割合.
- γ : エージェントの行いが報われる割合.
- $G_{ir}(k)$: 時刻 k で, P_i について, リンク l_r を選択したときの利益関数.
- $\rho_{irs}(k)$: 時刻 k で, P_i について, 戦略 s でリンク l_r をエージェントが選択する望ましい割合の推定値.
- $\eta_{is}(k)$: 時刻 k で, P_i について, 戦略 s を採るエージェントの望ましい割合.

図 3.2 のようにそれぞれのノードにおいて, N 個のエージェント集団 (目的地別

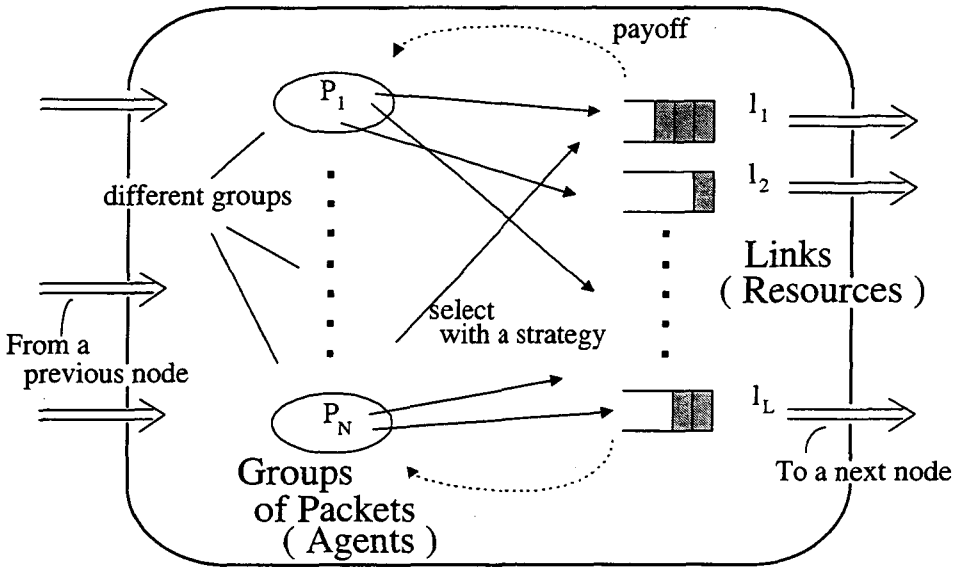


図 3.2: エージェント集団のリンクへの競合

のパケット群）が L 個の共通の資源（そのノードから出ているリンク）を取りあうという競合を持ったモデルに Hogg-Huberman モデルが拡張されている．このように，複数のエージェント集団が存在するように拡張したのは，一つのエージェント集団は換言すれば同じ価値基準（利益関数）を共有している集団であり，ネットワークにおいては，パケットは，その目的地が異なればリンク選択の判断基準が異なるため，各パケットを目的地別に異なったエージェント集団としてとらえる必要があるためである．

Hogg-Huberman モデルは資源から受ける利益が等しくなるように振舞うモデルである．よって，このモデルに基づきリンクの選択確率を動的に決定することにより，各パケットはその目的地に応じた経路を他のパケット集団の処理のポリシーを陽に知ること無く，現在の資源の選択状況に応じて適切にかつ，動的に選択することとなる．たとえば，ある時点においてはリンク（資源）1 の選択による利益が大きくても，多数のパケット（エージェント）がそのリンクを選択してしまうと，そのリンクを選択する利益が減少し，結果としてリンク 1 を選択する割合が減少して，全体として負荷の均衡化（実際は利益関数の意味での均衡化）を達成するように振舞う結果となる．

3.3.2 利益関数 G_{ir} の設定

利益関数はエージェントに環境の評価値を与える重要な関数である．各エージェントにとって，この値が高いリンクがすなわち良いリンクであるといえる．そしてまたパケットにとっては，なるだけ早く目的地に到着できる見込みのあるリンクが良いリンクであるといえる．

あるノードにおける，エージェント集団 P_i，リンク l_r の利益関数 G_{ir} を以下のように設定する．

$$G_{ir}(k) = w_1 \times \frac{\min_a \left(C_a^{-1} \times \bar{B}_a(k) + b\right)}{C_r^{-1} \times \bar{B}_r(k) + b} + w_2 \times \frac{\min_a D_a(k)}{D_r(k)} + w_3 \times \frac{V_{ri}}{\max_a V_{ra}} \quad (3.2)$$

$$w_1 + w_2 + w_3 = 1, \qquad \bar{B}_r(k) = \frac{\sum_{t=0}^{T-1} B_r(k-t)}{T}$$

- $D_r(k)$：時刻 k において，自ノードが把握しているリンク l_r の先のノードの輻輳の度合（以後，輻輳度と呼ぶこととする.）を表す指標.
- V_{ri}：リンク l_r が，パケット集団 $P_i(i = 1, \cdots\cdots, N)$ に対して持つ固有の評価指標.
- w_1, w_2, w_3：各項の重み係数.
- b：待ち時間に対するバイアス.
- $\bar{B}_r(k)$：時刻 k における，リンク l_r のバッファの過去 T 時間の待ちパケット数の平均値.

利益関数の各項の意味的な解釈は次のようになる.

第 1 項 リンク l_r 使用時の転送時間に関する評価値．待ち時間の大きいリンクほど評価が低くなる．なお，待ち時間に対するバイアス b はバッファ内パケットの変化に利益関数の値が敏感に反応しすぎないように導入している．このように待ち時間にバイアスを加えるという処理はリンク選択の発振を抑制するために効果があるとされている [2].

第2項 リンク l_r 以遠にあるノードに関しての，輻輳度に基づく評価値．輻輳度の定義式等については 3.3.3 節で述べる．輻輳度の大きいリンクほど評価が低くなる．

第3項 ネットワークのトポロジー的な評価値．目的ノードまで到達するのに，リンク l_r を選択することの，全体のトポロジーから見ての評価を行う．この項によりエージェント集団ごとに異なるリンクに対する評価を扱うことができる．V_{ri} が大きいリンクほど評価が高くなる．

このように，複数の視点からの評価値を利益関数という形で一元的に盛り込んでいる．そして，重み係数 w_1, w_2, w_3 によって，どの評価指標を重視するかという設計の余地を持たせている．

3.3.3 輻輳度 D_r の設定

隣接ノードから輻輳度として次ノード以遠の輻輳に関する指標を得られる仕組みを導入する．この輻輳度 D_r について，以下の式でもって定義する．また，その概念図を図 3.3 に示す．ここで，ノード N_1 と N_2 が N_1 側から見てリンク l_{r1}，N_2側 から見てリンク l_{r2} によってつながっているとする．さらに，ノード N_2 のリンクの数を L 本とする．このときノード N_1 へと伝達される輻輳度 D_{r1} は，ノード N_2 において次式で計算され，一定時間毎に N_1 に送られ新しい通知が来るまで保持される．

$$D_{r1} = w_4 \times \frac{\sum_{a \neq r2} \left(C_a^{-1} \times \bar{B}_a + b \right)}{L-1} + w_5 \times \frac{\sum_{a \neq r2} D_a}{L-1} \qquad (3.3)$$
$$w_4 + w_5 = 1$$

ノード N_1 からみて，第 1 項がノード N_2 の輻輳の度合の指標となり，第 2 項が N_2 以遠の輻輳の度合を計る指標となる．w_4, w_5 が各項の重み係数となる．

このような情報の伝達が一定時間ごとに隣接ノード間で相互に行われる．よってそれぞれのノードはお互いに隣接ノードのみから輻輳度として一元化された評価指標を少ないトラヒックしか要さずに受け取ることができる．この情報は輻輳をきちんと定量化したものではないが，この機構の導入により，ノードはリンクおよびそのリンク以遠についての輻輳の状況をある程度知ることができる．そして，利益関数へと反映させることによって環境の変化への適応性や耐故障性の実現へとつながることが期待される．また第 2 項の存在により，遠くのノードにおける輻輳の状況も徐々に周辺へ

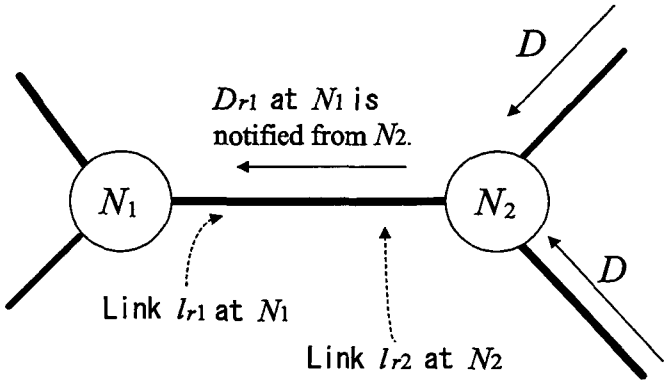


図 3.3: 輻輳度伝達の仕組み

と伝播され，それらを回避するようなルーチングを実現するように Hogg-Huberman モデルが振舞うことが期待される．

3.3.4 変数 ρ_{irs} の設定

変数 ρ_{irs} は，2.2.2 節でも述べたが利益関数をもとにその時刻でのエージェントの資源選択の目標値となる確率を与える．

変数 ρ_{irs} の設定の前に，前節の利益関数 G_{ir} に戦略の別を与えるように拡張する．戦略 s として資源数 +1 個 $(0,\cdots,L)$ の戦略を用意し，エージェント集団 P_i が戦略 s でもって資源 l_r を評価するときの利益関数を G_{irs} とし，次のように定義する [29]．

$$G_{irs}(k) = \begin{cases} G_{ir}(k) + S & r = s \text{ のとき} \\ G_{ir}(k) & \text{otherwise} \end{cases}$$

ここで S は G_{ir} に対するバイアス値を表す．この G_{irs} を用いて ρ_{irs} を定義する．ただし，t は $G_{irs} = \max_a\{G_{ias}\}$ を満たす資源 l_r の個数を表す．

$$\rho_{irs}(k) = \begin{cases} \frac{1}{t} & G_{irs}(k) = \max\limits_a\{G_{ias}(k)\} \text{ のとき} \\ 0 & \text{otherwise} \end{cases}$$

すなわち，ρ_{irs} は，それぞれの戦略について最大値をとる G_{irs} のなかで確率が等分される．この形式においては，戦略はそれぞれの資源への好みと言う形で表現されて

いるといえる．戦略 s を持つエージェントは資源 l_s を他の資源よりも好むようにバイアス S が加えられることになり，これが ρ_{irs} の決定に影響を与える．たとえば，戦略 1 をとるエージェント集団は S だけ通常よりよけいに資源 l_r を好むことになる．

3.4 シミュレーション

本節では，提案手法の評価のために 3 つの視点から，コンピュータシミュレーションによる評価実験を行う．

3.4.1 実験環境について

諸注意

計算機上での実装時の諸事項について述べる．

- シミュレーションのプログラムはC言語で記述され，UNIX ワークステーション上で実行された．
- 現実に存在する計算上の誤差に対し，確率和 $= 1$ となるように補正を加えている．
- いずれかの戦略について $\lambda_{is}^{str} = 0$ となると，以後，その戦略は消えてしまう．戦略の多様性を確保するため，これを防ぐように補正を加えている．

シミュレーションの処理手順

コンピュータシミュレーション時の各ノードでの処理手順は次のようになる．

Step 1 一定の分布に従いパケットを発生させる．

Step 2 到着したパケットについて，廃棄されるべきものは廃棄する．

Step 3 バッファに振り分けられていないパケットについて，各々のエージェント集団の Hogg-Hubreman モデルにより定まっている λ_{ir}^{res} に基づいてバッファに振り分ける．

Step 4 各バッファでリンクの転送速度に従いパケットを送信する．

Step 5 一定時間毎に隣接ノードに輻輳度 D_r の情報を伝達する．

Step 6 一定時間毎に Hogg-Huberman モデルを更新する.

Step 7 時間を更新し，Step1 に戻る.

3.4.2 1ノードに着目した評価

最初に，Hogg-Huberman モデルの振舞いを 1 つのノードに着目して調べる.

設定

提案方式においては，実際のリンクの選択は個々のエージェント（パケット）毎に各リンクの選択割合 λ_{ir}^{res} を用いて確率的に決められる（仮想的にサイコロを振っていると考えてもらえれば良い）. また当然バッファ内の状態は離散値のみをとるが，ノードにおけるモデルの定性的な振舞いを見るために，この 3.4.2 節では，パケットの振り分けに連続量を許し，パケットの振り分けをそれぞれの λ_{ir}^{res} の値そのもので行うとする. すなわち，

$$\text{ある時刻でバッファ } B_r \text{ に振り分けられるパケット数} = \sum_i (P_i \text{ の到着パケット数} \times \lambda_{ir}^{res})$$

としたうえで 1 つのノードに着目してその挙動を調べる.

対象として図 3.4 のような 3 つの送信バッファを持つノードを考え，上流から 20 *packets/unit-time* でパケットが送られてくるとする. エージェント集団数は 1 とし，モデルは 1 *unit-time* ごとに更新する. なお，実験上のすべての処理はこの *unit-time* を最小単位とした仮想的な時間軸にのっとって処理している. 各バッファの初期パケット数は各 100 個，初期選択確率は均等に与える. また，その他のパラメータは以下のように与える.

- $C_1 = 10,\ C_2 = 6,\ C_3 = 4$
- $\alpha = \gamma = 0.05$
- $T = 1,\ b = 10$
- $w_1 = 1,\ w_2 = w_3 = w_4 = w_5 = 0.$

重み係数により，直観的に分かりやすいようにバッファ内待ち時間のみが利益関数に用いられる指標となっている.

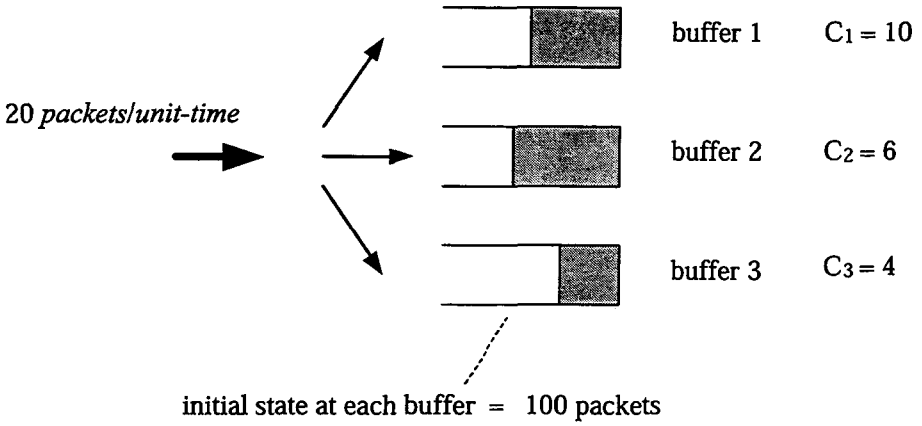


図 3.4: 実験モデル 1

さらに，時刻 $t = 750$ において $C_1 = 20$ と変化させることにより，環境の変化に対する適応性も調べる．なお，パケットの入出力量の平衡を保つため，このとき同時に流入パケット数を 20 *packets/unit-time* に増やしている．

報酬メカニズムによる影響

図 3.5 は戦略として与えるバイアス値 S のそれぞれについて，横軸に時間を，縦軸にそのときのバッファ 1 の選択割合をとり，その時間による変化を示している．また，図 3.6 はこのときの，各バッファ内の待ち時間の推移を示している．図 3.7，図 3.8 はそれぞれ図 3.5 と図 3.6 のデータをより長時間にわたって示したものである．そして，表 3.1 は図 3.7 に関し，時刻 10000 から 20000 の間について，その平均と分散を計算したものである．

資源 1 の選択割合は各リンクの速度 C_r の設定から，時刻 $t = 750$ までは 0.5，環境の変化後は $\frac{2}{3}$ となることが望ましい．いずれの戦略の場合においても理想値の周辺で資源の選択割合が変動しており，表 3.1 より，平均としてはいずれも理想値を示している．そして，各リンクそれぞれへのバイアスというシンプルな報酬メカニズムの導入により，より均衡点近くに選択割合の分散が抑えられていることが分かる．

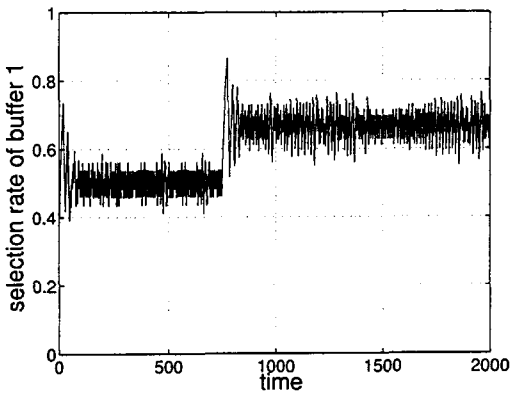
具体的に戦略の値による効果を見てみると，まず戦略の値 S として意味のある範囲は，利益関数 G_{ir} の値域が $0 \leq G_{ir} \leq 1$ であることと，変数 ρ_{irs} の設定により，$0 \leq S \leq 1$ である．図表より，この範囲のなかで戦略無しの状態から徐々に戦略の値を大きくしていくと，徐々に選択割合の変動が抑えられるようになり，ある程度大

きくなるとまた，割合の変動が増加していることがわかる．これは戦略の値が小さいと，与える影響が小さく，また大きいと利益関数に対して与える影響が支配的になり，利益の評価が適切に反映できないためと考えられる．

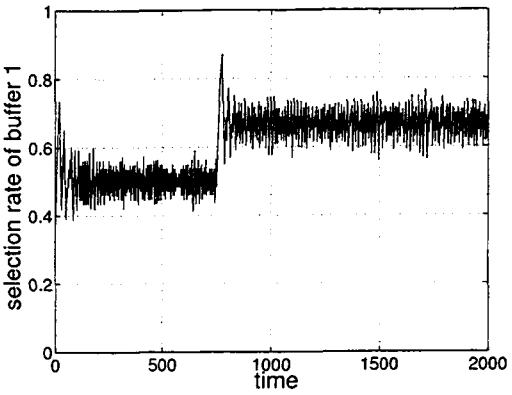
ただし，いずれの場合も各バッファの待ち時間に関して ほぼ等しくなっている．すなわち，各エージェントが資源から受け取る利益が等しい状態となっている．これらより報酬メカニズムの有効性と利益がエージェント間で Hogg-Huberman モデルにより均衡化されていること，そして，環境の変動にモデルがきちんと追従できていることが分かる．

戦略	平均	分散
なし	0.6666	1.3995×10^{-3}
S=0.001	0.6666	2.5650×10^{-3}
S=0.005	0.6666	3.8010×10^{-4}
S=0.01	0.6666	3.1746×10^{-4}
S=0.05	0.6666	1.4979×10^{-4}
S=0.1	0.6666	1.4861×10^{-4}
S=0.5	0.6666	1.5316×10^{-4}
S=1.0	0.6666	2.0029×10^{-4}

