コンピュータ囲碁

モンテカルロ法の理論と実践

松原 仁 編
美添一樹・山下宏 著
2012 共立出版

理論談話会2019 #4
M1 出原昇馬
2019/5/10

目次

目的：2000年代後半にかけてコンピュータ囲碁の大幅な強化をもたらした
モンテカルロ木探索のアルゴリズムについて理解する.

はじめに

都市・交通研究におけるモンテカルロ法

本文

はじめに(1)

都市・交通研究におけるランダムシミュレーション

モンテカルロ法：**乱数**を用いてシミュレーションを行う手法

■ シミュレーション法による経路選択肢列挙

- **モンテカルロ法によって確率的に経路を抽出**し，経路列挙を行う．
- 各リンクのコストが独立な確率分布に従うという条件の下で，最小コストとなる経路抽出を繰り返す．正規分布を仮定→プロビットモデル

■ パラメータ推定における乱数の利用

構造化プロビットモデルにおけるパラメータ推定（屋井ら(1998)）

経路選択効用（確定項＋誤差項）

$$U = V + \varepsilon$$

経路選択肢rの選択確率

$$P_r = \int_{\varepsilon_1=-\infty}^{\varepsilon_r+V_r-V_1} \cdots \int_{\varepsilon_r=-\infty}^{\infty} \cdots \int_{\varepsilon_R=-\infty}^{\varepsilon_r+V_r-V_R} \phi(\varepsilon) d\varepsilon_R \cdots d\varepsilon_1$$

密度関数（Σ：分散共分散行列（経路間の相関を考慮））

$$\phi(\varepsilon) = (2\pi)^{\frac{-R}{2}} |\Sigma|^{\frac{-1}{2}} \exp\left(\frac{-1}{2}\varepsilon\Sigma^{-1}\varepsilon^T\right)$$

選択確率の導出のために多重積分が必要：計算が困難！

経路選択肢rの選択確率（条件付確率として変形）（R=3の場合）

$$P_{1\zeta} = \Phi(a_1)\Phi(a_2(\zeta_1))$$

Φ：標準正規分布関数

確率変数（標準正規分布に従う）

$$\zeta_1 = \Phi^{-1}\left[u_1 \Phi(a_1)\right]$$

これを乱数により与える（複数回実行）

多重積分をせずにパラメータ推定が行える！

BinN：ネットワーク行動学 http://bin.t.u-tokyo.ac.jp/kaken/

屋井・中川・石塚：シミュレーション法における構造化プロビットモデルの推定特性．土木学会論文集，No.604(IV-41), pp.11-21, 1998.

はじめに(2)

■ 多目的最適化とネットワークデザイン

大山・羽藤：多目的最適化に基づく歩行者の活動ネットワークデザイン. 都市計画論文集, Vol.52-3, pp.810-817

- 歩行者活動量と歩道拡幅面積の2つの目的関数の下でネットワークデザイン問題（多段階最適化問題）を解く.
- 最適解は一意ではなく，パレート解集合.
- 解の更新の近傍探索アルゴリズムの一部において，歩道リンクを**ランダムに選び**幅員を変化させ，採択条件に従いパレート解を求める.

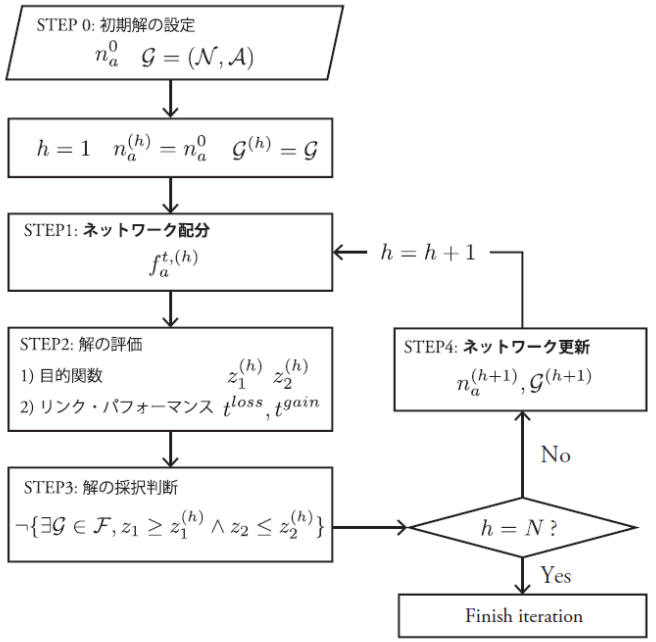


図 3 ネットワーク更新法のフロー

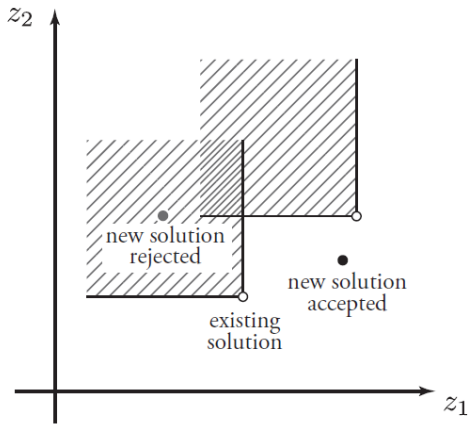


図 2 解の採択条件

第2章
背景；モンテカルロ木探索以前

ゲーム木の構築

ゲーム木

木：**節点**（node）と**枝**（edge）をもつデータ構造

ゲーム木：初期局面から終端局面まで，**ゲーム全体の推移を一つの木**で表したもの

二人ゼロ和完全確定情報ゲームとminimax探索

囲碁や将棋は以下の条件を満たす．

- **二人**のプレイヤー
- 互いに隠された情報が全くない（**完全情報**）
- 運（確率）が絡む要素がない（**確定情報**）
- 片方のプレイヤーが勝てばもう片方が負ける（**ゼロ和**）

