NAIST-IS-DD0661009

博士論文

モジュール構造を利用した ニューロコントローラの進化的構造探索法 および複数報酬強化学習法の提案

上岡　拓未

2009年3月17日

奈良先端科学技術大学院大学
情報科学研究科 情報生命科学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
博士 (工学) 授与の要件として提出した博士論文である。

上岡　拓未

審査委員：

石井　信 教授	（主指導教員）
銅谷　賢治 教授	（副指導教員）
杉本　謙二 教授	（副指導教員）
柴田　智広 准教授	（副指導教員）
吉本　潤一郎 准教授	（副指導教員）

モジュール構造を利用した
ニューロコントローラの進化的構造探索法
および複数報酬強化学習法の提案*

上岡　拓未

内容梗概

ロボットなどの自律システムが未知環境に対応するためには，機械学習や進化的計算といった何らかの最適化手法によって，環境に応じた制御器を獲得することが必要である．このような最適化手法を具体的問題へ適用する際は手法の選択だけでなく，制御器の構造，目的関数，メタパラメタなど様々な設計事項を決定する必要がある．多くの場合，このような設計は実験者の直感と試行錯誤によって行われる．ロボットの制御などの複雑な問題では特に制御器の構造と目的関数の設計は重要な問題となる．そこで，本研究では，(1) 制御器の構造の自律的な獲得手法と，(2) 目的関数の設定に対して頑健な学習手法を提案する．

制御器の構造を自律的に獲得する手法として，ニューラルネットワークの進化的探索法である NeuroEvolution (NE) と呼ばれる枠組みがある．これまでに提案されてきた多くの NeuroEvolution 手法は 1 つの遺伝子に対して 1 つの結合重みを割り当てていたため，結合単位の探索しか行うことができなかった．このため，大規模かつ複雑な構造を必要とする問題へ適用した場合に，進化の後期過程において探索効率が低下する問題があった．そこで，本研究ではモジュール単位での構造探索を可能にすることで，大規模なネットワークにおける探索効率を改善する．提案した遺伝的表現ではニューロン間の結合に実数値の重みとネットワークの部

*奈良先端科学技術大学院大学 情報科学研究科 情報生命科学専攻 博士論文，NAIST-IS-DD0661009, 2009 年 3 月 17 日.

分構造のどちらでも区別なく割り当てることが可能である．これにより，ネットワークをモジュールの組み合わせで表現でき，結合単位の探索とモジュール単位の探索が同列に扱える．シミュレーション実験により，提案手法がモジュール構造を利用した構造探索によって効率的に非マルコフ問題を解けることを示す．

また，目的関数の設定において，同時に複数の目的を考慮する必要があり，一意に目的関数を定めることが難しい場合がある．本研究では，特に強化学習を複数の目的からなる多目的最適化問題に適用することを考える．強化学習を多目的最適化問題へ適用する際には，目的ごとの報酬の足し合わせによるスカラー値の報酬を定義してきた．しかし，足し合わせによる手法では各報酬値の組み合わせを考慮する必要があるため，報酬設計が困難になるという問題があった．そこで，本研究では報酬値の組み合わせにロバストな強化学習法 Max-Min Actor-Critic (MMAC) を提案する．MMAC は目的別に与えられた報酬関数ごとに価値関数を学習する同一構造のモジュールを持ち，各状態における最小の価値関数を最大化する方策である Max-Min 最適方策を得る．シミュレーション実験によって，Max-Min 最適方策は複数報酬の和を最大化する手法に比べて各報酬値の組み合わせの影響を受けにくいことを示す．これにより，多目的最適化問題における報酬関数の設計問題を改善する．

キーワード

進化的計算，強化学習，モジュール構造，構造探索，報酬設計

Evolving Architecture of Neural Controller and Reinforcement Learning for Multiple Rewards based on Modular Structures*

Takumi KAMIOKA

Abstract

An optimization method such as machine learning and evolutionary computation is required to adopt autonomous systems to unknown environment. In order to apply optmization methods to real problems, there are many particulars of design such as a control architecture, an objective function and meta-parameters. These are usually designed by a trial and error approach. Especially, design of a control architecture and an objective function for complex problems becomes difficult. Here, we propose (1) an evolutionary acquisition method of a control architecture and (2) a robust learning method for a design of an objective function.

NeuroEvolution, which is a framework for optimizing architectures as well as connection weights of artificial neural networks, has been studied for an autonomous acquisition of control architecture. A conventional approach directly encodes each connection weight by one gene. However, such methods could not effectively explore a structure of a large network, because the unit of exploration is only one connection. In this study, we propose a novel method which allows to explore by the unit of a sub-structure obtained through evolution. The genetic

*Doctoral Dissertation, Department of Bioinformatics and Genomics, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD0661009, March 17, 2009.

representation of our proposed method enables to define a connection between two neurons either a real valued weight or a subnetwork. Therefore, a network can be represented by a combination of modules, and can be explored by the unit of both one connection and a sub-structure. In a computer simulation, we show that the proposed method effectively solve the non-Markovian task, and produce a wide variety of network architectures by exploration based on modular structures.

For design of an objective function, We have to consider a variety of aspects, and therefore it is a natural to formulate the task as a multi-objective problem. We consider to apply reinforcement learning (RL) to multi-objective problems. When RL is applied to a multi-objective problem, a common way is to scalarize multiple rewards by linear summation. However, RL based on linear scalarization causes sensitive for rewards design, because the structure of the problem which is defined by multiple rewards is ignored. We propose a modular reinforcement learning method for multi-objective problems, called Max-Min Actor-Critic (MMAC) algorithm, which updates a separate state value function for each objective and obtains a Max-Min optimal policy. Computer simulation shows that the Max-Min optimal policy is robust with respect to the scaling of reward functions. Thus MMAC can simplify reward designs in multi-objective problems.

Keywords:

evolutionary computation, reinforcement learning, modular structure, structural search, reward design

目 次

図 目 次

表 目 次

第1章　はじめに

ロボットなどの自律システムが未知環境に対応するためには，機械学習や進化的計算といった何らかの最適化手法によって，環境に応じた制御器を獲得することが必要である．最適化手法は，一般に与えられた目的関数を最大化するパラメタを探索する手法であると定義できる．このような最適化手法を具体的問題へ適用する際は少なくとも

1. アルゴリズムの選択
2. メタパラメタの設定
3. 制御器の構造の決定
4. 目的関数の設計

を行う必要がある．これまでの研究では，(2)〜(4) の設計事項については実験者の直感と試行錯誤によって行われ，(1) アルゴリズムの最適性などが主題となることが多い．実用上では設計事項の設計のしやすさも含めた，よりメタな視点での最適化が必要である．

メタパラメタはアルゴリズムの探索の進め方を調節するパラメタであり，学習係数や温度パラメタなどがある．メタパラメタの設定は有限次元の実数値最適化であるため，進化的計算を容易に適用できる場合が多い [1]．制御器の構造最適化は3層フィードフォワードニューラルネットワーク (feedforward neural networks; FNNs) などに制限して，最適化の対象を中間層のニューロン数に帰着することで，通常の進化的計算が適用可能である．しかし，システムがマルコフ決定過程に従わない場合など，FNNs によって実現することが困難な問題が多く存在する．非マルコフ課題は制御器がダイナミクスを保存することで解決することができる．

適切にダイナミクスを保存するためには事前に構造を限定しないネットワークの構造探索が必要となる．

実問題を解くためには目標とする問題を適切に表現する目的関数を設計する必要がある．ロボットの制御などの複雑な問題では同時に複数の目的を考慮する必要があり，一意に目的関数を定めることが難しい．このような場合の目的関数は各目的別に設計した関数を統合することによって決められることが一般的である．目的関数が複数の関数の足し合わせで表される場合，設計者は各関数値のバランスを考慮する必要があり，その調整は試行錯誤によるところが多く，一般には困難な問題となる．そこで，各目的における関数値の変化に頑健な最適化手法が望まれる．

1. NeuroEvolutionによる制御器の構造獲得

近年，制御器をニューラルネットワークで構築し，進化的計算によって最適化するNeuroEvolution(NE)と呼ばれる方法が盛んに研究されている[2–4]．NEは目的関数やニューロンモデルへの制約を与えることなく，結合重みとネットワーク構造を同時に最適化できることが大きな特徴である．

進化的計算における目的関数は適応度関数fで与えられる．適応度は表現型Pの関数であり，表現型は遺伝型Gで表される．進化的計算による探索は遺伝型の空間で進められ，遺伝型の変化ΔGは表現型ΔPの変化を介して適応度の変化Δfを決める．よって，進化的計算による探索が有効に働くためには遺伝型の変化ΔGに対する適応度の変化Δfの絶対値が十分に大きいことが重要である．このため遺伝型-表現型マッピングや遺伝的符号化，遺伝的表現などと呼ばれる遺伝型と表現型の関係は進化的計算における重要な課題の一つである．

NEの場合の表現型Pはニューラルネットワークであるが，これまでに提案されている多くのNEでは1つの遺伝子に対して1つの結合重みを割り当てる直接的符号化手法と呼ばれる遺伝的表現が多く用いられてきた．直接的符号化手法における1世代の表現型の変化ΔPは1結合ごとの変化に相当する．ここでPが大規模かつ複雑なネットワークである場合，1結合あたりの変化ΔPは相対的に非

常に小さく，適応度の変化 Δf は 0 に近くなる．このため，大規模なネットワークを獲得するには直接的符号化手法では難しいと言われている [5]．このような進化の後期過程で探索効率が低下する問題を解消するために，それまで獲得した部分構造をモジュールとして再利用することが考えられる．遺伝子型の空間でモジュールを利用することで，小さな ΔG に対しても表現型の空間での大きな変化 ΔP を引き起こし，結果として探索効率の低下を回避できる．

また，生物が対称性を持つように，ロボットなど多くの動的システムは同一のモジュールの組み合わせによって表される．このようなシステムの制御器の探索にはモジュール構造の探索とそのモジュールの再利用を可能とした遺伝的表現は有用であると期待できる．ここで，モジュール構造の探索には直接的符号化手法と同じように 1 結合単位での探索が必要になるが，モジュールの再利用を可能にするには同時にモジュール単位での変化が必要である．このような遺伝的表現を実現するには結合とモジュールを同一に扱えるような新しい枠組みが必要である．

2. 強化学習と報酬設計に対する頑健性

制御器を環境との相互作用からリアルタイムで最適化する枠組みとして強化学習 [6] がある．強化学習を具体的な問題に適用するには報酬関数を定義する必要がある．通常の強化学習では報酬はスカラー関数で与えられるが，実世界の問題では複数の目的や制約を満たすことが求められる．例えば，移動ロボットの制御では

1. エネルギー切れを起こさない．
2. 障害物にぶつからない．
3. 目的地へ移動する．

といった複数の目的を考えるのは自然である．この場合各目的別に，個別の報酬を

1. r_1:残りエネルギー量に比例した報酬を与える．
2. r_2:障害物と接触したときに負の報酬を与える．

3. r_3:目的地でゼロ，その他で負の報酬を与える．

のように設計することは容易である．目的地へ到達するためにエネルギーを補充する必要があるときや目的地までに障害物があるときは，(1),(2) の制約を満足しながら，本来の目的である (3) を行わなければならず，これらの目的間のトレードオフを考慮した多目的強化学習の枠組が必要となる．このような多目的最適化問題への強化学習の適用は，設計者が報酬関数をうまく調整することによって実現されてきた．例えば，複数の報酬関数の重み付き和によってスカラーの報酬関数を定義し，通常の強化学習を適用するといった方法が採用されることが多い．これは報酬関数のスカラー化と呼ばれる．報酬値のスカラー化による方法では目的間のトレードオフを報酬値のスケーリングによって解決している．このため，実問題への応用の際に報酬の設計が困難になるという問題が生じている．よって，報酬値の大きさに依らない目的間のトレードオフの解決法が必要となる．

3. 本研究の目的と論文の構成

本研究では進化的計算や機械学習の実用上の問題となっている制御器の構造と目的関数の設定を解決する新しい進化的計算手法及び機械学習手法を提案する．

2 章では大規模なネットワークの構造探索に適用可能な新しい NE 手法を示す．これまでの知見からモジュール構造とその再利用に着目し，モジュール内部にモジュール構造を持つことを可能にしたニューラルネットワークの遺伝型表現手法を提案する．提案手法の有効性を検証するためにベンチマーク課題として広く用いられている速度情報を用いない倒立振子安定課題に適用し，従来手法と比較する．シミュレーションの結果，提案手法によってモジュールを利用した構造探索を実現し，モジュールをもとにした探索を行わない場合と比べて，より早く解を見つけることができることを示す．

3 章では報酬設定に頑健な強化学習法である Max-Min Actor-Critic (MMAC) を示す．MMAC は各報酬に対する価値関数を学習する複数の Actor モジュールと方策を学習する 1 つの Critic モジュールで構成される．Max-Min 基準は最小の価値関数を最大化する方策を獲得する手法であり，多目的最適化問題の解法の中

でも「最も平等的」と言われている．この基準のもとでは各報酬は共通の上限値を持つという制約が必要であるが，これは容易に実現できる．Max-Min 法では加重線形和法などの線形変換と異なり，報酬値の大きさが異なっても得られる方策はほとんど変わらないため，設計者は目的別に報酬を設計することができる．さらに，報酬値のバランスを調整するパラメタも必要ないため，報酬関数を設計しやすいと考えられる．MMAC は目的ごとに与えられた報酬関数から各目的に対する価値関数を学習し，各状態における最小の価値関数を最大化する方策である Max-Min 最適方策を得る．各価値関数の学習器は同一構造のネットワークで実現される．シミュレーション実験によって，Max-Min 最適方策はその他の複数報酬に対する強化学習法に比べて各報酬値の組み合わせの影響を受けにくいことを示す．

最後に 4 章で本研究をまとめる．

第2章　階層的モジュール表現による NeuroEvolution

1. NeuroEvolution

近年，制御器をニューラルネットワークで構築し，進化的計算によって最適化する NeuroEvolution (NE) と呼ばれる方法が盛んに研究されている [2–4]．NE は他の学習法と異なり，目的関数やニューロンモデルへ制約を与えることなく，結合重みとネットワーク構造を最適化できることが大きな特徴である．

進化的計算において，遺伝型と表現型の関係である遺伝的表現は非常に重要な課題である．従来の NE 研究ではニューラルネットワークの結合重みなどの特徴を一つ一つ遺伝子に符号化する直接的符号化が行われてきた．直接的符号化によって多くの課題が解けることが示されているが，ゲノム長がネットワークサイズに比例すること，1 回の遺伝子操作では 1 つの結合重みのみしか変化させることできないことからネットワークの規模が大きくなるに従い，構造探索の効率が悪化することが指摘されている [5]．これに対して，複数の遺伝子の組み合わせで表現型上の特徴が定義される間接的符号化手法が提案されている [5, 7–12]．間接的符号化手法の大きな特徴は，1) ネットワークにモジュール性や規則性が表れやすいこと，2) 遺伝子の変化は複数の表現型上の特徴に影響することである．

進化の過程において適切な解を見つけるためには，遺伝的表現が高い ‘進化可能性 (Evolvability)’ を持つ必要がある．進化可能性とは，本来は生物の進化過程における環境へ適応する能力を意味するが，進化的計算においては適切に解を見つけることのできる可能性を意味する (2 節参照)．近年の研究によって，上記 2 つの間接的符号化の特徴は進化可能性の観点からも優位であることが示されている [5].

表現型の一部をモジュールとして扱い，モジュールを再利用することが進化可能性を高めることは生物学的，理論的双方から示されている．また，モジュール構造を明示的に取り入れた場合とそうでない場合を比較すると，多脚ロボットの歩行制御 [13] や人工生命シミュレーション [14] などでモジュール構造を取り入れることの重要性が示されている．

モジュールの定義とその再利用を明示的に探索する NE 手法として，Cellular Encoding with Automatic Definition of neural Subnetworks [15]，ModularNEAT [16]，embedded Cartesian Genetic Programming [17, 18] などが提案されている．しかし，これらの手法はモジュールの利用において強い制限があり，小さなモジュールの組み合わせによって階層的にモジュールを表現することは不可能である．そこで，本研究ではノード間の結合に重みとモジュールを区別なく割り当てることで，サブモジュールによるモジュール表現を実現した NE 手法を提案する．

2. 進化可能性

生物は進化の過程において，非常に広範な環境変動に頑健に適応しながら，多様化し，複雑化してきた．このような生物システムの進化の特徴を進化可能性 (evolvability) という．Krishner と Gerhard は進化可能性を「遺伝型上の変化によって，表現型上では多様な形質を生み出し，かつ致死的な変化にはなりにくいことである」と定義している [19]．このような特徴は生物の遺伝システムにおいて，(1) ゲノムが変化しにくい部分と柔軟に変化しやすい部分が存在すること (conservation and deconstraint)，(2) モジュール構造を持つことによって変異の影響が全体へ伝わらないこと (compartmentation and modularity)，(3) 遺伝子間が弱い接続によるネットワークを形成していること (weak linkage)，によって実現されている．Raff and Sly は発達機構におけるモジュール構造の重要性を指摘しており，モジュール自身もさらに小さなモジュールによって形成される階層的モジュール機構が重要であるとしている [20]．さらに，発達機構が階層的モジュールを形成するためには遺伝子発現システムが領域 (territory) を持つ必要があり，進化の過程において発達機構がモジュール化，分離，重複，分岐，再利用を実現

すると述べている．

進化計算法における進化可能性は進化の過程において，親個体よりも高い適応度を持つ個体が生まれる可能性があること [21]，適応度の低い突然変異個体が作られないこと [22] ，であるとされている．実際に進化可能性を高める決定的な方法は明らかになっていないが，

1. 遺伝的表現に多様性があり遺伝型と表現型が多対1の関係を持つこと [23, 24].
2. 局所解を避ける適切な遺伝子操作が定義されていること [21, 25].
3. 構造的なモジュールの複製及び再利用が可能であること [26].
4. 表現型上の適応度の変化に重要な部分では遺伝子操作による変化が微小であること [27, 28].
5. 進化の過程において遺伝子操作の方法を適応させることができること [23, 25].
6. 遺伝子操作に対して表現型が影響を受けない中立遺伝子が存在すること [22].

などの要素が重要であると言われている [22, 29].

NE においても進化可能性の議論は進められているが，未だ統一された測度や解析方法は示されていない．Schlessinger らは個体群における適応度の期待値をもとにした進化可能性の測度によって，遺伝子操作における進化可能性の変化を示した [29]．これにより，構造変化を生じさせる突然変異操作において，微小かつ適応的な操作が進化可能性を高めることを示している．適応的な遺伝子操作とは対象となる遺伝子によって変異の結果が異なる遺伝子操作である．Schlessinger らはニューロンを追加する際の結合重みを追加したニューロンと既存のニューロンとの距離によって決定する遺伝子操作を提案した．提案された進化可能性の測度は人工生命の実験環境である Mosaic World [30] に対して定義されており，一般的な測度としては扱えない．

Reisinger らは潜在的符号化 (3.1 節参照) が多次元入力からなる課題において進化可能性が高いことを示した [5]．進化可能性の比較は突然変異による性能の劣化

具合によって比較される．潜在的符号化は直接的符号化手法に比べ，突然変異による表現型の変化が大きいにも関わらず，性能劣化が小さいことを示した．各符号化手法で得られた最適解の表現型解析において，潜在的符号化によって得られたネットワークは，1) スケールフリーネットワーク [31] ではないこと，2) ネットワークモチーフ解析 [32] により，自然界によく見られるような多くのリカレント結合を持つことを示した．潜在的符号化がこのような進化可能性を持つ理由として，1) 遺伝子操作による影響が表現型上の複数の特徴に影響すること，2) 遺伝子の複製と分岐によってモジュールが形成されること，を挙げている．

3. 関連研究

3.1 符号化手法の分類

NE 手法は符号化の特徴によって直接的符号化と間接的符号化に大別される．直接的符号化手法は遺伝型上の各遺伝子と表現型上の特徴に 1 対 1 の関係がある手法である．例えば，結合重み行列を遺伝型として保持する方法 [33–36] や結合している 2 つのノードとその結合重みを符号化した NEAT [37] や EANT [38]，EANT2 [39] などが直接的符号化手法に分類される．これに対して，間接的符号化手法は遺伝型と表現型に多対 1 もしくは 1 対多の関係がある手法である．2 節に示したように，進化可能性の観点からは遺伝型と表現型が多対 1 の対応がある間接的符号化手法が有利であると言える．

Floreano らによる分類では，間接的符号化手法はさらに発達符号化手法 (developmental encoding) と潜在的符号化手法 (implicit encoding) に分けられる [4]．発達符号化手法は遺伝型にニューラルネットワークそのものではなく，ネットワークの発達過程を符号化する手法である．Stanley and Miikkulainen は発達規則の記述方法によって文法的アプローチと分子化学的アプローチに分類している [40]．文法的アプローチは発達規則を記述する文法を定義し，構文の組み合わせを遺伝型に符号化する手法である．代表的な手法としては Kitano の提案した手法 [41] や後述する Cellular Encoding (CE)[42] やその発展形である Edge Encoding (EE)[43] などがある．

軸索の伸張やシナプス結合の形成など生物の神経回路網の発達は遺伝子の発現を起因とする化学反応の結果である．分子科学的アプローチはこのような化学反応をモデル化した手法である．Nolfi らは軸索の成長方法を遺伝型に符号化した手法によって，ロボットの制御器の構造と結合重みを進化的に獲得できることを示した [7, 44]．Nolfi らの手法では 2 次元空間上に配置されたニューロンの発火に伴ってそのニューロンの軸索が成長し，軸索が他のニューロンに接触した時ニューロン間の結合が形成される．遺伝型は各ニューロンの位置，軸索の成長する距離と角度及び結合重みを符号化し，ニューラルネットワークの構造と結合重みの進化を可能にしている．

潜在的符号化手法は生物の遺伝子発現モデルをもとに考えられた手法である．生物の遺伝子は，発現によって作られるタンパク質の情報を記録する構造遺伝子と発現を制御する調節遺伝子とに大別される．遺伝子の発現は調節遺伝子と転写因子と呼ばれる特定のタンパク質との化学反応によって起こる．転写因子自身は他の遺伝子の発現によって作られることから，遺伝子の発現はタンパク質を介した遺伝子のネットワークとして表され，遺伝子発現調節ネットワーク (Gene Regulatory Networks: GRN) と呼ばれる．Mattiussi らはアルファベット文字列を用いた GRN モデルである Analog Genetic Encoding (AGE) を提案している [11, 45, 46]．AGE ではあらかじめ決められたトークンを用いて区切られた構造遺伝子領域と調節遺伝子領域からなる遺伝子が一つのノードを表し，構造遺伝子領域とその他の遺伝子の調節領域の類似度によってノードの結合を表現する．Dürr らは AGE をニューラルネットワークに適用し，遺伝子間の類似度によって結合重みを表す手法を提案した [10]．Reisinger and Miikkulainen は文字列の代わりに実数値ベクトルを用いた手法を提案し，特に大規模ネットワークの進化において直接的符号化手法よりも有効であることを示した [5]．Eggenberger は GRN モデルを用いて軸索の発達を制御する手法を提案し [47]，実ロボットの制御に適用している [48]．また，GRN モデルによる手法では制御器だけでなく，制御対象の形態進化にも適用した研究もある [49–56]．これらの研究では構造遺伝子の発現がネットワークや制御形態の発達を決定しており，発達符号化にも分類される．

間接的符号化手法の中で発達符号化にも潜在的符号化にも属さない手法も存在

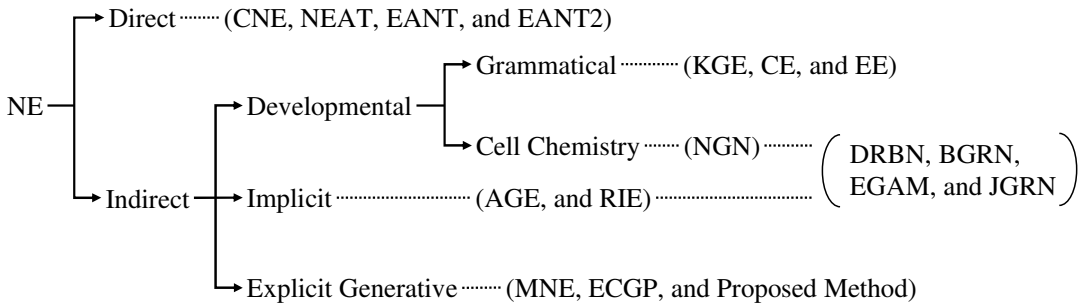


図 2.1 NE 手法の系統樹．括弧の中は代表的な手法を示す．それぞれ CNE: Conventional NeuroEvolution[33–36], NEAT[37],EANT[38]，EANT2[39], KGE: Kitano's Grammatical Encoding[41], CE: Cellular Encoding[42], EE: Edge Encoding[43], NGN: Nolfi's Growing Network[7, 44], AGE: Analog Genetic Encoding [10, 46], RIE: Reisinger's Implicit Encoding[5], DRBN: Dellaert's Random Boolean Network model [49, 50], BGRN: Bongard's GRN model[51–53], EGAM: Eggenberger's Growing Axon Model [47]，JGRN: Jin's GRN model[54–56], MNE: ModularNEAT[16], ECGP: embedded Cartesian Genetic Programming[17, 18], Proposed Method: 提案手法，を示す．

する．Reisinger らの提案した ModularNEAT はネットワークの発達過程や GRN モデルを符号化していないが，モジュール部分を符号化する領域とそのモジュールの利用方法を符号化する部分からなり，同一モジュールを複数の部分で再利用することができる [16]．モジュールを表す遺伝型の変化は表現型上で再利用された部分全てに影響するため，遺伝型と表現型は 1 対多の関係を持ち，間接的符号化手法である．Hornby は進化計算による自動設計手法の分類において，このような手法を明示的生成符号化手法 (explicit generative representation) と呼んでいる [57]．提案手法はこの明示的生成符号化手法に分類される．NE 手法以外の進化的計算手法では Walker and Miller の提案した Genetic Programming (GP) の拡張である embedded Cartesian Genetic Programming (ECGP)[17, 18] もこれに含まれる．

1

3.2 NeuroEvolution of Augmenting Topologies

Stanley and Miikkulainen が提案した NeuroEvolution of Augmenting Topologies (NEAT) は最も代表的な NE 手法の 1 つである [37, 58]．NEAT の遺伝型は node 遺伝子と connection 遺伝子の二つの遺伝子により構成される．node 遺伝子はノード番号と各ノードが入力，出力，隠れニューロンのいずれに属するかの情報を記録する．connection 遺伝子は接続する入力ノードと出力ノードのインデックス (In, Out)，実数値の重み (Weight)，発現するか否かのフラグ (Enable / Disable)，遺伝子に固有の番号 (Innovation) の 5 つの情報を符号化する．表現型のニューラルネットワークは node 遺伝子に書かれたノードを Enable である connection 遺伝子に従って接続することによって得られる．

NEAT は突然変異によって connection 遺伝子を追加することで新しい構造を探索する．このため個体によってゲノムの長さが異なり，通常の交叉を行うことができない．そこで NEAT では Innovation によって交叉位置の調節を行う．図 2.2 に示すように交叉時には Innovation に同じ番号を持つ遺伝子間で交叉を行う．選ばれた 2 つの親個体両方に同じ遺伝子がない場合はランダムに Enable / Disable を割り当てることで確率的にに 2 つの親の形質を受け継ぐことができる．Innovation 番号は集団全体で番号を保存し，突然変異によって新しい connection 遺伝子を加えるたびに 1 づつ追加していく．

NEAT では新しくできた構造を自然淘汰から保護するとともに，大きく異なる個体間での交叉を避けるために “種分化” を行う．個体間の類似度をもとに個体間の距離を算出し，近い個体によって種を形成する．2 個体間の類似度は

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \bar{W}$$

によって与えられる．ここで，$\bar{W}$ は 2 個体両方に存在する ‘一致遺伝子’ における重みの平均値，N は 2 個体で大きい遺伝子の数，E は非一致遺伝子のうち Innovation がもう一方の個体の持つ最大の Innovation よりも高い ‘excess 遺伝子’ の数，D は非一致遺伝子のうち excess 遺伝子ではない ‘disjoint 遺伝子’ の数 (図 2.2 参照)，c_1, c_2, c_3 は各項の依存度を決めるパラメタである．ある 2 個体間の類似度 δ が閾値 δ_t を超えたとき，その 2 個体は同じ種であるとされる．これにより，集団上にいく

つかの種のグループが形成され，交叉は主に同一種の個体間で行われる (異なる種の個体間での交叉は低い確率で行われる)．図 2.2 の例では $N = 10, E = 2, D = 3$ であり，$\bar{W} = 0.5, c_1 = c_2 = c_3 = 1$ とすると，$\delta = 1.0$ と計算される．

突然変異によって新しい構造を獲得した個体は，重みが適切でないために適応度が低い場合が多い．NEAT ではこのような革新的な個体を保護するため，explicit fitness sharing [59] を適用している．explicit fitness sharing では適応度を

$$f'_i = \frac{f_i}{\sum_{j=1}^{n} \mathrm{sh}(\delta(i,j))}$$

によって補正する．ただし，f_i は個体 i の適応度，n は集団サイズ，sh(.) は $\delta(i,j)$ が閾値 δ_t を超えたとき 1，それ以外では 0 の関数である．すなわち右辺の分母は同一種に属する個体の数であり，新規に生まれた種は相対的に高い適応度となり保護される．

NEAT では必ず全ての個体を入力から出力への接続のみで構成されるような最も単純な構造で初期化する．単純な構造から構造探索によって徐々に複雑化させることで，効率的に与えられた問題の解となる最小のネットワークを得ることができると説明している．実際にランダムな初期集団と比較して，単純な初期集団を用いた方が効率的であることが実験によって示されている [37].