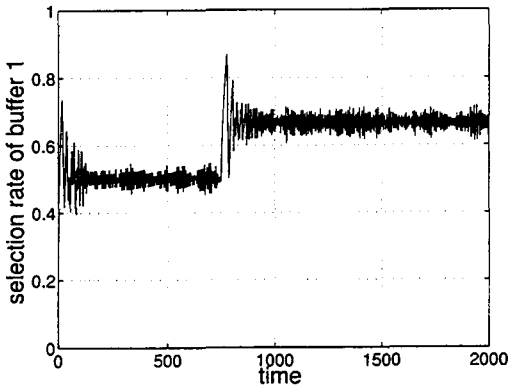
表 3.1: バッファの選択割合の変動



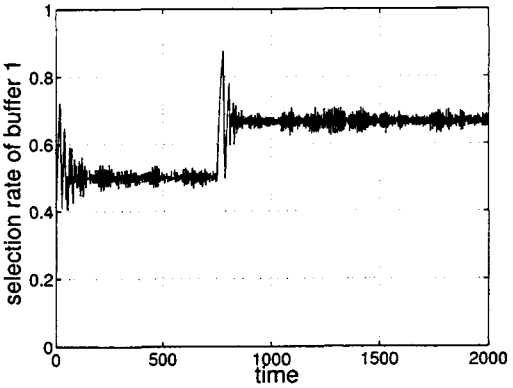
(a) 戦略なし



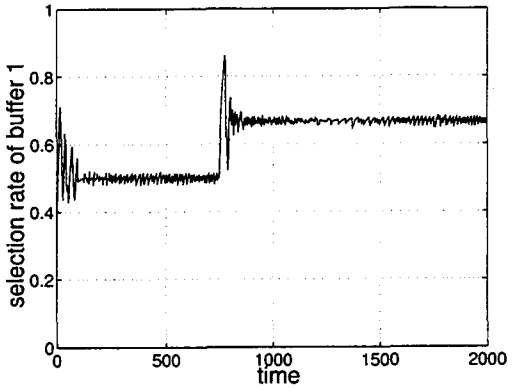
(b) 戦略 S = 0.001



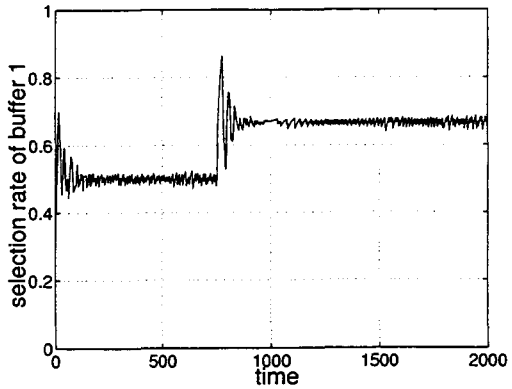
(c) 戦略 S = 0.005



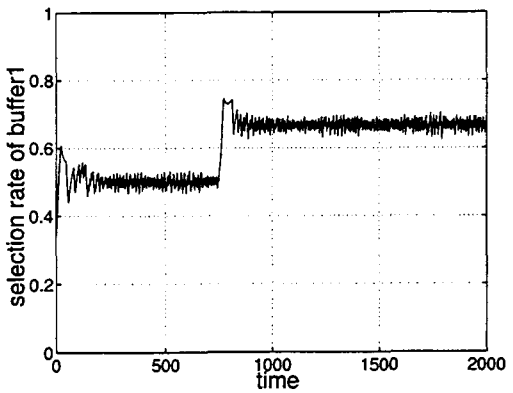
(d) 戦略 S = 0.01



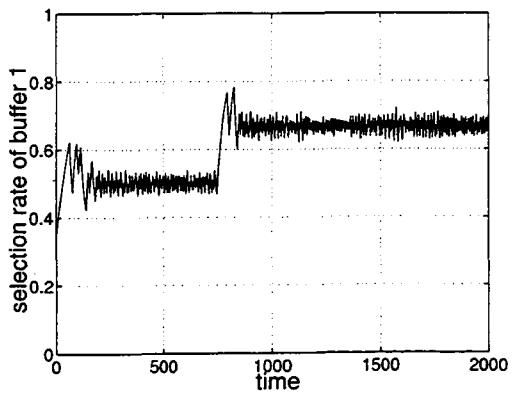
(e) 戦略 S = 0.05



(f) 戦略 S = 0.1

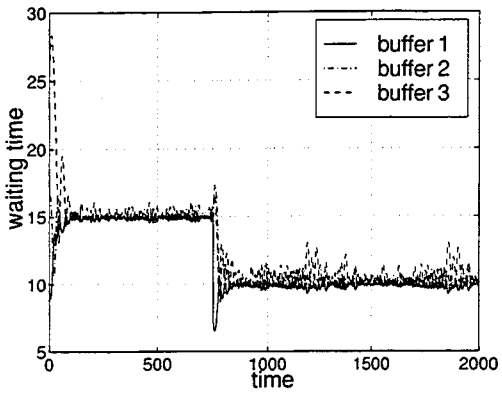


(g) 戦略 S = 0.5

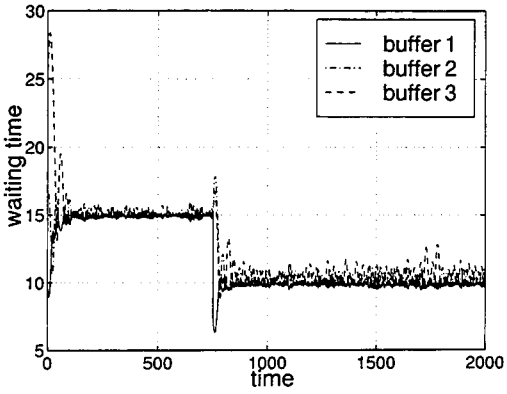


(h) 戦略 S = 1.0

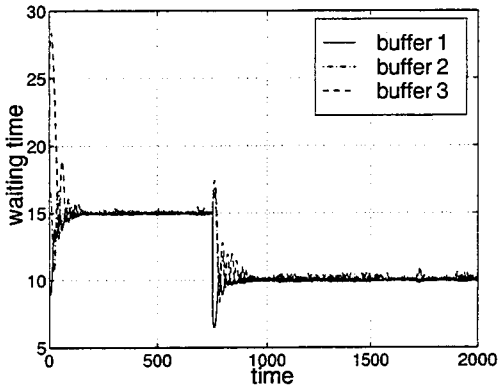
図 3.5: バッファ 1 の選択割合 (1)



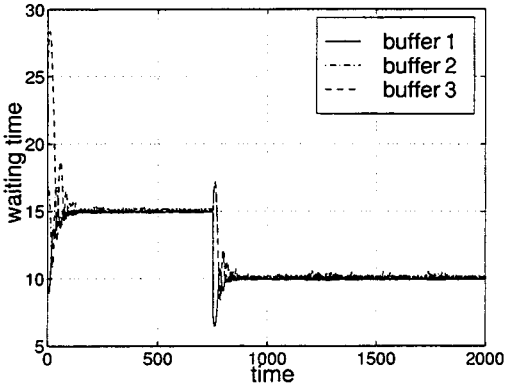
(a) 戦略なし



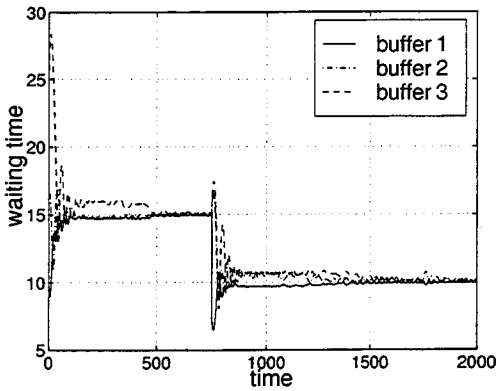
(b) 戦略 S = 0.001



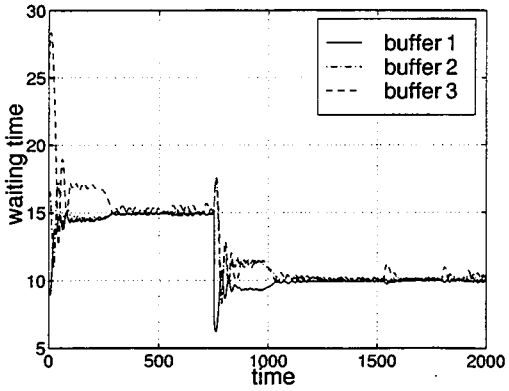
(c) 戦略 S = 0.005



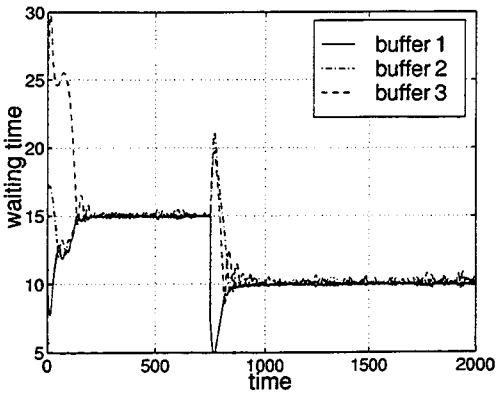
(d) 戦略 S = 0.01



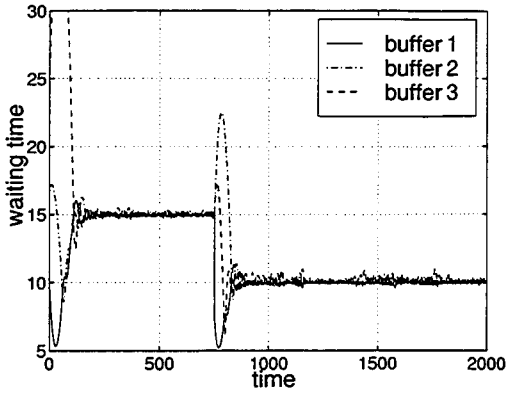
(e) 戦略 S = 0.05



(f) 戦略 S = 0.1

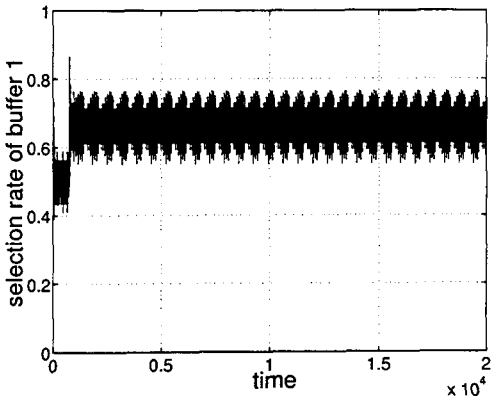


(g) 戦略 S = 0.5

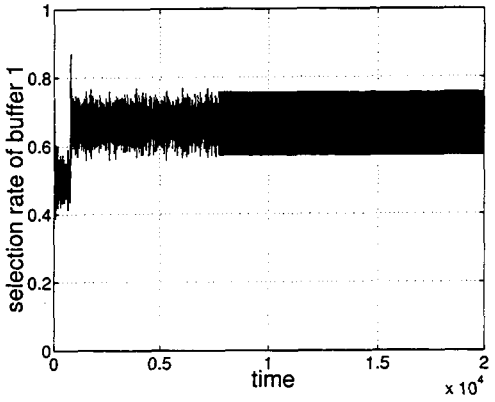


(h) 戦略 S = 1.0

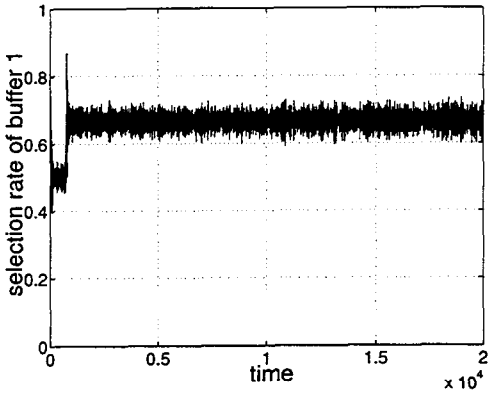
図 3.6: 各バッファの待ち時間 (1)



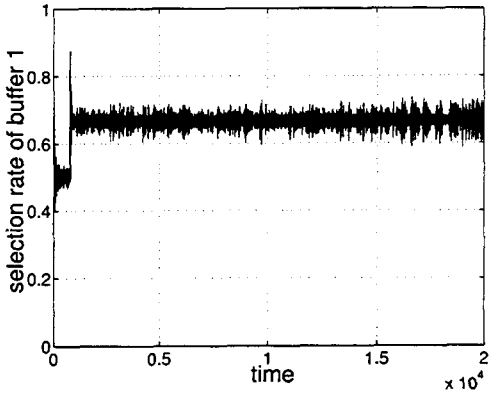
(a) 戦略なし



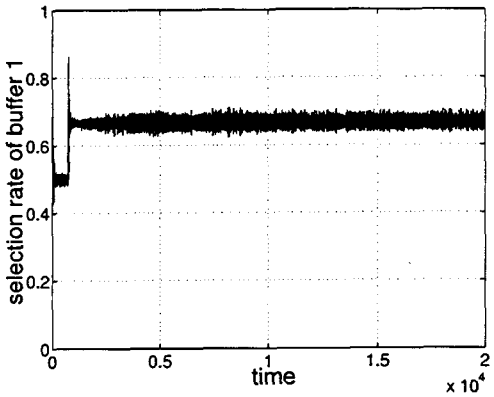
(b) 戦略 S = 0.001



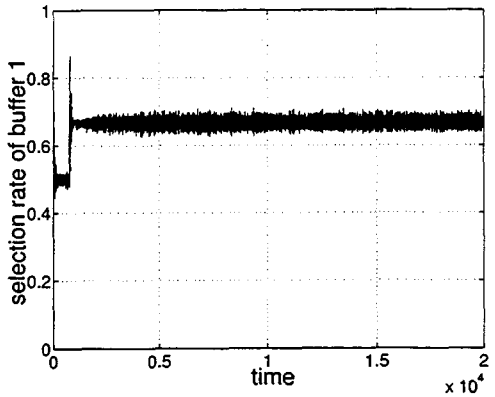
(c) 戦略 S = 0.005



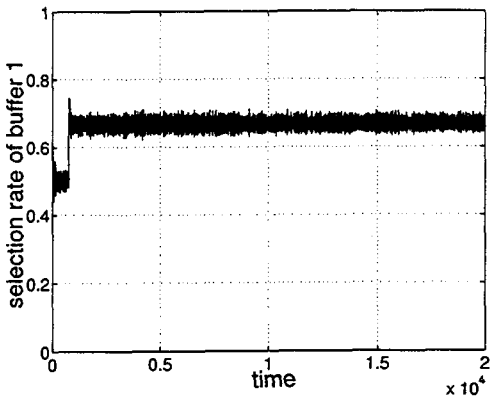
(d) 戦略 S = 0.01



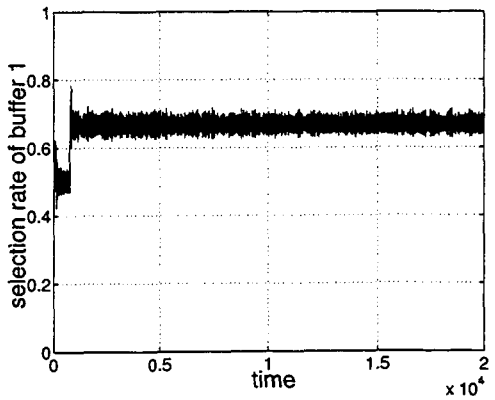
(e) 戦略 S = 0.05



(f) 戦略 S = 0.1

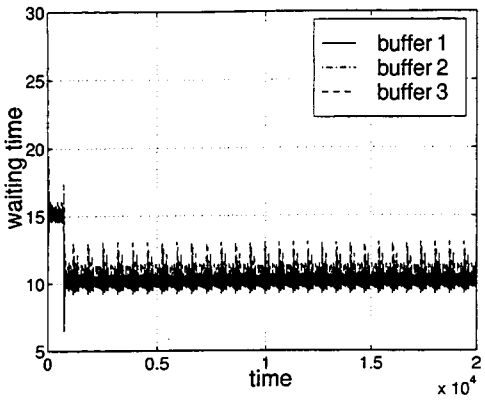


(g) 戦略 S = 0.5

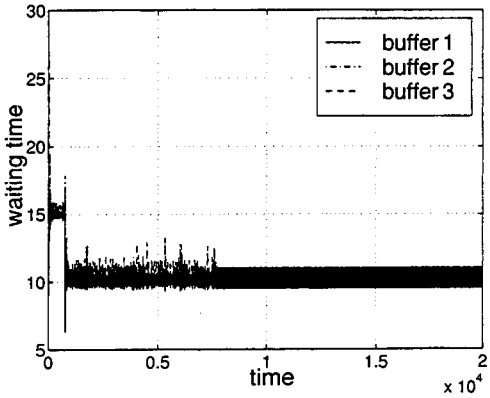


(h) 戦略 S = 1.0

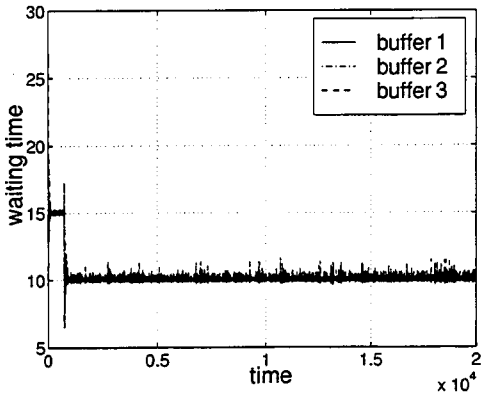
図 3.7: バッファ 1 の選択割合 (2)



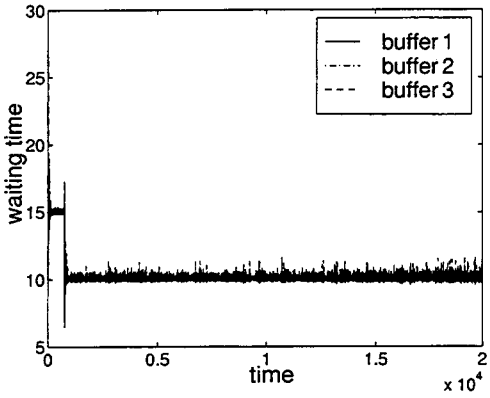
(a) 戦略なし



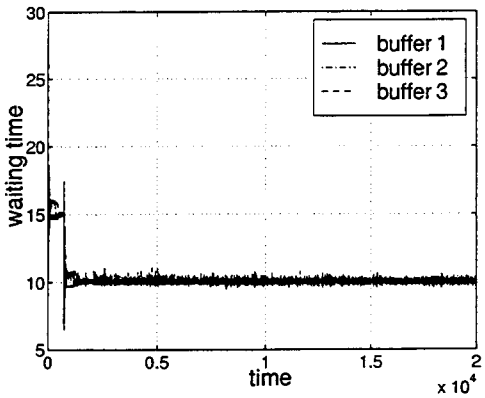
(b) 戦略 S = 0.001



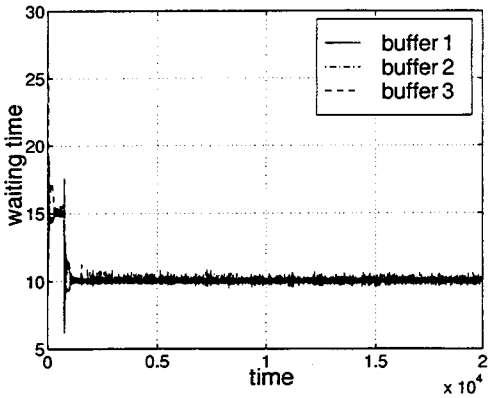
(c) 戦略 S = 0.005



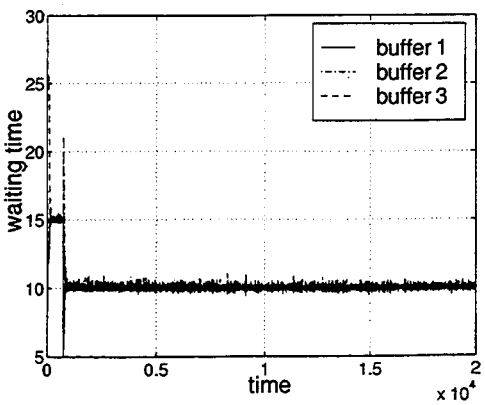
(d) 戦略 S = 0.01



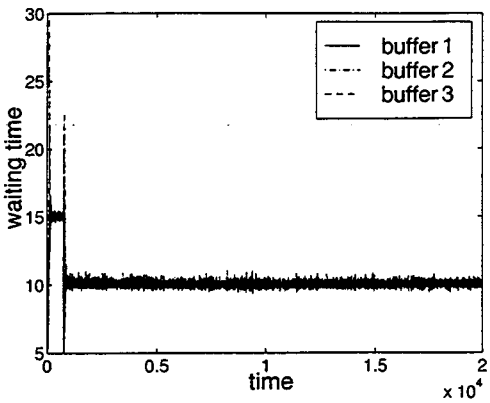
(e) 戦略 S = 0.05



(f) 戦略 S = 0.1



(g) 戦略 S = 0.5



(h) 戦略 S = 1.0

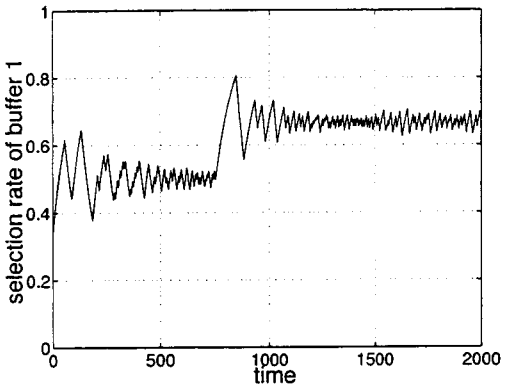
図 3.8: 各バッファの待ち時間 (2)

各種設計パラメータの評価

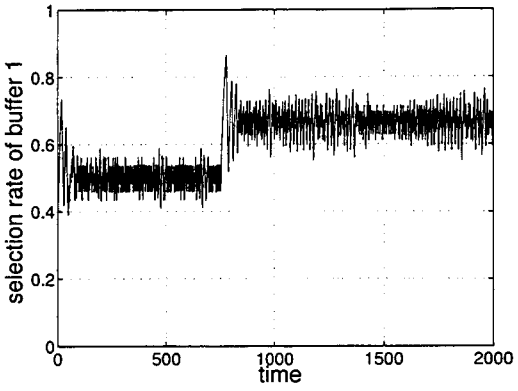
図 3.9 と図 3.10 はそれぞれ戦略なしと戦略 $S = 0.1$ の場合について，エージェントが資源の選択を判断し直す割合 α を変化させた場合のリンク 1 の選択割合の推移を示したものである．また，図 3.11 は戦略 $S = 0.1$ の場合について，エージェントの行いが報われる確率 γ を変化させた場合のリンク 1 の選択割合の推移を示している．