二人ゼロ和完全確定情報ゲームであれば**理論上**，最善手をいつでも求めることができる（＝**minimax探索**）

各プレイヤーの行動
先手（Max player）：スコアを最大化
後手（Min player）：スコアを最小化

※ゲームの状況に応じて節点ごとにスコア（**ゲーム値**）が割り当てられる．
先手番が勝利→スコア大，後手番が勝利→スコア小

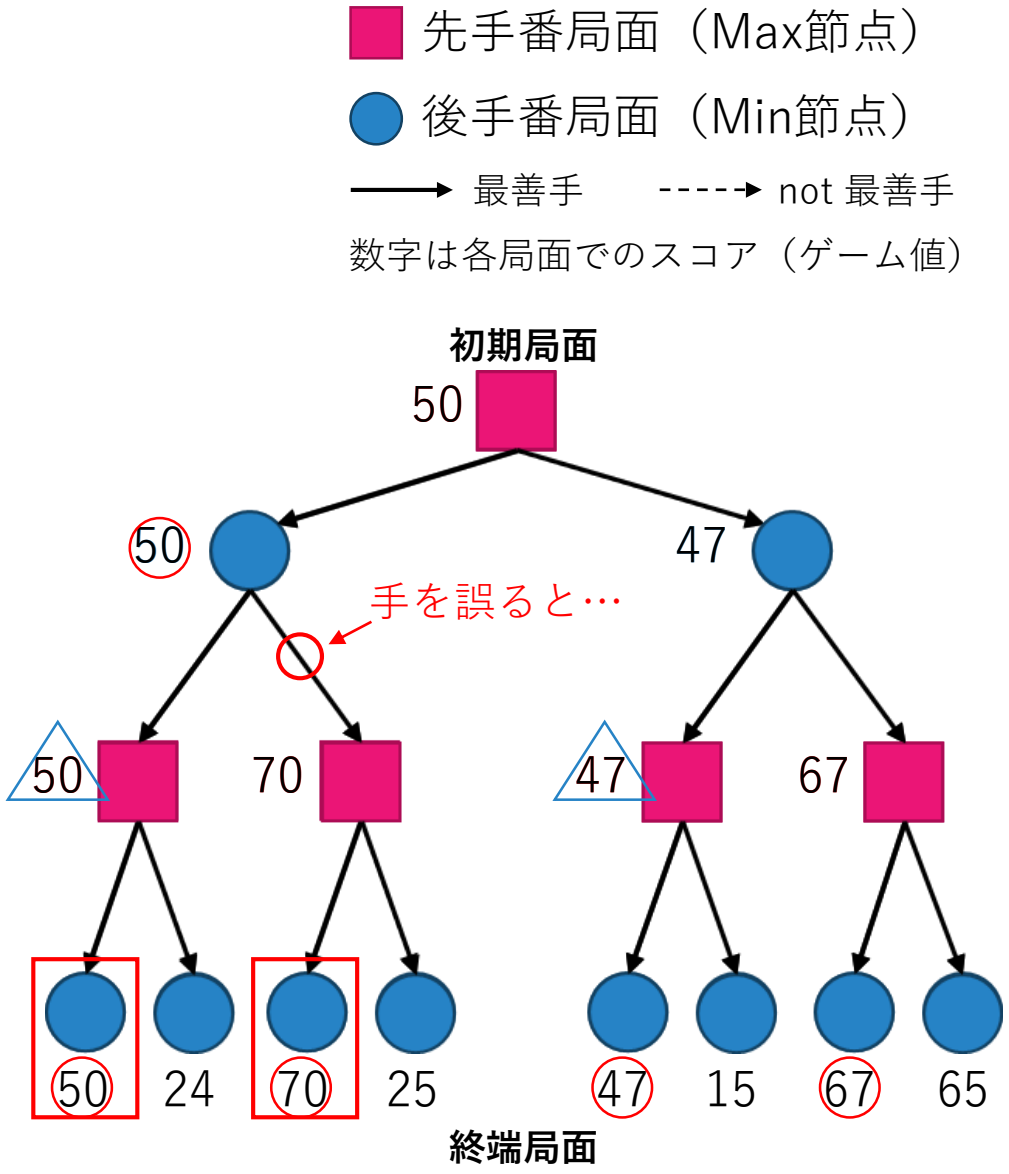
minimax探索(1)

