TIME MANAGEMENT FOR MONTE-CARLO TREE SEARCH

310551059　陳昱丞
310551069　余忠旻

Reference

- "Time Management for Monte-Carlo Tree Search Applied to the Game of Go" Published in 2010.
  - Authors: Shih-Chieh Huang, Rémi Coulom, Shun-Shii Lin
- "Time Management for Monte-Carlo Tree Search in Go" Published in 2011.
  - Authors: Hendrik Baier and Mark H.M. Winands
  - Provided by Maastricht University.

Introduction

- In the game of Go, there is only limited amount of time to think.
- MCTS has the advantage of stopping at any time.
- How much time should we allocate for each move ?

Types of strategies

1. Static strategies

- Decide about time allocation to all future moves before the start of the game.

2. Semi-dynamic strategies

- Determine the computation time for each move before the start of the respective move search.

3. Dynamic strategies

- Make “live” timing decisions while the search process is running.

STATIC STRATEGIES

STATIC STRATEGIES: FIXED PLAYOUT

- Idea: assign fixed thinking time to every move.
  - The time for every move is the same.
- Ex: for every move using 100 simulation in MCTS.

STATIC STRATEGIES: BASIC FORMULA

$$\mathbf{ThinkingTime} = \frac{\boldsymbol{RemaingTime}}{\boldsymbol{C}}$$

- Idea: It leads to more thinking time in the begining.
  - **RemainingTime**: Total usable time left in the whole game.
  - **ThinkingTime**: How much time to use in one single move.