NEAT は倒立振子 [37] をはじめ，移動ロボットの制御 [60]，コンピュータゲーム [61, 62]，模様デザインの進化 [63] など非常に幅広い分野に応用されている．しかし，少なくとも接続と同じ数の遺伝子長が必要なため，大規模なネットワークの進化に適用する場合は，スケール問題が生じることが指摘されている [5].

3.3 ModularNEAT

Reisinger らによって提案された ModularNEAT [16] はモジュール構造を複数の部分で利用できる NE 手法である．ModularNEAT は NEAT の拡張として提案された．ModularNEAT の概要を図 2.3 に示す．ModularNEAT は Blueprint 集団と Module 集団の 2 つの集団によって構成され，Module 集団は通常の NEAT と同様にニューラルネットワークを進化させる．Blueprint 集団，Module 集団共に NEAT と同じく種分化を行う．Blueprint 集団の各個体は入出力ノードと Module

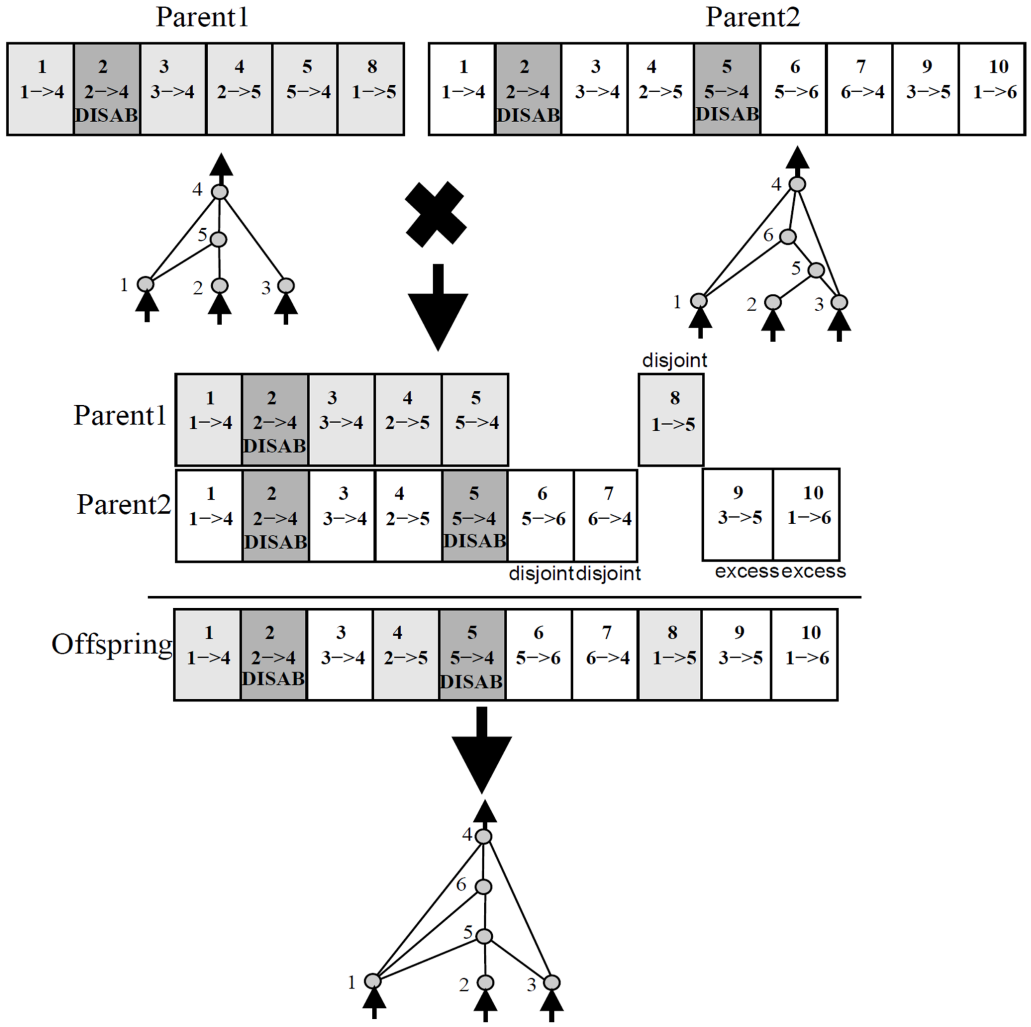


図 2.2 NEAT における交叉 ([37] より引用)

集団における種の情報を記録し，入出力ノードとの Module のペアを符号化する．表現型のニューラルネットワークは Blueprint に書かれた入力ノードと出力ノードを対応する Module 集団の種からランダムに取り出した個体を接続することで復号化される．

Blueprint 集団と Module 集団は次の共進化プロセスに従う．

- Module 集団の初期化: 通常の NEAT と同じく最小の構造によって初期化する．
- Blueprint 集団の初期化: ランダムに初期化する．
- 終了条件を満たすまで繰り返し：
  - Module の選択と交叉: バイナリトーナメント選択と NEAT と同じ交叉を行う．
  - Blueprint の選択と交叉: トーナメント選択と 1 点交叉を行う．
  - Module の突然変異: 突然変異率に従って，重みに乱数を加える，新しいニューロンを加える，新しい接続を加える操作を行う．
  - Blueprint の突然変異: 突然変異率に従って，入出力ノードと Module をランダムに変化させる．
  - 適応度の評価: Blueprint 集団を復号化して適応度を評価する．Module の適応度は種ごとに使われた Blueprint 個体の適応度の和を使われた回数で割った値で評価する．

Reisinger らは ModularNEAT をボードゲームに適用し，NEAT に比べて進化の速度が速く，盤面のサイズの変更に頑健であることを示した．ModularNEAT はモジュール構造の再利用を実現した NE 手法として，提案手法と同じアプローチを取った手法である．ただし，ModularNEAT で再利用可能なモジュールはセンサ/モータ間のみの非常に限られた範囲である．提案手法では階層的モジュール表現を用いることで，各モジュールの表現においてもより小さなモジュールの組み合わせで表すことを実現している．

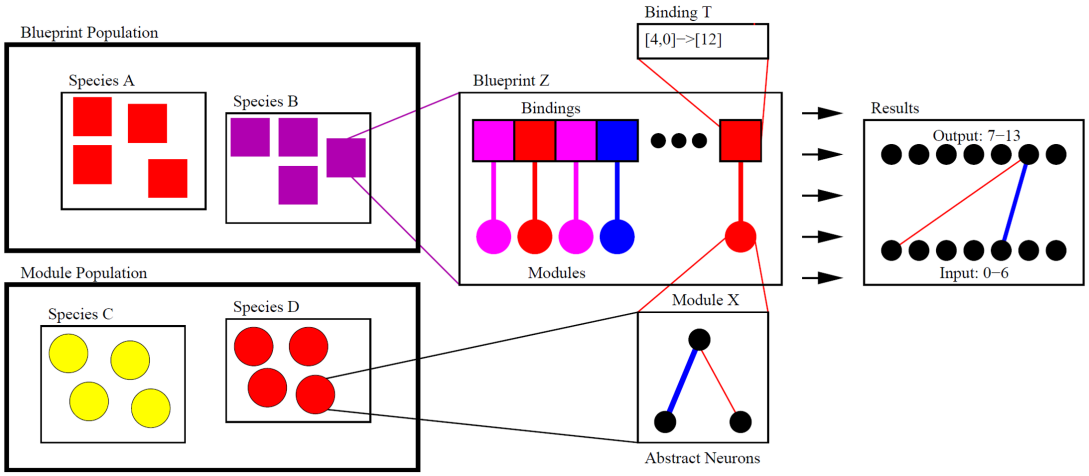


図 2.3　ModularNEAT の概要 ([16] より引用)．左：遺伝型，中-右：復号化プロセス

3.4 Cellular Encoding

Gruau の提案した Cellular Encoding (CE)[64, 65] は発達符号化手法に分類される．CE の遺伝型は構文木で定義される．遺伝型の各ノードがニューロンの分裂や結合の変化などの発達規則を表す．図 2.4 に CE の復号化の例を示す．図 2.4 の左に遺伝型となる構文木を，右に表現型となる ANN の発達過程を示す．ネットワークの発達は右図 1 のように一つのニューロンから始まり，構文木の根から順に一つずつ進められる．“Split” ノードでニューロンが追加され，“Increment” ノードで各ニューロンのパラメタが変化する．CE の遺伝型は木構造であるため，Genetic Programming (GP) [66] を容易に適用可能である．これにより，繰り返し構造や対称構造を持つネットワークを獲得しやすいという利点がある．Gruau らは倒立振子課題において，CE と直接的符号化手法の比較を行い，CE がより複雑な課題を解くことができることを示した [42].

また，Gruau は複数の構文木を用意することで表現型の複数の部分に同一モジュールを発達させる方法を提案した [15]．これは GP における Automatically Defined Functions (ADFs)[67] と呼ばれる方法で，遺伝型のノードに他の木への遷移するというルールを加えたものである．遺伝型に 3 本の構文木を用意するこ

とで，同一モジュールの必要な 6 脚ロボットの歩行制御器の進化的獲得を可能にした．しかし，この手法では表現型で発現する同一のモジュールの数は遺伝型内の構文木の数によって制限される．遺伝型内の構文木の数は固定であるため，柔軟なモジュール構造の探索は不可能である．提案手法では同一モジュールを制約なく再利用でき，柔軟なモジュール構造を使った探索を実現している．

3.5 embedded Cartesian Genetic Programming

Koza によって提案された Genetic Programming (GP)[66] は木構造を進化計算によって最適化する手法であるが，Miller および Miller and Thomson は GP を有向グラフへと拡張した Cartesian Genetic Programming (CGP) を提案している [68, 69]．CGP の表現型は n_i 入力 n_o 出力の ‘プログラム’ であり，それは n_f 種類の n_n 入力 1 出力の関数の組み合わせで表される．プログラムは n_r 行 n_c 列の行列に配置された関数をノードとする有向グラフで表される．遺伝型は固定長の整数ベクトルであり，各関数の結合を符号化する．遺伝型は長さ $n_n + 1$ の各ノードを表す遺伝子 $\{\underline{f_i}, C_{i0}, \ldots, C_{in_n-1}\}$ に分けられる．$0 \leq f_i < n_f$ が関数のインデックスを表し，C_{ik} が関数 f_i への k 番目の入力ノードのインデックスを表す．各関数は l 列前までの関数の出力を入力に持つことができる．また，表現型上では結合されない中立遺伝子を持つことができ，中立進化説 [70] に基づいた進化を実現している．図 2.5 に 4 入力 4 出力 $n_r = 3, n_c = 3, l = 2, n_n = 2$ の CGP による論理回路の応用例を示す．図 2.5(a) に示す遺伝型における下線付きの整数が関数を示し，$\underline{0}$ が AND 関数，$\underline{2}$ が XOR 関数を示す．点線で囲まれた遺伝子は中立遺伝子であり，表現型上では接続されない．CGP は論理回路の自動設計を目的として開発された [71, 72] が，その後も生物の発達モデル [73–75] やロボットの制御 [76] などに応用されている．

また，Walker and Miller は CGP を関数の組み合わせをモジュールとして扱えるように拡張した embedded Cartesian Genetic Programming (ECGP) を提案している [17, 18]．ECGP は GP における module acquisition [77] を CGP 上で実現したものである．module acquisition は遺伝型の一部を圧縮 (compresse) と呼ばれる遺伝子操作によって一つのモジュールとして扱い，まとまった単位で再利用を

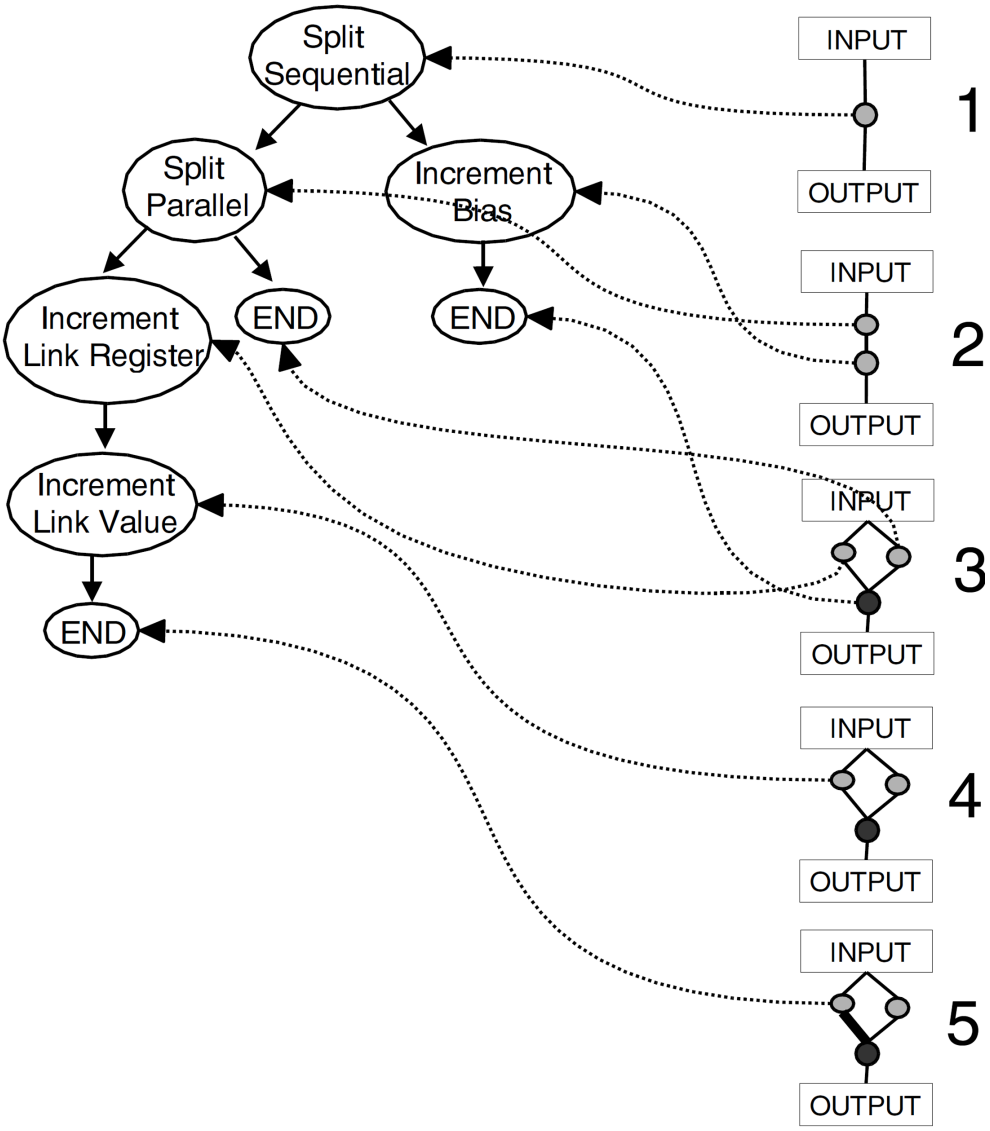


図 2.4 CE の例 ([40] より引用)

可能にした手法である．ECGP では多出力の関数に対応するため，整数の遺伝子 $\{\underline{f_i}, C_{i0}, \ldots, C_{in_n-1}\}$ から整数のペア $\{\underline{f_i : nt_i}, C^n_{i0} : C^o_{i0}, \ldots, C^n_{in_n-1} : C^o_{in_n-1}\}$ へと拡張された．関数 f_i への k 番目の入力は C^n_{ik} ノードの C^o_{ik} 番目の出力を受ける．node type: nt_i はノード i のプリミティブ関数，モジュール関数，モジュールの種類などを識別する番号である．ECGP では圧縮と展開と呼ばれる二つの遺伝子操作によって，それぞれモジュールの形成と解放を行う．圧縮操作ではランダムに選ばれた 2 点間の遺伝子をモジュール遺伝子として切り離し，元の遺伝型にはモジュールへの接続を表す遺伝子を新たに追加する．図 2.6 に圧縮操作の例を示す．図 2.6(a) は圧縮操作前の遺伝型を示す．遺伝子の中から 2 点 A,B がランダムに選ばれ，その部分が新しいモジュールとして抜き出される (図 2.6(b))．抜き出された遺伝子にはモジュールであることを示すヘッダー (module header: mh) が追加される．mh は 4 つの整数で定義され，それぞれモジュールを特定するインデックス，モジュールの入力ノード数，モジュール内の関数の数，モジュールの出力ノード数を符号化する．モジュール内の入力ノードは新たに 0 から順に割り振られ，関数のインデックスも入力ノードに続けて新たに割り振られる (図 2.6(c))．元の遺伝型にはモジュールを抜き出した部分にモジュールへの入力ノードを追加し，全てのノード及び関数のインデックスを更新する (図 2.6(d))．展開 (expand) は逆にモジュールを元の関数群へと戻す遺伝的操作である．

ECGP は GP および CGP に比べて，様々なベンチマーク課題において有効であることが示されている [18]．ECGP の NE への応用例は現在のところ発表されていないが，ECGP は提案手法と対応付けられる点がある．まず，遺伝型が有向グラフを表すことである．ECGP ではインデックス，提案手法ではポインタを使ってノード間の結合を表している．次に，ノードをまとめてモジュール単位の再利用を可能にしたことである．ECGP では圧縮，提案手法ではサブネットワークを追加する遺伝的操作によってモジュールを形成し, 再利用を可能にする．ただし，ECGP ではモジュールを使ってより大きなモジュールを表すことはできない．また，ECGP はまとめられたモジュールは完全に隔離されるため，圧縮されたモジュール内部の関数を単体で再利用することができない．提案手法では階層的なモジュール構造を実現し，下位のモジュールも単体で再利用可能であり，ECGP

よりも柔軟なモジュール構造の探索が可能である．

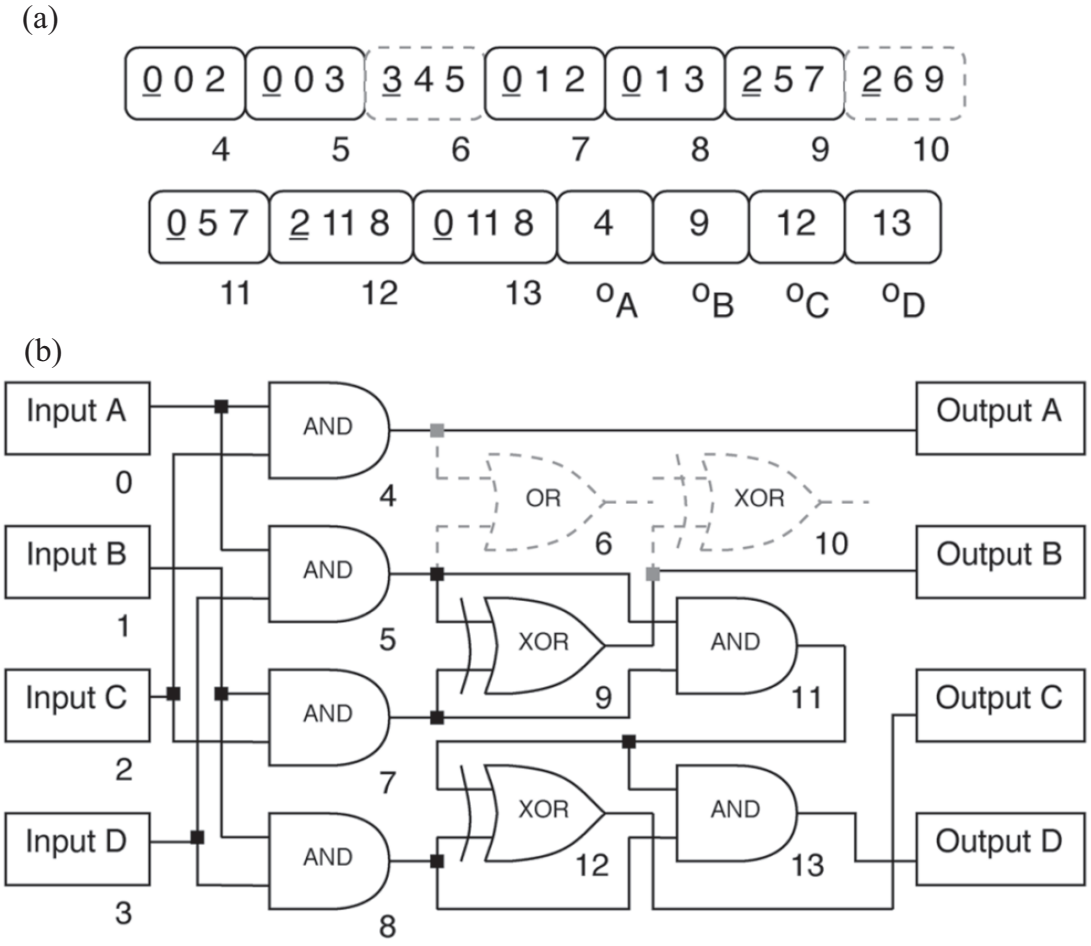


図 2.5　CGP の論理回路への応用例 ([18] より引用)．(a) 遺伝型，(b) 表現型．

4. 階層的モジュール表現

本手法における遺伝型は，サブネットワークと重みの二つの動的配列により構成される．重み配列の各要素は実数値を格納し，サブネットワーク配列の各要素はそれぞれサブネットワーク構造を符号化している．図 2.7 に示すように，サブネットワークは node と connection の二種類の遺伝子によって表される．node 遺

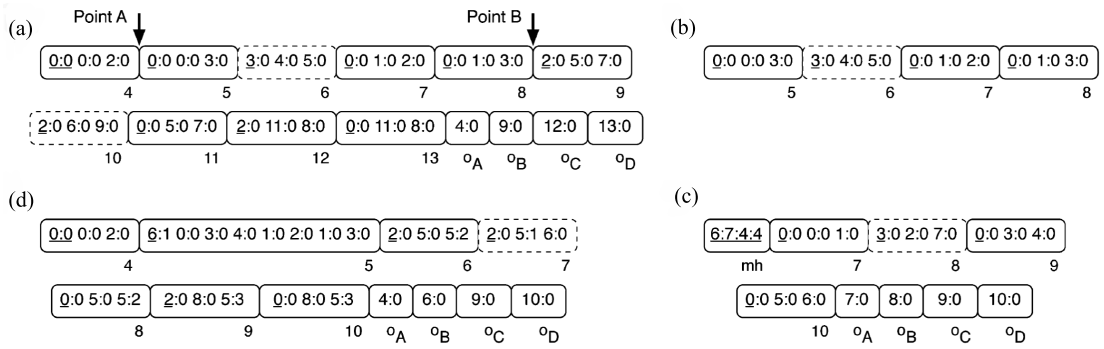


図 2.6 ECGP における圧縮操作 ([18] より引用). (a) 操作前の遺伝型からランダムに選ばれた 2 点 A,B 間を圧縮する. (b) モジュール化される関数郡. (c) モジュール化された遺伝子. (d) 圧縮操作後の遺伝型.