α は目標値としての ρ_{irs} と現在の λ_{irs} との差について，一種のゲインの大きさを与えている．実験結果では，図 3.9 と図 3.10 のいずれも，α が小さいときには過渡状態が長くなり，α が大きいときには定常状態での振動の振幅が大きくなっており，α の選択には過渡特性と定常特性の間でのトレードオフがあることが分かる．

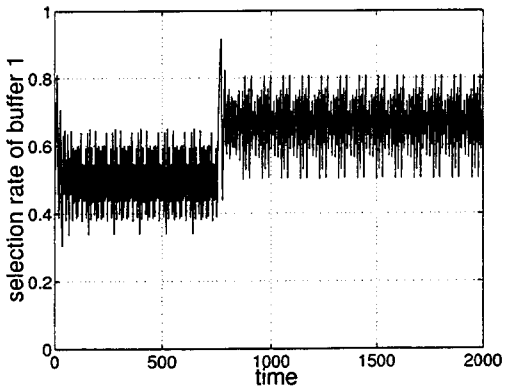
一方の γ は，報酬メカニズムによって得られる報酬の大きさに関しての一種のゲインを与えるものであるが，図 3.11 を見る限り，γ の増加につれて若干振幅が大きくなっているものの，α に比べて，モデルの挙動に与える影響は小さい．



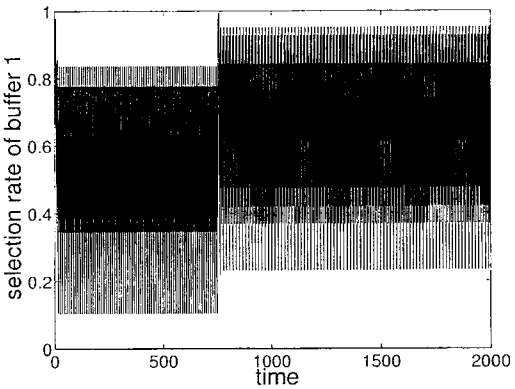
(a) $\alpha = 0.01$



(b) $\alpha = 0.05$

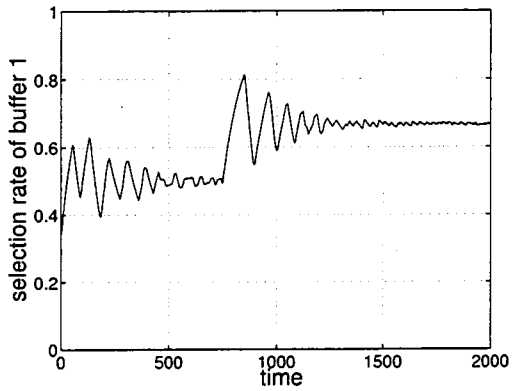


(c) $\alpha = 0.1$

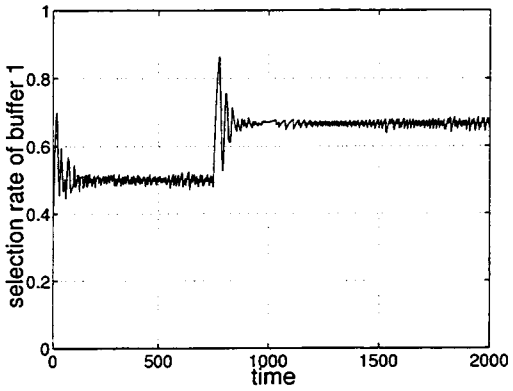


(d) $\alpha = 0.5$

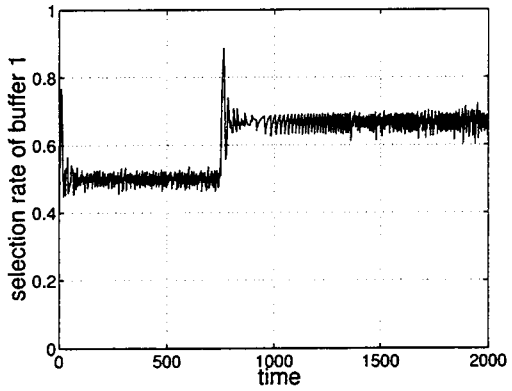
図 3.9: α による挙動の変化（戦略なし）



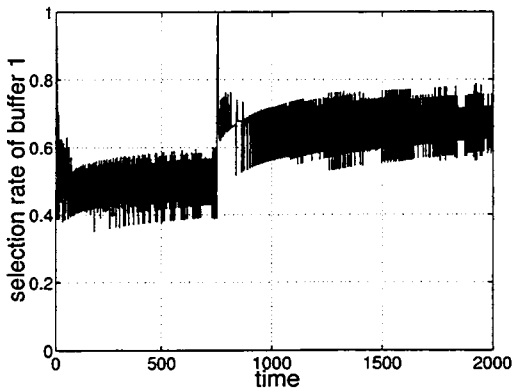
(a) $\alpha = 0.01$



(b) $\alpha = 0.05$

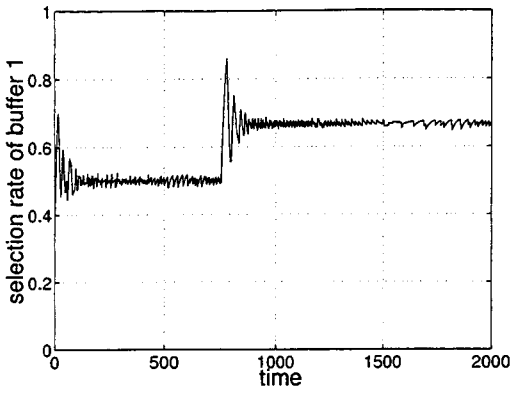


(c) $\alpha = 0.1$

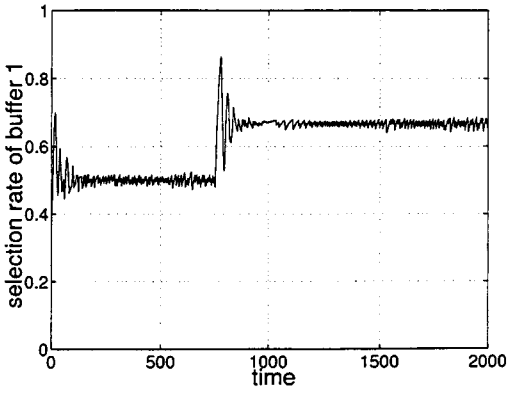


(d) $\alpha = 0.5$

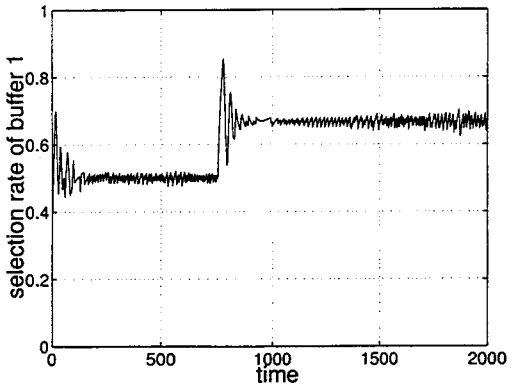
図 3.10: α による挙動の変化（戦略 S = 0.1 ）



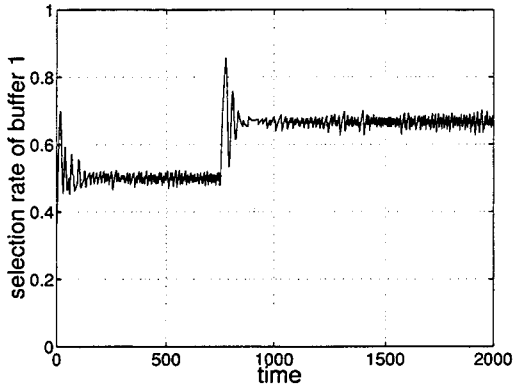
(a) $\gamma = 0.01$



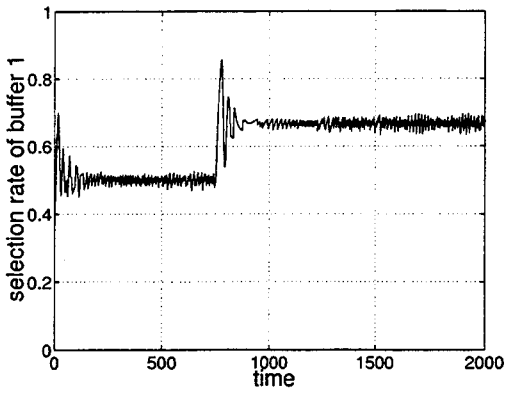
(b) $\gamma = 0.05$



(c) $\gamma = 0.1$



(d) $\gamma = 0.5$



(e) $\gamma = 1.0$

図 3.11: γ による挙動の変化（戦略 S = 0.1）

3.4.3 耐故障性

次に，今回導入した，隣接ノード間との情報交換による影響を調べる．特に，前方ノードにおいて通信不能などの故障が発生したときに，そのノードへのパケット転送が減り，迂回することによりパケット損失を防ぐことが可能であることを示す．

設定

図 3.12 に示す 7 個のノードを持つネットワークを用いて，ひとつのシナリオを仮定して実験を行う．

ノード 1 からノード 7 へ向けて 20 *packets/unit-time* のレートでパケットが発生している．リンクの転送速度は，ノード 1-2 間，1-3 間が 13 *packets/unit-time* それ以外は 10 *packets/unit-time* とする（図 3.12 のリンクへの添字を参照のこと）．ただし，ノード間のパケットの転送には最低でも 1 *unit-time* は要するとする．

リンク評価指標 V_{ri} としてはリンク選択後最短ホップ数で目的地に行ける経路数を用いる．例えば，ノード 1 から 3 へのリンクは，ノード 7 までの最短ホップ経路数が，$1-3-5-7$，$1-3-6-7$ の 2 つあるため，評価指標の値は 2 となる．

このような指標を用いたのは，最短ホップ数を取れる経路が多いほど，複数の最短経路が残されているという意味で，そのリンクの評価が高いと考えられるためである．ここで言う最短経路数は，転送速度までを考慮したものではなく，経由したノード数のみを考えている．そのため，細かいネットワークの情報は表現できないが，この指標が必要とするのは，各リンクの細かい性能ではなくネットワークのトポロジー情報のみで良いという長所がある．

モデルの更新と輻輳度の伝達を 5 *unit-times* ごとに行うとし，他の設計パラメータはそれぞれ，

- $\alpha = \gamma = 0.05$
- $T = 10,\ b = 10$
- 戦略の値 $S = 0.1$

とする．また，利益関数の重み係数としては，

1. $w_1 = w_2 = w_3 = 0.333, \quad w_4 = w_5 = 0.5$ (すなわち，輻輳度を得る場合)
2. $w_1 = w_3 = 0.5, \quad w_2 = w_4 = w_5 = 0$ （すなわち，輻輳度を得ない）

という 2 つのケースを用意する．

シナリオとして，時刻 5000 でノード 5 − 7 間のリンクが故障 (実験上はリンクの転送速度 = 0.001 として処理した) し，時刻 8000 で故障が復旧したケースを考えた．また利益関数の値は，最短ホップ経路上のリンク以外は常に 0 としている．ただし，その最短ホップ経路上にあるリンクがすべて故障しているときは除く．

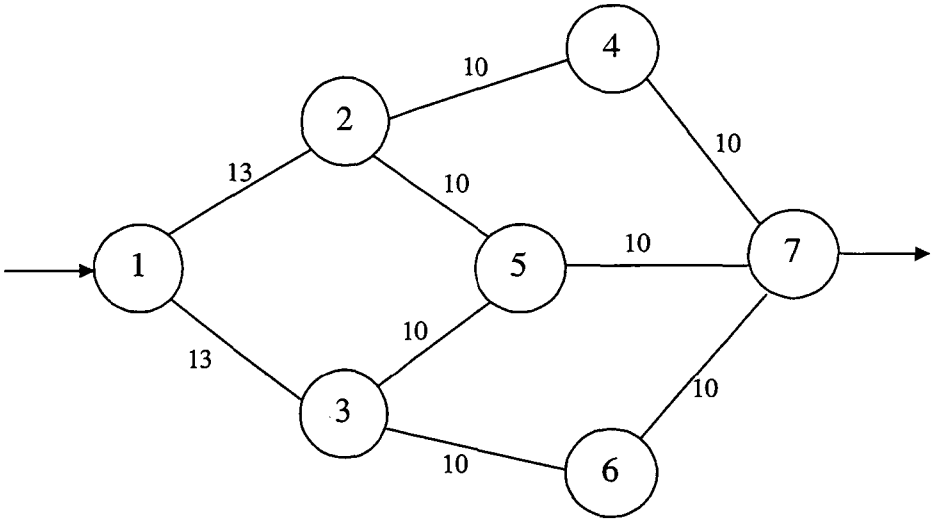


図 3.12: 7 node ネットワーク

実験結果

図 3.13 がこの時のパケットの目的ノードまでの平均到着遅延時間（パケットが目的ノードまで到着するのに要した時間の平均値）の変化を示したものである．全体的な性能においても，リンク故障時における反応においても，輻輳度を得た場合の方が明らかに良い結果を示している．このことは隣接ノードからも情報を得ているのであるから当然の結果ではあるが，この単純な仕組みがきちんと効果を発揮していることが確かめられた．また，図 3.14 は $w_2 = 0.333$ のときノード 3 からノード 5 へと向かうリンクの選択確率の推移を示したものである．この結果から次のことが考察される．第一に，正常状態において，ノード 5 にはノード 2 側からもパケットが流れ込

んでくるためにややノード5へ向けてのパケットの振り分けが抑えられていることである．このような資源配分は最適なものではないが，環境から入手できる情報からHogg-Hubermanモデルにより自律的にその挙動を調整している点で意味のあるものと考える．ただし，利益関数にまだ改善の余地があることは確かである．第二に，リンク故障時はノード5方面の輻輳度の増加として感知されたため，その方面へ振られるパケットが大幅に減らされたことが分かる．しかし，これはリンク故障というトポロジーレベルでの変化とまでは認識できていないため，若干のパケットの振り分けは行われている．これらから，提案した手法により，大幅なルーチングの変更も求められる長期間の故障への根本的な回避には至らないものの，一時的な輻輳，あるいは故障に関しての適応性，耐故障性が得られたことが分かる．

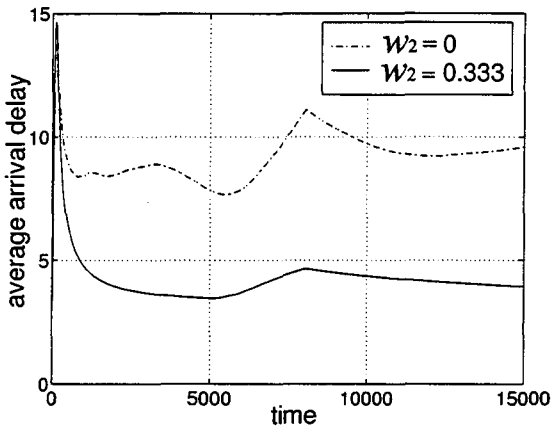


図 3.13: 平均遅延時間の推移（7ノードネットワーク）

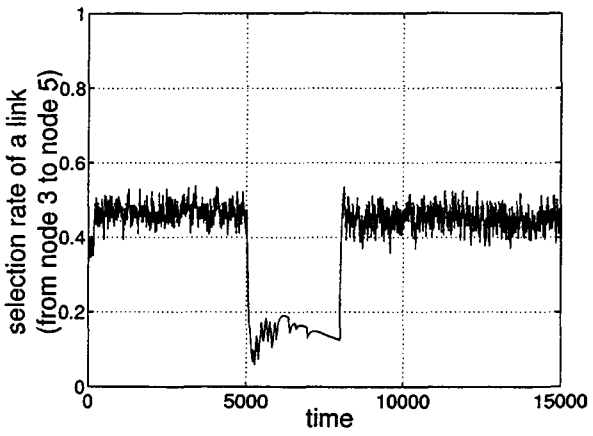


図 3.14: 資源選択確率の推移（7ノードネットワーク）

3.4.4 他方式との性能比較

相対的な性能比較のために，2 つの比較方式を用意し，提案方式との比較を行う．

比較方式

ネットワークの経路選択には大きく分けて，選択経路が常に一定である静的な経路選択アルゴリズムと，状況に応じて変化する適応型の経路選択アルゴリズムの 2 種類がある．それぞれについて比較方式を用意する．

■静的な経路選択アルゴリズム

ここでは，各パケットが目的地まで最短ホップ数で行ける経路を選択する方式を考える．最短ホップ数を距離の単位として考える実装は，例えばインターネット上で RIP(Routing Information Protocol) として採用されている [25]．また今回，最短ホップ数が等しい場合はランダムにいずれかのリンクを選ぶとしている．

■適応型経路選択アルゴリズム

ここでは，一定時間ごとに，各ノードに伝達される他のノード内バッファの状態およびリンクの転送速度，ノード内での処理遅延時間に基づいて最短パスを求める方式を考える [2]．具体的にはまず各ノードにおいてリンク l_r の長さ Len_r を次式で定義する．

$$Len_r = C_r^{-1} + \frac{1}{d}\left(C_r^{-1} \times \frac{\sum_{t=0}^{T-1} B_r(k-t)}{T}\right) \tag{3.4}$$

この式は，リンクの転送時間に，バイアス値として過去 T 時間におけるそのリンクのバッファ内の全パケットの転送にかかる時間の平均値を足した形となっている．d は第 2 項の重みを示す係数である．こうしたうえで，一定時間ごとにネットワーク全体の情報を利用して，Dijkstra のアルゴリズムにより最短経路を求め，パケットは次の経路情報の更新までその経路を選択するとする．この方式は実際には一定時間ごとに自ノードの輻輳情報をブロードキャストしているものと考えられるが，ここでは，リンク情報の伝達によるオーバーヘッドは考慮していない．

ネットワークモデル

対象して2つのネットワークモデルを用意する.

■シミュレーションモデル1

まず, 図 3.15 のような上下左右が環状につながったトーラス型の 10×10 の格子型ノードを用意する. 各リンクの速度は 10 *packets/unit-time* から 40 *packets/unit-time* の範囲でランダムに与える. バッファ容量は 300 とする.

■シミュレーションモデル2

2つ目として. ランダムグラフ法 [6] を用いて生成した図 3.16 のネットワーク (ノード数 30) を用意する. このネットワークは, 1つのノードから出るリンクの数が3から5本に制限されており, また, 転送速度はリンクの距離に比例させることにより, 8から 100 *packets/unit-time* の範囲で与えられている. また, バッファ容量は 150 とする.

以上2つのネットワークモデルを比較してみると, シミュレーションモデル1は最短ホップ数の意味で, とりうる経路の数が多く, シミュレーションモデル2はこれに対し選択の幅が狭くなっているといえる.

設定

設計パラメータについては, モデルの更新と輻輳度の伝達は 30 *unit-times* ごとに行うとする. その他のパラメータは以下のものを用いる.