終端局面から遡り初期局面までのスコアを求めることにより，最善手を選んだ場合のスコアを知ることが可能

この例のような木
：**minimax木**

最大値と最小値を交互に選択して節点のスコアを求め，最善手を求めること
：**minimax探索**

ただし，
一般的に普及しているゲームでは**ゲーム木全体の節点数が膨大**であり，minimax探索は容易ではない．
&
スコアが正しいことが前提



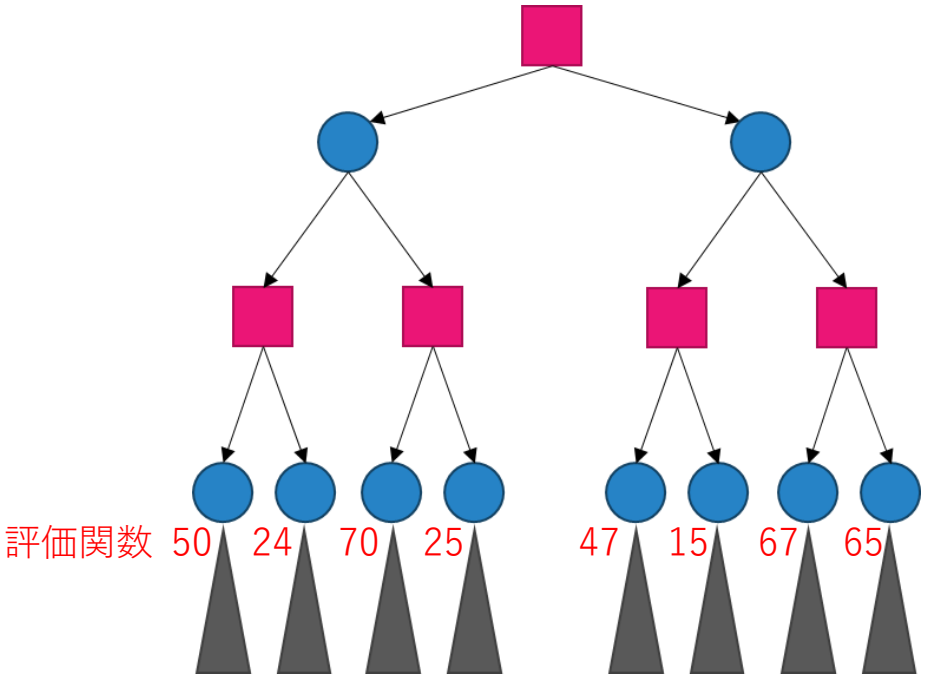
評価関数の利用

探索をある深さで打ち切り，その局面でのゲーム値の推測値（＝**評価関数**）を用いて着手を決める．

- **正確**であること
- **素早く**計算できること

が評価関数に求められる．

評価関数の作成には，オセロ，チェス，将棋などではヒューリスティックな手法から機械学習を用いたものまで，様々な手法がある（らしい）．



alpha-beta探索

minimax探索→最善手の発見にはN個すべての節点の探索が必要

alpha-beta探索→最善の場合，√N個の節点を探索するだけで最善手を発見可能

■ 先手番局面（Max節点）

● 後手番局面（Min節点）

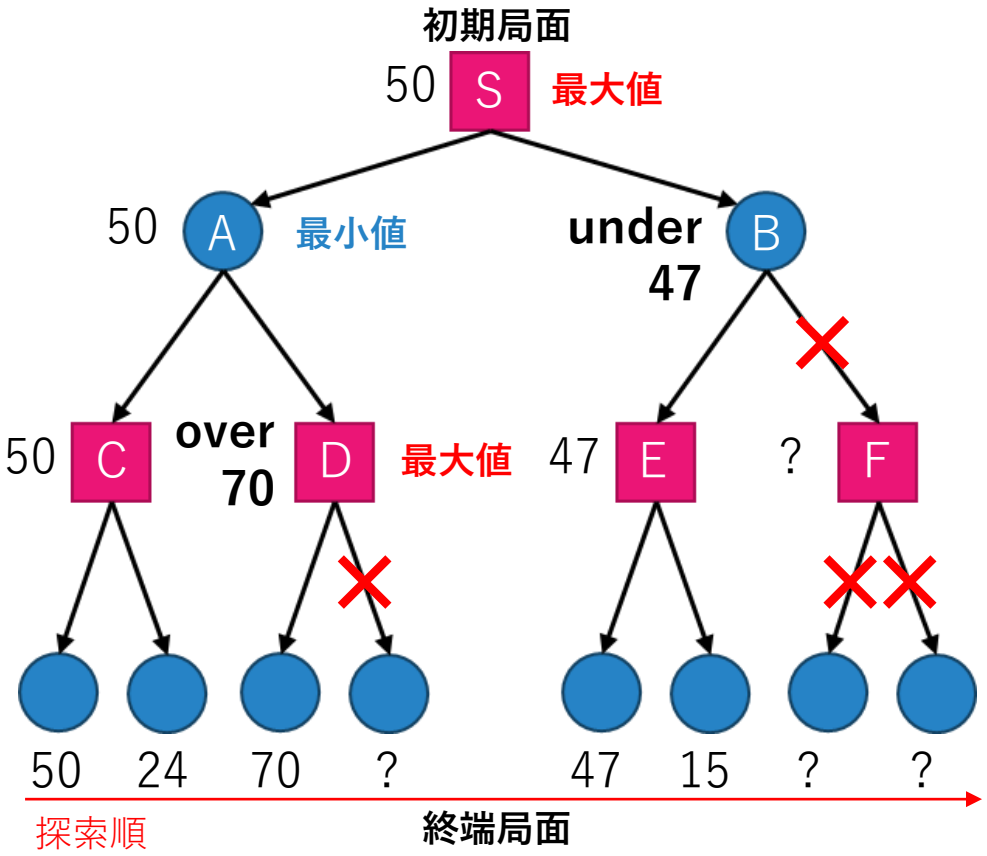
——→ 最善手　　- - - -→ not 最善手

数字は各局面でのスコア（ゲーム値）

左から順にスコアを計算する

最小値を求めるAでは「50」なのでDでは「70」が見つかった時点で残りの節点の探索は行わない．

最大値を求めるSでは「50」なのでBは47が見つかった時点で探索を行わない

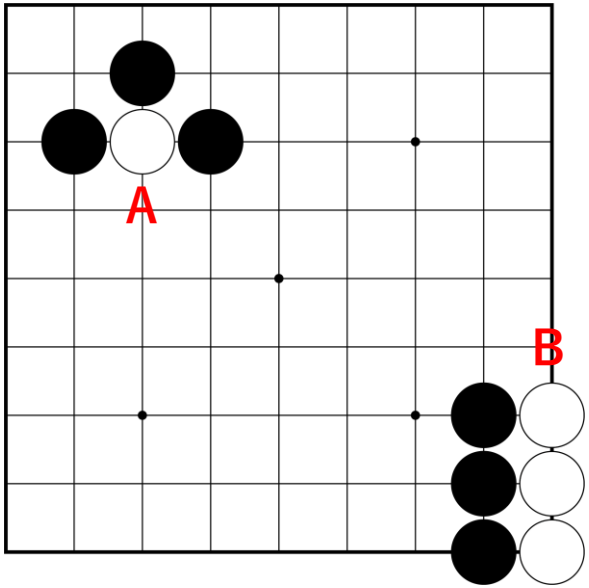


→ほとんどの二人ゼロ和完全確定情報ゲームで有効

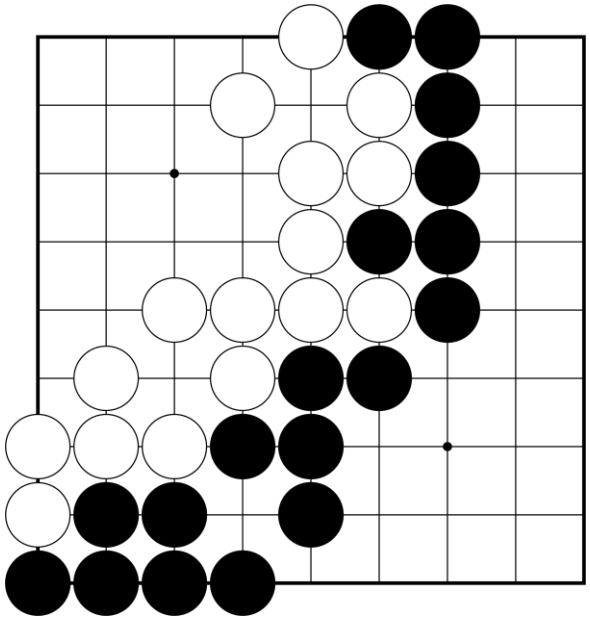
補足：囲碁について

チェス，将棋はじめほとんどの二人ゼロ和完全確定情報ゲームでは，
優秀な評価関数＋alpha-beta探索によりコンピュータが優れた実績を有する．
ただし，囲碁のみ際立って人間が優勢←minimax探索（alpha-beta探索）が適さず

