卒業論文

# ゲーム AI と深層学習を用いた燃料配置の最適化

名古屋大学

工学部

エネルギー理工学科

山本章夫研究室

笠間 陸斗

令和 6 年 3 月

# 目次

# 第1章　序論

## 1.1　背景

2024 年現在、100 か国以上の国が 2050 年の温室効果ガス実質ゼロ（カーボンニュートラル, Carbon Neutral）の実現を宣言している[1]。国際エネルギー機関（International Energy Agency, IEA）の 2022 年の報告書[2]では、

「Less nuclear power would make net zero ambitions harder and more expensive.」

と述べられており、原子力発電の重要性は高まりを見せている。

加圧水型原子炉（Pressurized Water Reactor, PWR）原子力発電の仕組みを図 1-1 に示す。まず、原子炉格納容器内の水が原子炉圧力容器内の原子燃料から熱をうばう。その後、蒸気発生器で蒸気を発生させ、蒸気を用いてタービンを回すことで発電を行っている。

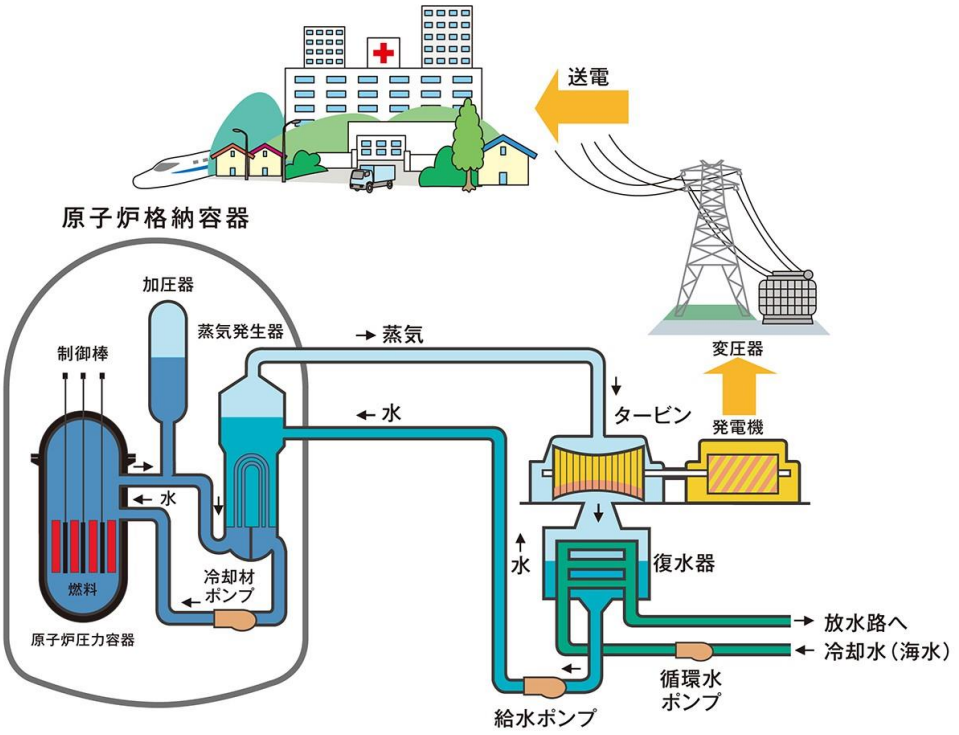
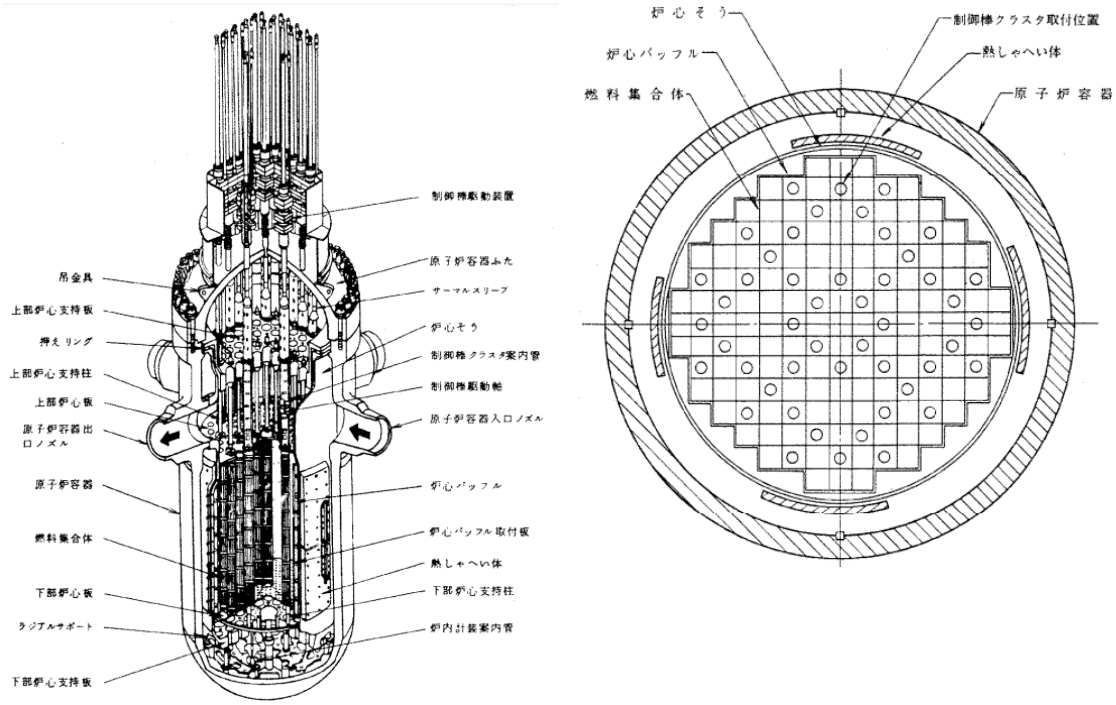



図 1-1　PWR 原子力発電の仕組み[3]

原子力発電所の稼働には、安全性を前提として経済性を高めることが必要となる。そこで、様々な最適化を行うことによって、安全性と経済性を両立した運転を目指している。例えば、原子力発電で使われる水管理(水化学)の最適化[4]や原子力発電を含めた電力構成最適化[5]などが行われている。最適化を行う際には、問題に応じて最適化の手法を選択しなければならないが、近年、コンピュータの計算速度の向上や膨大なデータを収集できるようになったことから、人工知能（Artificial Intelligence, AI）を活用した最適化の研究が盛んである。

### 1.1.1　燃料装荷パターンと炉心特性

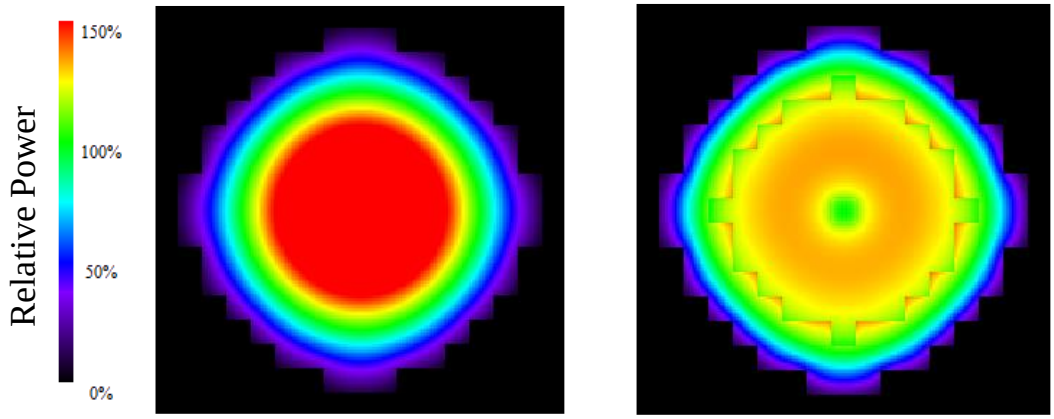
原子力分野における最適化の一つとして、炉心内の燃料集合体配置（燃料装荷パターン）の最適化（Loading Pattern Optimization, LP Optimization）がある。PWR の原子炉圧力容器内の構造を図 1-2 に示す。図 1-2(b)に示すように、PWR の炉心内には燃料集合体が配置（装荷）されている。




(a)　圧力容器の構造図　　(b)　炉心断面図

図 1-2　PWR の原子炉圧力容器構造[6]

炉心内の相対出力分布の例を図 1-3 に示す。図 1-3(a)は、炉心に装荷される燃料集合体がすべて同じ場合であり、中央付近の熱出力が高い。ゆえに、燃料集合体が高温になることで燃料損傷の可能性が高まり、安全性が失われる。しかし、特性の異なる複数の燃料集合体を用いることによって、図 1-3(b)のように炉心内の出力を平坦にすることができる。したがって、異なる種類の燃料集合体の特性を考慮しながら燃料装荷パターンを決定することが重要である。




(a)　すべて同じ燃料集合体　　(b)　異なる燃料集合体

図 1-3　燃料の種類数による相対出力の変化

（ICE[7]により作成）

PWR における燃料装荷パターン設計の安全性評価項目を表 1-1 に示す。主な安全性評価項目として、先ほど説明した熱出力に代表される熱的な制限のほかに、燃料集合体最高燃焼度、減速材温度係数、反応度停止余裕などの制限を満たす必要がある。

表 1-1　PWR における燃料装荷パターン設計の安全性評価項目[8]

| 評価項目 | 内容 |
|---|---|
| 径方向ピーキング係数 | 燃料棒一本の出力を軸方向に積分した時の最大値（炉心平均を 1 とする相対値） |
| 最大線出力密度 | 燃料棒単位長さ当たりの出力の最大値 |
| 制御棒落下時出力ピーキング係数 | 制御棒一本が落下した時の出力ピーキング係数 |
| 最大反応度添加率 | 制御棒を引き抜いた時の単位時間当たりの反応度添加量の最大値 |
| 制御棒飛出時出力ピーキング係数 | 制御棒一本が引き抜かれた時の出力ピーキング係数 |
| ドップラー係数 | 燃料温度が変化した時の炉心反応率の変化 |
| 燃料集合体最大燃焼度 | 燃料集合体の燃焼度の最大値 |
| 減速材温度係数 | 減速材の温度が変化した時の炉心反応度の変化率 |
| 反応度停止余裕 | 最大の反応度価値を持つ制御棒一体が完全に引き抜かれ、その他の制御棒が全挿入された時の未臨界度 |

また、経済性の観点では、取出燃焼度が大きな影響を与える。ここで、取出燃焼度とは、使用済みの燃料集合体の単位重さ当たりの熱エネルギー発生量のことである。取出燃焼度が高いほど、燃料集合体から発生したエネルギーが多くなるため、経済的に燃料を使用することができていることになる。また、新燃料装荷体数も経済性に影響を与える。新燃料装荷

体数とは、運転開始時に炉心内に装荷されている未燃焼の燃料集合体の数である。図 1-4 に示すように、原子燃料は多くの工程から作られるとともに、様々な輸送方法によって運ばれる。したがって、原子燃料の製造や輸送に要するコストを考慮すると、新燃料の数を減らすことは重要である。

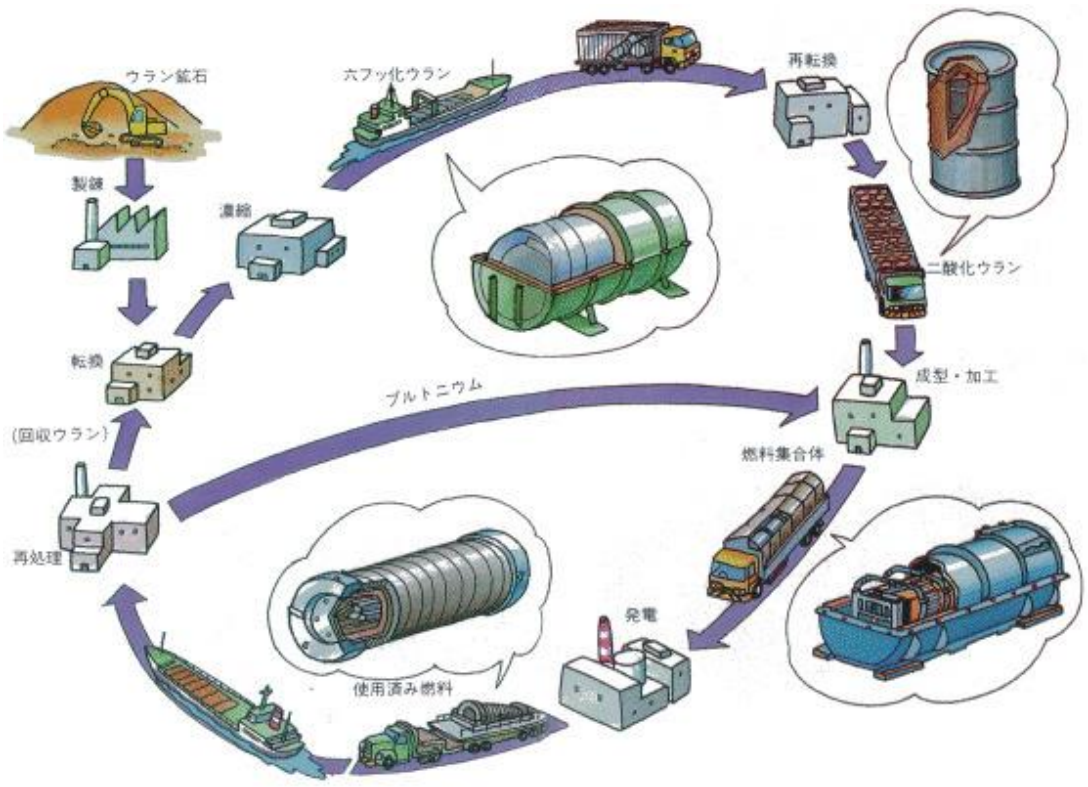



図 1-4　核燃料サイクル[9]

燃料装荷パターンの設計の難しい点は、安全性と経済性は相反するパラメータとなっていることである。経済性を上げるためには、燃焼度が高い燃料集合体を使わざるを得ない。燃焼による原子燃料の組成変化を図 1-5 に示す。燃料に含まれる $^{235}U$ は中性子を吸収し、核分裂を起こすことでエネルギーを放出する。燃焼度が高くなることにより、エネルギーの主要な発生源である $^{235}U$ の含有率が低くなるため、燃焼度が高い燃料集合体の出力は下がる。したがって、燃焼度が低い燃料集合体を用いた場合と出力を同じに保つには、ほかの燃料集合体の出力を上げなければならないが、熱的な制限値を必ず満足させる必要がある。

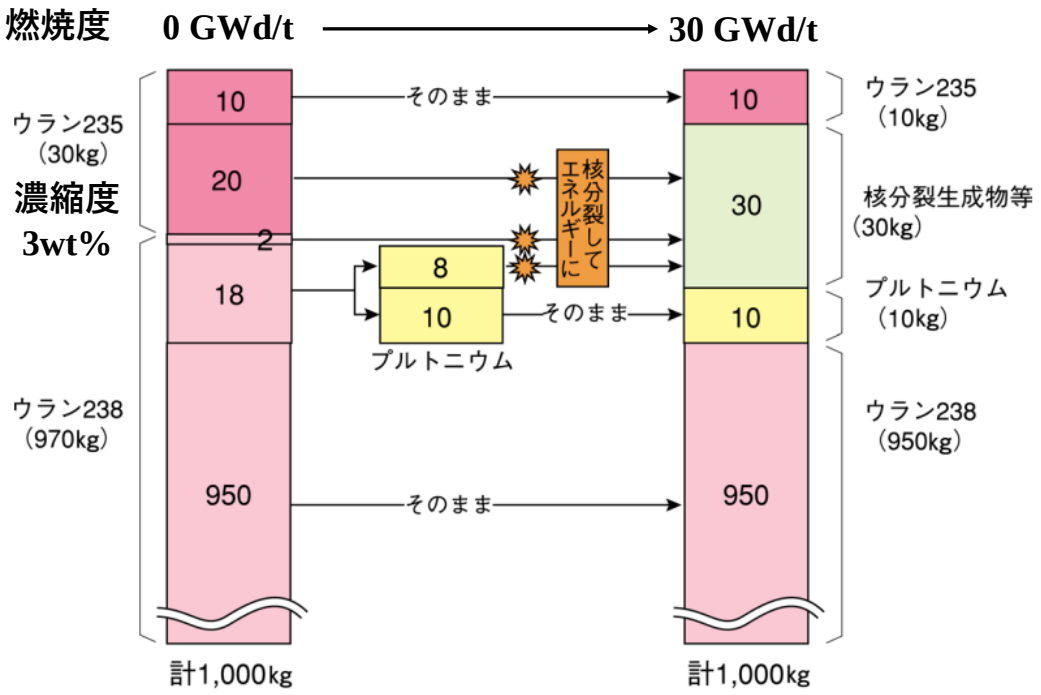



図 1-5　燃焼による原子燃料の組成変化（[10]の図を一部改訂）

現在、燃料装荷パターンの設計は技術者の経験に基づく反復施行によって行われている。このとき、技術者は以下の燃料集合体のパラメータなどを考慮している。

1. 無限増倍率（Infinite Multiplication Factor, k-infinity）：核分裂で生まれた中性子 1 個が吸収や漏えいなどで消滅した際、その中性子により核分裂で生まれる中性子数の期待値。ただし、燃料集合体が無限に装荷された条件を考慮する。核分裂の起こりやすさの指標である。
2. $^{235}$U 濃縮度（Enrichment）：燃料集合体のウラン中の $^{235}$U の含有率。単位は wt%。
3. 燃焼度（Burnup）：燃料集合体中の単位重金属（ウラン）重さ当たりの熱エネルギー発生量。単位は GWd/t。
4. 可燃性毒物（Burnable Poison）：中性子を吸収する物質。炉心全体として、運転初期は核分裂しやすい $^{235}$U が多くあるが、燃焼が進むにつれて核分裂生成物が多く生成し、また、核分裂性物質が減少することから核分裂を起こしにくくなる。このような傾向を軽減するため、可燃性毒物を燃料に添加している。可燃性毒物を含む燃料集合体の燃焼度と無限増倍率の関係を図 1-6 に示す。運転初期は、ガドリニウム（Gd）、エルビウム（Er）などの可燃性毒物が中性子を吸収するため、増倍率が抑えられる。燃焼が進むにつれて、可燃性毒物が中性子を吸収し、別の物質に変化するため中性子吸収が減少し、増倍率が高くなる。

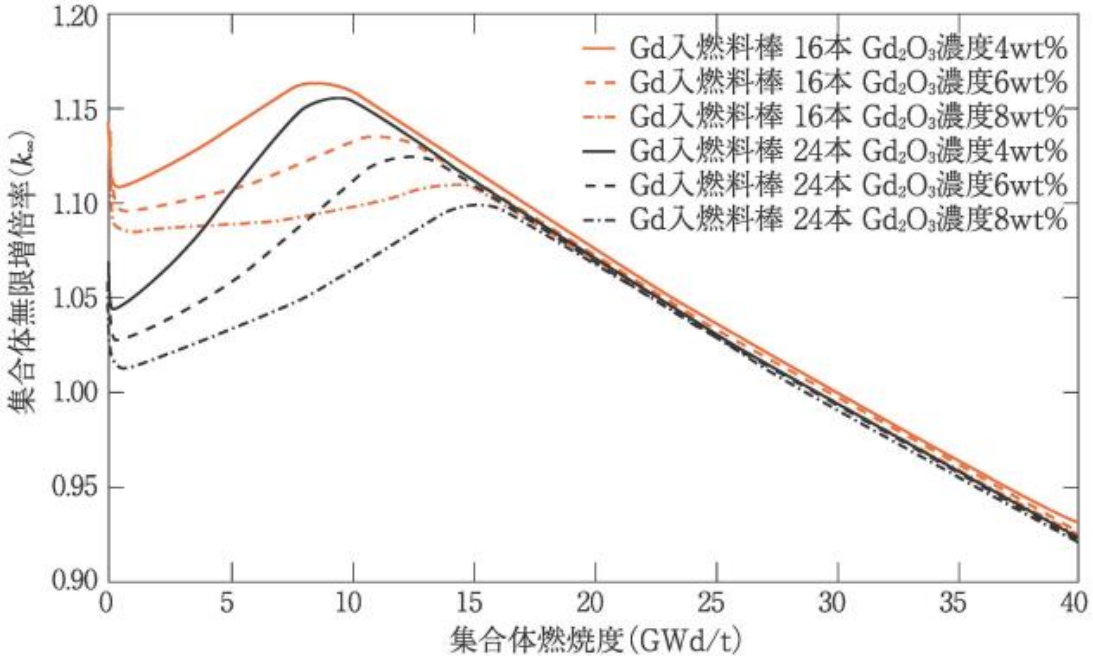



図 1-6　濃縮度 4.1wt%の可燃性毒物入り燃料集合体における無限増倍率[8]

このようなパラメータを考慮しながら燃料装荷パターンの設計を行うが、膨大な数の燃料装荷パターンの中から最適な燃料装荷パターンを決定する必要がある。首都圏全体の電力需要の 2%程度を賄うことができる電気出力 900 MW 級（文献[11]の記述内容「200 万 kW とは首都圏全体の電力需要の 5%程度」から概算）の PWR 炉心内には 157 体の燃料集合体が装荷されている。装荷体数が 157 体の燃料装荷パターンの数は最大で$157! \approx 10^{278}$となり、安全性を満足しつつ経済性が高い最適な燃料装荷パターンを探し出すことは極めて困難である。

さらに、燃料装荷パターンの設計は限られた時間の中で実施する。一般的な運転サイクルを図 1-7 に示す。燃料装荷パターンの詳細な設計は、定期検査で取り出された燃料集合体の検査後に行われている。また、実際の定期検査の流れを図 1-8 に示す。燃料集合体の取り出し終了から燃料装荷開始まで 3 週間ほどであり、燃料集合体の状態確認や安全な燃料装荷手順の決定を行うため、燃料装荷パターンを設計する時間はさらに短くなる。この短い期間で、事業者のニーズに沿うような安全性と経済性を兼ね備えた燃料装荷パターンを複数通り作成する必要がある。

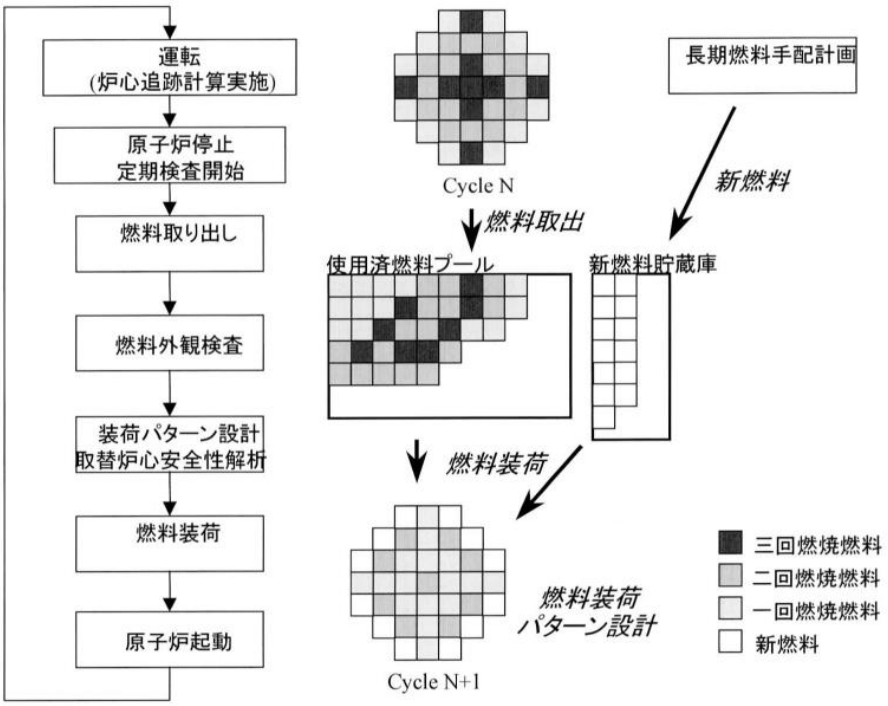



図 1-7　PWR の運転サイクル[12]

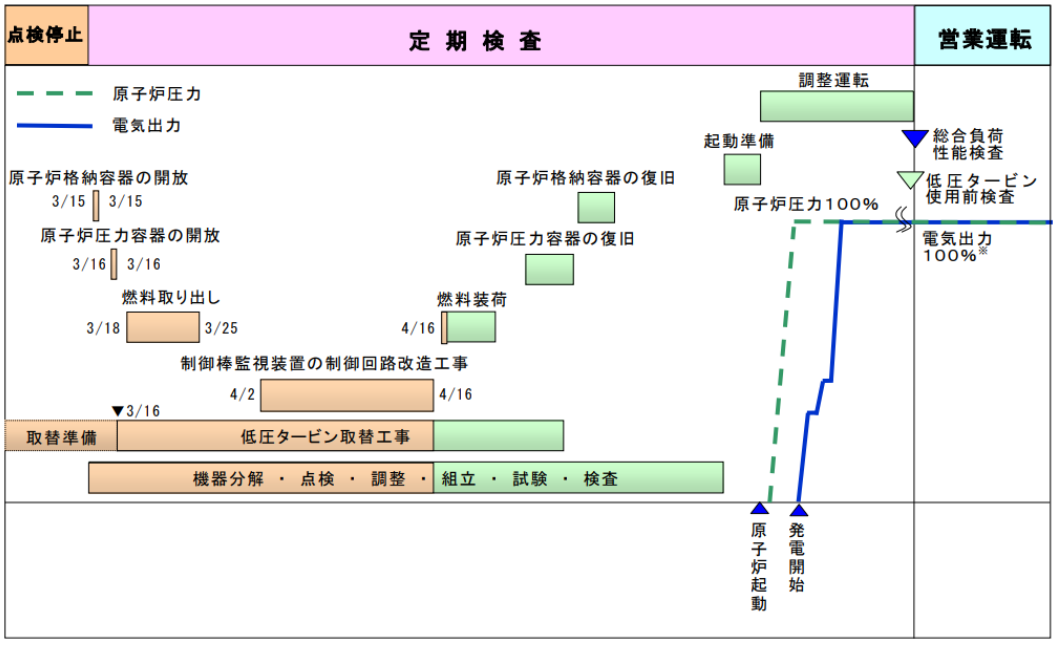



図 1-8　浜岡 5 号機第 4 回定期検査（2010 年）の工程[13]

以上のように、燃料装荷パターンの設計は、安全性と経済性が相反するパラメータであることや、設計を行うための時間的な限られているため非常に難しい問題である。良い燃料装荷パターンの設計を行うためには、自動最適化などの手法を検討する必要がある。

### 1.1.2　組合せ最適化問題としての燃料装荷パターン最適化

人による燃料装荷パターン設計の限界を超えるために、コンピュータによる自動燃料装荷パターン設計を行う研究が行われてきた。コンピュータによる自動設計では、燃料装荷パターン設計を組合せ最適化問題とみなしている。

まず、最適化とは「与えられた関数を、与えられた条件下で、最小化または最大化すること」である[14]。ここで、与えられた関数を目的関数（Objective Function）、与えられた条件を表す関数を制約関数（Constraint Function）という。最適化問題と最適化手法の例として、式(1.1)で表される関数$l$の最小化を説明する。

$$l(x,y) = (x-1)^2 + (y-1)^2, (x,y \in \mathbb{R}) \tag{1.1}$$

$x,y$に対する式(1.1)の値を図 1-9 に示す。$x=1, y=1$とすることで関数$l$の最小化が達成されるため、$x=1, y=1$がこの問題の最適解となる。

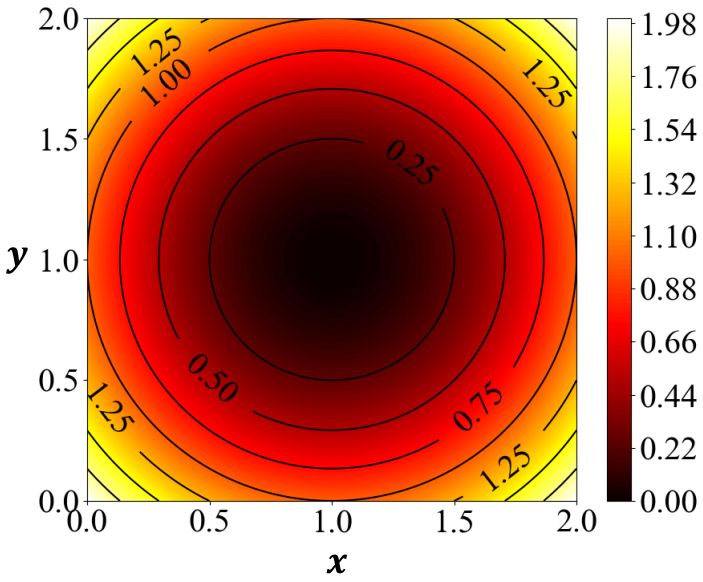



図 1-9　$x,y$に対する式(1.1)の値

この最適化問題の解法として、勾配（Gradient）を用いた手法（勾配降下法）がある。勾配とは、各点において変化が最も激しく、かつ、値が最も増加する方向と値である[15]。各点の勾配（逆方向）を図 1-10 に示す。任意の点$\boldsymbol{p}$に対して勾配を取得し、式(1.2)のように点$\boldsymbol{q}$を移動させる操作を繰り返すことによって、徐々に最適解に近づくことができる。

$$\boldsymbol{q} = \boldsymbol{p} - \alpha \cdot \mathrm{grad}\, \boldsymbol{p} \tag{1.2}$$

ここで、$\alpha$は点の変化量を決める正の定数である。$\alpha$が大きくなるほど、点の移動する距離が大きくなる。

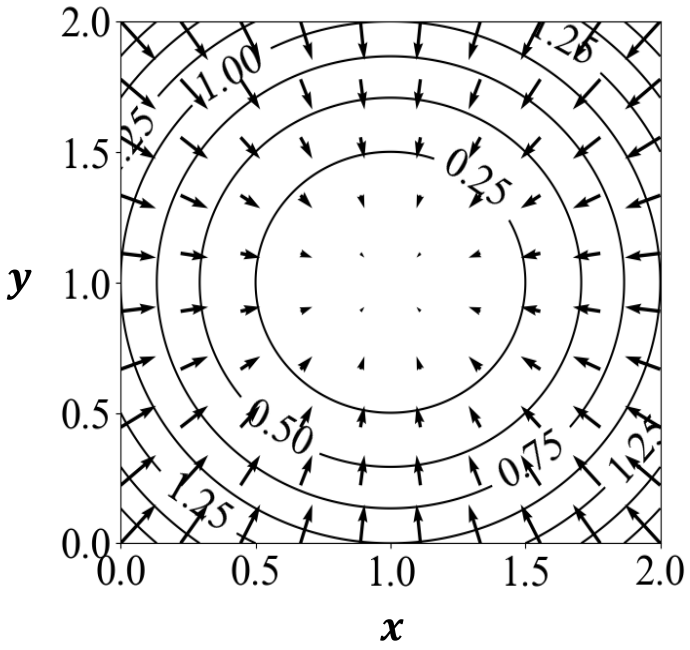



図 1-10　式(1.1)における各点の勾配（ただし、逆方向）
勾配は矢印で表示

また、組合せ最適化とは、変数が離散的な値をとる最適化問題である[14]。組合せ最適化はグラフ理論、アルゴリズム理論、計算理論と関連している。組合せ最適化問題の例として、最短経路問題を説明する。図 1-11 が表しているのは、都市 A–I と都市間の所要時間である。都市 A から出発し、都市 I に到着する時間を最短にすることを考える。この問題は、ダイクストラ法[16]などのグラフ理論に基づくアルゴリズムを用いて、図 1-12 の解を導くことができる。最短経路問題や燃料装荷パターン最適化では、都市や燃料集合体の数が増えると指数関数的に組合せが増加（組合せ爆発）するため、計算時間が膨大となる。そこで、時間削減のために、厳密な最適解を与えるわけではないが、良好な解を与える手法（ヒューリスティック手法）が使われている。

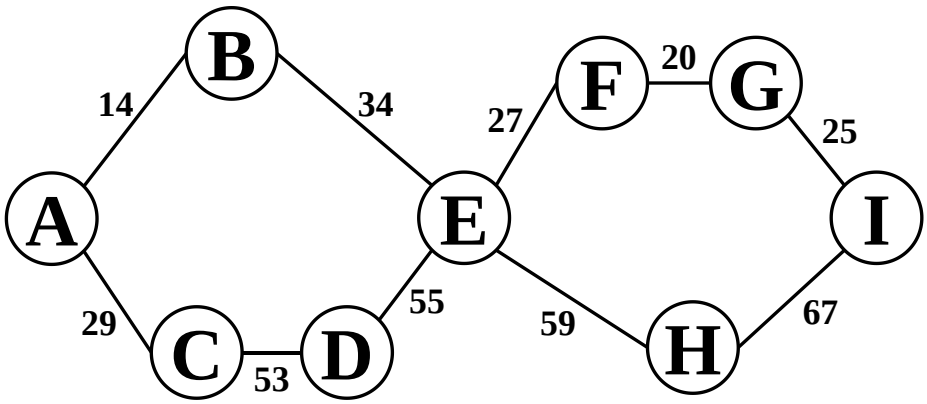



図 1-11　都市と都市間の所要時間

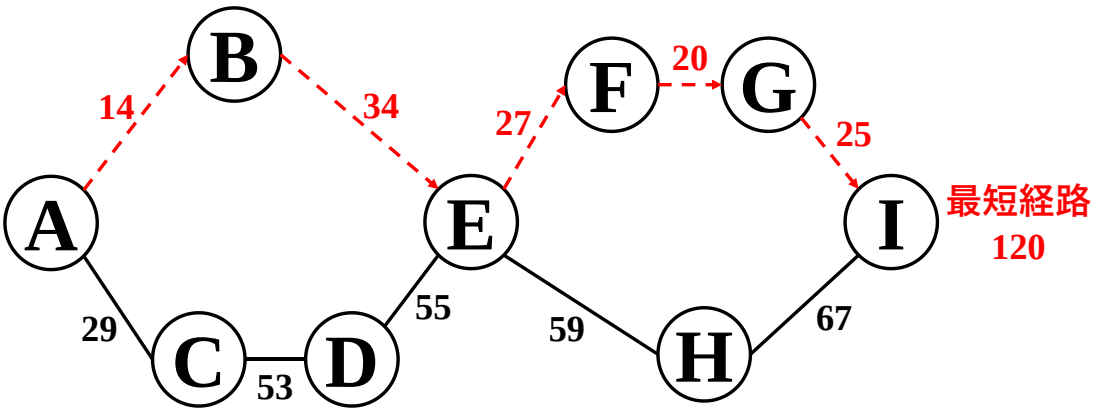



図 1-12　最短経路

燃料装荷パターン最適化は、燃料装荷パターンが離散的な値をとるため組合せ最適化問題である。また、燃料装荷パターン最適化の目的関数は炉心特性によって定義される。目的関数の例を式(1.3)に示す[17]。

$$\text{Objective Function} = 20 \times PPF_{\text{norm}} - CycleLenth_{\text{norm}} + MaxBurnup_{\text{norm}}$$

$$PPF_{\text{norm}} = \begin{cases} 1 + \dfrac{1}{PPF} \times (PPF - 1.4)^2 & (PPF > 1.4) \\ 0 & (PPF \leq 1.4) \end{cases}$$

$$CycleLenth_{\text{norm}} = \frac{1}{20} \times CycleLenth \tag{1.3}$$

$$MaxBurnup_{\text{norm}} = \begin{cases} 1 + \dfrac{1}{MaxBurnup} \times (MaxBurnup - 48)^2 & (MaxBurnup > 48) \\ 0 & (MaxBurnup \leq 48) \end{cases}$$

ここで、$PPF$は径方向ピーキング係数、$CycleLenth$は炉心の増倍率が 1 になるまでの炉心の平均燃焼度（サイクル長）[GWd/t]、$MaxBurnup$はサイクル終了時の最大の取出燃焼度[GWd/t]である。式(1.3)で定義される目的関数は、値が小さいほど良い燃料装荷パターンである。このような目的関数を最小化もしくは最大化するために、古典的な最適化手法やヒューリスティック手法、AI の一分野である機械学習（Machine Learning, ML）を用いた手法が開発されてきた。

燃料装荷パターン最適化のための自動最適化手法の歴史を図 1-13 に示す。1990 年以前は、動的計画法（Dynamic Programming）[18]、線形計画法（Linear Programming）[19]、変分法（Calculus of Variations）[20]などの古典的な最適化手法が用いられた。1990 年以降は、焼きなまし法（Simulated Annealing）[21]、遺伝的アルゴリズム（Genetic Algorithm）[22], [23]、蟻コロニー最適化（Ant Colony Optimization）[24]などのヒューリスティック手法の適用が研究された。2020 年以降は、計算速度やアルゴリズムの向上によって、ML とヒューリスティック手法を用いた手法[25], [26]が開発されている。これらの従来の手法を用いると、数千回～数万回の燃料装荷パターンの生成で、膨大な数の燃料装荷パターンから良好な燃料装荷パターンを見つけることが可能である。代表的な燃料装荷パターン最適化手法については第 2 章で最適化の詳細を述べる。

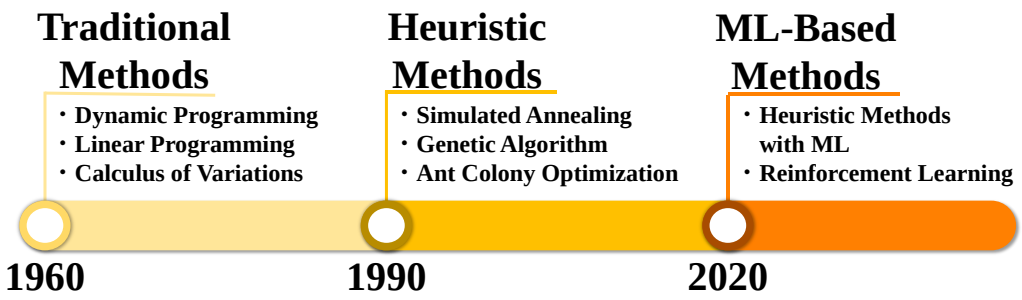



図 1-13　燃料装荷パターン最適化手法の歴史

燃料装荷パターン最適化の目的関数を得るためには、炉心計算によって炉心特性（径方向ピーキング係数、サイクル長など）を求めなければならない。詳細な炉心特性を計算するためには、1 つの燃料装荷パターンあたり数分～数十分の時間を要する。従来の手法では数千回～数万回の炉心計算が必要であり、さらに、限られた日程内で事業者のニーズに合った燃料装荷パターンを複数生成する必要があるため、時間的な制約が問題となる。したがって、現在の燃料装荷パターン最適化の課題は、少ない炉心計算回数で良好なパターンを得る手法の開発である。

### 1.1.3　ゲーム AI の躍進と最適化への応用

2020 年以降、AI を用いた燃料装荷パターン自動最適化手法が研究されていることを 1.1.2 項で述べたが、AI が革新を起こした分野のひとつとして、ボードゲームが挙げられる。囲碁の世界では、2015 年に AlphaGo[27]と呼ばれる囲碁 AI が世界のトッププレイヤーに初めて勝利した。さらに、2018 年には、AlphaGo をベースとした汎用型ゲーム AI である AlphaZero[28]が、囲碁だけでなく将棋とチェスでも世界のトップとなった。この躍進の要因は、AlphaGo と AlphaZero がモンテカルロ木探索（Monte Carlo Tree Search, MCTS）[29]と深層学習（Deep Learning, DL）[30]を組み合わせた手法を用いたことである。

さらに、ゲーム AI で革命を起こした AlphaGo と AlphaZero に触発されて、MCTS と DL に基づくアプローチが多くの分野で活用されている。分子合成化学の分野では、所望の有機分子を得るための合成手順を決定する問題に対して、MCTS と DL を用いた手法が従来のコンピュータを使った手法よりも 30 倍速く問題を解決した[31]。さらに、最適化の分野では、MCTS と DL に基づくアプローチは最適化問題に適用することができるとともに、従来手法を用いるよりも良い結果が得られている。例えば、流体トポロジー（形状）最適化[32]や組合せ最適化の代表的な問題である巡回セールスマン問題[33]などで適用されている。

## 1.2　本研究の目的

本研究の目的は、現在の燃料装荷パターンの課題を解決すること、つまり、少ない燃料装荷パターンの炉心特性計算回数で良好な解を与える手法の開発である。この目的を達成するために、MCTS と DL のアプローチを用いた燃料装荷パターン最適化手法を検討する。この手法に注目した理由は、ゲーム AI で活用されている MCTS と DL に基づくアプローチは最適化問題に応用できるとともに、他の最適化問題において従来手法を用いるよりも効率的に問題を解決できるという事実があるためである。具体的には、燃料装荷パターン最適化に対して以下の 3 つを新たに行う。

1. グラフ理論のアプローチによって、燃料装荷パターン最適化を MCTS が適用できる問題に帰着
2. MCTS を燃料装荷パターン最適化に適用
3. DL を MCTS と組み合わせた手法を燃料装荷パターン最適化に適用

## 1.3　本論文の構成

本論文は本章を含めた 6 つの章と 1 つの Appendix から成る。

本章では、燃料装荷パターン最適化の背景、及び、本研究の目的を述べた。1.1 節では、燃料装荷パターンの難しさと燃料装荷パターンが組合せ最適化問題であることを説明した後、モンテカルロ木探索と深層学習を組み合わせた手法について言及した。1.2 節では、少ない燃料装荷パターンの炉心特性計算回数で良好な解を与える手法の開発が本研究の目的であることを示し、この目的を達成するために、モンテカルロ木探索と深層学習のアプローチを燃料装荷パターンに適用することを述べた。

第 2 章では、燃料装荷パターン最適化で用いられる従来の代表的なヒューリスティック手法と機械学習を用いた手法を解説する。2.2 節ではヒューリスティック手法と燃料装荷パターン最適化の関係を述べる。2.3 節では焼きなまし法、2.4 節では遺伝的アルゴリズム、2.5 節では蟻コロニー最適化、2.6 節では機械学習を用いた手法を解説する。

第 3 章では、グラフ理論における木と木探索を説明した後、囲碁や将棋などのゲーム AI で使用される木探索手法であるモンテカルロ木探索を説明する。3.2 節では、木の理解に必要なグラフ理論の用語の説明を行い、木の定義を述べる。3.3 節では、木探索の定義を述べるとともに、三目並べを例に木探索を説明する。3.4 節では、モンテカルロ木探索の背景にある原始モンテカルロ木探索、3.5 節でモンテカルロ木探索の説明をする。

第 4 章では、燃料装荷パターン最適化にモンテカルロ木探索を適用する方法と適用結果について述べる。また、適用した結果を従来手法と比較し、モンテカルロ木探索による最適化の特徴を議論する。4.2 節ではグラフ理論のアプローチを用いて燃料装荷パターン最適化をゲーム木探索に帰着させる方法を説明する。4.3 節と 4.4 節でそれぞれ計算条件と計算結果について示し、4.5 節では、従来手法の計算結果と比較することで、モンテカルロ木探索を用いた最適化の特徴を議論する。

第 5 章では、モンテカルロ木探索に対して深層学習を組み合わせることで、モンテカルロ木探索の性能を向上する検討を行い、燃料装荷パターン最適化に適用した結果を述べる。5.2 節では深層学習をモンテカルロ木探索に組み合わせる方法を説明する。5.3 節と 5.4 節では、計算条件と計算結果をそれぞれ示す。5.5 節ではモンテカルロ木探索と深層学習を組み合わせた手法による結果から、燃料装荷パターンへの深層学習の適用可能性と検討手法による最適化の特徴を議論する。

第 6 章では、本研究のまとめと今後の課題を述べる。6.1 節では本研究の目的と結果を述べ、各章のまとめを行う。6.2 節ではモンテカルロ木探索と深層学習による手法の課題を述べる。

Appendix A では、最適化手法のパラメータについて感度解析した結果を記載する。

## 1.4 参考文献


[1] 資源エネルギー庁, “令和 3 年度エネルギーに関する年次報告（エネルギー白書 2022）”, Accessed: Jan. 29, 2024. [Online], https://www.enecho.meti.go.jp/about/whitepaper/2022/pdf/.

[2] International Energy Agency, “Nuclear Power and Secure Energy Transitions”, Accessed: Jan. 29, 2024. [Online], https://www.iea.org/reports/nuclear-power-and-secure-energy-transitions.