- $\alpha = \gamma = 0.05$
- $T = 10,\ b = 20$
- $w_1 = w_2 = w_3 = 0.333,\ w_4 = w_5 = 0.5$

この実験では 3.4.3 節とは違い, 最短ホップ数をとれるリンク以外に対してもとくに利益関数に制約を加えてはいない. ただし, パケットの経路選択の際, 通過してきたリンクから再び出ていくことは無いようにしている. 適応型の経路選択方式について

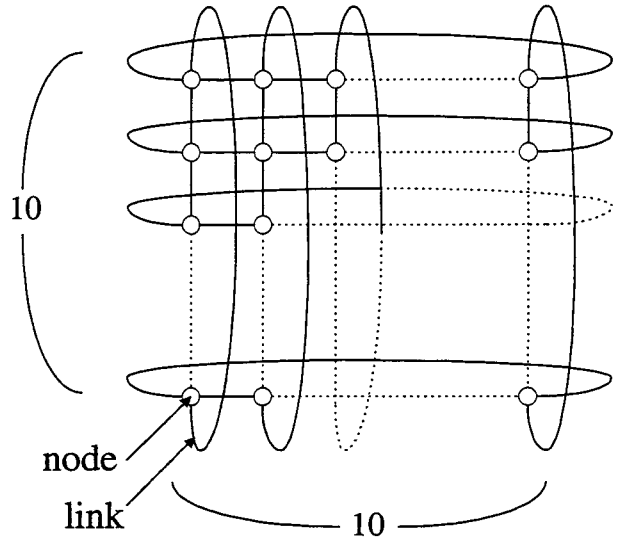


図 3.15: シミュレーションモデル 1

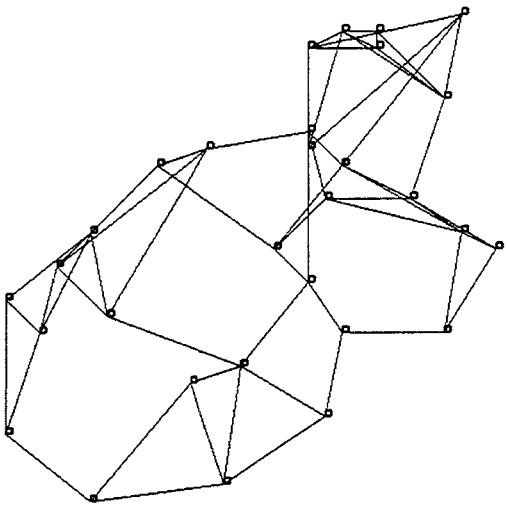


図 3.16: シミュレーションモデル 2

は，モデルの更新は提案方式と同じく 30 *unit-times* ごとに行い，$d = 500$ とする．シミュレーションは 100,000 *unit-times* の間行い，各オーバーフローしたパケットおよび，許容遅延 $W = 100$ を越えたパケットは廃棄する．また，パケットは各ノードで強度 x のポアソン分布に従って発生させ目的ノードはランダムに与えるとする．

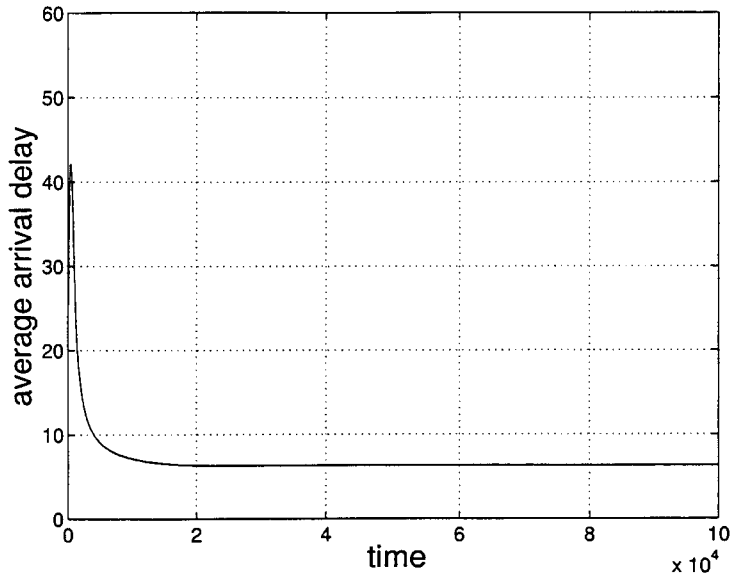
実験結果

図 3.17 はシミュレーションモデル 1 について強度 $x = 10$ のときのパケットの廃棄率と平均遅延時間それぞれの経過時間による変化を示している．Hogg-Huberman モデルに初期値として与えた均等にリンクを選択する状態からネットワークに適応するに従い平均遅延時間および廃棄率が減少していることが分かる．

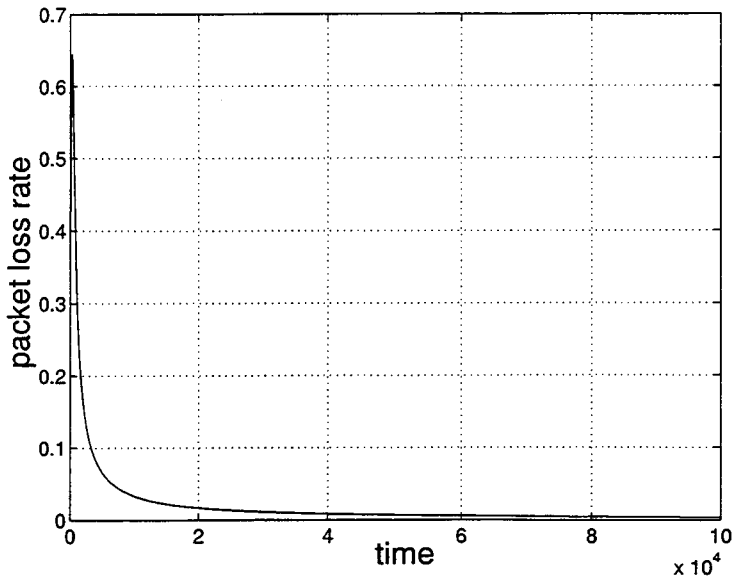
シミュレーションモデル 1 について，パケット発生のポアソン分布の強度を変化させたときのパケットの平均到着遅延の変化と廃棄率の変化を図 3.18 に示す．また，同様のシミュレーションをシミュレーションモデル 2 に対して行った結果を図 3.19 に示す．この際，Hogg-Huberman 戦略による方式については図 3.17 を見ても明らかのように，適切なリンク選択状態に至るまでの過渡状態が存在している．そのため，50000 *unit-times* 経過させた後に実際のシミュレーションを始めている．

シミュレーション結果から，まず，シミュレーションモデル 1 については平均到着遅延時間については提案方式は比較 2 方式の中間に位置し，廃棄率に関しては最も勝れている．一方，シミュレーションモデル 2 については廃棄率に関しては適応型の比較方式と同程度で静的な方式よりも勝れているものの，平均到着遅延では他方式にやや劣っている．適応型の比較方式の性能が勝れているのは，ネットワークの各ノードの完全な情報を定期的に得ているためであり，順当なものであるといえる．

廃棄率と平均到着遅延の間には，廃棄パケットが多くなったため結果として遅延時間が短くなるといったトレードオフが考えられるため，単純な評価はできないが，シミュレーションモデル 2 では，シミュレーションモデル 1 に比べて，全体的に比較方式との相対的な評価が低下している．このことは，シミュレーションモデル 2 のような非均一なグラフに対してはリンクの評価（主に V_{ri}）を画一的に決定するが難しいことに起因していると考えられる．

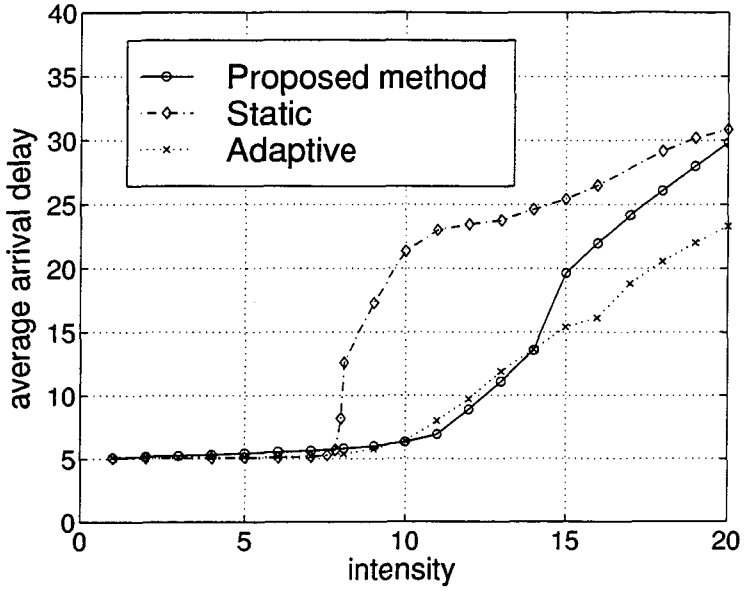


(a) 平均遅延時間の変化

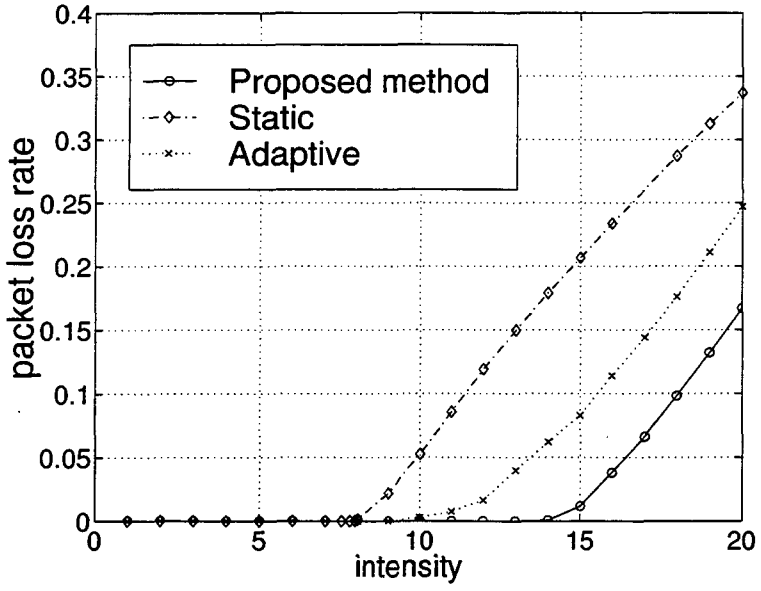


(b) パケットの廃棄率の変化

図 3.17: 環境への適応の様子 (強度 = 10)

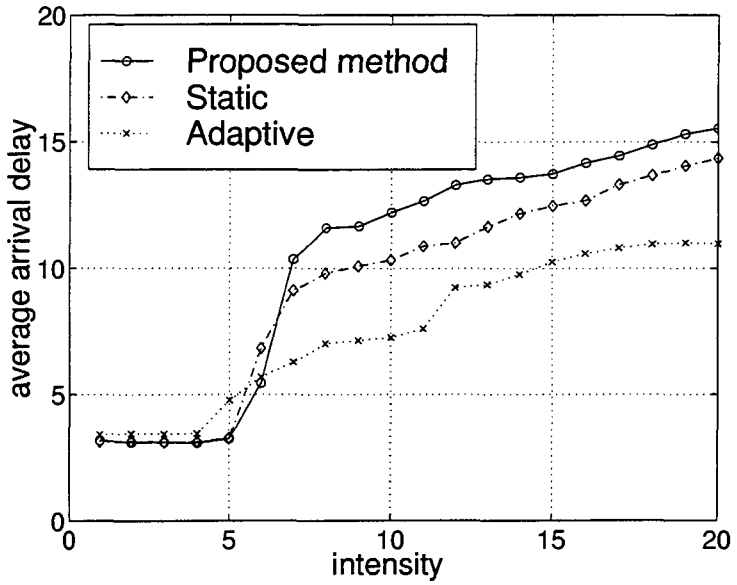


(a) パケットの平均到着遅延

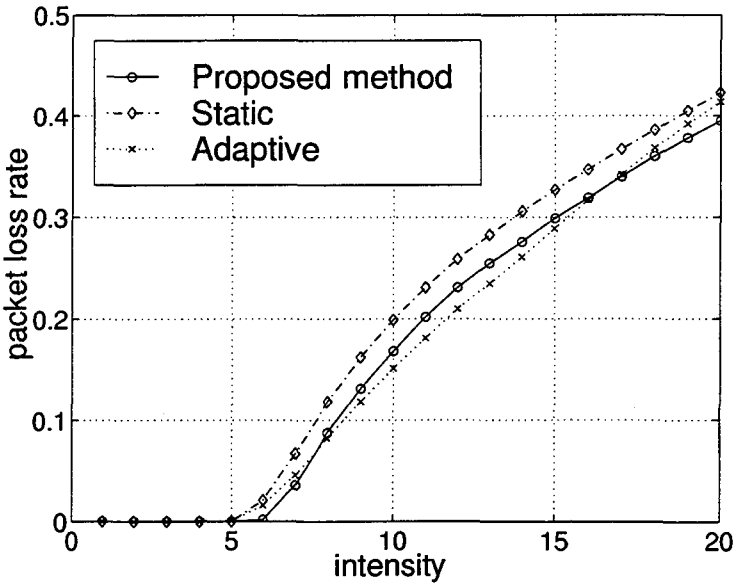


(b) パケットの廃棄率

図 3.18: シミュレーションモデル 1 による性能評価



(a) パケットの平均到着遅延



(b) パケットの廃棄率

図 3.19: シミュレーションモデル 2 による性能評価

3.4.5 考察

まずはじめに提案方式の位置付けについて考察する．提案方式はあらかじめネットワークのトポロジー情報を知っていれば，以後は自ノードの状態および隣接ノードから送られてくる輻輳情報のみで，利益関数の設定以外にはとくに制約も加える必要もなしに，エージェント間の利益の均衡を目指すことを通じてルーチングを実現している．また，Hogg-Huberman モデルによって，環境の変化に対し自律的にルーチングを変更するので，分散環境下での自律的なルーチングを達成していると言える．このことは本方式の利点であるが，一方，大域的な情報は輻輳度という一元化した指標でしか用いていないため，大幅なルーチングの変更はなかなかに難しいと言える．しかし，他ノードからの制御情報を抑えつつ，ある程度の耐故障性は持たせている．そのために純然たる適応型の一方式とは言えないにしても，ある程度トラヒックに対して能動的に振る舞う余地が残されているので，言うなれば完全に静的な方式と，完全に適応的な方式との中間に本論文での提案方式は位置していると考えられる．この解釈は他方式との性能比較のシミュレーション結果がおおよそそのような傾向を示していることからも言えると考える．

計算による負荷に対しては，まず，パケットの振り分けに対しては，通常行われているルーチングテーブルを参照してのパケットの振り分けと，確率に基づく振り分けで，特別に負荷が増加することはないと考える．一方，モデルの更新による負荷であるが，エージェント集団間での陽な交渉は必要としていないため，エージェント集団それぞれの Hogg-Huberman モデルの更新はそれぞれ独立して行うことができ，そのため並列化による高速化が容易にできると考えられる．この際，エージェント集団を，同じ形質を有するもののいくつかでひとまとめとして考えれば，より効率的な運用ができる．ここでも，モデルが各ノードで独立しているため，特に他ノードを意識すること無くそれぞれのノードでエージェント集団のグループ分けを行うことが可能である．

複数存在するパラメータの最適化は重要な問題であるが，α と γ については実験結果から見て小さめに設定しておけば良いようである．戦略の値 S については，定期的に S を変化させつつ資源選択割合の変動の分散が最小になる S を用いるようにするといった手法が考えられる．それ以外のパラメータについても最終的には，自律的

に最適化する仕組みを導入することが望ましいと考える.

3.5 結言

本章では，パケット網の経路選択問題を，パケットをエージェント，リンクを資源としてとらえたマルチエージェントシステムにおける資源配分問題として定式化した．そして，前章で示した制御方式を用いて，拡張された Hogg-Huberman モデルに基づくルーチング方式を提案した．これにより自律的に環境に適応するモデルの下で，エージェント間でのより良い資源配分の達成を目指すことにより，パケットの廃棄率や平均到着遅延などの性能向上を図った.

隣接ノードからの輻輳情報を定期的に取得する機構を導入し，これをエージェントの利益関数に取り込むことにより，分散環境において制御用トラヒックを抑えつつ耐故障性を持たせるように試みた．そしてコンピュータシミュレーションによりいくつかの視点からその挙動を調べた.

今後の課題としては，Hogg-Huberman モデルそのものに対する理論的なアプローチと，現実にある特定のネットワークを想定してのより踏み込んだ形でのルーチング方式の実装とが考えられる.

第4章

強化学習を用いた離散事象システムの最適スーパバイザ制御 ー完全観測の場合ー

4.1 緒言

事象の非同期的，並行的な生起により状態が遷移する離散事象システム（DES; Discrete Event System）は，データベースシステム，通信システム，生産システム，オペレーティングシステムなど様々なシステムに見ることができ，離散事象システムに対する制御手法の研究が活発に行われている [4, 40].

離散事象システムに対する論理的な制御法として，Ramadge と Wonham によって提案されたスーパバイザ制御 [33] がある．スーパバイザ制御において，制御仕様を満足する事象列のみを生起させるように，システムの可制御な事象の生起を許容，禁止する制御器をスーパバイザと呼ぶ．生起を許容する事象の集合を制御パターンという．

一方，近年注目を集めている学習手法のひとつに強化学習がある．強化学習は，環境から受け取る報酬をもとに，学習者たるエージェントが試行錯誤を通じてより良い制御規則を学習していく枠組みを提供している [3, 24, 41].

スーパバイザ制御では，制御仕様を満足するという制約の下に，システムの生成言語が最大になるという意味で最適な制御パターンを指定する．これは，できる限りシ

ステム本来の挙動を制限せずに多くの事象の生起を許容することが望ましいという考えに基づいている．このとき，あらかじめ制御対象および制御仕様が，厳密に形式言語あるいはオートマトンで記述されている必要がある．しかし多くの場合，“こうなって欲しい”という自然言語的に要求される仕様から直接に正確な形式言語での表現を求めるのは簡単ではない．また，スーパバイザ制御は論理的な制御の枠組みであるため，これにより望ましくない状態に至らないように制御をすることはできるが，事象の生起や禁止のためのコストは考慮していない．一方，多くの強化学習では，学習者の受け取る割引報酬の総和の期待値が最大となるという意味で最適な行動政策を，試行錯誤を通じて学習していく．

本章ではスーパバイザの設計を強化学習を用いて行うことにより，事象の生起および禁止のコストを考慮したスーパバイザを獲得する手法を提案する．制御仕様に報酬という比較的与えやすい指標を導入し，制御仕様の詳細は学習によって獲得することにより，スーパバイザの設計の簡単化，自動化を図る．学習は，どの事象を生起させるかではなく，どの事象を生起させても良いかという制御パターンの与え方に対して行う．ここでは，制御パターンを小さく制限するほど，制御対象の挙動が制限され，仕様を満たし易くなるが，事象の生起の禁止によるコストもかかるため，過度の制限もまた望ましくないという状況を想定している．そして，仕様及び事象の生起と禁止に伴うコストを報酬を通じて考慮した中で生成言語を最大とするスーパバイザを構成することを目指す．さらにいくつかの仮定を導入することにより複数の Q 値を同時更新でき，学習速度が向上することを示す．

関連する研究として，鈴木らは，スーパバイザが構成されたシステムに対し，強化学習によりコストが最小となるタスク系列を学習するという手法を提案している [42]．また，あらかじめ構成されたスーパバイザから，事象の生起と禁止に関してのコストの総和を最小化するように生起事象を指定する最適スーパバイザ制御も提案されている [13, 22]．これらに対し，本論文ではスーパバイザそのものの構成の段階から強化学習を用いることで，制御仕様とコストの両面を考慮しての最適な制御パターンを求めている．

以下，4.2 節では，本論文で考える，スーパバイザにより制御されるシステムの構成について述べ，これに基づいて，4.3 節でスーパバイザ構成のアルゴリズムを示す．次に，4.4 節で計算機実験により提案手法の有効性を示し，最後に 4.5 節で本章のま

とめと今後の課題を述べる．また，付録 A でスーパバイザ制御について，付録 B で強化学習について，簡単にまとめておく．

4.2 対象システム

スーパバイザ制御では，制御対象となる離散事象システム（DES）に対し，制御仕様を満たすように，スーパバイザがシステムの可制御な事象の生起を制御する．Ramadge と Wanham によって示された，スーパバイザ制御の枠組みは，図 4.1 で表される．

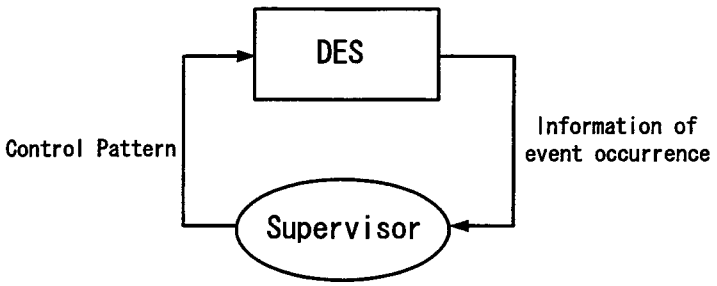


図 4.1: スーパバイザで制御される離散事象システム

制御の基本的な流れは，

1. スーパバイザが，生起を許可する事象の集合（制御パターン）を DES に提示する．
2. DES は制御パターンの中から生起事象を選択，新たな状態に遷移する．8
3. スーパバイザは，DES の生起事象を観測する．

というサイクルによって進められる．なお，スーパバイザ制御に関する基本的事項は付録 A に示す．

最初に，図 4.1 のスーパバイザで制御されるシステムの数理モデルを考える．ここでは制御対象である離散事象システム (DES) が有限マルコフ決定過程 (MDP; Markov Decision Process) であると仮定する．そして，スーパバイザとの間で閉ループを構成している．一般には，対象とする問題において必ずしもマルコフ性の仮定が成り立つとはいえないが，そのような場合でも，問題によっては近似的にマルコフ性が成り立つと考えて MDP の下で構築された強化学習の手法を適用することができ

る．なお，強化学習の一般的な枠組みからすれば，スーパバイザがエージェント，離散事象システムが環境に相当する．

このとき，以下の Bellman 最適方程式が成り立つ．

$$Q^*(x,\pi) = \sum_{x' \in X} \mathcal{P}(x,\pi,x') \left[\mathcal{R}(x,\pi,x') + \gamma \max_{\pi' \in \Pi(x')} Q^*(x',\pi') \right] \qquad (4.1)$$

各記号の意味は，以下の通りである．

- X ： 制御対象である離散事象システム (DES) の状態の集合．なお，本章では DES の状態および，生起する事象が，スーパバイザ側から完全に観測できる状況のみを扱う．
- Σ ： 事象の集合．
- Σ^c ： 可制御な事象の集合．
- Σ^{uc} ： 不可制御な事象の集合．
  （ここで，$\Sigma = \Sigma^c \cup \Sigma^{uc}$, $\Sigma^c \cap \Sigma^{uc} = \phi$.）
- $F(x)$ ： 状態 $x \in X$ において生起可能な事象の集合．
- $\Pi(x)$ ： 状態 $x \in X$ における制御パターンの集合．各制御パターン $\pi \in \Pi(x)$ は，状態 x で生起を許容する事象の集合である．
  （ただし，$\forall \pi \in \Pi(x)$ について，$F(x) \cap \Sigma^{uc} \subseteq \pi \subseteq F(x) \subseteq \Sigma$ を満たす．すなわち，生起可能な不可制御事象がある場合，制御パターンの中に，これは必ず含まれる．よって，各状態における制御パターンの数は最大で $2^{|F(x) \cap \Sigma^c|}$ 個になる．）
- $\mathcal{P}(x,\pi,x')$ ： 状態 $x \in X$ で制御パターン $\pi \in \Pi(x)$ を選択したときに，状態 $x' \in X$ になる確率．
- $Q^*(x,\pi)$: 状態 $x \in X$ で制御パターン $\pi \in \Pi(x)$ を選択し，以後は，各状態で最大の Q 値を持つ制御パターンを選択するときの期待収益（割引付報酬の総和）．
- $\mathcal{R}(x,\pi,x')$ ： 状態 $x \in X$ で制御パターン $\pi \in \Pi(x)$ を選択し，状態 $x' \in X$ に遷移するときに受け取る報酬の期待値．
- γ ： 報酬の割引率 $(0 \leq \gamma < 1)$.