- **探索空間の大きさ**←合法手の数，終局までの平均手数が多い
- **評価関数の難しさ**←石の価値が平等．局所的最善手≠全局的最善手（捨石）



黒番AやBに石を置くと囲われた白石を取り除ける



終局．囲われた領域の数（目）で勝敗がつく（この場合黒7目勝）

ゲーム	局面数
3目並べ	10^3
オセロ	10^28
チェス	10^50
将棋	10^69
囲碁（9路盤）	10^38
囲碁（19路盤）	10^170

→minimax探索（alpha-beta探索）に代わり**モンテカルロ木探索が台頭**

第3章
モンテカルロ囲碁の概要

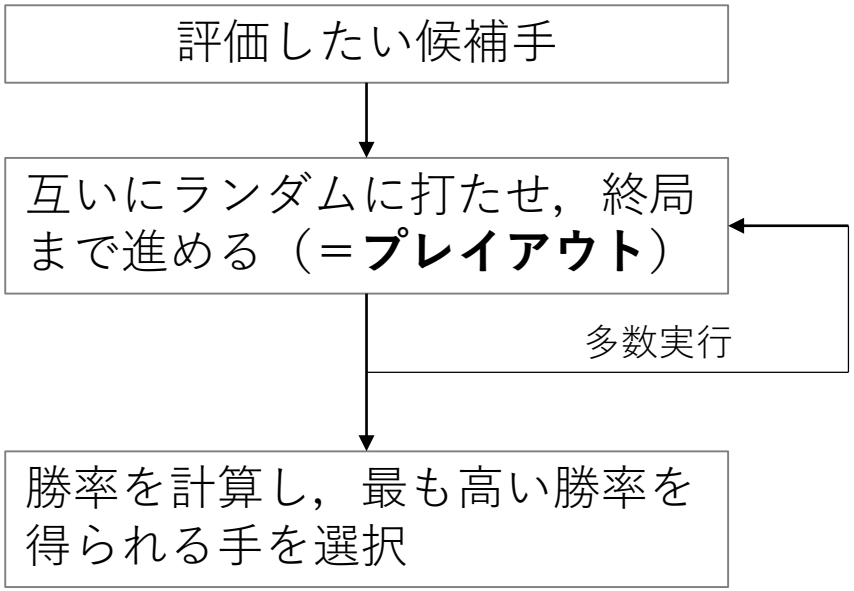
原始モンテカルロ囲碁

モンテカルロ法の囲碁への適用

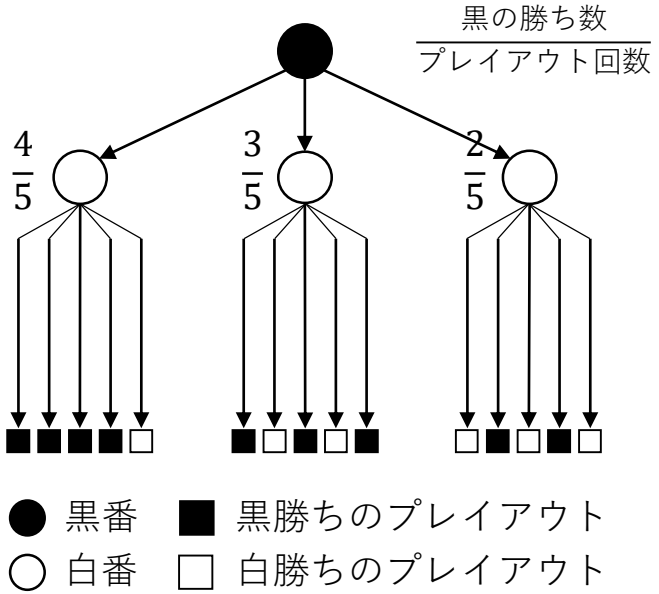
モンテカルロ法：**乱数**を用いてシミュレーションを行う手法

囲碁への適用：
黒と白のプレイヤーにランダムに終局まで打ち切らせ，何度もそれを繰り返すことにより盤面の評価を行う（原始モンテカルロ囲碁）．
Brugmann(1993), Bouzy(2003), Cazenave(2005)など

流れ



※スコア（目数）ではなく勝敗結果を用いる



問題点：相手のミスを期待した手を選択することが多く存在（例：シチョウを逃げる手）

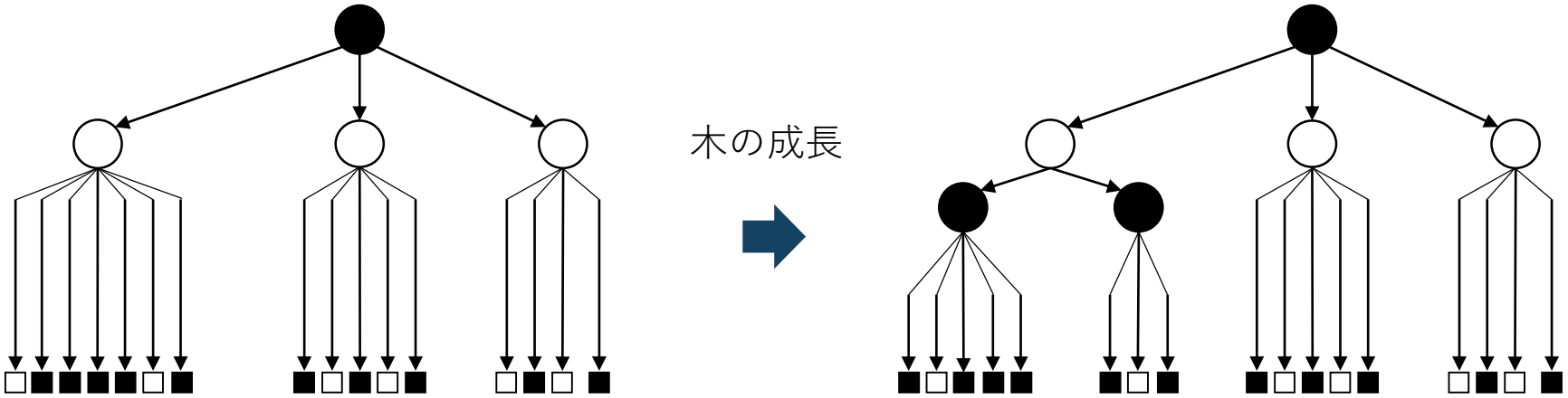
モンテカルロ木探索(1)