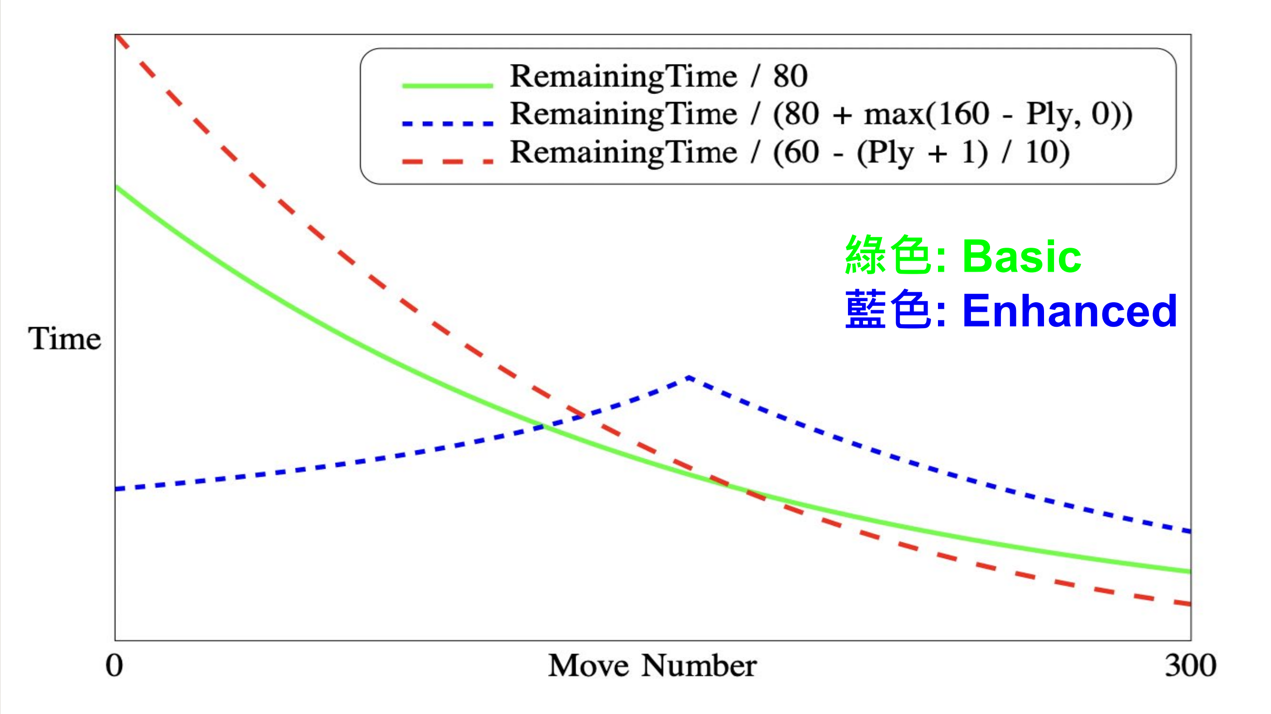
STATIC STRATEGIES: ENHANCED FORMULA

$$\mathbf{ThinkingTime} = \frac{\boldsymbol{RemaingTime}}{\boldsymbol{C} + \mathbf{max}(\boldsymbol{MaxPly} - \boldsymbol{Ply},\ \mathbf{0})}$$

- Idea: Use less time in the beginning and use more time in the middle of the game.
  - **Ply**: Total moves performed by both players.
  - **MaxPly**: Constant. Ply = MaxPly is the peak of ThinkingTime.

8

Comparison between Basic and Enhanced Formula



SEMI-DYNAMIC STRATEGIES

SEMI-DYNAMIC STRATEGIES: EXP (EXPECTED)

$$\mathbf{ThinkingTime} = \frac{\boldsymbol{RemaingTime}}{\boldsymbol{m_{expected}}}$$

- Idea: Like “improved” **basic formula** in static strategy.
  - **RemainingTime**: Total usable time left in the whole game.
  - $\boldsymbol{m_{expected}}$: Estimate remain moves to end game.

SEMI-DYNAMIC STRATEGIES: OPEN (OPENING)

$$\text{ThinkingTime} = f_{opening} \cdot \boxed{\frac{RemaingTime}{m_{expected}}}$$

- Strategy based on EXP.
  - $f_{opening}$: > 1, constant.

SEMI-DYNAMIC STRATEGIES: MID (MEDIUM)

$$\mathbf{ThinkingTime} = [1 + f_{Gaussian}(current\ move\ number)] \cdot \boxed{\frac{RemaingTime}{m_{expected}}}$$

- Strategy based on EXP.
  - Like **Enhanced formula** in static strategy but it is a semi-dynamic verison.
- The time is increased by a percentage determined by a Gaussian function over the set of move numbers.

SEMI-DYNAMIC STRATEGIES: KAPPA

- Critical moves have to spend more time.
- Take Chinese Chess for example:

MCTS game

game \ m	1	2	3	4	5	6	…	99	100
1	k^1	k^2	k^3	k^4	k^5	k^6	…	k^{99}	k^{100}
2	k^1	k^2	k^3	k^4	k^5	k^6	…	k^{99}	k^{100}
3	k^1	k^2	k^3	k^4	k^5	k^6	…	k^{99}	k^{100}
…	…	…	…	…	…	…	…	…	…
1000	k^1	k^2	k^3	k^4	k^5	k^6	…	k^{99}	k^{100}
	k_{avg}^1	k_{avg}^2	k_{avg}^3	k_{avg}^4	k_{avg}^5	k_{avg}^6		k_{avg}^{99}	k_{avg}^{100}