[3] 日本原子力文化財団, “エネ百科 原子力・エネルギー図面集”, https://www.ene100.jp/zumen (accessed: Jan. 29, 2024).

[4] 日本原子力学会, *原子炉水化学ハンドブック*, 東京, コロナ社, 2000.

[5] 七原俊也, “原子力の増大と電源の多様化から見た電源の最適構成,” *電中研レビュー*, no. 13, pp. 9–28, Apr. 1986.

[6] 関西電力株式会社, “高浜発電所 3・4 号炉 発電用原子炉設置許可申請書 完本版（令和 4 年 6 月）,” https://www.kepco.co.jp/energy_supply/energy/nuclear_power/info/knic/library/kyonin/gensiro.html (accessed: Jan. 29, 2024).

[7] 遠藤 知弘, 山本章夫, “各特性計算コード ICE,” https://www.fermi.energy.nagoya-u.ac.jp/ICE.html (accessed Jan. 22, 2024).

[8] 丸山 博見, 岩崎 智彦, 山本 章夫, 中島 健, 岡嶋 成晃, 熊谷 明, *原子炉物理*, 東京, 日本原子力学会, 2020.

[9] 核不拡散・核セキュリティ総合支援センター, “核物質の輸送,” https://www.jaea.go.jp/04/iscn/archive/trans_is/index.html (accessed Jan. 22, 2024).

[10] 鈴木篤之, *原子力の燃料サイクル*, 東京, エネルギーフォーラム, 1985.

[11] 東京電力ホールディングス, “【火力発電所に潜入！】電気はこうして作られる,” https://www.tepco.co.jp/toudenhou/fp/1299810_9045.html (accessed Jan. 22, 2024).

[12] 山本章夫, 蛇川 季嗣, 左藤 大介, 佐藤 仁, 山崎 正俊, “燃料配置を決定せよ！：実機軽水炉における燃料配置の最適化技術”, *日本原子力学会誌*, vol. 48, no. 12, pp. 924–948, Apr. 2006.

[13] 中部電力, “5 号機 4 回定期検査の工程”, https://www.chuden.co.jp/energy/nuclear/hamaoka/hama_info/hama_info_detail/__icsFiles/afieldfile/2020/02/14/220416koutei5u.pdf (accessed Jan. 22, 2024).

[14] 遠藤靖典, 宮本定明, *最適化の基礎*, 東京, コロナ社, 2018.

[15] 筧三郎, 米田元, *理工系基礎 ベクトル解析*, 東京, サイエンス社, 2018.

[16] E. W. Dijkstra, “A Note on Two Problems in Connexion with Graphs,” *Numer. Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959, doi: 10.1007/BF01386390.

[17] S. Ishiguro, T. Endo, A. Yamamoto, “Loading pattern optimization for a PWR using Multi-Swarm Moth Flame Optimization Method with Predator,” *J. Nucl. Sci. Technol.*, vol. 57, no. 5, pp. 523–536, May 2020, doi: 10.1080/00223131.2019.1700844.

[18] I. Wall and H. Fenech, “The Application of Dynamic Programing to Fuel Management Optimization,” *Nucl. Sci. Eng.*, vol. 22, no. 3, pp. 285–297, Jul. 1965, doi: 10.13182/NSE65-A20933.

[19] T. O. Sauar, “Application of Linear Programming to In-Core Fuel Management Optimization in Light Water Reactors,” *Nucl. Sci. Eng.*, vol. 46, no. 2, pp. 274–283, Nov. 1971, doi: 10.13182/NSE71-A22361.

[20] W. B. Terney and E. A. Williamson Jr., “The Design of Reload Cores Using Optimal Control Theory,” *Nucl. Sci. Eng.*, vol. 82, no. 3, pp. 260–288, Dec. 1982, doi: 10.13182/NSE82-4.

[21] G. T. Parks, “An Intelligent Stochastic Optimization Routine for Nuclear Fuel Cycle Design,” *Nucl. Technol.*, vol. 89, no. 2, pp. 233–246, Feb. 1990, doi: 10.13182/NT90-A34350.

[22] G. T. Parks, “Multiobjective Pressurized Water Reactor Reload Core Design by Nondominated Genetic Algorithm Search,” *Nucl. Sci. Eng*., vol. 124, no. 1, pp. 178–187. Sep. 1996, doi: 10.13182/NSE96-A24233.

[23] A. Yamamoto, “A quantitative Comparison of Loading Pattern Optimization Methods for In-Core Fuel Management of PWR,” *J. Nucl. Sci. Technol.*, vol. 34, no. 4, pp. 339–347, 1997, doi: 10.1080/18811248.1997.9733673.

[24] L. Machado and R. Schirru, “The Ant-Q algorithm Applied to the Nuclear Reload Problem,” *Ann. Nucl. Energy*, vol. 29, no. 12, pp. 1455–1470, Aug. 2002, doi: 10.1016/S0306-4549(01)00118-9.

[25] A. Naserbegi, M. Aghaie, and S. M. Mahmoudi, “PWR Core Pattern Optimization Using Grey Wolf Algorithm Based on Artificial Neural Network,” *Prog. Nucl. Energy*, vol. 129, p. 103505, Nov. 2020, doi: 10.1016/j.pnucene.2020.103505.

[26] C. Wan, K. Lei, and Y. Li, “Optimization Method of Fuel-Reloading Pattern for PWR Based on the Improved Convolutional Neural Network and Genetic Algorithm,” *Ann. Nucl. Energy*, vol. 171, p. 109028, Jun. 2022, doi: 10.1016/j.anucene.2022.109028.

[27] D. Silver *et al.*, “Mastering the Game of Go with Deep Neural Networks and Tree Search,” *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016, doi: 10.1038/nature16961.

[28] D. Silver *et al.*, “A General Reinforcement Learning Algorithm that Masters Chess, Shogi, and Go through Self-Play,” *Science*, vol. 362, no. 6419, pp. 1140–1144, Dec. 2018, doi: 10.1126/science.aar6404.

[29] R. Coulom, “Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search,” in *Computers and Games*, vol. 4630, H. J. Van Den Herik, P. Ciancarini, and H. H. L. M. Donkers, Eds., in Lecture Notes in Computer Science, vol. 4630, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 72–83. doi: 10.1007/978-3-540-75538-8_7.

[30] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A Fast Learning Algorithm for Deep Belief Nets,” *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006, doi: 10.1162/neco.2006.18.7.1527.

[31] M. H. S. Segler, M. Preuss, and M. P. Waller, “Planning Chemical Syntheses with Deep Neural Networks and Symbolic AI,” *Nature*, vol. 555, no. 7698, Art. no. 7698, Mar. 2018, doi: 10.1038/nature25978.

[32] A. Gaymann and F. Montomoli, “Deep Neural Network and Monte Carlo Tree Search Applied to Fluid-Structure Topology Optimization,” *Sci. Rep.*, vol. 9, no. 1, p. 15916, Nov. 2019, doi: 10.1038/s41598-019-51111-1.

[33] Z. Xing and S. Tu, “A Graph Neural Network Assisted Monte Carlo Tree Search Approach to Traveling Salesman Problem,” *IEEE Access*, vol. 8, pp. 108418–108428, Jun. 2020, doi: 10.1109/ACCESS.2020.3000236.

# 第2章　代表的な最適化手法

## 2.1　本章の概要

1.1.2 項では、燃料装荷パターン最適化が組合せ最適化問題になることを述べた。本章では、組合せ最適化問題で多用されるヒューリスティック手法の説明を行うとともに、機械学習を用いた燃料装荷パターン最適化について言及する。

2.2 節では、ヒューリスティック手法の説明を行う。

2.3 節、2.4 節、2.5 節では、ヒューリスティック手法の具体例として、焼きなまし法、遺伝的アルゴリズム、蟻コロニー最適化を用いた燃料装荷パターン最適化をそれぞれ解説する。

2.6 節では、ヒューリスティック手法と機械学習を用いた燃料装荷パターン最適化について言及する。

2.7 節では、本章のまとめを述べる。

## 2.2　ヒューリスティック手法と燃料装荷パターン最適化

ヒューリスティック手法（発見的手法, Heuristic Methods）とは、問題に固有の知識や経験的な知識を適用し、厳密な最適解ではないが、現実的な時間内で実用上に満足のいく近似解を求める手法である[1]。したがって、問題のサイズが大きくなるにつれて実現可能な解が爆発的に増える問題に対して、実用的な解を求めるためにヒューリスティック手法が有効である。問題に固有の知識や経験的な知識を獲得するために、物理現象、生物の進化過程、生物の集団行動などを模擬したヒューリスティック手法が開発されている。ヒューリスティック手法による解の探索過程を図 2-1 に示す。新しい解の探索と解の目的関数の評価に基づいて、問題に対する知識を更新し、良好な解の探索を行う。ここで、ヒューリスティック手法の適用には、解と目的関数の値に相関が不可欠である。これは、解と目的関数の値に相関がない場合、正確な知識や有益な経験を得ることができないためである。

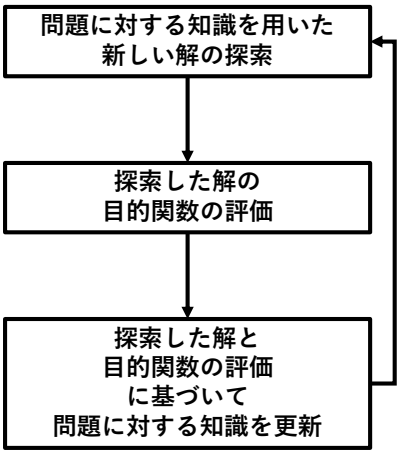



図 2-1　ヒューリスティック手法による解の探索過程

ゆえに、ヒューリスティック手法を燃料装荷パターン最適化に適用する場合に重要なことは、燃料装荷パターンと炉心特性で定義される目的関数の値の間に相関があるかということである。一例として、目的関数を式(2.1)としたとき、燃料装荷パターンと目的関数の値の関係を図 2-2 に示す。

$$\text{Objective Function} = CycleLength - 100 \times \max(0, PPF - 1.435) \qquad (2.1)$$

ここで、$CycleLength$は原子炉の運転サイクル長を、$PPF$は径方向ピーキング係数を示す。これらのパラメータの詳細については、1.1 節で説明を行った。図 2-2 の縦軸は目的関数の値、横軸は値が近いほど燃料配置が似ている燃料装荷パターンであることを示している。目的関数の形状としては細かい凹凸はあるものの、近しい燃料装荷パターンであれば目的関数は大きく変化しない。したがって、燃料装荷パターンと目的関数の値は相関を持つため、ヒューリスティック手法の燃料装荷パターン最適化への適用が可能であるといえる。

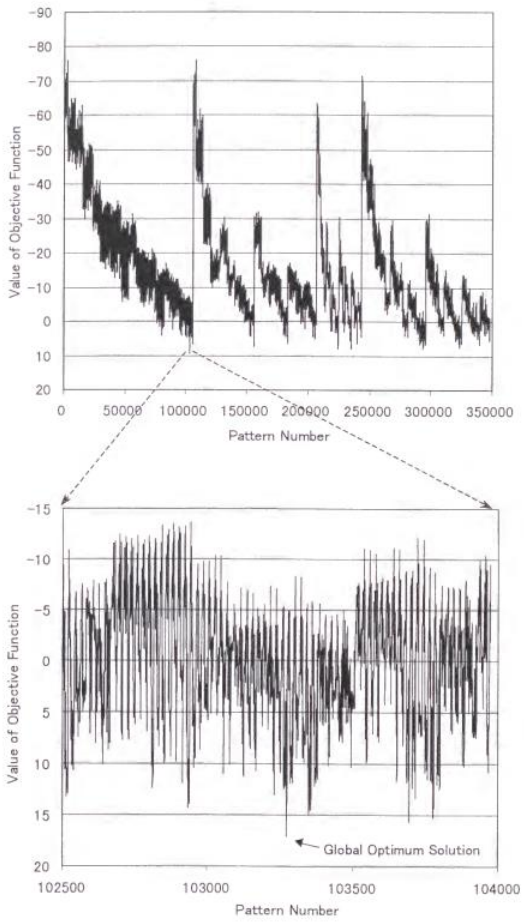



図 2-2　燃料装荷パターンと目的関数の値の関係[2]

続く 2.3 節、2.4 節、2.5 節では、ヒューリスティック手法の例として、結晶の生成過程を模倣した焼きなまし法、生物の進化過程を模倣した遺伝的アルゴリズム、蟻の集団行動を模倣した蟻コロニー最適化について、それぞれ説明する。2.3 節、2.4 節、2.5 節では目的関数の値の最小化を目的としている。

## 2.3　焼きなまし法（SA）

本節では、焼きなまし法の説明を行う。

### 2.3.1　結晶の生成過程

焼きなまし法（Simulated Annealing, SA）[3]–[6]は、結晶の生成過程を探索の戦略に使用したアルゴリズムである。結晶の生成過程と焼きなまし法の探索戦略を図 2-3 に示す。高温の物質を徐々に冷却することにより結晶を生成する過程を、焼きなまし法では初期解を徐々に良質な近似解に近づける過程に対応させている。結晶の生成過程では、金属を加熱することで金属中の原子や分子の構造が乱れるが、徐々に冷却するにつれて分子や原子がエネルギー的に低い状態に再配列することで結晶が生成される。SA による探索では、探索初期に温度が高い状態を模擬し、解空間を幅広く探索する。温度を下げることで探索する幅が狭くなり、良好な解の近傍解を緻密に探索することが可能となる。

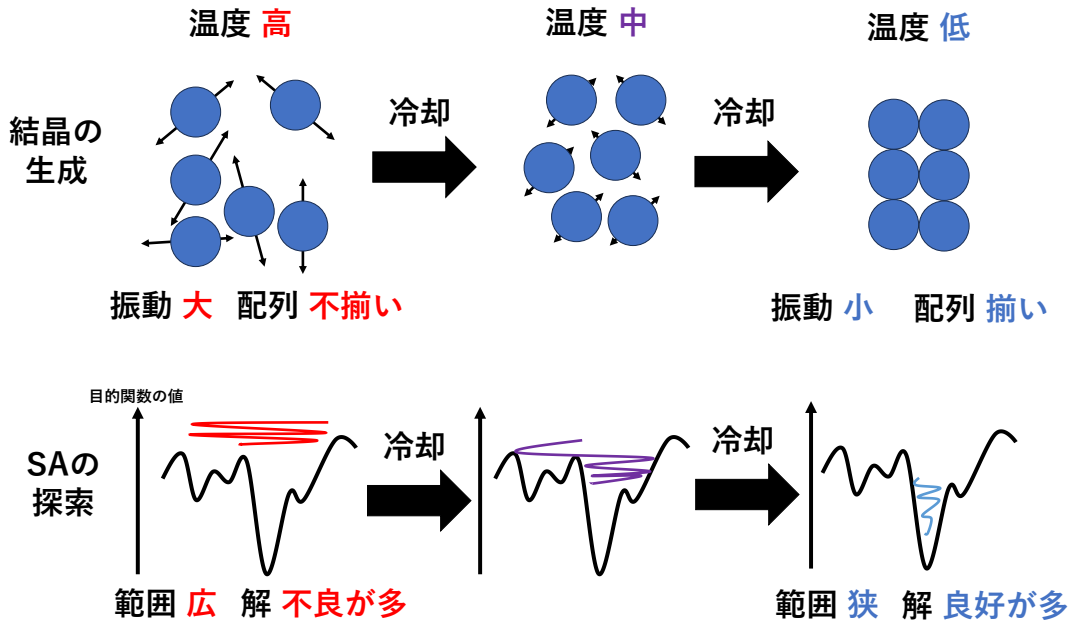



図 2-3　結晶の生成過程と焼きなまし法の探索戦略

目的関数の最小化を目的とした SA の探索過程を定式化すると式(2.2)になる。

$$P(x_i, x_j, T_{j-1}) = \begin{cases} 1 & f(x_j) \le f(x_i), i < j \\ \exp\left(-\dfrac{f(x_j) - f(x_i)}{T_{j-1}}\right) & f(x_j) > f(x_i), i < j \end{cases} \quad (2.2)$$

ここで、$P(x_i, x_j, T_{j-1})$は温度パラメータ$T_{j-1}$の時、目的関数の値$f(x_i)$を持つ解$x_i$から目的関数の値$f(x_j)$を持つ解$x_j$への遷移を受理する確率である。また、$x_i$は$i$個目に生成された解を表し、$T_{j-1}$は解$x_j$生成直前の体系温度を表す。目的関数の値$f$が小さいほど、良好な燃料装荷パターンである。式(2.2)は、目的関数の値が好転する場合は必ず遷移を行い、目的関数の値が悪化する場合は決められた確率で遷移が行われることを示している。目的関数の値が悪化する場合の遷移は、温度パラメータ$T$が大きい、または、目的関数の値の変化量が小さいほど起こりやすくなる。解が改悪する方向にも遷移するため、局所最適解に陥ることが少ないことが SA の利点である。SA では解の近傍と温度の冷却スケジュールを決定する必要があり、冷却スケジュールは探索の戦略に直結するため特に重要である。

### 2.3.2 SA による燃料装荷パターン最適化

SA による燃料装荷パターン最適化のフローチャートを図 2-4 に示す。また、最適化の過程を図 2-5 に示す。SA による最適化の流れは以下の通りである。

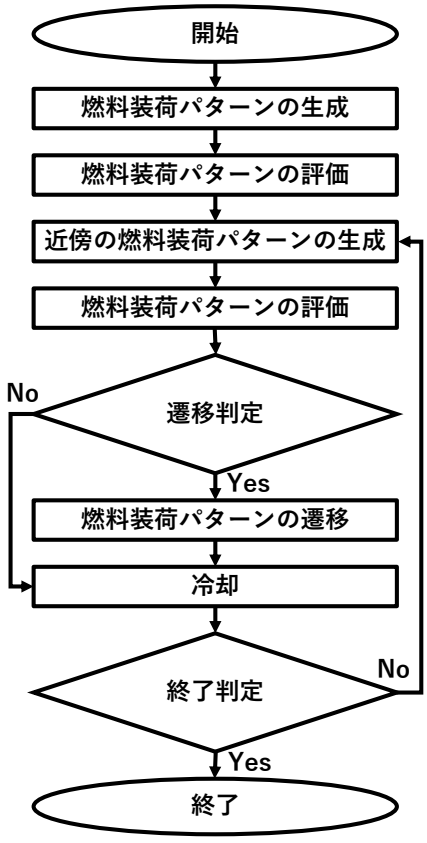



図 2-4　SA による燃料装荷パターンのフローチャート

A) ランダムに燃料装荷パターンを生成する。

B) 燃料装荷パターンの炉心特性計算を行い、目的関数の値を求める。

C) 燃料装荷パターンの近傍の燃料装荷パターンを生成する。例えば、近傍の燃料装荷パターンは、2 つの燃料集合体をランダムに交換することで得られる。

D) 近傍の燃料装荷パターンの炉心特性計算を行い、目的関数の値を求める。

E) 現在の燃料装荷パターンの目的関数の値と近傍の燃料装荷パターンの目的関数の値に基づき遷移の選択を行う。近傍の燃料装荷パターンの目的関数の値が現在の目的関数の値より良好ならば遷移を行い、悪化する場合は式(2.2)を用いて遷移の選択を行う。式(2.2)により遷移確率を求め、0 以上 1 以下の乱数を生成し、乱数が遷移確率以下であれば遷移を行い、乱数が遷移確率より大きい場合は遷移を行わない処理をする。

F) 冷却の処理として、温度を低下させる。

G) C)–F)を終了条件まで繰り返す。

H) 最も目的関数の値が良い燃料装荷パターンを最適解とする。

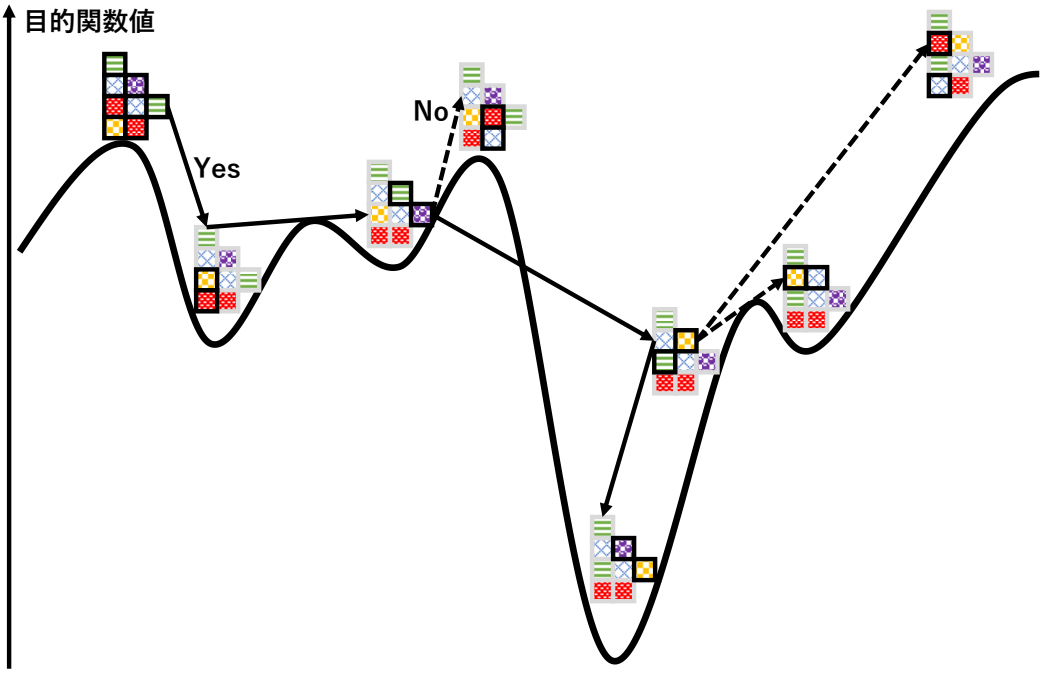



図 2-5　SA による燃料装荷パターンの過程

## 2.4　遺伝的アルゴリズム（GA）

本節では、遺伝的アルゴリズムを説明する。

### 2.4.1　生物進化の過程

遺伝的アルゴリズム（Genetic Algorithm、GA）[6]–[9]は、生物進化の過程を使用した探索アルゴリズムである。GA では、以下の 3 種類の操作を行い、解を進化させる。

1. 選択淘汰（Selection）
2. 交差（Crossover）
3. 突然変異（Mutation）

生物は環境に適用し生き残るために、環境に適合できない個体が滅び（淘汰され）、より環境に適合した個体を生み出すために遺伝子の交配（交差）や変化（突然変異）を行っている。GA による探索では、解を遺伝子に見立てることで良好な解を得ている。良好な解を保持し、解同士の一部を交換することや解に変化を与えることでさらに良好な解を得ている。

### 2.4.2　GA による燃料装荷パターン最適化

GA による燃料装荷パターン最適化のフローチャートを図 2-6 に示す。また、最適化の過程を図 2-7 に示す。GA による最適化の流れは以下の通りである。

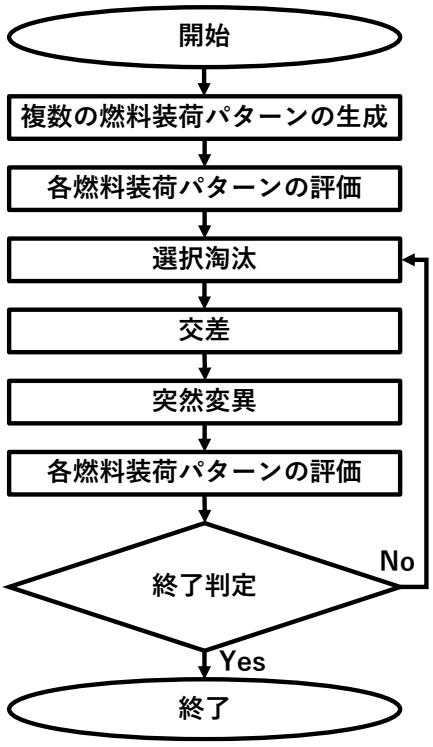



図 2-6　GA による燃料装荷パターン最適化のフローチャート

A) ランダムに複数の燃料装荷パターン（初期世代）を生成する。
B) それぞれの燃料装荷パターンの目的関数の値を求める。
C) 選択淘汰を行い、次世代の燃料装荷パターンを生成する。選択淘汰では、目的関数の値が良い燃料装荷パターンを次世代の装荷パターンを作成する「親」として確率的に選択する。
D) 次世代の燃料装荷パターンに対して交差を行う。交差では、複数の燃料装荷パターンの一部をランダムに組み替えることで、新たな燃料装荷パターンを生成する。
E) 次世代の燃料装荷パターンに対して突然変異を行う。突然変異では、燃料装荷パターンの一部の燃料集合体をランダムに交換することで、新たな燃料装荷パターンを生成する。
F) それぞれの燃料装荷パターンの目的関数の値を求める。
G) C)–F)を終了条件まで繰り返す。
H) 最も目的関数の値が良い燃料装荷パターンを最適解とする。

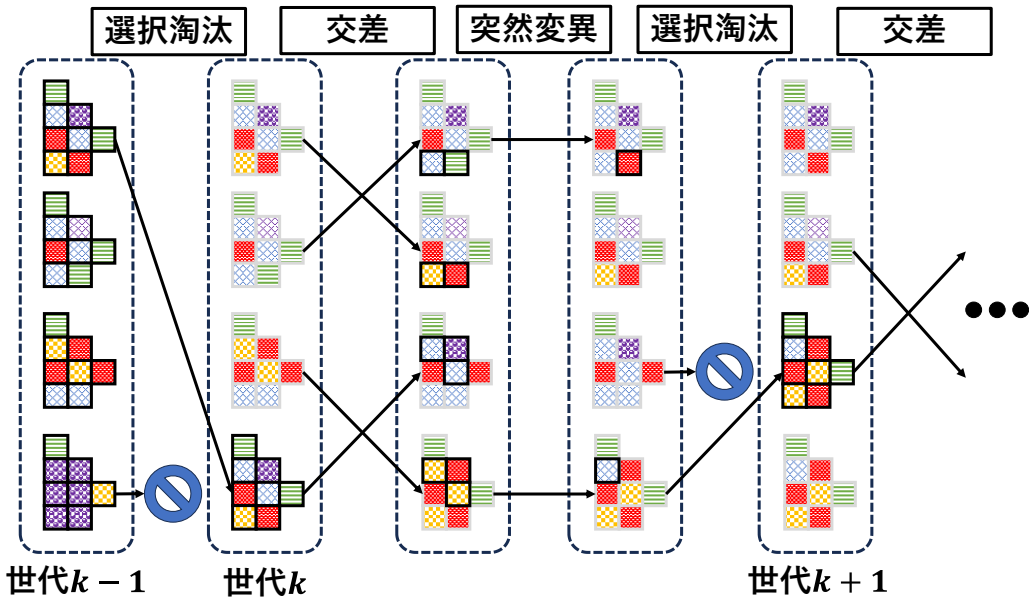



図 2-7　GA による燃料装荷パターン最適化の過程

## 2.5　蟻コロニー最適化（ACO）

本節では、蟻コロニー最適化の説明を行う。

### 2.5.1　蟻の集団行動

蟻コロニー最適化（Ant Colony Optimization, ACO）[10]–[13], [6]とは、蟻の集団行動を利用した探索アルゴリズムである。蟻が一列になってエサと巣を行き来している光景を見たことがある人も多いだろう。これは、エサを見つけた蟻がフェロモンを落としながら巣にエ

サを持ち帰り、別の蟻がフェロモンに導かれてエサに到達するためである。蟻による効率的なエサの採餌方法を図 2-8 に示す。エサが巣の周りに 3 つある場合を考え、3 匹の蟻が同時に巣を出発する（図 2-8(a)）。最も短い経路を移動した蟻が巣に帰ってくる（図 2-8(b)）。フェロモンは揮発性のため、短い経路に蓄積されるフェロモン量が多くなり、後続の蟻が短い経路を選びやすくなる（図 2-8(c)）。十分な時間が経過すると、すべての蟻が短い経路を選ぶようになる（図 2-8(d)）。このようなフェロモンを介したコミュニケーションによって、効率的なエサの採餌を可能にしている。ACO では、組合せ最適化問題を 1.1.2 項で説明した最短経路問題に置き換えている。経路を解とみなし、最短経路を最適解とすることで最適化を行っている。

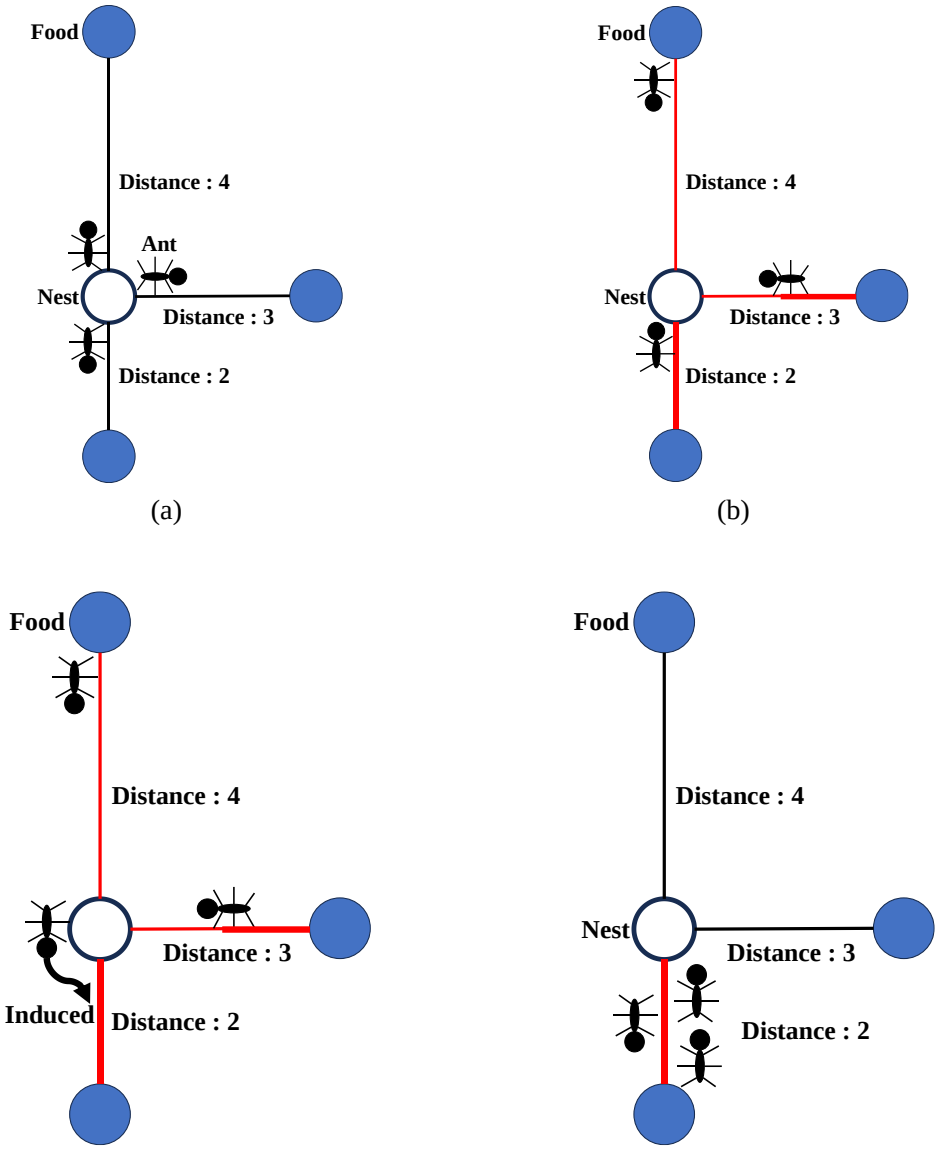



図 2-8　蟻による効率的なエサの採餌方法

### 2.5.2　ACO による燃料装荷パターン最適化

ACO による燃料装荷パターン最適化のフローチャートを図 2-9 に示す。ACO による最適化の流れは以下の通りである。

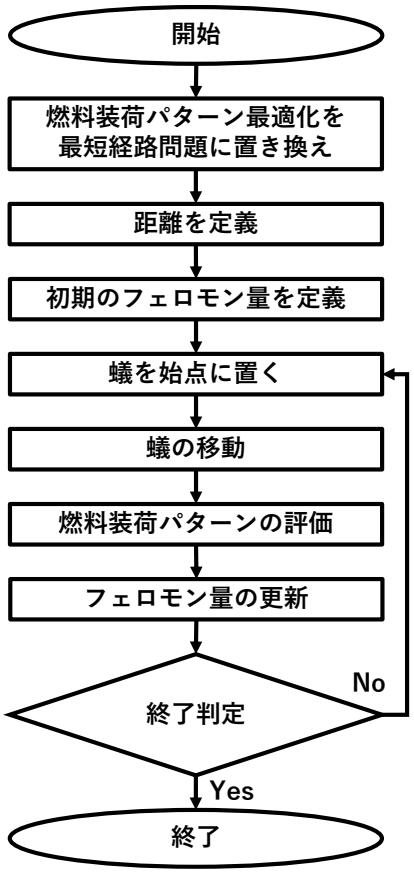



図 2-9　ACO による燃料装荷パターン最適化のフローチャート

A) 燃料装荷パターン最適化を最短経路問題に置き換えるために、図 2-10 のように、燃料装荷位置を番号の順番に訪れながら燃料集合体を装荷する問題にする。すべての燃料装荷位置を訪れることで、燃料装荷パターンが作成される。装荷手順の最適化とみなすことで、図 2-11 に示すような最短経路問題としている。

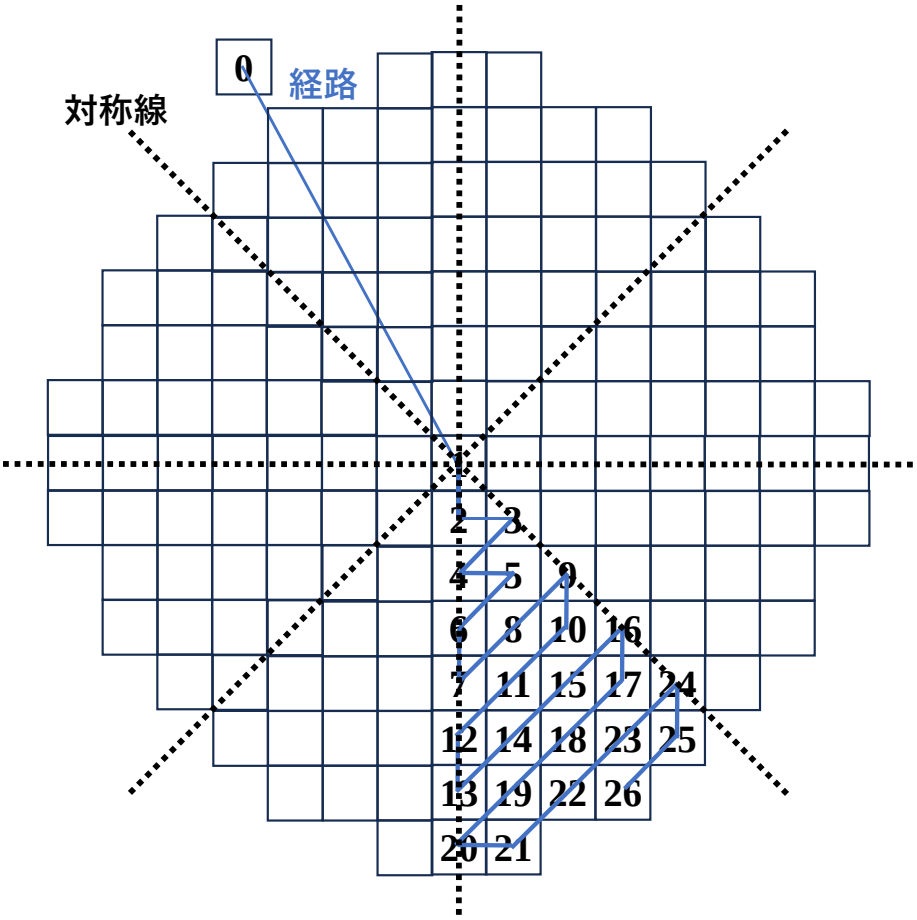



図 2-10　燃料集合体の装荷位置が 157 個ある PWR 炉心
1/8 対称性を仮定することで装荷位置を 26 個としている

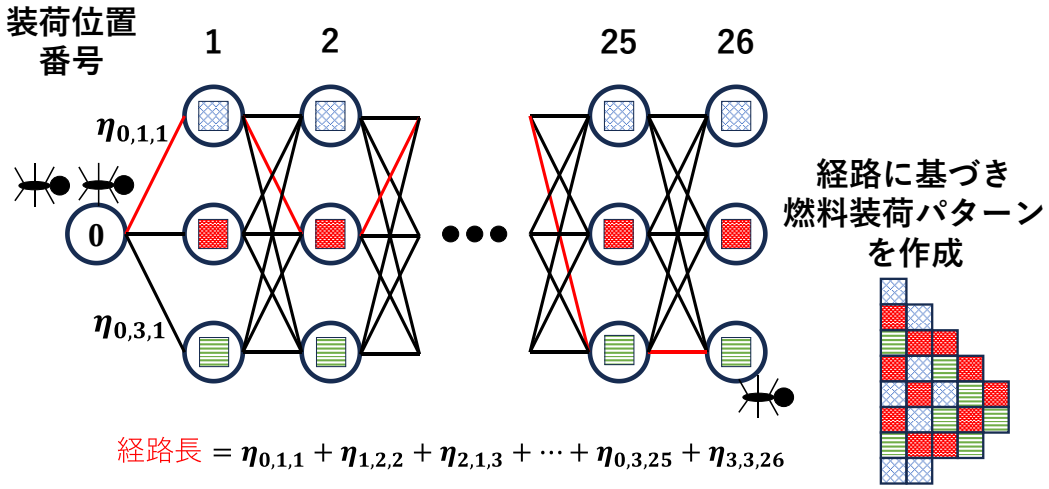



図 2-11　燃料装荷パターン最適化の最短経路問題

B) 燃料装荷位置の間の距離を定義する。例えば、燃料集合体$r$が装荷されている燃料装荷位置$p-1$と燃料集合体$s$が装荷されている燃料装荷位置$p$の間のヒューリスティックな（経験的な）距離$\eta_{r,s,p}$を式(2.3)で定義する。

$$\eta_{r,s,p} = \left| \frac{Q(r)+Q(s)}{2} - \text{average of Q} \right| \tag{2.3}$$

ここで、$Q(i)$は燃料集合体$i$の無限増倍率である。局所的な無限増倍率の平均が炉心全体の無限増倍率に近いほど、距離が短くなる。炉心性能の面から考えると、隣接する燃料の無限増倍率を炉心平均に近づけることにより、注目している隣接燃料において出力が過大、あるいは過小になることを避ける効果がある。距離の定義方法については、文献[11]と[12]に多様な方法が記載されている。

C) 初期の燃料装荷位置の間におけるフェロモン量を定義する。例えば、燃料集合体$r$が装荷されている燃料装荷位置$p-1$と燃料集合体$s$が装荷されている燃料装荷位置$p$の間のフェロモン量$\tau_{r,s,p}$をすべて 1 とする。

D) 蟻を番号 0 に配置する。

E) 蟻を最後の装荷位置番号まで蟻を移動させる。燃料装荷位置$p-1$に装荷されている燃料集合体$r$から燃料装荷位置$p$に装荷されている燃料集合体$s$に移動する確率$p_{r,s,p}$は式(2.4)で定義される。

$$p_{r,s,p} = \frac{\tau_{r,s,p}\left(\frac{1}{\eta_{r,s,p}}\right)^{\alpha}}{\sum_{i\in\text{loadable assemblies}} \tau_{r,i,p}\left(\frac{1}{\eta_{r,i,p}}\right)^{\alpha}} \tag{2.4}$$

ここで、燃料装荷位置の間のフェロモン量$\tau_{r,s,p}$、燃料装荷位置の間の距離$\eta_{r,s,p}$、$\alpha$は移動する確率に対するフェロモン量と距離の寄与を決めるパラメータである。フェロモン量が多いほど、または距離が近いほど移動しやすくなる。

F) 作成した燃料装荷パターンの目的関数の値$f$を求める。ここで、$f$が小さいほど、良好な燃料装荷パターンである。

G) 式(2.5)を用いて、すべての燃料装荷位置の間のフェロモン量$\tau_{r,s,p}$を更新する。

$$\tau_{r,s,p} = \begin{cases} (1-\rho)\tau_{r,s,p} + \dfrac{\beta}{f} & \text{蟻が移動した燃料装荷位置の間} \\ (1-\rho)\tau_{r,s,p} & otherwise \end{cases} \tag{2.5}$$

ここで、$\rho(\in(0,1])$はフェロモンの減少量を決めるパラメータ、$\beta$は目的関数の値のフェロモン量への影響度を決めるパラメータ、$f$は目的関数の値である。$\rho$が 1 に近いほどフェロモンが早く蒸発する。また、$\beta$が大きいほど目的関数の値がフェロモン量に影響を与える。

H) D)–G)を終了条件まで繰り返す。

I) 最も目的関数の値が良い燃料装荷パターンを最適解とする。

ACO では、燃料装荷パターン最適化を装荷手順の最適化とみなすことで最短経路問題としている。第 3 章で説明を行うモンテカルロ木探索でも同様に、燃料装荷パターン最適化を装荷手順の最適化とみなしている。ただし、モンテカルロ木探索の場合は、最短経路問題とするのではなく、木探索問題としている。燃料装荷パターン最適化を木探索問題に帰着させる方法の詳細は 4.2.1 項で説明する。

## 2.6　ヒューリスティック手法と機械学習を用いた手法

本節ではヒューリスティック手法と機械学習を用いた手法を説明する。

1.1 節で述べたように、燃料装荷パターン最適化の時間的な制約は次の 2 つ要因で生じる。

1. 膨大な燃料装荷パターンの中から最適な燃料装荷パターンを見つける必要がある。
2. 目的関数を得るためには詳細な炉心特性を計算する必要があるが、1 つの燃料装荷パターンあたり数分～数十分の時間を要する。

1 つ目の組合せ爆発による時間的な制約を解消するためにヒューリスティック手法を用いるが、2 つ目の炉心計算による時間的な制約を解消するために機械学習を用いる研究が行われている。

### 2.6.1　機械学習と深層学習

機械学習（Machine Learning, ML）とは、計算機科学の分野に起源をもち、工学的に言い換えるとパターン認識となる[14]。パターン認識の例として、図 2-12 にアルファベット認識を示す。

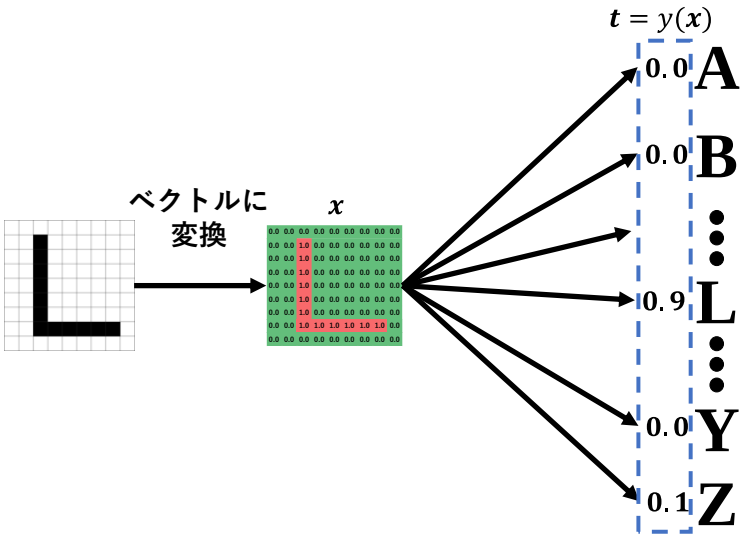
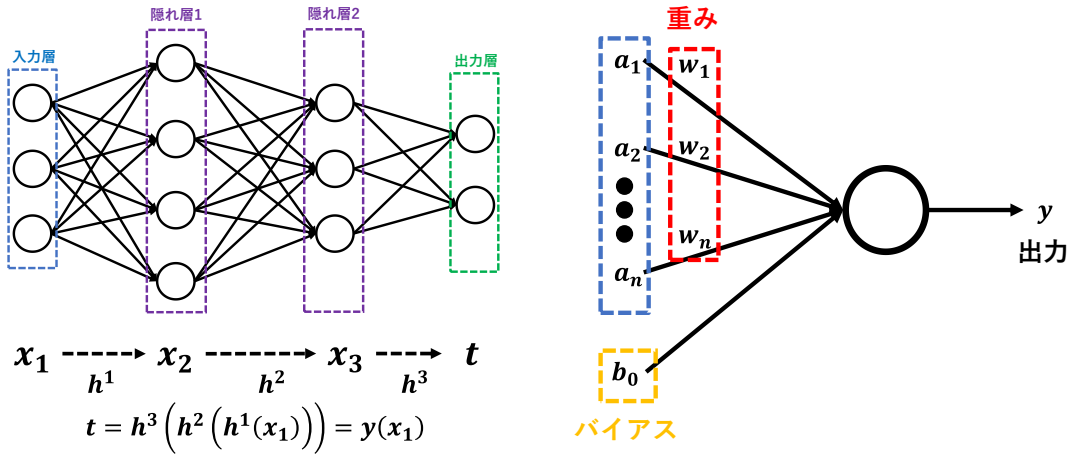