伝子はそのサブネットワーク内の入力，出力，隠れ層の各ノードの数を保持する．connection 遺伝子は入出力ノードと重みもしくは他のサブネットワークへのポインタを保持している．これにより，ノード間の接続を重みとサブネットワーク両方で表すことができる．また，subnetwork0 の入出力ノードはニューラルネットワーク全体における入出力ノードを定義する．

図 2.8 に示すように，遺伝型はサブネットワークと重みによる有向グラフで表される．有向グラフの根はサブネットワーク配列の最初の要素 subnetwork0 であり，その他のサブネットワークと重みが複数存在することは可能である．図の sn2 のように複数のリンクを受けるとき，そのサブネットワークが再利用されていることを示す.

4.1 復号化

各サブネットワークは次の手順により復号化される．

1. node 遺伝子に従って，入力ノード，出力ノード，隠れニューロンを配置する．
2. 入力ノード，出力ノード，隠れニューロンの順に各ノードにインデックスを割り当てる．

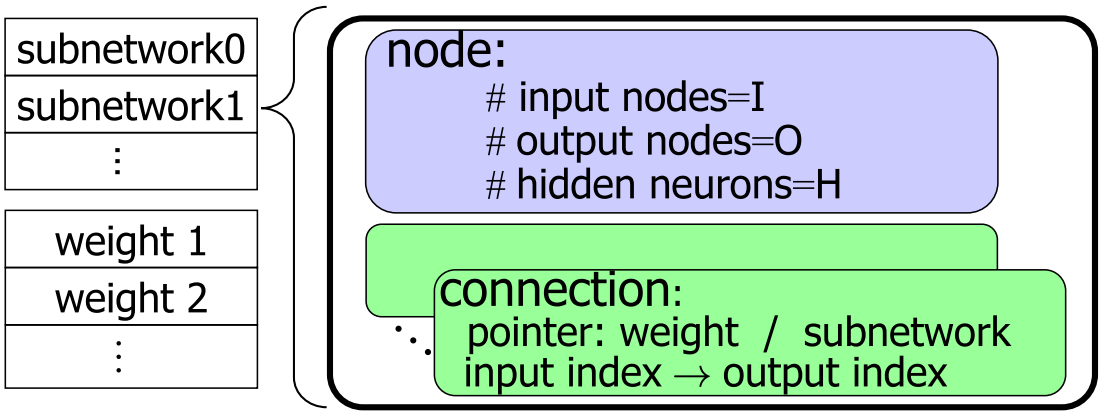


図 2.7 遺伝的表現

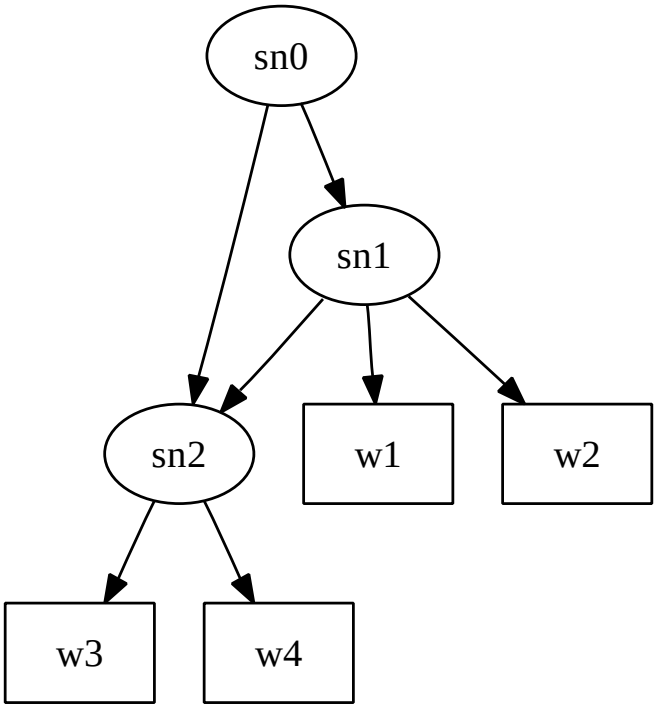


図 2.8 遺伝型の有向グラフ表現．表記を簡潔にするため，subnetwork0 を sn0，weight1 を w1 のようにしている．

3. connection 遺伝子に従って各ノードを接続する．

   (a) ポインタが重み配列を指している場合は，重み付き結合によって接続する．

   (b) ポインタが他のサブネットワークを指している場合は，対象となるサブネットワークを作り，connection 遺伝子の入出力ノードと作られたサブネットワークの入出力ノードを同一のノードとする．

遺伝型からニューラルネットワークへの復号化は subnetwork0 を復号化することで行われる．

図 2.9 に復号化の例を示す．図 2.9(a) は図 2.8 の有向グラフによって表される遺伝型を示す．図 2.9(b) は同じ遺伝型で定義される各サブネットワークを示す．各サブネットワークは図 2.9(c) に示すように connection 遺伝子に従って接続される．図中の点線は同一ノードであることを示す．図 2.9(d) にこれらをまとめた表現型を示す．

5. 進化的アルゴリズム

NE では competing convention 問題を考慮する必要がある．competing convention は全く異なる遺伝型が非常に近いネットワークを表す可能性を持つため，それらの交叉によってできるネットワークが元のネットワークと全く異なる問題である [34, 78]．

提案手法では competing convention 問題を避けるために，交叉を行わない突然変異のみによる構造探索を行う．EANT [38] や EANT2 [39] と同じく突然変異による構造探索と固定した構造での重みの学習による 2 段階の最適化を行う．

図 2.10 にアルゴリズムの概要を示す．集団数を M としたとき，i 番目の個体 $G_i, i = 1, \ldots, M$ はサブネットワークと重みのポインタを格納する動的配列 SN_i と重みベクトル $\boldsymbol{w}_i$ に分けられる．重みの最適化には Covariance Matrix Adaptation-Evolution Strategy (CMA-ES)[79](詳しくは 5.2 節参照のこと．) を適用した．個体 G_i において表現型に発現する重みベクトル $\boldsymbol{w_i}$ だけを取り出し，CMA-ES に

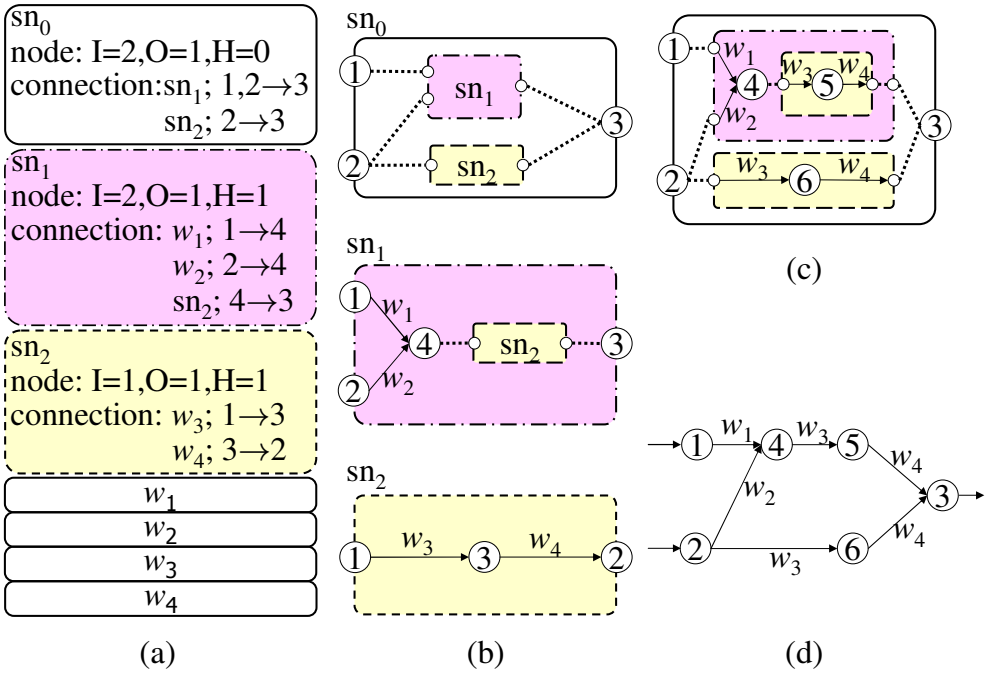


図 2.9 復号化の例. (a) 遺伝型, (b) 各サブネットワーク, (c) 接続されたサブネットワーク, (d) 表現型.

1

おける集団数 λ 個の複製を作りノイズを加える．λ 個の各重みベクトルに対して，適応度 $f_{ij}, j = 1, \ldots, \lambda$ を評価する．適応度 f_{ij} に従い，選択と次世代個体の生成を繰り返すことで重み $\boldsymbol{w}_i$ の探索が進められる．CMA-ES による解の探索が終了条件を満たしたとき，個体 G_i の適応度を $f_i = \max_j\{f_{ij}\}$ として評価する．適応度 f_i に基づいて次世代の個体を選択する．サブネットワーク $SN_i, i = 1, \ldots, M$ に突然変異を加える．以上を繰り返すことでニューラルネットワークの構造と重みの最適化を図る．

5.1 構造探索

提案手法では突然変異によって構造探索を行う．図 2.11 に突然変異によって行われる構造探索の例を示す．左列が遺伝型，中列が遺伝型によって表されるサブネットワークの結合，右列が対応する表現型のニューラルネットワークを示す．遺伝型における重み遺伝子は省略した．図 2.11(a) は突然変異を行う前のネットワークを示す．以下 6 種類の突然変異によって図 2.11 に示すような構造変化を加える．図 2.11(b)-(d) の 3 つの操作は新しい遺伝子を追加することで，各サブネットワークの構造探索を行う．

- 重み接続の追加 : サブネットワークの connection 遺伝子に新しく追加した重みによる接続を加える．図 2.11(b) では新しい重み w_5 を作り，sn0 内の connection 遺伝子に w_5 を参照する結合 $(w_5; 1 \rightarrow 4)$ が追加される．

- ニューロンの追加 : サブネットワークに隠れニューロンを追加する．図 2.11(c) では，sn_2 内の connection 遺伝子の隠れニューロン数を $\mathrm{H} = 2$ とし，新しく作られたノード 4 への結合 $(w_6; 3 \rightarrow 4$,$w_7; 4 \rightarrow 3)$ が connection 遺伝子に追加される．

- サブネットワークの追加 : 既存の 2 つの connection 遺伝子を持つ新しいサブネットワークを作り，元の connection 遺伝子と入れ替える．図 2.11(d) では w_5 と sn_1 を connection 遺伝子に持つ 1 入力 2 出力のサブネットワーク sn_3 が新しく追加され，sn_0 内の connection 遺伝子 $(\mathrm{sn}_1; 1 \rightarrow 3, w_5; 1 \rightarrow 4)$ が

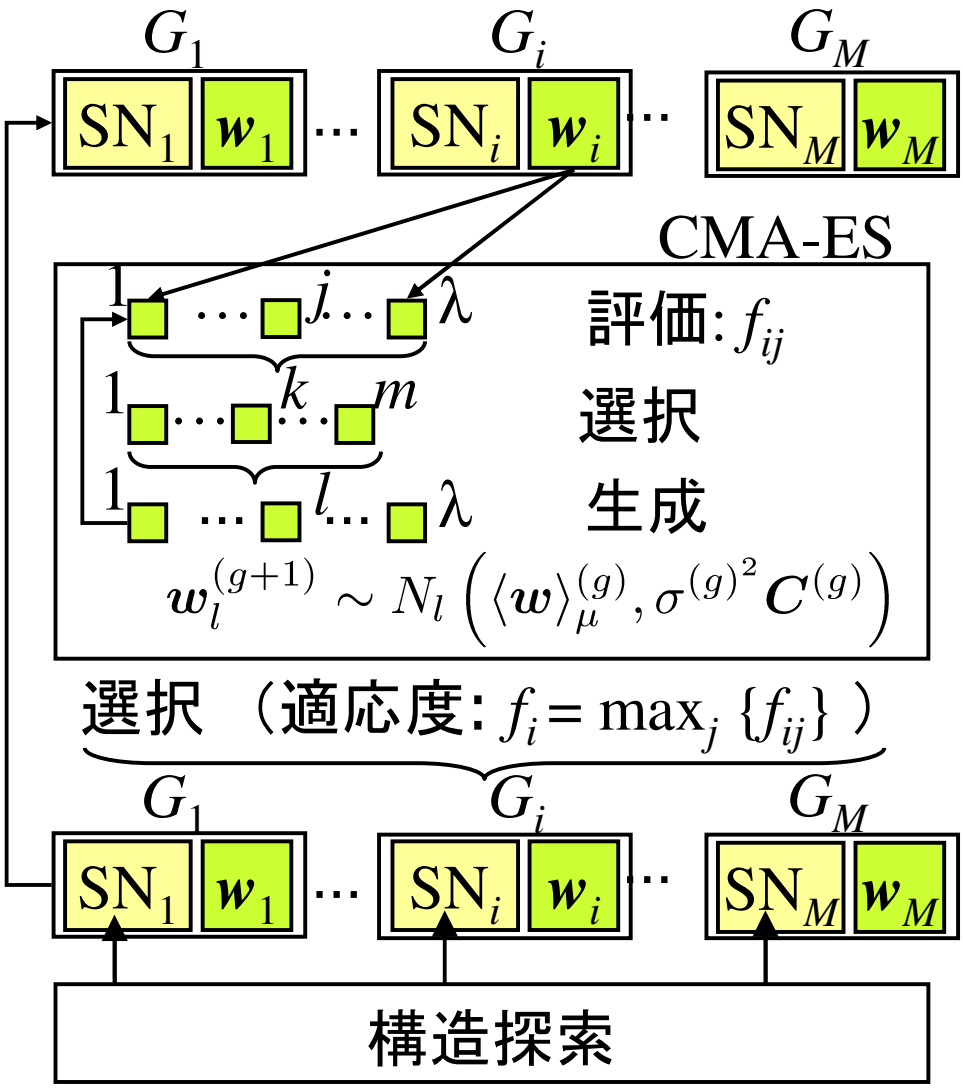


図 2.10 アルゴリズムの概要

$(\mathrm{sn}_3; 1 \rightarrow \{3, 4\})$ に置き換えられる．この操作は表現型を直接変化させることはないが，より大規模なサブネットワークの再利用が可能になる．

図 2.11(e)-(g) の 3 つの操作は既存のサブネットワークの再利用した構造探索を行う．

- 結合の削除 : ランダムに選ばれた connection 遺伝子を削除する．図 2.11(e) では sn_0 における $(\mathrm{sn}_2; 2 \rightarrow 4)$ の connection 遺伝子が削除される．ただし，サブネットワーク遺伝子 sn_2 は削除されず，中立遺伝子として保存される．

- 結合の追加 : ランダムに選択した入出力ノードと既存のサブネットワークによる connection 遺伝子を追加する．図 2.11(f) では sn_0 に connection 遺伝子 $(\mathrm{sn}_3; 2 \rightarrow \{4, 3\})$ が追加される．これにより表現型上には新しいモジュールが複製される．

- 結合の置換 : ランダムに選ばれた connection 遺伝子のポインタをその他のサブネットワークへのポインタへと置き換える．図 2.11(g) では sn_3 内の connection 遺伝子 $(\mathrm{sn}_1; 1 \rightarrow 2)$ の参照先が sn_2 へと置換される．ただし，置換操作はランダムに選ばれた connection 遺伝子の入出力ノード数とサブネットワークの入出力ノード数が一致した場合のみ行われる．

各突然変異アルゴリズムを以下に示す．ただし，$\textsc{RandInt}(n)$ は範囲 $[0, n)$ の乱整数を返す関数，sn, sn_1, sn_2 はランダムに選ばれたサブネットワーク遺伝子，$sn.I, sn.O, sn.H$ はサブネットワーク sn における入力ノード数，出力ノード数，隠れニューロン数，$weights$ は重み遺伝子の配列，connection は connection 遺伝子，$connections$ はそのリスト，$\textsc{SizeOf}(x)$ はリスト x の要素数を返す関数，$\textsc{Unique}(x, y)$ はリスト x, y から重複を削除したリストを返す関数，$\textsc{ConvertIndex}(x)$ はリスト x を新しいサブネットワーク内のノードに再割り当てをしたインデックスのリストを返す関数を示す．

Algorithm 5.1: AddWeightMutation($sn, weights$)

$in \leftarrow$ RandInt($sn.I + sn.O + sn.H$)
$out \leftarrow sn.I +$ RandInt($sn.O + sn.H$)
$w \leftarrow 0.0$
$C \leftarrow$ new connection $(w; in \rightarrow out)$
Add A Pointer Of C To $sn.connections$
Add A Pointer Of w To $weights$

Algorithm 5.2: AddNeuronMutation($sn, weights$)

$in \leftarrow$ RandInt($sn.I + sn.O + sn.H$)
$out \leftarrow sn.I +$ RandInt($sn.O + sn.H$)
$h \leftarrow sn.I + sn.O + sn.H$
$sn.H \leftarrow sn.H + 1$
$w1 \leftarrow 0.0$
$C1 \leftarrow$ new connection $(w1; in \rightarrow h)$
$w2 \leftarrow 0.0$
$C2 \leftarrow$ new connection $(w2; h \rightarrow out)$
Add A Pointer Of $C1$ To $sn.connections$
Add A Pointer Of $C2$ To $sn.connections$
Add A Pointer Of $w1$ To $weights$
Add A Pointer Of $w2$ To $weights$

Algorithm 5.3: AddSNMutation(sn)

$nc \leftarrow$ SizeOf($sn.connections$)
if $nc \geq 3$
then
- $c1 \leftarrow$ RandInt(nc)
- $c2 \leftarrow$ RandInt($nc - 1$)
- **if** $c1 \leq c2$
  - **then** $c2 \leftarrow c2 + 1$
- $C1 \leftarrow sn.connections[c1]$
- $C2 \leftarrow sn.connections[c2]$
- $inputs \leftarrow$ Unique($C1.inputs, C2.inputs$)
- $outputs \leftarrow$ Unique($C1.outputs, C2.outputs$)
- $NewSN \leftarrow$ new subnetwork
- $NewSN.I \leftarrow$ SizeOf($inputs$)
- $NewSN.O \leftarrow$ SizeOf($outputs$)
- $NewSN.H \leftarrow 0$
- $NewSNInputs1 \leftarrow$ ConvertIndex($C1.inputs$)
- $NewSNOutputs1 \leftarrow$ ConvertIndex($C1.outputs$)
- $NewC1 \leftarrow$ new connection
  - ($C1.pointer; NewSNInputs1 \rightarrow NewSNOutputs1$)
- $NewSNInputs2 \leftarrow$ ConvertIndex($C2.inputs$)
- $NewSNOutputs2 \leftarrow$ ConvertIndex($C2.outputs$)
- $NewC2 \leftarrow$ new connection
  - ($C2.pointer; NewSNInputs2 \rightarrow NewSNOutputs2$)
- Add A Pointer Of $C1$ To $NewSN.connections$
- Add A Pointer Of $C2$ To $NewSN.connections$
- $NewC3 \leftarrow$ new connection($NewSN; inputs \rightarrow outputs$)
- Add A Pointer Of $NewC3$ To $sn.connections$
- Delete $C1$ From $sn.connections$
- Delete $C2$ From $sn.connections$

Algorithm 5.4: DeleteConnectionMutation(sn)

$nc \leftarrow$ SizeOf($sn.connections$)
if $nc > 1$
then
- $c \leftarrow$ RandInt(nc)
- $C \leftarrow sn.connections[c]$
- Delete C From $sn.connections$

Algorithm 5.5: AddConnectionMutation(sn_1, sn_2)

$nin \leftarrow sn_2.I$
$nout \leftarrow sn_2.O$
for $i \leftarrow 0$ **to** nin
do
- $in \leftarrow$ RandInt($sn_1.I + sn_1.O + sn_1.H$)
- Add in To $inputs$

for $j \leftarrow 0$ **to** $nout$
do
- $out \leftarrow sn_1.I +$ RandInt($sn_1.O + sn_1.H$)
- Add out To $outputs$

$C \leftarrow$ new connection ($sn_2; inputs \rightarrow outputs$)
Add A Pointer Of C To $sn_1.connections$

Algorithm 5.6: ReplaceConnectionMutation(sn_1, sn_2)

$nc \leftarrow$ SizeOf($sn_1.connections$)
$ci \leftarrow$ RandInt(nc)
$C \leftarrow sn_1.connections[ci]$
$inputs \leftarrow C.inputs$
$outputs \leftarrow C.outputs$
$ncin \leftarrow$ SizeOf($inputs$)
$ncout \leftarrow$ SizeOf($outputs$)
$sn2in \leftarrow sn_2.I$
$sn2out \leftarrow sn_2.O$
if $ncin = sn2in$ **and** $ncout = sn2out$

then
- $Cnew \leftarrow$ new connection ($sn_2; inputs \rightarrow outputs$)
- Add A Pointer Of $Cnew$ To $sn_1.connections$
- Delete C From $sn_1.connections$

5.2 重みの学習 (CMA-ES)

提案手法では重みの学習に任意の手法を用いることができる．本論文においてはネットワーク構造に特に制約が必要なく，進化的計算による実数値最適化問題の解を探索する手法である Covariance Matrix Adaptation-Evolution Strategy (CMA-ES) [79] を採用した．Hansen and Ostermier が提案した CMA-ES は勾配法との関係も指摘されており [80]，高次元問題や多峰性の問題も効果的に解くことができることが示されている [81, 82]．また，Igel は固定構造のもとでのニューラルネットワークの重みの学習に効果的であることを示している [83].