criticality rate

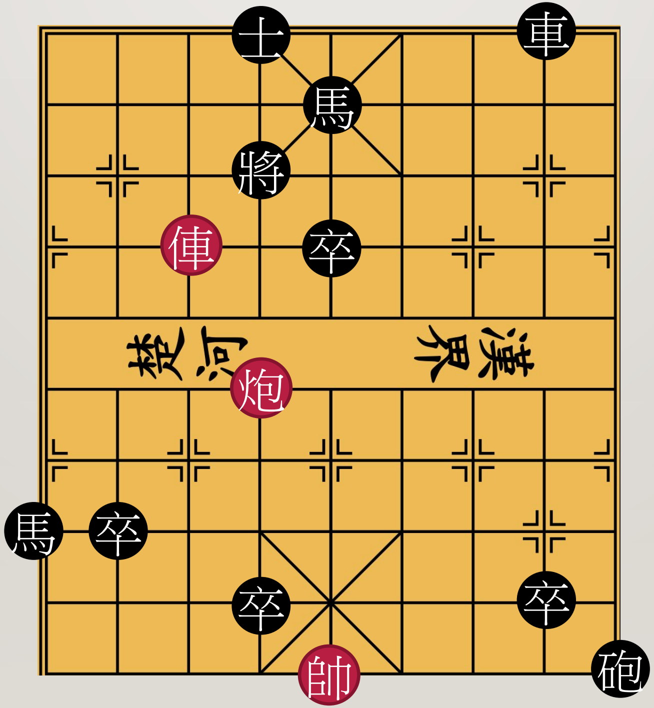
KAPPA

- $k_{avg}^{1}, k_{avg}^{2}, k_{avg}^{3}, k_{avg}^{4}, k_{avg}^{5} \ldots k_{avg}^{100}$
- Store all k in a database.
- We have to spend more time on m when it has higher k.
- “Precompute” time allocated (from database) before every game was playing.

KAPPA – IDENTIFY CRITICALITY

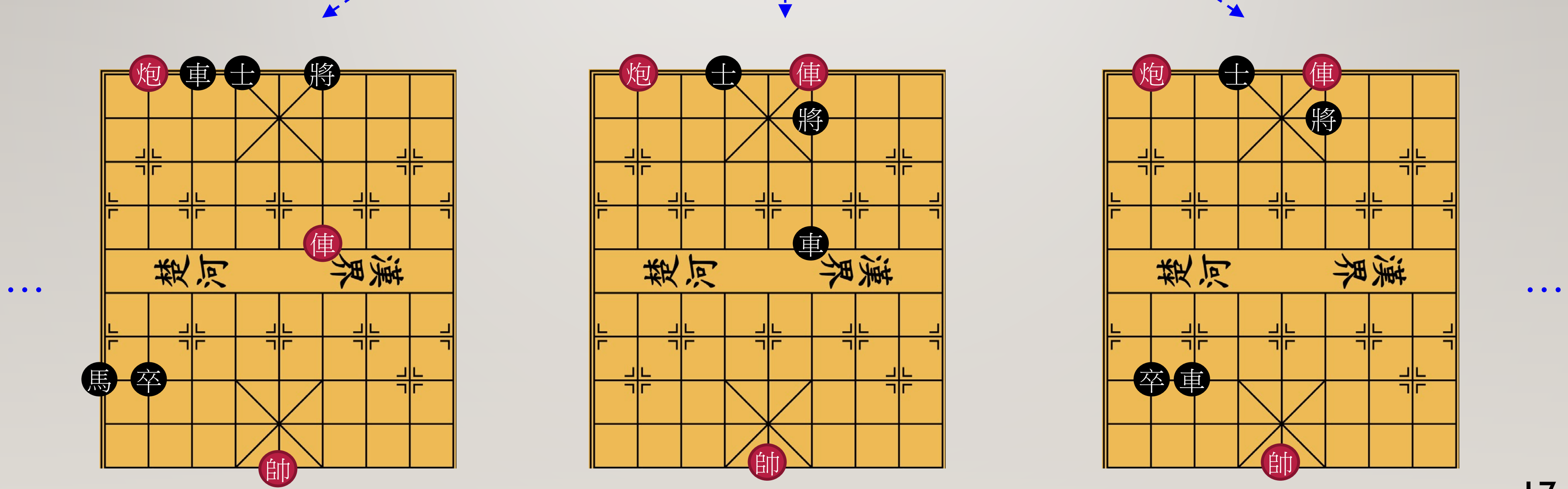
In one MCTS game:

m = 60



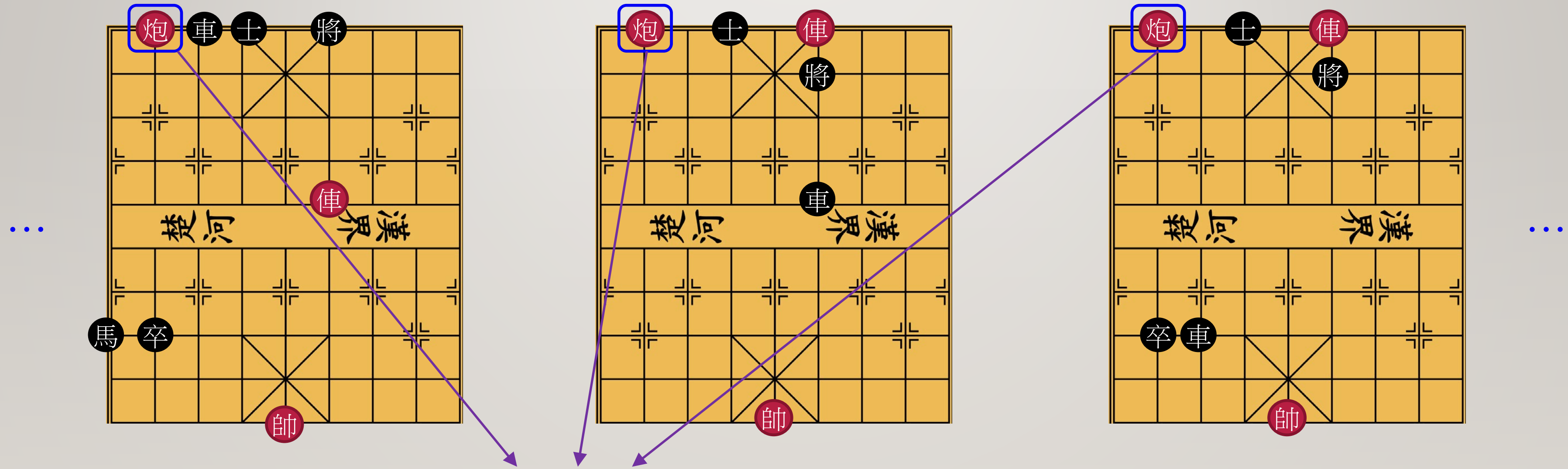
KAPPA – IDENTIFY CRITICALITY

simulate until end game



KAPPA – IDENTIFY CRITICALITY

critical move



$$k^{60} \propto \frac{3500}{5000} \longrightarrow \text{add into database} \longrightarrow k^{60}_{avg} \text{ increase}$$

playout count N

DYNAMIC STRATEGIES

DYNAMIC STRATEGIES: BEHIND (THINK LONGER WHEN BEHIND)

- Idea: After the regular search, but the win rate of the best move is lower than a threshold (ex: win rate < 0.5).
  - We can use more time to think longer (ex: +30% of ThinkingTime).
- We can trigger it N times.
  - If N = 1, we call it **BEHIND-1**.
  - If N > 1, we call it **BEHIND-N**.

DYNAMIC STRATEGIES: UNST (UNSTABLE)

- Idea: After the regular search, if the most-visited move doesn't match the highest-valued (win rate) move, then we can think longer.
  - Maybe a better move had just been discovered.
- We can trigger it N times.
  - If N=1, we call it **UNST-1**.
  - If N>1, we call it **UNST-N**.