原始モンテカルロ法＋木探索による囲碁プログラムの強化

モンテカルロ木探索（Monte Carlo Tree Search: MCTS）
：有望な手にプレイアウトを集中させ，さらに有望な手についてはゲーム木を展開して深く読む

1. 原始モンテカルロ法同様各候補手でプレイアウトを行い，スコアを集計．「有望な手」に多くのプレイアウトを割り当て
2. 各節点でプレイアウトが行われた回数を記録．回数がある閾値を超えた場合，その手を展開し，プレイアウトを開始する節点を一つ深くする

→相手のミスに期待した手を打つという弱点が解消

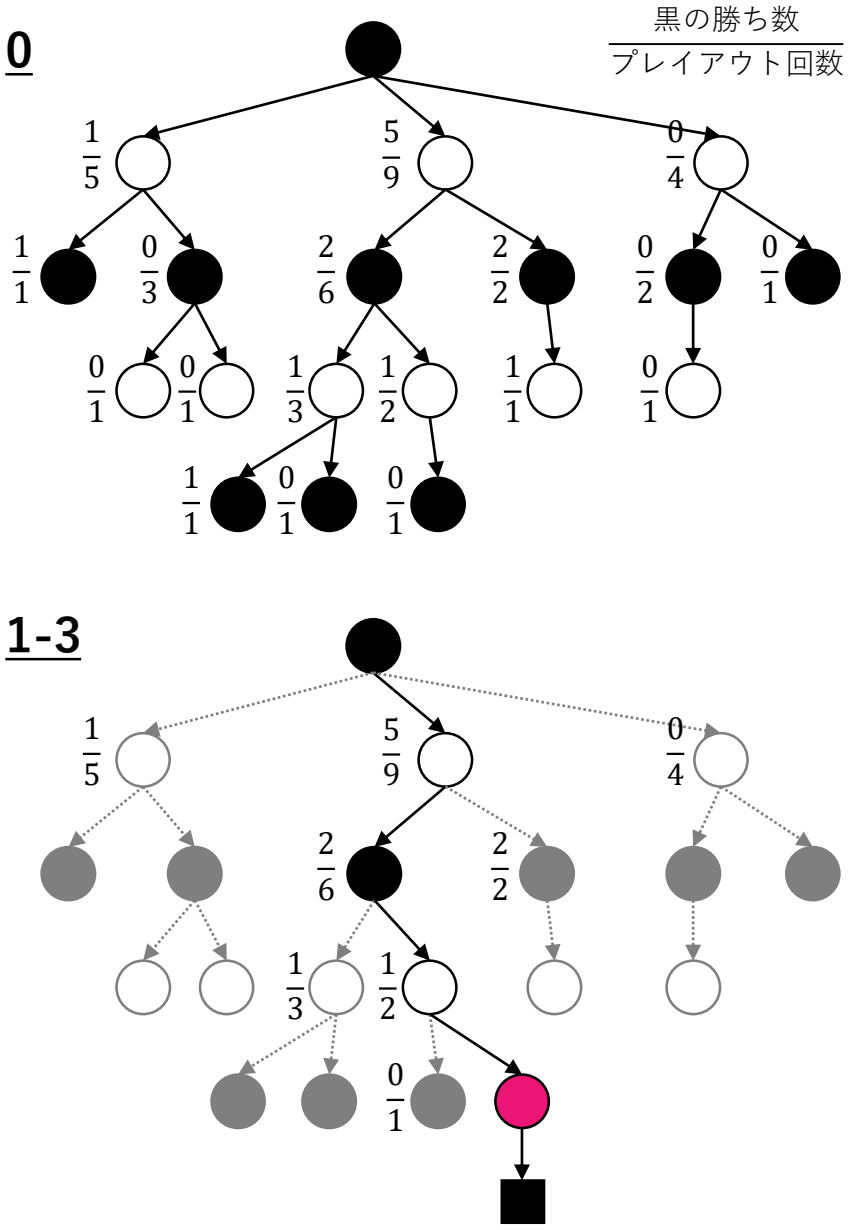


モンテカルロ木探索(2)

流れ（閾値：1に設定）

1. 根節点から「有望な子節点」を選択しながら末端まで枝をたどる.
2. 末端の節点のプレイアウト回数が閾値を超えていれば，その節点を展開し，さらに一段木を下りる（例では展開なし）.
3. 末端の節点で1回プレイアウトを行う.
4. プレイアウトの結果によってたどった経路上の節点を更新する.
5. 制限時間や総プレイアウト数の制限に達していれば終了する．そうでなければ1に戻る.

※子節点の合計が親節点の勝敗の合計と異なるのは子節点を展開する前に親節点から行われたプレイアウトの勝敗が1回分含まれているため



有望な子節点をたどり，末端で1回プレイアウト

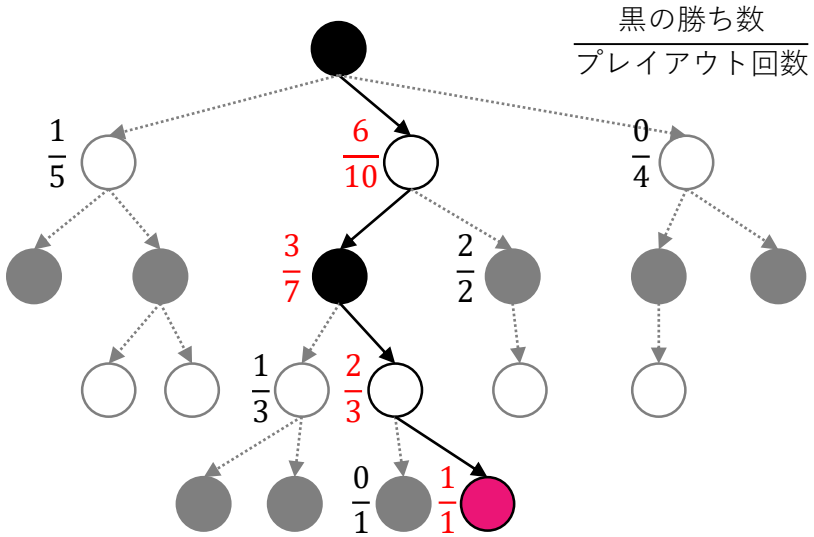
モンテカルロ木探索(3)