CMA-ES では正規分布に従って解の探索を行う．世代 g における個体 k のネットワークの結合重みをベクトル $\boldsymbol{w}_k^{(g)} = (w_1, \ldots, w_n)^T$ で表すと，子孫 $\boldsymbol{w}_k^{(g+1)}, k = 1, \ldots, \lambda$ は

$$\boldsymbol{w}_k^{(g+1)} = \langle \boldsymbol{w} \rangle^{(g)} + \boldsymbol{z}_k, \quad \boldsymbol{z}_k \sim N_k^{(g)}(\boldsymbol{0}, {\sigma^{(g)}}^2 \boldsymbol{C}^{(g)}),$$

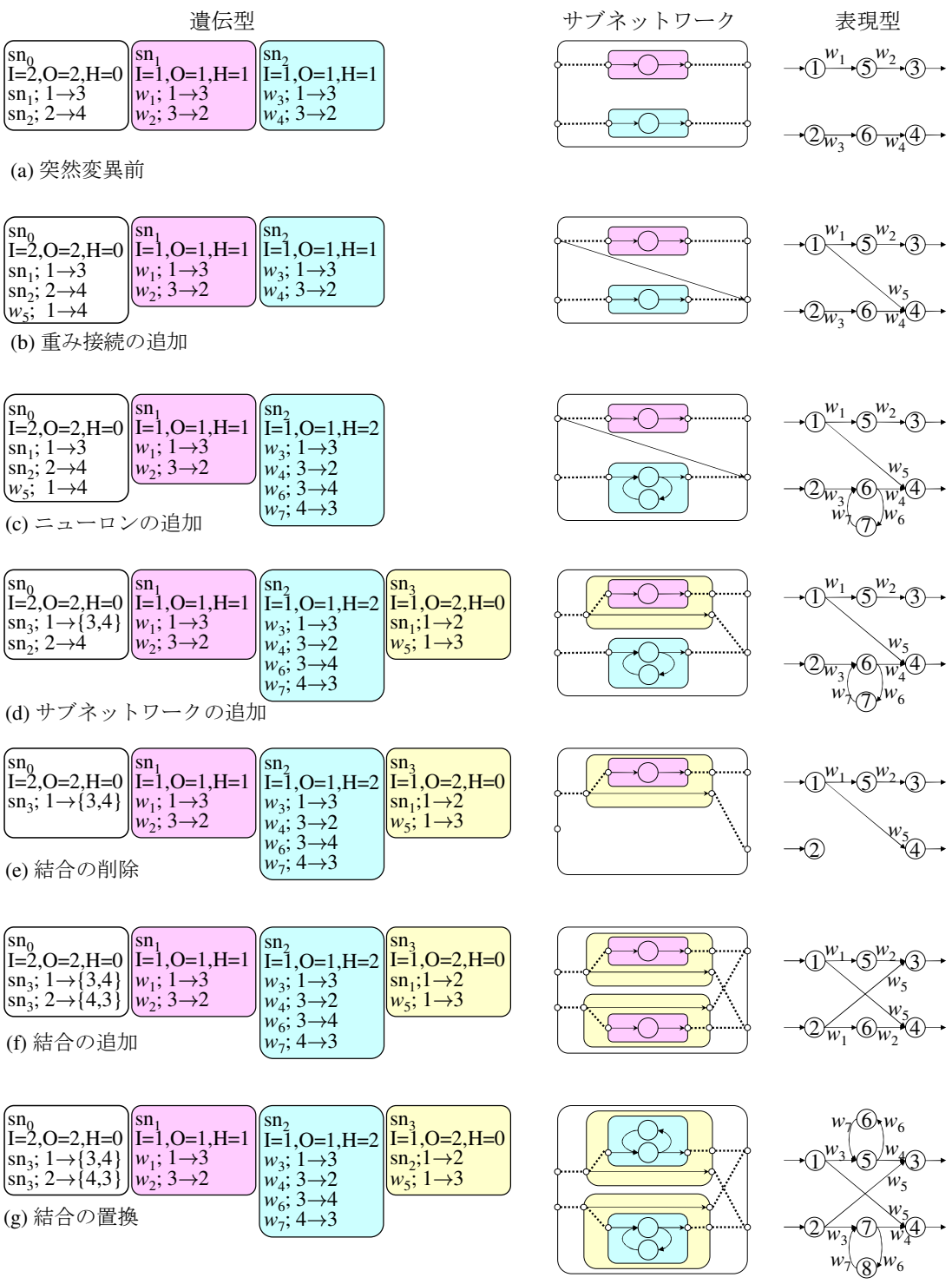


図 2.11 突然変異による構造探索

に従って生成される．ここで，$\langle \boldsymbol{w} \rangle^{(g)}$ は世代 g における適応度の高い μ 個の個体の重み付き平均であり，$\boldsymbol{w}_{i:\mu}^{(g+1)}$ を適応度順に μ まで並べた上位 i 番目の重みとしたとき，$\langle \boldsymbol{w} \rangle^{(g)} = \sum_{i=1}^{\mu} \omega_i \boldsymbol{w}_{i:\mu}^{(g+1)}$．ここで，重み $\omega_i,\ i = 1, \ldots, \mu$ は $\omega_i > 0, {}^{\forall} i$ かつ $\sum_{i=1}^{\mu} \omega_i = 1$ を満たす．$N_k^{(g)}(\boldsymbol{0}, {\sigma^{(g)}}^2 \boldsymbol{C}^{(g)})$ は平均 0，共分散行列 ${\sigma^{(g)}}^2 \boldsymbol{C}^{(g)}$ の正規分布で，$\boldsymbol{z}_k$ はそれに従う独立な変数である．共分散行列 $\boldsymbol{C}^{(g)}$ とグローバルステップサイズ $\sigma^{(g)}$ は突然変異パラメタである．共分散行列 $\boldsymbol{C}^{(g)}$ は

$$
\begin{aligned}
\boldsymbol{p}_c^{(g+1)} &= (1 - c_c)\boldsymbol{p}_c^{(g)} + H_\sigma^{(g+1)} \sqrt{c_c(2 - c_c)} \frac{\sqrt{\mu_{eff}}}{\sigma^{(g)}} \left(\langle \boldsymbol{w} \rangle^{(g+1)} - \langle \boldsymbol{w} \rangle^{(g)} \right) \\
\boldsymbol{C}^{(g+1)} &= (1 - c_{cov})\boldsymbol{C}^{(g)} + c_{cov} \frac{1}{\mu_{cov}} \boldsymbol{p}_c^{(g+1)} \left(\boldsymbol{p}_c^{(g+1)} \right)^T \\
&\quad + c_{cov} \left(1 - \frac{1}{\mu_{cov}} \right) \sum_{i=1}^{\mu} \frac{\omega_i}{{\sigma^{(g)}}^2} \left(\boldsymbol{w}_{i:\lambda}^{(g+1)} - \langle \boldsymbol{w} \rangle^{(g)} \right) \left(\boldsymbol{w}_{i:\lambda}^{(g+1)} - \langle \boldsymbol{w} \rangle^{(g)} \right)^T
\end{aligned}
$$

に従い更新される．ここで，c_c, c_{cov}, μ_{cov} は進化パラメタであり，

$$
H_\sigma^{(g+1)} = \begin{cases} 1 & \text{if } \frac{||\boldsymbol{p}_\sigma^{(g+1)}||}{\sqrt{1-(1-c_\sigma)^{2(g+1)}}} < \left(1.5 + \frac{1}{n-0.5}\right) \mathrm{E}\left(||\mathcal{N}(0, \mathrm{I})||\right) \\ 0 & \text{otherwise} \end{cases},
$$

$$
\mu_{eff} = 1 / \sum_{i=1}^{\mu} \omega_i^2
$$

である．$\mathrm{E}\left(||\mathcal{N}(0, \mathrm{I})||\right)$ は n 変量標準正規乱数のノルムの平均値を示し，通常近似値 $\sqrt{n}(1 - 1/(4n) + 1/(21n^2))$ が用いられる．グローバルステップサイズ $\sigma^{(g)}$ は

$$
\begin{aligned}
\boldsymbol{p}_\sigma^{(g+1)} &= (1 - c_\sigma)\boldsymbol{p}_\sigma^{(g)} + \sqrt{c_\sigma(2 - c_\sigma)} \boldsymbol{B}^{(g)} {\boldsymbol{D}^{(g)}}^{-1} \boldsymbol{B}^{(g)^T} \frac{\sqrt{\mu_{eff}}}{\sigma^{(g)}} \left(\langle \boldsymbol{w} \rangle^{(g+1)} - \langle \boldsymbol{w} \rangle^{(g)} \right) \\
\sigma^{(g+1)} &= \sigma^{(g)} \exp\left(\frac{c_\sigma}{d_\sigma} \left(\frac{||\boldsymbol{p}_\sigma^{(g+1)}||}{\mathrm{E}\left(||\mathcal{N}(0, \mathrm{I})||\right)} - 1 \right) \right)
\end{aligned}
$$

により更新される．ここで，c_σ, d_σ は進化パラメタであり，直交行列 $\boldsymbol{B}^{(g)}$ と対角行列 $\boldsymbol{D}^{(g)}$ は共分散行列 $\boldsymbol{C}^{(g)}$ の固有値分解 $\boldsymbol{C}^{(g)} = \boldsymbol{B}^{(g)} {\boldsymbol{D}^{(g)}}^2 \boldsymbol{B}^{(g)^T}$ によって得られる．

実験では全て先行研究で推奨されているパラメタを用いた．すなわち，重みの

数を n として

$$
\begin{aligned}
&\lambda = 4 + 3\ln n, \\
&\mu = \lambda/4, \\
&\omega_{i=1} = \frac{\ln(\mu+1) - \ln(i)}{\sum_{j=1}^{\mu} \ln(\mu+1) - \ln(j)}, \quad i = 1, \ldots, \mu \\
&c_\sigma = \frac{\mu_{eff} + 2}{n + \mu_{eff} + 3}, \\
&d_\sigma = 1 + 2\max\left(0, \sqrt{\frac{\mu_{eff} - 1}{n+1}} - 1\right) + c_\sigma, \\
&c_c = \frac{4}{n+4}, \\
&\mu_{cov} = \mu_{eff}, \\
&c_{cov} = \frac{1}{\mu_{cov}} \frac{2}{\left(n + \sqrt{2}\right)^2} + \left(1 - \frac{1}{\mu_{cov}}\right) \min\left(1, \frac{2\mu_{eff} - 1}{(n+2)^2 + \mu_{eff}}\right)
\end{aligned}
$$

とした [79]．また，探索の停滞を防ぐためグローバルステップサイズの最小値についての閾値 $\sigma^{(g)}\lambda_{min}^{(g)} \geq \sigma_{\min} = 0.05$ を導入した．ここで，$\lambda_{min}^{(g)}$ は共分散行列 $\boldsymbol{C}^{(g)}$ の最小固有値を示す．初期値は $\sigma^{(0)} = 0.1, \boldsymbol{C} = \mathrm{diag}(1, \ldots, 1)$ とした [83]．

また，解を得た場合以外に以下 3 つの条件のうちいずれかを満たした時に CMA-ES を終了して構造探索を行う．

- サンプリング回数 $g \geq 500$.
- 個体間の重みの標準偏差 $std\{\boldsymbol{w}^{(g)}\} \geq 10$.
- 個体間の重みの標準偏差 $std\{\boldsymbol{w}^{(g)}\} < 0.05$.

6. シミュレーション実験

6.1 2 重倒立振子安定課題

提案手法を速度情報を用いない 2 重倒立振子安定課題 (Double Pole Balancing without Velocity; DPNV) へと適用した．DPNV 課題は NE のベンチマーク課題

としてこれまでにも多くの適用事例がある [10, 37, 38, 42, 83, 84]. DPNV 課題は図 2.12 に示すように 2 本のポールを取り付けられた台車によって構成される．台車とポールの力学系は付録 1 に示す．実験では $N_p = 2, m_c = 1\text{kg}, m_1 = 0.1\text{kg}, l_1 = 0.5\text{m}, l_2 = 0.1l_1, m_2 = 0.1m_1, \mu_c = 5 \times 10^{-4}, \mu_1 = \mu_2 = 2 \times 10^{-6}$ とした．各状態はステップサイズ $\tau = 0.01\text{sec}$ で 4 次のルンゲクッタ法によって更新される．シミュレーションプログラムは Stanley らが配布しているものを使用した [37].

台車の移動できる範囲は限定されており，台車を動かすことで 2 本のポールの角度を一定の範囲に留めなくてはならない．制御器の入力は (x, θ_1, θ_2) のみで，台車の速度 $\dot{x}$ 及び各ポールの各速度 $\dot{\theta}_1, \dot{\theta}_2$ は入力されないので，DPNV は部分観測問題である．このため DPNV 課題を解くための構造として，少なくとも 1 つのリカレント接続が必要である．制御器の出力は正規化された台車へ与える力 $F = [-10\text{N}, 10\text{N}]$ である．

適応度関数と終了条件は Gruau らによって提案された方法を用いた [42]．同手法は以降多くの NE 研究で適用されている [37, 38, 83, 84]．適応度関数を以下で与える．

$$
\begin{aligned}
fitness &= 0.1f_1 + 0.9f_2 \\
f_1 &= T/1000, \\
f_2 &= \begin{cases} 0 & \text{if } T < 100 \\ \frac{0.75}{\sum_{i=T-100}^{T}(|x^i| + |\dot{x}^i| + |\theta_1^i| + |\dot{\theta_1}^i|)} & \text{otherwise} \end{cases}
\end{aligned}
$$

ここで，T は初期値 $\{x, \dot{x}, \theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2\} = \{0.0\text{m}, 0.0\text{m/sec}, 1°, 0°/\text{sec}, 0°, 0°/\text{sec}\}$ から 2 本のポールの角度を $\theta_1, \theta_2 \in [-36°, 36°]$ に保った時間ステップを示し，最大で $T \leq 1000$ である．f_2 に二番目のポールに関する評価が考慮されていないことに注意されたい．

1 試行は 2 つの終了条件を満たす個体が得られるまで行われる．最初の終了条件は適応度関数の評価と同じ初期状態から 2 本のポールを 10^5 時間ステップ $(10^5 \times 0.01 = 1000\text{sec})$ の間 $x \in [-2.4\text{m}, 2.4\text{m}]$ および $\theta_1, \theta_2 \in [-36°, 36°]$ を保持すること．2 つ目の条件は一般化テストと呼ばれ，625 のうち 200 以上の初期状態で 1000 時間ステップ $(1000 \times 0.01 = 10\text{sec})$ 各状態を保持することである．625 種類の初期状態

は$(x, \dot{x}, \theta_1, \dot{\theta_1}, \theta_2, \dot{\theta_2}) \in \{4.32k_1-2.16, 2.70k_2-1.35, 0.123k_3-0.062, 0.3k_4-0.15, 0, 0\}$で与えられる．ここで，各変数は5つの値を取り$k_1, k_2, k_3, k_4 \in \{0.05, 0.25, 0.5, 0.75, 0.95\}$である．

制御器は離散時間リカレントニューラルネットワーク (RNN) によって構成する．各ニューロンの状態x_jは

$$x_j(t+1) = \sum_{i \in \mathcal{I}} w_{ij} input_i(t+1) + \sum_{i \in \mathcal{H} \bigcup \mathcal{O}} w_{ij} \sigma(x_i(t)) \tag{2.1}$$

によって更新される．ここで，$\mathcal{I}, \mathcal{O}, \mathcal{H}$はそれぞれ入力，出力，隠れニューロンのインデックスを示す．$input_i$は制御器が受け取るi番目のセンサ入力である．w_{ij}はニューロンiからjへの結合重みであり，$\sigma(x)$はシグモイド関数$1/(1+\exp(-x))$である．

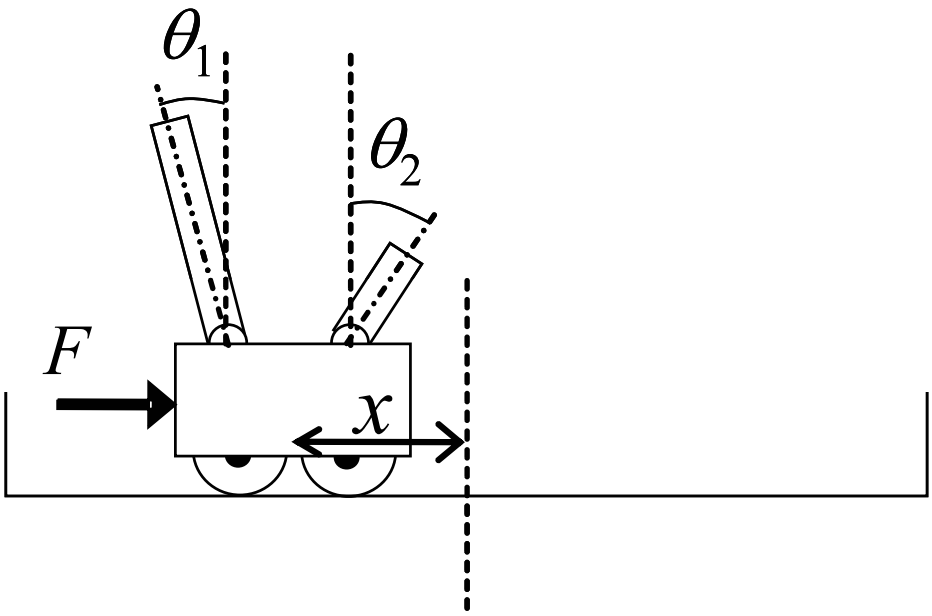


図 2.12　2 重倒立振子安定課題

結果と考察

実験では各世代の集団数に $M = 10$ と非常に小さい集団数を用いた．初期集団における個体は，全て入力から出力への重みつき結合のみの単純な構造を持つ．各個体は固定構造のもとで CMA-ES によって重みの最適化を行う．その後，各個体の適応度に比例した確率に応じて次の世代の個体を選択する (ルーレット選択)．選択された個体に対して，5.1 節で示した突然変異が行われる．各サブネットワークに新しい構造を追加する 3 つの遺伝子操作に対する突然変異率は $p_n = 0.8$ とした．既存サブネットワークの再利用による構造探索を行う 3 つの遺伝子操作に対する突然変異率 p_e は本研究で重要な意味を持つ．そこで $p_e \in \{0.0, 0.4\}$ と 2 つの値を試した．$p_e = 0.0$ のとき，モジュール構造の再利用は行われず，構造探索の方法は直接的符号化手法と同じとなる．

提案手法と既存の手法を比較した実験結果を表 2.1 に示す．表中の Evaluations は適応度を評価した個体の数であり，世代数と集団数に依存しない数である．数値は 20 回試行した平均値である．提案手法は全ての試行において成功し，CE よりも早く解を見つけることができたが，NEAT，AGE を上回ることはできなかった．提案手法における $p_e = 0.4$ と $p_e = 0.0$ の比較から分かるようにモジュールを利用した探索が有効に働いていないことが分かる．DPNV 課題は隠れニューロンを 1 つ持つ RNN によって解けることが示されており [37]，モジュール構造を必要としない課題である．このため，モジュール構造を活用した構造探索を特徴とする提案手法が，DPNV 課題において，その他の手法と比べて効率的でなくとも不思議ではない．提案手法において構造探索を行った回数である世代数が非常に低いことからも DPNV 課題においての構造探索が容易であることが分かる．ただし，提案手法 ($p_r = 0.4$) において，モジュール構造を利用した構造探索が見られた．図 2.13 に提案手法 $p_e = 0.4$ によって得られた解を示す．図 2.13(b) から分かるようにサブネットワーク sn_4 がモジュールとして $\mathrm{sn}_{0,3,5}$ で利用され，表現型上でも同一の結合重みを複数使ってネットワークを表現していることが分かる．

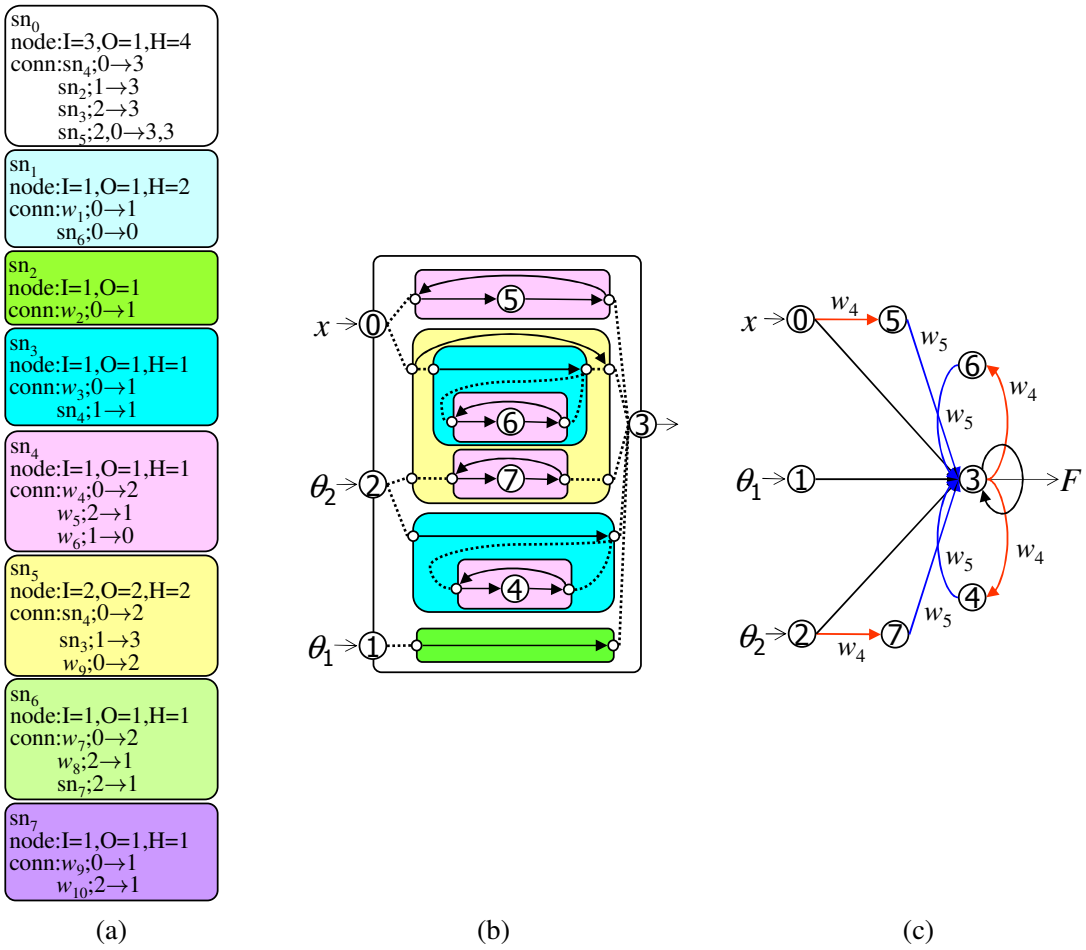


図 2.13 モジュールを利用した構造探索．(a) 遺伝型 (b) 復号化されたサブネットワーク構造 (c) 表現型

表 2.1 実験結果の比較．CE [42]，NEAT [37] および提案手法 $p_e \in \{0.0, 0.4\}$.

Method	Evaluations	generation
proposed($p_e = 0.4$)	198,800	2.8
proposed($p_e = 0.0$)	169,980	2.0
CE	840,000	-
NEAT	24,543	-
AGE	25,065	-

6.2 並列倒立振子安定課題

モジュールを再利用することの有効性を検証するため，提案手法を並列倒立振子安定課題に適用した．課題は N_c 台の倒立振子によって構成される．各倒立振子はそれぞれ独立に付録 1 に従う．ただし，$N_p = 1, m_c = 1\text{kg}, m_1 = 0.1\text{kg}, \mu_c = \mu_1 = 0$ とした．各状態はステップサイズ $\tau = 0.01\text{sec}$ のオイラー法によって更新される．シミュレーションプログラムは Barto ら [85] によるものを使用した．

DPNV 課題と同じく，制御器は (2.1) 式で定義し，台車 $i, i = 1, \ldots, N_c$ の位置 x_i とポールの角度 θ_i を入力する．各台車の速度 $\dot{x}_i$ とポールの各速度 $\dot{\theta}$ は入力されず，部分観測問題である．制御器の各出力は各台車へ与える力 $F_i = [-10\text{N}, 10\text{N}]$ である．よって，制御器の入力ノード数 $N_{in} = 2 \times N_c$，出力ノード数 $N_{out} = N_c$ である．各倒立振子は同じダイナミクスに従うため，DPNV 課題と比較して同一モジュールを利用することが有効であると考えられる．

初期状態 $x_i, \theta_i, \dot{x}_i, \dot{\theta}_i, i = 1, \ldots, N_c$ は全てランダムに決められ，2 回続けて 10^5 ステップ全ての台車 i の位置 $x_i \in [-2.4\text{m}, 2.4\text{m}]$ とポールの角度 $\theta_i \in [-12°, 12°]$ を保持できたとき終了となる．

適応度関数は

$$fitness = \frac{T}{\ln(N_n)}$$

により与える．ここで T は全ての台車の状態を定められた範囲内に保った時間ステップを表し，N_n はニューラルネットワークのノード数である．

結果と考察

実験では $N_c = 4$，各台車のポールの長さをそれぞれ 1m, 1m, 0.5m, 0.5m とした．集団数 $M = 50$，最大の世代数を 50 とした．各初期個体は図 2.14 のように各台車の入力と出力を結合したサブネットワークにより初期化される．ただし，各個体における重みの初期値はランダムに決定される．選択方法にはトーナメント数 2 のトーナメント戦略とエリート戦略を採用した．モジュール利用の有効性を比較するため，突然変異率は $p_n = 0.6$，$p_e \in \{0.0, 0.4\}$ とした．

図 2.15 に実験結果の比較を示す．値は 30 試行の平均値であり，全ての試行において 50 世代以内に終了条件を満たした．図 2.15(a) が終了までの評価回数，(b) が世代数の比較を示す．non-modular がモジュール構造を再利用しない探索による結果 ($p_e = 0.0$)，modular がモジュール構造の再利用を有効にした探索による結果 ($p_e = 0.4$) を示す．図から分かるように，モジュール構造の再利用を有効にした探索はモジュール構造を再利用できない場合に比べて，有意に早く解を見つけていることが分かる ($P < 0.001$)．図 2.15(c) に実験によって得られた解の比較を示す．#connection，#nodes，#sn，#weights がそれぞれ表現型における結合数とノード数，遺伝型における subnetwork 遺伝子と weight 遺伝子の数を示す．得られた解は結合数，ノード数共に non-modular，modular どちらにも差が見られなかった．しかし，それを表す遺伝型では，モジュールの再利用を有効にした探索が有意に少ない遺伝子によって解を表現していることが分かる．このことから提案手法ではモジュールの再利用により，遺伝子長の削減が可能であることが分かる．

図 2.16 に $p_e = 0.4$ によって得られた特徴的な解の例を示す．図 2.16(a) に示すように，得られたニューラルネットワークは台車 $1, 4$ を制御するモジュールと台車 $2, 3$ を制御する同一構造のモジュールによって構成されていることが分かる．図 2.16(b) に有向グラフによって表した対応する遺伝型を示す．sn_0 から 2 度の参照があることから，表現型に見られるモジュールは sn_5 によって表されていることが分かる．sn_5 はさらに下位のモジュールによって表現されており，sn_3 は sn_1 と sn_6 から再利用が行われている．これにより sn_3 は表現型上において 4 箇所で発現することが分かる．

図 2.17 に図 2.16 に示した個体が得られるまでの進化の過程を示す．世代 g における遺伝型と世代間に起こった突然変異を示す．各世代における T は CMA-ES による重みの学習における最大の値である．簡略化のため，重み w のインデックスは全て省略した．突然変異:結合の置換における # エラーは connection 遺伝子の入出力ノード数と置換する対象となるサブネットワークの入出力ノード数が一致しなかったことを示す．世代 $g = 5, 6, 7$ において変化がないのはエリート戦略によるものである．世代 $g = 3$ までは性能の変化は見られないが，$g = 4$ で急激な改善が見られる．これは sn_0 における 2 番目の connection 遺伝子の sn_5 への置換操作 $(\mathrm{sn}_0, (\mathrm{sn}_5, \mathrm{con}_2))$ によると考えられる．これにより，図 2.16(a) に見られる 2 台の台車を制御するネットワークを 2 度再利用する基礎が作られている．その後の sn_0 以外に起こる突然変異は全て 2 つのモジュールへ同様に作用する．これによりモジュールを活用した効果的な構造探索を実現していると考えられる．

図 2.14 に示した初期化を行うには各台車に対応する入力ノードと出力ノードについての先見的知識が必要となる．先見的知識を必要としない初期化方法として，sn_0 において定義される入力ノードと出力ノードをランダムに接続し, その結合に決められた回数のサブネットワークの追加操作を行うことで無作為な初期化を行う方法を試した．しかし，この初期化では全く解を見つけることができなかった．これはサブネットワークの接続時における組み合わせが問題だと考えられる．ランダムな初期化であっても，図 2.14 のような 2 入力 1 出力の下位サブネットワークは多く作られる．しかし，先見的知識によって構造化が成されていない場合にサブネットワークの再利用を行うには，結合の追加操作によって，適切な入出力ノードに接続する必要がある．例えば，隠れニューロンがなく 2 入力 1 出力のサブネットワークを接続する組み合わせは ${}_{12}C_2 \times 12 = 792$ 個であり，50 個体のランダム探索では現実的でないことが分かる．

先見的知識を用いた初期化では，入出力ノードの組み合わせは適切であるため，サブネットワークの置換によるモジュールの再利用が有効に作用する．置換操作は同一の入出力ノード数の場合にのみ適用されるため，接続する組み合わせは大幅に減少する．