図 2-12　アルファベットの認識

アルファベットの画像は$9 \times 9$ピクセルの画像であり、2 次元のベクトル$\boldsymbol{x}$で表現できる。与えられた画像がそれぞれのアルファベットである確率のベクトルを$\boldsymbol{t}$とすると、ML では式(2.6)の関数$y(\boldsymbol{x})$を得ることが目的となる。

$$\boldsymbol{t} = y(\boldsymbol{x}) \tag{2.6}$$

ML では、関数$y(\boldsymbol{x})$としてニューラルネットワーク（Neural Network, NN）が頻繁に用いられている。NN とは、パターン認識を行うことができる人間の神経系を模擬するために開発されたものである[15]。NN の構成と最小単位（パーセプトロン）を図 2-13 に示す。




(a) ニューラルネットワーク　　(b) パーセプトロン

図 2-13　ニューラルネットワークの構造と最小単位

NN は多数のパーセプトロンがネットワークを構成する。NN の出力は、重み$\boldsymbol{w}$によって変化する。学習では、学習データの入力ベクトルと出力ベクトルをそれぞれ$\boldsymbol{x}, \boldsymbol{t}$、学習データの入力$\boldsymbol{x}$に対する NN の出力ベクトルを$\hat{\boldsymbol{t}}$とすると、$\boldsymbol{t}$と$\hat{\boldsymbol{t}}$の差が小さくなるように重み$\boldsymbol{w}$の調整を行う。$\boldsymbol{t}$と$\hat{\boldsymbol{t}}$の差を表す関数を損失関数$L$といい、数値を出力する場合には 2 乗損失、複数の分類から 1 つを選ぶ場合ではクロスエントロピー損失を使用するのが一般的である[15]。$r$個の要素から成る出力ベクトルに対する 2 乗損失とクロスエントロピー損失をそれぞれ式(2.7)と式(2.8)示す。

$$L = \sum_{i=1}^{r} (t_i - \hat{t_i})^2 = \sum_{i=1}^{r} (y(\boldsymbol{x}, \boldsymbol{w})_i - \hat{t_i})^2 \tag{2.7}$$

$$L = -\sum_{i=1}^{r} t_i \log(\hat{t_i}) = -\sum_{i=1}^{r} y(\boldsymbol{x}, \boldsymbol{w})_i \log(\hat{t_i}) \tag{2.8}$$

損失関数$L$を最小化するように重み$\boldsymbol{w}$を調整するには、1.1.2 項で述べたように損失関数の勾配をとり、式(2.9)に従って重み$\boldsymbol{w}$を更新すればよい。

$$\boldsymbol{w} = \boldsymbol{w} - \alpha \cdot \mathrm{grad}\, L \tag{2.9}$$

ここで、$\alpha$は学習率といい、$\alpha$が大きいほど重み$\boldsymbol{w}$の変化量が大きくなる。

アルファベットの認識のように入出力の関係が単純な問題であれば、少ない層のニューラルネットワークで学習が可能であるが、言語や音声認識などの複雑な入出力の問題は、多くの層を持つ NN により達成可能である[16], [17]。これは、層の数を増やすことにより、NN のパラメータ数が多くなり、NN の表現力が向上するためである。このように、多くの層を持つ NN を深層ニューラルネットワーク（Deep Neural Network, DNN）といい、DNN を用いた ML を深層学習（Deep Learning, DL）という。ML では、学習データの前処理を用いてパターンが持つ有用な特徴の抽出をすることができれば、高速なパターン認識が可能となる。

### 2.6.2　機械学習を用いた燃料装荷パターン最適化

ML を用いたアルファベット認識のように、ML を用いて燃料装荷パターンの目的関数の値を高速に予測することにより、炉心計算による時間的な制約を解消する手法の研究が行われてきた[18]–[22]。ML を用いた目的関数の値の推定方法の例を図 2-14 に示す。燃料装荷パターンの 2 次元的な特徴に注目し、燃料集合体のパラメータを要素とした 2 次元ベクトルに変換する。パラメータの 2 次元ベクトルを NN 入力することによって、高速に燃料装荷パターンの目的関数の値を得ることができる。

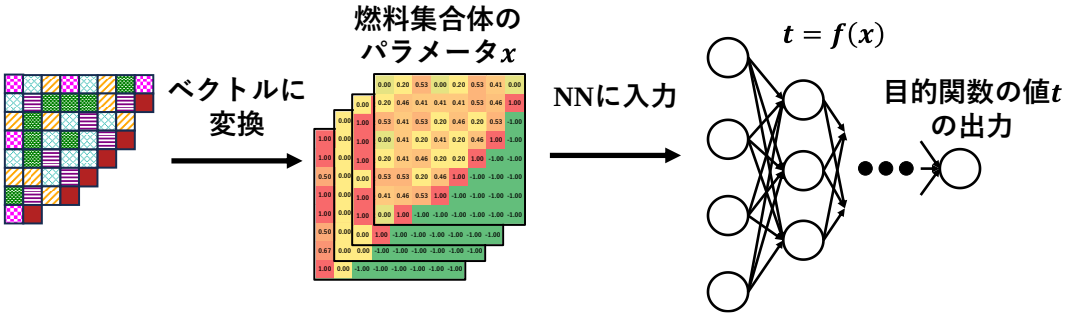



図 2-14　機械学習による目的関数の値の推定方法の例

計算速度やアルゴリズムの向上による画像認識の精度向上にしたがって、ML を用いた燃料装荷パターンの目的関数の値の推定が精度よくできるようになっている。精度の向上によって、燃料装荷パターン最適化では、ヒューリスティック手法による最適化の際に ML を用いた燃料装荷パターン評価をおこなうことによる高速化が行われている[21], [22]。

なお、従来手法では高速な燃料装荷パターンの評価に ML を用いているが、第 5 章で述べるモンテカルロ木探索と深層学習を用いた燃料装荷パターン最適化では、効率的に最適な燃料装荷パターンを見つけることを目的として ML を用いているため、本項で説明した手法とは全く別の手法である。

## 2.7　本章のまとめ

本章では、燃料装荷パターン最適化に対する代表的な手法を紹介した。

2.2 節では、ヒューリスティック手法の説明を行った。ヒューリスティック手法とは、問題に固有の知識や経験的な知識を適用し、厳密な最適解ではないが、現実的な時間内で実用上に満足のいく近似解を求める手法である。また、燃料装荷パターンに対するヒューリスティック手法の適用可能性について議論し、燃料装荷パターンと目的関数の値は相関を持つため適用が可能であることを述べた。

2.3 節では、焼きなまし法の説明を行った。2.3.1 項では、焼きなまし法は結晶の生成過程を模倣した手法であることを説明した。2.3.2 項では、焼きなまし法による燃料装荷パターン最適化を解説した。

2.4 節では、遺伝的アルゴリズムの説明を行った。2.4.1 項では、遺伝的アルゴリズムは生物の進化過程を模倣した手法であることを説明した。2.4.2 項では、遺伝的アルゴリズムによる燃料装荷パターン最適化を述べた。

2.5 節では、蟻コロニー最適化の説明を行った。2.5.1 項では、蟻コロニー最適化は生物の集団行動を模倣した手法であることを説明した。2.5.2 項では、蟻コロニー最適化による燃料装荷パターン最適化を説明した。蟻コロニー最適化では、燃料装荷パターン最適化を装荷手順の最適化とみなすが、第 4 章で解説するモンテカルロ木探索を用いた燃料装荷パターン最適化でも同様に、装荷手順の最適化とみなす。

2.6 節では、ヒューリスティック手法と機械学習を用いた燃料装荷パターン最適化について言及した。2.6.1 項では、機械学習と深層学習の説明を行った。2.6.2 項では、機械学習による燃料装荷パターン評価の高速化を利用した手法に言及した。第 5 章では、2.6.2 項での ML の利用方法とは全く別の方法で ML を用いる。

第 3 章では、燃料装荷パターン最適化の話題から一旦離れ、モンテカルロ木探索の説明を行う。第 4 章で再び燃料装荷パターン最適化に戻り、モンテカルロ木探索による燃料装荷パターン最適化の説明と数値実験を行う。

## 2.8　参考文献


[1] 竹原有紗, “用語解説：第 7 回テーマ：ヒューリスティックアプローチ,” *電気学会論文誌 b（電力・エネルギー部門誌）*, vol. 131, no. 5, p. NL5_7, May 2011, doi: 10.1541/ieejpes.131.NL5_7.

[2] A. Yamamoto, “Study on Advanced In-Core Fuel Management for Pressurized Water Reactors Using Loading Pattern Optimization Methods,” Ph.D. dissertation, Socio-Environ. Energy Sci. Dept., Kyoto Univ., Kyoto, Japan, 1998. [Online]. Available: https://repository.kulib.kyoto-u.ac.jp/dspace/bitstream/2433/156982/2/D_Yamamoto_Akio.pdf.

[3] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by Simulated Annealing,”

*Science*, vol. 220, no. 4598, pp. 671–680, May 1983, doi: 10.1126/science.220.4598.671.

[4] V. Černý, "Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm," *J. Optim. Theory Appl.*, vol. 45, no. 1, pp. 41–51, Jan. 1985, doi: 10.1007/BF00940812.

[5] T. Šmuc, D. Pevec, and B. Petrović, "Annealing Strategies for Loading Pattern Optimization," *Ann. Nucl. Energy*, vol. 21, no. 6, pp. 325–336, Jun. 1994, doi: 10.1016/0306-4549(94)90028-0.

[6] 相吉英太郎, 安田恵一郎, *メタヒューリスティクスと応用*, 東京, オーム社, 2007.

[7] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, Ann Arbor, Michigan, USA: U Michigan Press, 1975.

[8] G. T. Parks, "Multiobjective Pressurized Water Reactor Reload Core Design by Nondominated Genetic Algorithm Search," *Nucl. Sci. Eng.*, vol. 124, no. 1, pp. 178–187. Sep. 1996, doi: 10.13182/NSE96- A24233

[9] A. Yamamoto, "A Quantitative Comparison of Loading Pattern Optimization Methods for In-Core Fuel Management of PWR," *J. Nucl. Sci. Technol.*, vol. 34, no. 4, pp. 339–347, 1997, doi: 10.1080/18811248.1997.9733673.

[10] M. Dorigo, G. D. Caro, and L. M. Gambardella, "Ant Algorithms for Discrete Optimization," *Artif. Life*, vol. 5, no. 2, pp. 137–172, Apr. 1999, doi: 10.1162/106454699568728.

[11] L. Machado and R. Schirru, "The Ant-Q Algorithm Applied to the Nuclear Reload Problem," *Ann. Nucl. Energy*, vol. 29, no. 12, pp. 1455–1470, Aug. 2002, doi: 10.1016/S0306-4549(01)00118-9.

[12] 岸宏憲, 北田考典, "蟻コロニー最適化法の燃料装荷問題への適用性," *日本原子力学会和文論文誌*, vol. 12, no. 1, pp. 103‐112, 2013, doi: 10.3327/taesj.j12.018.

[13] 伊庭斉志, *深層学習とメタヒューリスティクス：ディープ・ニューラルエボリューション*, 東京, オーム社, 2019.

[14] C. M. Bishop, *Pattern Recognition and Machine Learning*, New York, NY, USA: Springer New York, 2006 (監修: 元田浩, 栗田多喜夫, 樋口知之, 松本裕治, 村田昇, *パターン認識と機械学習: ベイズ理論による統計的予測 上*, 東京, 丸善出版, 2012).

[15] C. C. Aggarwal, *Neural Networks and Deep Learning*, Switzerland: Springer Cham, 2018 (訳: 李鍾賛 ほか, *ニューラルネットワークとディープラーニング*, 東京, 学術図書出版社, 2022).

[16] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A Fast Learning Algorithm for Deep Belief Nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006, doi: 10.1162/neco.2006.18.7.1527.

[17] S. Theodoridis, *Machine Learning: A Bayesian and Optimization Perspective*, Cambridge, Massachusetts, USA: Academic Press, 2020 (訳: 石川達也 ほか, *機械学習: ベイズと最*

*適化の観点から*, 東京, 共立出版, 2022).

[18] H. G. Kim, S. H. Chang, and B. H. Lee, "Pressurized Water Reactor Core Parameter Prediction Using an Artificial Neural Network," *Nucl. Sci. Eng.*, vol. 113, no. 1, pp. 70–76, Jan. 1993, doi: 10.13182/NSE93-A23994.

[19] J. Jang, S.-H. Seong, and U.-C. Lee, "A Fast-Running Core Prediction Model Based on Neural Networks for Load-Following Operations in a Soluble Boron-Free Reactor," *Ann. Nucl. Energy*, vol. 34, no. 9, pp. 752–764, Sep. 2007, doi: 10.1016/j.anucene.2007.03.008.

[20] H. Jang, H. C. Shin, and H. C. Lee, "Refinement of Convolutional Neural Network for Neutronic Desing Parameter Prediction of a Loading Pattern," in *Proc. Reactor Physics Asia 2019 (RPHA19) Conf.*, Osaka, Japan, Dec 2–3, 2019, pp. 175–178. [Online]. Available: https://www.rri.kyoto-u.ac.jp/PUB/report/09_kurns/temp/kurns-ekr-005.pdf.

[21] A. Naserbegi, M. Aghaie, and S. M. Mahmoudi, "PWR Core Pattern Optimization Using Grey Wolf Algorithm Based on Artificial Neural Network," *Prog. Nucl. Energy*, vol. 129, p. 103505, Nov. 2020, doi: 10.1016/j.pnucene.2020.103505.

[22] C. Wan, K. Lei, and Y. Li, "Optimization Method of Fuel-Reloading Pattern for PWR Based on the Improved Convolutional Neural Network and Genetic Algorithm," *Ann. Nucl. Energy*, vol. 171, p. 109028, Jun. 2022, doi: 10.1016/j.anucene.2022.109028.

# 第3章　木と木探索手法

## 3.1　本章の概要

本章では、木とゲーム木探索を説明した後、囲碁や将棋などのボードゲームで使用される木探索手法であるモンテカルロ木探索を説明する。モンテカルロ木探索の燃料装荷パターン最適化への適用については、第 4 章で解説する。

3.2 節では、木の理解に必要なグラフ理論の用語の説明を行い、木の定義を述べる。

3.3 節では、ゲーム木とゲーム木探索を説明する。

3.4 節では、モンテカルロ法を用いた木探索手法である原始モンテカルロ木探索を説明する。3.4.1 項では、原始モンテカルロ木探索の探索戦略を説明する。3.4.2 項では、グラフ理論の概念を用いて原始モンテカルロ木探索を詳説する。

3.5 節では、原始モンテカルロ木探索を改良したモンテカルロ木探索を説明する。3.5.1 項では、原始モンテカルロ木探索の探索戦略を改良したモンテカルロ木探索の探索戦略を説明する。3.5.2 項では、モンテカルロ木探索の探索戦略実現のための Upper Confidence Bounds applied to trees というアルゴリズムを解説する。3.5.3 項では、グラフ理論の概念を用いてモンテカルロ木探索を解説する。

3.6 節では、本章のまとめを述べる。

## 3.2　グラフ理論における木

本節では、グラフ理論の用語と木の説明を行う。

グラフ（Graph）$G$とは、頂点（Node, Vertex）の集合$V(G)$と辺（Edge）の集合$E(G)$のペア$\big(V(G), E(G)\big)$に、各辺に頂点のペアを対応させる接続関数$\psi_G$を加えたものである[1], [2]。辺$e$が頂点$u$と頂点$v$を結ぶとき、$\psi_G(e) = \{u, v\}$と表す。このとき、$u$と$v$は$e$の端点であり、$u$と$v$は隣接するという。例えば、式(3.1)で表されるグラフの図形的表現は図 3-1 のようになる。

$$
\begin{gathered}
G = \big(V(G), E(G)\big) \\
V(G) = \{v_1, v_2, v_3, v_4\} \\
E(G) = \{e_1, e_2, e_3, e_4, e_5, e_6\} \qquad (3.1) \\
\psi_G(e_1) = \{v_1, v_2\} \quad \psi_G(e_2) = \{v_2, v_3\} \quad \psi_G(e_3) = \{v_3, v_4\} \\
\psi_G(e_4) = \{v_4, v_1\} \quad \psi_G(e_5) = \{v_1, v_3\} \quad \psi_G(e_6) = \{v_2, v_4\}
\end{gathered}
$$

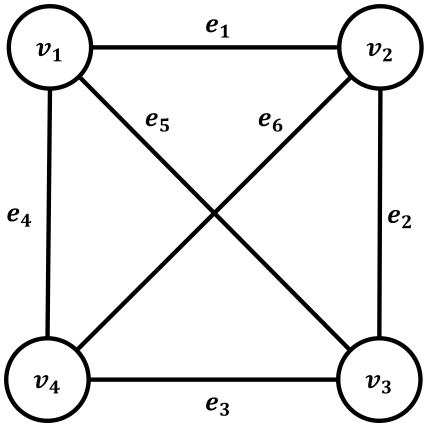


図 3-1　式(3.1)で表されるグラフの図形的表現

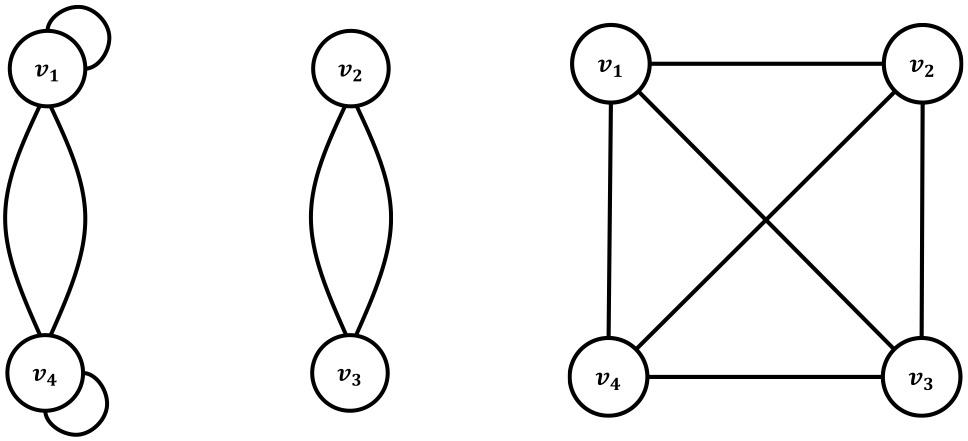
また、グラフ$G = (V, E)$には経路（Path）が存在する。経路とは、頂点の系列であって、連続する頂点が隣接しているものである。例えば、式(3.1)で表されるグラフには、式(3.2)で表される経路$P$がある。

$$P = (v_1, v_2, v_3, v_4) \tag{3.2}$$

このとき、経路の長さは経路を構成する辺の長さで表される。各辺の長さを 1 としたとき、式(3.2)で表される経路$P$の長さは 3 である。また、経路$P$の始まりとなる頂点と終わりとなる頂点が等しいとき、$P$を閉路（Cycle）という。例えば、式(3.1)で表されるグラフでは、式(3.3)で表される長さ 4 の閉路がある。

$$P = (v_1, v_2, v_3, v_4, v_1) \tag{3.3}$$

次に、グラフの連結性について説明する。グラフ$G = (V, E)$が連結であるとは、$V$を 2 つの集合$X, Y$にどのように分割しても、$X$と$Y$の両方に端点を持つ辺が存在することである。言い換えると、どのように分割しても、$X$と$Y$とつなぐ辺(共有される辺)が存在していることである。非連結グラフ（連結でないグラフ, Disconnected Graph）と連結グラフ（Connected Graph）の例を図 3-2 に示す。図 3-2(a)では、$V = \{v_1, v_2, v_3, v_4\}$を$X = \{v_1, v_4\}, Y = \{v_2, v_3\}$の 2 つの集合に分割すると、$X$と$Y$の両方に端点を持つ辺が存在しないため、非連結グラフである。一方、図 3-2(b)では、任意の頂点同士が隣接しているため連結グラフである。




(a) 非連結グラフ　　(b) 連結グラフ

図 3-2　連結グラフと非連結グラフ

これまでは辺の方向を考慮していなかったが、方向を考慮するグラフが存在する。辺$e$の方向を考えない、つまり、$\psi_G(e) = \{u, v\} = \{v, u\}$とするとき、$G$を無向グラフ（Undirected Graph）という。一方、辺の方向を考慮するグラフは、有向グラフ（Directed Graph）という。有向辺$e$が頂点$u$から頂点$v$を結ぶとき、$\psi_G(e) = (u, v)$と表し、$u$を$e$の始点、$v$を$e$の終点という。例えば、式(3.4)で表されるグラフの図形的表現は、図 3-3 のようになる。

$$G = \big(V(G), E(G)\big)$$
$$V(G) = \{v_1, v_2, v_3, v_4\}$$
$$E(G) = \{e_1, e_2, e_3, e_4, e_5, e_6\} \tag{3.4}$$
$$\psi_G(e_1) = (v_1, v_2) \quad \psi_G(e_2) = (v_2, v_3) \quad \psi_G(e_3) = (v_3, v_4)$$
$$\psi_G(e_4) = (v_4, v_1) \quad \psi_G(e_5) = (v_1, v_3) \quad \psi_G(e_6) = (v_2, v_4)$$

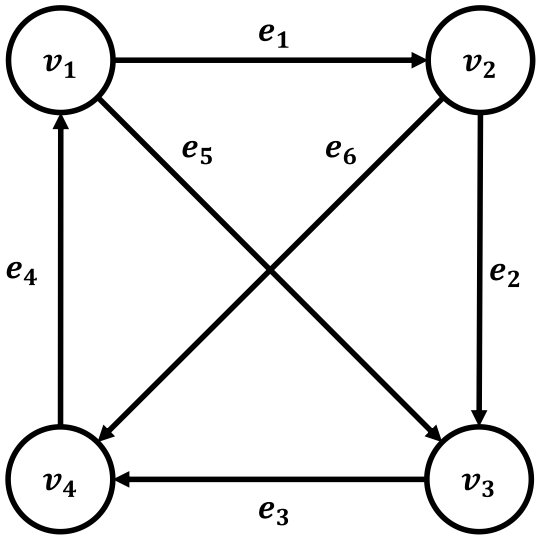



図 3-3　式(3.4)で表されるグラフの図形的表現

有向グラフの場合、頂点を始点とする有向辺の個数を出次数、頂点を終点とする有向辺の個数を入次数という。例として、式(3.4)で表されるグラフの頂点における出次数と入次数を表 3-1 に示す。

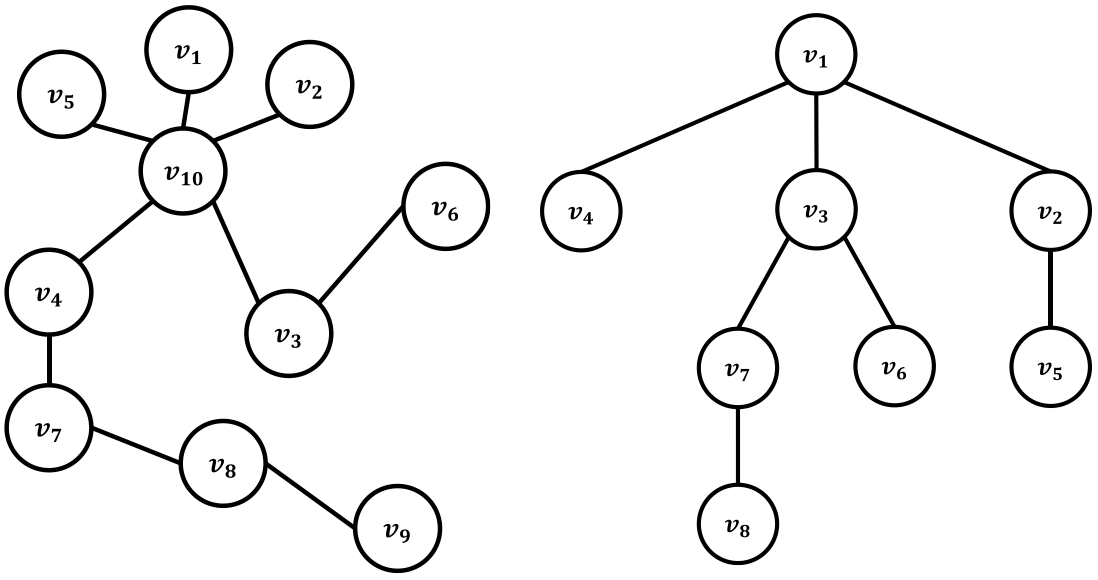
表 3-1　式(3.4)で表されるグラフの頂点における出次数と入次数

| 頂点 | 出次数 | 入次数 |
|---|---|---|
| $v_1$ | 2 | 1 |
| $v_2$ | 2 | 1 |
| $v_3$ | 1 | 2 |
| $v_4$ | 1 | 2 |

また、経路を有向グラフに拡張したものを有向路といい、式(3.5)で表される有向辺が式(3.4)で表されるグラフに存在している。

$$P = (v_1, v_3, v_4, v_1, v_2) \tag{3.5}$$

最後に、木の説明を行う。木（Tree）とは、長さ 1 以上の閉路を持たない連結な無向グラフである。木の例を図 3-4 に示す。図 3-4(a)と図 3-4(b)の図形的表現は異なるが、どちらも木である。




(a)　　(b)

図 3-4　木の例

辺の方向を考慮した木を有向木という。有向木では、入次数が 0 となる頂点$r$が存在し、それ以外の頂点の入次数は 1 となる。入次数が 0 となる頂点$r$を根（Root）という。有向木において、$\psi_G(e) = (u, v)$となる辺が存在するならば、$u$は$v$の親（Parent）、$v$は$u$の子（Child）である。また、子を持たない頂点を葉（Leaf）という。$u$から$v$への長さ 0 以上の有向路が存在すれば、$u$を$v$の祖先（Ancestor）、$v$を$u$の子孫（Descendant）という。根$r$から頂点$v$までの有向路の長さが$k$であるとき、深さ（Depth）$k$であるといい、すべての頂点における深さの最大値を有向木の高さ（Height）という。

有向木の例を図 3-5 に示す。頂点$v_1$は入次数が 0 であるため、木の根である。また、$v_2$を始点、$v_5$を終点とする辺が存在するため、$v_2$は$v_5$の親、$v_5$は$v_2$の子である。$v_9$は出次数が 0 であるため、木の葉である。

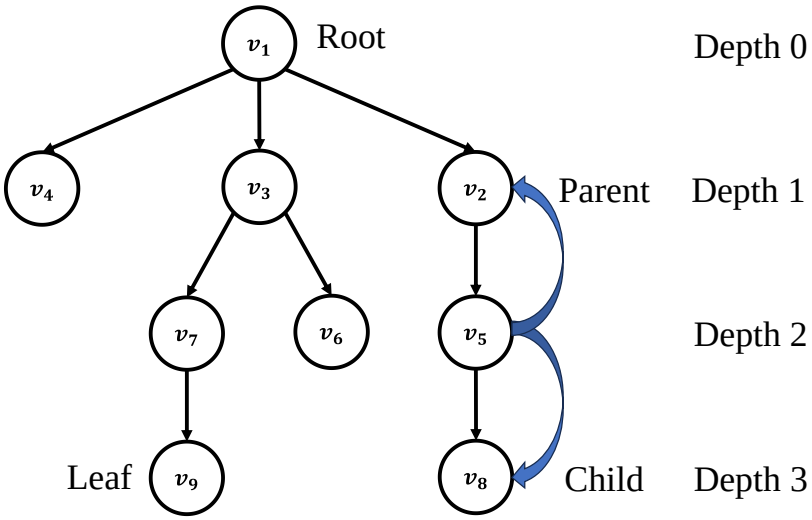



図 3-5　高さ 3 の有向木の例

### 3.3　ゲーム木探索

本節では、ゲーム木とゲーム木探索の説明を行う。

ゲーム木とは、ゲームの局面から次の局面に有向辺を接続した有向木である[3]。三目並べのゲーム木を図 3-6 に示す。頂点が局面、辺が○×を書きこむ行動に対応するゲーム木である。

ゲーム木はゲームの複雑さに応じて深さ方向に関して指数関数的に大きくなることから、良い局面を探すためにゲーム木のすべてを解析することは現実的でない。したがって、ゲームの一部をゲーム木として表し、優先順位をつけながら良い局面を探すことが重要となる。効率的に良い局面を探すための手法の総称をゲーム木探索という[4]。

ゲーム木とゲーム木探索の理解を深めるために、三目並べにおけるゲーム木探索を解説する。三目並べのゲーム木の例を図 3-7 に示す。図 3-7 に示されたゲーム木の根は「中心のマスに×が書かれた局面」である。

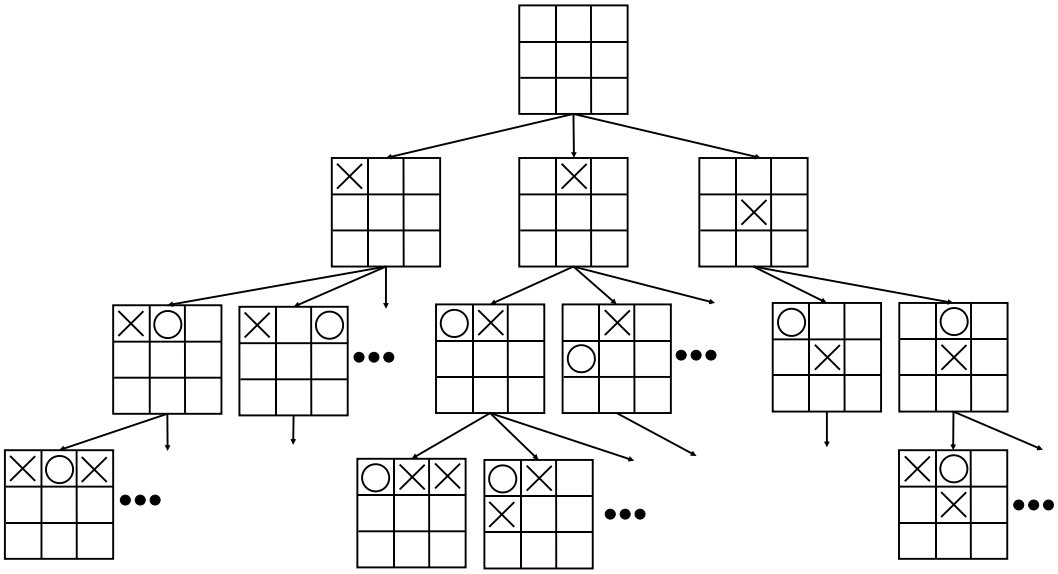


図 3-6　三目並べのゲーム木

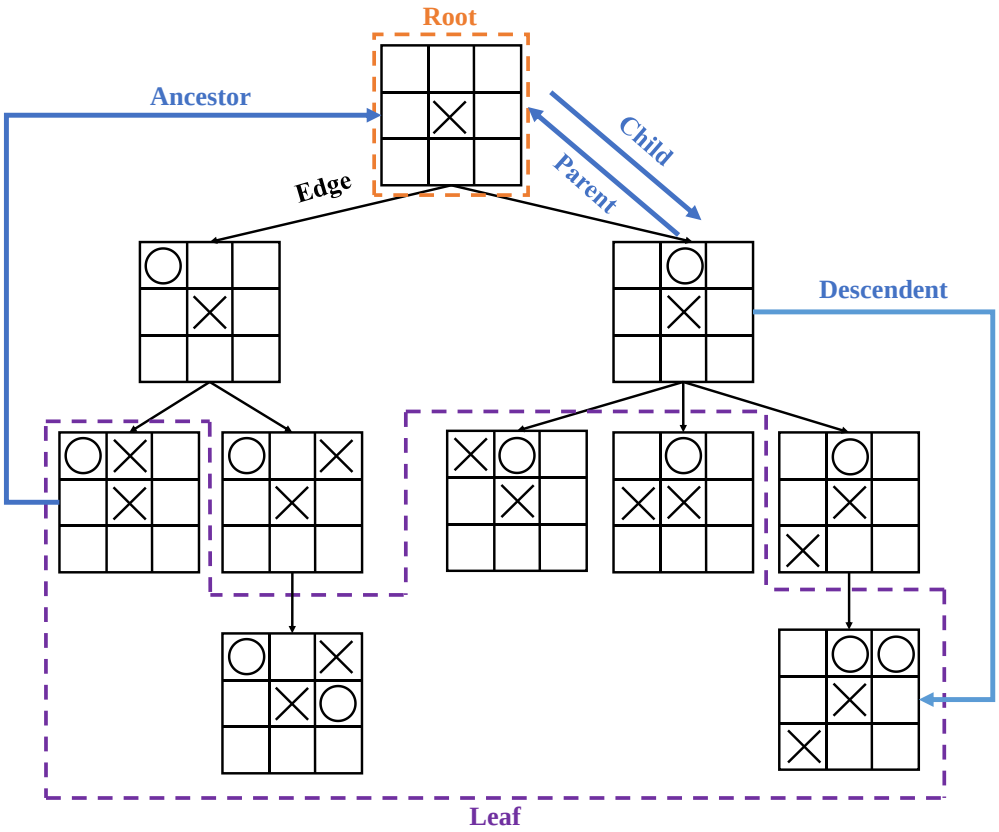
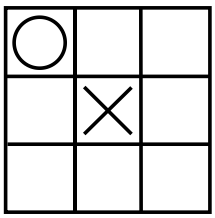
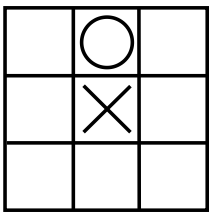



図 3-7　三目並べのゲーム木の例

ゲーム木探索では、頂点間を辺に沿って移動することで探索を行う。図 3-7 の根の子を図 3-8 を示す。根の子である図 3-8(a)と図 3-8(b)を比較すると、次に×を書くプレイヤーにとって、図 3-8(b)の局面のほうが良い局面であることは経験的にわかるかもしれない。実際に図 3-8(b)に示す局面から×を書くプレイヤーが最善手を選び続ければ必ず勝利することができるが、図 3-8(a)に示す局面から×を書くプレイヤーが最善手を選び続けたとしても引き分けになる場合がある。もし、図 3-8(b)が良い局面であると認識できれば、効率的に良い局面を探すことができるため、ゲーム木探索の本質は「良い局面と悪い局面を区別できるようにすること」といえる。



(a)　根の左の子



(b)　根の右の子

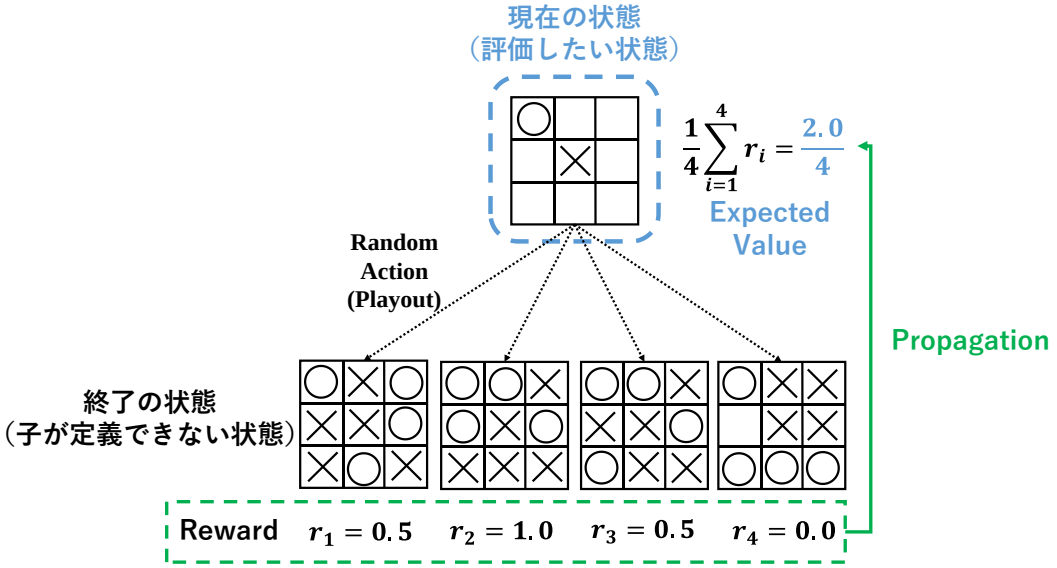
図 3-8　根の子

## 3.4　原始モンテカルロ木探索（PMCTS）

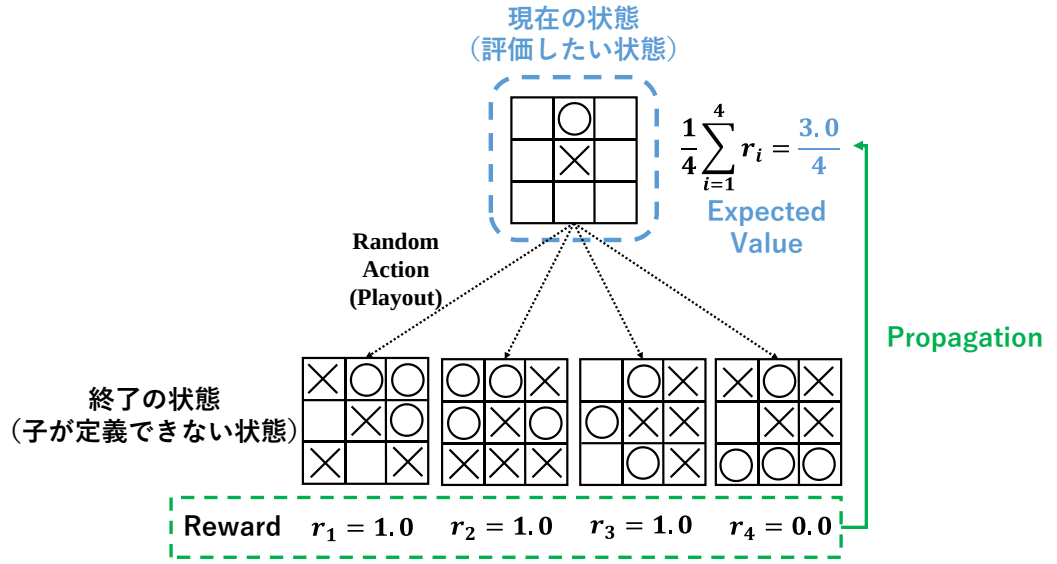
本節では、モンテカルロ法を用いた木探索手法である原始モンテカロ木探索（Pure Monte Carlo Tree Search, PMCTS）を解説する。

### 3.4.1　PMCTS の概念

PMCTS では、状態（例えば、三目並べの局面）の良し悪しを評価するために、ランダムプレイアウトを行う。ランダムプレイアウトとは、現在の状態から終了の状態（例えば、三目並べの決着がついた局面）までランダムに行動（例えば、○か×を書きこむ）を行うことである。具体的には、現在の状態から複数回のランダムプレイアウトを行うことで終了の状態の報酬（例えば、勝ち 1, 引き分け 0.5, 負け 0）の合計を求め、ランダムプレイアウトで得られた報酬の期待値を状態の良し悪しを評価する指標とする。PMCTS による状態の良し悪しの評価方法を図 3-9 に示す。図 3-9 では、現在の状態から終了の状態までランダムに行動し続けることを複数回行い、終了の状態で得た報酬を現在の状態に伝播することで現在の頂点の評価を行っている。図 3-9(a)の報酬の期待値は 2/4、図 3-9(b)の報酬の期待値は 3/4 であるため、図 3-9(b)の状態が良いことが評価できる。




(a)




(b)

図 3-9　PMCTS による状態評価方法の概念図（三目並べを例に）

### 3.4.2　PMCTS のアルゴリズム

本項では、グラフ理論を用いて 3.4.1 項で行った PMCTS の概念を解説する。

ゲーム木の頂点$s$は状態、辺$a$は可能な行動に対応する。また、それぞれの頂点は統計量$\{N(s), W(s), Q(s)\}$を持つ。ここで、$N(s)$は訪問回数（探索で訪れた回数）、$W(s)$は累積報酬（伝播された報酬の合計値）、$Q(s)$は期待値（伝播された報酬の期待値、つまり、$W(s)/N(s)$）である。PMCTS では頂点の統計量を更新しながら探索と頂点の評価を行う。以下で、三目並べを例に PMCTS のアルゴリズムを説明する。

三目並べの開始の状態と移動可能な状態を表すゲーム木を図 3-10 に示す。

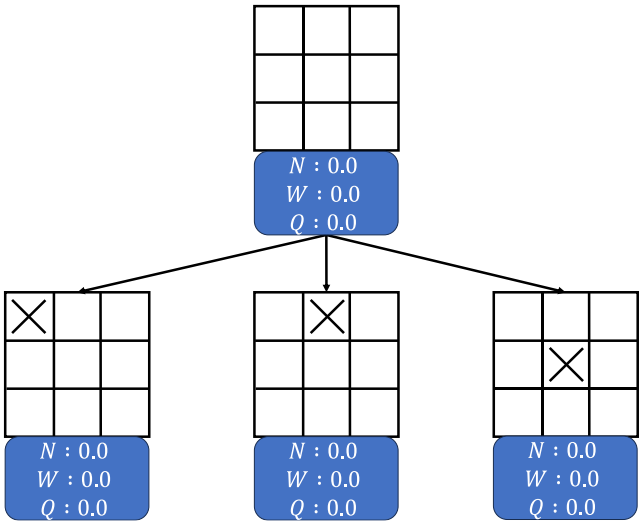
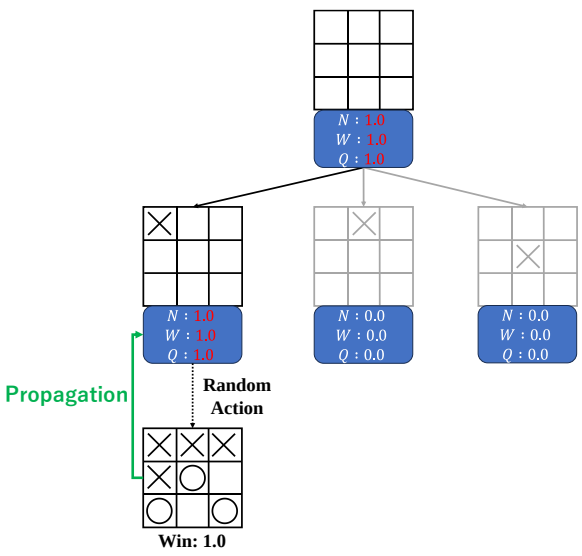
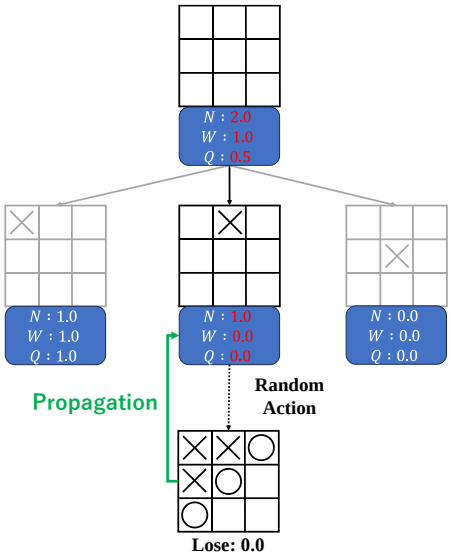