流れ（閾値：1に設定）

1. 根節点から「有望な子節点」を選択しながら末端まで枝をたどる.
2. 末端の節点のプレイアウト回数が閾値を超えていれば，その節点を展開し，さらに一段木を下りる（例では展開なし）.
3. 末端の節点で1回プレイアウトを行う.
4. プレイアウトの結果によってたどった経路上の節点を更新する.
5. 制限時間や総プレイアウト数の制限に達していれば終了する．そうでなければ1に戻る.

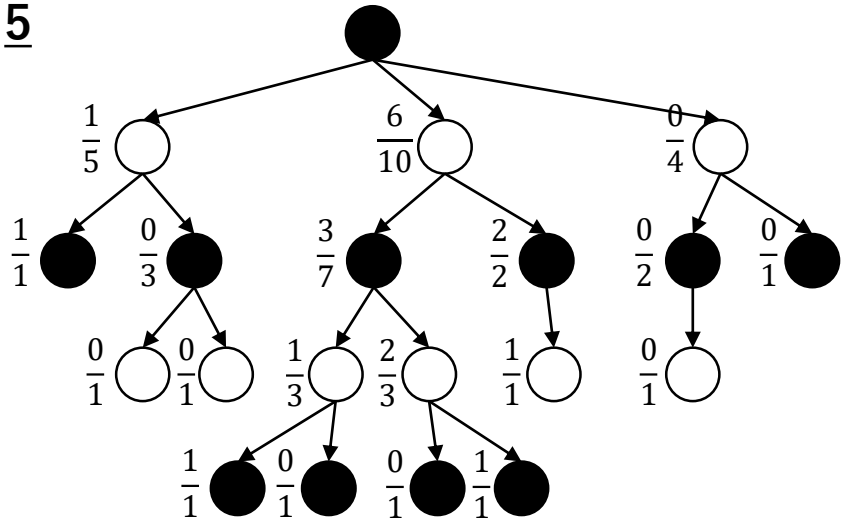
※子節点の合計が親節点の勝敗の合計と異なるのは子節点を展開する前に親節点から行われたプレイアウトの勝敗が1回分含まれているため

4



経路上の節点の勝敗を更新

5



更新後の木

第4章
モンテカルロ木探索の詳細

Multi-Armed Bandit 問題(1)

確率分布の比較

モンテカルロ木探索において「有望な子節点」はどう選ぶべきか？
→各節点はランダムサンプリングの結果得られた確率分布を持つため，**確率分布同士を比較**し，有望な確率分布を知る必要がある．

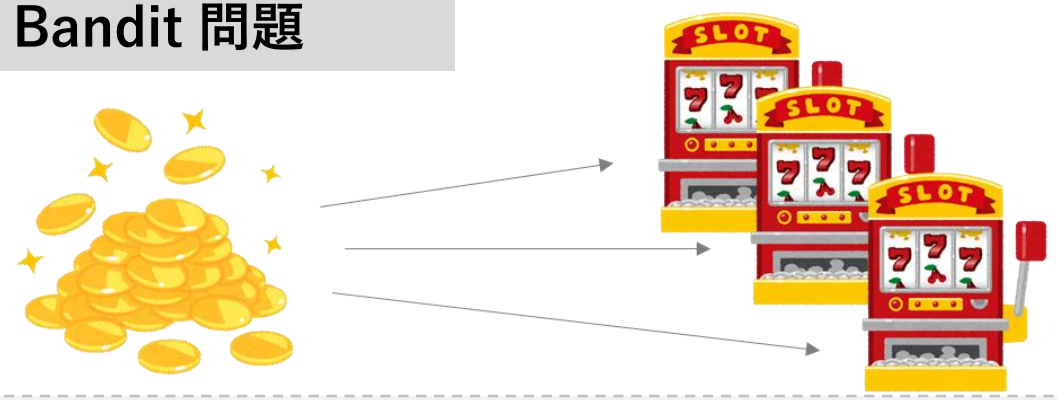
勝率の一番高い手を選ぶ？
→**たまたま**最初の1回プレイアウトが勝ちで終わった手がその後もずっと選ばれてしまう

勝率が最大でない手をある程度試す（**ε -greedy**）
：**確率 ε** で勝率が最大でない手のどれかを選び，**確率1- ε** で勝率最大の手を選ぶ

より良い手法として**Multi-Armed Bandit 問題**の解法がある．

Multi-Armed Bandit 問題(2)

Multi-Armed Bandit 問題



- 手元にはある枚数のコインがある
- 複数のスロットマシンがあり，それぞれの報酬は**未知の確率分布**に従う

この状況下で，スロットマシンから得られる報酬の期待値を最も高くするためにはどのようにしたら良いか？

一部のコインをスロットマシンに投入 → 報酬を観測 → 次にコインを投入するマシンを選択

regret

：理想的な場合（報酬の期待値が最も高いマシンにすべてのコインを投入した時）の期待値－ある戦略に従った際に得られる報酬の期待値

の最小化問題

Multi-Armed Bandit 問題(3)

UCB1アルゴリズム (Auer(2002))

UCB: Upper Confidence Bound（信頼上限）

1. すべてのスロットマシンに1枚ずつコインを投入する.
2. 各スロットマシンについて次の**UCB1値**を計算し，最大の値を持つスロットマシンにコインを投入する.

$$\bar{x}_j + \sqrt{\frac{2\log n}{n_j}}$$

- 期待値の高いところにより多くのコインを投入する
- コインが少ない場合は，単に運が悪いという可能性があるのでその分を考慮し優遇

$\bar{x}_j$:その時点でのj番目のスロットマシンの報酬の平均
n_j:そのスロットマシンに投入されたコインの数
$n = n_1 + \cdots + n_K$:それまでに投入したコインの合計

3. 制限時間が来るまで（コインが尽きるまで）2を繰り返す.