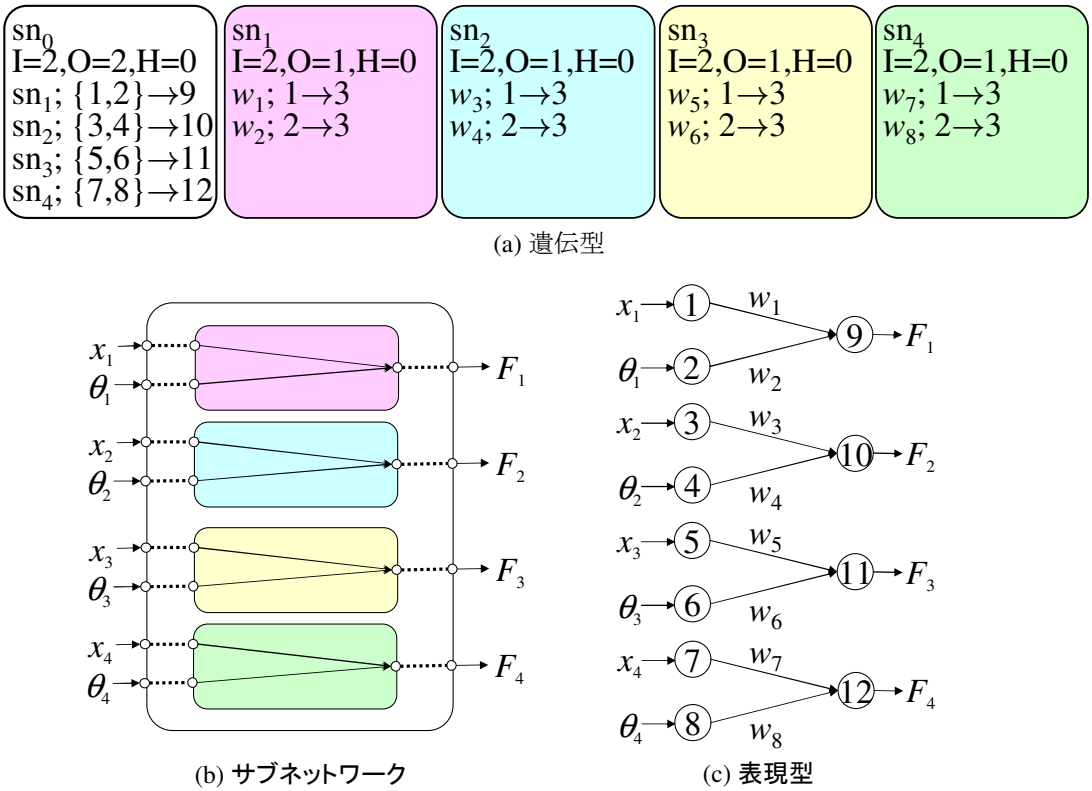
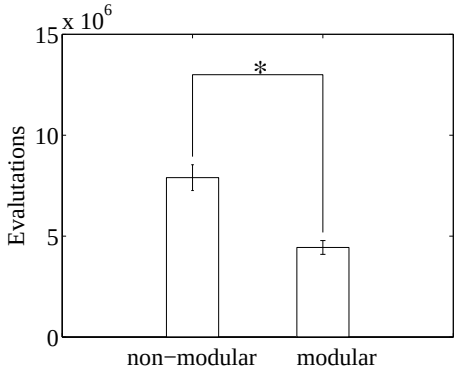
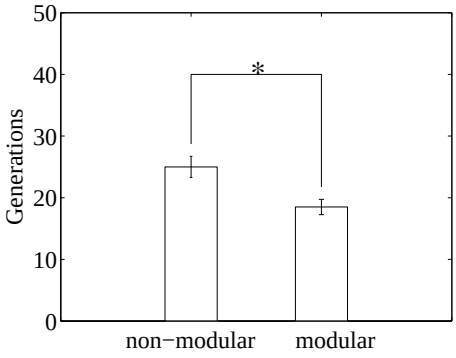


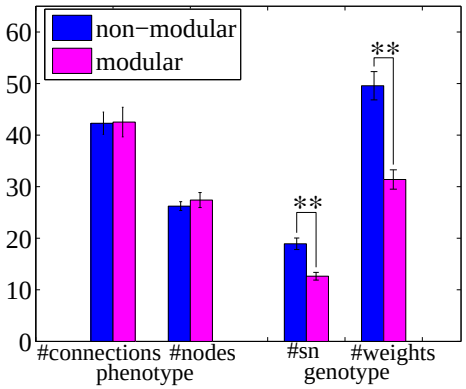
図 2.14　並列倒立振子課題の初期個体



(a) 評価回数

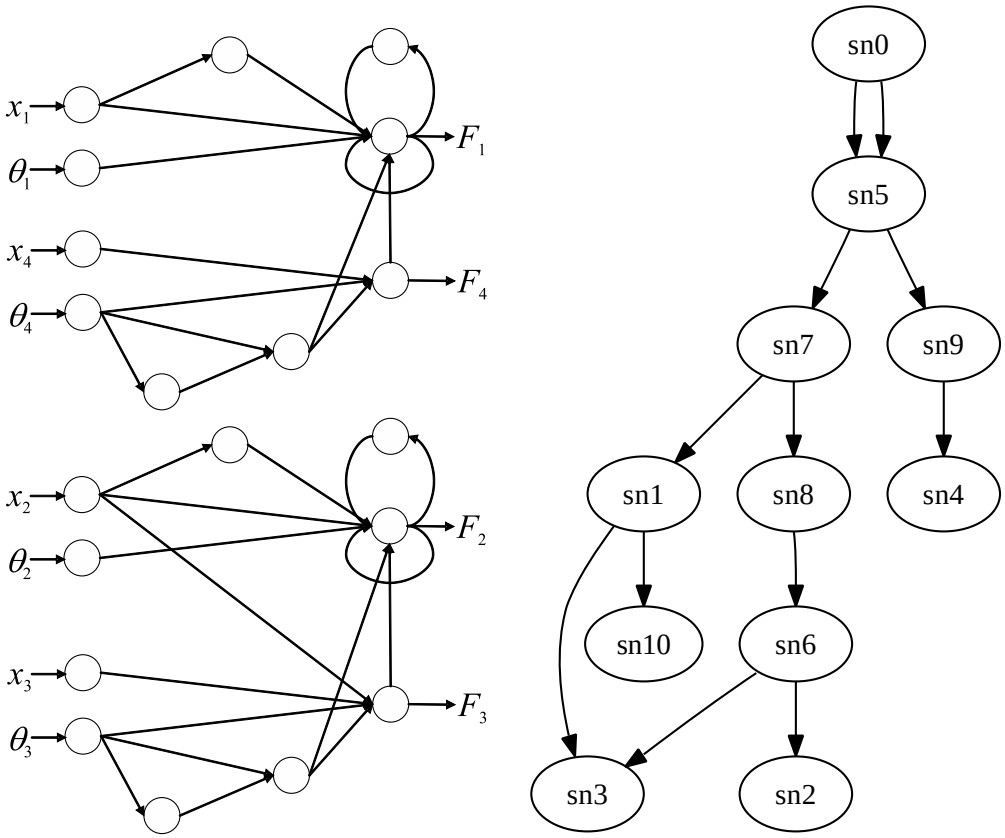


(b) 世代数



(c) 得られた解の比較

図 2.15 並列倒立振子課題の実験結果 ($*$:$P < 0.001$, $**$:$P < 0.00005$)



(a) 表現型　　(b) 遺伝型

図 2.16　モジュールの再利用によって得られた解の例

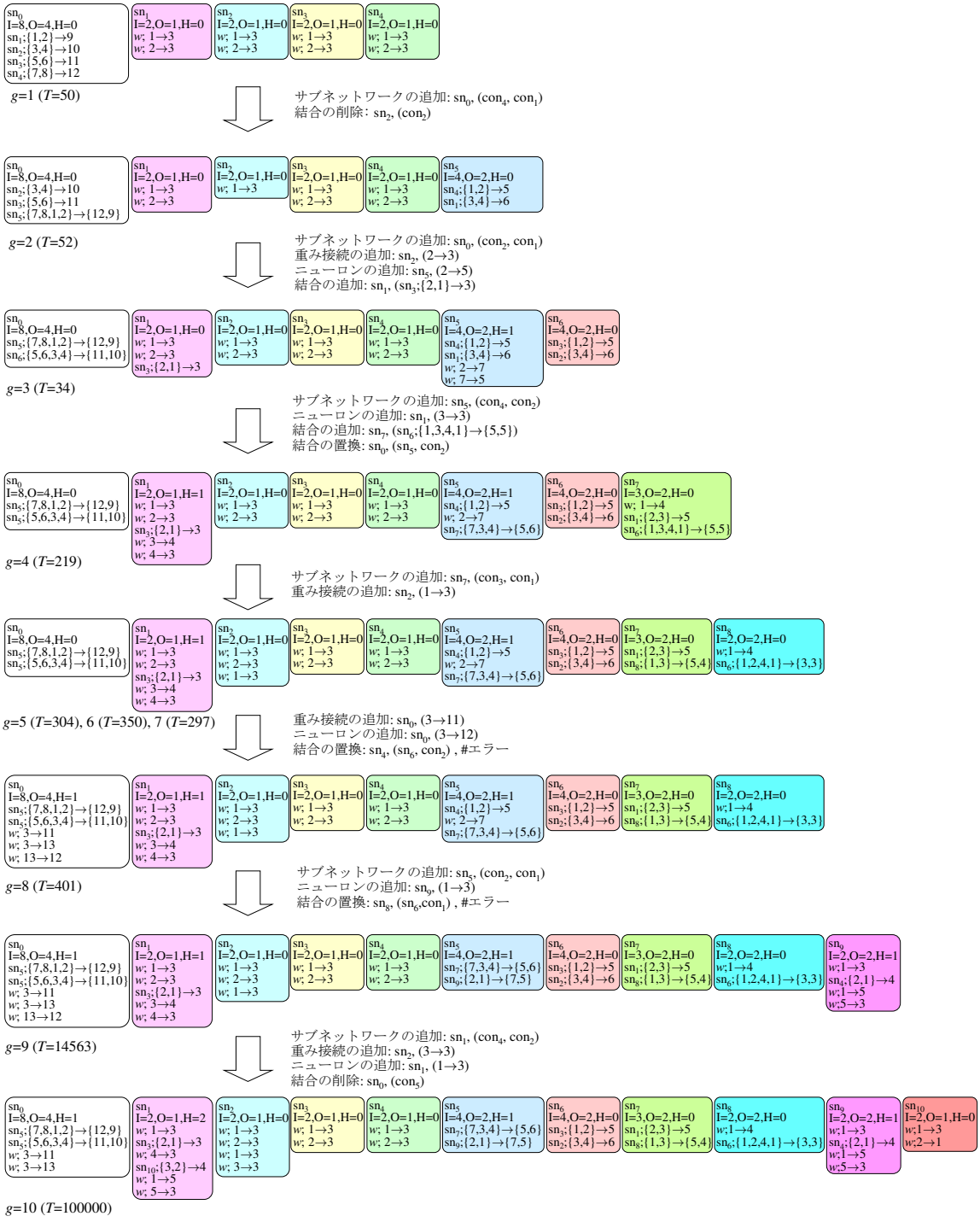


図 2.17 モジュールを利用した進化過程

7. 本章のまとめ

本研究では大規模なネットワークの構造探索に適用可能な新しいNE手法を提案した．提案手法ではニューロン間の接続に重み結合とサブネットワークを同様に扱える遺伝的表現を用いることで，同一モジュールを複数の部分で利用することを可能にした．さらに，これまでに提案されている手法と異なり，下位のモジュールを使ったモジュールの表現が可能である．シミュレーション実験により，提案手法がモジュール構造を再利用しながら問題に合わせた構造探索を実現可能であることを確認した．

提案手法の遺伝的表現は階層的にモジュールを表すことによって，非常に柔軟な構造探索を可能とした．しかし，並列倒立振子課題では，少ない個体をランダムに初期化したのでは解が求まらなかったように構造探索の手法には改善の余地がある．先見的知識を用いずにモジュール構造を探索する場合，探索空間は組み合わせによって増大するため，ランダムな突然変異では困難である．そこで，突然変異に何らかの制約を取り入れる必要ある．制約として制御対象の幾何学的構造が考えられる．Gauci and Stanleyは幾何学的構造を活用した手法であるHyperNEATによってチェッカーを効率的に解けることを示した[86]．ロボットなどの動的システムはボードゲームよりも対称性や繰り返し構造が顕著に見られる．D'Ambrosio and StanleyはHyperNEATを移動ロボットの制御に応用したがロボットの持つ幾何学的構造をそのまま利用するよりも単純化した構造によって表現した方が有効であることを実験的に示した[9]．幾何学的構造にはモジュール構造も含まれるが，そこからモジュール構造を取り出すのは容易ではなく抽象化したモジュール構造を明示的に考慮することは重要であると考えられる．

また，本研究では重みの最適化にCMA-ESを用いたが，他の学習手法との組み合わせも重要な課題である．CMA-ESは進化的計算の一手法であり，生存時間(評価時間)に重みの変化はない．生存時間における学習は進化との間に相互作用があることも示されている[87]．NEによる構造探索と強化学習との組み合わせはこれまでにも研究されており[88–90]，今後の課題としたい．

第3章　Max-Min Actor-Criticによる複数報酬課題の強化学習

1. 複数報酬課題の強化学習

制御器を環境との相互作用からリアルタイムで最適化する枠組みとして強化学習 [6] がある．強化学習を具体的な問題に適用するには報酬関数を定義する必要があるが，実世界の問題では複数の目的や制約を満たすことが求められる多目的最適化問題を解く必要がある．これまで多目的最適化問題への強化学習の適用は，設計者が報酬関数をうまく調整することによって実現されてきた．例えば，

$$r = w_1 r_1 + w_2 r_2 + \cdots + w_M r_M \tag{3.1}$$

のような各目的ごとの報酬 r_i に適当な重み w_i をかけて足し合わせることによるスカラー化が通常行われる．

Natarajan and Tadepalli はこの加重線形和スカラー化に対して，動的に変化させた複数の重みから得られる方策を学習する手法を提案している [91]．この手法では時間的に重み付けが変化する問題において，過去の重み付けに対する方策を用いることで新しい重みに対しての学習が早く収束することを示した．しかし，重みを決定するのは設計者であり，各報酬に適切な重み付けを行うために設計者は，重みの設定，学習，評価といった試行錯誤を繰り返さねばならない．平岡，三島は (3.1) 式で与えられる報酬関数における全ての重みに対する行動価値関数の集合が有限時間と有限の記憶量で同時に獲得できることを示した [92]．理論上は同時に無限通りの報酬関数に対する最適方策が得られるが，実用上は得られた方策から求める方策を探す必要があり，重みの設定と評価の試行錯誤が必要である．Gabor et al. は目的間に明確な辞書式順序が存在する場合の強化学習法を提

案している [93]．しかし，3 目的以上の問題では辞書式順序が設定できない場合が考えられる ($a > b$ かつ $b > c$ かつ $a < c$ のような場合)．

複数の報酬に対して個別に行動価値関数を学習するモジュール強化学習 (modular reinforcement learning; MRL) が提案されている [94–96]．MRL は M 個の報酬関数 r_i に対して個別に行動価値関数 $Q_i(s,a),\ i = 1,\ldots,M$ を学習し，これらの行動価値関数から合成された行動価値関数 $Q(s,a)$ をもとに行動選択を行う手法である．MRL は多目的強化学習問題に有効な方法であるが，その報酬の設計には加重線形和によるスカラー化と同じ問題が残る．Bhat et al. は各モジュールの行動価値 $Q_i(s,a)$ から合成した行動価値 $Q(s,a)$ への変換手法についての検証を行っているが，一般的な変換手法はないと結論付けている [97]．例えば，モジュール間の和によって合成する Greatest Mass (GM) は，等しい割引率のもとでは (3.1) 式によって行動価値関数を学習することと等しく，目的間の競合を報酬の大きさによって解決する手法である．このため，加重線形和によるスカラー化と同じく，適当に重み付けした報酬を設計する必要がある．Humphrys が提案した Negotiated W-learning (NegW) も W 値の計算に Q_i の値が直接影響するため報酬値の大きさによって結果が大きく異なる．このため，複雑な報酬値の調節が不可欠である．モジュール間で最大の価値関数に従う Top Q-learning (TopQ) は行動価値関数が最大値のみに依存するため，GM や NegW ほど報酬値の大きさの影響を受けない．しかし，TopQ は最大の目的を最大化する方策であり，Max-Max 法とも言える．このため，常に一つの目的のみを最大化する可能性がある．上述の移動ロボットの例のように危険な状態や恒常性を扱う場合には全ての目的を偏りなく達成することが必要になる．

これまでの多目的問題に対する強化学習手法の多くは目的間のトレードオフを報酬値のスケーリングによって解決している．このため，実問題への応用の際に報酬の設計が困難になるという問題が生じている．よって，報酬値の大きさに依らない目的間のトレードオフの解決法が必要となる．Max-Min 法は TopQ とは対照的に最小の価値関数を最大化する方策を獲得する手法であり，多目的最適化問題のスカラー化手法の中でも「最も平等的」と言われている．ただし，各報酬の取り得る値域が異なると大きな値を取る報酬は無視されてしまうため，各報酬は

共通の上限値を持つ必要がある．これは，例えば報酬が常に 0 以下の値を取るように決めることで容易に実現することが可能である．Max-Min 法では加重線形和法などの線形変換と異なり，報酬値の大きさが異なっても得られる方策はほとんど変わらないため，設計者は目的別に報酬を設計することができる．さらに，特別なパラメタも必要無いため，報酬関数を設計しやすいと考えられる．Kang and Bien は Max-Min 法による複数報酬に対する動的計画法を提案し，それを基にした強化学習手法も提案している [98] が，Kang and Bien の提案した強化学習手法は最小の価値関数を最大化する Max-Min 最適方策を獲得できない．そこで本研究では Max-Min 最適方策を獲得する強化学習手法を提案する．

1.1 関連研究

複数報酬に対するモジュール強化学習

ここでは複数の報酬に対する MRL について説明する．MRL では M 個の報酬 $r_1, \ldots, r_M$ に対して個別に状態 s と行動 a における各行動価値を $Q_1(s,a), \ldots, Q_M(s,a)$ を学習する．行動選択は合成された行動価値関数 $Q(s,a)$ に従って行われる．

- Greatest Mass (GM)；各価値関数 $Q_i(s,a)$ の和

$$Q(s,a) = \sum_{i=1}^{M} Q_i(s,a)$$

によって合成価値関数 $Q(s,a)$ を計算する．Karlsson は Q-learning によって各 $Q_i(s,a)$ を学習する GM Modular Q-learning[94] を，Sprague and Ballard は Sarsa(0) によって各 $Q_i(s,a)$ を学習する GM Sarsa(0)[95] を提案している．

- Top Q-learning (TopQ)；各価値関数 $Q_i(s,a)$ を Q-learning によって学習し，最大の価値関数

$$Q(s,a) = \max_i Q_i(s,a)$$

によって合成価値関数 $Q(s,a)$ を計算する．Q-learning によって価値関数を学習し，同様の手法が Karlsson[94] と Humphrys[96] によって提案されている．

- Negotiated W-learning (NegW)；Humphrys が提案した NegW では各価値関数 $Q_i(s,a)$ を Q-learning によって学習し，

$$\begin{aligned} W_i(s) &= \max_{a_k}\{\max_a Q_i(s,a) - Q_i(s,a_k)\}, \\ a_k &= \arg\max_a Q_l(s,a),\ l \neq i \end{aligned}$$

として，

$$Q(s,a) = Q_i(s,a), \quad i = \arg\max_j W_j(s)$$

に従い行動選択を行う [96]．NegW は価値の高さに加えて，目的間の独立性を考慮する．行動価値関数において貪欲 (greedy) な行動をとらなかった場合であっても高い価値を持つ目的よりも，貪欲な行動をとらなければ価値が低い目的を優先する．図 3.1 に示す W_i 値は目的 i 以外の目的に従って行動を決定したときにその目的がどの程度達成されるかを定義している．すなわち，ある目的 i が他の目的に従って行動してもある程度報酬を得られるのであれば W 値は小さく，逆にその目的に対する貪欲な行動 a_i^* をとらなければ報酬を得られないような目的に対して W 値は高くなる．よって，NegW は報酬が他の目的の報酬から独立している目的を優先して行動選択を行う手法である．

各手法の詳しいアルゴリズムは付録 2 に示す．

2. 問題

2.1 多目的マルコフ決定過程

マルコフ決定過程 (Markov Decision Process; MDP) を拡張して多目的マルコフ決定過程 (Multi-Objective MDP; MOMDP) を定義する．MDP では報酬はスカラーだが，MOMDP ではベクトルとして与えられる．エージェントは時刻 t における状態 $s_t \in \mathbb{S}$ である行動 $a_t \in \mathbb{A}$ を選択すると確率 $P^{a_t}_{s_t,s_{t+1}}$ で次状態 $s_{t+1} \in \mathbb{S}$ に遷移し，ベクトル報酬 $\boldsymbol{r} = (r_1, \ldots, r_M)^T$ を得る．M は目的の個数であり，ま

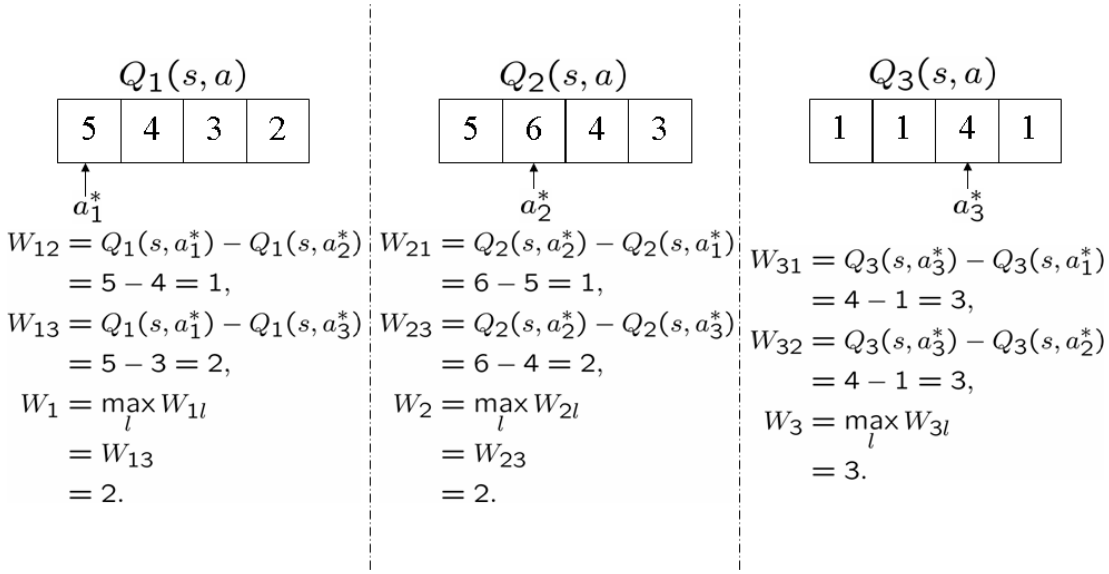


図 3.1 W 値の計算: 目的の数 $M = 3$，行動の数 $|\mathbb{A}| = 4$ とした場合にある状態 s における行動価値関数 $Q_i(s,a)$ が目的 i と行動 a に対して正方形の中の数値であるとする．a_i^* は各行動価値関数における最適行動を示す．このときの W_i は各目的 i での最適行動とその他の目的 l での最適行動の行動価値関数の差である．

た ^{T} はベクトル，行列の転置を表す．報酬は全てゼロ以下 $r_i \leq 0,\ i = 1, \ldots, M$ であると仮定する．状態 s において選択される行動 a は方策 $\pi(s,a)$ と呼ばれる確率に従って決定される．

2.2 価値関数ベクトル

ある方策 π のもとでの MOMDP における各状態の価値として，M 次元のベクトルで表される状態価値関数ベクトル $\boldsymbol{V}^\pi$ を

$$\boldsymbol{V}^\pi(s) = (V_1^\pi(s), \ldots, V_M^\pi(s))^T,$$
$$V_i^\pi(s) = E\left\{\sum_{k=0}^{\infty} \gamma_i^k r_{i,t+k+1} | s_t = s, \pi\right\}, \quad i = 1, \ldots, M$$

で定義する．γ_i は割引率 $(0 \leq \gamma_i < 1)$ で，即時報酬と将来に渡って得られる積算報酬のバランスを決定する．

ここで，求める解は各状態 s において，価値関数ベクトル $\boldsymbol{V}(s)$ を最大化する方策 π を見つけ出すことであるが，これは複数の目的関数 $V_1, \ldots, V_M$ によって定義される多目的最適化問題である．もし，

$$V_i^{\pi^*}(s) \geq V_i^{\pi}(s), \quad {}^{\forall}i, s \in \mathbb{S}$$

なる V^{π^*} が存在すれば，π^* は最適方策である．しかし，通常このような複数の価値関数を同時に最適化するような方策は存在しない．

3. MOMDP の解

3.1 Pareto 最適方策

多目的最適化問題の解として，Pareto 最適解が知られている．Pareto 最適解は次のように定義される．関数 $\{f_1(\boldsymbol{x}), \ldots, f_M(\boldsymbol{x})\}$ において

$$\nexists \boldsymbol{x} \in \mathbb{X} : \boldsymbol{f}(\boldsymbol{x}) \geq \boldsymbol{f}(\boldsymbol{x}^*)$$

なる Pareto 最適解 $\boldsymbol{x}^*$ が定義される [99]．ただし，$\nexists$ は “存在しない” を意味し，ベクトルに対する不等号 $\boldsymbol{a} \geq \boldsymbol{b}$ は

$$\boldsymbol{a} \geq \boldsymbol{b} \overset{\text{def}}{\longleftrightarrow} a_i \geq b_i, \qquad {}^{\forall}i = 1, \ldots, M \text{ かつ } \boldsymbol{a} \neq \boldsymbol{b}$$

で定義される．すなわち，Pareto 最適解とは一つの目的関数を向上させるためにはその他の目的関数を低下させる必要がある解である．MOMDP における Pareto 最適方策 π^p は，

$$\begin{aligned}\{\pi^p \in \Pi \mid {}^{\nexists}\pi \in \Pi,& \\ &\text{such that } \boldsymbol{V}^{\pi}(s) \geq \boldsymbol{V}^{\pi^p}(s), \quad {}^{\forall}s \in \mathbb{S}\},\end{aligned}$$

で定義される．

3.2 Max-Min 最適方策

Pareto 最適解のひとつを求めるには複数の目的関数を単一の目的関数へとスカラー化する方法が一般的である．代表的なスカラー化手法には Max-Min 法や加重線形和法，制約変換法などがある．本研究ではスカラー化法の中でも Max-Min 法によって Pareto 最適方策のひとつを獲得するためのアルゴリズムを提案する．Max-Min 法は目的関数の中で最小の目的関数に対しての最大化を行う手法である．すなわち，Max-Min 最適方策 π^{MM} は

$$\{\pi^{MM} \in \Pi | V_{min}^{\pi^{MM}}(s) \geq V_{min}^{\pi}(s), \pi \in \Pi, ^{\forall} s \in \mathbb{S}\}$$

で定義される．ただし，

$$V_{min}(s) = \min_{i=1,\ldots,M} V_i(s)$$

である．また，

$$V_1^{\pi^e}(s) = V_2^{\pi^e}(s) = \cdots = V_M^{\pi^e}(s)$$

なる方策 $\pi^e \in \Pi$ が存在するとき $\pi^{MM} = \pi^e$ である．このように Max-Min 法では全ての目的関数を等しくする解が得られることから「最も平等的」と言われている [99]．Max-Min 法による最適化では各目的関数の重みのような特別なパラメタは必要ない．問題において Max-Min 最適方策が望ましい方策であれば，単純な複数の報酬の設計によって解くことが可能になる．

4. Max-Min Actor-Critic

本研究では MOMDP に対して Max-Min スカラー化により Max-Min 最適方策を獲得する手法である Max-Min Actor-Critic (MMAC) を提案する．図 3.2 に MMAC の概略図を示す．MMAC は Actor-Critic アルゴリズムを基にしている．評価器はベクトル型の報酬を受け，行動器の方策のもとでの M 次元ベクトルの状態価値関数 $\boldsymbol{V} = (V_1, \ldots, V_M)^T$ を学習する．なお，表記を簡単にするために，

V^π の π を以下では省略する．状態遷移 $s \to s'$ に対して報酬 $\boldsymbol{r} = (r_1, \ldots, r_M)^T$ が得られたとき，各価値関数の TD 誤差を

$$\delta_i = r_i + \gamma_i V_i(s') - V_i(s), \quad i = 1, \ldots, M \tag{3.2}$$

とする．状態価値関数は基底ベクトル $\boldsymbol{\phi}(s)$ とパラメタベクトル $\boldsymbol{v}_i$ によって

$$V_i(s) = \boldsymbol{v}_i^T \boldsymbol{\phi}(s)$$

で表される．パラメタベクトル $\boldsymbol{v}_i$ の更新則は

$$\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + \alpha_c \delta_i \boldsymbol{\phi}(s)$$

となる．ここで，$\alpha_c (0 < \alpha_c < 1)$ は学習率を示す.