図 3-10　三目並べの開始の状態と可能な行動後の状態を表すゲーム木

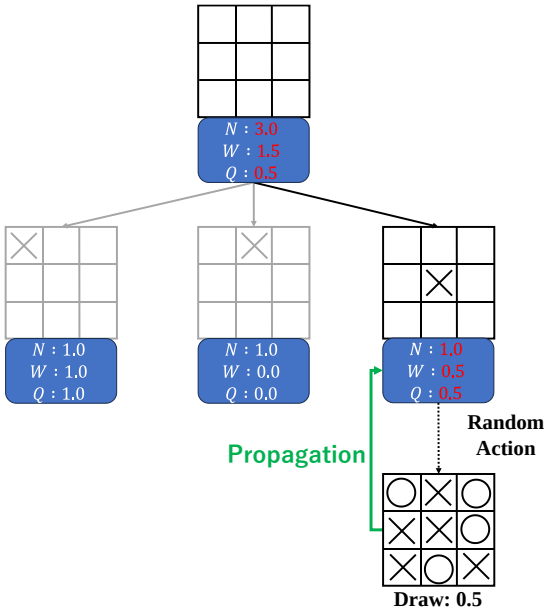
状態の良し悪しを決めるために、ランダムプレイアウトを指定回数だけ行う。ランダムプレイアウトで得られる評価値は、勝ち：1、引き分け：0.5、負け：0 と設定する。根から 100 回の探索を行い、子の評価を行うために子からランダムプレイアウトを行う例を図 3-11 に示す。



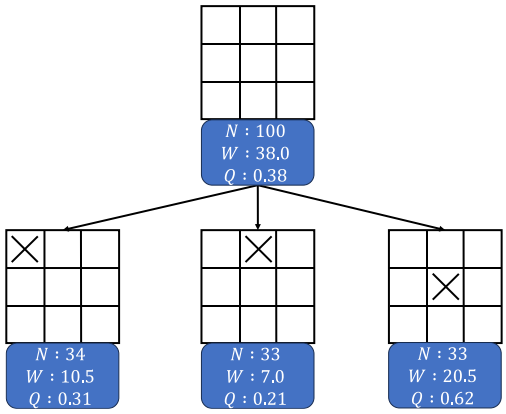

(a) ランダムプレイアウト 1 回後




(b) ランダムプレイアウト 2 回後




(c) ランダムプレイアウト 3 回後




(d) ランダムプレイアウト 100 回後

図 3-11　ランダムプレイアウトを 100 回行う例

図 3-11(a)はランダムプレイアウト 1 回後のゲーム木である。根から探索を行い左の子を選択する。子からランダムプレイアウトを行った結果は勝ちである。したがって、根と左の子の訪問回数がインクリメント（+1）されるとともに、累積報酬に 1 が加算され、期待値が更新される。図 3-11(b)はランダムプレイアウト 2 回後のゲーム木である。真ん中の子からランダムプレイアウトを行った結果は負けである。したがって、根と真ん中の子の訪問回数がインクリメントされるとともに、累積報酬に 0 が加算され、期待値が更新される。図 3-11(c)も同様に 3 回目のランダムプレイアウトが行われ、頂点の統計量が更新される。図 3-11(d)は 100 回のランダムプレイアウト終了後のゲーム木である。左の子から順に期待値が 0.31, 0.21, 0.62 であり右の子が最も良い局面、真ん中の子が最も悪い局面という評価をすることができる。

## 3.5　モンテカルロ木探索（MCTS）

本節では、PMCTS に改良を加えたモンテカルロ木探索（Monte Carlo Tree Search、MCTS）[5], [6]を説明する。

### 3.5.1　MCTS の概念

PMCTS では、ランダムプレイアウトと伝播を用いて状態の良し悪しを評価することを可能にした。しかし、ゲーム木探索の「効率的に良い局面を探索する」という目的の達成のためには、状態の良し悪しを判断だけでなく、良さそうな局面を優先的に探索しなければならない。ゆえに、新たな状態の良し悪しを評価するための探索だけでなく、過去のプレイアウトの結果を活用して良さそうな局面を探索する必要がある。MCTS では新たな状態の探索と過去のプレイアウトの活用を行うために、PMCTS に対して 2 つの新たな要素を追加している。

1 つ目の追加要素は、探索したゲーム木の葉に子を追加することで、ゲーム木を深くすることである。三目並べのゲーム木の葉に子を追加する様子を図 3-12 から図 3-14 に示す。1 回目のプレイアウトを行うために、根から左の子に移動する（図 3-12）。左の子はゲーム木の葉であるため、プレイアウト前に葉に子を追加する（図 3-13）。その後、プレイアウトと伝播を行う（図 3-14）。プレイアウト 6 回目の様子を図 3-15 に示す。ゲーム木に子を付け加えることによって、ゲーム木の深さが深くなり、多くの局面を評価することができる。

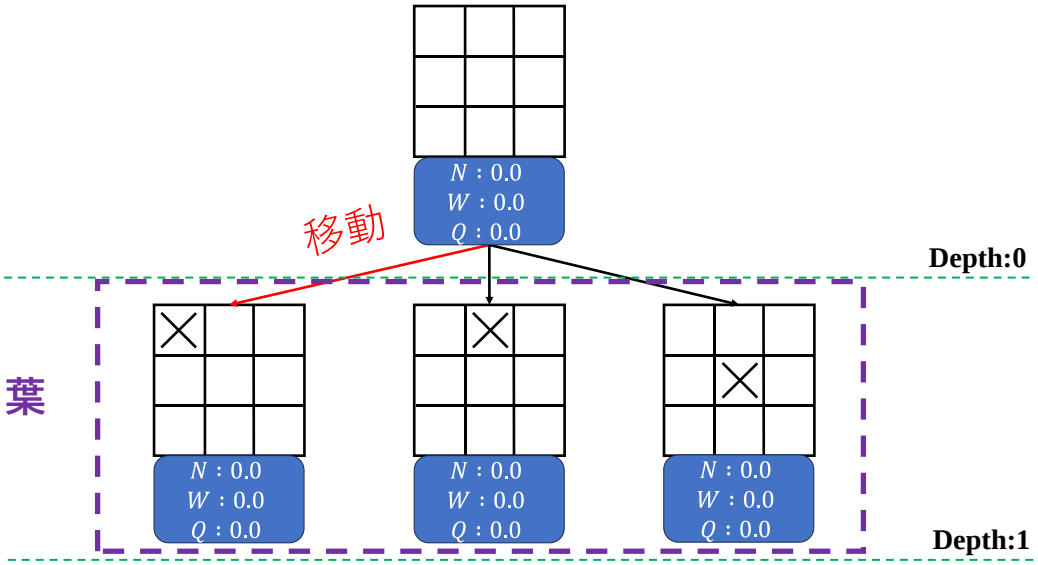



図 3-12　プレイアウトのための移動

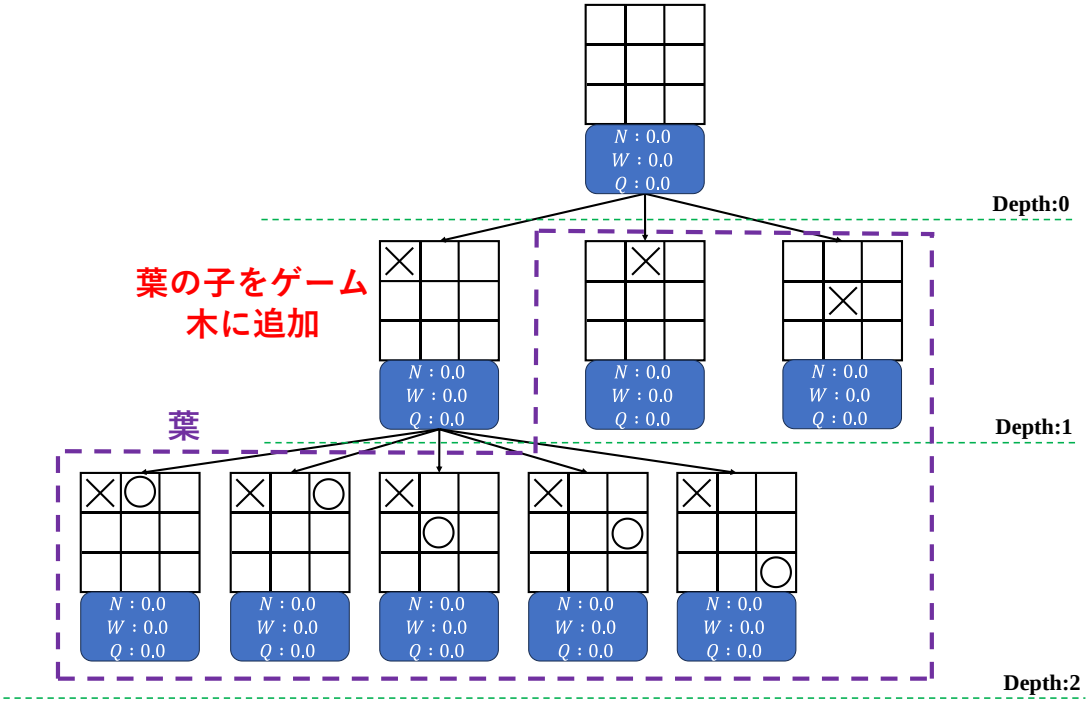



図 3-13　ゲーム木の拡張

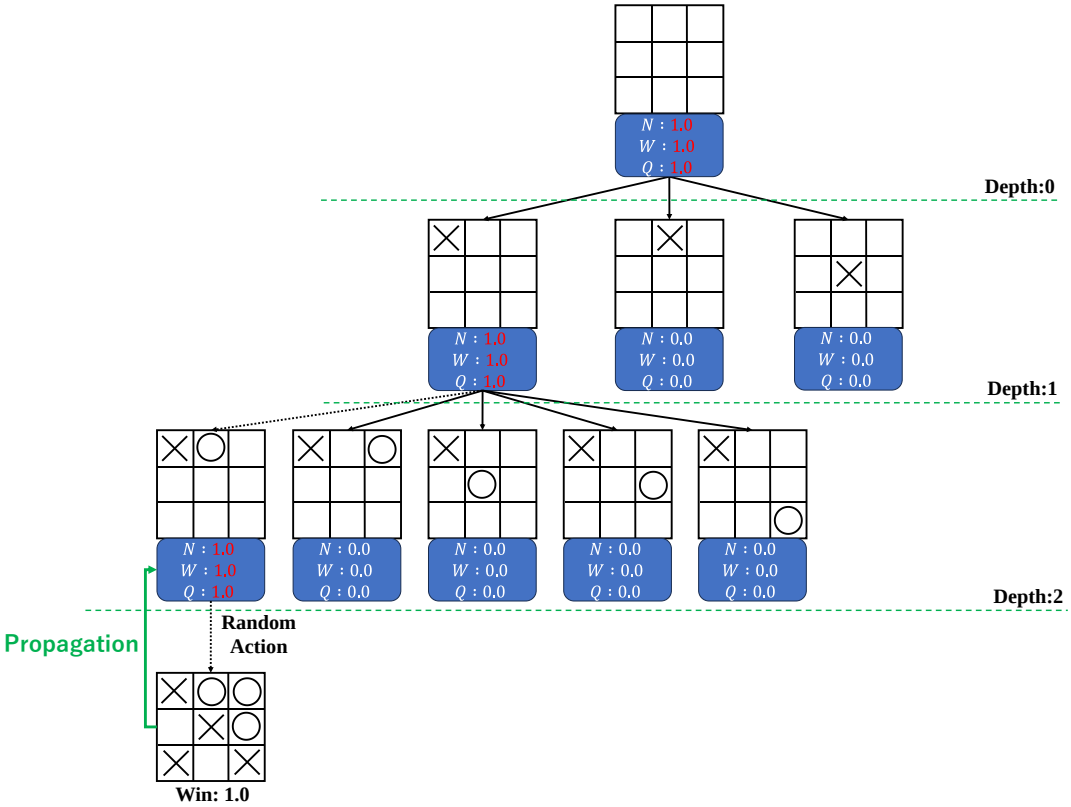



図 3-14　ゲーム木のプレイアウトと伝播

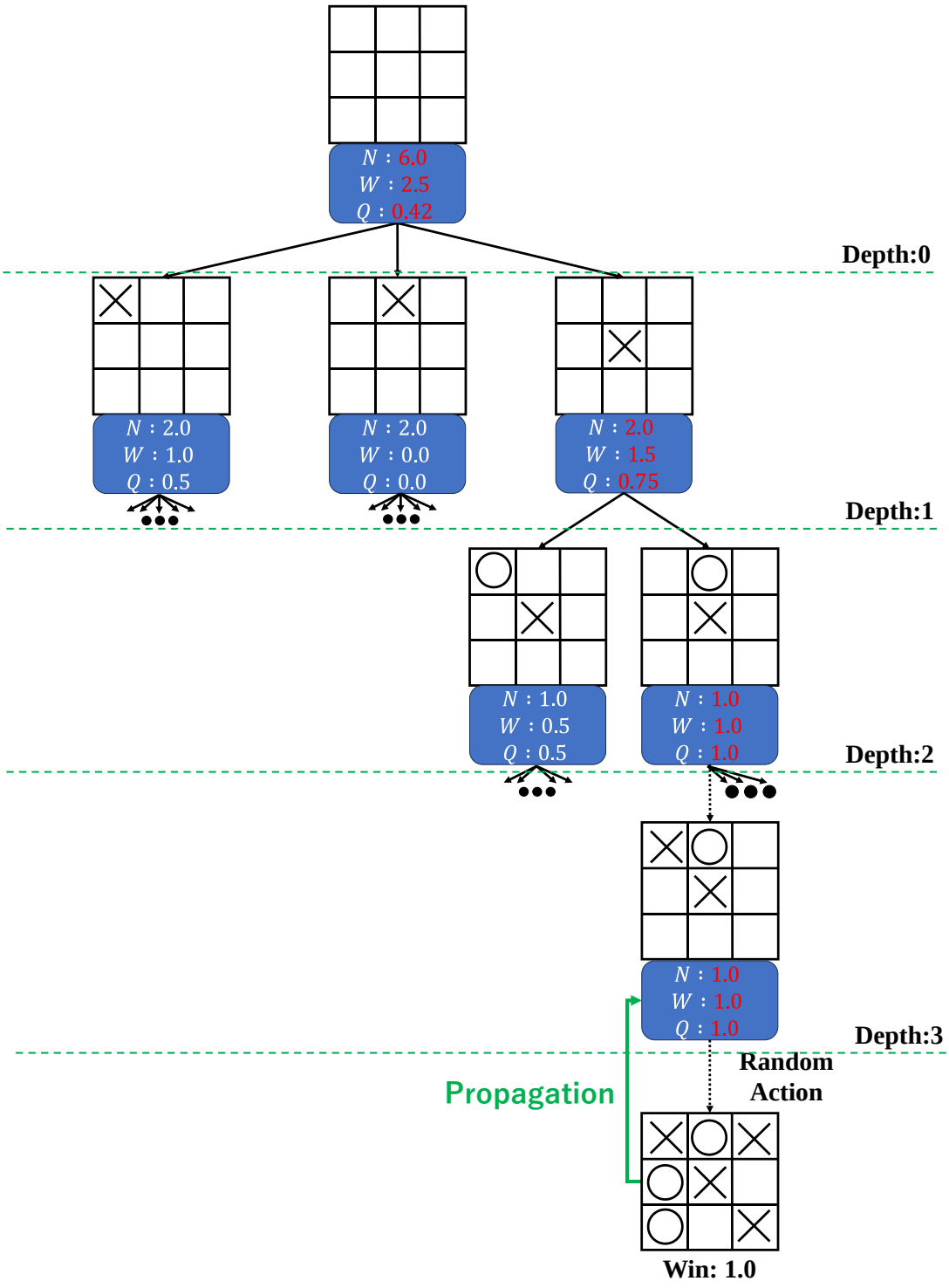



図 3-15　プレイアウト 6 回目の様子

2 つ目の追加要素は、新たな状態を評価する探索と過去のプレイアウトの活用の配分戦略を立てることである。新たな状態の探索（Exploration）に重点を置くと、過去のプレイアウトの結果を用いた探索ができなくなる。一方、過去のプレイアウトの活用（Exploitation）に重点を置くと、新たな状態の訪問回数が減るため状態の評価が不正確になる。以上で述べた探索と活用のジレンマ（Exploration-Exploitation Dilemma）を克服するために、MCTS では 3.5.2 節で説明する Upper Confidence Bounds applied to trees（UCT）を用いている。

### 3.5.2 Upper Confidence Bounds applied to trees（UCT）

MCTS では探索と活用のジレンマを克服するために Upper Confidence Bounds applied to trees（UCT）[6]を用いている。UCT は、多腕バンディット問題[7]に基づくアルゴリズムである。多腕バンディット問題とは、「$K$本のスロットのアームと$T$枚のコインがある。各スロットには 1 枚のコインを入れた時に正の利得が得られるが、利得の期待値を知ることはできない。この条件下で$T$枚のコインを用いて利得を最大にする」という問題である。利得を最大にするためには、期待値が高いアームを探すためにアームを引くこと（探索）と期待値が高そうなアームを引くこと（活用）をバランス良く行うことが必要であり、新たな状態を評価する探索と過去のプレイアウトを活用した探索をバランス良く行うという木探索の戦略と対応している。

多腕バンディット問題を解くアルゴリズムとして、Upper Confidence Bounds（UCB）[8]が広く使われている。UCB は UCB 値を定義し、UCB 値が最大となるアームを選ぶ方法である。UCB 値の定義を式(1.3)に示す。

$$\bar{X}_i + C_p\sqrt{\frac{\ln t}{N_i}} \tag{3.6}$$

ここで、$\bar{X}_i$はこれまでのアーム$i$の利得の期待値、$N_i$はこれまでにアーム$i$を引いた回数、$t$はアームを引いた総和（$t \coloneqq \sum_{i=1}^{K} N_i$）、$C_p$は探索と活用のバランスを決める正の定数である。アーム$i$の期待値が大きい場合、第 1 項が大きくなり、アーム$i$を引く回数がほかのアームを引いた回数と比べて相対的に少ない場合、第 2 項が大きくなる。$C_p$が大きな正の値の場合、期待値が高いアームを探すために、これまでに引いた回数が少ないアームを選択して引くようになる（探索）。一方、$C_p$が小さな正の値の場合、これまでの探索において期待値が高いアームを引くようになる（活用）。

UCT は UCB を木探索に応用したアルゴリズムである。探索中の頂点を$s$とすると、詳細に探索を行う（木を展開して深掘りする）ために移動する子の頂点$c^*$は式(3.7)で表される。

$$c^* = \underset{c \in \text{children of } s}{\arg\max} \left( Q(c) + C_p\sqrt{\frac{\ln N(s)}{N(c)}} \right) \tag{3.7}$$

ここで、$Q(c)$はこれまでの探索で得られた頂点$c$の期待値、$N(c)$はこれまでの探索で頂点$c$を

訪問した回数、$N(s)$は頂点$s$を訪問した回数である。また、$C_p$は探索と活用のバランスを決める正の定数である。期待値が大きい場合、第 1 項が大きくなり、頂点$c$の探索回数が少ない場合、第 2 項が大きくなる。$C_p$が大きな正の値の場合、探索回数が少ない子が選択されるようになる（探索）。一方、$C_p$が小さな正の値の場合、期待値が大きい子が選択される（活用）。

MCTS では UCT を用いることで、期待値が大きい子を選択しながら、訪問回数が少ない子もバランス良く選択する。注意すべきことは、UCT はすべての子を 1 回以上訪れているときにのみ定義できることである。したがって、訪問回数が 0 の子がある場合、その頂点を優先的に選択する。三目並べにおける UCT による選択の例を図 3-16 に示す。子の UCT 値は左の子から順に 0.68, 0.58, 0.99 である。したがって、UCT 値が最大である右の子を次に探索する状態として選択する。

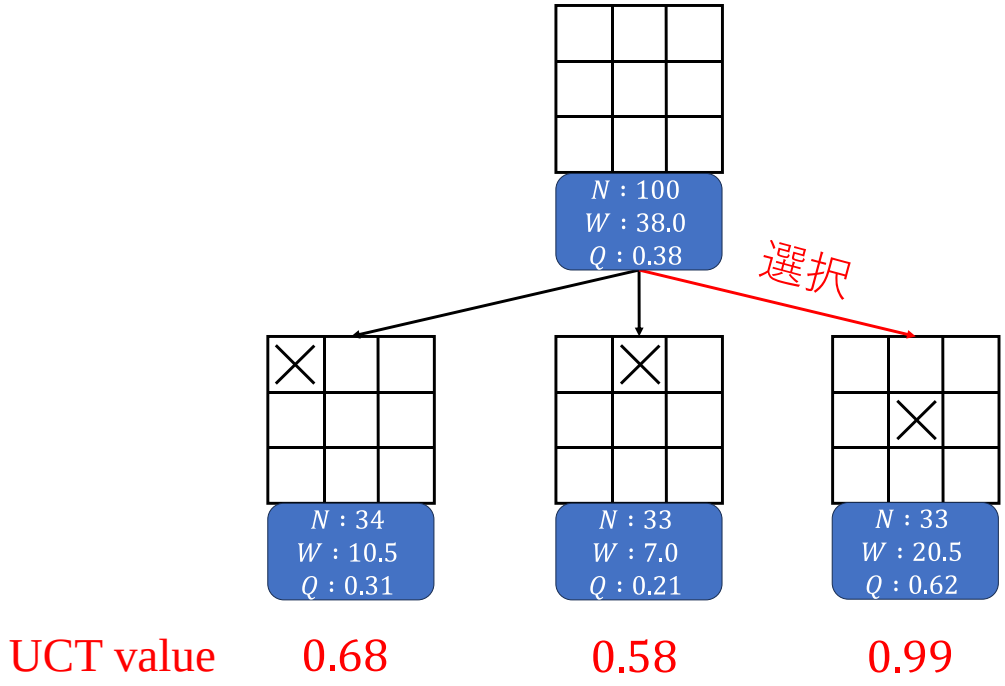



図 3-16　UCT による選択の例（ただし、$C_p = 1.0$）

### 3.5.3 MCTS のアルゴリズム

本項では MCTS のアルゴリズムを説明する。

MCTS のアルゴリズムは図 3-17 に示すメカニズム従って、ゲーム木の探索と構成を行う。MCTS は、探索の間、選択（Selection）、展開（Expansion）、シミュレーション（Simulation）、バックプロパゲーション（Backpropagation）の 4 つのステップを繰り返す。これらのステップを以下で説明する。

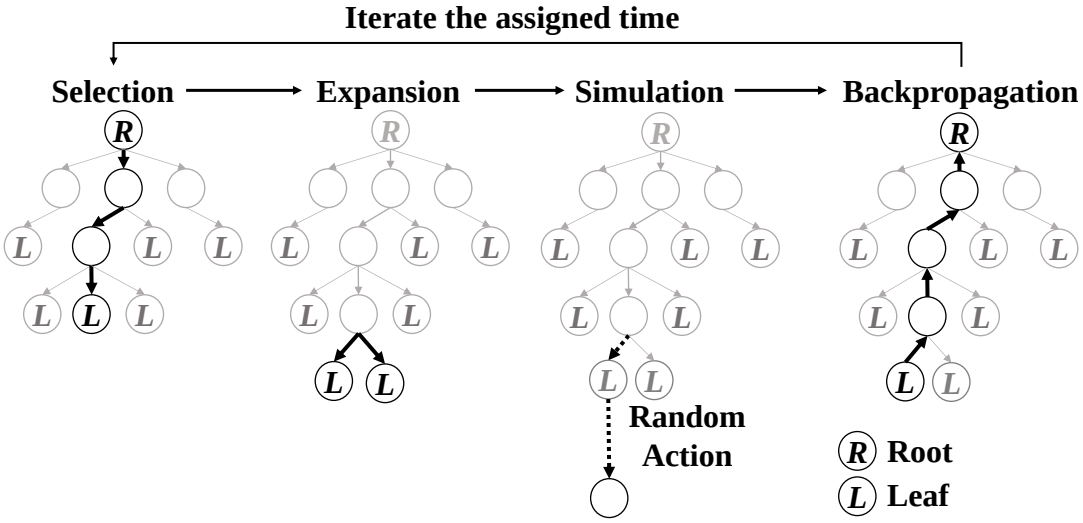



図 3-17　モンテカルロ木探索のメカニズム

**選択（Selection）**：選択では、根を始まりとして葉にたどり着くまで、UCT に基づいて子を選び続ける。三目並べにおける選択の図 3-18 に示す。根の状態から UCT に基づいて右の子を選び、深さが 0 の状態から深さが 1 の状態に移動する。深さ 1 の移動した状態から UCT に基づいて右の子を選び、深さが 1 の状態から深さが 2 の状態に移動する。深さ 2 の移動した状態から UCT に基づいて左の子を選び、深さが 2 の状態から深さが 3 の状態に移動する。深さが 3 の移動した状態は子を持っていない（葉）ため選択は終了する。

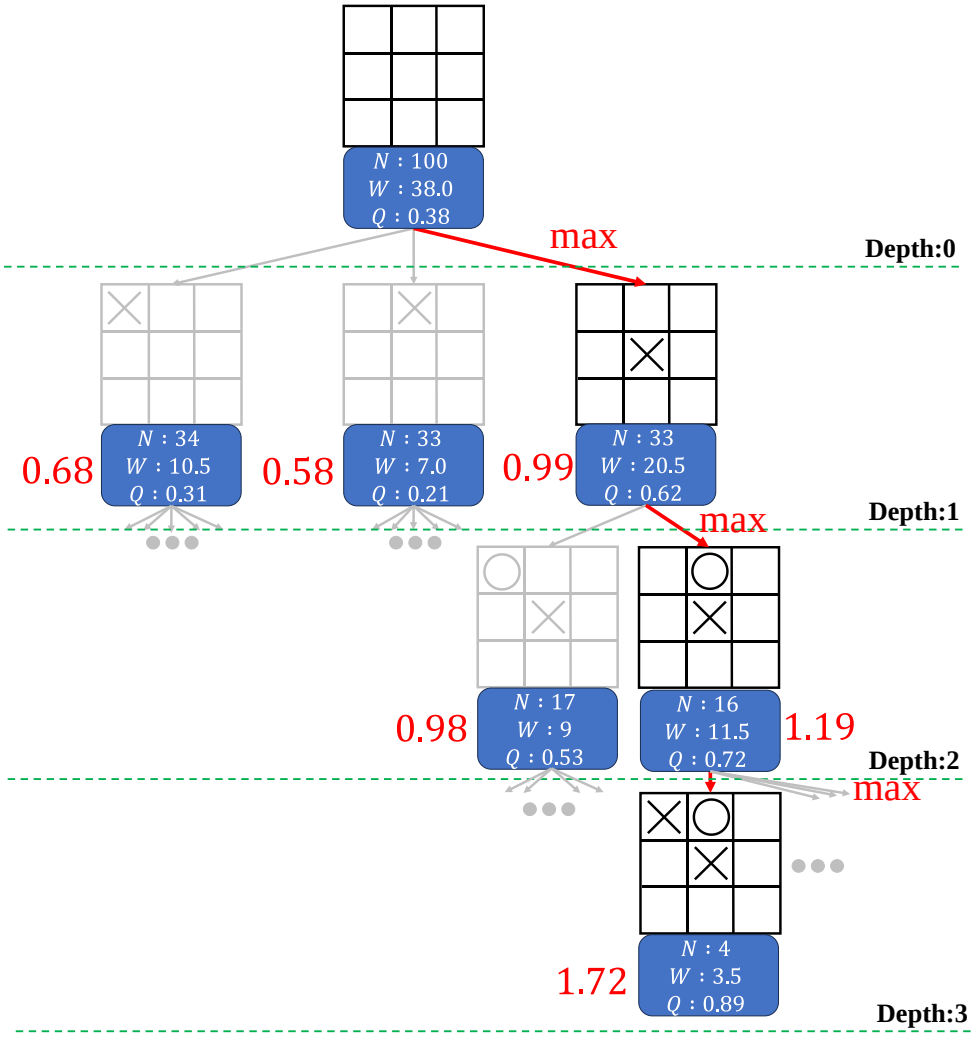



図 3-18　三目並べにおける選択の例（赤字が UCT 値。ただし、$C_p = 1.0$）

**展開（Expansion）**：展開では、選択でたどり着いた葉に対して、新たな子をゲーム木に追加する。選択でたどり着いた葉が、子を定義できない頂点であれば、展開は行われない。三目並べにおける展開の例を図 3-19 に示す。選択で選ばれた深さ 3 の葉の状態から実現可能な状態を子として木に追加する。

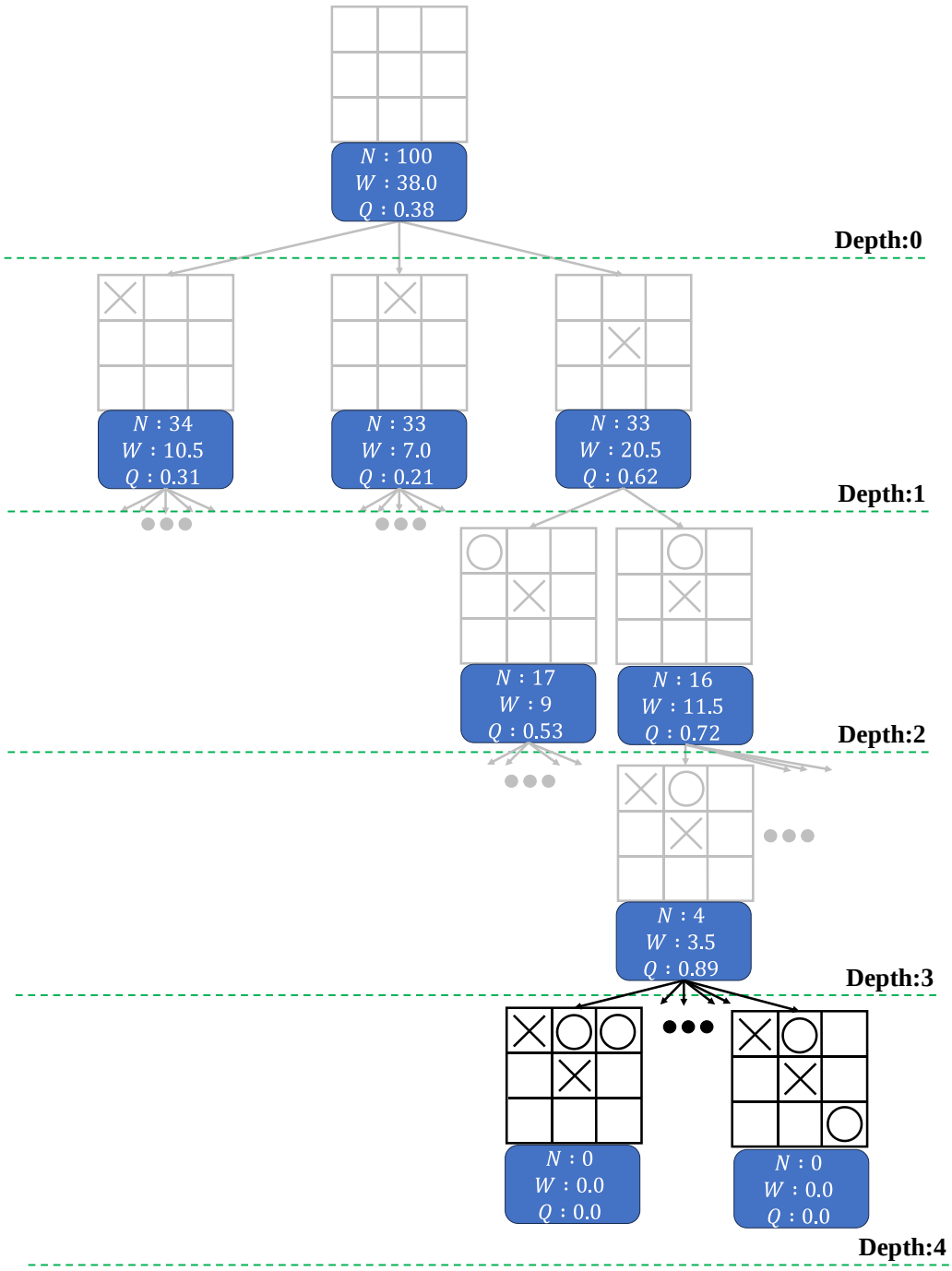



図 3-19　三目並べにおける展開の例

**シミュレーション（Simulation）**：シミュレーションでは、選択でたどり着いた頂点から子が定義できない頂点までランダムに行動し続ける（ランダムプレイアウト）。子が定義できない頂点は評価され、報酬$r$が得られる。三目並べにおけるシミュレーションの例を図 3-20 に示す。選択された葉の状態から終了の状態まで行動を続け、終了の状態から報酬$r$を獲得する。

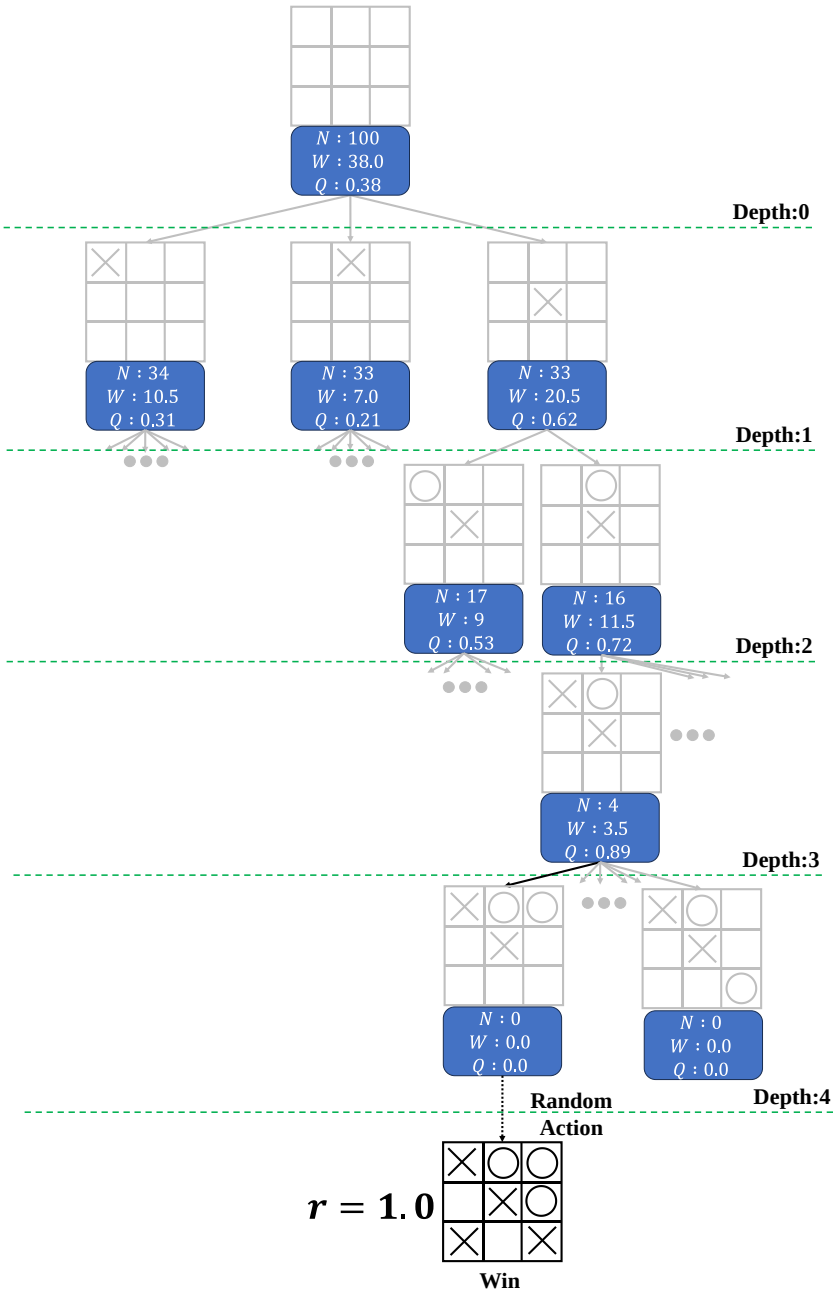



図 3-20　三目並べにおけるシミュレーションの例

**バックプロパゲーション（Backpropagation）**：新たに葉になった頂点から根までの経路に沿って、頂点の訪問回数がインクリメントされるとともに、累積報酬と期待値が更新される。頂点の統計量の更新式を式(3.8)、式(3.9)、式(3.10)に示す。ここで$s$は新たに葉になった頂点から根までの経路に沿った全ての頂点の要素である。

$$N(s) = N(s) + 1 \tag{3.8}$$

$$W(s) = W(s) + r \tag{3.9}$$

$$Q(s) = W(s)/N(s) \tag{3.10}$$

三目並べにおけるバックプロパゲーションの例を図 3-21 に示す。シミュレーションで得た報酬$r$に基づいて、選択でたどり着いた状態から根の状態までの統計量を式(3.8)、式(3.9)、式(3.10)を用いて更新する。

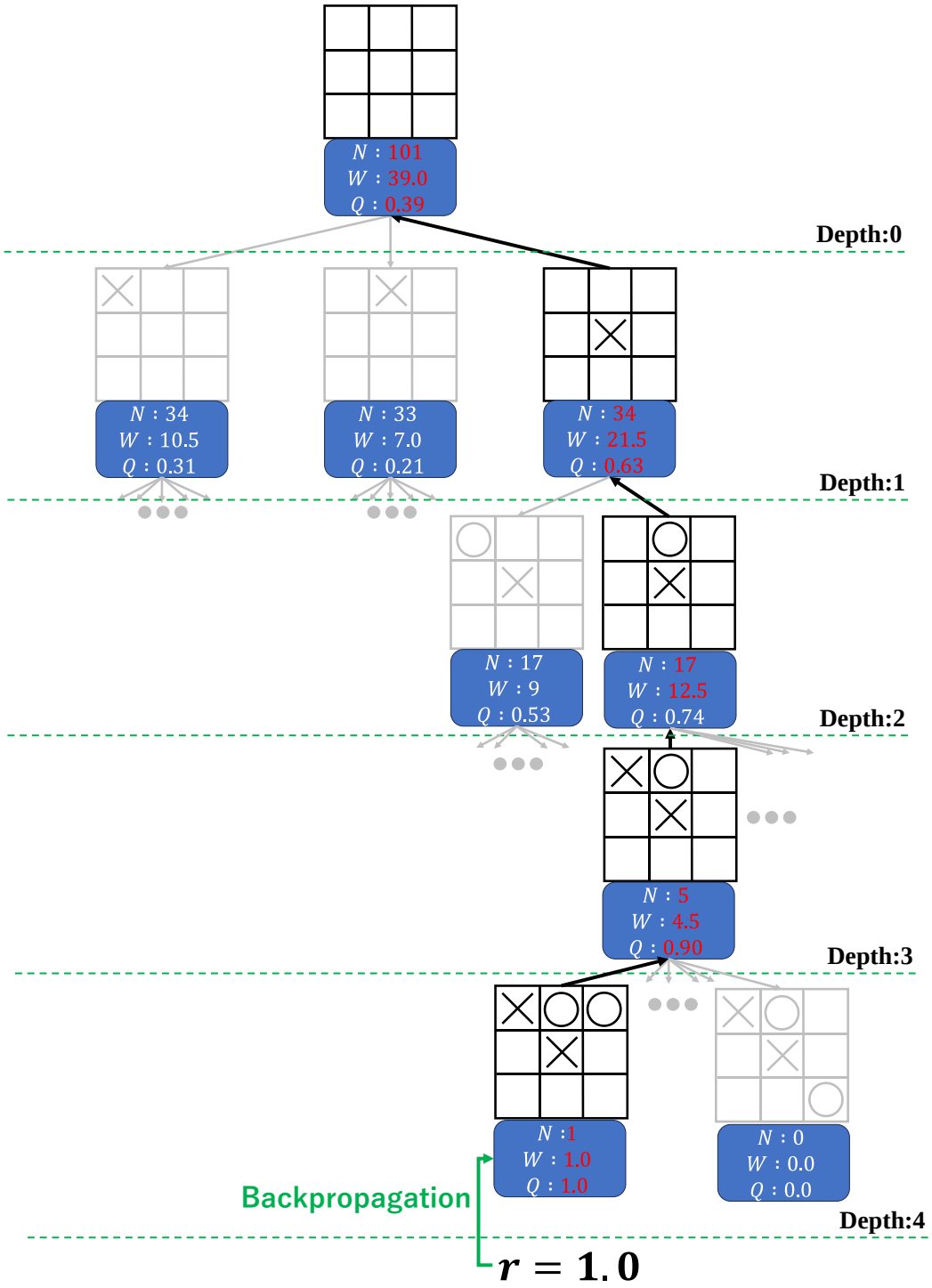



図 3-21　三目並べにおけるバックプロパゲーションの例

以上の 4 つのステップ（選択、展開、シミュレーション、バックプロパゲーション）を決められた回数繰り返すことによって、良い状態を効率的に探索している。

## 3.6　本章のまとめ

本章では、木、ゲーム木探索、木探索手法である原始モンテカルロ木探索（PMCTS）とモンテカルロ木探索（MCTS）の説明を行った。

3.2 節では、木とは長さ 1 以上の閉路を持たない連結な無向グラフであることを述べた。

3.3 節では、ゲームの局面を有向辺で結ぶことでゲーム木が構成されることを説明した。ゲーム木探索の目的は、効率的に良い状態を探すことであることを述べた。

3.4 節では、三目並べを例にモンテカルロ法を利用した木探索手法である PMCTS の説明をした。3.4.1 項では、PMCTS ではランダムプレイアウトと伝播を用いて状態の良し悪しを評価することを述べ、3.4.2 項でグラフ理論を用いて PMCTS のアルゴリズムを解説した。PMCTS では頂点が統計量を持ち、統計量の更新を行うことで状態の良し悪しを評価できる。しかし、PMCTS では状態の良し悪しの評価を行うのみで、効率的に良い状態を探索することができないという欠点がある。

3.5 節では、三目並べを例にして、PMCTS を改良した MCTS の説明をした。3.5.1 項では、探索と活用のジレンマを克服するために PMCTS に対して 2 つの新たな要素を追加していることを述べた。1 つ目は、ゲーム木に葉の子を追加し、木を深くすることである。2 つ目は、新たな頂点の探索と過去のランダムプレイアウトの結果を活用した探索の配分戦略を決めることである。3.5.2 項では、Upper Confidence Bounds applied to trees（UCT）を用いて探索と活用のジレンマを克服する方法を解説した。3.5.3 項では、モンテカルロ木探索が 4 つのステップ（選択、展開、シミュレーション、バックプロパゲーション）の繰り返しによって、良い状態を効率的に探すことを述べた。

第 4 章では、本章で説明を行った MCTS を用いた燃料装荷パターン最適化について説明する。

## 3.7　参考文献


[1] J. A. Bondy and U. S. R. Murty, *Graph theory*, Graduate texts in mathematics, no. 244. New York, NY, USA: Springer, 2008

[2] 茨木俊秀, 永持仁, 石井利昌, *グラフ理論 : 連結構造とその応用*, 東京, 朝倉書店, 2010

[3] M. Albert, R. Nowakowski, and D. Wolfe, *Lessons in play: an introduction to combinatorial game theory*, Boca Raton, FL, USA: CRC Press, 2007 (訳: 川辺治之, *組合せゲーム理論 : 勝利の方程式*, 東京, 共立出版, 2011).

[4] 青木栄太, *ゲームで学ぶ探索アルゴリズム実践入門 : 木探索とメタヒューリスティクス*, 東京, 技術評論社, 2023

[5] R. Coulom, “Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search,” in *Computers and Games*, vol. 4630, H. J. Van Den Herik, P. Ciancarini, and H. H. L. M. Donkers, Eds., in Lecture Notes in Computer Science, vol. 4630, Berlin, Heidelberg: Springer Berlin

Heidelberg, 2007, pp. 72–83. doi: 10.1007/978-3-540-75538-8_7.

[6] L. Kocsis and C. Szepesvári, “Bandit Based Monte-Carlo Planning,” in *Machine Learning: ECML 2006*, vol. 4212, J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, Eds., in Lecture Notes in Computer Science, vol. 4212, Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 282–293. doi: 10.1007/11871842_29.

[7] T. L. Lai and H. Robbins, “Asymptotically Efficient Adaptive Allocation Rules,” *Adv. in Appl. Math.*, vol 6, no 1, pp 4–22, Mar. 1985, doi: 10.1016/0196-8858(85)90002-8.

[8] P. Auer, N. Cesa-Bianchi, and P. Fischer, “Finite-Time Analysis of the Multiarmed Bandit Problem,” *Mach. Learn.*, vol. 47, no. 2/3, pp. 235–256, 2002, doi: 10.1023/a:1013689704352.