UCB1アルゴリズムを用いた場合，最善でないスロットマシンに投入されるコインの枚数の期待値は$O(\log n)$に抑えられる

$\frac{\log n}{n} \to 0 (n \to \infty)$なので，**コインの枚数nが大きくなれば最善でないスロットマシンに投入されるコインの割合は0に収束**する.

※実際にはUCB1値は定数Cを用いて$\bar{x}_j + C\sqrt{\frac{2\log n}{n_j}}$と書かれる
（報酬が[0,1]の時はC=1．問題に応じて調整）

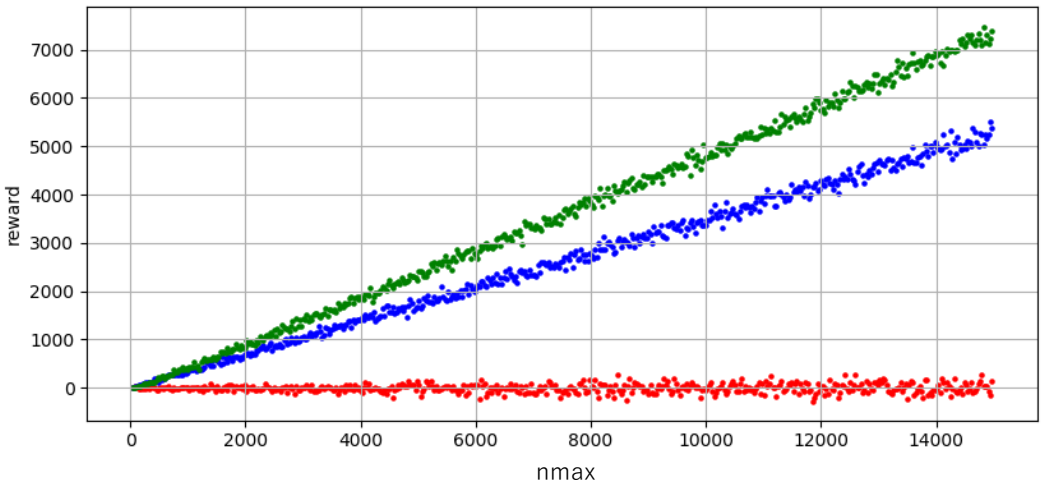
Multi-Armed Bandit 問題(4)

実装してみる

スロット1：平均0,分散1の正規分布
スロット2：平均0.5,分散1の正規分布
スロット3：平均-0.5,分散1の正規分布

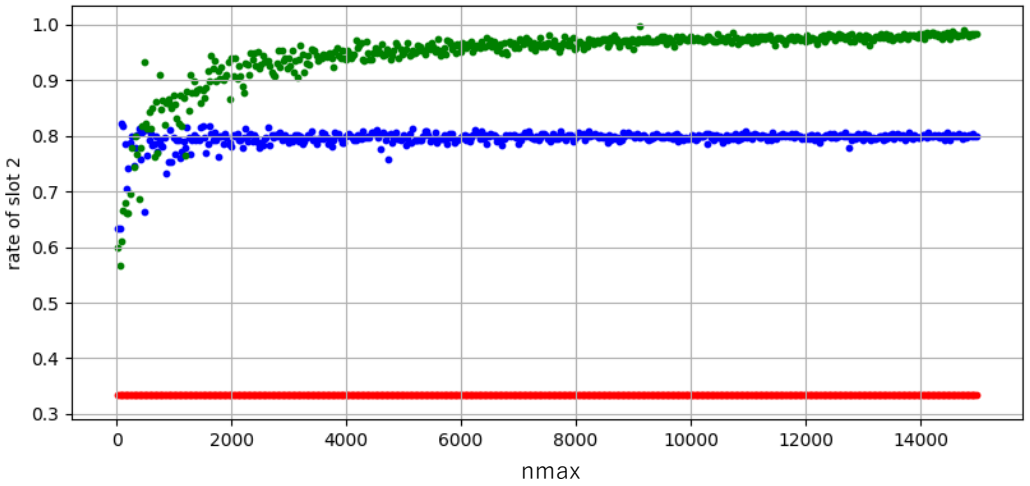
nmax:コインの総数を変えて数値実験

method1（赤）
：それぞれに同じ割合でコインを投入
method2（青）
：ε-greedy(ε=0.2)によりコインを投入
method3（緑）
：UCB1法(C=2)によりコインを投入