行動器は行動優先度 $A(s, a)$ を持ち，状態 s で行動 a を実行する価値である $A(s, a)$ は基底ベクトル $\boldsymbol{\psi}(s, a)$ とパラメタベクトル $\boldsymbol{\theta}$ によって

$$A(s, a) = \boldsymbol{\theta}^T \boldsymbol{\psi}(s, a)$$

で表される．行動選択は A に基づいて確率的に行われる．ϵ-greedy の場合では確率 $1 - \epsilon$ で貪欲 (greedy) な行動 $\arg\max_a \{A(s, a)\}$，確率 ϵ でランダムに選択する．Softmax の場合には

$$\pi(s, a) = \frac{\exp\{\beta A(s, a)\}}{\sum_{a \in \mathbb{A}} \exp\{\beta A(s, a)\}}$$

のように行動選択確率を定義する．β は逆温度パラメタで，行動選択のランダム性を制御する.

時刻 t における状態 s で方策 $\pi(s, a)$ に従い行動 a を選択し，次状態 s' に遷移したとき，行動器のパラメタ $\boldsymbol{\theta}$ は $\alpha_a (0 < \alpha_a < 1)$ を学習率として，

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_a \tilde{\delta} \boldsymbol{\psi}(s, a)$$

に従い更新する．ここで，行動器の更新係数 $\tilde{\delta}$ は以下に述べるスカラー化によって得られた TD 誤差である．MMAC は各状態 s において最小の価値関数 $V_i(s)$ から得られる TD 誤差

$$\begin{aligned} \tilde{\delta} &= \delta_k \\ k &= \arg\min_i \{V_i(s)\} \end{aligned} \tag{3.3}$$

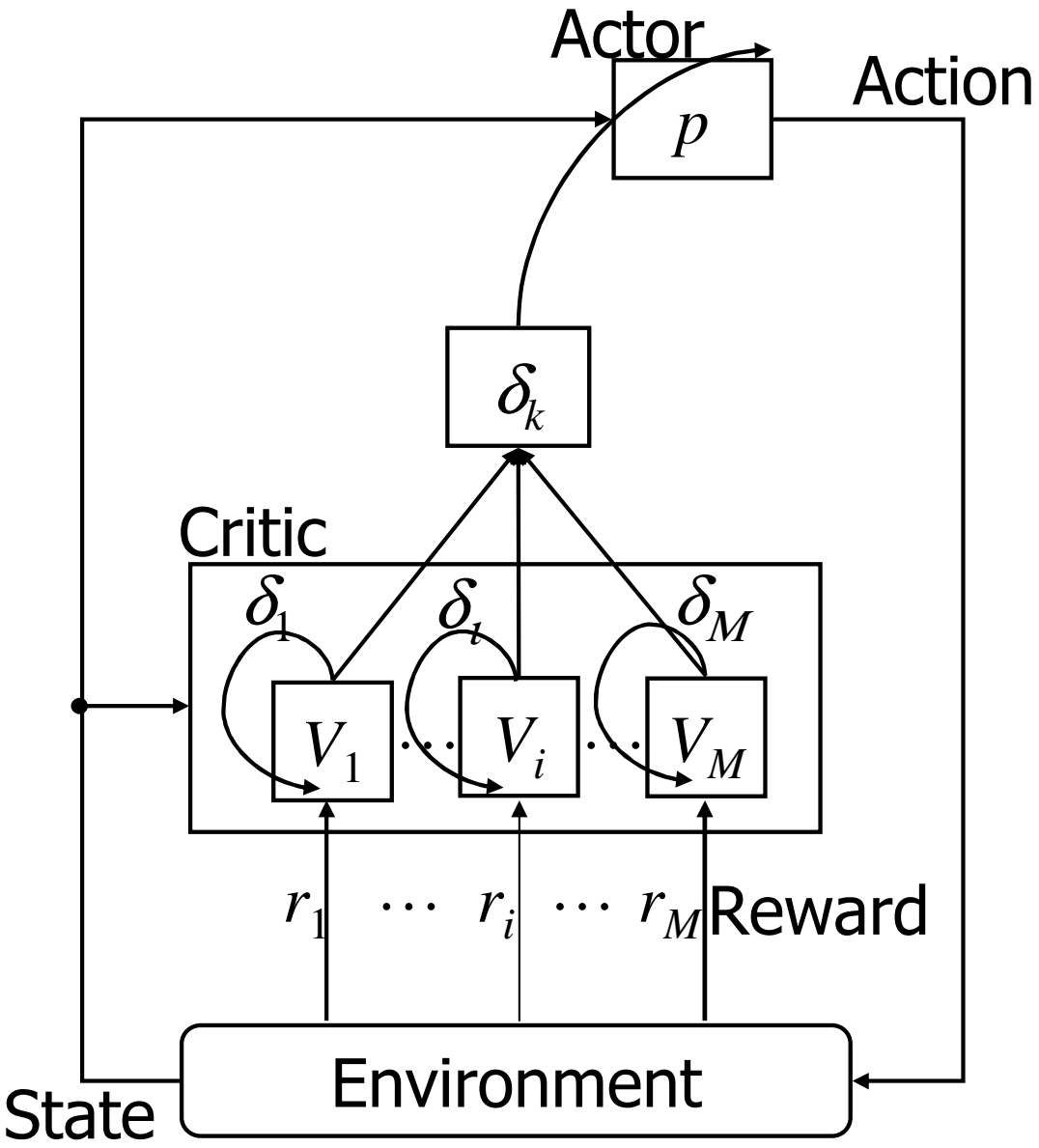


図 3.2 Max-Min Actor-critic の概要図

をスカラー化関数として用いる．

この手法はKang and Bienの提案した手法[98]に似ている．Kang and Bienは Max-Min法をもとにした多目的動的計画法(Multi-Objective Dynamic Programming; MODP)法を提案している．MODPでは決定論的方策のもとで，現在の方策π_tと異なる行動aをとった場合の価値誤差$\Delta(s,a) = Q^{\pi_t}(s,a) - V^{\pi_t}(s)$を定義し，最適行動$a^*$を

$$a^* = \arg\max_a \min_i \{V_i^{\pi_t}(s) + \Delta_i(s,a)\}$$

とした．この方法は環境モデルを既知とした場合の収束が証明されている．また同時に，環境モデルが未知の場合のActor-Critic強化学習法として，

$$\begin{aligned} \tilde{\delta} &= \delta_k \\ k &= \arg\min_i \{V_i(s) + \delta_i\} \end{aligned} \tag{3.4}$$

による方策の更新則を提案している．しかし，環境モデルが未知の問題では真の価値関数は得られず，TD誤差δ_iが価値誤差Δ_iを近似しているとは考えにくい．

5. Max-Min最適方策の確認

5.1 3状態MOMDP課題

図3.3に示す3状態，2行動，2報酬のMOMDP課題によってMMACとKang and Bienの手法(KBAC)を比較する．状態s_0は開始状態，状態s_1とs_2は終端状態である．タスク開始時にエージェントは開始状態s_0において2種類の行動a_1, a_2の中からどちらかを選択して実行する．このとき状態遷移確率は$P_{s_0 s_1}^{a_1} = 1$，$P_{s_0 s_2}^{a_1} = 0$，$P_{s_0 s_1}^{a_2} = p$，$P_{s_0 s_2}^{a_2} = 1 - p$である．エージェントは状態$s_1$に遷移したとき報酬$\boldsymbol{r} = (-1, 0)^T$を，状態$s_2$に遷移したとき報酬$\boldsymbol{r} = (0, -1)^T$を受け取る．各報酬に対する価値関数を$\boldsymbol{V}(s) = (V_1(s), V_2(s))^T$とする．この問題では，2つの報酬の設定が対称であることから，状態遷移確率からMax-Min最適方策が分かるため，方策の評価が容易である．

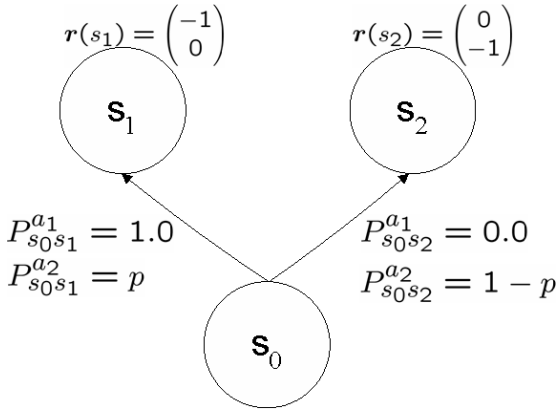


図 3.3　3 状態 MOMDP

評価器の学習率を $\alpha_c = 0.1$，行動器の学習率を $\alpha_a = 0.1$，割引率を $\gamma = 0.9$，逆温度パラメタを $\beta = 1.0$ とした Softmax 法を用いる．学習器のパラメタベクトル $\boldsymbol{v}$ と $\boldsymbol{\theta}$ の初期値は全てゼロとした．1000 時間ステップを 1 エピソードとして，MMAC と KBAC によりそれぞれ 30 エピソード行った．

5.2 結果と考察

図 3.4(a) に MMAC，図 3.4(b) に KBAC によって $p = 0.5$ のもとで得られた報酬の移動平均値を示す．図より，MMAC では学習が進むに従って，報酬 r_1 と r_2 の移動平均がほぼ等しい値へ収束しているのに対し，KBAC では r_1 と r_2 が大きく異なる値へ収束している．Max-Min 最適方策では各目的の得られる報酬の期待値は等しい．このことから，MMAC では Max-Min 方策に近い方策を獲得していると考えられる．

図 3.5 に $p = 0.5$ において状態 s_0 における方策 $\pi(s_0, a_2)$ の変化を示す．$p = 0.5$ のもとでは $\pi(s_0, a_2) = 1$ としたとき状態 s_1 と s_2 への遷移確率が等しくなることから，Max-Min 最適方策は $\pi^{MM}(s_0, a_2) = 1$ であることが分かる．図のように MMAC では $\pi(s_0, a_2) = 1$ を獲得しているのに対し，KBAC では 0.1 付近に収束し Max-Min 最適方策を獲得できていない．

図 3.6 に状態遷移確率 p の変化による平均報酬の比較を示す．$p = 0$ で状態遷移

が確率的でなければ MMAC，KBAC ともに得られる報酬に差が無く，Max-Min 最適方策を獲得していると考えられる．しかし，KBAC では状態遷移の不確実性が増すに連れて報酬の差が大きくなるのに対して，MMAC では KBAC に比べて小さい．これは状態 s_1 と s_2 へ等確率で遷移することが不可能になる $p > 0.5$ であっても変わらず，MMAC が Max-Min 最適方策に近い方策を獲得していることが分かる．

これらの理由は学習過程の TD 誤差を見ることで理解できる．図 3.7(a) に MMAC，図 3.7(b) に KBAC によるあるエピソードにおける学習過程の TD 誤差を示す．図の δ_1, δ_2 はそれぞれ V_1, V_2 の TD 誤差，$\circ, \triangle$ が (3.3) 式もしくは (3.4) 式によって選択されたことを示し，δ_k が選択された TD 誤差を表す．MMAC によって選択された TD 誤差は正負に広く散らばっているのに対し，KBAC ではほとんど 0 以下であり，高い値は選択されない．とくに δ_i が低い値を示したときに選択されていることが分かる．KBAC は (3.4) 式により，価値関数と TD 誤差との和によって評価するため，TD 誤差が大きな負の値の場合に小さい価値関数を最大化するように方策を学習できない．例えば，学習の初期においてはランダムに行動が選択されるため，$p > 0$ のとき s_1 に多く到達し $V_1(s_0) < V_2(s_0)$ となる．行動 a_2 を選択し，状態 s_2 に到達した場合，報酬 $\boldsymbol{r} = (0, -1)^T$ が与えられる．状態 s_2 は終端状態なので $V_1(s_2) = V_2(s_2) = 0$ である．このときの TD 誤差は (3.2) 式より

$$\delta_1 = -V_1(s_0)$$
$$\delta_2 = -1 - V_2(s_0)$$

よって，

$$V_1(s_0) + \delta_1 = 0$$
$$V_2(s_0) + \delta_2 = -1$$

となり，MMAC では $\tilde{\delta} = \delta_1$，KBAC では $\tilde{\delta} = \delta_2$ によって方策が更新される．このとき Max-Min 法による最適化では，低い価値関数 V_1 を最大化するように方策を学習するためにより高い TD 誤差 δ_1 によって方策を更新する必要がある．こ

のように最小の価値関数の TD 誤差が大きい場合にはその探索行動が方策の学習に反映されない場合があるため，KBAC では Max-Min 最適方策を学習することができない．

6. 報酬のスケーリングに関するロバスト性の検証

6.1 実験課題

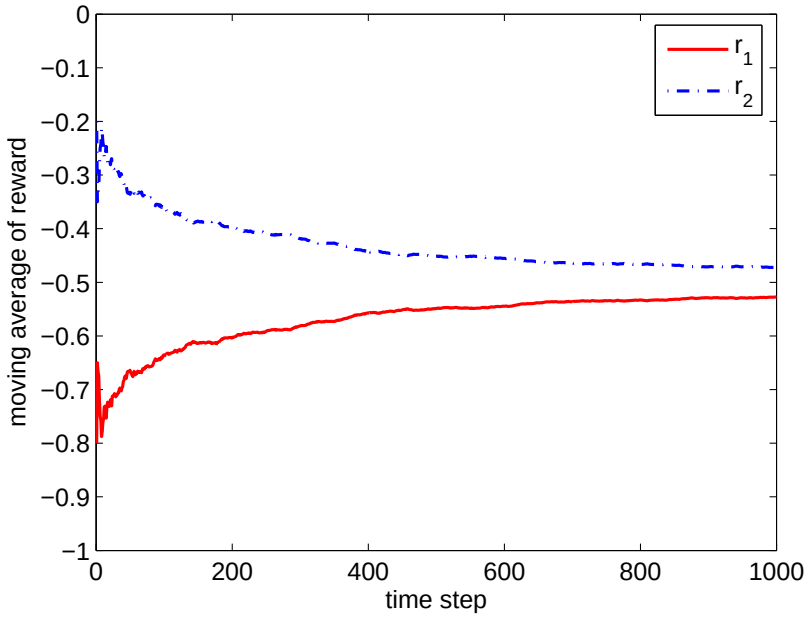
次に，各報酬値のスケーリングが学習に及ぼす影響を調べるために，図 3.8 に示すような環境においてシミュレーション実験を行った．環境は障害物のある 10×10 の格子世界で $(0,0)$ の位置にゴールがあるとする．エージェントは格子世界上の位置 (x,y) を状態として持つ．エージェントは 1 ステップごとに up, down, left, right の 4 つの行動のうち一つを選択する．エージェントの状態は確率 $1-P_r$ で選択した行動に対応した方向の格子に遷移し，確率 P_r で上下左右の格子にランダムに遷移する．エージェントがゴールへ到達した後は障害物やゴールが存在しない格子へランダムに移動する．

エージェントの課題は壁や障害物への衝突を避けつつゴールへより多く到達することである．報酬は 2 次元ベクトル $\boldsymbol{r}=(r_{goal},r_{collision})^T$ で与える．ここで，

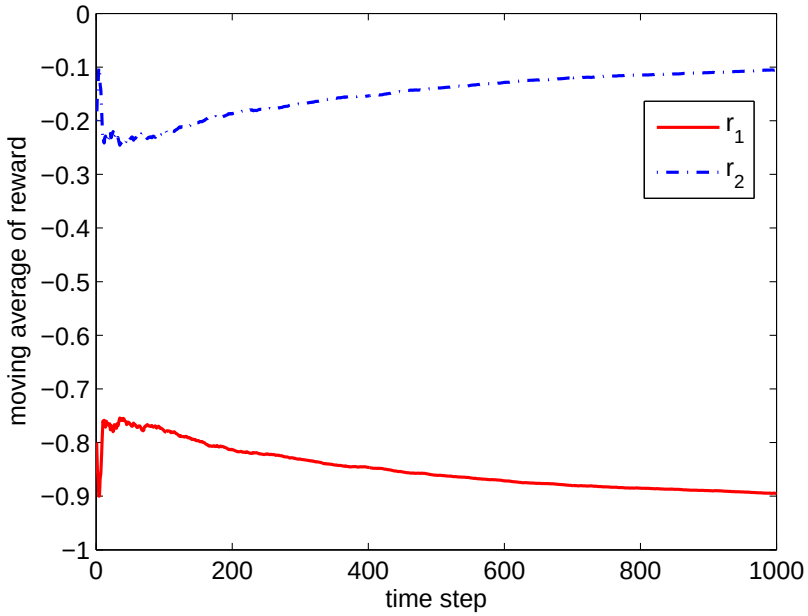
$$r_{goal}=\begin{cases}0 & \text{if the agent reaches the goal}\\ r_1 & \text{otherwise}\end{cases}$$

$$r_{collision}=\begin{cases}r_2 & \text{if a collision occurs}\\ 0 & \text{otherwise}\end{cases}$$

とする．障害物はゴールに近い位置に多く配置される．このため，$P_r>0$ のとき衝突を完全に回避してゴールへ到達することは難しい．実験は 500 時間ステップを 1 エピソードとして，500 エピソード行った．報酬 r_1,r_2 の組み合わせを変化させて，足し合わせ報酬 $r=r_{goal}+r_{collision}$ に対するスカラー価値関数の Actor-Critic(Summed Reward Actor-Critic; SRAC) [6]，Kang and Bien の手法



(a) MMAC



(b) KBAC

図 3.4 移動平均報酬

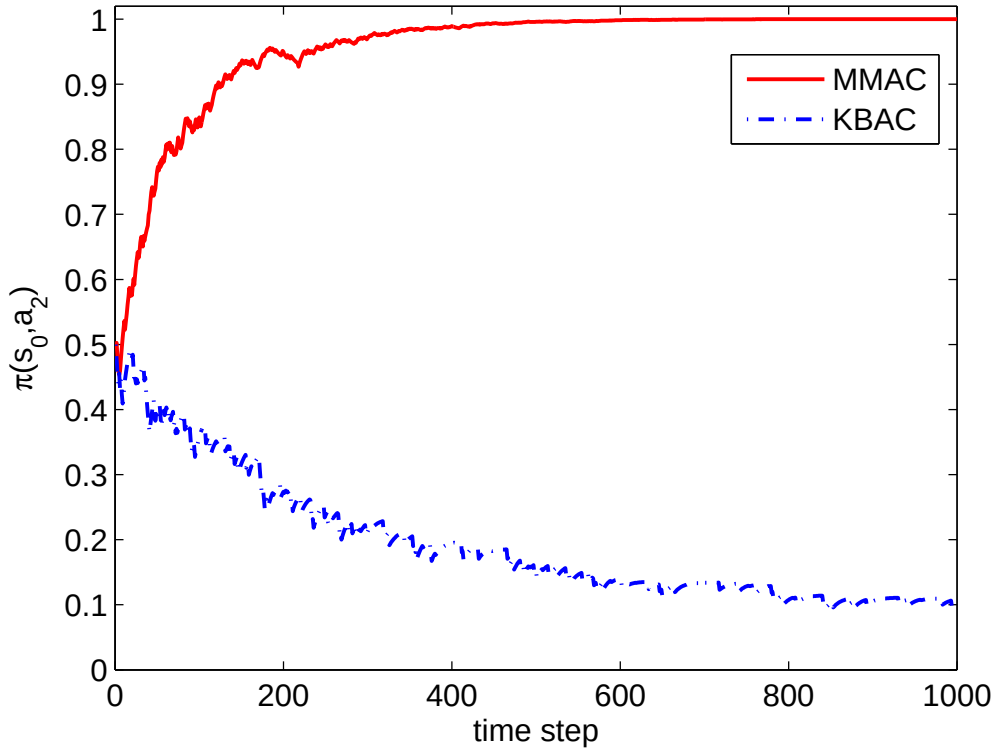


図 3.5 $\pi(s_0, a_2)$ の変化

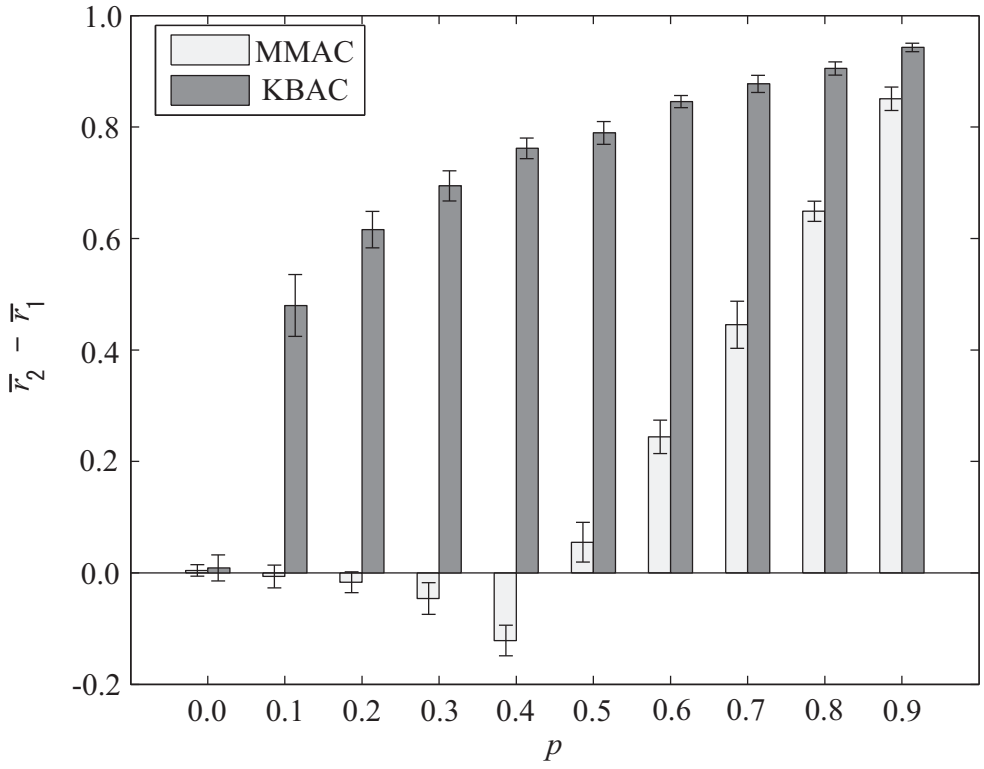
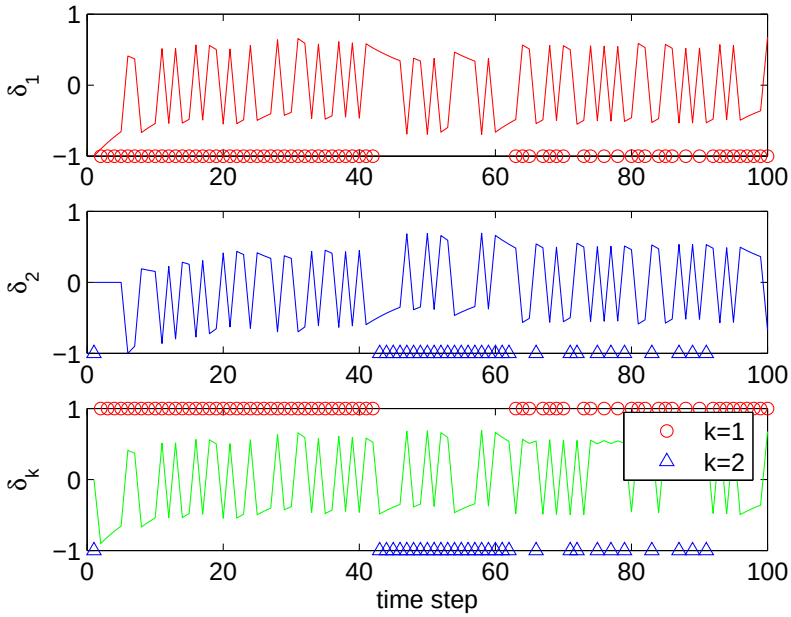
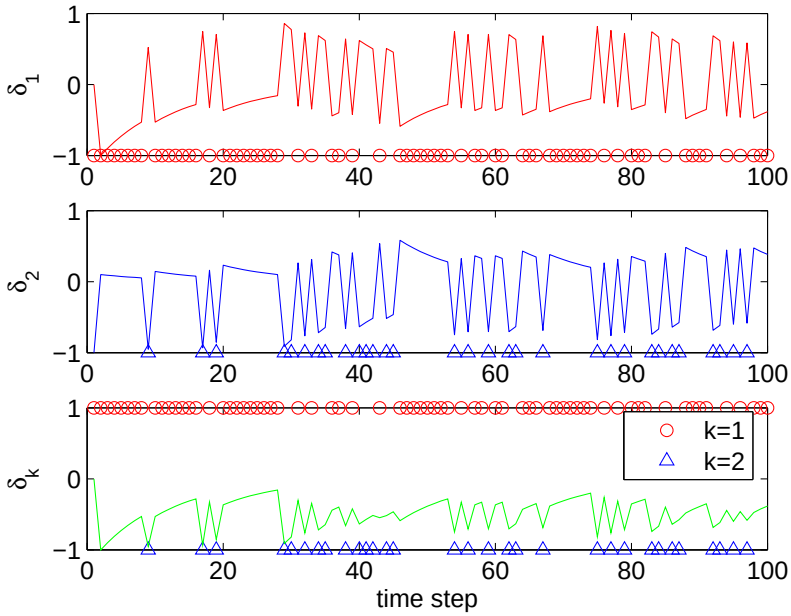