# 第4章　MCTS の燃料装荷パターン最適化への適用

## 4.1　本章の概要

本章では、モンテカルロ木探索（MCTS）を燃料装荷パターン最適化に適用する方法と適用した結果について述べる。

4.2 節では、MCTS を燃料装荷パターン最適化に適用する方法を説明する。4.2.1 項では、燃料装荷パターン最適化をゲーム木に変換する方法を説明する。4.2.2 項では、4.2.1 項で作成した燃料装荷パターン最適化のゲーム木に対して MCTS を適用する方法を説明する。

4.3 節では、燃料装荷パターン最適化の計算条件を示す。

4.4 節では、MCTS を用いた燃料装荷パターン最適化の結果と従来手法を用いた燃料装荷パターン最適化の比較結果を記載する。

4.5 節では、MCTS による燃料装荷パターン最適化の特徴について得られた結果を用いて議論する。

4.6 節では、本章のまとめを述べる。

## 4.2　MCTS を用いた燃料装荷パターン最適化手法

本節では、燃料装荷パターン最適化をゲーム木探索に帰着させる方法と燃料装荷パターンのゲーム木に対する MCTS の適用方法を説明する。

### 4.2.1　燃料装荷パターン最適化の木構造設計

本項では、燃料装荷パターン最適化をゲーム木に変換する方法を説明する。

燃料装荷パターン最適化をゲーム木探索に変換するために、燃料装荷パターン最適化を燃料集合体が装荷されていない空の炉心に 1 体ずつ燃料を装荷する手順の最適化とみなす。例えば、26 の装荷位置がある 1/8 対称炉心では、26 回の燃料集合体の装荷から手順が構成される。炉心の中心から装荷を行う手順の一例を図 4-1 を示す。

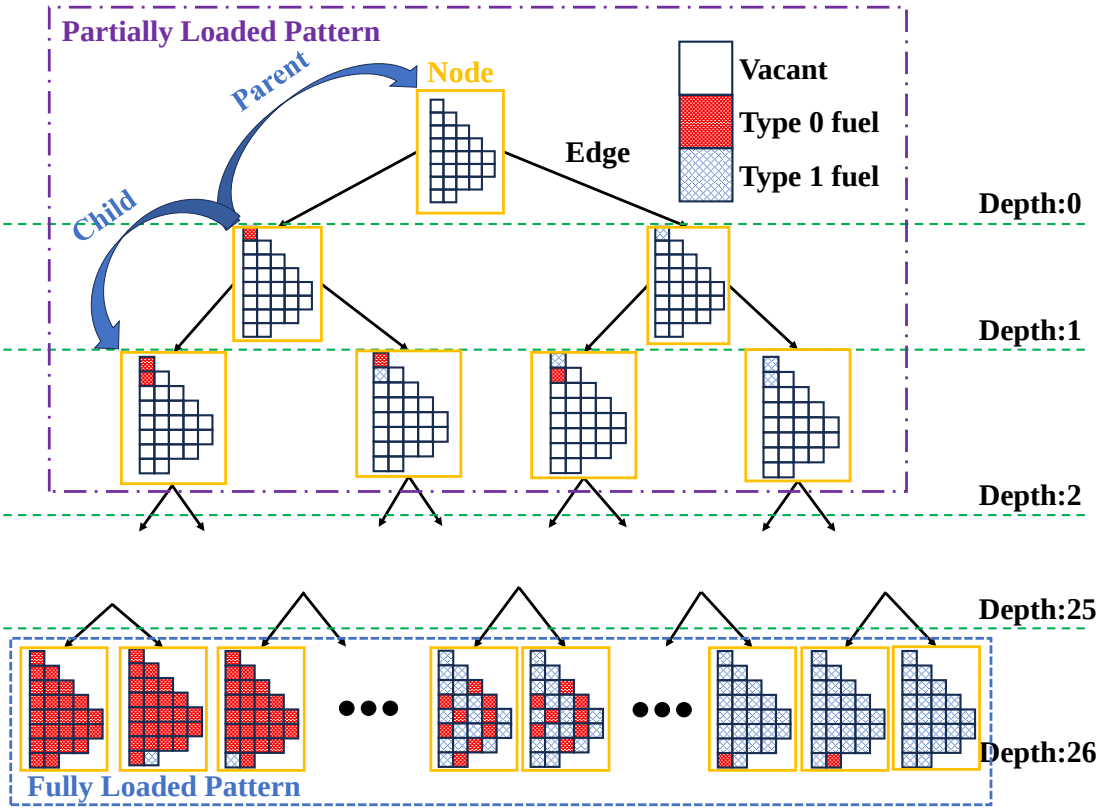
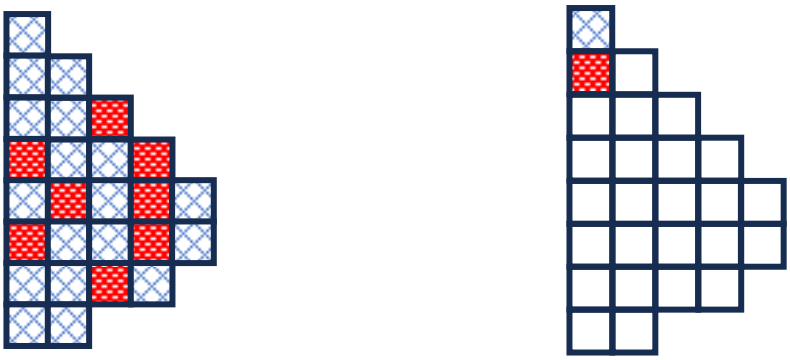



図 4-1　最適化手順の一例
燃料集合体の種類は 2 つ、内側から装荷を行う場合

この手順は、頂点が燃料装荷パターン、辺が燃料集合体の装荷に対応するゲーム木を構成する。ここで、燃料装荷パターンは、燃料集合体が装荷されていない燃料装荷位置があるパターンも含んでいる。ゲーム木の根は、燃料集合体が全く装荷されていない空の燃料装荷パターンである。燃料装荷パターンの子は、手順に従って 1 体の燃料集合体を追加で装荷した燃料装荷パターンである。子が存在する、つまり、すべての燃料装荷位置に燃料集合体が装荷されている燃料装荷パターンを完全な燃料装荷パターン（Fully Loaded Pattern）と定義する。また、子が定義できる、つまり、燃料集合体が装荷されていない燃料装荷位置がある燃料装荷パターンを部分的な燃料装荷パターン（Partially Loaded Pattern）と定義する。完全な燃料装荷パターンと部分的な燃料装荷パターンの例を図 4-2 に示す。



(a)　完全な燃料装荷パターン　　(b)　部分的な燃料装荷パターン

図 4-2　完全な燃料装荷パターンと部分的な燃料装荷パターンの例

### 4.2.2　探索アルゴリズム

本項では、4.2.1 項で作成した燃料装荷パターン最適化のゲーム木に対して MCTS を適用する方法を説明する。燃料装荷パターン最適化では、第 4 章で説明した MCTS による三目並べの探索と同様に、ランダムプレイアウトと伝播を用いた燃料装荷パターンの評価、ゲーム木の展開、及び、探索と活用の配分戦略を決めることによって効率的な探索を実現している。

燃料装荷パターン最適化のゲーム木の頂点$s$は燃料装荷パターン、辺$a$は燃料集合体の装荷に対応する。また、それぞれの頂点は統計量$\{N(s), W(s), Q(s)\}$を持つ。ここで、$N(s)$は訪問回数（探索で燃料装荷パターン$s$を訪れた回数）、$W(s)$は累積報酬（伝播された完全な燃料装荷パターンの報酬の合計値）、$Q(s)$は期待値（伝播された完全な燃料装荷パターンの報酬の期待値、つまり、$W(s)/N(s)$）である。MCTS における報酬と組合せ最適化における目的関数は、どちらも燃料装荷パターンの性能を表す指標であるため同じ意味を持つ。したがって、完全な燃料装荷パターンの報酬は、ピーキング係数、サイクル長、取出平均燃焼度などの炉心特性によって定義される。ただし、組合せ最適化は目的関数の最大化または最小化を目的とするが、MCTS における報酬は最大化のみを目的としているため、期待値$Q(s)$が大きい燃料装荷パターン$s$が良好な燃料装荷パターンである。

燃料装荷パターン最適化における MCTS のメカニズムを図 4-3 に示す。MCTS は、燃料装荷パターン最適化の間、選択（Selection）、展開（Expansion）、シミュレーション（Simulation）、バックプロパゲーション（Backpropagation）の 4 つのステップを繰り返す。これらのステップを以下で説明する。

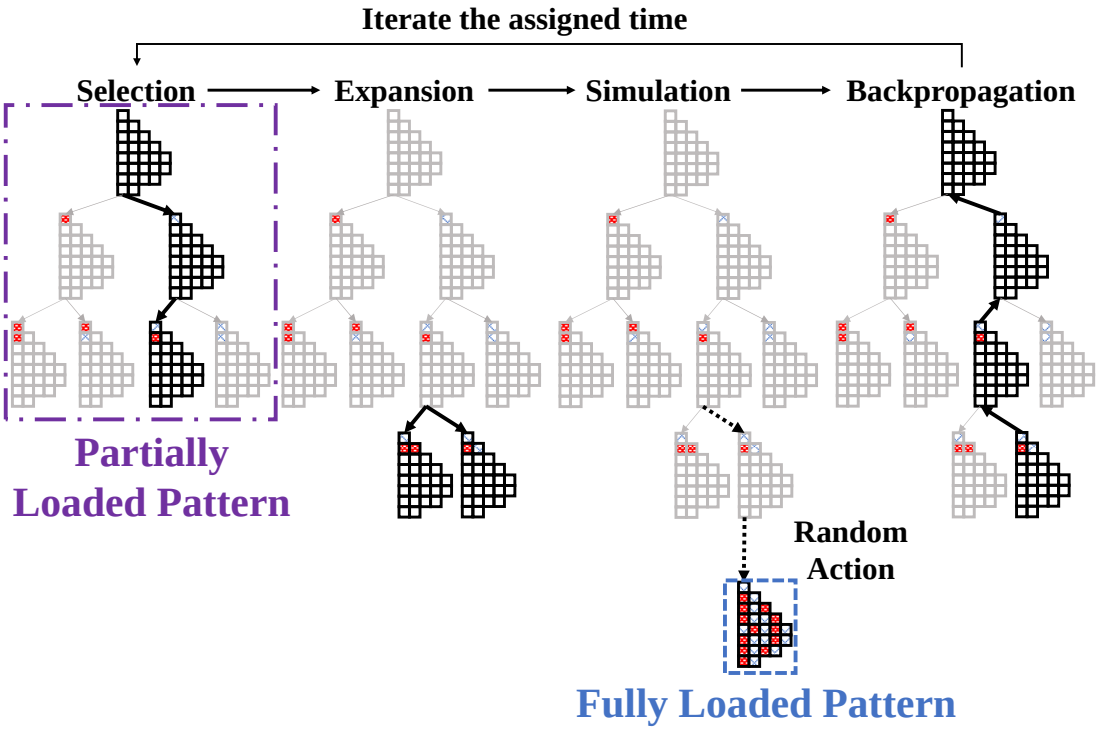



図 4-3　燃料装荷パターン最適化における MCTS のメカニズム
炉心の中心から装荷を行い、装荷する燃料集合体が 2 種類の場合

**選択（Selection）**：選択では、根の燃料装荷パターン（全く装荷されていない燃料装荷パターン）を始まりとして、葉の燃料装荷パターン（子を持たない部分的な燃料装荷パターン）にたどり着くまで、3.5.2 項で説明した Upper Confidence Bounds applied to trees (UCT)に基づいて子の燃料装荷パターン（部分的な燃料装荷パターンに対して燃料集合体を 1 体だけ追加で装荷した燃料装荷パターン）を選び続ける。探索中の燃料装荷パターンを$s$とすると、次に探索する子の燃料装荷パターン$c^*$は式(4.1)で与えられる。

$$c^* = \underset{c \in \text{children of } s}{\arg\max} \left( Q(c) + C_p \sqrt{\frac{\ln N(s)}{N(c)}} \right) \tag{4.1}$$

ここで、$Q(c)$はこれまでの探索で得られた燃料装荷パターン$c$の期待値、$N(c)$はこれまでの探索で燃料装荷パターン$c$を訪問した回数、$N(s)$は燃料装荷パターン$s$を訪問した回数である。また、$C_p$は探索と活用のバランスを決める正の定数である。期待値が大きい場合、第 1 項が大きくなり、頂点$c$の探索回数が少ない場合、第 2 項が大きくなる。$C_p$が大きな正の値の場合、探索回数が少ない燃料装荷パターンが選択されるようになる（探索）。一方、$C_p$が小さな正の値の場合、期待値が大きい燃料装荷パターンが選択される（活用）。UCT によって、期待値が大きい燃料装荷パターンを選択しながら、訪問回数が少ない燃料装荷パターンもバランス良く探索することができる。

燃料装荷パターン最適化における選択の例を図 4-4 に示す。選択は根の燃料装荷パターン（全く装荷されていない燃料装荷パターン）から探索が始まる。根の燃料装荷パターンの子を図 4-5 に示す。根の燃料装荷パターンの子は、炉心の中心に 1 体だけ燃料を装荷した燃料装荷パターンである。右の子の UCT 値が大きいと仮定すると、右の子の燃料装荷パターンが選択される。ここで、UCT 値は式(4.2)で与えられる。

$$Q(c) + C_p \sqrt{\frac{\ln N(s)}{N(c)}} \tag{4.2}$$

同様にして、選択された深さ 1 の燃料装荷パターンから UCT によって左の子の燃料装荷パターンが選択されたとすると、新たに選択された深さ 2 の燃料装荷パターンは葉の燃料装荷パターン（子を持たない部分的な燃料装荷パターン）であるため、選択は終了する。

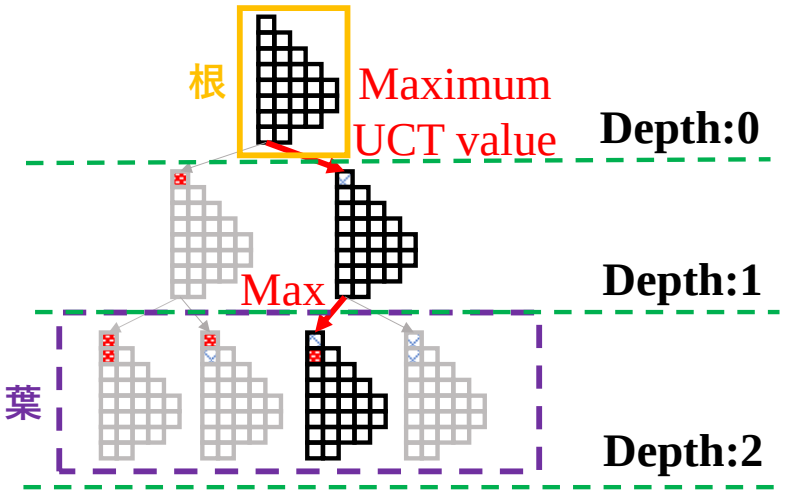



図 4-4　燃料装荷パターン最適化における選択の例
炉心の中心から装荷を行い、装荷する燃料集合体が 2 種類の場合

(a)　図 4-4 における左の子　　(b)　図 4-4 における右の子

図 4-5　根の子の燃料装荷パターン

**展開（Expansion）**：展開では、選択でたどり着いた葉の燃料装荷パターンに対して、子の燃料装荷パターンを木に追加する。選択でたどり着いた葉の燃料装荷パターンが、完全な燃料装荷パターンであれば、展開は行われない。展開によって、木を深くし、期待値を評価する燃料装荷パターンを増やすことができる。

燃料装荷パターン最適化における展開の例を図 4-6 に示す。選択でたどり着いた燃料装荷パターン（深さ 2 の燃料装荷パターンのうち右から 2 番目）の子を木に追加する。この燃料装荷パターンは、選択でたどり着いた燃料装荷パターンに 1 体だけ追加で燃料集合体を装荷した燃料装荷パターンである。

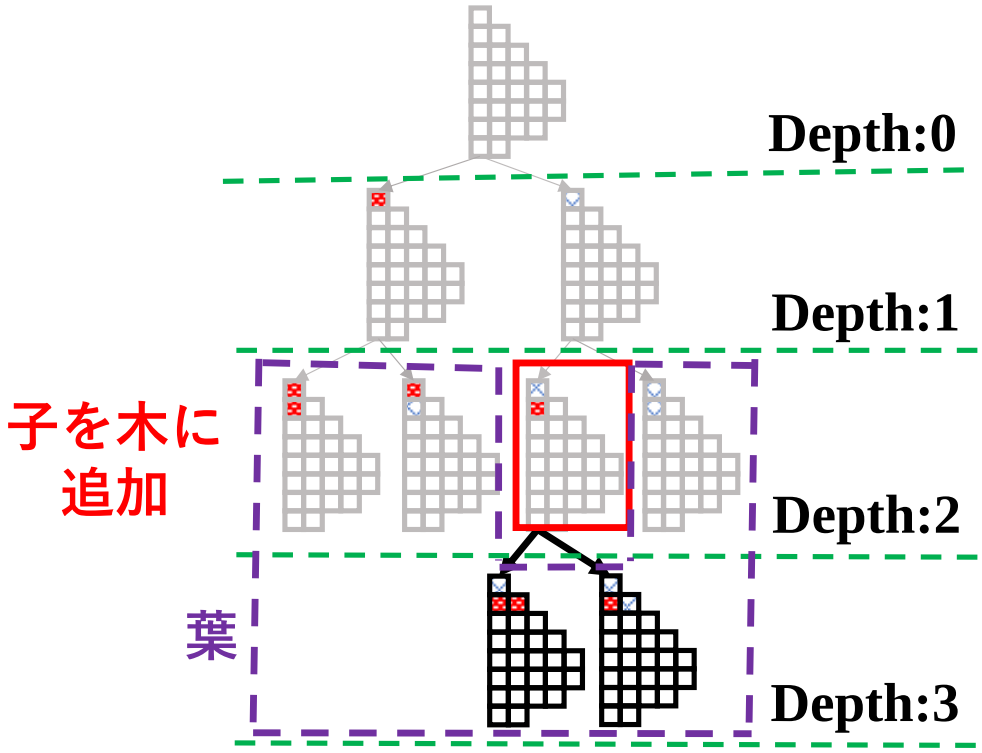



図 4-6　燃料装荷パターン最適化における展開
装荷する燃料集合体が 2 種類の場合

**シミュレーション（Simulation）**：シミュレーションでは、たどり着いた葉の燃料装荷パターンから完全な燃料装荷パターンまでランダムに子の燃料装荷パターンを選び続ける（ランダムに燃料装荷を行う）。完全な燃料装荷パターンは炉心解析コードで炉心特性が評価され、得られた炉心特性値から報酬$r$が得られる。

燃料装荷パターン最適化におけるシミュレーションの例を図 4-7 に示す。選択でたどり着いた燃料装荷パターンに内側から 1 体ずつランダムに燃料を装荷していく。図 4-7 では、24 回のランダムな燃料装荷後に得られる完全な燃料装荷パターンの炉心特性を計算し、報

酬$r$を得ている。

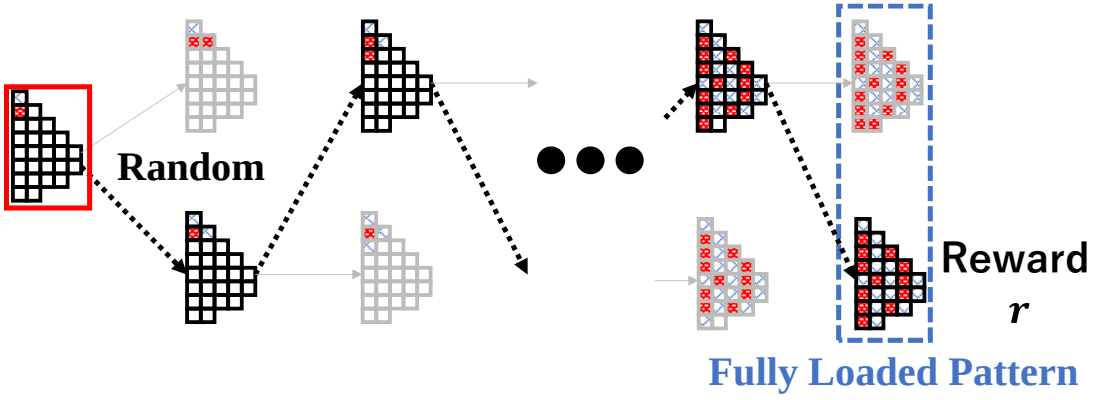



図 4-7　燃料装荷パターン最適化におけるシミュレーションの例
装荷する燃料集合体が 2 種類の場合

**バックプロパゲーション（Backpropagation）**：バックプロパゲーションでは、新たに葉になった燃料装荷パターンから根までの経路に沿って、頂点の訪問回数がインクリメント（+1）されるとともに、累積報酬と期待値が更新される。更新式を式(4.3)、式(4.4)、式(4.5)に示す。ここで、$s$は新たに葉になった燃料装荷パターンから根の燃料装荷パターンまでの経路に沿ったすべての頂点の要素である。

$$N(s) = N(s) + 1 \tag{4.3}$$

$$W(s) = W(s) + r \tag{4.4}$$

$$Q(s) = W(s)/N(s) \tag{4.5}$$

燃料装荷パターン最適化におけるバックプロパゲーションの例を図 4-8 に示す。新たに葉になった燃料装荷パターンから根の燃料装荷パターンまでの経路に沿った頂点の統計量が更新される。図 4-8 では新たに葉になった燃料装荷パターンの深さが 3 のため、4 つの燃料装荷パターン（経路の含まれている深さが 3, 2, 1, 0 の燃料装荷パターン）の統計量が更新される。

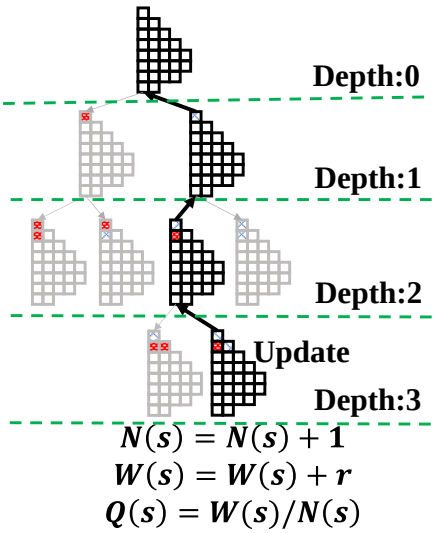



図 4-8　燃料装荷パターン最適化におけるバックプロパゲーションの例
装荷する燃料集合体が 2 種類の場合

以上の 4 つのステップ（選択、展開、シミュレーション、バックプロパゲーション）を決められた回数繰り返すことによって、良好な燃料装荷パターンを効率的に探索している。探索中に得られた最良の報酬を持つ完全な燃料装荷パターンを最適な燃料装荷パターンとする。

### 4.3　計算条件

本節では、MCTS による燃料装荷パターン最適化を行う計算条件を述べる。

本研究で用いる炉心体系は図 4-9 に示す PWR 炉心を模擬している。炉心体系を図 4-10 に示す。図 4-9 に示す PWR 炉心は燃料装荷位置が 157 あるが、1/8 対称性を考えることで 26 となる。また、図 4-10 に示す燃料装荷位置の番号は、燃料集合体装荷位置の番号は、炉心中心からの距離によって定義される。燃料装荷位置の外縁には水とバッフル板が存在し、炉心の外側は真空であると仮定する。

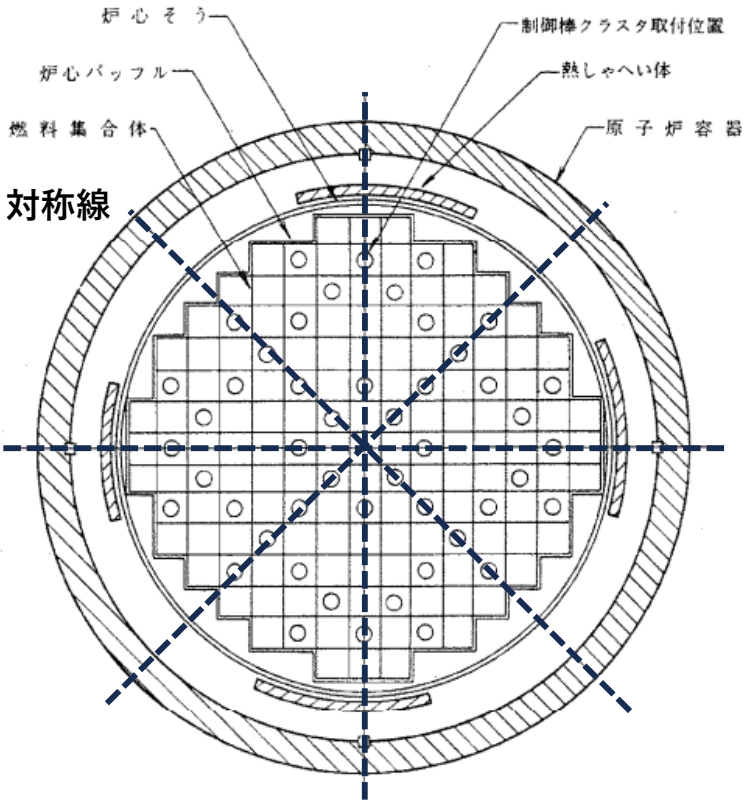



図 4-9　本研究で模擬する PWR 炉心

[1]を一部改訂

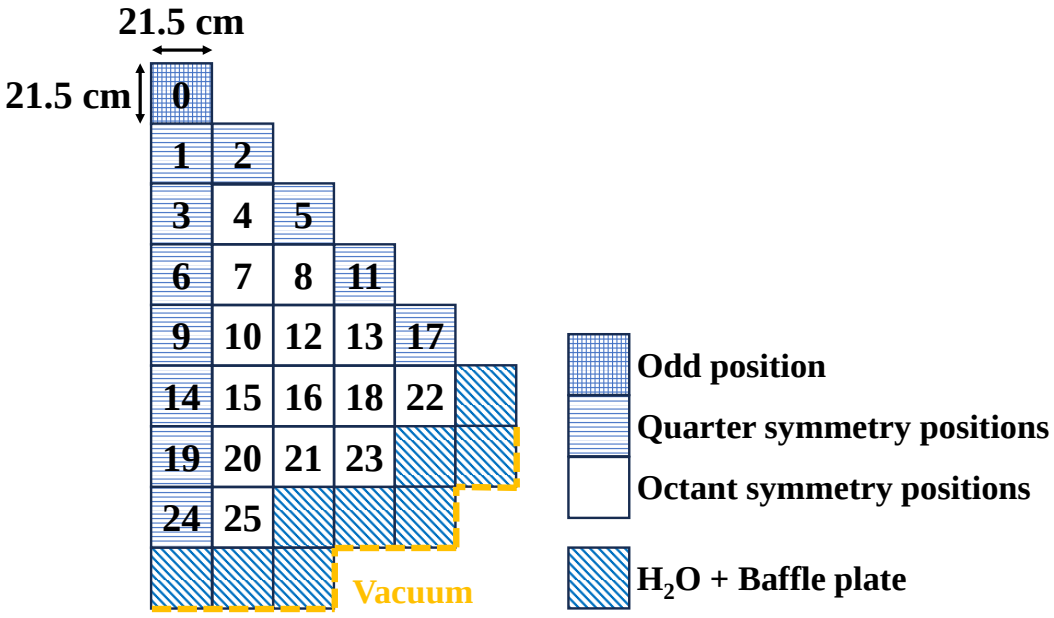



図 4-10　1/8 対称性を持つ PWR 炉心

次に、最適化に使用する燃料集合体の種類と体数（燃料インベントリ）を表 4-1 に示す。表 4-1 に示す燃料インベントリは、定期検査ごとに 1/3 の燃料集合体を新燃料に交換した場合を仮定して作成している。$^{235}$U 濃縮度が 3wt%もしくは 4wt%、可燃性毒物として Gd を含む燃料集合体を用いている。また、対称性を考慮した PWR で最適化を行うため、燃料集合体ごとの対称性を考えている。この燃料インベントリを用いた燃料装荷パターンの数は

$$\text{Odd の通り} \times \text{Quarter の通り} \times \text{Octant の通り} = \frac{1!}{1!} \times \frac{11!}{2!\,3!\,2!\,3!\,1!} \times \frac{14!}{3!\,2!\,3!\,3!\,3!} \approx 10^{13}$$

である。10 分で詳細な炉心計算ができると仮定した場合、$10^{13}$の燃料装荷パターンの計算を行うには 1 億年以上かかるため、全探索は現実的に不可能である。

表 4-1　燃料インベントリ

| U235 Enrichment [wt%] | Burnup [GWd/t] | Burnable Poison | Symmetry | Quantity |
|---|---|---|---|---|
| 3 | 30 | - | Odd | 1 |
| 3 | 30 | - | Quarter | 2 |
| 4 | 0 | - | Octant | 3 |
| 4 | 15 | - | Quarter | 3 |
| 4 | 15 | - | Octant | 2 |
| 4 | 20 | - | Quarter | 2 |
| 4 | 20 | - | Octant | 3 |
| 4 | 30 | - | Quarter | 3 |
| 4 | 30 | - | Octant | 3 |
| 4 | 0 | Gd | Quarter | 1 |
| 4 | 0 | Gd | Octant | 3 |
| **Total** | | | | 26 |

炉心計算は簡易計算版 ICEBURN で行う。簡易計算版 ICEBURN は、炉心特性を計算するための 2 群拡散理論に基づく燃焼計算コードである。ここで、2 群拡散理論とは、炉心内の中性子を高いエネルギー（高速群）と低いエネルギー（熱群）を持つ 2 つの中性子に分類し、拡散方程式に基づき中性子の分布を計算する手法である。計算領域の分割では、燃料集合体ごとに$1 \times 1$メッシュを使用している。燃焼計算は、{0, 2.5, 5.0, 7.5, 10.0, 12.0, 12.5, 13.0, 13.5, 14.0, 16.5} GWd/t で行われる。

本研究における燃料装荷パターン最適化の目的は、サイクルを通じた径方向ピーキング係数の最小化である。これは、一例として新燃料装荷体数を固定した場合、径方向ピーキング係数の制限値を満たす燃料装荷パターンは全燃料装荷パターンの 0.04%のみである[2]と

の報告があり、制限値を満足する燃料装荷パターンを作成することが困難なことによる。

したがって、報酬（または、目的関数）はサイクルを通じた径方向ピーキング係数にのみ依存する。ボードゲームの報酬が 0（負け）、0.5（引き分け）、1（勝ち）の値をとるのに対して、径方向ピーキング係数は 1 以上の任意の実数をとりうる。報酬をボードゲームと同じ範囲にするため、報酬を径方向ピーキング係数の逆数とすることで、0 より大きく 1 以下の値をとるように設定する。報酬が大きくなるほど、径方向ピーキング係数は小さくなっており、より良好な炉心特性が得られることに対応する。

本検討では、MCTS の最適化性能をランダムサーチと焼きなまし法（Simulated Annealing, SA）の最適化性能と比較する。最適化で行われる炉心特性計算の回数の条件は、1000 回と 10000 回の 2 通りである。1000 回は SA のなどの従来手法を使った最適化を行うには少ない回数であるが、10000 回は今回の計算条件であれば実用上十分な回数である。

MCTS では、燃料装荷パターン最適化を装荷手順の最適化とみなしているため、どこから装荷するかの手順によって最適化性能が変化することが考えられる。したがって、炉心外縁から炉心中心への装荷（MCTS-Periphery）と炉心中心から炉心外縁への装荷（MCTS-Center）の 2 種類の装荷方法で最適化を行う。具体的には、MCTS-P では図 4-11 に示す番号の 25 から 0 まで降順に燃料装荷を行い、MCTS-C では図 4-11 に示す番号の 0 から 25 まで昇順に燃料装荷を行う。

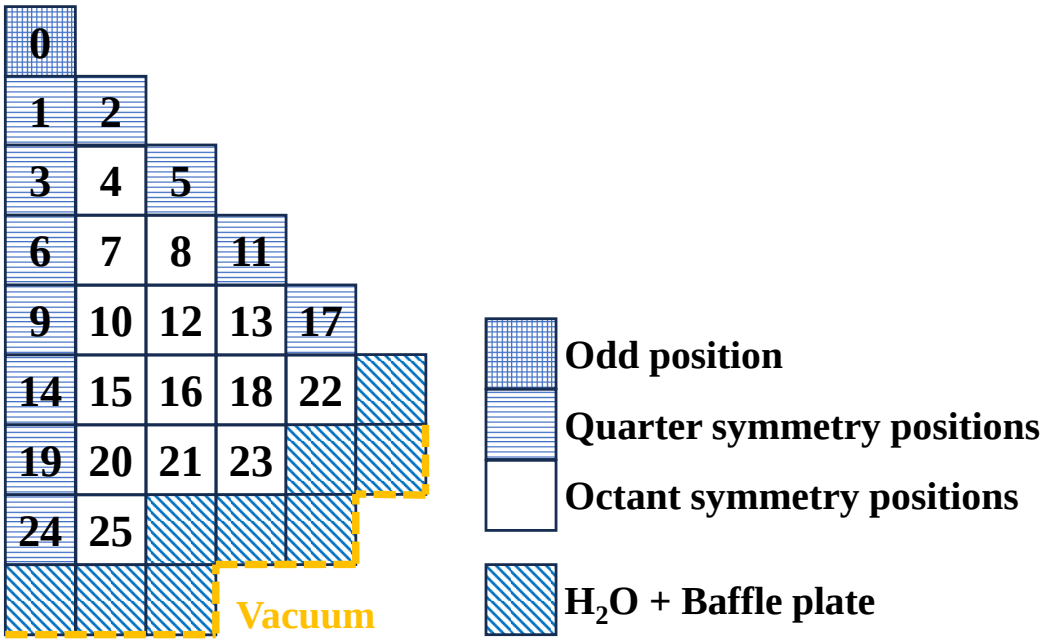



図 4-11　燃料装荷を行う順番

ランダムサーチでは、図 4-11 に示す番号の 25 から 0 まで降順にランダムに燃料装荷を行う。

SA では、初期の燃料装荷パターンがランダムに生成される。同じ対称性(Quarter または Octant)を持つ 2 箇所の燃料集合体がランダムに交換されることで近傍の燃料装荷パターンの生成を行う。

表 4-2 に MCTS および SA の最適化パラメータを示す。これらのパラメータは感度解析により得られたものであり、感度解析の結果を Appendix A に記載する。

表 4-2　MCTS と SA の最適化パラメータ

| 炉心計算回数 | 手法 | パラメータ | 値 |
|---|---|---|---|
| 1000 回 | MCTS | $C_p$ in Equation (4.1) | 0.10 |
| | SA | Initial Temperature | 0.10 |
| | | Final Temperature | 0.10 |
| | | Cooling Rate | 1 |
| 10000 回 | MCTS | $C_p$ in Equation (4.1) | 0.25 |
| | SA | Initial Temperature | 0.10 |
| | | Final Temperature | 0.0025 |
| | | Cooling Rate | $0.025^{1/10000}$ |

最適化された燃料装荷パターンは初期乱数に依存するため、MCTS、ランダムサーチ、SA それぞれについて異なる初期乱数で 100 回の最適化を行い、結果を統計的に処理する。

すべてのコードは C++で記述されている。計算環境として、CPU：Intel i9-10980XE @ 3.00GHz、RAM：128 GB のデスクトップコンピュータを用いる。

## 4.4　計算結果

### 4.4.1　炉心特性計算回数が 1000 回の場合

最適化における炉心特性計算の回数が 1000 回の場合の最適化結果を表 4-3 に示す。表 4-3 は 100 回の独立した最適化を行った結果を示している。100 回の最適化で得られる最小のピーキング係数の平均値、標準偏差、最大値、最小値は MCTS-P で最も小さな値となった。

また、MCTS における装荷手順の違いを見ると、MCTS-P（外側から装荷）は従来手法である SA に比べて良い最適化性能を示したが、MCTS-C（内側から装荷）は従来手法である SA に比べて悪い結果となった。

MCTS と SA の最適化では、どちらもランダムサーチに比べて良い最適化性能を持っている。

表 4-3　100 回の最適化における最小の径方向ピーキング係数
ただし、1000 回の炉心特性計算

| Method | Average | Std. Dev. | Minimum | Maximum |
|---|---|---|---|---|
| **MCTS-P** | **1.370** | **0.028** | **1.321** | **1.449** |
| **MCTS-C** | 1.449 | 0.036 | 1.354 | 1.541 |
| **Random** | 1.485 | 0.041 | 1.376 | 1.574 |
| **SA** | 1.432 | 0.070 | 1.328 | 1.671 |

また、MCTS-P と MCTS-C、MCTS-P とランダムサーチ、MCTS-P と SA の炉心計算で得られる最小の径方向ピーキング係数の推移をそれぞれ図 4-12、図 4-13、図 4-14 に示す。図 4-12 が示す通り、MCTS-P と MCTS-C では 100 回の炉心特性計算以降から径方向ピーキング係数の差が出ている。図 4-13 が示す通り、MCTS-P とランダムサーチでは 25 回の炉心特性計算以降から径方向ピーキング係数の差が出ている。図 4-14 が示す通り、初期の最適化の段階から径方向ピーキング係数の差が出ているが、炉心特性計算回数が増えるにつれて差が小さくなることが分かる。

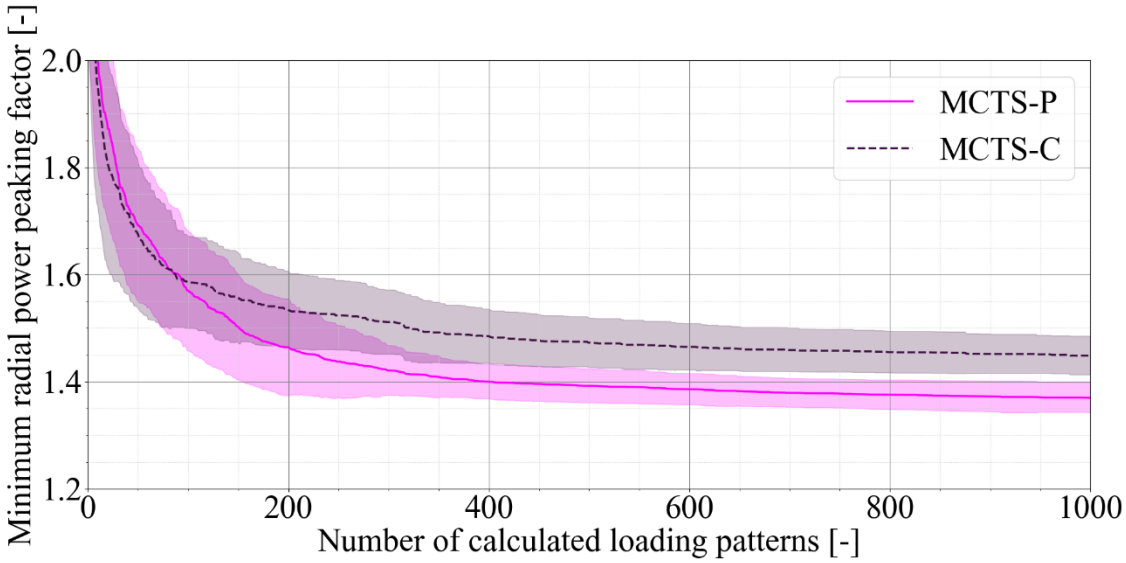



図 4-12　MCTS-P と MCTS-C における最小の径方向ピーキング係数の比較
ただし、1000 回の炉心特性計算
100 回の最適化における平均であり、ハッチング部分は標準偏差(1σ)を示す

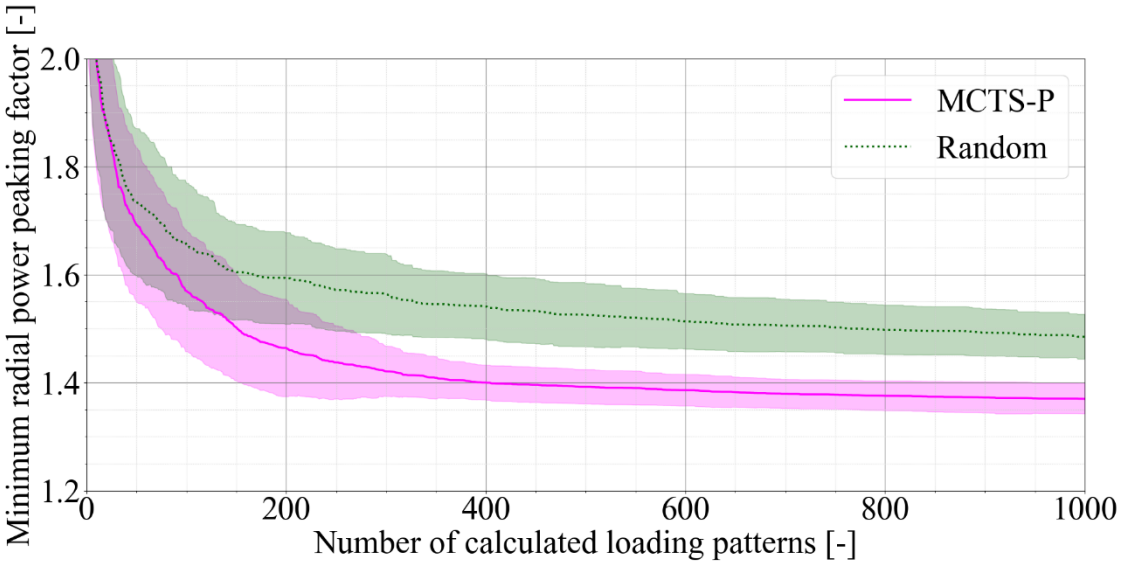



図 4-13　MCTS-P とランダムサーチにおける最小の径方向ピーキング係数の比較
ただし、1000 回の炉心特性計算
100 回の最適化における平均であり、ハッチング部分は標準偏差(1σ)を示す

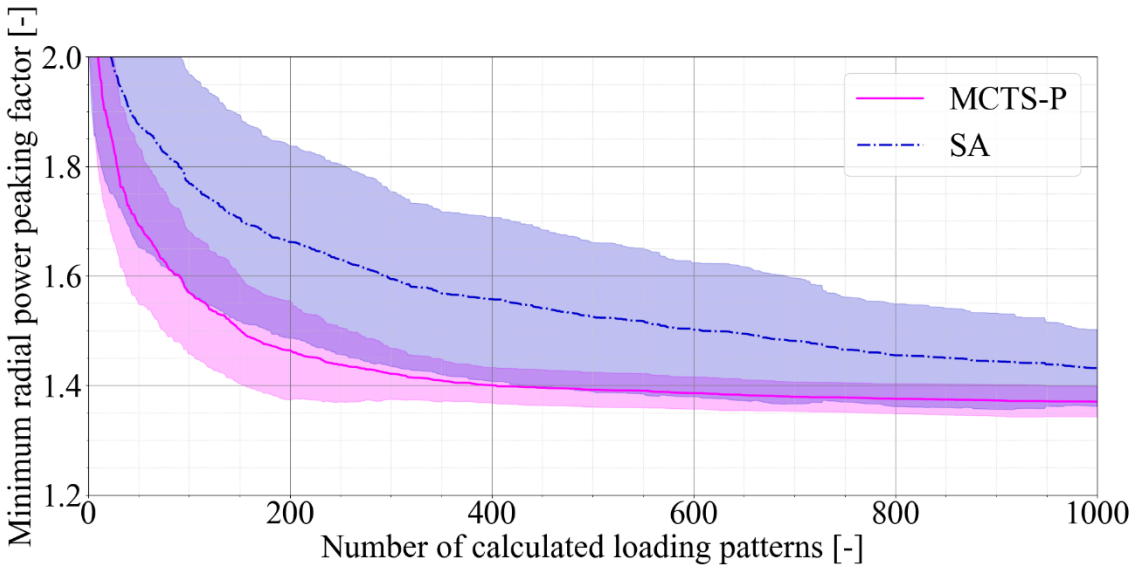



図 4-14　MCTS-P と SA における最小の径方向ピーキング係数の比較
ただし、1000 回の炉心特性計算
100 回の最適化における平均であり、ハッチング部分は標準偏差(1σ)を示す

### 4.4.2　炉心特性計算回数が 10000 回の場合

最適化における炉心特性計算の回数が 10000 回の場合の最適化結果を表 4-4 に示す。表 4-4 は 100 回の独立した最適化を行った結果を示している。100 回の最適化で得られる最小のピーキング係数の平均値、最小値は SA で最も小さくなり、標準偏差と最大値は MCTS-P

で最も小さくなった。

表 4-4　100 回の最適化における最小の径方向ピーキング係数
ただし、10000 回の炉心特性計算

| Method | Average | Std. Dev. | Minimum | Maximum |
|---|---|---|---|---|
| **MCTS-P** | 1.338 | **0.014** | 1.319 | **1.376** |
| **MCTS-C** | 1.400 | 0.022 | 1.339 | 1.446 |
| **Random** | 1.425 | 0.024 | 1.357 | 1.483 |
| **SA** | **1.324** | 0.024 | **1.293** | 1.544 |