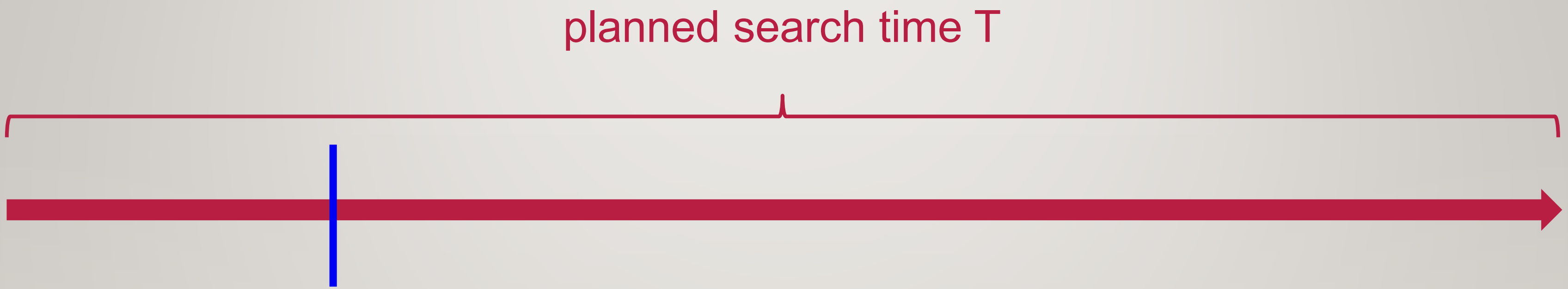
DYNAMIC STRATEGIES: CLOSE

- Idea: After the regular search, but the most-visited move and the second-most-visited move are "close", then we can think longer.
- We can trigger it N times.
  - If N=1, we call it **CLOSE-1**.
  - If N>1, we call it **CLOSE-N**.

DYNAMIC STRATEGIES: EARLY

- Idea: Terminating the search process as early as possible when the best move cannot change anymore.
  - When the difference between the most-visited move and the second-most-visited move is large, then we can terminate the search process.
- **EARLY-A EARLY-B EARLY-C** are strategies based on the idea of EARLY.

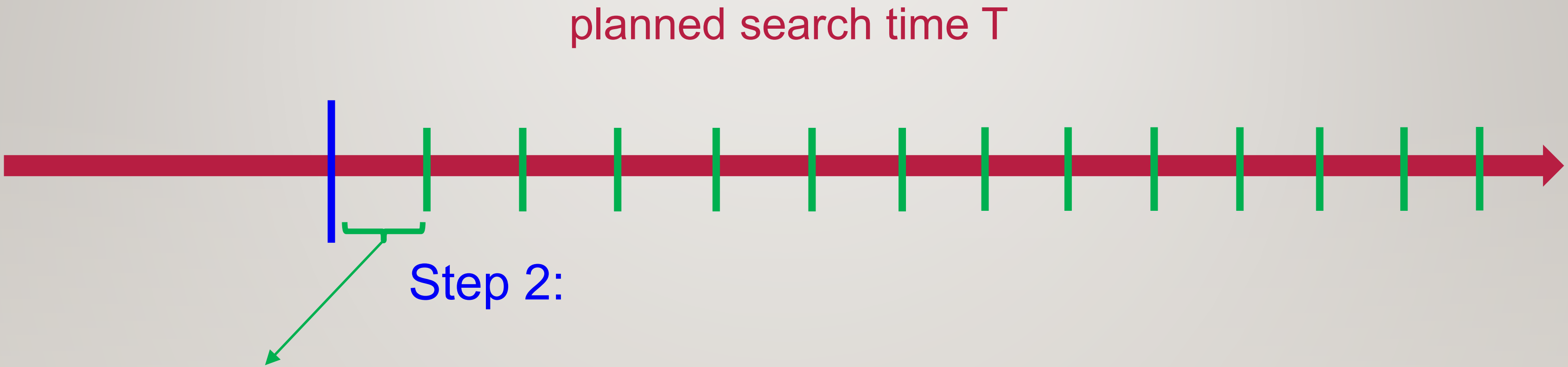
EARLY-A



Step 1:

- Calculate playout count per second.
- We use this to estimate how many times of playout [n] we can do in given remaining time [t].

EARLY-A



- In regular intervals (ex: 50 playouts), check 1^{st} best move and 2^{nd} best move.
- If total remaining expected playout all given to 2^{nd} , and 2^{nd} still can't pass 1^{st} , then exit safely.