状態の取り方に関しては，DES の状態とスーパバイザの状態を別々に取るというアプローチも考えられる．しかし，この場合は，DES の状態の変化に対し，スーパバイ

ザの状態の変化をどう割り当てるかを決める必要がある．本章では，DES に関する情報をスーパバイザ側から完全に観測できるとしており，簡単のために，DES の状態とスーパバイザの状態を一致させるとした．

ここで，DES はスーパバイザによって与えられた制御パターンの中から生起事象を選択することから，

$$\mathcal{P}(x,\pi,x') = \sum_{\sigma\in\pi} \mathcal{P}_1(x,\pi,\sigma)\mathcal{P}_2(x,\sigma,x')$$

が成り立つ．ただし，

- $\mathcal{P}_1(x,\pi,\sigma)$ は，状態 $x \in X$ で，スーパバイザが制御パターン $\pi \in \Pi(x)$ を選択したとき，事象 $\sigma \in \pi$ が DES によって選択される確率，
- $\mathcal{P}_2(x,\sigma,x')$ は，状態 $x \in X$ で，事象 $\sigma \in F(x)$ が生起したとき，状態 $x' \in X$ に遷移する確率，

である．

提案手法においては，DES に対して以下の 2 つの仮定を設ける．

[仮定 4.1]

1. 各状態 $x \in X$ について，事象の選ばれやすさを表すパラメータとして，$\eta^*(x,\sigma)$ を導入する．このとき，

$$\mathcal{P}_1(x,\pi,\sigma) = \frac{\eta^*(x,\sigma)}{\sum_{\sigma'\in\pi}\eta^*(x,\sigma')} \tag{4.2}$$

$$\forall\sigma \in F(x)\ \ \eta^*(x,\sigma) > 0\,,\ \sum_{\sigma\in F(x)} \eta^*(x,\sigma) = 1$$

の関係が成り立っているとする．すなわち，スーパバイザが選択した制御パターンの中から DES は生起事象を選択するが，このとき事象を選択する比率は，与えられた制御パターンに依存せず一定である．ただし，η^* の真値をスーパバイザは知らない．

2. 報酬 $\mathcal{R}(x,\pi,x')$ について，

$$\mathcal{R}(x,\pi,x') = \mathcal{R}_1(x,\pi) + \mathcal{R}_2(x,\sigma,x') \tag{4.3}$$

という構造をもつとする．ここで，$\mathcal{R}_1, \mathcal{R}_2$ の意味は以下の通りである．

- $\mathcal{R}_1(x,\pi)$: 状態 x で制御パターン π を選んだことによる報酬の期待値．制御パターンの与え方に依存して決定される．直観的には制御パターンに含まれない事象を禁止したことに伴うコストを表している．
- $\mathcal{R}_2(x,\sigma,x')$: 状態 x で事象 σ が生起され，状態 x' に遷移したときの報酬の期待値．直観的には事象の生起に伴うコスト及び，タスクの出来不出来に対する評価を表している．

仮定 4.1 より，(4.1) 式は，

$$\begin{aligned} Q^*(x,\pi) &= \sum_{x'\in X}\left(\sum_{\sigma\in\pi}\frac{\eta^*(x,\sigma)}{\sum_{\sigma'\in\pi}\eta^*(x,\sigma')}\mathcal{P}_2(x,\sigma,x')\right) \\ &\qquad \left[\mathcal{R}_1(x,\pi)+\mathcal{R}_2(x,\sigma,x')+\gamma\max_{\pi'\in\Pi(x')}Q^*(x',\pi')\right] \\ &= \mathcal{R}_1(x,\pi)+\sum_{\sigma\in\pi}\frac{\eta^*(x,\sigma)}{\sum_{\sigma'\in\pi}\eta^*(x,\sigma')}\sum_{x'\in X}\mathcal{P}_2(x,\sigma,x') \\ &\qquad \left[\mathcal{R}_2(x,\sigma,x')+\gamma\max_{\pi'\in\Pi(x')}Q^*(x',\pi')\right] \\ &= \mathcal{R}_1(x,\pi)+\sum_{\sigma\in\pi}\frac{\eta^*(x,\sigma)}{\sum_{\sigma'\in\pi}\eta^*(x,\sigma')}T^*(x,\sigma) \end{aligned} \tag{4.4}$$

となる．ここで，$T^*(x,\sigma)$ は，状態 x で事象 σ が生起し，以後は，各状態で最大の Q 値を持つ制御パターンを選択するときの期待収益を表しており，

$$T^*(x,\sigma)=\sum_{x'\in X}\mathcal{P}_2(x,\sigma,x')\left[\mathcal{R}_2(x,\sigma,x')+\gamma\max_{\pi'\in\Pi(x')}Q^*(x',\pi')\right] \tag{4.5}$$

で定義される．$T^*(x,\sigma)$ の中には状態 x で制御パターン π を選択した事による報酬は含まれていない．

仮定 4.1 の 2 つの仮定は，制御パターンの与え方の学習を行う際に，効率的な学習を行えるように導入した．仮定を導入したことで，(4.4) 式より，Q^* は，T^*, $\mathcal{R}_1$, η^* を用いて求めることができ，Q^* を推定する際に，これを直接推定するのではなく，T^*, $\mathcal{R}_1$, η^* を推定し，そこから間接的に Q^* を求めることが可能となった．このとき，T^* と η^* は複数の Q^* に影響を与えているので，一度の経験から，複数の Q^* を更新することができる．制御パターン数は生起可能な事象数が増えると指数的に増加するので，学習効率を上げることは重要である．

まず，1. の仮定であるが，制御パターンの中からどの事象が生起するかは，従来のスーパバイザ制御では単に非決定であるとのみされている．しかし今回，マルコフ過

程を考えているので，事象の選択に確率を導入することが必要となる．ここで，制御パターンの与え方に依存して各事象の選択確率が変わってくるというのが一般的な形である．しかし，その場合は，各 Q^* の間に同時更新を可能とするような関係を持たすことができず，それぞれの Q^* を個別に学習していくことが必要になってしまう．そこで，1. のように，制御パターンの与え方に依存せず，事象の選択比率は一定という仮定を導入した．これにより，複数の Q^* の同時更新が可能になった．ただし，η^* が存在すると仮定はしているが，その値そのものは既知ではなく，学習によって推定するとしているので，過度にきつい仮定ではないと考える．次に，2. の仮定についてであるが，今回，事象の生起と禁止のコストの両方を考慮したスーパバイザの設計を目指しているので，生起と禁止のコストを別に考えるという仮定は，目的の上からも必要となる．また実際のシステムにおいては，受け取る報酬の中のコストを反映させている部分は，使用したエネルギーなどをセンサから獲得して利用していると考えられる．そのため，コストの評価が個別に可能なようにセンサが設置され，利用できる場合には，この仮定は成り立つと考えられる．

4.3 スーパバイザ学習のアルゴリズム

4.3.1 提案アルゴリズムの概要

本章で提案する強化学習を用いた離散事象システムのスーパバイザ制御の概念図を図 4.2 に示す．学習の目的は，どの事象を生起させるかを学習することではなく，制御パターンの与え方を学習することにある．スーパバイザは DES に対して，現在の状態 x の Q 値をもとに制御パターン π を決定し，DES に示す．DES は π の中から生起事象 σ を選択する．このとき，DES はスーパバイザに生起事象 σ と状態遷移 $x \xrightarrow{\sigma} x'$，2 種類の報酬 r_1, r_2 の情報を伝達する．これらをもとに，スーパバイザは Q 値を更新する．前節の仮定 4.1 を利用することで，複数の Q 値の同時更新を行う．終了状態に至るまで，新たな状態で同様の手順を繰り返す．また，最低限満足したい仕様 $Spec$ があった場合は，これを満たさない状態遷移については強制的な枝刈りを行うとする．

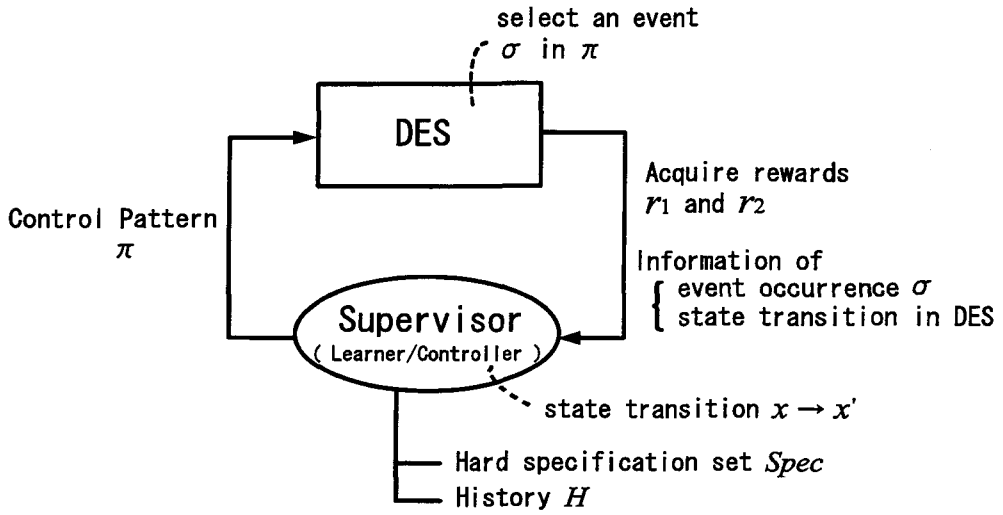


図 4.2: 学習スーパバイザによって制御される離散事象システム

4.3.2 アルゴリズムの詳細

提案する学習アルゴリズムを図 4.3 に示す．以下，図 4.3 に沿ってアルゴリズムの詳細を述べる．

1. 最初に，各 $T(x,\sigma)$, $R_1(x,\pi)$, $\eta(x,\sigma)$ に初期値を与える．事前知識がある場合は，それらを反映させた値を設定すればよい．
2. 各状態での Q 値の初期値を，次式により求める．

$$Q(x,\pi) \leftarrow R_1(x,\pi) + \sum_{\sigma\in\pi} \frac{\eta(x,\sigma)}{\sum_{\sigma'\in\pi}\eta(x,\sigma')} T(x,\sigma) \tag{4.6}$$

3. ここからが学習の主要部分であり，各エピソードにおける手順となる．エピソード (episode) は，初期状態 (initial state) から始まり，終端状態 (terminal state) で終わるひと続きの系列を表す．エピソードを多数回経験することで学習が進んでいく．十分な性能のスーパバイザの設計ができたと判断されるまで，あるいは，環境が変化する場合には，継続的に 3. の手順が繰り返される．
   (a) 後で viii. で用いる履歴 H を空にする．
   (b) DES の初期状態を x にセットする．
   (c) この部分が，エピソード内の各ステップにおける手順となる．状態 $x \in X$ が DES の現在の状態を示しており，x が終端状態にいたるまで (c) 内の手順を繰り

返す.

i. 学習者であるスーパバイザは制御パターン $\pi \in \Pi(x)$ として，生起を許容する事象の集合を DES に提示する．今回，制御パターンの選択には，ϵ-greedy 選択を用いるとした．これは，確率 $1-\epsilon$ で最大の Q 値を持つ制御パターンを，確率 ϵ でランダムにいずれかの制御パターンを選択する．

ii. DES の側で，生起させる事象 $\sigma \in \pi$ を選択する．この選択は，スーパバイザではなく DES 側が生起を許容された事象の範囲内 π の中から選択するが，仮定 4.1 の 1 にしたがった選択である．

iii. 報酬 r_1, r_2 を獲得する．ただし，r_1 は制御パターン π に対する報酬，r_2 は生起事象 σ に対する報酬である．

iv. DES の状態 x が x' に遷移する．

v. 状態 x で事象 σ が生起したという履歴情報を H に追加する．

vi. $T(x,\sigma)$, $\mathcal{R}_1(x,\pi)$, $\eta(x,\dot{\sigma}')$ の更新を行う．提案アルゴリズムは Q-learning の更新式に基づいているが，仮定 4.1 のもとで学習効率の改善を図っている．(4.4) 式より，Q^* は，T^*, $\mathcal{R}_1$, η^* を用いて求めることができる．そこで，提案アルゴリズムでは，

$$T(x,\sigma) \leftarrow T(x,\sigma) + \alpha[r_2 + \gamma \max_{\pi' \in \Pi(x')} Q(x',\pi') - T(x,\sigma)] \tag{4.7}$$

$$R_1(x,\pi) \leftarrow R_1(x,\pi) + \beta[r_1 - R_1(x,\pi)] \tag{4.8}$$

$$\text{For all} \quad \sigma' \in \pi$$

$$\eta(x,\sigma') \leftarrow \begin{cases} (1-\delta)\,\eta(x,\sigma') & (\text{ if } \sigma' \neq \sigma\ ) \\ \eta(x,\sigma') + \delta\left[\sum_{\sigma'' \in \pi} \eta(x,\sigma'') - \eta(x,\sigma')\right] & (\text{ if } \sigma' = \sigma) \end{cases} \tag{4.9}$$

として，T, R_1, η を推定する．α, β, δ は学習率である．

vii. 先に vi. で更新された情報を用いて，実際に選択した制御パターン π のみではなく，π に含まれる事象を含む全制御パターンに対し同時に Q 値の更新を行う．すなわち，

$$\text{For all } \pi' \in \Pi(x)\ s.t.\ \pi' \cap \pi \neq \emptyset$$

$$Q(x,\pi') \leftarrow R_1(x,\pi') + \sum_{\sigma'' \in \pi'} \frac{\eta(x,\sigma'')}{\sum_{\sigma''' \in \pi'} \eta(x,\sigma''')} T(x,\sigma'') \tag{4.10}$$

として間接的に Q 値を推定する．

viii. エピソードの各ステップの最後に，現在の状態が最低限の要求仕様として定める状態の集合 $Spec \subseteq X$ を満たしているかどうかをチェックする．満たされていない場合は，A. から C. を行う．$Spec$ は，報酬という形で与えられる仕様とは別のものである．特に必要が無ければ，$Spec = X$ とすればよい．ここでは，強制的に $Spec$ を満たさない事象列を取り除くという，一種の枝刈りを行っている．

A. 履歴 H から，最後に生起した可制御事象 $\sigma^c \in \Sigma^c$ と，そのときの状態 w を特定する．

B. σ^c を w における生起可能な事象の集合 $F(w)$ から削除する．

C. 状態 w における η の和が 1 となるように，各 η の値を更新し，Q 値の再計算を行う．

ix. 現在の状態を x とする．

アルゴリズム全体の構成としては，1. と 2. が全体に関する初期化で，3. が 1 つのエピソードに対する処理となる．3(a), 3(b) が各エピソードに関する初期化となり，エピソード内の各ステップでの手順が 3(c) であり，DES が終了状態に至るまで 3(c) の手順を繰り返す．

1. Initialize $T(x,\sigma), R_1(x,\pi)$, and $\eta(x,\sigma)$ at each state.
2. Calculate the initial Q value at each state by

$$Q(x,\pi) \leftarrow R_1(x,\pi) + \sum_{\sigma\in\pi} \frac{\eta(x,\sigma)}{\sum_{\sigma'\in\pi}\eta(x,\sigma')} T(x,\sigma)$$

3. Repeat until x is *terminal state* (for each episode) :
   (a) Clear history H.
   (b) $x \leftarrow$ *initial state.*
   (c) Repeat (for each step of episode)
      i. Select a control pattern $\pi \in \Pi(x)$ based on the Q value by the supervisor.
      ii. Observe event occurrence $\sigma \in \pi$ and state transition in DES.
      iii. Acquire rewards r_1 and r_2.
      iv. Make transition $x \xrightarrow{\sigma} x' \in X$.
      v. Add (x,σ) to history H.
      vi. Update $T(x,\sigma)$, $\mathcal{R}_1(x,\pi)$, and $\eta(x,\sigma')$:

$$T(x,\sigma) \leftarrow T(x,\sigma) + \alpha[r_2 + \gamma \max_{\pi'\in\Pi(x')} Q(x',\pi') - T(x,\sigma)]$$

$$R_1(x,\pi) \leftarrow R_1(x,\pi) + \beta[r_1 - R_1(x,\pi)]$$

For all $\sigma' \in \pi$

$$\eta(x,\sigma') \leftarrow \begin{cases} (1-\delta)\eta(x,\sigma') & (\text{if } \sigma' \neq \sigma) \\ \eta(x,\sigma') + \delta\left[\sum_{\sigma''\in\pi}\eta(x,\sigma'') - \eta(x,\sigma')\right] & (\text{if } \sigma' = \sigma) \end{cases}$$

      vii. Update the Q values

For all $\pi' \in \Pi(x)$ *s.t.* $\pi' \cap \pi \neq \emptyset$

$$Q(x,\pi') \leftarrow R_1(x,\pi') + \sum_{\sigma''\in\pi'} \frac{\eta(x,\sigma'')}{\sum_{\sigma'''\in\pi'}\eta(x,\sigma''')} T(x,\sigma'')$$

      viii. If $x' \notin Spec$
         A. Search the latest controllable event $\sigma^c \in \Sigma^c$ and the corresponding state $w \in X$ from the history H.
         B. Remove σ^c from the feasible event set $F(w)$.
         C. Normalize $\eta(w,\sigma')$ so as to satisfy $\sum_{\sigma'\in F(w)}\eta(w,\sigma') = 1$ and update the Q values at the state w.
      ix. $x \leftarrow x'$

図 4.3: スーパバイザ獲得のアルゴリズム

4.3.3 アルゴリズムに対する考察

提案アルゴリズムでは，DES の状態に対してスーパバイザは制御パターンを決定するという枠組みになっており，状態フィードバック制御を行っている．そして，複数の Q 値の同時更新による学習効率の向上が期待できる．また，与えられた制御パターンの中から，いずれかの生起事象が選択されるという構成になっているため，単に制御パターンを行動ととらえて学習しただけでは受け取る報酬の分散が大きくなり学習性能が劣化すると考えられるが，提案手法では，その影響を抑えることができる．

今回，制御パターンの選択には ϵ-greedy 選択を用いるとしており，その場合は，T, R_1, η を保持しておけば，Q 値については，最大の Q 値の情報のみを保持することで，メモリ効率を改善させることができる．また，制御パターンの選択に Boltzman 選択のように，すべての Q 値を必要とする手法を用いた場合には，計算時間を優先するならば全 Q 値を保持する方式が，メモリ優先の場合には，必要になった時点で毎回 Q 値を計算する方式が考えられる．

提案するアルゴリズムは，ある厳密な仕様 $Spec$ の下で，期待収益の点で最適となる制御パターンを学習していくことを通じ，制御仕様とコストを考慮した中で，生成言語を最大とするスーパバイザを獲得するアルゴリズムであるといえる．

強化学習との関連でとらえると，制御パターンのそれぞれを行動と考え Q-learning を用いて最適な制御パターンを求める方法に対し，提案手法では仮定 4.1 を利用することによって，複数の Q 値の同時更新を行い，学習速度の向上を図っている．その上で，枝刈りの仕組みを付け足したものと位置付けることが出来る．あるいは，$T(x, \sigma)$ を従来考えていた Q 値ととらえると，スーパバイザ制御を導入するために，R_1, η というパラメータを用いて，従来の Q-learning の枠組みを拡張したものととらえることも出来る．また，提案手法では，最終的には Q 値を推定しているので，他の Q 値を学習させるタイプの強化学習法と親和性が良く，それらと組み合わせての利用も考えられる．