また、MCTS-P と MCTS-C、MCTS-P とランダムサーチ、MCTS-P と SA の炉心計算で得られる最小の径方向ピーキング係数の推移をそれぞれ図 4-15、図 4-16、図 4-17 に示す。図 4-15 が示す通り、MCTS-P と MCTS-C では最適化初期から径方向ピーキング係数の差が出ている。また、図 4-16 が示す通り、MCTS-P とランダムサーチでも最適化初期から径方向ピーキング係数の差が出ている。図 4-17 が示す通り、MCTS-P と SA では最適化の初期段階では MCTS-P の最適化が良いが、炉心特性計算回数が増えることで SA の最適化が MCTS-P を上回ることが分かる。

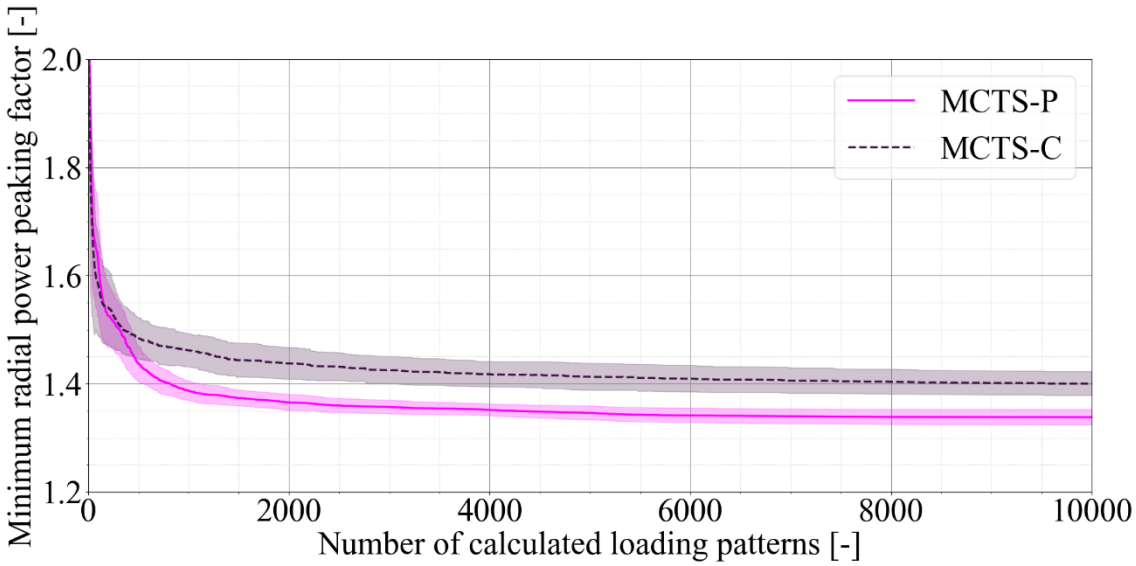



図 4-15　MCTS-P と MCTS-C における最小の径方向ピーキング係数の比較
ただし、10000 回の炉心特性計算
100 回の最適化における平均であり、ハッチング部分は標準偏差(1σ)を示す

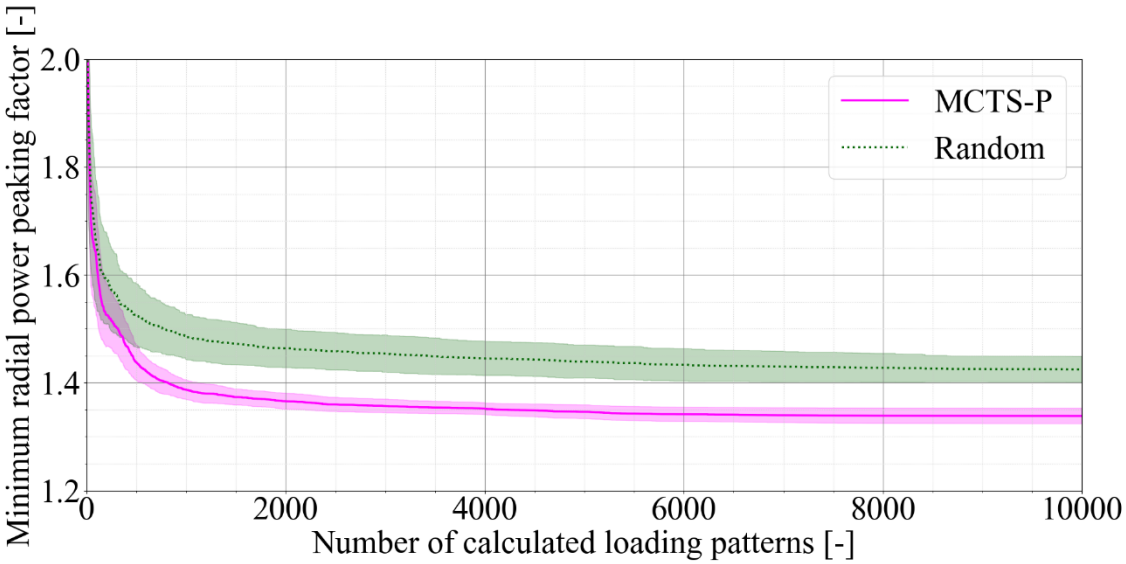



図 4-16　MCTS-P とランダムサーチにおける最小の径方向ピーキング係数の比較
ただし、10000 回の炉心特性計算
100 回の最適化における平均であり、ハッチング部分は標準偏差(1σ)を示す

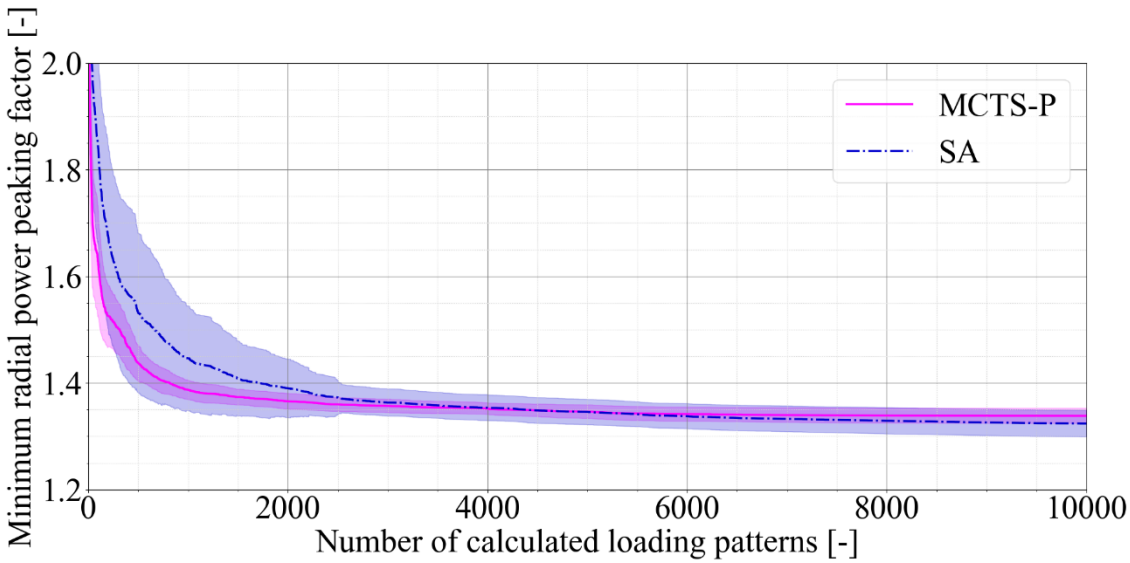



図 4-17　MCTS-P と SA における最小の径方向ピーキング係数の比較
ただし、10000 回の炉心特性計算
100 回の最適化における平均であり、ハッチング部分は標準偏差(1σ)を示す

## 4.5　考察

### 4.5.1　MCTS における装荷手順の影響

MCTS-P（外側から燃料を装荷する手順）と MCTS-C（内側から燃料を装荷する手順）の最適化性能を比較して、炉心特性計算回数にかかわらず MCTS-P の性能が良くなる結果が

得られた。MCTS-P の性能が良くなる理由は、外側の燃料集合体が径方向ピーキング係数に影響しやすいためである。MCTS-P と MCTS-C の最適化によって構成された木の一部をそれぞれ図 4-18 と図 4-19 に示す。MCTS-P の木では、1 体目の装荷で燃焼度 0 GWd/t の Gd を含まない燃料集合体を装荷した燃料装荷パターンの期待値が 0.618 である。他の燃料装荷パターンの期待値が 0.4 前後に対して期待値が 0.2 ほど大きいため、径方向ピーキング係数が小さくなるような燃料装荷を特定することができている。しかし、MCTS-C の木では、すべての燃料装荷パターンの期待値が 0.4 前後であるため、方向ピーキング係数が小さくなるような燃料装荷を特定することができない。

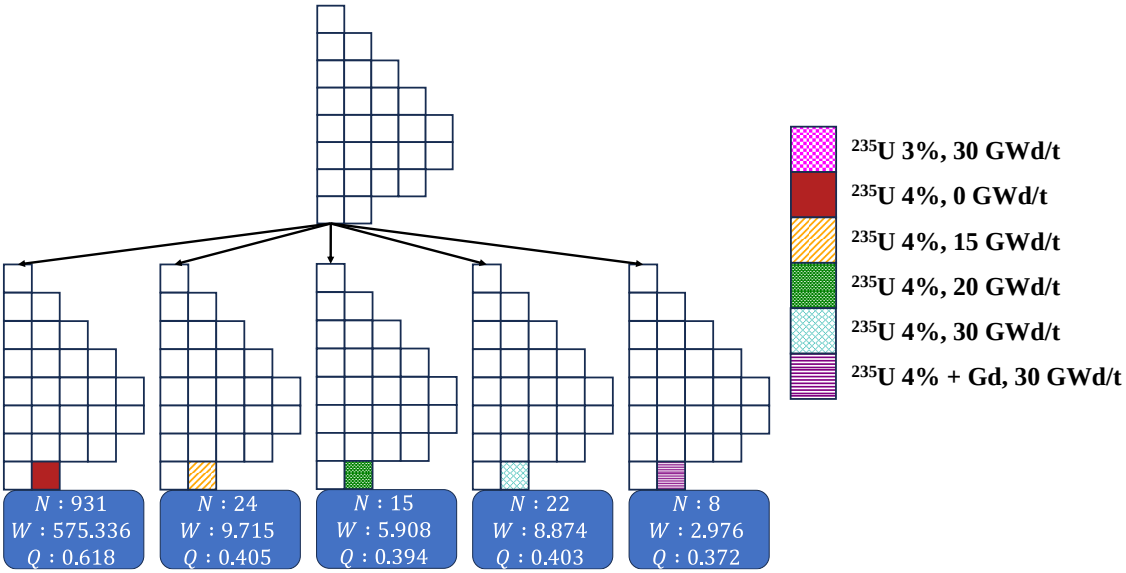



図 4-18　最適化終了後の MCTS-P の木の一部

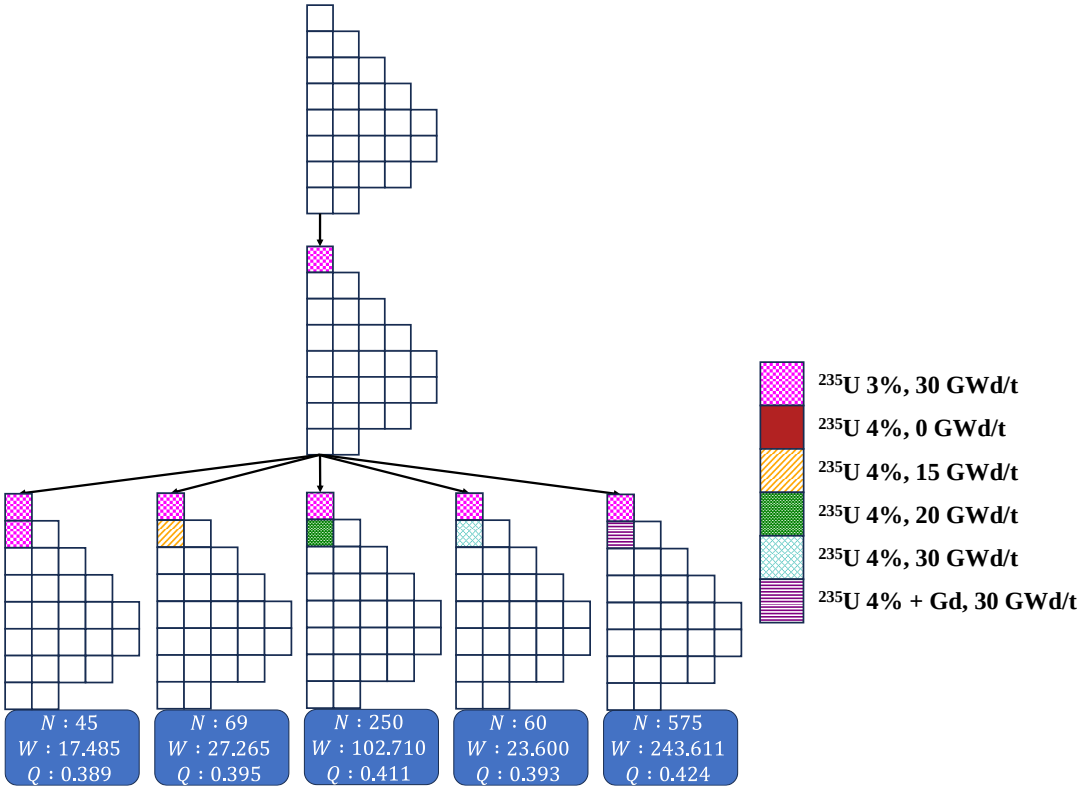



図 4-19　最適化終了後の MCTS-C の木の一部

### 4.5.2　木を構築することの利点

図 4-13 が示すように、MCTS-P とランダムサーチを比較すると、最適化の初期では性能が同じであるが、最適化が進むにつれて MCTS-P の性能が上回る。これは、MCTS による木の構築によって、有望な装荷する燃料集合体を絞った最適化が可能であることを示している。

炉心特性計算回数 25 回終了後における MCTS-P の木の一部を図 4-20 に示す。炉心特性計算回数 25 回終了後では、深さ 1 の燃料装荷パターンの期待値がすべて 0.4 前後であり、有望な装荷を知ることはできない。また、1 体目の装荷で燃焼度 0 GWd/t の Gd を含まない燃料集合体を装荷した燃料装荷パターンの子が図 4-20 に示されているが、炉心特性計算回数が少ないため、2 体目の有望な装荷を知ることもできない。

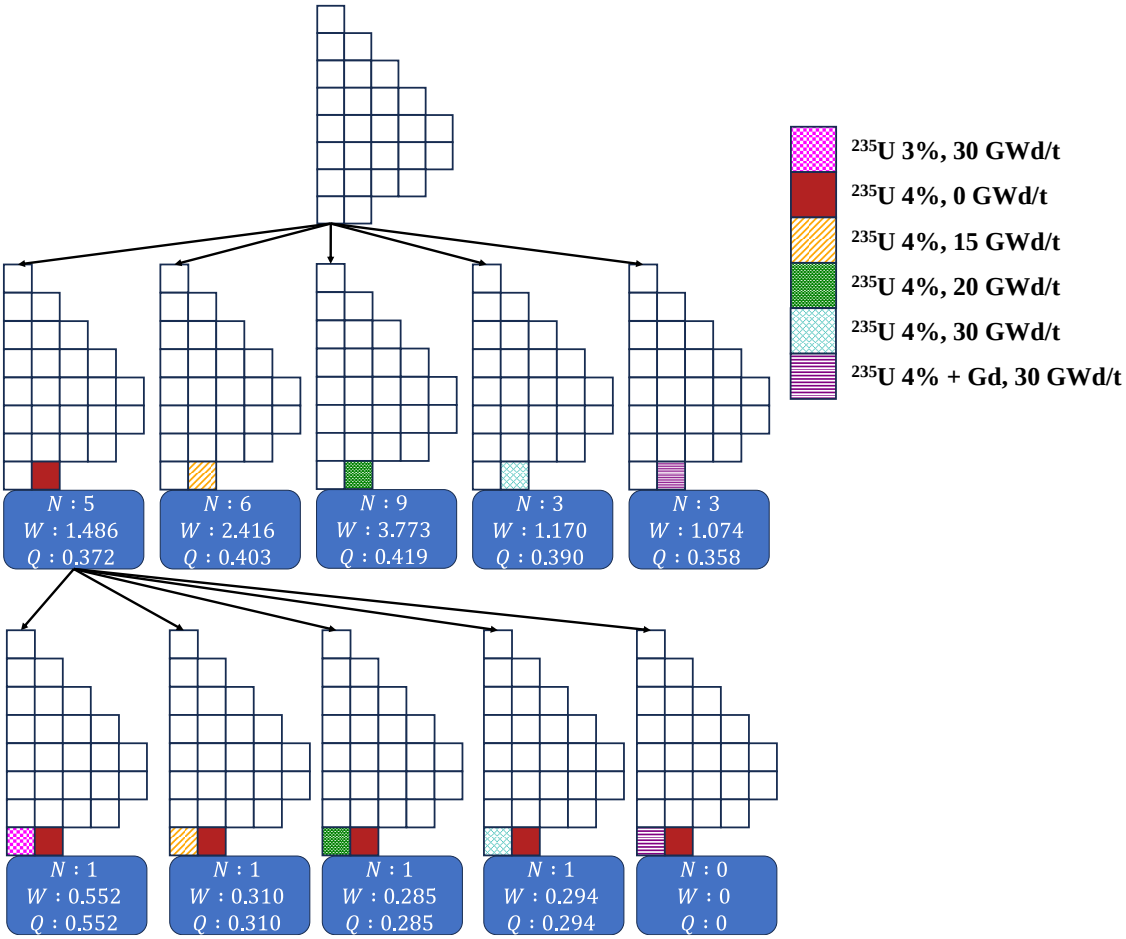



図 4-20　炉心特性計算回数 25 回終了後における MCTS-P の木の一部

次に、炉心特性計算回数 125 回終了後における MCTS-P の木の一部を図 4-21 に示す。1 回目の装荷は、燃焼度 0 GWd/t の Gd を含まない燃料集合体を装荷した燃料装荷パターンの期待値が 0.07 ほど高く、有望な装荷であることが分かる。さらに、2 体目の装荷を見ると、燃焼度 0 GWd/t の Gd を含む燃料集合体を装荷した燃料装荷パターンの期待値が高く、2 体目の有望な装荷を知ることができる。このように、炉心計算回数が増えると、木の深さが深くなるとともに、有望な装荷手順を得ることができる。

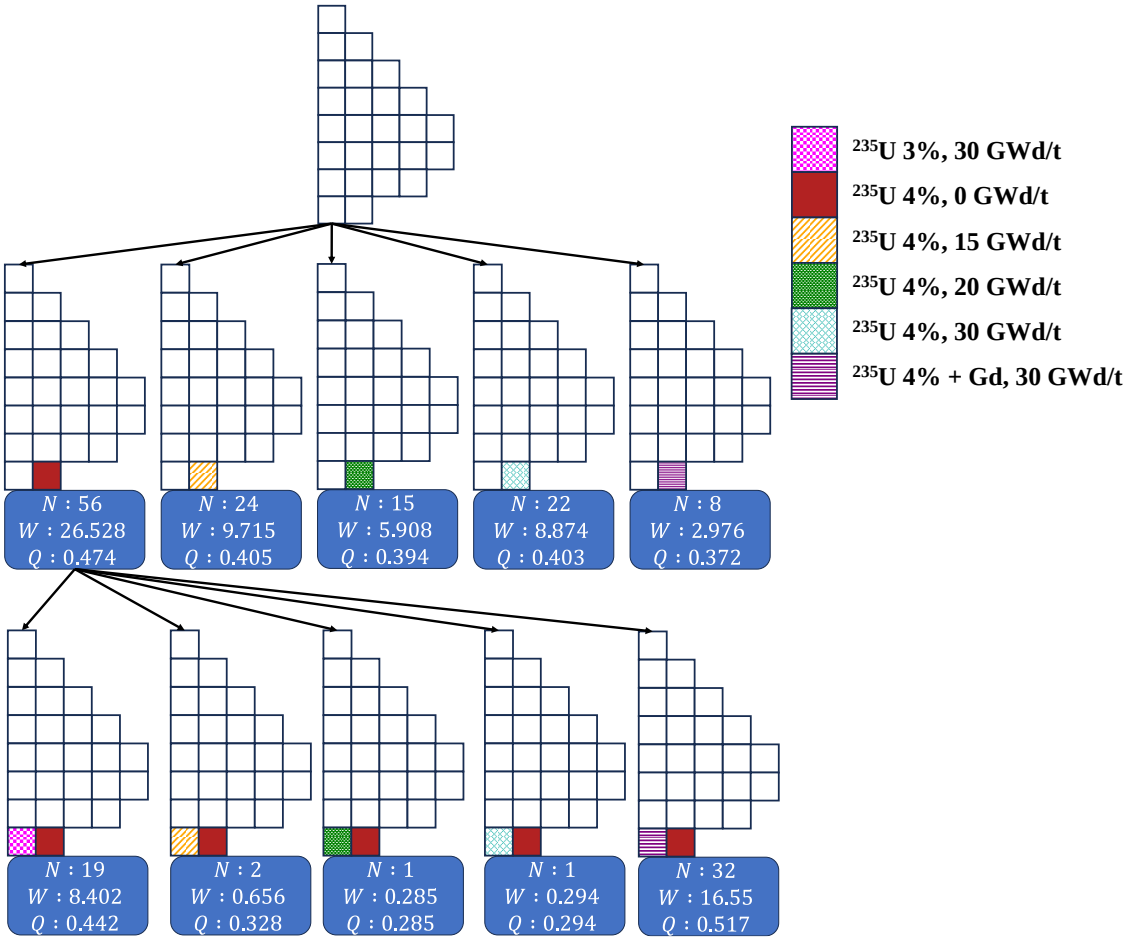



図 4-21　炉心特性計算回数 125 回終了後における MCTS-P の木の一部

### 4.5.3　炉心特性計算回数の違いによる最適化性能の変化

1000 回の炉心特性計算回数による最適化結果では、SA に比べて MCTS-P で得られる径方向ピーキング係数の平均値、標準偏差、最小値、最大値が良い結果となった。

MCTS-P では、近傍の解の生成を行わない幅広い探索と木の構築による効率的な探索が可能であるためである。SA や他の最適化手法では、主として燃料集合体位置の交換による近傍の探索による最適化が行われているため、初期解の影響が大きく、最適化結果のばらつきが大きくなる。

10000 回の炉心特性計算回数による最適化結果では、SA に比べて MCTS-P で得られる径方向ピーキング係数の標準偏差と最大値で良い結果、平均値と最小値で悪い結果となった。このような結果が得られるのは、局所最適解の影響によるものである。

目的関数の値と燃料装荷パターンの関係のイメージを図 4-22 に示す。MCTS-P では、式(4.6)の UCT を用いて期待値に基づく探索を行うため、期待値$Q$が良い燃料装荷パターンが選ばれやすくなる。

$$c^* = \underset{c \in \text{children of } s}{\arg\max} \left( Q(c) + C_p \sqrt{\frac{\ln N(s)}{N(c)}} \right) \tag{4.6}$$

したがって、図 4-22 に示すように、近傍の目的関数の値が良い燃料装荷パターンの探索が行われる。MCTS では、悪い燃料装荷パターンを最適解とすることはないが、以上により燃料装荷パターンも得ることができない傾向があると推察できる。

SA では、燃料集合体位置の交換による近傍の燃料装荷パターンの探索が行われる。現在の燃料装荷パターンより近傍の燃料装荷パターンが悪い場合、現在の燃料装荷パターンから抜け出すことができなくなることで探索が止まる。SA は近傍に良い燃料装荷パターンが存在すれば、良い探索が可能であるが、近傍に良い燃料装荷パターンがなければ良い探索ができない。ゆえに、100 回の最適化で得られる径方向ピーキングの最小値は小さくなるが、最大値と標準偏差が大きくなる。

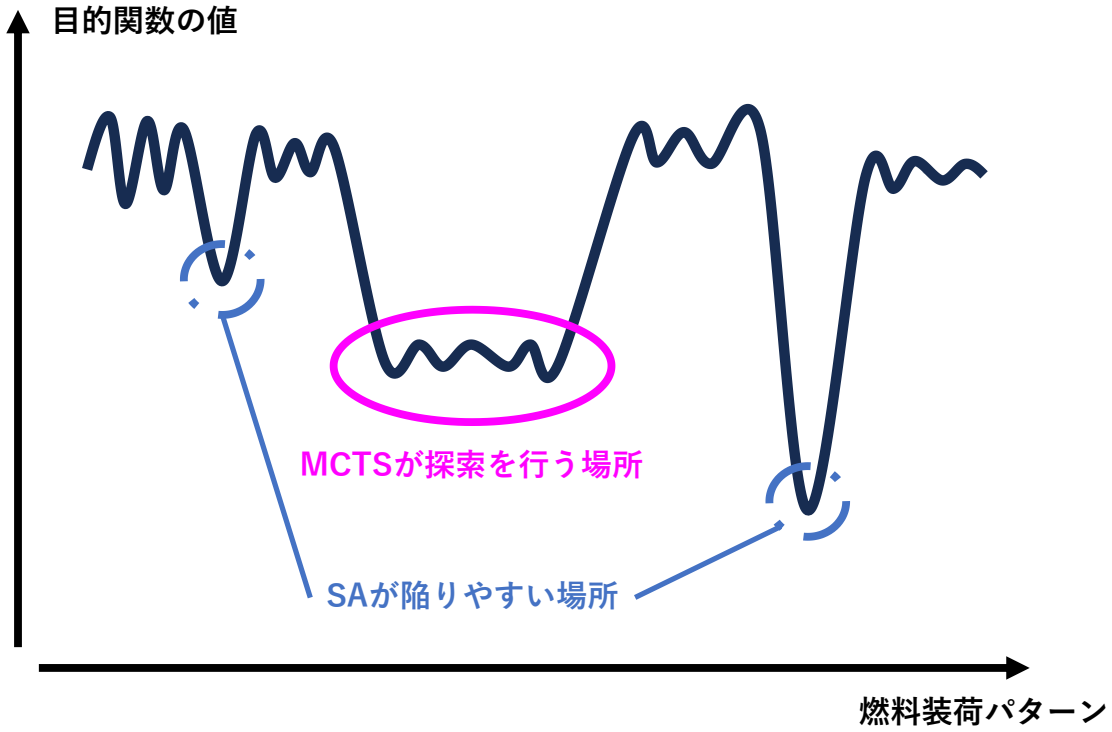



図 4-22　目的関数と燃料装荷パターンの関係のイメージ

## 4.6　本章のまとめ

本章では、MCTS を燃料装荷パターン最適化に適用する方法と適用した結果を述べた。PWR 炉心を対象として、MCTS、ランダムサーチ、SA の 3 つの手法で最適化を行い、MCTS による最適化の特徴について考察した。

4.2 節では、燃料装荷パターン最適化を燃料集合体が装荷されていない空の炉心に 1 体ず

つ燃料を装荷する手順の最適化とみなすことで、燃料装荷パターンのゲーム木が構成できることを述べた。さらに、このゲーム木に対して MCTS を用いた探索アルゴリズムを説明した。

4.3 節では、最適化を行う計算体系、燃料インベントリ、計算条件を述べた。PWR 炉心の燃料装荷パターン最適化を行い、サイクルを通じた径方向ピーキング係数の最小化を目的とした。

4.4 節では、MCTS、ランダムサーチ、SA の 3 つの手法で最適化を行った結果を記載し、少ない炉心特性計算であれば MCTS、多い炉心特性計算回数であれば SA が良い性能を持つことが分かった。

4.5 節では、MCTS による燃料装荷パターン最適化の特徴として、次の 3 点があることを述べた。

・装荷する順番が最適化に影響を与える。径方向ピーキング係数は外側の燃料集合体の影響を受けやすいため、外側から燃料装荷を行うことが良い。

・ランダムサーチと同様に乱数を用いた手法であるが、木探索を用いることで効率的な最適化が可能となる。

・従来手法と比較して少ない計算回数でそれなりに良好な燃料装荷パターンを見つけることが可能である。

第 5 章では、少ない計算回数でそれなりに良好な燃料装荷パターンを見つけることが可能であることを活用し、深層学習を組み合わせることで少ない計算回数でさらに良好な燃料装荷パターンを見つける手法を検討する。

### 4.7 参考文献


[1] 関西電力株式会社, “高浜発電所 3・4 号炉 発電用原子炉設置許可申請書 完本版（令和 4 年 6 月）,” https://www.kepco.co.jp/energy_supply/energy/nuclear_power/info/knic/library/kyonin/gensiro.html (accessed: Jan. 29, 2024).

[2] A. Yamamoto, “Effect of radial peaking factor limitation on discharge burnup,” *J. Nucl. Sci. Technol*. Vol. 39, no. 12, pp. 1260–1268, Dec. 2002, doi: 10.1080/18811248.2002.9715319.

# 第5章　MCTS と深層学習の燃料装荷パターン最適化への適用

## 5.1　本章の概要

本章では、少ない回数で良好な燃料装荷パターンを見つけることができるという MCTS の特徴に注目し、深層学習を組み合わせることで、さらなる性能の向上を目的とした手法の検討を行う。MCTS と深層学習を組み合わせた手法の説明と燃料装荷パターン最適化に適用した結果を記載する。

5.2 節では、MCTS と深層学習を組み合わせた手法を説明する。5.2.1 項で本手法の概念を述べる。5.2.2 項では、ゲーム AI で用いられている深層学習を解説する。5.2.3 項では、深層学習を用いた燃料装荷パターンの学習方法、5.2.4 項では MCTS と深層学習を用いた探索アルゴリズムを説明する。

5.3 節では、本手法の最適化性能及び汎化性能を評価するための計算条件を示す。

5.4 節では、本手法を用いた燃料装荷パターン最適化の結果を MCTS と SA の最適化結果と比較する。

5.5 節では、得られた結果から本手法による燃料装荷パターン最適化の特徴について議論する。

5.6 節では、本章のまとめを述べる。

## 5.2　MCTS と深層学習を用いた燃料装荷パターン最適化手法

本節では、MCTS と深層学習を組み合わせた木探索（Neural Network MCTS, N-MCTS）を説明する。

### 5.2.1　MCTS と深層学習を用いた木探索（N-MCTS）の概念

第 4 章のモンテカルロ木探索のシミュレーションでは、葉の燃料装荷パターン（子を持たない部分的な燃料装荷パターン）から完全な燃料装荷パターンまでランダムに燃料を選択して装荷を続ける（ランダムプレイアウト）。しかし、ランダムではなく、燃料装荷に関する何らかの知識に基づき、良好な装荷パターンになると予想される装荷を行うことで、最適化性能を向上することができると考えられる。この考え方に基づき、深層学習を用いた知識に基づく燃料装荷によって、有望な燃料装荷を実現する。

### 5.2.2　ゲーム AI における深層学習

2.6 節では、機械学習と深層学習について説明を行い、機械学習を用いた燃料装荷パターン最適化について言及した。本項では、2.6 節の内容を前提として、1.1.3 項で紹介したゲーム AI である AlphaZero[1]の深層学習で使用されている ResNet[2]について解説する。

深層学習とは、層が深いニューラルネットワーク、すなわち、深層ニューラルネットワーク（Deep Neural Network, DNN）を用いたパターン認識であることを 2.6.1 項で説明した。DNN の構造を図 5-1 に示す。入力層と出力層の間に多くの隠れ層が存在することで、入力

を$\boldsymbol{X}$、出力を$\boldsymbol{t}$とする複雑な関数を表現することが可能となる。

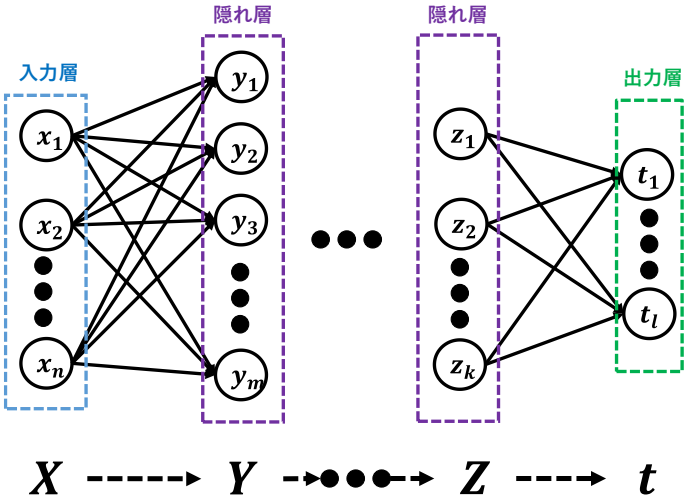



図 5-1　深層ニューラルネットワークの構造

入力層の変数と 1 層目の隠れ層の変数の関係を式(5.1)と式(5.2)、及び図 5-2 に示す。

$$A_m = b_m + \sum_n w_{m,n} x_n \tag{5.1}$$

$$y_m = g(A_m) \tag{5.2}$$

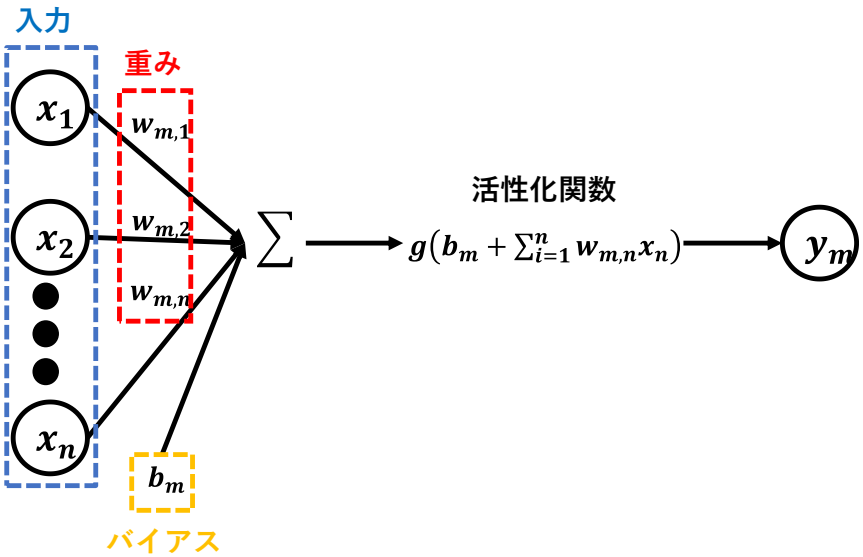



図 5-2　ニューラルネットワークにおける変数の関係

重みを変数とする$n$変数一次関数によって、入力層の変数$\boldsymbol{X}=\{x_1,x_2,\cdots,x_n\}$は、$\boldsymbol{A}=\{A_1,\cdots,A_m\}$に変換される。次に、$\boldsymbol{A}$に関数$g$を作用することで隠れ層の変数$\boldsymbol{Y}=\{y_1,\cdots,y_m\}$が出力される。式(5.1)と式(5.2)のように、隠れ層の変数が入力層のすべての変数に依存する層を全結合層（Fully-Connected Layer）という。また、関数$g$のことを活性化関数といい、複雑な関数を表現するために、非線形関数が採用されている[3]。

活性化関数として、正規化線形関数（Rectified Linear Unit, ReLU）の使用が推奨されている[4]。ReLU を式(5.3)と図 5-3 に示す。ReLU は、入力が 0 以下であれば出力が 0 となり、入力が 0 以上であれば入力値がそのまま出力値となる関数である。

$$y_i=g_i(\boldsymbol{A})=\max(A_i,0) \tag{5.3}$$

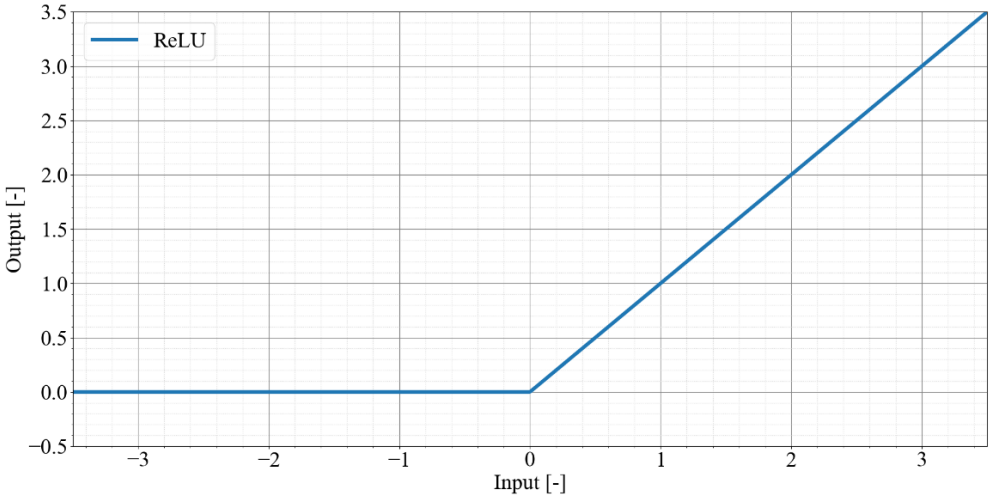



図 5-3　ReLU のグラフ

また、ゲーム AI は、すべての行動（例えば、実行可能な駒の動かし方）に対してその行動が最善である確率を出力している。したがって、活性化関数として、総和が 1 となる正の推定値を返す関数であるソフトマックス関数（Softmax Function）を用いている[4], [5]。式(5.4)にソフトマックス関数を示す。

$$y_i=g_i(\boldsymbol{A})=\frac{\exp(A_i)}{\sum_m \exp(A_m)} \tag{5.4}$$

ボードゲームの局面や燃料装荷パターンの認識のように 2 次元の特徴をもつパターン認識では、2 次元の特徴を抽出することが重要となる。そこで、2 次元のままデータの処理を行いながら特徴を抽出する層（畳み込み層, Convolutional Layer[6], [7]）を用いることで、優れた特徴抽出を可能にしている。畳み込み層で行われている畳み込みの処理を図 5-4 に示す。図 5-4 では、1 枚の画像（1 チャネル）から 2 種類のフィルターを用いて 2 枚の画像（2 チャネル）を生成しており、複数のフィルターを用いることでさまざまな空間的な距離に基

づく相互の依存関係を抽出することが可能である。

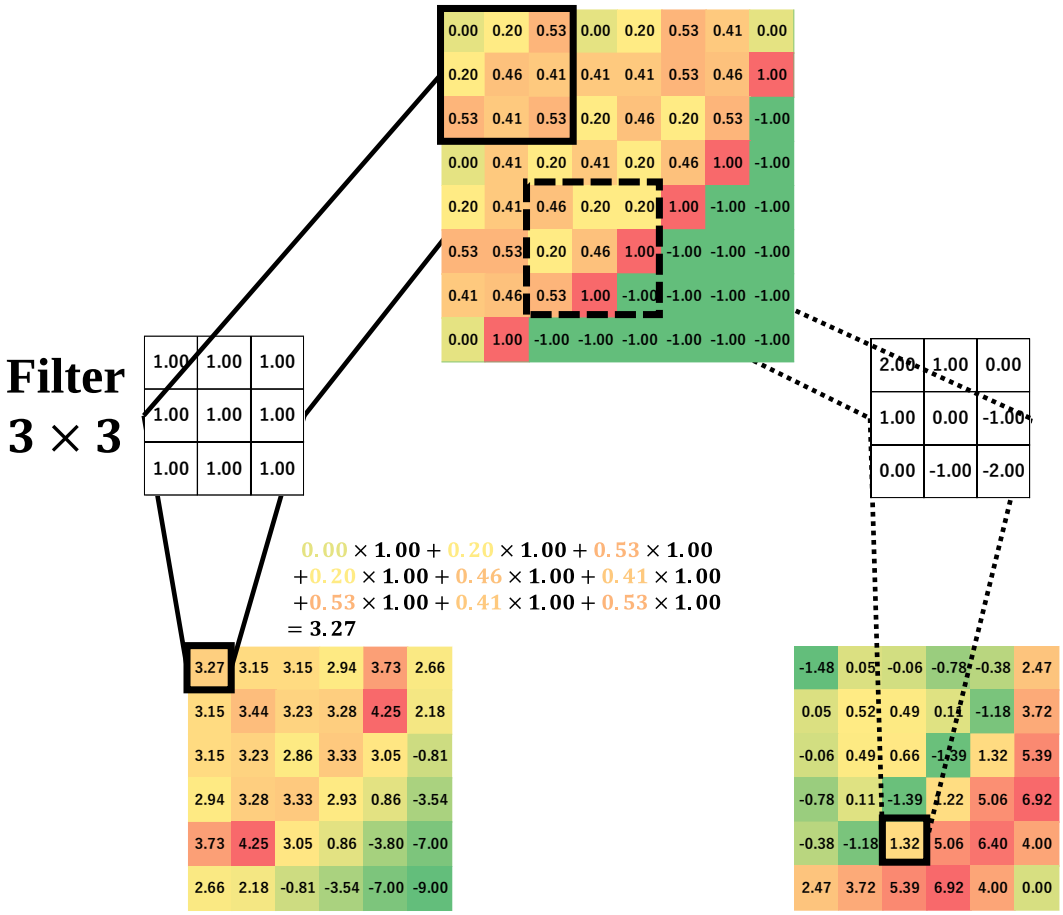



図 5-4　畳み込みの仕組み

また、畳み込みの処理後にサイズが小さくなることを避けるために、図 5-5 に示すパディング（Padding）を畳み込みの処理前に行う場合がある。パディング処理では、画像の周囲をゼロで埋めることで、畳み込みにより画像のサイズが小さくならないように工夫している。

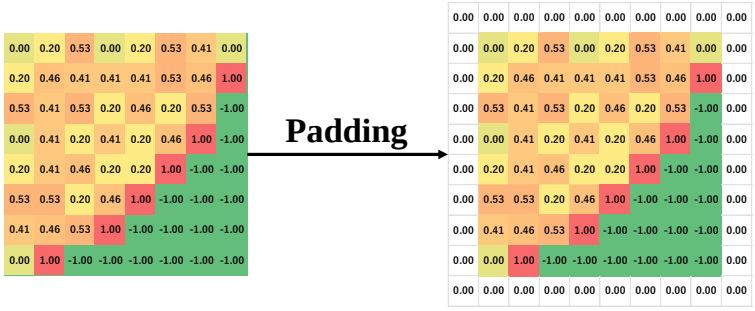



図 5-5　パディング処理

畳み込み層を用いたニューラルネットワークを畳み込みニューラルネットワーク（Convolutional Neural Network, CNN）といい、CNN の構造を図 5-6 に示す。フィルターに通しながらチャネル数の増減を行い、最後に画像の平坦化と出力を行っている。CNN で更新するパラメータは各フィルターの値である。

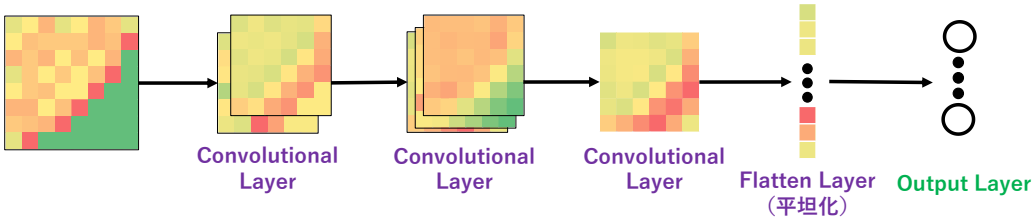



図 5-6　畳み込みニューラルネットワークの構造

将棋や囲碁などのボードゲームは駒の配置一つで大きく局面の評価が異なるため、高精度な CNN を使う必要がある。高精度な CNN を得るためには、層を深くすることで複雑な関数を表現できるようにすることが重要となる。しかし、層を深くすると NN のパラメータ更新が上手くいかず、精度が落ちるという問題が生じる。これを勾配消失問題といい、層数の多い NN では入力層に近い層ほど勾配が小さくなることに起因する問題である[6]。

18 層の NN と 34 層の NN を用いて、画像データベースである ImageNet[8]の画像のパターン認識を行った結果を図 5-7 に示す。横軸が NN のパラメータ更新回数、縦軸が正答率である。縦軸の値が低いほど良い精度で画像のパターン認識ができることを表しているが、層を深くすることで精度が悪くなることが分かる。勾配消失問題の解決法として、残差ネットワーク（Residual Network）[2], [6]を用いる方法がある。