図 3.6 状態遷移確率 p の違いによる結果の比較



(a) MMAC



(b) KBAC

図 3.7 TD 誤差

(Kang and Bien's Actor-Critic; KBAC) [98]，MRL の Greatest Mass Q-learning (GMQL) と Top Q-learning (TopQ) [94]，Greatest Mass Sarsa (GMSarsa) [95]，Negotiated W-learning (NegW)[96] と MMAC の比較を行う．アルゴリズムの詳細は付録 2 を参照されたい．

全ての手法において，近似誤差の影響をなくすため，状態価値関数の基底ベクトル $\boldsymbol{\phi}(s)$ と行動器の基底ベクトル $\boldsymbol{\psi}(s,a)$ は状態と状態行動対にそれぞれ 1 対 1 に対応するルックアップテーブルを用いる．行動選択則として $\epsilon = 0.05$ の ϵ-greedy 法を用いる．各学習パラメタは学習率 $\alpha_c = 0.15$，$\alpha_a = 0.15$，割引率 $\gamma = 0.8$ とする．

6.2 結果と考察

図 3.9 に学習後に 1 エピソード中にゴールへ到達した回数の平均を示す．図 3.9(a) は $P_r = 0.1$，図 3.9(b) は $P_r = 0.3$ の結果である．図の横軸は報酬 r_1, r_2 の設定を表す．図より報酬を $r_1 < r_2$ とし，状態遷移のランダム性 P_r に応じて r_1 を低く設定することでゴールへの到達回数が増えることが分かる．

MMAC では報酬の設定に関わらず高いゴール到達回数を示しているのに対し，その他の手法では $r_1 \geq r_2$ の場合にゴールへ到達する方策を学習できないことが分かる．KBAC は MMAC と同様に報酬の設定に対する変化が少なかった．このことからも Max-Min 手法が報酬の大きさの影響を受けにくく，報酬の設計に有利であることが分かる．ただし，KBAC は 5 節で示したように正しく Max-Min 最適方策を得られない場合があった．実際に $(r_1, r_2) = (-1, -4)$ の場合，MMAC は性能がほとんど変わらなかったのに対し，KBAC では性能が悪くなった．

TopQ がほとんどゴールへ到達できなかった理由は，ほとんどの状態で $r_{collision} = 0$ であるため，高い Q 値に従って行動選択を行う TopQ では衝突回避の行動しか学習できず，報酬の設定に関わらずゴールへ到達しなかったためである．

SRAC の目的関数は

$$\sum_{t}^{\infty} \gamma^t \sum_{i}^{M} r_{i,t}$$

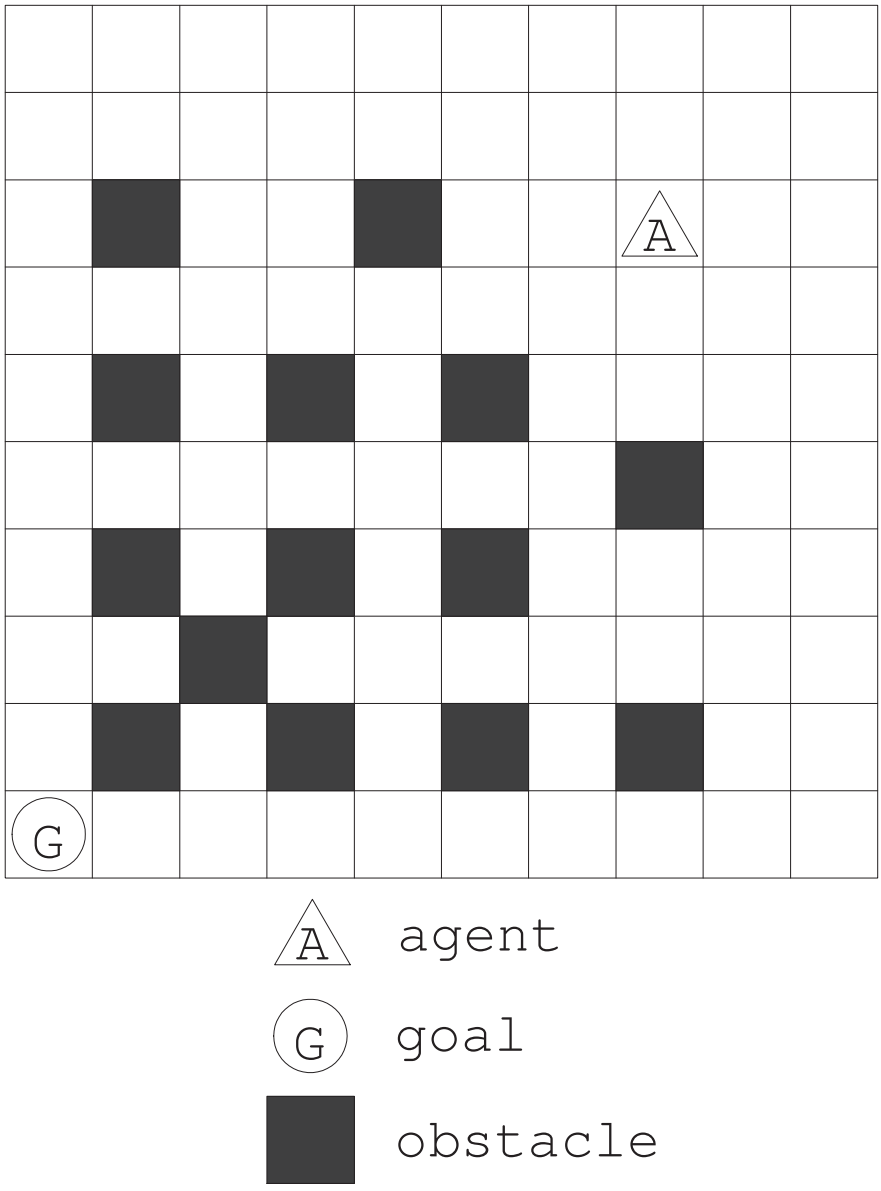


図 3.8 障害物のある grid world

GMSarsa，GMQL の目的関数は

$$\sum_{i}^{M}\sum_{t}^{\infty}\gamma^{t}r_{i,t}$$

である．等しい割引率のもとでは SRAC，GMSarsa，GMQL の目的関数は等しい．報酬値の変化はこれらの目的関数を大きく変えるため，学習に大きな影響を与える．その中でも SRAC と GMSarsa は非常に近い結果となった．両手法は方策オン型学習であることから本質的に同じ方策を学習していると考えられる．GMQL も SRAC と GMSarsa と同じく報酬の和を最大化する方策を学習するが両手法と異なる結果となった．全ての手法で高いゴール到達回数を示した報酬設定 $(r_1, r_2) = (-4, -1)$ において GMQL の方がゴール到達回数が少ないのは Sprague and Ballard が指摘した [95] 方策オフ型学習による正のバイアスによるものと思われる．

NegW は，どちらか一方の行動価値関数に従って行動選択を行うが，その選択において $W_i(s)$ の計算に行動価値関数を直接用いた演算が必要であるために報酬設定の影響を大きく受ける．比較した手法の中で方策の劣化が最も多く見られ，報酬の設定がもっとも難しいと思われる．

図 3.10 に $P_r = 0.3$，$(r_1, r_2) = (-1, -4)$ において MMAC と GMQL によって得られた価値関数を示す．図の白い格子はゴールを示し，黒が障害物を示す．(a),(b) が MMAC，(d),(e) が GMQL によって，報酬 r_{goal} と $r_{collision}$ からそれぞれ学習した価値関数を示す．MMAC，GMQL ともに r_{goal} から個別に学習した価値関数は，ゴールに近づくほど価値が高く，$r_{collision}$ から学習した価値関数は障害物の少ないフィールドの右上部分の価値が高いことが分かる．それに対し，最大化したい価値関数である (c)MMAC における最小の状態価値関数 $\min\{V_{goal}(s), V_{collision}(s)\}$ と (f)GMQL における行動価値関数の足し合わせ $\max_a\{Q_{goal}(s,a) + Q_{collision}(s,a)\}$ には明確な違いが見られる．MMAC ではゴールに向かって価値が高くなるのに対し，GMQL ではフィールドの右上部分に価値の高い状態が存在する．このため，MMAC ではゴールに到達できるのに対して，GMQL はゴールへ到達することができない．

図 3.11 に $P_r = 0.1$，$(r_1, r_2) = (-4, -1)$ における学習の課程を示す．MMAC，

表 3.1 各手法における探索空間

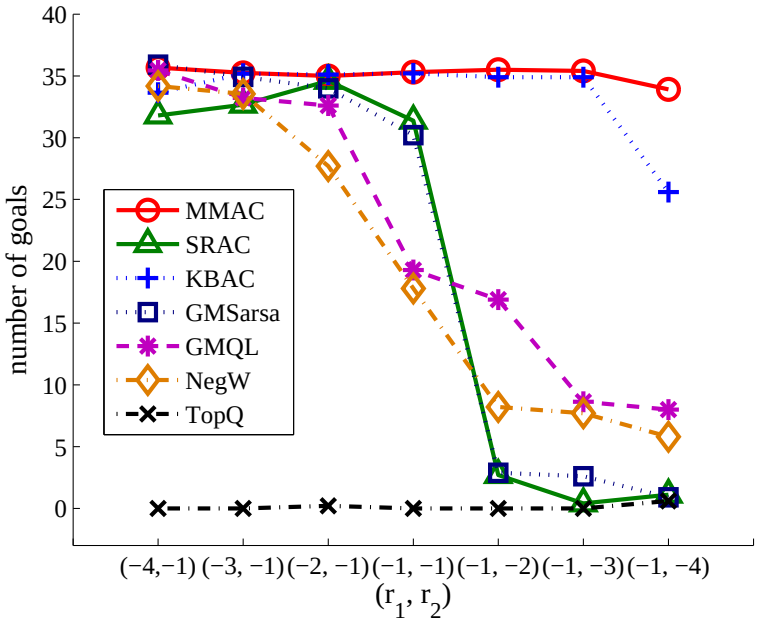
MMAC,KBAC	$M\mathbb{S}+\mathbb{S}\times\mathbb{A}$
SRAC	$\mathbb{S}+\mathbb{S}\times\mathbb{A}$
GMQL, GMSarsa, TopQ, NegW	$M(\mathbb{S}\times\mathbb{A})$

KBAC，SRAC は MRL に比べて非常に早く収束した．学習の収束速度は各手法の探索空間の広さによって決まる．各手法の探索空間を表 3.1 に示す．実験では目的の数 $M=2$ であることから SRAC と MMAC，KBAC の学習速度に差が無かったが報酬の数 M が増えるにつれて MMAC，KBAC の学習速度は遅くなると思われる．提案手法 MMAC は各報酬に対して Q 値を学習する MRL よりも必要メモリ量も少ないため，学習速度は速い．

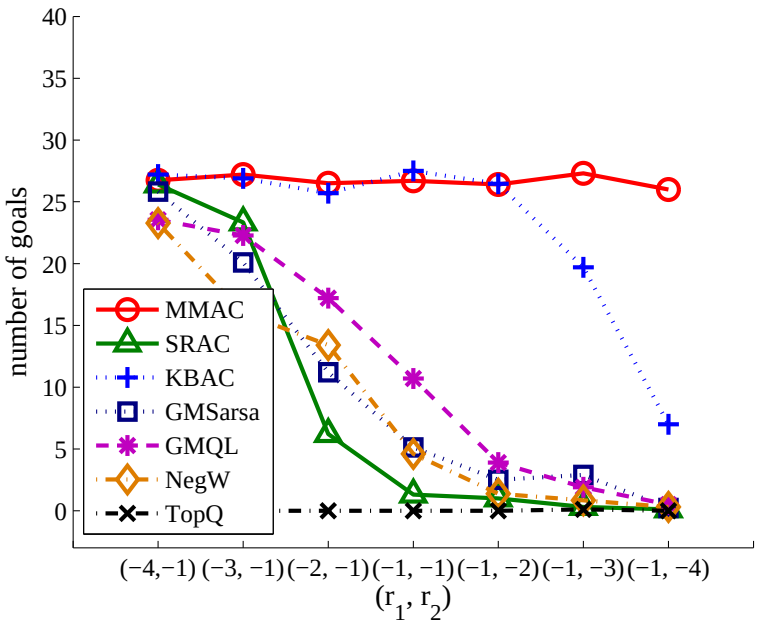
7. 本章のまとめ

強化学習を多目的問題へ適用する際には複数の報酬を適切に設計する必要がある．従来の強化学習ではスカラーの報酬による定式化が行われているため，足し合わせによって複数の報酬を単一の報酬にすることで多目的問題を解決してきた．足し合わせによって報酬をスカラー化する方法では，設計者が目的間のトレードオフを考慮して適切な報酬の重み付けを設計する必要があった．複数の報酬ごとに価値を学習する MRL(GMQL, GMSarsa, TopQ, NegW) も価値関数のバランスが方策に大きな影響を与えるため，報酬の設計には単一報酬に対する強化学習と同様の問題がある．

本研究ではベクトル報酬に対する強化学習を定式化し，各価値関数に対する Max-Min 最適方策を学習する Max-Min Actor-Critic 法を提案した．Max-Min 最適方策は報酬の線形和を最大化する方法と異なり，min 演算によって各状態において最もクリティカルな価値関数を考慮して方策を更新するため，価値関数のバランスが適応的に求められる．類似手法として Kang and Bien の手法があるが Max-Min 最適方策へ収束しない場合があることを確認した．シミュレーション実験によって従来の手法に比べて提案手法は学習速度が速く，報酬のスケーリング



(a) $P_r = 0.1$



(b) $P_r = 0.3$

図 3.9 報酬による学習結果の変化

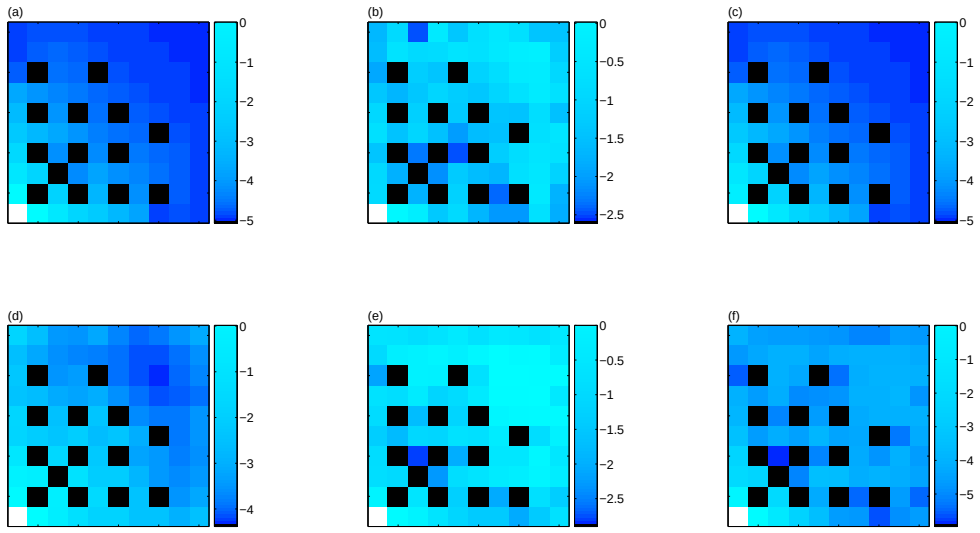


図 3.10 $Pr = 0.3$, $(r_1, r_2) = (-1, -4)$ における価値関数の比較: (a)(b)(c) MMAC によって学習した状態価値関数；(a)$V_{goal}(s)$, (b) $V_{collision}(s)$, (c) 各状態における最小の状態価値関数 $\min\{V_{goal}(s), V_{collision}(s)\}$, (d)(e)(f) GMQL によって学習した行動価値関数; (d) $\max_a\{Q_{goal}(s,a)\}$, (e) $\max_a\{Q_{collision}(s,a)\}$, (f) 足し合わせた行動価値関数 $\max_a\{Q_{goal}(s,a) + Q_{collision}(s,a)\}$.

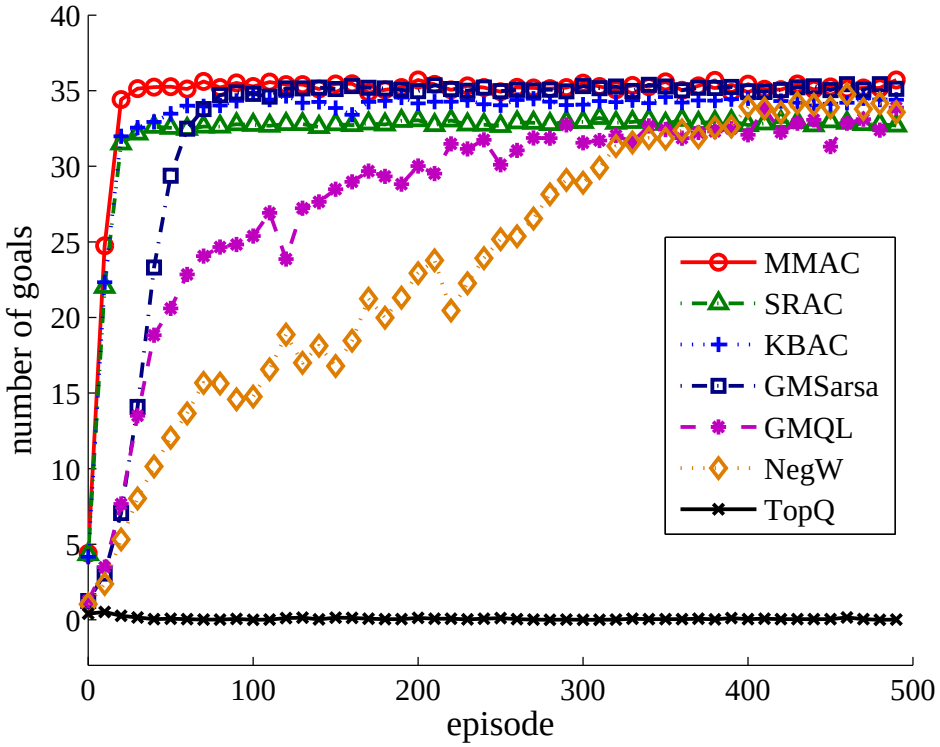

40
35
30
25
20
15
10
5
0
number of goals
0
100
200
300
400
500
episode
MMAC
SRAC
KBAC
GMSarsa
GMQL
NegW
TopQ

図 3.11　学習過程

の影響を受けにくいことを確認した．このことから多目的強化学習問題を解く場合，各目的に対する報酬を個別に設計し，MMAC を適用することが有効である．これにより，多目的意思決定問題に強化学習を適用する際の困難な報酬の設計問題を回避することができる．

本研究は数値実験により，提案手法が多目的問題において簡単な報酬の設計によって，好ましい方策を獲得できることを確認した．しかし，Max-Min 最適方策の有効性は問題に依存する．Max-Min 最適方策は等しい目的間のトレードオフにおける方策であるが，その他のトレードオフに対応した方策を学習することができない．足し合わせ報酬に基づく手法では報酬関数の設計によって目的間のトレードオフを変化させることができるが，報酬関数によらずに柔軟なトレードオフに対応することが今後の課題となる．

第4章　本研究のまとめ

進化的計算や機械学習などの最適化手法を具体的問題へ適用する際は手法の選択だけでなく，制御器の構造，目的関数，メタパラメタなど様々な設計事項を決定する必要がある．本研究では制御器の構造と目的関数の設計問題に関して，進化的計算による制御器の構造獲得手法と目的関数の設定に頑健な強化学習手法を提案した．

2 章では，従来の NE 手法では大規模なネットワークの構造探索が困難であった問題に対して，同一モジュールを複数の部分で利用可能な NE 手法を提案した．シミュレーション実験により，提案手法がモジュール構造を再利用しながら問題に合わせた構造探索を実現可能であることを確認した．大規模なネットワークを必要とする問題には対称性や繰り返し構造を持つなど，同一構造が複数存在する場合が多い．そのような問題に提案手法は有効であると考えられる．

3 章では，複数の目的からなる問題に強化学習を適用する際の報酬設計問題に対して，報酬設定にロバストな強化学習法を提案した．提案手法は各報酬関数に対する価値関数を学習する複数の Critic モジュールと Max-Min 最適方策を学習する 1 つの Actor モジュールによって構成されている．シミュレーション実験によって，Max-Min 最適方策は複数報酬の和を最大化する手法に比べて各報酬値の組み合わせの影響を受けにくいことを示した．

本研究では 2 つの異なる問題に対して，モジュール構造の活用という同じアプローチを取った．3 章で紹介したように，実問題の多くは複数の小問題に分割できる．これらは個別のモジュールによって解かれることが望ましい．本研究で提案した MMAC の Critic における各モジュールは同一構造を与えたが，問題によっては各モジュールが異なる構造であったり，同一構造を持つグループが複数あることが望ましいと考えられる．2 章で示した遺伝的表現手法ではモジュール構造

の再利用方法を探索可能であることから，MMAC における Critic の構造獲得に適している．NE によるモジュール構造を活用した制御器の構造探索と MMAC による重みの学習により，実験者の設計に依存することなく，より複雑な多目的最適化問題を解くことができると考えられる．

謝辞

非常に多くの方々のおかげで，この論文を完成させることができました．ここに感謝の意を表します．

本研究は，銅谷賢治教授及び沖縄大学院大学先行的研究事業神経計算ユニットの内部英治博士の御指導の元で進めてまいりました．銅谷教授にはテーマを決めるに当たって重要なアドバイスを頂きました．そして，迷走しがちな研究をいつも方向修正してくださいました．内部博士は早朝から深夜まで議論に付き合って頂き，常に適切なヒントを授けてくださいました．同じく，神経計算ユニットの研究員の方々には輪講やミーティングで大変貴重な御意見を頂きました．吉本潤一郎准教授には計算論の観点から，伊藤真博士には生物学的な観点から多くの助言を頂きました．さらに，同ユニットの動的システムグループのメンバーとはグループ内ミーティングなどでより密度の濃い議論を行ってきました．特に，ユニット内で数少ない進化計算研究を行っている Stefan Elfwing 博士には非常に重要な御意見を数多く頂きました．中々英語を聞き取れなかったにも関わらず，いつも根気良くアドバイスしてくださいました．また，神経計算学講座の仲間達とは共に遊び，共に飲み，共に研究を進めてきました．特に大塚誠君と中野高志君とはよく遊び，よく議論してきました．平成 20 年度に卒業された森村哲郎博士には本論分執筆に関して，様々なアドバイスを頂きました．また，ここに全ての方のお名前を挙げることはできませんが，同ユニットのメンバー及び元メンバー，一時滞在された研究者の方々，論理生命学講座の方々，計算神経科学講座及び ATR 脳情報研究所の方々には研究を発表させて頂いたり，研究内容を聞かせていただくことで，大変大きな刺激を受けました．計算機技術者の稲嶺安洋さんと井上智文さんにはサーバ管理やライブラリの導入等で大変お世話になりました．生命システム学講座の齋藤絵里さん，論理生命学講座の谷本史さん，神経計算ユニット