4.4 実験

2種類の例題に対し，計算機実験を行った．なお，今回は，報酬に基づく学習の部分の評価に重点を置いており，*Spec* による枝刈りは用いていない．

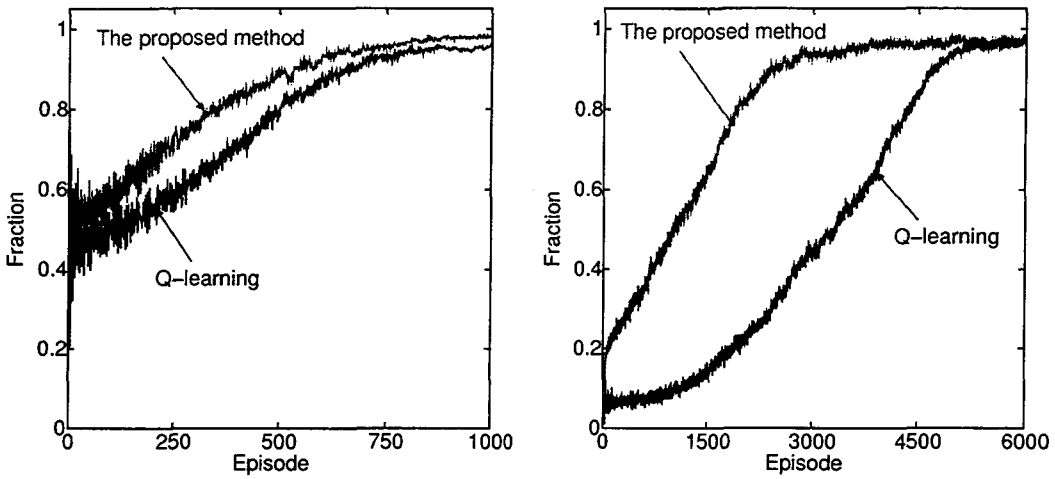
4.4.1 n 本腕バンディット問題

最初の例題として，n 本腕バンディット問題を考える [24]．DES は n 種類の行動から1つを選択し報酬を受け取る．報酬は，行動ごとに定められた平均値を持つ正規分布にしたがって返される．通常の n 本腕バンディット問題では，期待報酬が最大となる行動を n 個の行動選択肢の中から学習することを目的とするが，今回，全ての行動を可制御とし，$2^n - 1$ 個の制御パターンの与え方の中から学習する．

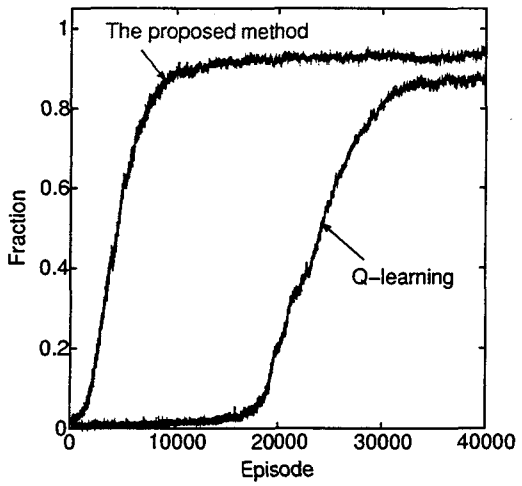
図 4.4 は，それぞれ $n = 2$, $n = 5$, $n = 8$ の場合について提案手法と通常の Q-learning とで性能の差を比較したものである．これは制御パターン数で考えると，それぞれ 3 個，32 個，255 個となる．グラフの横軸はエピソード数を，縦軸は期待収益が最大となる制御パターンが求まった割合を示している．本例題では，1 エピソードは 1 回の制御パターンの選択で終了する．それぞれについて 10 個の異なる問題を生成し，さらに各問題について 100 回の学習を行った結果を平均した．問題の生成にあたっては，各事象の生起に対する報酬は 0 から-100 の範囲でランダムに設定した．制御パターンの与え方に対する報酬は，各事象の生起の禁止に対する報酬を 0 から-20 の範囲でランダムに設定し，制御パターンで生起を禁止された事象に関してこの和を取ることで決定した．各 $\eta*$ は平均 $1/n$，分散 1 の正規分布にしたがって生成した乱数をもとに，$\sum \eta^* = 1$ になるように調整した．ただし，最低でも各 η^* が 0.05 以上となるようにしている．また，受け取る報酬は，真値を平均とし分散を 0.1 とした正規分布にしたがって返すとした．いずれの学習率も 0.01 に固定し，Q 値の初期値は 0 とした．制御パターンの選択には，$\epsilon = 0.1$ の ϵ-greedy 選択を用いた．

グラフより，提案手法は Q-learning よりも学習速度が速く，制御パターン数が多くなるほど，その差が顕著になっていることがわかる．この結果から，制御パターン数が少ない場合は計算が少なくてすむ通常の Q-learning を用い，制御パターンが多いところでは提案手法を用いて学習するといった使い分けも考えられる．なお，最終

的に確率 1 で最適解を学習できていないのは，学習率が (B.2) 式の条件を満たしておらず，生成した問題のいくつかで最適解と準最適解との差が僅少であるためと考えられる．実際，学習率を徐々に減らしていった場合，時間はかかるものの両手法とも最適解を求めることができている．



(a) n=2 の場合　　(b) n=5 の場合



(c) n=8 の場合

図 4.4: n 本腕バンディット問題におけるエピソード数とスーパバイザが最適制御パターンを見つけた割合の関係

4.4.2 猫とねずみの迷路の問題

前節の例題では，1回の行動で1つのエピソードが終了するので，報酬の時間遅れという要素がなく，また，すべての事象が可制御であった．ここでは，スーパバイザ制御の例題としてしばしば用いられる猫とねずみの迷路の問題 [33] に対して，提案手法を適用する．図 4.5 で示す，ドアで区切られた5つの部屋がある．猫とねずみが存在し，矢印の方向にのみ移動が許された専用のドアが用意されている．図 4.5 において，c1～c7 が猫用のドア，m1～m6 がねずみ用のドアである．部屋 1-3 間のドア c7 のみ不可制御なドアで，他は可制御なドアである．このとき，猫とねずみが同じ部屋に入ることの無いようにドアの開閉を制御することを学習させたい．初期状態では，猫は部屋2に，ねずみは部屋4にいるとし，報酬として，ドアを閉じるにはドア1つにつき平均-1，分散 0.1 の正規分布にしたがうコストがかかるとした．また，ひとつのエピソードは 20 回移動を終えた場合か制御仕様を満たさなくなった場合，すなわち，猫とねずみが同じ部屋になった場合（-100 の報酬を与える）に終了するとした．学習率はいずれも 0.1 を用い，$\gamma = 0.9$ とした．制御パターンの選択には，$\epsilon = 0.1$ の ϵ-greedy 選択を用いた．

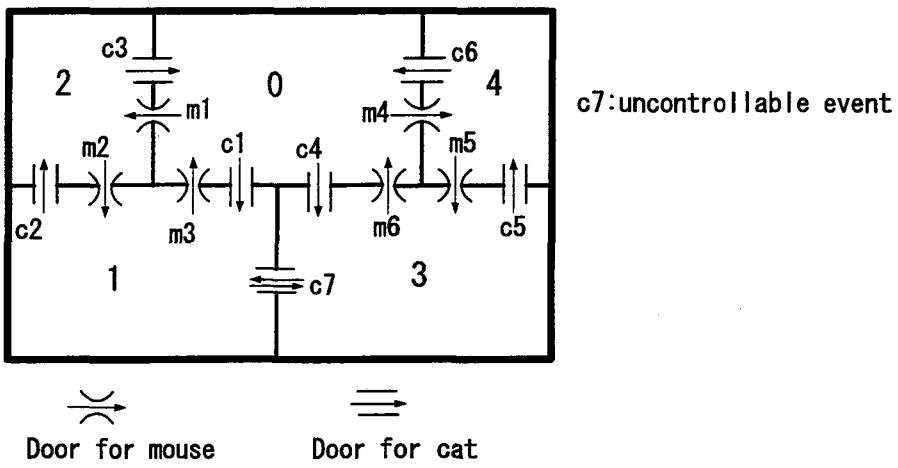


図 4.5: 猫とねずみの迷路の問題

図 4.6 は，横軸にエピソードを，縦軸に文献 [33] で示されたスーパバイザが求まった割合を示している．結果は 100 回の学習の平均である．最終的には両手法ともこのスーパバイザが求まっているが，提案手法の方が Q-learning よりも速く，求まる

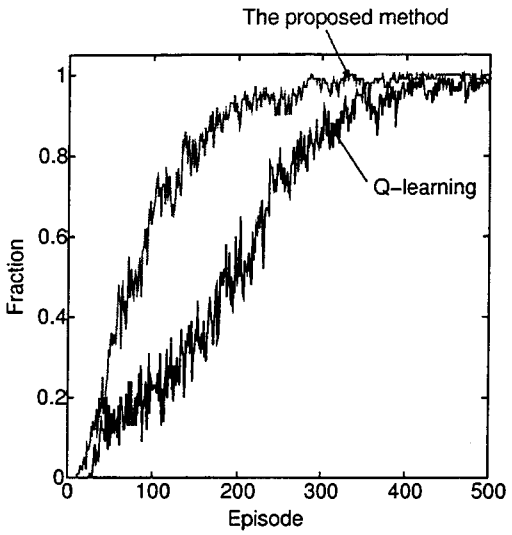


図 4.6: 猫とねずみの迷路の問題におけるエピソード数とスーパバイザが最適制御パターンを見つけた割合の関係

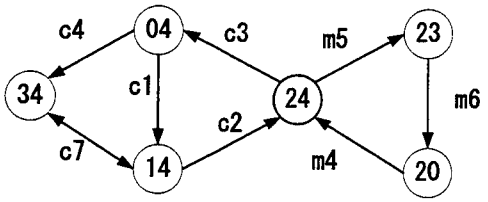


図 4.7: 獲得された制御パターンの状態遷移図

確率が 1 に収束している．また，図 4.7 は，学習の結果得られた制御パターンを状態遷移図で表したものである．円内の数値の 10 の位が猫のいる部屋を，1 の位がねずみのいる部屋を表しており，遷移の矢印が，各状態での生起を許可する事象を示している．猫とねずみが同じ部屋に居ないという論理的な制御仕様を満たす制御パターンを提示するスーパバイザが学習を通じて獲得されている．

4.5 結言

本章ではスーパバイザの制御機構の特性を利用した，Q-learning に基づくスーパバイザの構成法を提案した．スーパバイザ制御を応用するとき，正確に制御仕様を記述することが重要な問題点となっている．提案手法では，制御仕様の詳細は報酬を元に学習を通じて獲得するとした．スーパバイザの設計を強化学習を用いて行うことにより，環境が不確かであったり，変化する場合に対してスーパバイザ制御を適用する

ための一つの方法論を示した.

今後の課題としては, より詳細なアルゴリズムの検討と, 性能向上が挙げられる. 特に, 事象の数が増えると, それに伴い制御パターンの数が急激に増大するので, さらに効率的な学習方法の開発は重要な検討課題である. 最大可制御言語を生成するスーパバイザを提案手法によって求められるかを理論的に明らかにすることも重要な課題である. また, 事象の生起について部分観測な場合を考慮したアルゴリズムの開発も必要であるが, これに関しては第5章で述べる.

第5章

強化学習を用いた離散事象システムの最適スーパバイザ制御 ー部分観測の場合ー

5.1 緒言

4章では，制御対象となる離散事象システム（DES）の状態が，スーパバイザ側から完全に観測できる場合に関し，強化学習を用いて事象の生起／禁止コストと制御仕様の両方を考慮したスーパバイザの獲得手法を提案した．しかし，このような完全観測の状況が一般に成立するとは限らない．例えば，DES内部でのいくつかの事象の生起は，センサの設置コストや物理的な制約から，スーパバイザ側からは観測できないかもしれない．また，DESの内部では二つの異なる事象であっても，スーパバイザ側からは同じ一つの事象であるように観測されてしまうこともありうる．これらの状況を扱うためには，4章で提案した制御手法を，必ずしもスーパバイザがDESの事象の生起の全てを観測できないという，部分観測環境に拡張することが必要となる．部分観測環境を表現するために本章では，生起事象から不可観測事象を除き，可観測事象のみを出力する射影関数を導入する．そして，スーパバイザは，射影関数を通して得られる観測情報のみを用いて，制御パターンの学習を行う．ここでは，4章の場合と異なり，DESの状態とスーパバイザの状態は異なるものとして扱われる．

制御仕様は正確に言語で与えられるのではなく，試行錯誤による経験から得られる

報酬をもとに詳細を学習していく．そして，強化学習により，最適な制御パターンを求める．また，いくつかの仮定を導入することで学習の高速化を図る．さらに，部分観測環境も対象とすることで，より汎用的な，強化学習を用いた最適スーパバイザ制御を提案する．

部分観測のもとでの，言語で与えられた論理的な制御仕様を満たすスーパバイザ制御に関しては，多くの研究成果が得られている [21]．また，Marchand らは完全観測の場合の最適スーパバイザ制御 [22] を部分観測環境に拡張することで，事象の生起と禁止のコストの総和を最小化する最適スーパバイザ制御を提案している [17]．

以下，5.2 節では，前章で示した完全観測の場合のシステムのモデルを部分観測環境に拡張する．そして，5.3 節で学習のアルゴリズムを示す．5.4 節で，簡単な計算機実験により提案手法の有効性を調べ，最後に 5.5 節で本章のまとめと今後の課題を述べる．

5.2 部分観測環境システム

先に，4 章で示したスーパバイザで制御されるシステムの数理モデルは，概念的には図 4.1 で示したように，DES における事象の生起の情報の全てを，スーパバイザは直接得るとしていた．すなわち，完全観測の場合を考えていた．しかし，一般には様々な制約から，完全観測の状況が常に成立するとは考え難く，スーパバイザは DES 内部での変化の一部しか観測できない，すなわち部分観測の場合を考えることが必要となる．

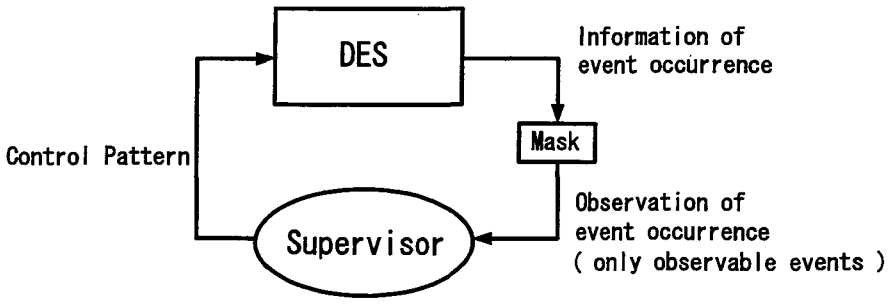


図 5.1: スーパバイザで制御される離散事象システム（部分観測の場合）

そこで新たに，図 5.1 のように，射影関数を導入してスーパバイザは，射影（マスク）を通してしか DES の情報を得られないとする．そして，このシステムの数理モ

デルを考える．ここでは，スーパバイザの観測情報に関して，システムが MDP であると仮定する．

このとき，(4.1) 式を DES 内部での変化に対して部分観測の場合に拡張して考えることで，以下の Bellman 最適方程式が成立する．

$$Q^*(s,\pi) = \sum_{s' \in S} \mathcal{P}(s,\pi,s') \left[\mathcal{R}(s,\pi,s') + \gamma \max_{\pi' \in \Pi(s')} Q^*(s',\pi') \right] \tag{5.1}$$

各記号の意味は，以下の通りである．

- X : 制御対象である離散事象システム (DES) の状態の集合．本章では，スーパバイザが知ることはできないとする．
- S : スーパバイザの状態の集合．DES の状態集合の部分集合で表される．すなわち，$S \subseteq 2^X$ である．スーパバイザは DES の正確な状態を知ることはできないが，状態の候補は知ることができ，候補となる DES の状態の組でスーパバイザの状態が表現される．
- Σ : 事象の集合．
- Σ^c : 可制御な事象の集合．
- Σ^{uc} : 不可制御な事象の集合．
- Σ^o : 可観測な事象の集合．
- Σ^{uo} : 不可観測な事象の集合．

  (ただし，$\Sigma = \Sigma^c \cup \Sigma^{uc}$, $\Sigma^c \cap \Sigma^{uc} = \phi$, $\Sigma = \Sigma^o \cup \Sigma^{uo}$, $\Sigma^o \cap \Sigma^{uo} = \phi$. とする．事象の可制御と可観測の間の関係には特に制限を設けていない．)
- $F(s)$: 状態 $s \in S$ において生起可能な事象の集合．
- $\Pi(s)$: 状態 $s \in S$ における制御パターンの集合．各制御パターン $\pi \in \Pi(s)$ は，状態 s で生起を許容する事象の集合である．

  (ただし，$\forall \pi \in \Pi(s)$ について，$F(s) \cap \Sigma^{uc} \subseteq \pi \subseteq F(s) \subseteq S$ を満たす．すなわち，生起可能な不可制御事象がある場合，制御パターンの中に，これは必ず含まれる．よって，各状態における制御パターンの数は最大で $2^{|F(s) \cap \Sigma^c|}$ 個になる．)
- $\mathcal{P}(s,\pi,s')$: 状態 $s \in S$ で制御パターン $\pi \in \Pi(s)$ を選択したときに，状態 $s' \in S$ になる確率．

- $Q^*(s,\pi)$: 状態 $s \in S$ で制御パターン $\pi \in \Pi(s)$ を選択し，以後は，各状態で最大の Q 値を持つ制御パターンを選択するときの期待収益.
- $\mathcal{R}(s,\pi,s')$: 状態 $s \in S$ で制御パターン $\pi \in \Pi(s)$ を選択し，状態 $s' \in S$ に遷移するときに受け取る報酬の期待値.
- γ : 報酬の割引率 $(0 \leq \gamma < 1)$.

そして新たに，スーパバイザが観測した出力事象に対して，DES の現在の状態を推定する仕組みを導入する.

まず，以下のように各記号を定義する.

- f : $X \times \Sigma \to X$，DES の状態遷移関数. 本章では，スーパバイザにとって既知であると仮定する. さらに，事象列に対しても用いることができるように，f : $X \times \Sigma^* \to X$ として，以下のように拡張する.

$$f(x,\epsilon) = x$$
$$f(x,t\sigma) = f(f(x,t),\sigma) \quad \text{for} \quad t \in \Sigma^* \quad \text{and} \quad \sigma \in \Sigma$$

- $x_0 \in X$: DES の初期状態.
- Σ^* : Σ の Kleene 閉抱. すなわち，空列 ϵ を含む，Σ の要素からなるすべての有限列の集合.
- M_e: $\Sigma \to \Sigma^o \cup \{\epsilon\}$，DES 内部で生起した事象 σ に対し，スーパバイザの観測事象 $\sigma^o \in \Sigma^o$（または空事象 ϵ）を返す射影関数.

$$M_e(\sigma) = \begin{cases} \sigma & \text{if} \quad \sigma \in \Sigma^o \\ \epsilon & \text{if} \quad \sigma \in \Sigma^{uo} \end{cases} \tag{5.2}$$

さらに，$M_e : \Sigma \to \Sigma^{o*}$ を以下のように帰納的に定義する.

$$M_e(\epsilon) = \epsilon$$
$$\forall t \in \Sigma^*, \forall \sigma \in \Sigma$$
$$M_e(t\sigma) = \begin{cases} M_e(t)\sigma & \text{if} \quad \sigma \in \Sigma^o \\ M_e(t) & \text{if} \quad \sigma \in \Sigma^{uo} \end{cases}$$

M_e により，DES で生起した事象列 t に対し，$M_e(t)$ は t より不可観測事象を取り除いた，スーパバイザが観測する事象列を返す.