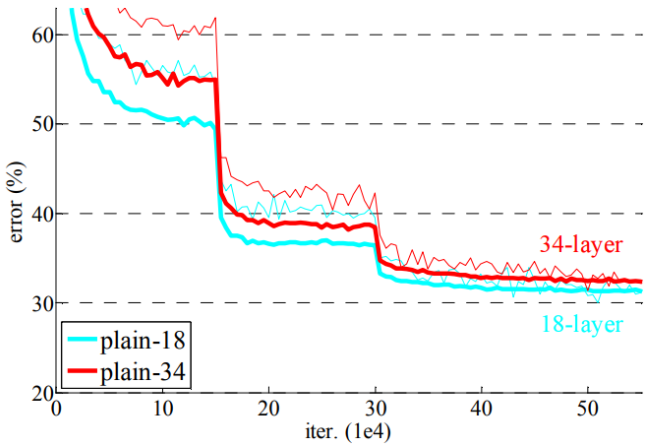



図 5-7　18 層の NN と 34 層の NN による ImageNet の学習結果[2]
太線が未学習データに対する正答率、細線が学習データに対する正答率である

残差ネットワークの例を図 5-8 に示す。Conv 3 × 3 Padding Filter 64は、Padding 処理を行ったベクトルに対して、64 個の3 × 3のフィルターを用いた畳み込みを行うことを示している。入力ベクトル$\boldsymbol{x}$と畳み込み層を通したベクトル$H(\boldsymbol{x})$を足し合わせたベクトル$\boldsymbol{y} = \boldsymbol{x} + H(\boldsymbol{x})$を出力とすることで、出力ベクトル$\boldsymbol{y}$に入力ベクトル$\boldsymbol{x}$の情報を多く持たせることができる。したがって、入力層に近いベクトルの情報を出力層まで伝えることができるため、勾配を効率的に入力層に近い層に伝播させることが可能となる。

また、畳み込み層が$H(\boldsymbol{x}) = \boldsymbol{y} - \boldsymbol{x}$という、入力$\boldsymbol{x}$と出力$\boldsymbol{y}$の差（残差）を表す関数となるため、残差ネットワークと呼ばれている。残差ネットワークを含む NN は ResNet と呼ばれ、層を深くすることで図 5-9 のように精度よく画像のパターン認識を行うことが可能となる。

ResNet を用いた深層学習はゲーム AI だけでなく、燃料装荷パターンの高速な炉心特性予測にも使う研究が行われている[9]。

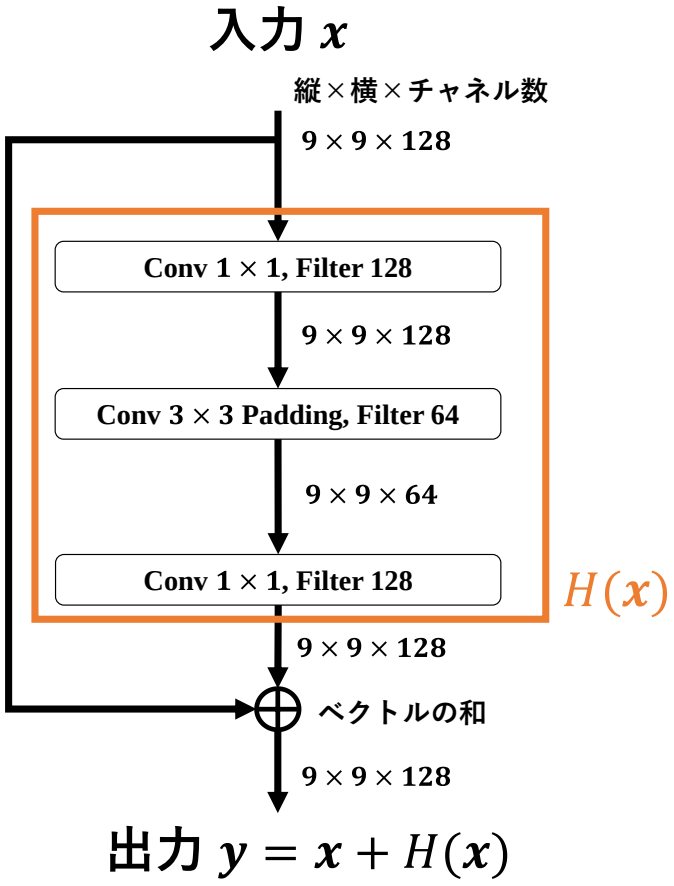



図 5-8　残差ネットワークの例

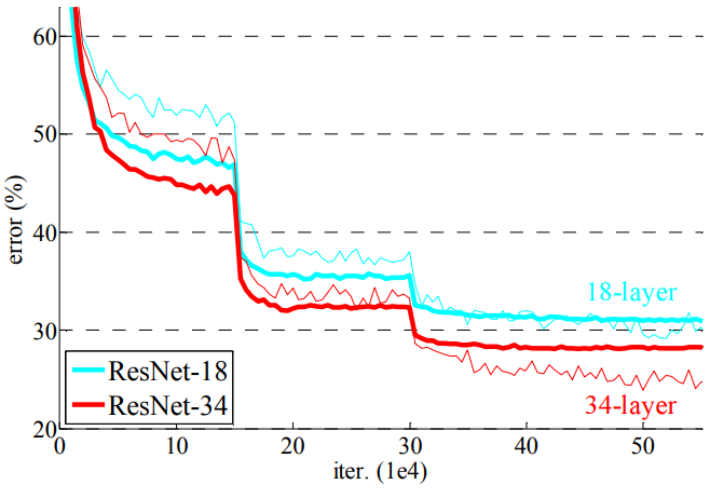



図 5-9　18 層 ResNet と 34 層 ResNet を用いた ImageNet の学習結果[2]
太線が未学習データに対する正答率、細線が学習データに対する正答率である

最後に、ResNet の例として本検討で用いる NN を図 5-10 に示す。図 5-10 に示す NN は、AlphaZero の学習速度を 50 倍以上向上させた KataGo[10]の Nested Bottleneck Residual Net を参考に作成している。NN の詳しい情報は 5.3 項の計算条件で記述する。

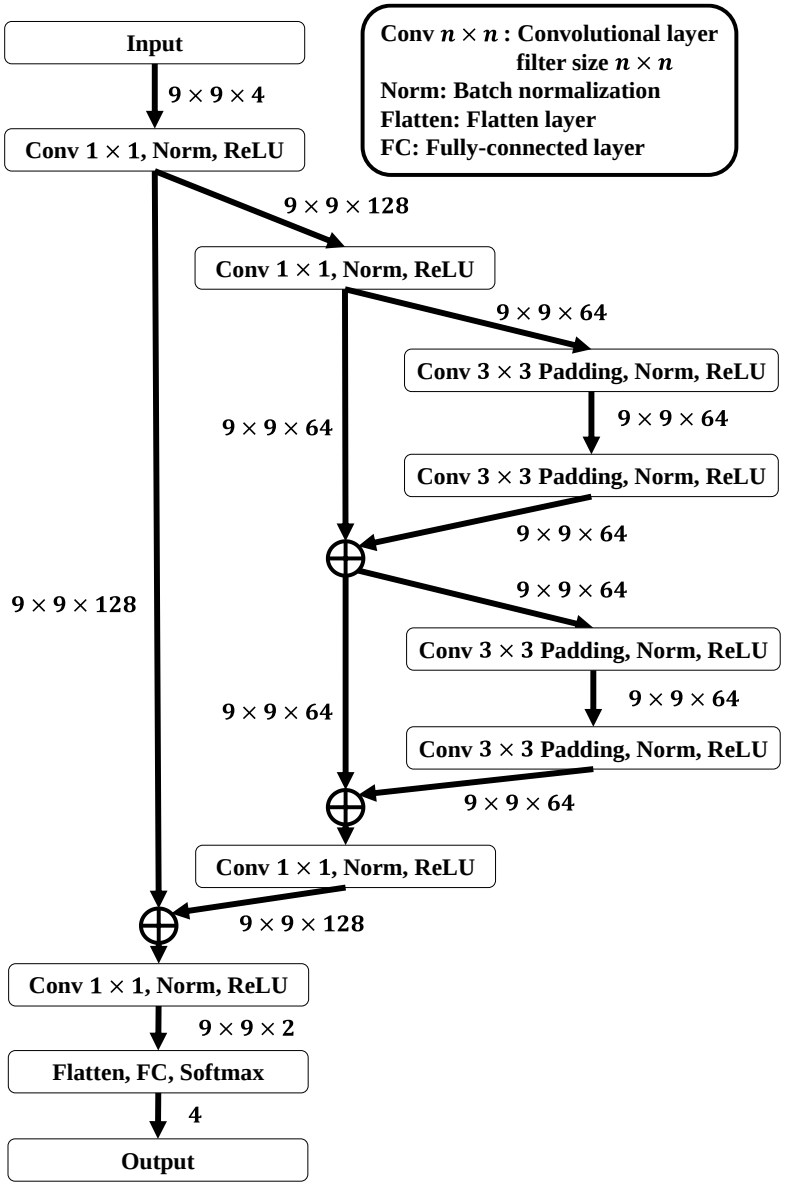



図 5-10　本検討で用いるニューラルネットワーク

### 5.2.3　N-MCTS の学習

N-MCTS では、NN に従う燃料装荷を行うが、この燃料装荷は良好な燃料装荷パターン生成のための燃料装荷でなければならない。したがって、本項では、最善の燃料集合体選択を模擬するための NN 作成方法を説明する。

N-MCTS における最善の燃料集合体選択を定式化すると式(5.5)になる。

$$\boldsymbol{p} = y(\boldsymbol{s}) \tag{5.5}$$

ここで、$\boldsymbol{s}$は部分的な燃料装荷パターン、$\boldsymbol{p}$は各燃料集合体を装荷する確率である。学習では、ニューラルネットワーク$\boldsymbol{y(s)}$を作成することが目的である。これは、部分的な燃料装荷

パターンを入力データ、最善な燃料集合体の選択を出力データとする学習データを用いることで達成される。

最善の燃料集合体選択を模擬した学習データの作成方法を図 5-11 に示す。手順は以下のとおりである。

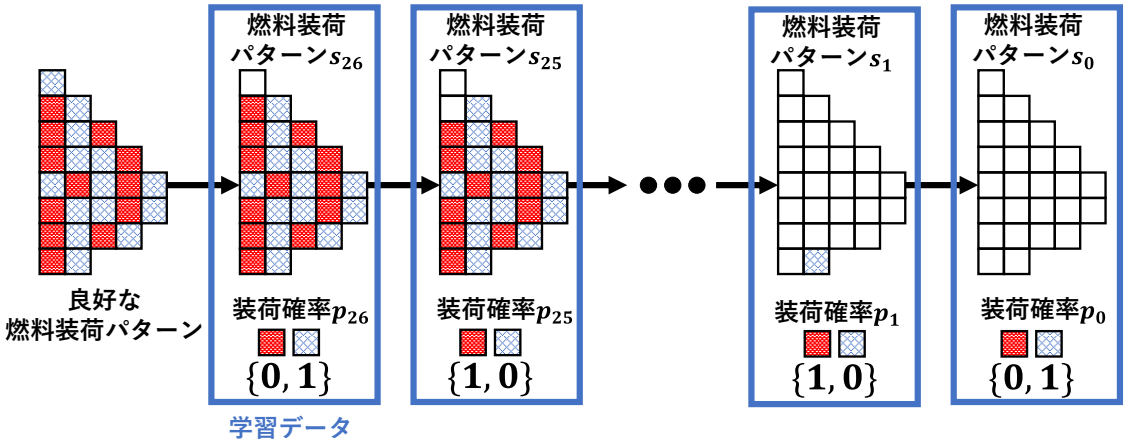



図 5-11　学習データの作成

A)　良好な燃料装荷パターンを従来の代表的な手法などを用いて作成する。

B)　良好な燃料装荷パターンから全く装荷されていない燃料装荷パターンまでの経路に沿って、学習データを作成する。それぞれの学習データは、燃料装荷パターンのベクトルを入力データ、装荷される燃料集合体を 1 次元の one-hot ベクトル（1 の要素が 1 つだけあり、そのほかの要素が 0 であるベクトル）を出力データとしている。燃料装荷パターンのベクトルを図 5-12 に示す。無限増倍率、濃縮度、燃焼度、可燃性毒物の有無の 4 つの特徴量を用いて、燃料装荷パターンを表現する。それぞれの特徴量のスケールをなるべくそろえるために、燃料が装荷されていない位置は−1、燃料が装荷されている位置は考えられる最大値と最小値を用いて、式(5.6)のように正規化している。

$$x_{\mathrm{norm}} = \frac{x_{\mathrm{data}} - \min}{\max - \min} \tag{5.6}$$

ここで、$x_{\mathrm{data}}$は正規化前の値、maxとminは考えられる最大値と最小値であり、第 4 章で用いた燃料インベントリの燃焼度は最大 30 GWd/t、最小 0 GWd/t であるため、燃焼度の正規化を行う場合は$\max = 30, \min = 0$となる。学習データに同じ燃料装荷パターンがある場合、より良好な燃料装荷パターンになる学習データを優先する。1 つの燃料装荷パターンに対して、有望な燃料装荷が 2 つ以上存在することがないように設定し、有望な燃料装荷の学習を安定させる。

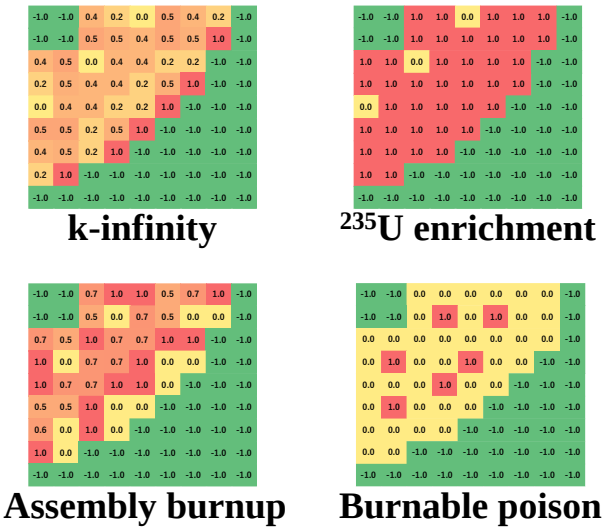



図 5-12　燃料装荷パターンのベクトル表現
将棋と同じように9 × 9のベクトルである

C)　学習データの入力ベクトルと出力ベクトルをそれぞれ$\boldsymbol{x}, \boldsymbol{t}$、学習データの入力$\boldsymbol{x}$に対する NN の出力ベクトルを$\hat{\boldsymbol{t}} = \boldsymbol{y}(\boldsymbol{x})$とすると、$\boldsymbol{t}$と$\hat{\boldsymbol{t}}$の差が小さくなるように NN のパラメータの更新を行う。損失関数とパラメータ更新手法は 5.3 節で述べる。

以上の手順によって、最善の燃料集合体選択を模擬するための NN を得ることができる。

### 5.2.4　N-MCTS の探索アルゴリズム

燃料装荷パターン最適化における N-MCTS のメカニズムを図 5-13 に示す。

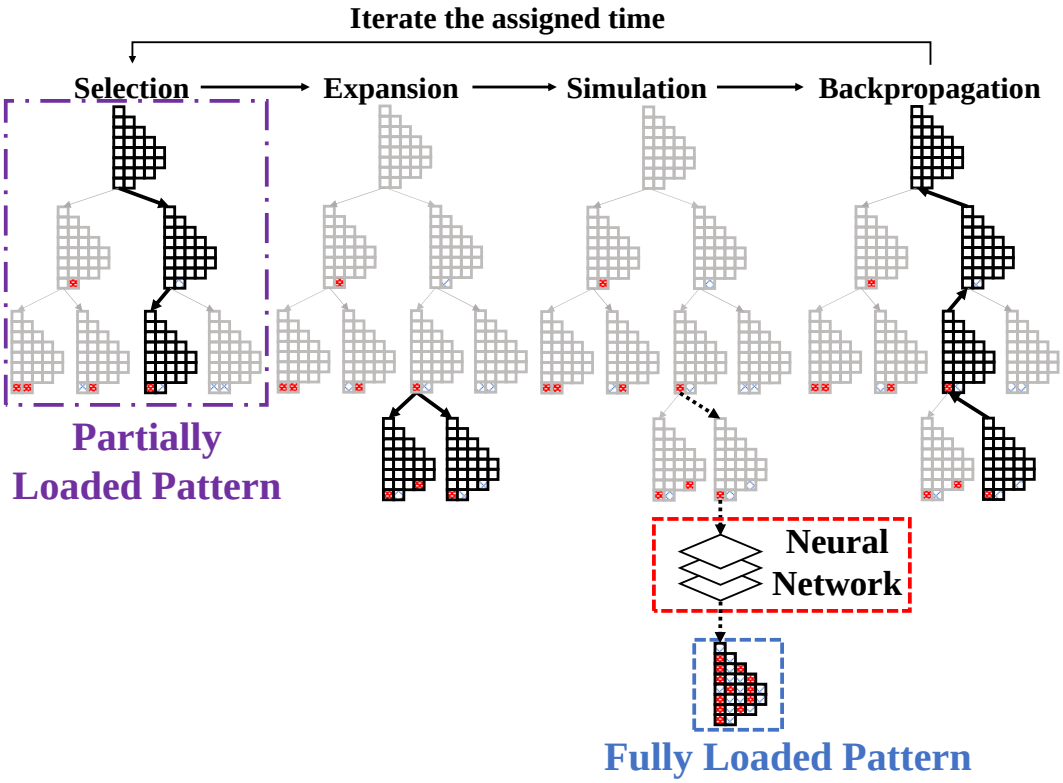



図 5-13　燃料装荷パターン最適化における N-MCTS のメカニズム
装荷する燃料集合体が 2 種類の場合

N-MCTS は、最善の燃料集合体選択を模擬するために、MCTS のランダムプレイアウトをニューラルネットワークに従うプレイアウトに置き換えるのみである。ニューラルネットワークの学習は MCTS の探索前に行われる。読者が 4.2.2 項に戻り、MCTS の探索アルゴリズムを再び見ることがないように、同様の説明を以下で行う。

N-MCTS は、燃料装荷パターン最適化の間、選択（Selection）、展開（Expansion）、シミュレーション（Simulation）、バックプロパゲーション（Backpropagation）の 4 つのステップを繰り返す。これらのステップを以下で説明する。

**選択（Selection）**：選択では、根の燃料装荷パターン（全く装荷されていない燃料装荷パターン）を始まりとして、葉の燃料装荷パターン（子を持たない部分的な燃料装荷パターン）にたどり着くまで、3.5.2 項で示した Upper Confidence Bounds applied to trees（UCT）に基づいて子の燃料装荷パターン（部分的な燃料装荷パターンに対して燃料を 1 体だけ追加で装荷した燃料装荷パターン）を選び続ける。探索中の燃料装荷パターンを$s$とすると、次に探

索する燃料装荷パターン$c^*$は式(5.7)で与えられる。

$$c^* = \underset{c \in \text{children of } s}{\arg\max} \left( Q(c) + C_p \sqrt{\frac{\ln N(s)}{N(c)}} \right) \tag{5.7}$$

ここで、$Q(c)$はこれまでの探索で得られた燃料装荷パターン$c$の期待値、$N(c)$はこれまでの探索で燃料装荷パターン$c$を訪問した回数、$N(s)$は燃料装荷パターン$s$を訪問した回数である。また、$C_p$は探索と活用のバランスを決める正の定数である。期待値が大きい場合、第 1 項が大きくなり、頂点$c$の探索回数が少ない場合、第 2 項が大きくなる。$C_p$が大きな正の値の場合、探索回数が少ない燃料装荷パターンが選択されるようになる（探索）。一方、$C_p$が小さな正の値の場合、期待値が大きい燃料装荷パターンが選択される（活用）。選択によって、期待値が大きい燃料装荷パターンを選択しながら、訪問回数が少ない燃料装荷パターンもバランス良く探索することができる。

燃料装荷パターン最適化における選択の例を図 5-14 に示す。選択は根の燃料装荷パターン（全く装荷されていない燃料装荷パターン）から探索が始まる。根の燃料装荷パターンの子を図 5-15 に示す。探索中の燃料装荷パターンは、炉心の中心に 1 体だけ燃料を装荷した燃料装荷パターンである。右の子の UCT 値が大きいと仮定すると、右の子の燃料装荷パターンが選択される。ここで、UCT 値は式(5.8)で与えられる。

$$Q(c) + C_p \sqrt{\frac{\ln N(s)}{N(c)}} \tag{5.8}$$

同様にして、選択された深さ 1 の燃料装荷パターンから UCT によって左の子の燃料装荷パターンが選択されたとすると、新たに選択された深さ 2 の燃料装荷パターンは葉の燃料装荷パターン（子を持たない部分的な燃料装荷パターン）であるため、選択は終了する。

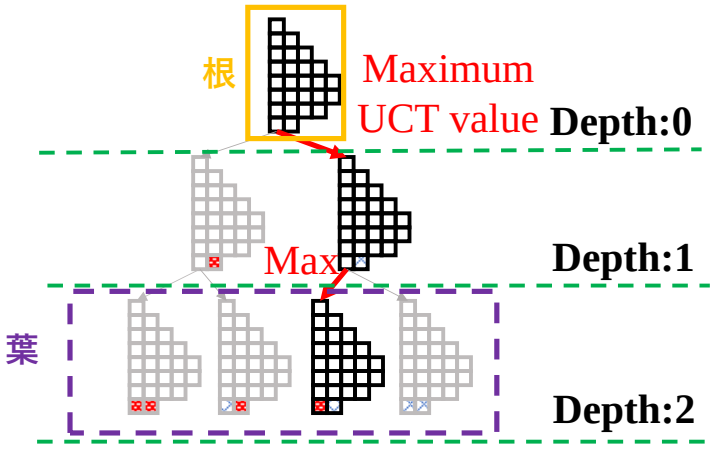
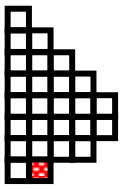
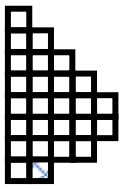



図 5-14　燃料装荷パターン最適化における選択の例
装荷する燃料集合体が 2 種類の場合



(a)　図 5-14 における左の子



(b)　図 5-14 における右の子

図 5-15　根の子の燃料装荷パターン

**展開（Expansion）**：展開では、選択でたどり着いた葉の燃料装荷パターンに対して、子の燃料装荷パターンを木に追加する。選択でたどり着いた葉の燃料装荷パターンが、完全な燃料装荷パターンであれば、展開は行われない。展開によって、木を深くし、期待値を評価する燃料装荷パターンを増やすことができる。

燃料装荷パターン最適化における展開の例を図 5-16 に示す。選択でたどり着いた燃料装荷パターン（深さ 2 の燃料装荷パターンのうち右から 2 番目）の子を木に追加する。この燃料装荷パターンは、選択でたどり着いた燃料装荷パターンに 1 体だけ追加で燃料集合体を装荷した燃料装荷パターンである。

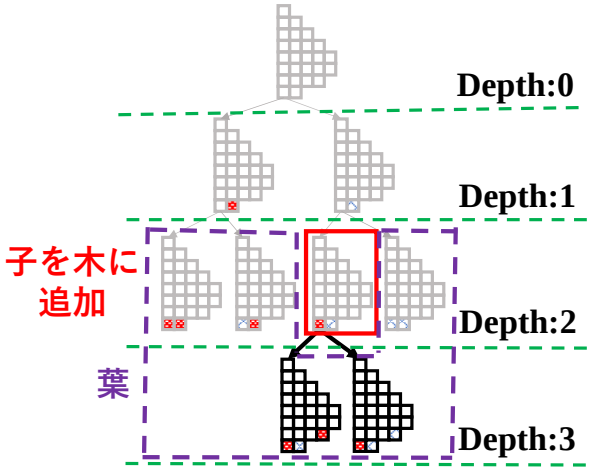



図 5-16　燃料装荷パターン最適化における展開の例
装荷する燃料集合体が 2 種類の場合

**シミュレーション（Simulation）**：シミュレーションでは、たどり着いた葉の燃料装荷パターンから完全な燃料装荷パターンまで、事前に学習したニューラルネットワークに従って装荷を選び続ける。完全な燃料装荷パターンは炉心解析コードで炉心特性が評価され、得られた炉心特性値から報酬$r$が計算される。

NN を用いたシミュレーションの例を図 5-17 に示す。部分的な燃料装荷パターンが NN に入力され、すべての実行可能な燃料装荷に対して、最善な燃料装荷である確率が出力され

る。完全な燃料装荷パターンになるまで、出力された確率が最大である燃料装荷を行い、炉心特性に基づき報酬$r$を得る。

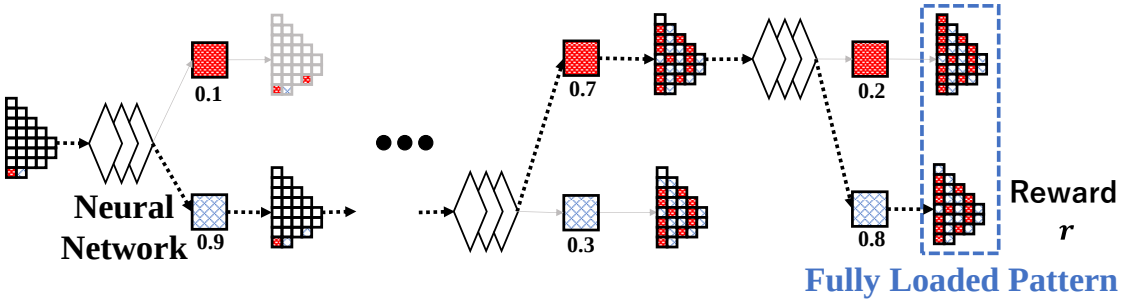



図 5-17　NN を用いた燃料装荷パターン最適化におけるシミュレーションの例
装荷する燃料集合体が 2 種類の場合

**バックプロパゲーション（Backpropagation）**：バックプロパゲーションでは、新たに葉になった燃料装荷パターンから根までの経路に沿って、頂点の訪問回数がインクリメント（+1）されるとともに、累積報酬と期待値が更新される。更新式を式(5.9)、式(5.10)、式(5.11)に示す。ここで、$s$は新たに葉になった燃料装荷パターンから根の燃料装荷パターンまでの経路に沿ったすべての頂点の要素である。

$$N(s) = N(s) + 1 \tag{5.9}$$

$$W(s) = W(s) + r \tag{5.10}$$

$$Q(s) = W(s)/N(s) \tag{5.11}$$

燃料装荷パターン最適化におけるバックプロパゲーションの例を図 5-18 に示す。新たに葉になった燃料装荷パターンから根の燃料装荷パターンまでの経路に沿った頂点の統計量が更新される。図 5-18 では新たに葉になった燃料装荷パターンの深さが 3 のため、4 つの燃料装荷パターン（経路の含まれる深さが $3, 2, 1, 0$ の燃料装荷パターン）の統計量が更新される。

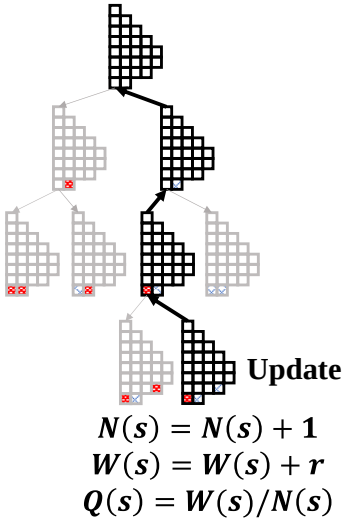



図 5-18 燃料装荷パターン最適化におけるバックプロパゲーションの例
装荷する燃料集合体が 2 種類の場合

以上の 4 つのステップ（選択、展開、シミュレーション、バックプロパゲーション）を決められた回数繰り返すことによって、良好な燃料装荷パターンを効率的に探索している。探索中に得られた最良の報酬を持つ完全な燃料装荷パターンを最適な燃料装荷パターンとする。

### 5.3 計算条件

第 4 章と同様に、計算体系は図 5-19 に示す 1/8 対称性を持つ PWR 炉心、炉心計算は簡易計算版 ICEBURN を用いて行う。また、燃料装荷パターン最適化の目的は第 4 章と同様に、サイクルを通じた径方向ピーキング係数の最小化であり、完全な燃料装荷パターンの報酬を径方向ピーキング係数の逆数とする。最大の報酬を持つ完全な燃料装荷パターンを最良な燃料装荷パターンとする。

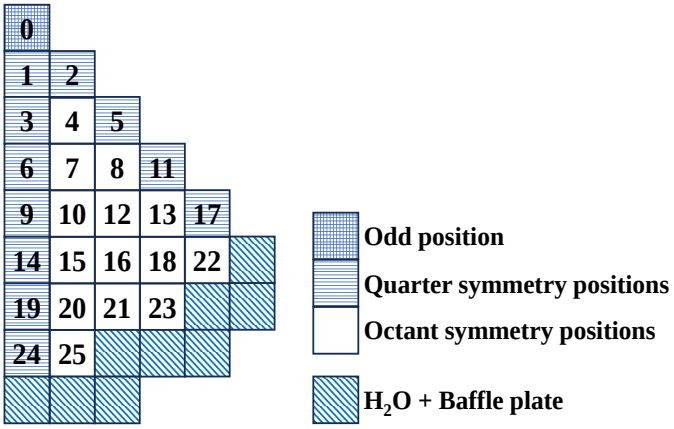



図 5-19 1/8 対称性を持つ PWR 炉心

本検討では、N-MCTS の最適化性能と汎化性能を評価するために、N-MCTS の最適化結果を MCTS、SA の最適化結果と比較する。100 種類の燃料インベントリに対して燃料装荷パターン最適化を行い、結果を統計的に処理する。

100 種類の燃料インベントリは、表 5-1 に示す標準燃料インベントリにランダムな摂動を与えて作成する。表 5-1 に示す標準燃料インベントリは第 4 章で用いたインベントリと同じである。標準燃料インベントリのランダムな摂動では、新燃料（燃焼度が 0 GWd/t）体数を固定し、燃焼燃料 4 種類（3wt% 30 GWd/t, 4wt% 15 GWd/t, 4wt% 20 GWd/t, 4wt% 30 GWd/t）の体数の総和が一定である条件下で、それぞれの燃焼燃料体数をランダムに変更する。摂動させた燃料インベントリの例を表 5-2 に示す。新燃料体数を固定する理由は、1.1.1 項で示したように、新燃料は長期計画に基づいて手配されているためであり、炉心内の燃料集合体の状態が不明であることを模擬している。摂動させた燃料インベントリの例を表 5-2 に示す。考えられる摂動させた燃料インベントリの個数は

$$\text{Odd の通り} \times \text{Quarter の通り} \times \text{Octant の通り}$$

$$= \binom{1+4-1}{4-1} \times \binom{10+4-1}{4-1} \times \binom{8+4-1}{4-1} \approx 10^5$$

である。それぞれの燃料インベントリで考えられる燃料装荷パターンの数が、4 章と同じく $10^{13}$ と仮定すると、考えられるすべての燃料装荷パターン数はおよそ $10^{18}$ である。1 つの燃料装荷パターン当たり 10 分で詳細な炉心計算ができると仮定した場合、すべての燃料装荷パターンの炉心計算を行うには 10 兆年以上の時間がかかる。

表 5-1　標準燃料インベントリ

新燃料(体数固定)は赤色で塗られている

| **U235 Enrichment [wt%]** | **Burnup [GWd/t]** | **Burnable Poison** | **Symmetry** | **Quantity** |
|---|---|---|---|---|
| 3 | 30 | - | Odd | 1 |
| 3 | 30 | - | Quarter | 2 |
| 4 | 0 | - | Octant | 3 |
| 4 | 15 | - | Quarter | 3 |
| 4 | 15 | - | Octant | 2 |
| 4 | 20 | - | Quarter | 2 |
| 4 | 20 | - | Octant | 3 |
| 4 | 30 | - | Quarter | 3 |
| 4 | 30 | - | Octant | 3 |
| 4 | 0 | Gd | Quarter | 1 |
| 4 | 0 | Gd | Octant | 3 |
| **Total** | | | | 26 |

表 5-2　摂動させた燃料インベントリの例

| U235 Enrichment [wt%] | Burnup [GWd/t] | Burnable Poison | Symmetry | Quantity (標準) | Quantity (例 1) | Quantity (例 2) | Quantity (例 3) |
|---|---|---|---|---|---|---|---|
| 3 | 30 | - | Odd | 1 | 0 | 1 | 1 |
| 3 | 30 | - | Quarter | 2 | 7 | 5 | 3 |
| **3** | 30 | - | Octant | 0 | 5 | 4 | 1 |
| 4 | 0 | - | Octant | 3 | 3 | 3 | 3 |
| **4** | 15 | - | Odd | 0 | 0 | 0 | 0 |
| 4 | 15 | - | Quarter | 3 | 0 | 1 | 0 |
| 4 | 15 | - | Octant | 2 | 0 | 0 | 3 |
| **4** | 20 | - | Odd | 0 | 0 | 0 | 0 |
| 4 | 20 | - | Quarter | 2 | 3 | 0 | 5 |
| 4 | 20 | - | Octant | 3 | 3 | 0 | 2 |
| **4** | 30 | - | Odd | 0 | 1 | 0 | 0 |
| 4 | 30 | - | Quarter | 3 | 0 | 4 | 2 |
| 4 | 30 | - | Octant | 3 | 0 | 4 | 2 |
| 4 | 0 | Gd | Quarter | 1 | 1 | 1 | 1 |
| 4 | 0 | Gd | Octant | 3 | 3 | 3 | 3 |
| **Total** | | | | 26 | 26 | 26 | 26 |

N-MCTS と MCTS の最適化では、外側から装荷が行われる（図 5-19 で示される燃料装荷位置番号の降順に燃料装荷する）。少ない炉心計算回数で良好な燃料装荷パターンを求めることを目的としているため、最適化における炉心計算回数は 1000 回である。

N-MCTS で用いる NN は図 5-20 に示すものであり、学習で更新するパラメータ数は 166480 である。Conv 3 × 3 Padding, Norm, ReLUは、Padding 処理を行ったベクトルに対して、3 × 3のフィルターを用いた畳み込み、バッチ正規化（Batch Normalization）、ReLU を順に行うことを示している。バッチ正規化では、各チャネルのスケールをそろえることを目的として平均 0 分散 1 になるように、ベクトルの要素を正規化している。入力は9 × 9のベクトル 4 つ（無限増倍率、濃縮度、燃焼度、可燃性毒物の有無）、出力は燃焼燃料 4 種類（3wt% 30GWd/t, 4wt% 15GWd/t, 4wt% 20GWd/t, 4wt% 30GWd/t）から装荷燃料を選ぶため、4 つの要素を持つ総和が 1 のベクトルである。

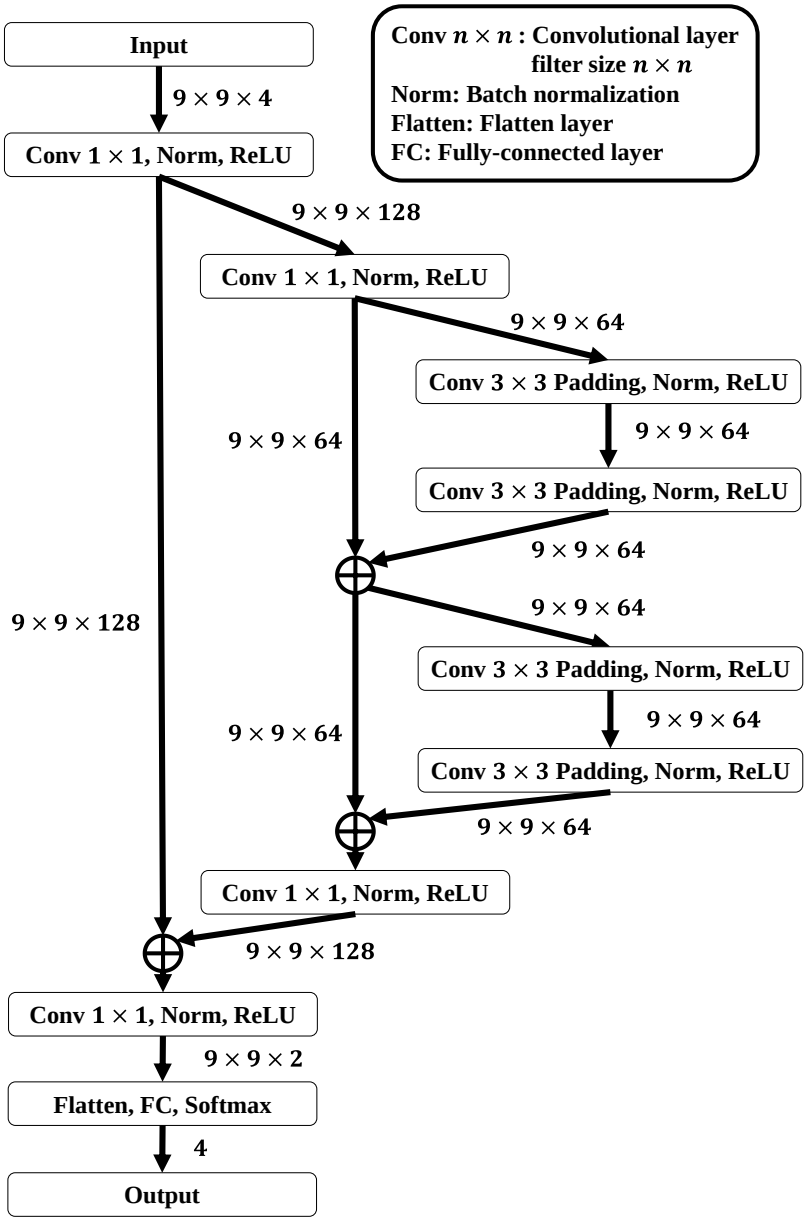



図 5-20　本研究で用いる NN（図 5-10 の再掲）

学習データは、標準インベントリを SA（炉心特性計算回数 10000 回）で最適化した 100 個の良好な燃料装荷パターンまでの経路を用いており、学習データの総数は 1853 である。燃料装荷パターンが重複した場合、良い燃料装荷パターンのみを学習データとしているため、$26 \times 100 = 2600$より小さい学習データ数となる。学習データとして摂動させた燃料インベントリを用いない理由は、N-MCTS の汎化性能（燃料インベントリが異なる場合でも、NN によって良好な装荷を予測できるか）を評価するためである。なお、SA では、初期の燃料装荷パターンがランダムで生成され、対称性が同じ燃料集合体の交換により最適化を行う。

最適化前に行う NN のパラメータ更新は、学習データ 1 つごとに行い（オンライン学習）、勾配降下法を改良した AdamW[11], [12]を利用する。学習率の初期値は$10^{-4}$と設定し、学習データ 1853 個に対する正解率が{50, 80, 90%}を上回るごとに学習率を半分に減らす。ここで、正解とは、NN で出力される最大の確率を持つ燃料装荷と学習データの燃料装荷が同じことである。また、損失関数$L$は 2.6.1 項で紹介したクロスエントロピー損失を用いる。クロスエントロピー損失を式(5.12)に示す。

$$L = -\sum_{i=1}^{r} t_i \log(\hat{t}_i) \tag{5.12}$$

ここで、$t_i$は学習データの出力ベクトル$\boldsymbol{t}$の$i$番目の要素、$\hat{t}_i$は学習データの入力ベクトルに対する NN の出力ベクトル$\hat{\boldsymbol{t}}$の$i$番目の要素である。

数値計算には、Intel Core i9-10980XE CPU @ 3.00GHz、NVIDIA RTX A4000 GPU、C++、PyTorch C++ API[13]を用いる。

## 5.4 計算結果

学習回数（Epoch 数）と NN の正解率の関係を図 5-21 に示す。学習回数（Epoch 数）1 回とは、1853 個の学習データの一つ一つを用いて 1853 回パラメータ更新を行うことである。およそ 7 時間で 1000 回の学習を行い、正解率はおよそ 98%である。高い正解率を達成し、将棋や囲碁などのボードゲームと同様に、燃料装荷パターンのパターン認識が可能であることが分かる。

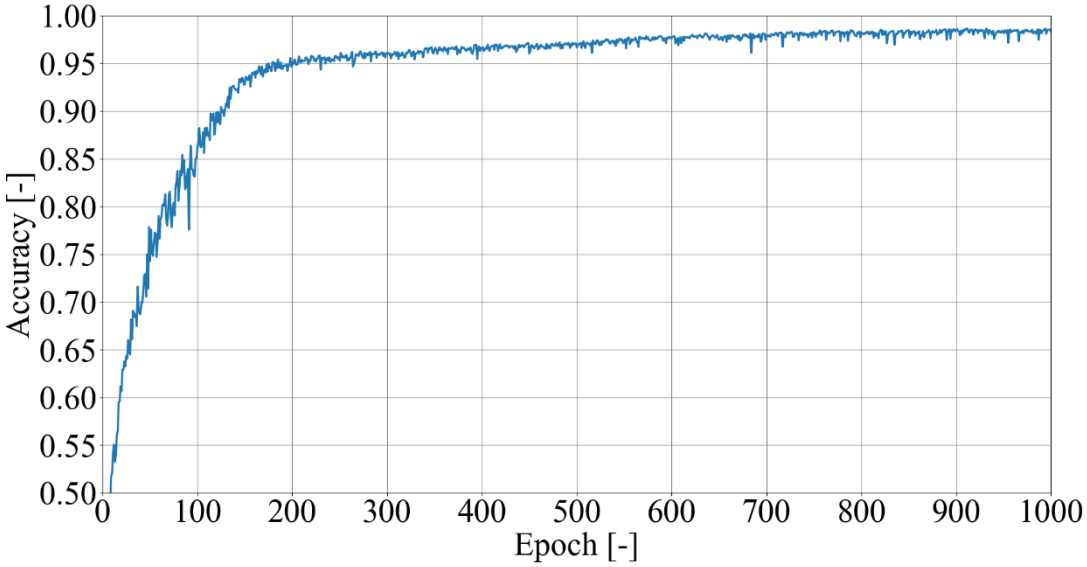



図 5-21　学習回数と正答率の関係

次に、100 種類の燃料インベントリに対して燃料装荷パターン最適化を行った結果を表 5-3 に示す。この最適化における MCTS および SA の炉心特性計算回数はそれぞれ 1000 回

である。また、100 回の燃料装荷パターン最適化における最小の径方向ピーキング係数の N-MCTS と MCTS の比較、N-MCTS と SA の比較をそれぞれ図 5-22 と図 5-23 に示す。最小の径方向ピーキング係数の平均では、N-MCTS が MCTS を 0.044 ほど最小化することができているとともに、標準偏差が小さく安定した最適化が可能である。

表 5-3　100 種類の燃料装荷パターン最適化で得られた
最小の径方向ピーキング係数の平均及び標準偏差

| Method | Average | Std. Dev. |
|---|---|---|
| **N-MCTS** | **1.413** | **0.113** |
| **MCTS** | 1.457 | 0.236 |
| **SA** | 1.539 | 0.203 |

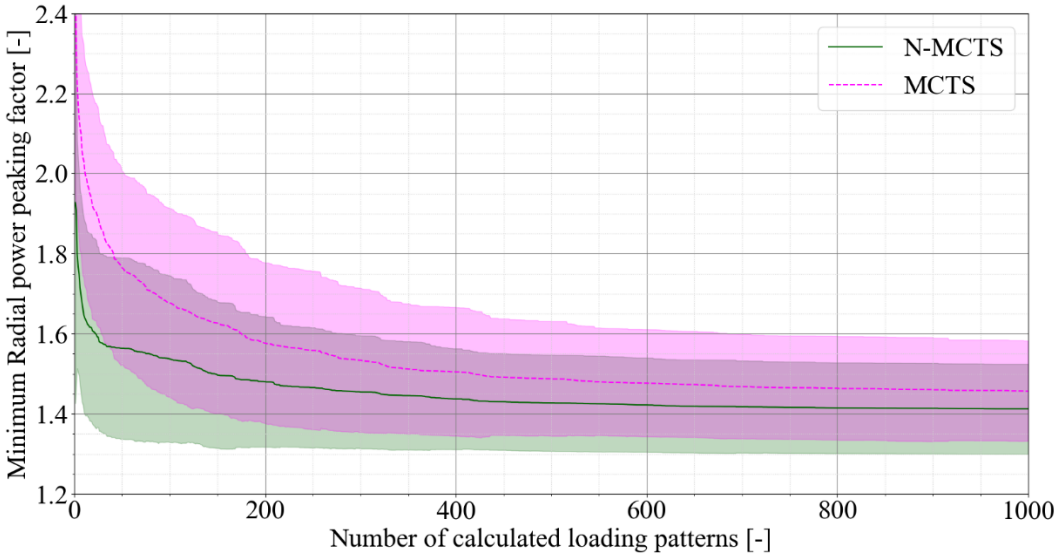



図 5-22　N-MCTS と MCTS における最適化中の最小の径方向ピーキング係数
100 回の最適化における平均であり、ハッチング部分は標準偏差(1σ)を示す