EARLY-A

For example:

In a certain interval:

1st visit count: 10000

2nd visit count: 4000

and we have expected 5000 playout left.

If all 5000 visits 2nd , then 2nd has 9000. We still selected 1st .

So we don’t have to do the remaining 5000 playout.

EARLY-B

For example:

Origin:

total time = time limit = 40(s)

We expected that EARLY-A can save 1/5 of time.

$f_{earlyexit}$

So we set total time = 40*(5/4) = 50(s)

If EARLY-A works as expected, we will spend 50*(4/5) = 40(s)

still meets the time limit.

EARLY-C

In general, not all remaining will visit 2nd. So we estimate a proportion $p_{earlyexit}$

For example:

In a certain interval:

1st visit count: 10000

2nd visit count: 4000

and we have expected 9000 playout left, and $p_{earlyexit} = 0.25$

If 9000*0.25 = 2250 visits 2nd , then 2nd has 6250. We still selected 1st .

So we don’t have to do the remaining 9000 playout.

EXPERIMENT

CASE1: ERICA-BASELINE

- Erica-Baseline: Use Enhanced formula with BEHIND-1 and UNST-1.
  - Combine static strategy and dynamic strategy.

Algorithm 1 Think Longer When Behind

ThinkingTime $\leftarrow$ AllocateThinkingTime()
Think(ThinkingTime)

Enhanced formula

if Root.UCTmean $< T$ **then**
  Think($P \times$ ThinkingTime)
end if

Think-Longer-When-Behind

if MostVisitedMove is not HighestValueMove **then**
  Think(ThinkingTime/2)
end if
Play(MostVisitedMove)

Unstable-Evaluation

CASE1: ERICA-BASELINE

Table 1. Performance of ERICA's time management according to [13].

Player	Win rate against GNU Go	95% conf. int.
Basic formula	28.6%	27.3%–29.9%
Enhanced formula	31.4%	30.1%–32.7%
ERICA-BASELINE	34.3%	33.0%–35.7%

CASE2: SEMI-DYNAMIC STRATEGIES

Player	Win rate against GNU Go	95% conf. int.
EXP-STONES	25.5%	24.3%–26.7%
EXP-STONES with OPEN	32.0%	30.8%–33.4%
EXP-STONES with MID	30.6%	29.3%–31.9%
EXP-STONES with KAPPA-EXP	31.7%	30.5%–33.0%
EXP-STONES with KAPPA-LM	31.1%	29.9%–32.4%
ERICA-BASELINE	34.3%	33.0%–35.7%

EXP-STONES computes $m_{expected}$ by total number of stones in current board.

CASE3: DYNAMIC STRATEGIES

- Strategies based on EXP.
  - Combine dynamic strategy with EXP.

Player	Win rate against GNU Go	95% conf. int.
EXP-STONES with BEHIND	25.6%	24.4%–26.9%
EXP-STONES with UNST-1	33.6%	32.3%–34.9%
EXP-STONES with UNST-N	35.8%	34.4%–37.1%
EXP-STONES with CLOSE-1	32.6%	31.3%–33.9%
EXP-STONES with CLOSE-N	36.5%	35.2%–37.9%
EXP-STONES with EARLY-A	25.3%	24.1%–26.5%
EXP-STONES with EARLY-B	36.7%	35.4%–38.0%
EXP-STONES with EARLY-C	39.1%	38.0%–40.8%
EXP-STONES	25.5%	24.3%–26.7%
ERICA-BASELINE	34.3%	33.0%–35.7%

PONDERING

- Using Opponent's time.
- Standard Pondering.
  - At opponent's turn, keep thinking until my turn.
  - Re-use subtree of the opponent's played move.
- Focused Pondering.
  - Only search for N best moves.
  - If the opponent's played move is among the selected moves, then reduce ThinkingTime.