- M_s: $\Sigma^{o*} \to S$, 観測事象列 $t \in \Sigma^{o*}$ に対し，スーパバイザの状態を返す関数 M_s を以下のように定義する．

$$M_s(t) = \{x \in X \ ; \ \exists u \in \Sigma^* \ , \ M_e(u) = t \ , \ f(x_0, u) = x\} \tag{5.3}$$

M_s により，スーパバイザの観測事象列が t となる DES の状態集合が返される．すなわち，DES の現在の状態の候補を求め，これをスーパバイザの状態として定める．

スーパバイザは観測した DES の生起事象列から，状態 $s \in S$ として，現在の DES の状態の推定はできるが，正確な状態 $x \in X$ そのものは知ることができない．この制約のもとで，スーパバイザは DES に対して制御パターンの提示を行う．

また，DES はスーパバイザによって与えられた制御パターンの中から生起事象を選択することから，スーパバイザ側からの観測という立場で考えた場合，

$$\mathcal{P}(s, \pi, s') = \sum_{\sigma \in \pi \cap \Sigma^o} \mathcal{P}_1(s, \pi, s') \mathcal{P}_2(s, \sigma, s')$$

が成り立つ．ただし，

- $\mathcal{P}_1(s, \pi, \sigma)$ は，状態 $s \in S$ で，スーパバイザが制御パターン $\pi \in \Pi(s)$ を選択したとき，事象 $\sigma \in \pi$ が DES によって選択される確率，
- $\mathcal{P}_2(s, \sigma, s')$ は，状態 $s \in S$ で，事象 $\sigma \in F(s)$ が生起したとき，状態 $s' \in S$ に遷移する確率，

である．

前章と同様に，DES に対して以下の仮定を設ける．

[仮定 5.1]

1. 各状態 $s \in S$ と可観測事象 $\sigma^o \in \Sigma^o$ について，事象の選ばれやすさを表すパラメータとして，$\eta^*(s, \sigma^o)$ を導入する．このとき，

$$\mathcal{P}_1(s, \pi, \sigma^o) = \frac{\eta^*(s, \sigma)}{\sum_{\sigma^{o\prime} \in \pi \cap \Sigma^o} \eta^*(s, \sigma^{o\prime})} \tag{5.4}$$

$$\forall \sigma^o \in F(s) \ \ \eta^*(s, \sigma^o) > 0 \ , \ \sum_{\sigma^o \in F(s)} \eta^*(s, \sigma^o) = 1$$

の関係が成り立っているとする．すなわち，スーパバイザが選択した制御パターンの中からDESは生起事象を選択していくが，選択の結果，それぞれの事象を観測する比率は，与えられた制御パターンに依存せず一定である．ただし，η^* の真値をスーパバイザは知らない．

2. 報酬 $\mathcal{R}(s,\pi,s')$ について，

$$\mathcal{R}(s,\pi,s') = \mathcal{R}_1(s,\pi) + \mathcal{R}_2(s,\sigma^o,s') \tag{5.5}$$

という構造をもつとする．ここで，$\mathcal{R}_1$, $\mathcal{R}_2$ の意味は以下の通りである．

- $\mathcal{R}_1(s,\pi)$：状態 s で制御パターン π を選んだことによる報酬の期待値．制御パターンの与え方に依存して決定される．直観的には制御パターンに含まれない事象を禁止したことに伴うコストを表している．
- $\mathcal{R}_2(s,\sigma^o,s')$：状態 s から，可観測事象 $\sigma^o \in \Sigma^o$ の生起を観測して，状態 s' に遷移したときに得る報酬の期待値．これらは，直観的には事象の生起に伴うコスト及び，タスクの出来不出来に伴うコストを表している．

このとき (5.1) 式は，前章 (4.4) 式と同様にして，

$$\begin{aligned}
Q^*(s,\pi) &= \sum_{s'\in S}\left(\sum_{\sigma^o\in\pi\cap\Sigma^o}\frac{\eta^*(s,\sigma^o)}{\sum_{\sigma^{o\prime}\in\pi\cap\Sigma^o}\eta^*(s,\sigma^{o\prime})}\mathcal{P}_2(s,\sigma^o,s')\right) \\
&\qquad \left[\mathcal{R}_1(s,\pi)+\mathcal{R}_2(s,\sigma^o,s')+\gamma\max_{\pi'\in\Pi(s')}Q^*(s',\pi')\right] \\
&= \mathcal{R}_1(s,\pi) + \sum_{\sigma\in\pi\cap\Sigma^o}\frac{\eta^*(s,\sigma^o)}{\sum_{\sigma^{o\prime}\in\pi\cap\Sigma^o}\eta^*(s,\sigma^{o\prime})}\sum_{s'\in S}\mathcal{P}_2(s,\sigma^o,s') \\
&\qquad \left[\mathcal{R}_2(s,\sigma^o,s')+\gamma\max_{\pi'\in\Pi(s')}Q^*(s',\pi')\right] \\
&= \mathcal{R}_1(s,\pi) + \sum_{\sigma^o\in\pi\cap\Sigma^o}\frac{\eta^*(s,\sigma^o)}{\sum_{\sigma^{o\prime}\in\pi}\eta^*(s,\sigma^{o\prime})}T^*(s,\sigma^o)
\end{aligned} \tag{5.6}$$

となる．ここで，$T^*(s,\sigma^o)$ は，状態 s で事象 σ^o の生起を観測し，以後は，各状態で最大の Q 値を持つ制御パターンを選択するときの期待収益を表しており，

$$T^*(s,\sigma^o) = \sum_{s'\in S}\mathcal{P}_2(s,\sigma^o,s')\left[\mathcal{R}_2(s,\sigma^o,s')+\gamma\max_{\pi'\in\Pi(s')}Q^*(s',\pi')\right] \tag{5.7}$$

で定義される．$T^*(s,\sigma^o)$ の中には状態 s で制御パターン π を選択した事による報酬は含まれていない．

5.3 スーパバイザ学習のアルゴリズム（部分観測の場合）

5.3.1 提案アルゴリズムの概要

本章で提案する部分観測のもとでの強化学習を用いた離散事象システムのスーパバイザ制御の概念図を図 5.2 に示す.

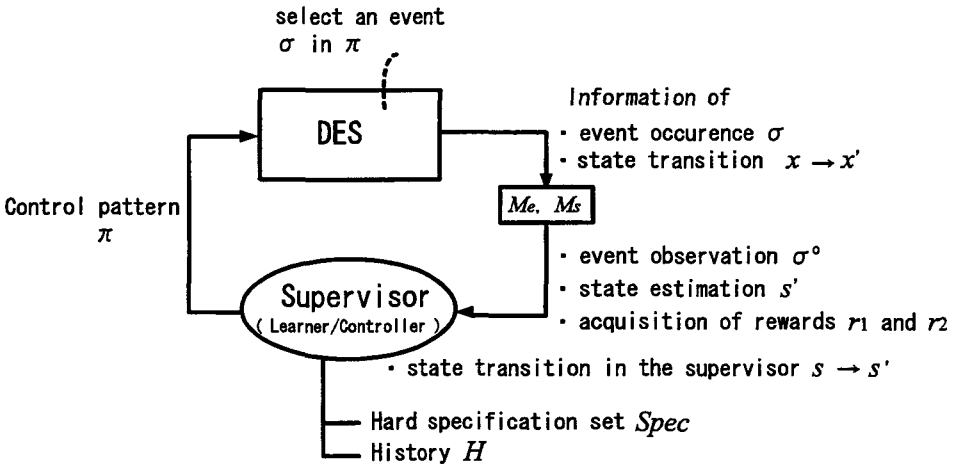


図 5.2: 学習スーパバイザによって制御される離散事象システム（部分観測の場合）

最適となる制御パターンの与え方をスーパバイザが学習することは前章と同じであるが，前章ではスーパバイザの状態と DES の状態は同じものとしてあつかっていたのに対し，本章では射影関数 M_e を通してしか DES を観測できない．そのため，制御パターンの変更は，可観測事象が DES 内部で生起したにときにしか行えない.

スーパバイザは DES に対し現在のスーパバイザの状態 s の Q 値をもとに制御パターン π を決定し，DES に示す．DES は π の中から生起事象 σ を選択する．σ が不可観測事象の場合，DES 内部では $x \xrightarrow{\sigma} x'$ の状態遷移が起きているが，スーパバイザ側からはこれを観測できない．可観測事象が生起するまで，DES 内部では複数回の状態遷移が起きる．可観測事象が生起した場合，そのときの生起事象 $\sigma^o \in \Sigma^o$ は射影関数 M_e を通してスーパバイザに観測される．そして，これまで観測してきた事象列から，状態推定関数 M_s により，DES の状態を推定する．また，2 種類の報酬 r_1, r_2 を得る．これらの情報から，スーパバイザは状態を $s \xrightarrow{\sigma^o} s'$ と遷移し，Q 値を更新する．このとき，仮定 5.1 を利用しての Q 値の同時更新を行う．終了状態に至

るまでこの手順を繰り返し，学習を進めていく．また，前章と同様に，最低限満足したい仕様 $Spec$ に対しては，強制的な枝刈りにより，これを満足させるようにする．

5.3.2 アルゴリズムの詳細

提案する学習アルゴリズムを図 5.3 に示す．以下，主に，4.3.2 節で示した完全観測の場合のアルゴリズムとの違いを中心に，図 5.3 に沿ってアルゴリズムの詳細を述べる．

1. 各 $T(s,\sigma^o)$, $R_1(s,\pi)$, $\eta(s,\sigma^o)$ に初期値を与える．ただし，$\sigma^o \in \Sigma^o$ である．
2. 次式で Q 値の初期値を計算する．

$$Q(s,\pi) \leftarrow R_1(s,\pi) + \sum_{\sigma^o \in \pi \cap \Sigma^o} \frac{\eta(s,\sigma^o)}{\sum_{\sigma^{o\prime} \in \pi \cap \Sigma^o} \eta(s,\sigma^{o\prime})} \tag{5.8}$$

3. ここで，各エピソードに対する学習を行う．
   (a) 履歴 H を空にする．また，観測事象列 t を ϵ に初期化する．
   (b) スーパバイザの初期状態として，DES の初期状態を s にセットする．
   (c) エピソード内の各ステップについて，s が終端状態に至るまで，以下の手順を繰り返す．DES の内部でどのような状態遷移が起きていても，スーパバイザの制御パターンの学習自体は，スーパバイザが観測した情報のみから行われる．
      i. スーパバイザは現在の状態 $s \in S$ において，制御パターン $\pi \in \Pi(s)$ を選択し，DES に提示する．
      ii. DES の内部で π の中から，事象の選択と状態の遷移が進んでいく．スーパバイザが DES での可観測事象 σ^o の生起を観測すると，観測事象列が $t \leftarrow t\sigma^o$ と更新され，更新された t を用いて，状態推定関数 $M_s(t)$ の出力 s' として，DES の現在の状態候補からなる，スーパバイザの状態 s' を得る．
      iii. 2 種類の報酬，r_1, r_2 を得る．
      iv. スーパバイザの内部状態を $s \xrightarrow{\sigma^o} s'$ に遷移させる．
      v. 状態 s で事象 σ^o の生起を観測したという履歴情報を H に追加する．
      vi. 以下の式で $T(s,\sigma^o)$, $\mathcal{R}_1(s,\pi)$, $\eta(s,\sigma^{o\prime})$ の更新を行う．

$$T(s,\sigma^o) \leftarrow T(s,\sigma^o) + \alpha[r_2 + \gamma \max_{\pi' \in \Pi(s')} Q(s',\pi') - T(s,\sigma^o)] \tag{5.9}$$

$$R_1(s,\pi) \leftarrow R_1(s,\pi) + \beta[r_1 - R_1(s,\pi)] \tag{5.10}$$

$$\text{For all} \quad \sigma^{o'} \in \pi$$

$$\eta(s,\sigma^{o'}) \leftarrow \begin{cases} (1-\delta)\eta(s,\sigma^{o'}) & (\text{if } \sigma^{o'} \neq \sigma^{o}) \\ \eta(s,\sigma^{o'}) + \delta\left[\sum_{\sigma^{o''}\in\pi\cap\Sigma^{o}} \eta(s,\sigma^{o''}) - \eta(s,\sigma^{o'})\right] & (\text{if } \sigma^{o'} = \sigma^{o}) \end{cases} \tag{5.11}$$

vii. Q 値の同時更新を行う．

$$\text{For all} \quad \pi' \in \Pi(s) \ s.t. \ \pi' \cap \pi \neq \emptyset$$

$$Q(s,\pi') \leftarrow R_1(s,\pi') + \sum_{\sigma^{o''}\in\pi'\cap\Sigma^{o}} \frac{\eta(s,\sigma^{o''})}{\sum_{\sigma^{o'''}\in\pi'\cap\Sigma^{o}} \eta(s,\sigma^{o'''})} T(s,\sigma^{o''}) \tag{5.12}$$

DES の内部でどのような状態遷移が起きていても，スーパバイザの制御パターンの学習自体は，スーパバイザが観測した情報のみから行われる．

viii. *Spec* が与えられている場合，各ステップの最後には，完全観測の場合と同様の枝刈りを行う．ただし，履歴 H には観測した情報しか入っていない．

A. 履歴 H から，最後に生起した可制御可観測事象 $\sigma^{c,o} \in \Sigma^{c} \cap \Sigma^{o}$ と，そのときの状態 w を特定する．

B. $\sigma^{c,o}$ を w における生起可能な事象の集合 $F(w)$ から削除する．

C. 状態 w における η の和が 1 となるように，各 η の値を更新し，Q 値の再計算を行う．

ix. 現在のスーパバイザの状態を s とする．

アルゴリズム 3(c) は，s が終端状態に至るまで繰り返される．そのため，DES 内部において終端状態にいたり，エピソードが終了したことはスーパバイザから観測事象として知ることができる必要がある．

1. Initialize $T(s,\sigma^o), R_1(s,\pi)$, and $\eta(s,\sigma^o)$ at each state in the supervisor.
2. Calculate the initial Q value at each state by

$$Q(s,\pi) \leftarrow R_1(s,\pi) + \sum_{\sigma^o \in \pi \cap \Sigma^o} \frac{\eta(s,\sigma^o)}{\sum_{\sigma^{o'} \in \pi \cap \Sigma^o} \eta(s,\sigma^{o'})} T(s,\sigma^o)$$

3. Repeat until s is *terminal state* (for each episode):
   (a) Clear history H and initialize strings $t \leftarrow \epsilon$.
   (b) $s \leftarrow$ *initial state in DES.*
   (c) Repeat (for each step of episode)
      i. Select a control pattern $\pi \in \Pi(s)$ based on the Q value by the supervisor.
      ii. Observe event occurrence $\sigma^o \in \Sigma^o$, update $t \leftarrow t\sigma^o$ and estimate new state $s'(= M_s(t))$.
      iii. Acquire rewards r_1 and r_2.
      iv. Make transition $s \xrightarrow{\sigma^o} s'$ in the supervisor.
      v. Add (s,σ^o) to history H.
      vi. Update $T(s,\sigma^o)$, $\mathcal{R}_1(s,\pi)$, and $\eta(s,\sigma^{o'})$:

$$T(s,\sigma^o) \leftarrow T(s,\sigma^o) + \alpha[r_2 + \gamma \max_{\pi' \in \Pi(s')} Q(s',\pi') - T(s,\sigma^o)]$$

$$R_1(s,\pi) \leftarrow R_1(s,\pi) + \beta[r_1 - R_1(s,\pi)]$$

For all $\sigma^{o'} \in \pi$

$$\eta(s,\sigma^{o'}) \leftarrow \begin{cases} (1-\delta)\eta(s,\sigma^{o'}) & (\text{if } \sigma^{o'} \neq \sigma^o) \\ \eta(s,\sigma^{o'}) + \delta \left[\sum_{\sigma^{o''} \in \pi \cap \Sigma^o} \eta(s,\sigma^{o''}) - \eta(s,\sigma^{o'}) \right] & (\text{if } \sigma^{o'} = \sigma^o) \end{cases}$$

      vii. Update the Q values

For all $\pi' \in \Pi(s)$ *s.t.* $\pi' \cap \pi \neq \emptyset$

$$Q(s,\pi') \leftarrow R_1(s,\pi') + \sum_{\sigma^{o''} \in \pi' \cap \Sigma^o} \frac{\eta(s,\sigma^{o''})}{\sum_{\sigma^{o'''} \in \pi' \cap \Sigma^o} \eta(s,\sigma^{o'''})} T(s,\sigma^{o''})$$

      viii. If $s' \notin Spec$
         A. Search the latest observable and controllable event $\sigma^{c,o} \in \Sigma^c \cap \Sigma^o$ and the corresponding state $w \in S$ from the history H.
         B. Remove $\sigma^{c,o}$ from the feasible event set $F(w)$.
         C. Normalize $\eta(w,\sigma^{o'})$ so as to satisfy $\sum_{\sigma^{o'} \in F(w)} \eta(w,\sigma^{o'}) = 1$ and update the Q values at the state w.
      ix. $s \leftarrow s'$

図 5.3: スーパバイザ獲得のアルゴリズム（部分観測の場合）

5.3.3 アルゴリズムに対する考察

完全観測の場合が，スーパバイザの状態と DES の状態を同一としていたのに対して，本章では，DES から観測されるのは，可観測な生起事象とこれに伴う報酬のみで，観測した生起事象列から DES の状態を推定する機構を別途用意している．そのために，スーパバイザは DES の状態遷移関数は知ることができると仮定している．このことは，スーパバイザが観測する状態について通常の強化学習の枠組みよりも強い仮定を置いていると言える．

また，完全観測の場合は，本章での学習アルゴリズムについて，$\Sigma^o = \Sigma$ の場合，すなわち，事象がすべて可観測で，そのため DES の状態を 1 つに特定できる場合に相当するので，本章のアルゴリズムは前章のものをより一般化したものであるといえる．

しかし，基本的な学習の式は前章と同様なので，Q 値や強化学習との関連についての議論は，観測される事象のみに基づいて学習しているという点に注意すれば 4.3.3 節とほぼ同じくできる．すなわち，提案アルゴリズムは，ある厳密な仕様 $Spec$ の下で，観測事象のみから考えての期待収益の点で最適となる制御パターンを学習していくことを通じ，制御仕様とコストを考慮した中で，生成言語を最大とするスーパバイザを獲得するアルゴリズムであるといえる．

注意点として，もしも不可観測事象による遷移で DES がこれ以上の遷移ができないデッドロックの状態に陥った場合，観測情報が得られないスーパバイザからは制御パターンの変更は行えず，システム全体が停止してしまう．そのため，観測できないデッドロックが生じるの恐れのある DES では，長期間事象の生起が観測されない場合は，これを 1 つの事象と考えてスーパバイザ側に通知するような工夫が必要となるであろう．

5.4 実験

提案アルゴリズムの有効性の検証として，4.4.2 で用いた猫とねずみの例題を，事象の生起のいくつかが部分観測の場合に変更して，計算機実験を行った．

基本的な問題設定は 4 章の場合と同じである．ただし，図 5.4 のように事象 $c7$ を

不可制御不可観測事象としている．報酬として，ドアを閉じるにはドア1つにつき平均-1，分散0.1の正規分布にしたがうコストがかかるとした．また，ひとつのエピソードは20回移動を終えた場合か制御仕様を満たさなくなった場合，すなわち，猫とねずみが同じ部屋になった場合（-100の報酬を与える）に終了するとした．学習率はいずれも0.1を用い，$\gamma = 0.9$ とした．制御パターンの選択には，$\epsilon = 0.1$ の ϵ-greedy 選択を用いた．

図5.5は，横軸にエピソードを，縦軸に500エピソードからなる学習を100回行ったときの，制御仕様（猫とねずみが同じ部屋にならない）を満たす最適なスーパバイザが求まった割合を示している．最終的には両手法とも最適なスーパバイザが求まっているが，完全観測の場合と同じく提案手法の方が Q-learning よりも速く，求まる確率が1に収束している．また，図5.6は，学習の結果得られた制御パターンを状態遷移図で表したものである．円内の数字がスーパバイザが推定したDESの状態となっており，10の位が猫のいる部屋に，1の位がねずみのいる部屋に対応する．遷移の矢印が，各状態での生起を許可する事象を示している．可観測事象の生起に伴い状態が遷移し，制御パターンが変化している．状態14と34における制御パターンが，不可観測事象の存在により，完全観測の場合から変化している．