の安里恵美子さん，上原智佳子さん，永野いづみさんには事務手続きに関して，本来の職務を越えて御協力頂きました．このような方々の御協力無くしては本論文を提出することはできなかったと思います．ひとえに御厚情の賜物と深く感謝致しております．

そして，論文審査を引き受けて下さった石井信教授，銅谷賢治教授，杉本謙二教授，柴田智広准教授，吉本潤一郎准教授に深く感謝致します．

最後に，健康な体を授け，博士後期課程修了まで精神的，経済的に支えてくれた両親に深く感謝致します．私の体を常に心配し，成功を喜んでくれる家族に感謝致します．本当にありがとうございました．

付録

1. 倒立振子における状態遷移

N_p 本のポールを持つ倒立振子における台車の加速度 $\ddot{x}$ とポールの角加速度 $\ddot{\theta}$ は

$$\ddot{x} = \frac{F - \mu_c \mathrm{sgn}(\dot{x}) + \sum_{i=1}^{N_p} \tilde{F}_i}{m_c + \sum_{i=1}^{N_p} \tilde{m}_i}$$

$$\ddot{\theta} = -\frac{3}{4l_i}\left(\ddot{x}\cos\theta_i + g\sin\theta_i + \frac{\mu_i \dot{\theta}_i}{m_i l_i}\right)$$

$$\tilde{F}_i = m_i l_i \dot{\theta}_i^{\,2}\sin\theta_i + \frac{3}{4} m_i \cos\theta_i \left(\frac{\mu_i \dot{\theta}_i}{m_i l_i} + g\sin\theta_i\right)$$

$$\tilde{m}_i = m_i\left(1 - \frac{3}{4}\cos^2\theta_i\right)$$

に従う [100]．ここで，F[N] は台車を押す力，m_c, μ_c はそれぞれ台車の質量，摩擦係数，$m_i, l_i, \theta_i, \mu_i$ はそれぞれポール i の質量 [kg]，0.5× 長さ [m]，垂直からの角度 [degree]，摩擦係数である．

2. 複数報酬に対する強化学習アルゴリズム

状態 s_t で行動 a_t を実行した結果，報酬 $\boldsymbol{r} = (r_1, \ldots, r_M)^T$ を受け取り，状態 s_{t+1} に遷移したとする．状態 s_{t+1} で実行する行動を a_{t+1} とする．このとき各学習アルゴリズムは次のようになる．ただし，状態価値関数 $V(s)$ と行動価値関数 $Q(s,a)$ はそれぞれルックアップテーブルによって表現されていると仮定する．

Summed Reward Actor-Critic(SRAC)

$$
\begin{aligned}
a_{t+1} &= \epsilon\text{-greedy}(\boldsymbol{\theta}^T \boldsymbol{\psi}(s_{t+1}, *)) \\
\delta &= \sum_{i=1}^{M} r_i + \gamma V(s_{t+1}) - V(s_t) \\
V(s_t) &\leftarrow V(s_t) + \alpha_c \delta \\
\boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} + \alpha_a \delta \boldsymbol{\psi}(s_t, a_t)
\end{aligned}
$$

Kang and Bien's Actor-Critic(KBAC)

$$
\begin{aligned}
a_{t+1} &= \epsilon\text{-greedy}(\boldsymbol{\theta}^T \boldsymbol{\psi}(s_{t+1}, *)) \\
\delta_i &= r_i + \gamma V_i(s_{t+1}) - V(s_t) \\
V_i(s_t) &\leftarrow V_i(s_t) + \alpha_c \delta_i \\
\boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} + \alpha_a \delta_k \boldsymbol{\psi}(s_t, a_t), \\
k &= \arg\min_i \{V_i(s_t) + \delta_i\}
\end{aligned}
$$

Greatest Mass Q-learning(GMQL)

$$
\begin{aligned}
Q(s_{t+1}, a) &= \sum_{i=1}^{M} Q_i(s_{t+1}, a) \\
a_{t+1} &= \epsilon\text{-greedy}(Q(s_{t+1}, *)) \\
\delta_i &= r_i + \gamma \max_{b \in \mathbb{A}} Q_i(s_{t+1}, b) - Q_i(s_t, a_t) \\
Q_i(s_t, a_t) &\leftarrow Q_i(s_t, a_t) + \alpha \delta_i
\end{aligned}
$$

Top Q-learning(TopQ)

$$
\begin{aligned}
Q(s_{t+1}, a) &= \max_i Q_i(s_{t+1}, a) \\
a_{t+1} &= \epsilon\text{-greedy}(Q(s_{t+1}, *)) \\
\delta_i &= r_i + \gamma \max_{b \in \mathbb{A}} Q_i(s_{t+1}, b) - Q_i(s_t, a_t) \\
Q_i(s_t, a_t) &\leftarrow Q_i(s_t, a_t) + \alpha \delta_i
\end{aligned}
$$

Greatest Mass Sarsa(GMSarsa)

$$
\begin{aligned}
Q(s_{t+1}, a) &= \sum_{i=1}^{M} Q_i(s_{t+1}, a) \\
a_{t+1} &= \epsilon\text{-greedy}(Q(s_{t+1}, *)) \\
\delta_i &= r_i + \gamma Q_i(s_{t+1}, a_{t+1}) - Q_i(s_t, a_t) \\
Q_i(s_t, a_t) &\leftarrow Q_i(s_t, a_t) + \alpha\delta_i
\end{aligned}
$$

Negotiated W-learning(NegW)

$$
\begin{aligned}
W_i(s_{t+1}) &= \max_{a_k}\{\max_{a} Q_i(s_{t+1}, a) - Q_i(s_{t+1}, a_k)\}, \\
a_k &= \arg\max_{a} Q_l(s_{t+1}, a), i \neq l \\
Q(s_{t+1}, a) &= Q_i(s_{t+1}, a), \quad i = \arg\max_{j} W_j(s_{t+1}) \\
a_{t+1} &= \epsilon\text{-greedy}(Q(s_{t+1}, *)) \\
\delta_i &= r_i + \gamma \max_{b \in \mathbb{A}} Q_i(s_{t+1}, b) - Q_i(s_t, a_t) \\
Q_i(s_t, a_t) &\leftarrow Q_i(s_t, a_t) + \alpha\delta_i
\end{aligned}
$$

ただし，ϵ-greedy$(f(s, *))$ は確率 $1 - \epsilon$ で $\arg\max_{b \in \mathbb{A}} f(s, b)$ を返し，確率 ϵ でランダムな $b \in \mathbb{A}$ を返す関数とする．

参考文献

[1] A. Eriksson, G. Capi, and K. Doya. Evolution of meta-parameters in reinforcement learning algorithm. In *In Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

[2] X. Yao. Evolving artificial neural networks. *Proc. of the IEEE*, Vol. 87, No. 9, pp. 1423–1447, 1999.

[3] R. Miikkulainen. Evolving neural networks. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO'07)*, pp. 3415–3434, New York, NY, USA, 2007. ACM.

[4] D. Floreano, P. Dürr, and C. Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, Vol. 1, No. 1, pp. 47–62, 2008.

[5] J. Reisinger and R. Miikkulainen. Acquiring evolvability through adaptive representations. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'07)*, pp. 1045–1052. ACM, 2007.

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

[7] S. Nolfi and D. Parisi. *"Genotypes" for neural networks*, pp. 431–434. MIT Press Cambridge, MA, USA, 1998.

[8] C.W. Seys and R.D. Beer. Effect of encoding on the evolvability of an embodied neural network. In *GECCO 2006 Workshop Proceedings, Workshop on Complexity through Development and Self-Organizing Representations (CODESOAR)*, 2006.

[9] D. B. D'Ambrosio and K. O. Stanley. A novel generative encoding for exploiting neural network sensor and output geometry. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO'07)*, 2007.

[10] P. Dürr, C. Mattiussi, and D. Floreano. Neuroevolution with analog genetic encoding. In *Proc. of the 9th International Conference on Parallel Problem Solving from Nature (PPSN IX)*, Vol. 9 of *Lecture Notes in Computer Science*, pp. 671–680, 2006.

[11] C. Mattiussi and D. Floreano. Analog genetic encoding for the evolution of circuits and networks. *IEEE Transactions on Evolutionary Computation*, Vol. 11, No. 5, pp. 596–607, 2007.

[12] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci. A hypercube-based indirect encoding for evolving large-scale neural networks. *Artificial Life journal (to appear)*, 2009.

[13] V. K. Valsalam and R. Miikkulainen. Modular neuroevolution for multilegged locomotion. In *Proc. of conference on Genetic and evolutionary computation (GECCO'08)*. ACM, 2008.

[14] E. Schlessinger, P. J. Bentley, and R. B. Lotto. Modular thinking: evolving modular neural networks for visual guidance of agents. In *Proc. of the 8th annual conference on Genetic and evolutionary computation (GECCO'06)*, pp. 215–222, New York, NY, USA, 2006. ACM Press.

[15] F. Gruau. Automatic definition of modular neural networks. *Adaptive Behaviour*, Vol. 3, No. 2, pp. 151–183, 1995.

[16] J. Reisinger, K.O. Stanley, and R. Miikkulainen. Evolving reusable neural modules. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO'04)*, pp. 69–81. Springer, 2004.

[17] J.A. Walker and J.F. Miller. Evolution and acquisition of modules in cartesian genetic programming. In *Proc. of 7th European Conference on Genetic Programming, (EuroGP'04)*, Vol. 3003 of *Lecture Notes in Computer Science*, pp. 187–197. Springer, 2004.

[18] J.A. Walker and J.F. Miller. Automatic acquisition, evolution and re-use of modules in cartesian genetic programming. *IEEE Transactions on Evolutionary Computation*, Vol. 12, No. 4, Aug 2008.

[19] M. Kirschner and J. Gerhart. Evolvability. *Proceedings of the National Academy of Sciences*, Vol. 95, No. 15, pp. 8420–8427, 1998.

[20] Rudolf A. Raff and Belinda J. Sly. Modularity and dissociation in the evolution of gene expression territories in development. *Evolution & Development*, Vol. 2, No. 2, pp. 102–113, 2000.

[21] L. Altenberg. The evolution of evolvability in genetic programming. In *Advances in genetic programming*, pp. 47–74. MIT Press, Cambridge, MA, USA, 1994.

[22] T. Smith, P. Husbands, P. Layzell, and M. O'Shea. Fitness landscapes and evolvability. *Evol. Comput.*, Vol. 10, No. 1, pp. 1–34, 2002.

[23] M. A. Bedau and N. H. Packard. Evolution of evolvability via adaptation of mutation rates. *Biosystems*, Vol. 69, pp. 143–162, 2003.

[24] G.P. Wagner and L. Altenberg. Complex adaptations and the evolution of evolvability. *EVOLUTION-LAWRENCE KANSAS-*, Vol. 50, pp. 967–976, 1996.

[25] M. Glickman and K. Sycara. Comparing mechanisms for evolving evolvability. In *Workshop Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'99)*, 1999.

[26] S. Nolfi and D. Parisi. Evolution of artificial neural networks. In *Handbook of brain theory and neural networks*, pp. 418–421. MIT Press, 2002.

[27] L. Altenberg. Genome growth and the evolution of the genotype-phenotype map. In *Lecture Notes in Computer Science*, Vol. 899, pp. 205–259. Springer-Verlag, 1995.

[28] S. Kumar. Implicit evolvability: An investigation into the evolvability of an embryogeny. In *Late Breaking Papers in the Genetic and Evolutionary Computation Conference (GECCO'00)*, Las Vegas, Nevada, USA, July 8-12 2000.

[29] E. Schlessinger, P. J. Bentley, and R. B. Lotto. Analysing the evolvability of neural network agents through structural mutations. In *Advances in Artificial Life: 8th European Conference ECAL'05 Proceedings*, pp. 5–9. Springer, 2005.

[30] E. Schlessinger, P. J. Bentley, and R. B. Lotto. Evolving visually guided agents in an ambiguous virtual world. In *Proc. of the 2005 conference on Genetic and evolutionary computation (GECCO'05)*, pp. 115–120, New York, NY, USA, 2005. ACM.

[31] L.F. Costa, F.A. Rodrigues, G. Travieso, and P.R.V. Boas. Characterization of complex networks: A survey of measurements. *Advances in Physics*, Vol. 56, No. Issue 1, pp. 167–242, January 2007.

[32] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, Vol. 298, No. 5594, pp. 824–827, 2002.

[33] D. J. Montana and L. Davis. Training feedforward neural networks using genetic algorithms. In *Proc. of the 11th International Joint Conference on Artificial Intelligence*, pp. 762–767, San Francisco, 1989. Kauffman.

[34] J. D. Schaffer, D. Whitley, and L. J. Eshelman. Combinations of genetic algorithms and neural networks: a survey of the state of the art. In *Combinations of Genetic Algorithms and Neural Networks, 1992. COGANN-92. International Workshop on*, pp. 1–37, 1992.

[35] D. Whitley, S. Dominic, R. Das, and C. W. Anderson. Genetic reinforcement learning for neurocontrol problems. *Mach. Learn.*, Vol. 13, No. 2-3, pp. 259–284, 1993.

[36] A. P. Wieland. Evolving controls for unstable systems. In *Connectionist Models: Proceedings of the 1990 Summer School*, pp. 91–102. Kauffman, 1990.

[37] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evol. Comput.*, Vol. 10, No. 2, pp. 99–127, 2002.

[38] Y. Kassahun. *Towards a Unified Approach to Learning and Adaptation.* PhD thesis, Inst. f. Informatik u. Prakt. Math. der Christian-Albrechts-Universität zu Kiel, 2006.

[39] N. T. Siebel and G. Sommer. Evolutionary reinforcement learning of artificial neural networks. *International Journal of Hybrid Intelligent Systems*, Vol. 4, No. 3, pp. 171–183, October 2007.

[40] K. O. Stanley and R. Miikkulainen. A taxonomy for artificial embryogeny. *Artif. Life*, Vol. 9, No. 2, pp. 93–130, 2003.

[41] H. Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, Vol. 4, No. 4, pp. 461–476, 1990.

[42] F. Gruau, D. Whitley, and L. Pyeatt. A comparison between cellular encoding and direct encoding for genetic neural networks. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic*

Programming 1996: Proceedings of the First Annual Conference, pp. 81–89, Stanford University, CA, USA, 1996. MIT Press.

[43] S. Luke and L. Spector. Evolving graphs and networks with edge encoding: Preliminary report. In John R. Koza, editor, *Late Breaking Papers at the Genetic Programming 1996 Conference*, pp. 117–124, Stanford University, CA, USA, July 28–31 1996. Stanford Bookstore.

[44] S. Nolfi, O. Miglino, and D. Parisi. Phenotypic plasticity in evolving neural networks. In *Proc. of the Conference From Perception to Action*, pp. 146–157, 1994.

[45] C. Mattiussi. *Evolutionary synthesis of analog networks.* PhD thesis, EPFL, Lausanne, Switzerland, 2005.

[46] C. Mattiussi, D. Marbach, P. Dürr, and D. Floreano. The age of analog networks. *AI Magazine*, Vol. 29, No. 3, pp. 63–76, 2008.

[47] P. Eggenberger. The use of the ligand-receptor concept for the evolution of a simulated foveating retina. *3D Forum: The Journal of Three Dimensional Pictures*, Vol. 15, pp. 164–169, 2001.

[48] P. Eggenberger, G. Gomez, and R. Pfeifer. Evolving the morphology of a neural network for controlling a foveating retina - and its test on a real robot. In *Proc. of The Eighth International Conference on Artificial Life*, pp. 243–251, 2002.

[49] F. Dellaert and R. D. Beer. Toward an evolvable model of development for autonomous agent synthesis. In *Artificial Life IV; Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, 1994.

[50] F. Dellaert and R. D. Beer. Co-evolving body and brain in autonomous agents using a developmental mode. Technical Report CES-94-16, Depart-

ment of Computer Engineering and Science; Case Western Reserve University, Cleveland; OH 44106, 1994.

[51] J. Bongard and C. Paul. Investigating morphological symmetry and locomotive efficiency using virtual embodied evolution. In J.-A. Meyer et al., editor, *From Animals to Animats: The Sixth International Conference on the Simulation of Adaptive Behaviour*, pp. 420–429. MIT Press, 2000.

[52] J. C. Bongard and R. Pfeifer. Repeated structure and dissociation of genotypic and phenotypic complexity in artificial ontogeny. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO'01)*, pp. 829–836, San Francisco, California, USA, 2001. Morgan Kaufmann.

[53] J. Bongard. Evolving modular genetic regulatory networks. In *Proc. of the 2002 Congress on Evolutionary Computation (CEC'02)*, pp. 1872–1877, Piscataway, NJ, 2002. IEEE Press.

[54] Y. Jin, L. Schramm, and B. Sendhoff. A gene regulatory model for the development of primitive nervous systems. In *INNS-NNN Symposia on Modeling the Brain and Nervous Systems*, Lecture Notes in Computer Science (to appear), 2008.

[55] Y. Jin. Computational modeling of gene regulatory networks: Analysis, synthesis and applications. In *Proc. of 15th International Conference on Neural Information Processing of the Asia-Pacific Neural Network Assembly (ICONIP'08)*, Lecture Notes in Computer Science (to appear), 2008.

[56] B. Jones, Y. Jin, X. Yao, and B. Sendhoff. Evolution of neural organization in a hydra-like animat. In *Proc. of 15th International Conference.on Neural Information Processing of the Asia-Pacific Neural Network Assembly (ICONIP'08)*, Lecture Notes in Computer Science (to appear), 2008.

[57] G.S. Hornby. Functional scalability through generative representations: the evolution of table designs. *Environment and Planning B: Planning and Design*, Vol. 31, No. 4, pp. 569–587, 2004.

[58] K. O. Stanley. *Efficient evolution of neural networks through complexification.* PhD thesis, The University of Texas at Austin, 2004.

[59] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proc. of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pp. 41–49, Hillsdale, NJ, USA, 1987. L. Erlbaum Associates Inc.

[60] K. O. Stanley and R Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, Vol. 21, pp. 63–100, 2004.

[61] K. O. Stanley, B. D. Bryant, and R. Miikkulainen. Real-time neuroevolution in the nero video game. *IEEE Transactions on Evolutionary Computation*, Vol. 9, No. 6, pp. 653–668, 2006.

[62] J. Reisinger, E. Bahceci, I. Karpov, and R. Miikkulainen. Coevolving strategies for general game playing. In *Computational Intelligence and Games, 2007. (CIG'07) IEEE Symposium on*, pp. 320–327, 2007.

[63] K. O. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines Special Issue on Developmental Systems*, Vol. 8, No. 2, pp. 131–162, 2007.

[64] F. Gruau. Genetic synthesis of modular neural networks. In *Proc. of the 5th International Conference on Genetic Algorithms table of contents*, pp. 318–325. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1993.

[65] F. Gruau. *Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm.* PhD thesis, Ecole Normale Superieure de Lyon, France, 1994.

[66] J. R. Koza. *Genetic Programming: on the programming of computers by means of natural selection.* MIT Press, 1992.

[67] J.R. Koza. *Genetic programming II: automatic discovery of reusable programs.* MIT Press Cambridge, MA, USA, 1994.

[68] J. F. Miller. An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach. In *Proc. of the 1999 Genetic and Evolutionary Computation Conference (GECCO'99)*, pp. pp. 1135–1142, Orlando, Florida, USA, 1999.

[69] J. F. Miller and P. Thomson. Cartesian genetic programming. In *Proc. of the 3rd European Conference on Genetic Programming*, Vol. 1802, pp. 121–132. Springer, 2000.

[70] M. Kimura. *The Neutral Theory of Molecular Evolution.* Cambridge University Press, Cambridge, 1983.

[71] J.F. Miller, D. Job, and V.K. Vassilev. Principles in the evolutionary design of digital circuits-part i. *Genetic Programming and Evolvable Machines*, Vol. 1, No. 1, pp. 7–35, 2000.

[72] J.F. Miller, D. Job, and V.K. Vassilev. Principles in the evolutionary design of digital circuits-part ii. *Genetic Programming and Evolvable Machines*, Vol. 1, No. 3, pp. 259–288, 2000.

[73] J.F. Miller. Evolving developmental programs for adaptation, morphogenesis, and self-repair. In *Proc. of ECAL*, Vol. 2801, pp. 256–265. Springer, 2003.

[74] J.F. Miller and W. Banzhaf. Evolving the program for a cell: from french flags to boolean circuits. *On Growth, Form and Computers*, pp. 278–301, 2003.

[75] J.F. Miller. Evolving a self-repairing, self-regulating, french flag organism. In *Proc. of Genetic and Evolutionary Computation (GECCO'04)*, Vol. 3102 of Lecture Notes in Computer Science, pp. 129–139. Springer, 2004.

[76] S. L. Harding and J. Miller. Evolution of robot controller using cartesian genetic programming. In M. Keijzer, A. Tettamanzi, P. Collet, J. van Hemert, and M. Tomassini, editors, *Proc. of EuroGP 2005*, Vol. 3447 of *Lecture Notes in Computer Science*, pp. 62–72, 2005.

[77] P. J. Angeline and J. Pollack. Evolutionary module acquisition. In *Proc. of the Second Annual Conference on Evolutionary Programming*, pp. 154–163. MIT Press, 1993.

[78] N. J. Radcliffe. Forma analysis and random respectful recombination. In *Proc. of the Fourth International Conference on Genetic Algorithms*, pp. 222–229. Morgan Kaufmann Publishers, 1991.

[79] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, Vol. 9, No. 2, pp. 159–195, 2001.

[80] V. H. Meisner and C. Igel. Similarities and differences between policy gradient methods and evolution strategies. In M. Verleysen, editor, *16th European Symposium on Artificial Neural Networks (ESANN'08)*, pp. 149–154, Belgium, 2008. d-side publications.

[81] A. Auger and N. Hansen. Performance evaluation of an advanced local search evolutionary algorithm. In *the IEEE Congress on Evolutionary Computation*, 2005.

[82] A. Auger and N. Hansen. A restart cma evolution strategy with increasing population size. In *the IEEE Congress on Evolutionary Computation, (CEC'05)*, 2005.

[83] C. Igel. Neuroevolution for reinforcement learning using evolution strategies. *the IEEE Congress on Evolutionary Computation, (CEC'03)*, Vol. 4, , 2003.

[84] F. Gomez, J. Schmidhuber, and R. Miikkulainen. Accelerated neural evolution through cooperatively coevolved synapses. *The Journal of Machine Learning Research*, Vol. 9, pp. 937–965, 2008.

[85] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. Syst., Man, Cybern.*, Vol. 13, pp. 834–846, 1983.

[86] J. Gauci and K. O. Stanley. A case study on the critical role of geometric regularity in machine learning. In *Proc. of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI-2008).* AAAI Press, 2008.

[87] S. Nolfi and D. Floreano. *Evolutionary Robotics. The Biology, Intelligence, and Technology of Self-organizing Machines.* MIT Press, Cambridge, MA, 2001.

[88] S. Whiteson and P. Stone. Evolutionary function approximation for reinforcement learning. *The Journal of Machine Learning Research*, Vol. 7, pp. 877–917, 2006.

[89] A. Soltoggio, P. Dürr, C. Mattiussi, and D. Floreano. Evolving neuromodulatory topologies for reinforcement learning-like problems. In *the IEEE Congress on Evolutionary Computation (CEC'07)*, 2007.

[90] S. Elfwing, E. Uchibe, K. Doya, and H.I. Christensen. Evolutionary development of hierarchical learning structures. *IEEE Transactions on Evolutionary Computations*, Vol. 11, No. 2, pp. 249–264, 2007.

[91] S. Natarajan and P. Tadepalli. Dynamic preferences in multi-criteria reinforcement learning. In *Proc. of the 22nd International Conference on Machine Learning*, pp. 601–608, 2005.

[92] 平岡和幸, 三島健稔. 荷重報酬和モデルで表されるタスク族に対する一括強化学習法. 日本神経回路学会論文誌, Vol. 13, No. 4, pp. 137–145, Dec. 2006.

[93] Z. Gabor, Z. Kamlar, and C. Szepesvari. Multi-criteria reinforcement learning. In *Proc. of International Conference on Machien Learning*, July 1998.

[94] J Karlsson. *Learning to Solve Multiple Goals*. PhD thesis, University of Rochester, 1997.

[95] N. Sprague and D. Ballard. Multiple-goal reinforcement learning with modular sarsa(0). In *Proc. of the Eighteenth International Joint Conference on Artificial Intelligence*, 2003.

[96] M. Humphrys. *Action Selection Methods Using Reinforcement Learning*. PhD thesis, University of Cambridge, 1997.

[97] B. Bhat, C. Isbell, and M. Mateas. On the difficulty of modular reinforcement learning for real world partial programming. In *Proc. of the Twenty-First National Conference on Artificial Intelligence*. AAAI, 2006.

[98] D. O. Kang and Z. Bien. Multiobjective control problems by reinforcement learning. In Jennie Si, Andrew G. Barto, Warren B. Powell, and Donald Wunsch, editors, *HANDBOOK OF LEARNING AND APPROXIMATE DYNAMIC PROGRAMMING*, chapter 17, pp. 433–461. IEEE Press, 2004.

[99] 中山弘, 谷野哲三. 多目的計画法の理論と応用. 社団法人計測自動制御学会, 1994.

[100] A. Wieland. Evolving neural network controllers for unstable systems. In *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, Vol. 2, 1991.

業績リスト

論文

- 上岡 拓未, 内部 英治, 銅谷 賢治. “Max-Min Actor Critic による複数報酬課題の強化学習”. 電子情報通信学会論文誌, Vol.J90-D, No.9, pp.2510-2521,(2007).

国際会議

- T. Kamioka, E. Uchibe and K. Doya. “NeuroEvolution based on Reusable and Hierarchical Modular Representation”. INNS-NNN Symposia (New directions in Neural Networks) 2008, pp 54-55, Auckland, New Zealand, (Nov. 2008).

研究会

- 上岡 拓未, 内部 英治, 銅谷 賢治, “複数の価値関数を用いた多目的強化学習”. 電子情報通信学会技術研究報告, vol.105, no.658, pages 127–132, (Mar. 2006).