コインの総数と報酬

確かに，UCB1法を用いた場合の報酬の期待値が最も高い



コインの総数とスロット2への投入割合

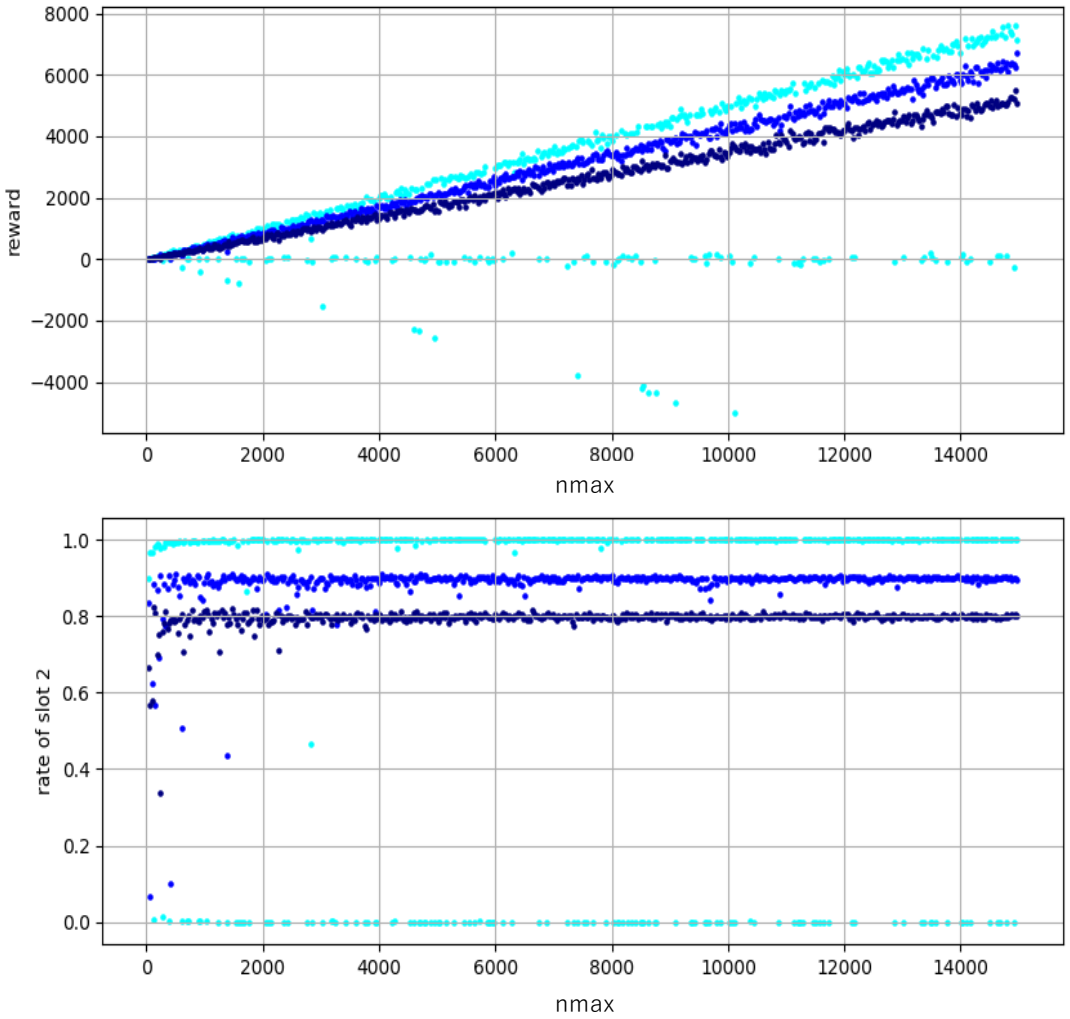
補足：Multi-Armed Bandit 問題(5)

ε -greedyについて

確率 ε で勝率が最大でない手のどれかを選び，**確率1- ε** で勝率最大の手を選ぶ

ε が小さい方がよいか？

ε ＝ 0 （現時点で平均値が高いところにのみ投入）だと，**最初の一回の運に大きく左右されてしまい**，真に期待値が高いところに投入されない恐れが生じてしまう．



囲碁プログラムへの適用

UCTアルゴリズム (Kocsis&Szepesvari(2006))

UCT: UCB applied to Trees

囲碁プログラムにおけるモンテカルロ木探索もMulti-Armed Bandit 問題として捉えることが可能.

- 根節点の子節点（1手先の局面）＝一つのスロットマシン
- プレイアウトの回数＝コインの枚数n
- 勝率orスコア＝報酬

この場合，**一つの子節点から得られる報酬はある確率過程に従う**とみなす（各スロットマシンの報酬の確率分布はそれより下の探索が進むにつれて徐々に変化するため，）.

UCTアルゴリズム：
モンテカルロ木探索における「有望な子節点」を選択する基準としてUCB1値を用いる.

1. 根節点から順にUCB1値の高い子節点を辿る
2. 末端節点でのプレイアウト回数が（適当に決めた）閾値を超えるとその節点を展開.

※2006年，囲碁プログラムMoGoがUCTアルゴリズムを採用し，インターネット囲碁サーバKGSで初段の段位を達成（コンピュータプログラム初）．2008年には9路盤でタラヌ・カタリン五段に**1勝**2敗.
それでもアマチュア有段レベルに達するのが限界.

第5章
モンテカルロ木探索の強化

木探索部分の改良

探索順序の制御

ゲームの知識等によって事前に有望な手を知ることができる場合，すべての手を平等に探索するのは非効率的.
UCB1値を用いると，プレイアウト回数が0の手はbias項が∞になるため，必然的にすべての候補手で最低1回ずつプレイアウトが行われてしまう.

ゲームの知識を用いて合法手を有望な順にソートしておく（距離，パターンなどに基づく）. そのうえで，よさそうな候補手を木に加え，駄目そうな手はプレイアウト回数が0であっても当面はプレイアウトの対象としない.

Progressive Widening

ある節点以下のプレイアウト回数をnとし，その節点以下の合法手の数の上限t_nを以下で与える

$$t_0 = 0, t_{n+1} = t_n + 40 \times 1.4^n$$

プレイアウト回数が大きくなるにつれて，当初は有望でないために無視されていた候補手が探索木に追加される.
→有望な手から探索木が深くなる

プレイアウトの改良

通常のモンテカルロ木探索におけるプレイアウト
：最低限のルールに従うのみで，**完全にランダムな手**が実行される

現実的には限られた時間内に最善手にたどり着くことは（ほぼ）不可能.

（理論的にははっきりとしていないが）様々な改良により効率的に木探索を行うことで，プログラムを強くできる．ただし最善手に収束する保証はない

- パターンの制約
- 罠の回避（ex.シチョウ：少数の手が他の手よりもはるかに良い/悪い）

そのほか，コンピュータの並列化による計算処理速度の向上などが挙げられる.
複数のコンピュータ上で異なる乱数を用いてそれぞれ探索
結果の一部を→定期的に集計
※ただしそれぞれのコンピュータが探索を行っている木の深さは並列化しない場合とあまり変わらない

※2016年にはモンテカルロ木探索（学習時）にディープニューラルネットワークを組み合わせた**AlphaGo**（Google, Deepmind）が登場．樊麾二段に5戦全勝.
https://www.nature.com/articles/nature24270.epdf?author_access_token=VJXbVjaSHxFoctQQ4p2k4tRgN0jAjWel9jnR3ZoTv0PVW4gB86EEpGqTRDtpIz-2rmo8-KG06gqVobU5NSCFeHILHcVFUeMsbvwS-lxjqQGg98faovwjxeTUgZAUMnRQ