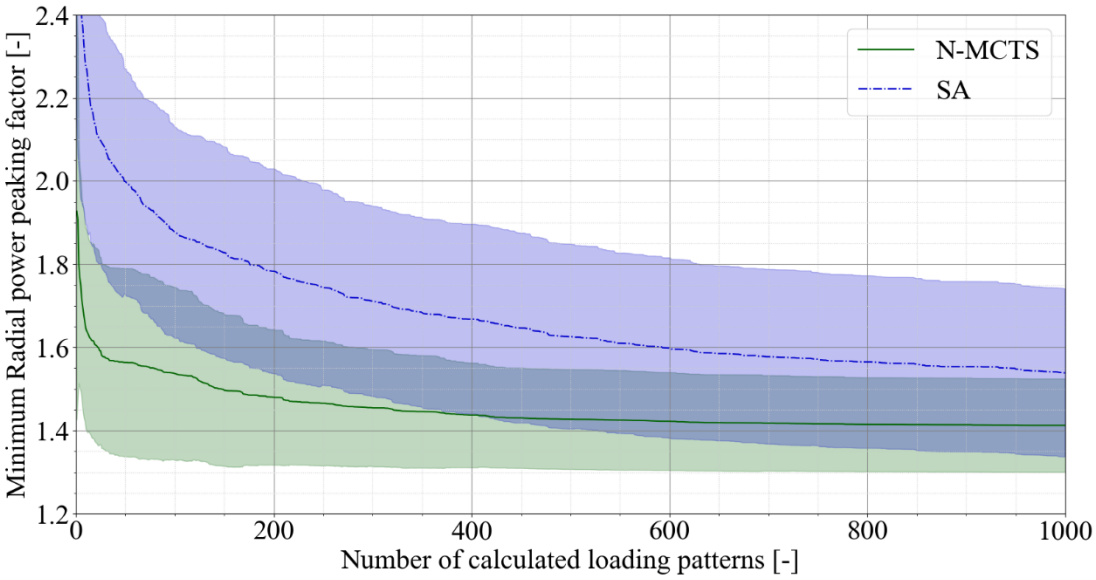



図 5-23　N-MCTS と SA における最適化中の最小の径方向ピーキング係数
100 回の最適化における平均であり、ハッチング部分は標準偏差(1σ)を示す

また、100 種類の燃料インベントリに対して燃料装荷パターン最適化において、MCTS に対する N-MCTS の勝率を図 5-24 に、SA に対する N-MCTS の勝率を図 5-25 に示す。ここで、勝ちとは N-MCTS が比較手法より小さい径方向ピーキング係数を得ることである。MCTS に対する N-MCTS の勝率は 80%であり、優れた最適化性能及び汎化性能を持つことが確認できる。

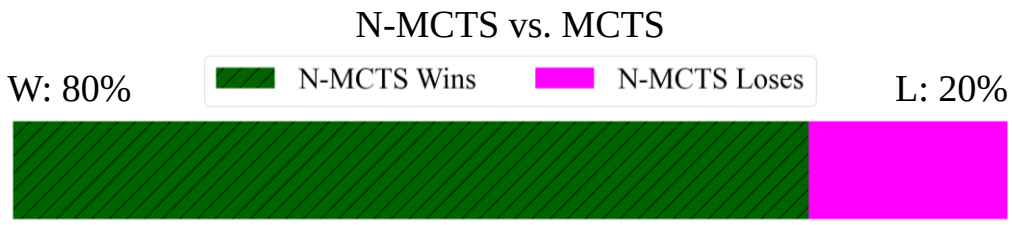



図 5-24　MCTS に対する N-MCTS の勝率

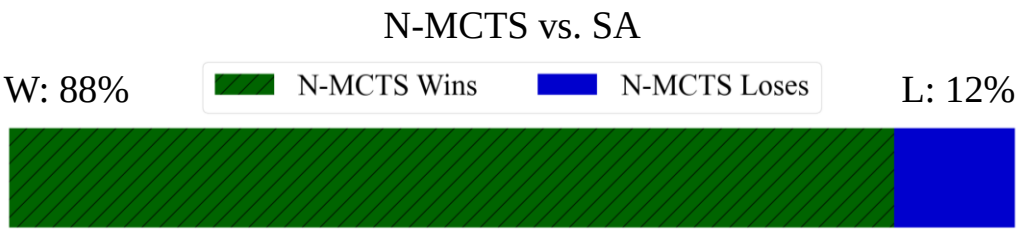



図 5-25　SA に対する N-MCTS の勝率

## 5.5　考察

N-MCTS は優れた最適化性能及び汎化性能を持つことが確認された。最適化の詳細を確認するために、N-MCTS と MCTS の最適化における最小ピーキング係数の探索例を図 5-26 に示す。図 5-26 では、N-MCTS は最適化初期からピーキング係数が小さい燃料装荷パターンを見つけることがわかる。ニューラルネットワークに従った装荷によって、有望な装荷ができていることを示している。

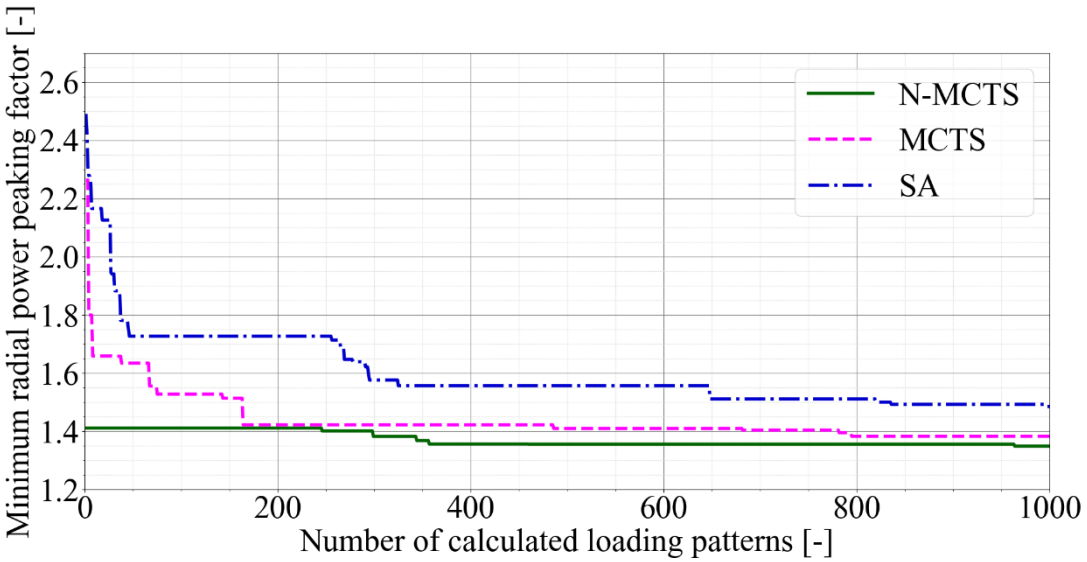



図 5-26　最適化における最小ピーキング係数の探索例

また、最適化で行った 1000 回の炉心計算で得られたピーキング係数の頻度分布例を図 5-27 と図 5-28 に示す。図 5-27 と図 5-28 に示すように、N-MCTS の最適化で得られるピーキング係数は最適化を通じて、全体的に小さいことがわかる。初期の最適化だけでなく、木を構築しながら NN に従う燃料装荷を行うことで、効率的に良好な燃料装荷パターンを探索することが可能である。

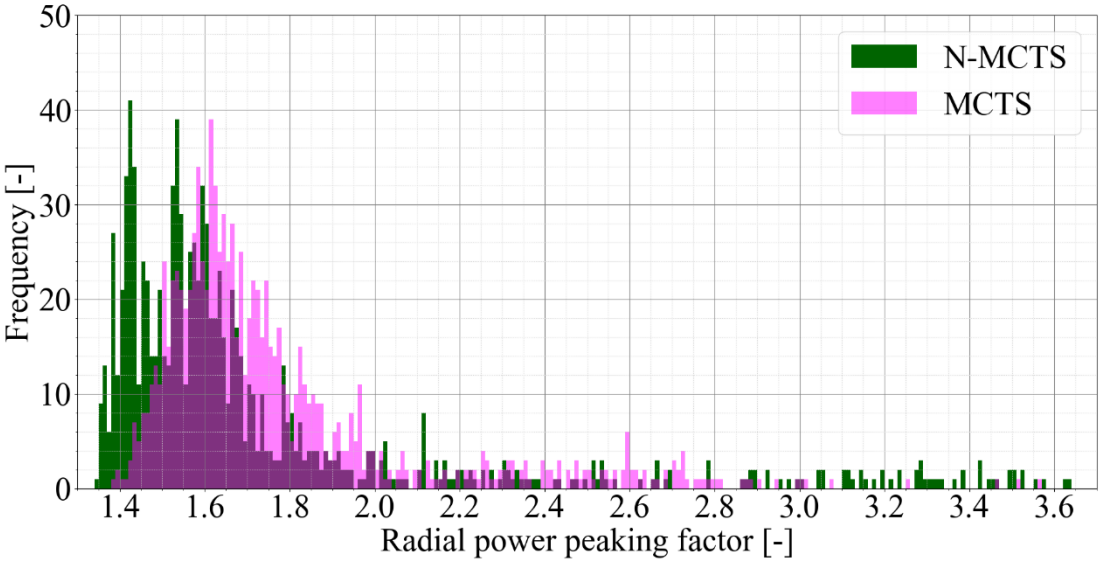



図 5-27　N-MCTS と MCTS の最適化で得られたピーキング係数の頻度分布例

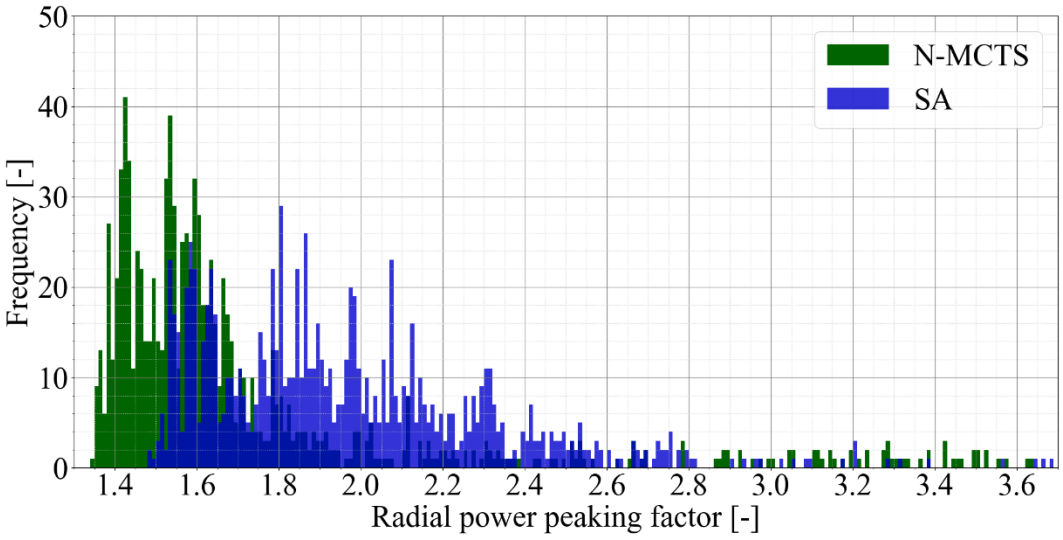



図 5-28　N-MCTS と SA の最適化で得られたピーキング係数の頻度分布例

## 5.6　本章のまとめ

本章では、少ない回数で良好な燃料装荷パターンを見つけることができる MCTS と深層学習を組み合わせることで、さらなる性能の向上を目的とした。MCTS と深層学習を組み合わせた手法（N-MCTS）を説明し、N-MCTS の最適化性能及び汎化性能の確認のために数値実験を行った。

5.2 節では、N-MCTS の説明を行った。5.2.1 項では、本手法は MCTS のランダムな燃料装荷を深層学習による有望な燃料装荷に置き換えていることを述べた。5.2.2 項では、深層学習を用いたゲーム AI に使用されている ResNet を解説した。5.2.3 項では、有望な燃料装荷

を行う NN の作成方法について説明し、5.2.4 項では、NN と MCTS を組み合わせた手法を説明した。MCTS のシミュレーションでは、NN に従う有望な燃料装荷を行う。

5.3 節では、N-MCTS の最適化性能及び汎化性能を確認するための計算条件を示した。燃料装荷パターン最適化を行う燃料インベントリは、学習した燃料インベントリを摂動させた燃料インベントリである。摂動させた燃料インベントリを最適化することで、N-MCTS の汎化性能及び最適化性能を評価することができる。

5.4 節では、N-MCTS の最適化結果を MCTS と SA の最適化結果を比較し、少ない炉心計算回数で N-MCTS は優れた最適化性能及び汎化性能を持ち、良好な燃料装荷パターンを生成できることが確認された。

5.5 節では、N-MCTS は優れた最適化性能及び汎化性能を持つ理由を考察した。N-MCTS では、NN を用いた燃料装荷によって有望な燃料装荷が行われるため、最適化の早い段階で良好な燃料装荷パターンを見つけることができる。また、MCTS による木の構築によって、さらに良好な燃料装荷パターンを見つけることができる。

以上のことから、N-MCTS は、高い汎化性能と最適化性能を持ち、少ない炉心計算回数で良好な燃料装荷パターンを見つけることが可能である。

## 5.7 参考文献


[1] D. Silver *et al.*, “A General Reinforcement Learning Algorithm that Masters Chess, Shogi, and Go through Self-Play,” *Science*, vol. 362, no. 6419, pp. 1140–1144, Dec. 2018, doi: 10.1126/science.aar6404.

[2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[3] 布留川英一, *AlphaZero 深層学習・強化学習・探索　人工知能プログラミング実践入門*, 東京, ボーンデジタル, 2019.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, Cambridge, Massachusetts, USA: MIT Press, 2016 (訳: 黒滝紘生 ほか, *深層学習*, 東京, KADOKAWA, 2018).

[5] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning. Springer Series in Statistics*. New York, USA: Springer New York, 2009. doi: 10.1007/978-0-387-84858-7 (訳: 杉山将 ほか, *統計的学習の基礎 : データマイニング・推論・予測*, 東京, 共立出版, 2014).

[6] C. C. Aggarwal, *Neural Networks and Deep Learning*, Switzerland: Springer Cham, 2018 (訳: 李鍾賛 ほか, *ニューラルネットワークとディープラーニング*, 東京, 学術図書出版社, 2022).

[7] 石川聡彦, *Python で動かして学ぶ!あたらしい深層学習の教科書 : 機械学習の基本から深層学習まで*, 東京, 翔泳社, 2018.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 248–255. doi: 10.1109/CVPR.2009.5206848.

[9] C. Wan, K. Lei, and Y. Li, “Optimization Method of Fuel-Reloading Pattern for PWR Based on the Improved Convolutional Neural Network and Genetic Algorithm,” *Ann. Nucl. Energy*, vol. 171, p. 109028, Jun. 2022, doi: 10.1016/j.anucene.2022.109028.

[10] D. J. Wu, “GTP Engine and Self-Play Learning in Go”, https://github.com/lightvector/KataGo (accessed: Feb. 11, 2024).

[11] I. Loshchilov and F. Hutter, “Decoupled Weight Decay Regularization,” presented at *the International Conference on Learning Representations*, Sep. 2018. [Online]. Available: https://openreview.net/forum?id=Bkg6RiCqY7 (accessed: Feb. 11, 2024).

[12] PyTorch, “Class AdamW; PyTorch Main Documentation,” https://pytorch.org/cppdocs/api/classtorch_1_1optim_1_1_adam_w.html (accessed: Feb. 25, 2024).

[13] PyTorch, “C++; Pytorch 2.2 Documentation”, https://pytorch.org/docs/stable/cpp_index.html (accessed: Feb. 25, 2024).

# 第6章　結論

## 6.1　まとめ

本節では、本研究の目的、手法、結果を簡潔に述べた後、各章における内容をまとめる。

本研究の目的は、少ない炉心特性計算回数で良好な燃料装荷パターンを発見する最適化手法の開発である。本研究の目的を達成するために、以下の 3 つを新たに行った。

1. グラフ理論のアプローチを用いて、燃料装荷パターン最適化をゲーム木探索に帰着
2. 将棋や囲碁などのゲーム AI で使用されているモンテカルロ木探索（Monte Carlo Tree Search, MCTS）を燃料装荷パターン最適化のゲーム木に適用
3. 効率的なゲーム木探索のために、MCTS と深層学習を組み合わせる

MCTS を用いた燃料装荷パターン最適化では、従来の代表的な最適化手法である焼きなまし法（Simulated Annealing, SA）に比べて、少ない炉心特性計算回数で良好な燃料装荷パターンを見つける可能性が高い。さらに、MCTS と深層学習を組み合わせることで、さらに良好な燃料装荷パターンを見つけることが可能である。

各章に関する内容を以下にまとめる。

第 1 章では、燃料装荷パターンは原子炉の安全かつ経済的な運転を行う上で重要な役割を果たすことを述べた。実際の燃料装荷パターンの設計では、次の要因によって時間的な制約が生じる。

・約 2 か月の定期検査内で燃料装荷パターンを決定することが必要
・事業者のニーズに適合した複数の燃料装荷パターンの候補を求めることが必要
・膨大な燃料装荷パターンから良好な燃料装荷パターンを見つけることが必要
・1 つの燃料装荷パターンの詳細な炉心特性を求めるためには数分～数十分を要する

従来の最適化手法では、良好な燃料装荷パターンを得るために、数千～数万の燃料特性計算を行わなければならないため、時間的な制約を満たすことが難しいという課題がある。

そこで、少ない炉心特性計算回数で良好な燃料装荷パターンを得ることを目的として、ゲーム AI と深層学習を組み合わせた手法に注目した。ゲーム AI と深層学習を組み合わせた手法は、ボードゲームで世界のトッププログラムであるだけでなく、巡回セールスマン問題などの最適化にも応用されており、燃料装荷パターン最適化への適用を考えることは有益である。

第 2 章では、燃料装荷パターン最適化と最適化手法の理解を深めるために、代表的な最適化手法である焼きなまし法（Simulated Annealing, SA）、遺伝的アルゴリズム（Genetic Algorithm, GA）、蟻コロニー最適化（Ant Colony Optimization, ACO）を用いた燃料装荷パターン最適化について解説した。SA は金属結晶の生成過程、GA は生物の進化過程、ACO は

蟻の集団行動を模擬することで、効率的な探索を行う。また、機械学習を用いた燃料装荷パターン最適化について言及した。近年、コンピュータ性能と計算アルゴリズムの向上により、機械学習を用いて炉心特性の高速な予測が可能である。

第 3 章では、将棋や囲碁のゲーム AI で使用されている探索手法であるモンテカルロ木探索（Monte Carlo Tree Search, MCTS）の説明を行った。グラフ理論における木とは、長さ 1 以上の閉路を持たない連結な無向グラフのことであり、ゲームの局面から次の局面に辺を接続するとゲーム木になる。次に、MCTS を説明した。MCTS は、選択（Selection）、展開（Expansion）、シミュレーション（Simulation）、バックプロパゲーション（Backpropagation）の 4 つのステップを繰り返すことで、木を探索および構築するアルゴリズムである。モンテカルロ木探索の大きな特徴は、新しい局面の探索と過去の探索で得た情報の活用による有望な局面の探索をバランスよく行うことである。

第 4 章では、燃料装荷パターン最適化をゲーム木に変換する方法について述べ、MCTS を用いた最適化を行った。燃料装荷パターン最適化をゲーム木に変換するために、全く燃料が装荷されていない炉心に燃料集合体を 1 体ずつ装荷し、炉心が燃料集合体で満たされている燃料装荷パターンを作成するアプローチを新たにとった。PWR 炉心のサイクルを通じた径方向ピーキング係数を最小化するという目的で最適化をおこない、MCTS と従来の最適化手法である SA の最適化結果を比較した。SA に比べて、100 試行で得られる最小のピーキング係数の平均が 0.05 ほど小さくなり、少ない炉心特性計算回数で良好な燃料装荷パターンを見つける可能性が高いという結果が得られた。MCTS による木の構築により、良好な燃料装荷パターンを作成するための有望な燃料集合体の候補が得られるためである。

第 5 章では、少ない炉心特性計算回数で良好な燃料装荷パターンが得られるという MCTS の利点を活用し、深層学習を用いることによる性能のさらなる向上を目的とした。ニューラルネットワーク（Neural Network, NN）と MCTS を組み合わせた手法（N-MCTS）は、MCTS のシミュレーションに NN を導入することによって、有望な燃料集合体の装荷の実現を試みる。NN で使用する学習データは、全く燃料が装荷されていない燃料装荷パターンから SA によって最適化された燃料装荷パターンまでの経路であり、最適化前に NN の学習が行われる。N-MCTS、MCTS、SA を用いて 100 種類の燃料インベントリの燃料装荷パターン最適化を行った。N-MCTS の汎化性能および最適化性能を評価するために、最適化では学習に用いたものと異なる 100 種類の燃料インベントリを最適化の対象とした。N-MCTS は MCTS より 100 試行中 80 試行、SA より 100 試行中 88 試行で小さい径方向ピーキング係数が得られ、NN を用いた有望な燃料集合体の装荷によって、N-MCTS は高い汎化性能および最適化性能を持つことが示された。以上より、MCTS と深層学習を組み合わせた手法である N-MCTS を用いることで、少ない炉心計算回数で従来手法よりも良好な燃料装荷パターンを

発見することが可能であると結論付けた。

## 6.2　今後の課題

本節では、MCTS と深層学習を組み合わせた手法について、今後の課題を述べる。

### 径方向ピーキング係数以外の最適化指標の考慮

本研究では、MCTS の性能を評価するために、サイクルを通じた径方向ピーキングの最小化を行った。径方向ピーキングは外側の燃料集合体の影響を受けやすいため、外側から装荷する手順の MCTS で良い性能が得られたが、径方向ピーキング係数以外の最適化指標も考慮した最適化を行う場合には、外側の燃料集合体の影響を受けやすいとは限らない。そこで、安全性の指標として減速材温度係数、集合体最高燃焼度など、経済性の指標として取出平均燃焼度などを考慮した目的関数と MCTS の最適化における装荷手順の設定を検討する必要がある。

### 詳細な炉心特性計算による最適化

本研究の計算条件は、実際の燃料装荷パターン最適化の問題設定を大幅に簡略化している。燃焼燃料の種類を詳細に分類した条件での NN の汎化性能の確認や計算メッシュ数を増やした高精度な炉心計算を行うことが、実機炉心の最適化を行う上で必須である。

### 大規模な学習データを用いた汎化性能及び最適化性能の向上

NN の学習データとして 100 個の良好な燃料装荷パターンまでの経路を用いた。学習データとして用いる燃料装荷パターンの個数や燃料インベントリの種類を増やすことで、N-MCTS の性能向上が可能であると考えられる。

# Appendix A　　最適化パラメータの感度解析

本付録では、MCTS、SA、N-MCTS の最適化パラメータの感度解析を行う。計算条件は 4.3 節と同様である。

## A.1　MCTS のパラメータ感度解析

MCTS の最適化パラメータは、3.5.2 項で紹介した Upper Confidence Bounds applied to trees（UCT）の$C_p$である。UCT を式(A.1)に示す。

$$c^* = \underset{c \in \text{children of } s}{\arg\max} \left( Q(c) + C_p \sqrt{\frac{\ln N(s)}{N(c)}} \right) \tag{A.1}$$

$C_p$は探索と活用のバランスを決める正の定数である。期待値が大きい場合、第 1 項が大きくなり、頂点$c$の探索回数が少ない場合、第 2 項が大きくなる。$C_p$が大きな正の値の場合、探索回数が少ない子が選択されるようになる（探索）。一方、$C_p$が小さな正の値の場合、期待値が大きい子が選択される（活用）。炉心特性計算回数が 1000 回と 10000 回の場合に分けて、実際に$C_p$を変化させながら最適化を行うことで、パラメータサーベイを行う。

### A.1.1　炉心特性計算回数が 1000 回の場合

炉心の外側から装荷する手順の MCTS（MCTS-P）の$C_p$と 100 試行で得られる最小の径方向ピーキング係数の平均の関係を図 A-1 に示す。MCTS-P では、$C_p = 0.1$で得られる最小のピーキング係数の平均が最小となった。

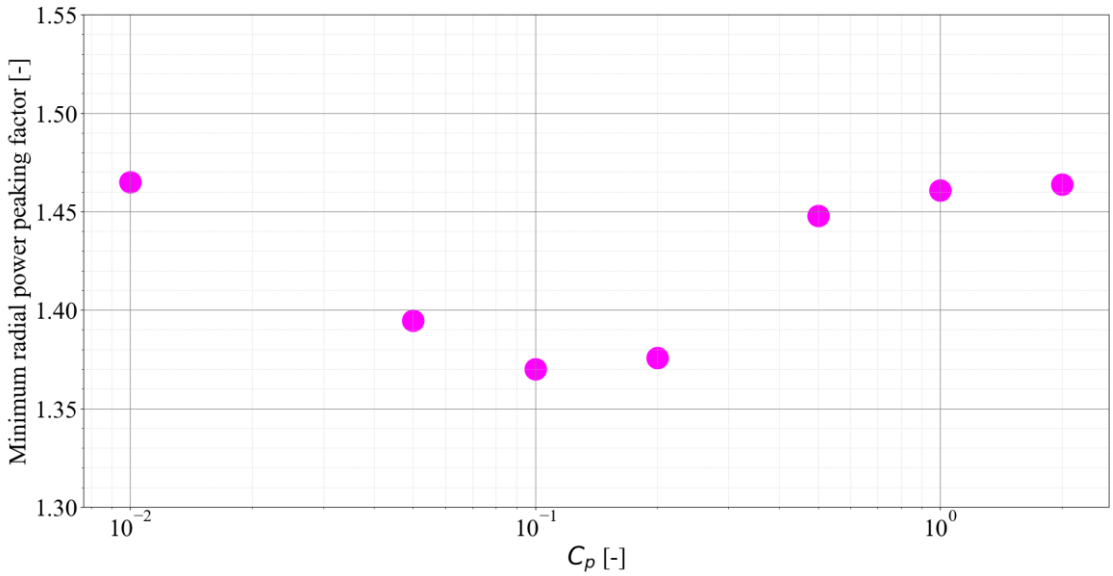



図 A-1　MCTS-P の$C_p$と 100 試行で得られる最小ピーキング係数の平均

また、炉心の内側から装荷する手順の MCTS（MCTS-C）の$C_p$と最適化結果の関係を図 A-2 に示す。MCTS-C では、$C_p = 0.1$で得られる最小のピーキング係数の平均が最小となった。MCTS-C では$C_p$を 0.1 から大きくした場合でも、最適化性能に大きな影響が見られなかった。これは、$C_p = 0.1$以上ではデータを活用した探索ができず、ランダムな探索になることを示している。

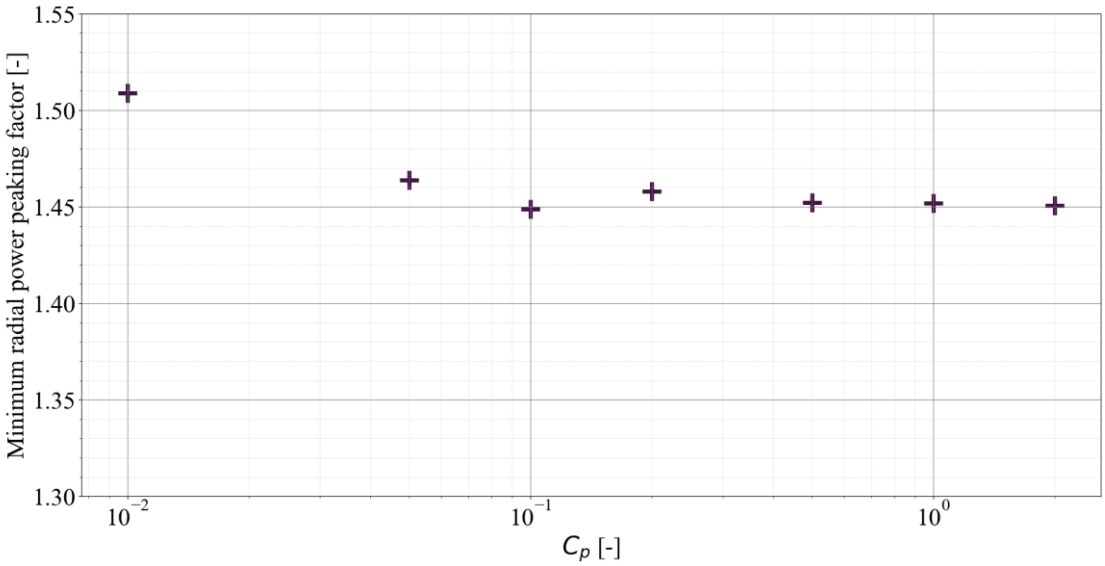



図 A-2　MCTS-C の$C_p$と 100 試行で得られる最小ピーキング係数の平均

炉心特性計算回数が 1000 回の場合、MCTS-P と MCTS-C ともに$C_p = 0.1$で最も良い最適化性能が得られた。

### A.1.2　炉心特性計算回数が 10000 回の場合

MCTS-P の$C_p$と 100 試行で得られる最小の径方向ピーキング係数の平均の関係を図 A-3 に示す。$C_p = 0.5$で最も良い最適化性能が得られた。1000 回では、MCTS-P と MCTS-C の最適な$C_p$が同じ値であるため、10000 回の MCTS-C でも$C_p = 0.5$で最も良い最適化性能が得られると仮定した。

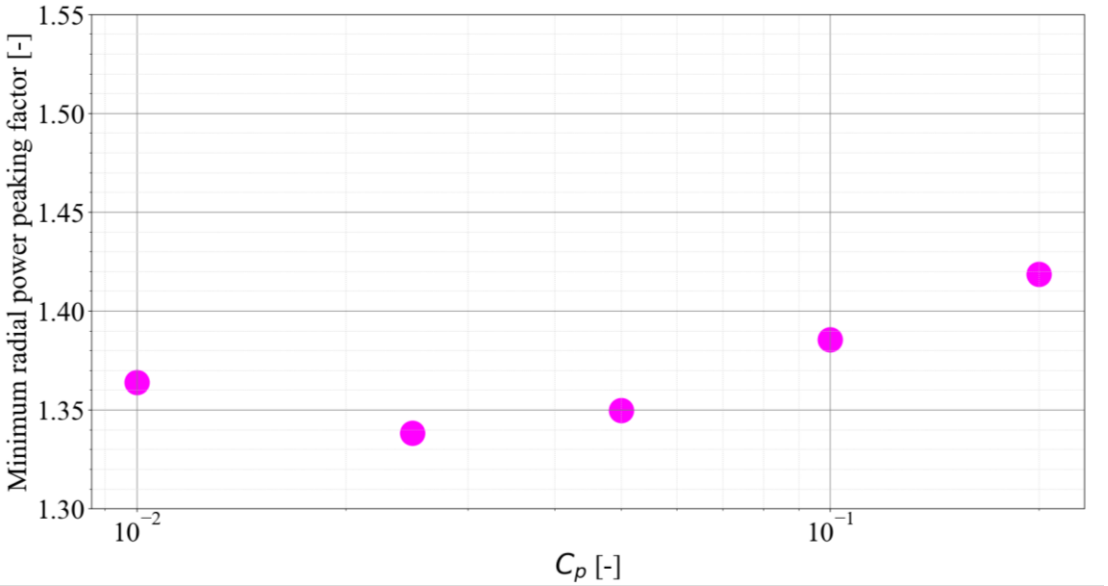



図 A-3　MCTS-P の$C_p$と 100 試行で得られる最小の径方向ピーキング係数の平均

## A.2　SA のパラメータ感度解析

文献[6]に従って、SA のパラメータサーベイを行う。

SA の最適化パラメータは、次の 4 つである。

1. 初期温度（Initial Temperature）
2. 各温度での反復回数と温度の減少関数
3. 最終温度（Final Temperature）

炉心特性計算回数が 1000 回と 10000 回の場合に分けて、それぞれのパラメータサーベイを行う。

### A.2.1　炉心特性計算回数が 1000 回の場合

#### 1.　初期温度

通常は、式(A.2)で表すことができる受理率が 0.8 前後の値になるように初期温度を設定する。

$$\eta(T) = \frac{\text{提案された解に移動した回数}}{\text{提案された回数（}T\text{での反復回数）}} \tag{A.2}$$

そこで、初期温度$T_0$での受理率を以下の手順で求める。

A) 4.3 節と同様の燃料インベントリを用いてランダムに燃料装荷パターン$x_0$を作成

B) 同じ対称性を持つ 2 箇所の燃料集合体をランダムに交換し、近傍の燃料装荷パターン$x_1$を作成

C) 2.3.1 項で説明した式(A.3)に基づいて受理する確率$P(x_0, x_1, T_0)$を求め、受理するか決定

$$P(x_0, x_1, T_0) = \begin{cases} 1 & f(x_1) \leq f(x_0) \\ \exp\left(-\dfrac{f(x_1) - f(x_0)}{T_0}\right) & f(x_1) > f(x_0) \end{cases} \quad \text{(A.3)}$$

ここで$f(x_i)$は燃料装荷パターン$x_i$の径方向ピーキング係数である。$x_0$の径方向ピーキング係数に比べて、$x_1$の径方向ピーキング係数が小さい場合には必ず受理される。$x_0$の径方向ピーキング係数に比べて、$x_1$の径方向ピーキング係数が大きい場合には式(A.3)に基づいて受理するか決定する。

D) A)–C)を 10000 回繰り返すことで受理率を算出

以上の手順を 0.05 から 0.2 までの 0.05 刻みの初期温度で行った結果を図 A-4 に示す。受理率が最も 0.8 に近くなる温度は 0.10 であるため、初期温度$T_0 = 0.10$とする。

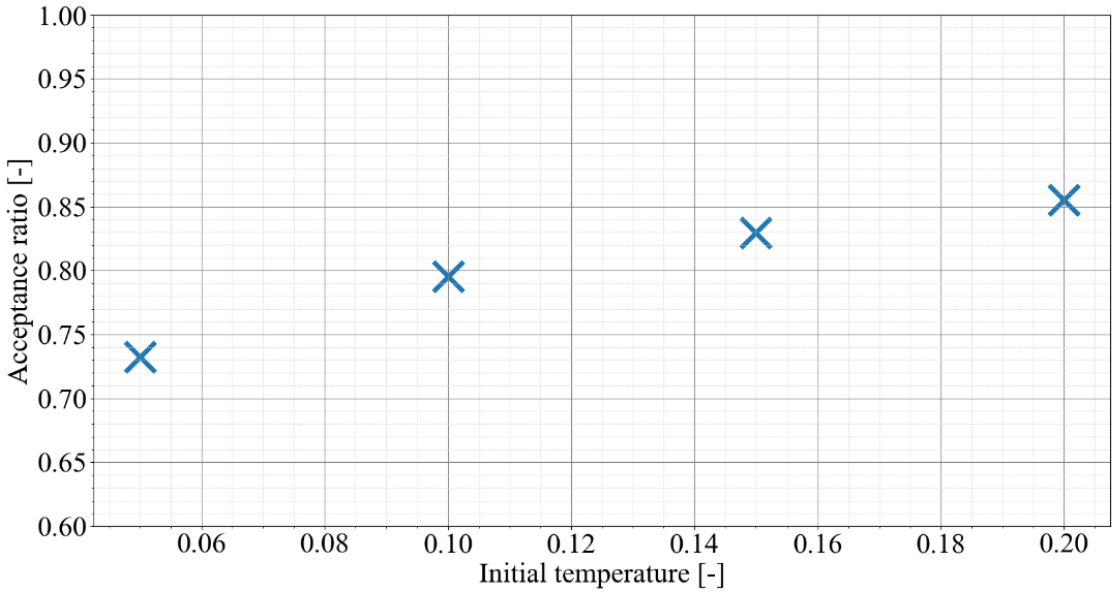



図 A-4 初期温度と受理率の関係

**2. 各温度での反復回数と温度の減少関数**

温度の減少関数は式(A.4)が良く用いられているため、今回は式(A.4)を採用する。1 回の解の生成を行うごとに、温度の冷却を行う。

$$T_{k+1} = \alpha T_k \quad \text{(A.4)}$$

$k$は解の生成回数、$\alpha(> 0)$は冷却率であり、$\alpha$が大きいほど冷却が遅くなる。

### 3. 最終温度

最終温度を 0.0001, 0.001, 0.01, 0.075, 0.1, 0.15, 0.2 の 7 通り変化させ、それぞれ 100 回初期乱数を変更し、径方向ピーキングの最小化を行った結果を図 A-5 に示す。最終温度が 0.1 の場合が最も良く、冷却を行うことで最適化性能が悪くなることが分かった。これは 1000 回という少ない回数で温度冷却を行うと、急激な温度の変化となり、SA で想定している広域探索から局所探査に徐々に移行するというプロセスを再現出来ないためであると考えられる。

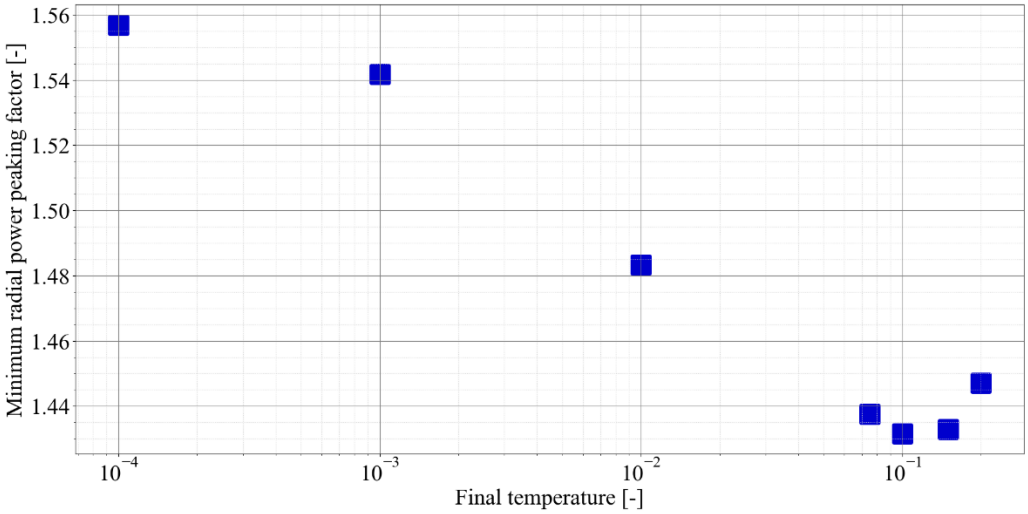



図 A-5　最終温度と最小のピーキング係数の平均

### 4. 初期温度の設定（2 度目）

1000 回という少ない回数では、温度冷却を行わないほうが良いことが分かった。したがって、初期温度によって最適化結果がどのように変化するかを調べる。初期温度を 0.05, 0.1, 0.15 の 3 通り変化させ、それぞれ 100 回初期乱数を変更し最適化を行った。結果を図 A-6 に示す。初期温度 0.1 に設定した場合が最も良い結果となった。

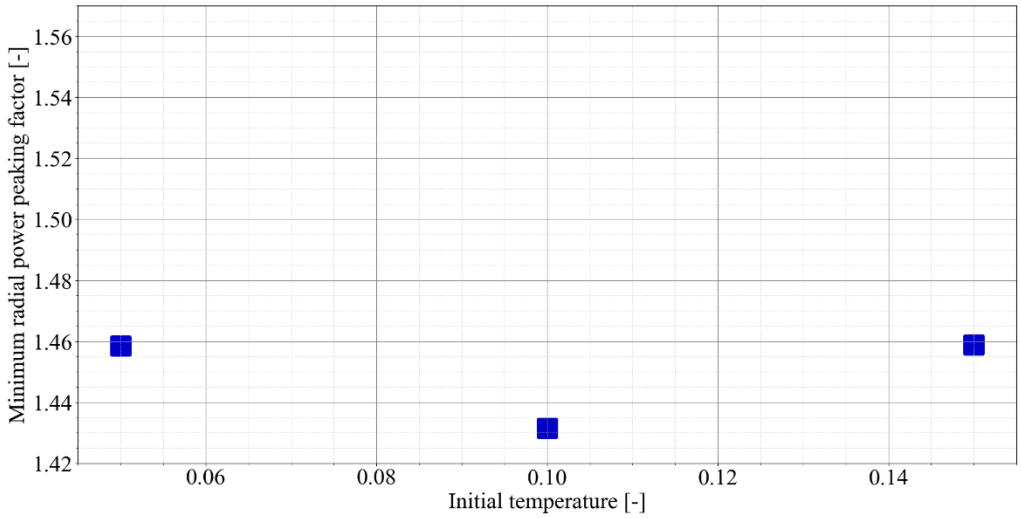



図 A-6　冷却率が 1 の時、初期温度と最小のピーキング係数の平均

以上より、初期温度 0.1、最終温度 0.1、冷却率 1 と設定する。

### A.2.2　炉心特性計算回数が 10000 回の場合

**1.　初期温度**

炉心特性計算回数が 1000 回の場合と同様に、初期温度 0.10 である。

**2.　各温度での反復回数と温度の減少関数**

炉心特性計算回数が 1000 回の場合と同様に、式(A.5)を採用する。

$$T_{k+1} = \alpha T_k \tag{A.5}$$

**3.　最終温度**

最終温度を 0.0001, 0.001, 0.0025, 0.005, 0.01, 0.1 の 6 通り変化させ、それぞれ 100 回初期乱数を変更し、径方向ピーキングの最小化を行った結果を図 A-7 に示す。最終温度が 0.0025 の場合が最も良くなった。

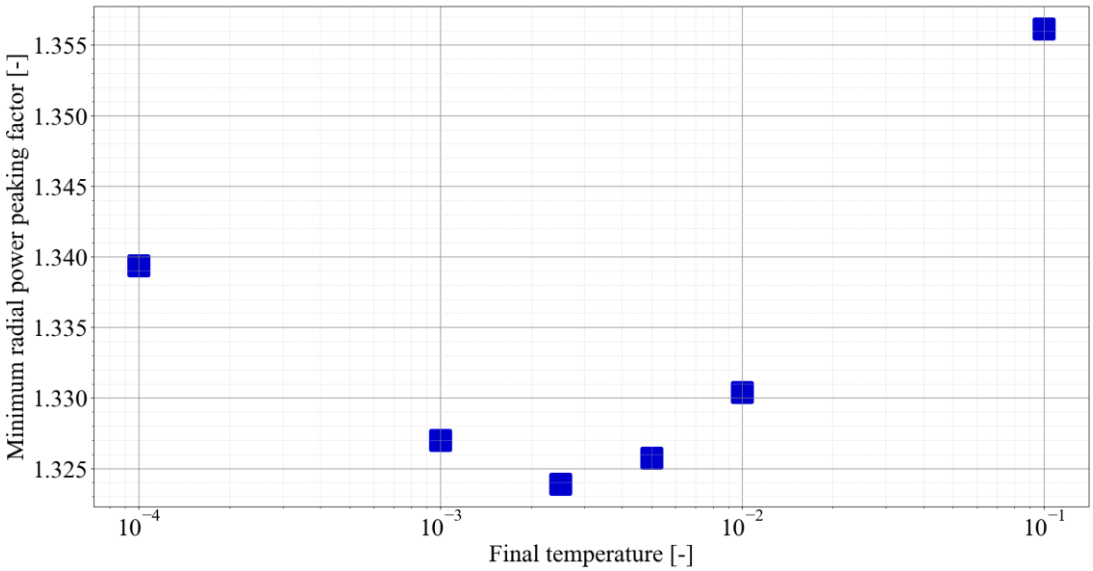



図 A-7　最終温度と最小のピーキング係数の平均

以上より、初期温度 0.1、最終温度 0.0025 と設定した。10000 回の炉心特性計算回数で 0.1 から 0.0025 まで冷却を行うため、冷却率は$\left(\frac{0.0025}{0.1}\right)^{1/10000} = 0.025^{1/10000}$である。

## A.3　参考文献

[1]　相吉英太郎, 安田恵一郎, *メタヒューリスティクスと応用*, 東京, オーム社, 2007.