部分観測環境において，猫とねずみが同じ部屋に居ないという論理的な制御仕様を満たす制御パターンを提示するスーパバイザが学習を通じて獲得されている．

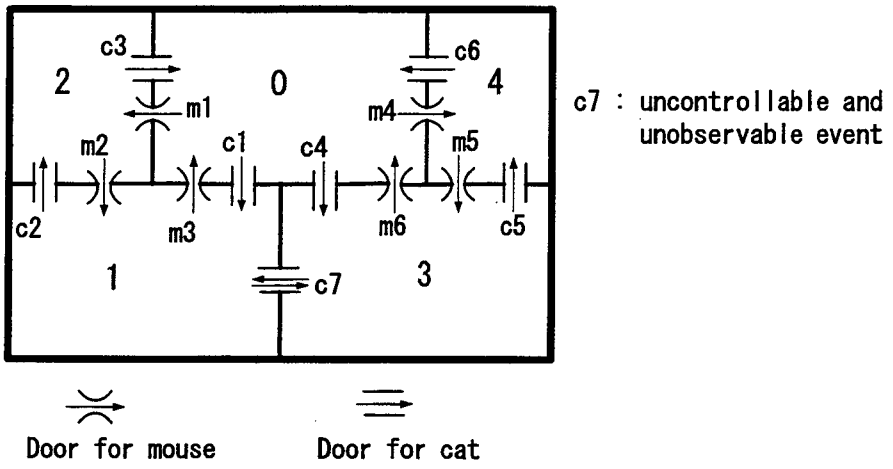


図5.4: 猫とねずみの迷路の問題

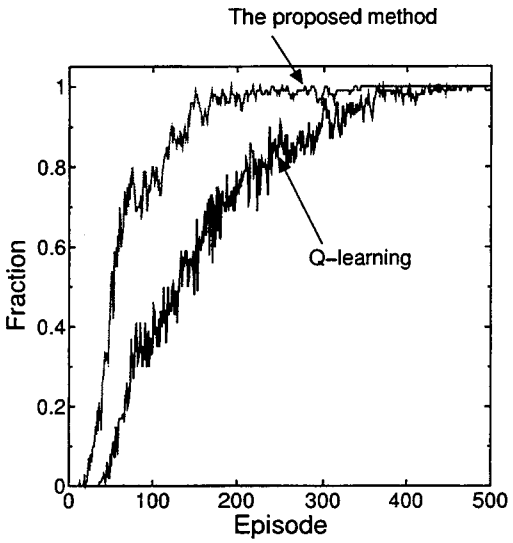


図 5.5: 猫とねずみの迷路の問題におけるエピソード数とスーパバイザが最適制御パターンを見つけた割合の関係（c7 が不可観測事象の場合）

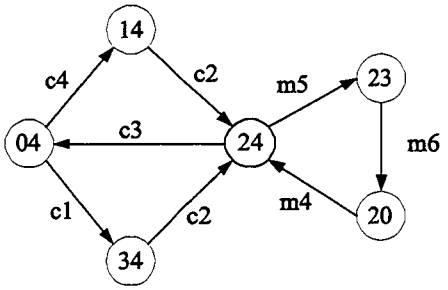


図 5.6: 獲得された制御パターンの状態遷移図（c7 が不可観測事象の場合）

5.5 結言

本章では，4 章で提案した強化学習に基づく最適スーパバイザ制御法を，DES 内部での生起事象が，スーパバイザから部分観測の場合に拡張した．スーパバイザは，DES の正確な状態を知ることができないため，観測した生起事象のみから DES の状態をスーパバイザが推定する機構を導入した．その上で，推定した状態と観測事象のみから，4 章と同様の手法で，スーパバイザに制御パターンの与え方を学習させるアルゴリズムを提案した．

部分観測環境を扱うとしたことで，4 章でのアルゴリズムよりも，より広い問題に対してスーパバイザ制御を適用できるようになった．しかし，状態の推定のための十

分な情報が常に与えられるとは限らない．そのため，DES の特性を考慮して，アルゴリズムの検討と，学習の効率化を図ることは重要な今後の課題である．また，より大きな問題を用いたアルゴリズムの有効性の検証を行うことも必要である．

第 6 章

結論

本論文では，報酬に基づく創発的制御手法として，計算生態学アプローチと強化学習に着目した．そして，

1. マルチエージェント系の資源配分問題に対しての，計算生態学アプローチに基づく制御手法の提案と，ネットワークの経路選択問題への応用
2. 強化学習を用いた離散事象システムにおける最適スーパバイザ制御法の提案．DES の状態がスーパバイザから完全観測の場合および，部分観測の場合への拡張

を行った．また，計算機実験による有効性の検討を行った．

以下では，本論文で得られた成果についてのまとめと，今後の課題を示す．

1. 論文前半では，報酬に基づく制御手法として，計算生態学アプローチに着目した．まず，2 章において Hogg-Huberman モデルを複数のエージェント集団が存在する場合に拡張したモデルを示した．計算生態学モデルは，エージェント集団のマクロ的挙動の解析のために導かれたモデルであるが，報酬メカニズムにより，各エージェントが資源から受け取る利益を等しくするように，各資源の選択割合を動的に調整する．そのため，Hogg-Huberman モデルを資源配分のための制御手法として積極的に利用することを主張し，基本的な定式化を示した．続く 3 章では，異なる目的地を持つパケット集団を，複数のエージェント集団ととらえることで，2 章の定式化をネットワークの経路選択問題にあてはめた．抽象化したパケットネットワークを考え，各エージェント集団の利益関数について複数の視点を反映した設計を行った．そして計算機実験により有効性の検証を

行った.

計算生態学アプローチの工学的利用は十分になされているとはいえず，どのような問題に適しているかを見極めての応用が望まれる．また，Hogg-Huberman モデルによるカオス抑制の詳細なメカニズムなど理論的に明らかにすべき点も多い．マルチエージェント系における，利己的なエージェント間の資源競合は，社会ジレンマのように，全体の利益を損なってしまう場合がある．このような状況を簡潔に表現する枠組みとしても，計算生態学アプローチは有望であり，ジレンマ解消のための制御手法として応用できるか検討を行う必要がある．

2. 論文の後半では，報酬に基づく制御手法として，強化学習に着目した．そして，従来，仕様と制御対象の正確な記述が必要なことが一つの問題であった，離散事象システムにおけるスーパバイザ制御問題に対して，強化学習を用いた制御パターンの学習法を提案した．論理的な仕様だけでなく，事象の生起／禁止のコストも報酬として考慮することで，最適なスーパバイザを獲得するアルゴリズムを示した．4 章では，DES の状態がスーパバイザから完全に観測できる場合を扱った．このとき，複数の Q 値を同時に学習できるように仮定を設けることで，学習速度の向上を図った．5 章では，生起事象の観測が部分的にしか行えない部分観測の場合に 4 章のアルゴリズム拡張した．観測事象列のみから，DES の状態をスーパバイザが推定する機構を導入した.

学習された制御パターンによる DES の生成言語の性質を理論的に明らかにすることは，強化学習と離散事象システム理論との関連を語る上で重要である．また，アルゴリズムそのものについての，性能向上のための検討も必要である．このとき，DES の特性を考え，無理のない適切な仮定を導入できるかが重要となる．学習を通じたスーパバイザの獲得は，従来より大規模な問題に対しても適用の可能性が広がると考えるが，結局は，時間的，空間的な計算量が問題となってくる．ニューラルネットなどを用いた Q 値の近似手法を利用することで，さらに広い状態空間も扱えるような拡張を行えるであろう．そうして，現実の大規模なシステムで提案手法の有効性の検討を行うことも今後の課題である．また，マルチエージェント系への提案手法の拡張も重要な課題である.

創発的手法を用いた制御手法は，現在の大規模化，複雑化するシステムに対する一つのアプローチとして今後も様々な研究がなされていくと考える．本研究が，これら

の研究の発展に寄与することを期待する.

参考文献

[1] 青木, 青山, 濃沼. 広帯域ISDNとATM技術. コロナ社, 1995.

[2] D. P. Bertsekas and R.Gallager（八星監訳）. データネットワーク. オーム社, 1990.

[3] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, 1996.

[4] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems.* Kluwer Academic Pub., 1999.

[5] S. H. Clearwater, M. Dixon R. Costanza, and B. Schroeder. Saving energy using market-based control. In S. H. Clearwater, editor, *Market-Based Control*, pp. 253–273. World Scientific, Singapore, 1996.

[6] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Math.*, Vol. 1, pp. 269–271, 1959.

[7] P. Gohari and W. M. Wonham. On the complexity of supervisory control desighn in the RW framework. *IEEE Trans. Syst., Man, Cybern. B*, Vol. 30, No. 5, pp. 643–652, 2000.

[8] T. Hogg and B. A. Huberman. Controlling chaos in distributed systems. *IEEE Trans. Syst., Man, Cybern.*, Vol. 21, No. 6, pp. 1325–1332, 1991.

[9] B. A. Huberman and T. Hogg. The behavior of computational ecologies. In B. A. Huberman, editor, *The Ecology of Computation*, pp. 77–115. Elservier Science, Amsterdam, 1988.

[10] 今森, 潮. バイアスを用いた離散時間 Hogg-Huberman 戦略. 信学論 (A), Vol. J80-A, No. 12, pp. 2097–2102, 1997.

[11] 井上, 都丸. 新・情報通信早わかり講座 1. 日経 BP 社, 1998.

[12] 石田, 片桐, 桑原. 分散人工知能. コロナ社, 1996.

[13] R. Kumar and V. K. Garg. Optimal supervisory control of discrete event dynamical systems. *SIAM J. Control Optim.*, Vol. 33, No. 2, pp. 419–439, 1995.

[14] J. F. Kurose and R. Simha. A microeconomic approach to optimal resource allocation in distributed computer systems. *IEEE Trans. Computer*, Vol. 38, No. 5, pp. 705–717, 1989.

[15] 桑原. 市場モデルに基づく制御とその応用. システム/制御/情報, Vol. 41, No. 9, pp. 359–364, 1997.

[16] K. Kuwabara, T. Ishida, Y. Nishibe, and T. Suda. An equilibratory market-based approach for distributed resource allocation and its applications to communication network control. In S. H. Clearwater, editor, *Market-Based Control A Paradigm For Distributed Resource Allocation*, pp. 53–73. World Scientific Pub., Singapore, 1996.

[17] H. Marchand, O. Boivineau, and S. Lafortune. Optimal control of discrete event systems under partial observation. In *Proc. of the 40the IEEE Conference on Decision and Control*, pp. 2335–2340, 2001.

[18] J. M. Mcquillan, I. Richer, and E. C. Rosen. The new routing algorithm for the ARPANET. *IEEE Trans. Commun.*, Vol. 28, No. 5, pp. 711–719, 1980.

[19] 永田. パケット交換入門. オーム社, 1990.

[20] 沼岡, 大沢, 長尾. マルチエージェントシステム. 共立出版, 1998.

[21] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of The IEEE*, Vol. 77, No. 1, pp. 81–98, 1989.

[22] R. Sengupta and S. Lafortune. An optimal control theory for discrete event systems. *SIAM J. Control Optim.*, Vol. 36, No. 2, pp. 488–541, 1998.

[23] 柴田, 田中, 奥原. 生産者の好み及び市場情報の不確かさと時間遅れを考慮した競合状況での市場選択問題. 信学論 (A), Vol. J85-A, No. 6, pp. 653–660, 2002.

[24] R. S. Sutton and A. G. Barto. *Reinforcement Learning.* MIT Press, 1998.

[25] 竹下, 村山, 荒井, 苅田. マスタリング TCP/IP 入門編第 2 版. オーム社, 1998.

[26] 戸出, 尾尻, 山本, 岡田, 池田. 要求通信品質別コスト導出関数を用いた経路選択法とその評価. 信学論 (B-I), Vol. J80-B-I, No. 1, pp. 1–11, 1997.

[27] 潮, 山崎. 計算生態学モデルの非線形現象解析と工学的応用. システム/制御/情報, Vol. 45, No. 7, pp. 382–390, 2001.

[28] T. Ushio and M. Motonaka. Controlling chaos in a manufacturing system with two machines and two part-types based on the Hogg-Huberman strategy. In *Proc. of Int. Conf. Control of Oscillations and Chaos*, pp. 215–218, 1997.

[29] 潮, 今森. 計算生態学モデルにおける社会ジレンマの解消. 信学技報 NLP98-3, pp. 17–22, 1998.

[30] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, and W. S. Stornetta. Spawn: A distributed computational economy. *IEEE Trans. Software Engineering*, Vol. 18, No. 2, pp. 103–117, 1992.

[31] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, Vol. 8, pp. 279–292, 1992.

[32] M. P. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, Vol. 1, pp. 1–23, 1993.

[33] W. M. Wonham and P. J. Ramadge. On the supremal controllable sublanguage of a given language. *SIAM J. Control Optim.*, Vol. 25, No. 3, pp. 637–659, 1987.

[34] 八槇, ウェルマン, 石田. 市場モデルに基づくアプリケーション QoS の制御. 信学論 (D-I), Vol. J81-D-I, No. 5, pp. 540–547, 1998.

[35] 山岡, 酒井. ゲーム理論によるパケット網ルーチング方式. 信学論 (B-I), Vol. J79-B-I, No. 3, pp. 73–79, 1996.

[36] 山崎, 潮. Hogg-Huberman 戦略に基づくパケット網ルーチング方式. 信学論 (B-I), Vol. J81-B-I, No. 3, pp. 135–142, 1998.

[37] 山崎, 潮. Hogg-Huberman 戦略による耐故障性を考慮したパケット網ルーチン

グ方式. 信学技報 CST98-31(1999-01), pp. 15–22, 1999.

[38] T. Yamasaki and T. Ushio. An application of a computational ecology model to a routing method in computer networks. *IEEE Trans. Syst., Man, Cybern. B*, Vol. 32, No. 1, pp. 99–106, 2002.

[39] M. Youssefmir and B. Huberman. Resource contention in multiagent systems. In *Proc. of ICMAS-95*, pp. 398–403, 1996.

[40] 児玉, 潮. 離散事象システム理論-来し方と現在. システム/制御/情報, Vol. 42, No. 8, pp. 415–420, 1998.

[41] 木村, 宮崎, 小林. 強化学習システムの設計指針. 計測と制御, Vol. 38, No. 10, pp. 618–623, 1999.

[42] 鈴木, 残間, 稲葉, 大熊. 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 -. 日本ロボット学会誌, Vol. 14, No. 7, pp. 1042–1052, 1996.

業績リスト

学術論文

1. 山﨑 達志, 潮 俊光, "Hogg-Huberman 戦略に基づくパケット網ルーチング方式, " 電子情報通信学会論文誌 (B-I), Vol.J81-B-I, No. 3, pp.135–142, 1998.
   (English translation: Tatsushi Yamasaki and Toshimitsu Ushio, "A Packet Routing Method Based on a Hogg-Huberman Strategy," Electronics and Communications in Japan, Part1: Communication, Scripta Technica, Inc., Vol.82, No. 5, pp.16–23, 1999.)
2. Tatsushi Yamasaki and Toshimitsu Ushio, "An Application of a Computational Ecology Model to a Routing Method in Computer Networks,"IEEE Transactions on Systems, Man, and Cybernetics Part B:Cybernetics, Vol.32, No.1, pp.99–106, 2002.
3. 山﨑 達志, 潮 俊光, "強化学習を用いた離散事象システムのスーパバイザ制御, " システム制御情報学会論文誌, Vol.47, No.3, 2003 (to appear).

国際会議

1. Toshimitsu Ushio and Tatsushi Yamasaki, "A packet routing method

based on a Hogg-Huberman strategy, " IFAC LSS'98, pp.786–791, 1998.

2. Tatsushi Yamasaki and Toshimitsu Ushio, "Routing Method Based on a Computational Ecology Model," IEEE SMC'99, pp.261–271, 1999.

著書（分担執筆）

1. Toshimitsu Ushio, Takehiro Imamori, and Tatsushi Yamasaki, "Controlling Chaos in Discrete-time Computational Ecosystems, " In G. Chen, ed., Controlling Chaos and Bifurcations in Engineering Systems, CRC Pres, Chapter 28, pp.625–644, 1999.

解説記事

1. 潮 俊光, 山﨑 達志, "計算生態学モデルの非線形現象解析と工学的応用, " システム/制御/情報, Vol.45, No.7, pp.382–390, 2001.

国内会議

1. 山﨑 達志, 潮 俊光, "Hogg-Huberman 戦略に基づくパケット網ルーチング方式, " 電子情報通信学会非線形問題研究会 (NLP97-85), pp.1–8, 1997.
2. 山﨑 達志, , 潮 俊光, "Hogg-Huberman 戦略による耐故障性を考慮したパケット網ルーチング方式, " 電子情報通信学会コンカレント工学研究会 (CST98-31), pp.15–22, 1999.
3. 山﨑 達志, 潮 俊光, "計算生態学アプローチによるネットワークルーチング, " 第 43 回システム制御情報学会研究発表講演会, pp.149–150, 1999.
4. 山﨑 達志, 潮 俊光, " 強化学習を用いた離散事象システムの最適スーパバイザ制御, " 第 45 回システム制御情報学会研究発表講演会, pp.3–4, 2001.

(2002 年度システム制御情報学会学会賞奨励賞受賞)

5. 山﨑 達志, 達志, 潮 俊光, “ 強化学習を用いた最適スーパバイザの構成, ” 電子情報通信学会コンカレント工学研究会 (CST2001-1), pp.1–6, 2001.

付録 A

スーパバイザ制御

離散事象システムにおけるスーパバイザは，FIFO 処理やデッドロックの回避など，論理的に与えられた制御仕様を満たすように制御対象を制御する．ここでの制御とは，どの事象の生起を禁止，許容するかということである．このとき，事象にはスーパバイザが生起を禁止できる可制御事象と，できない非可制御事象があると仮定する．例えば，タスクの開始要求や，自身の移動方向の決定は可制御な事象である．また，いつタスクが終了するかや割り込みの要求などは非可制御な事象の例といえる．

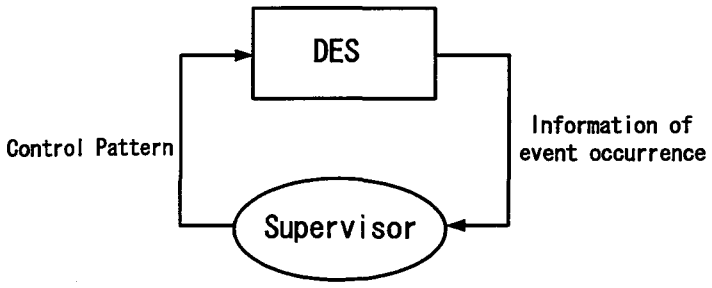


図 A.1: スーパバイザで制御される離散事象システム

制御対象である離散事象システムとスーパバイザの関係は，図 A.1 のようにとら

えることができる．スーパバイザは，制御パターンを選んで制御対象に伝える．このとき，非可制御事象は生起を禁止できないため，必ず制御パターンの中に含まれる．制御対象は，制御パターンの中から事象を生起させ次の状態へ移る．制御パターンの中からの生起事象の選択はスーパバイザ側から干渉はできない．スーパバイザは，制御対象で生起した事象を観測し，次の状態へ移る．以降，このサイクルが繰り返される．制御仕様を満たした上で，できる限り多くの事象の生起を許容するスーパバイザを構成する，多項式時間アルゴリズムが示されている [33]．また，このときシステムが生成する言語を最大可制御言語と呼ぶ．しかし，このアルゴリズムを適用するためには制御仕様を形式言語で正確に記述する必要がある．これには大きな計算量を必要とし，例えば複数個の仕様を同時に満足するスーパバイザを構成するために必要な計算量は $\mathcal{NP}$ 困難となる [7]．すなわち，制御仕様を求めるところに多くの労力が必要となる．

付録 B

強化学習

強化学習は，環境から受け取る報酬をもとに学習者が，より良い行動指針を見つけ出す学習手法である [3, 24, 41]．試行錯誤を通じた学習を行うため，環境に不確実さがある場合や，未知のパラメータがある場合にも，制御規則を自動的に獲得することができる．また，環境の変化に対しても自律的に対応することができる．

強化学習の概念は，図 B.1 のように表される．学習者であるエージェントは，環境に対し行動し，環境は，行動に対する報酬をエージェントに返す．また，行動の結果，環境の状態に変化が生じる．

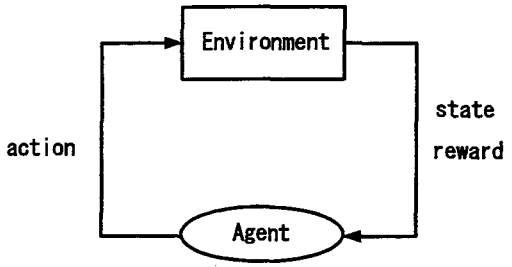


図 B.1: 強化学習の概念図

代表的な環境同定型の強化学習のアルゴリズムである，Q-learning[31] について述

べる．Q-learning では，Q 値と呼ばれる，状態と行動の組に対する評価値を，経験をもとに更新していくことにより学習を進める．学習者が状態 x で行動 a を選択し，状態 x' に遷移して，報酬 r を受け取ったとするとき，以下の式によって，Q 値を更新する．

$$Q(x,a) \leftarrow Q(x,a) + \alpha\left[r + \gamma \max_{a'} Q(x',a') - Q(x,a)\right] \tag{B.1}$$

ここで，$Q(x,a)$ は状態 x で行動 a を行ったときの期待収益（以後に獲得する割引報酬の総和の期待値）の推定値，α は学習率 $(0 < \alpha \leq 1)$，γ は報酬の割引率 $(0 \leq \gamma < 1)$ である．各状態で，期待収益が最大となるように，最大の Q 値を持つ行動を選択していくことが，その時点で最適な行動政策となる．

$\alpha_k(x,a)$ を，状態 x で行動 a が k 回目に選択されたときの学習率とする．マルコフ環境下では $\alpha_k(x,a)$ について，

$$\sum_{k=1}^{\infty} \alpha_k(x,a) = \infty \quad \text{かつ} \quad \sum_{k=1}^{\infty} \alpha_k(x,a)^2 < \infty \tag{B.2}$$

の条件を満たすとき，全ての行動に対して十分な回数の学習を行うことにより，$Q(x,a)$ は確率 1 で真値に収束する [31].