Scheduling for Parallel Alpha-Beta Search Based on Probability of Search Necessity

Chikayama & Tsuruoka Lab.
D1    Akira URA

2011.12.16
Outline

- Introduction
- Proposed method
- Evaluation
- Conclusion
Outline

- Introduction
- Proposed method
- Evaluation
- Conclusion
Background

- Searching deeper game trees is necessary to build stronger computer game players.

→ Parallelizing game tree search is important.

Our goal is to realize high performance in large-scale parallel game tree search.
Minimax search

- An evaluation value of a parent is selected from its children.
- Each player maximizes the evaluation value from its own viewpoint.

![Minimax Tree](image-url)
There are subtrees which can be cut off.
The subtrees need not to be searched.

$\alpha\beta$ search

$\max(40, \min(35, x)) = 40$

![Diagram of tree with nodes labeled A, B, C, D, E, F, G. Node A is a maximizer with value 40. Node B is a minimizer with value 40. Node C is a minimizer with value x. The subtree rooted at F is cut off.]}
Move ordering

- When the best child is searched first, pruning occurs most often.

→ Children are ordered before they are searched.

\[
\text{max}(40, \text{min}(35, x)) = 40
\]

Good ordering

maximize

Bad ordering

minimize
Difficulty of parallel $\alpha\beta$ search

- Parallel $\alpha\beta$ search is difficult because some computation may be wasted.
YBWC (Young Brothers Wait Concepts) [Feldmann, 1993]

- After search of the leftmost child has finished, the rest are searched in parallel.
- If the leftmost child is the best one, the number of nodes visited is the same as the sequential search.
YBWC (Young Brothers Wait Concepts) [Feldmann, 1993]

- After search of the leftmost child has finished, the rest are searched in parallel.
- If the leftmost child is the best one, the number of nodes visited is the same as the sequential search.
Problems in YBWC

- There are many waiting points for results.
  → Processes tend to be idle.
Speculative execution and two-level scheduling [Ura, et al., 2010]

- Ordering of children is not perfect.
  - Nodes expected to be pruned may become necessary.
  - Such nodes are searched speculatively.

- Children expected to be pruned have low priorities until the termination of the eldest brother.
To achieve further improvements

- Low priority nodes which are less likely to be pruned must be selected.
- In previous studies, nodes are searched left to right.

→ Search necessity should be considered more.
Purpose

To achieve further improvements of large-scale parallel $\alpha\beta$ search with the speculative execution, we proposed a scheduling method in which nodes more likely not to be pruned have higher priorities.
Outline

- Introduction
- Proposed method
- Evaluation
- Conclusion
Minimal tree [Knuth, et al., 1975]
A minimal tree consists of nodes which must be searched by $\alpha\beta$ search.

- The root node is PV.
- The best child of PV is PV and the rest are CUT.
- The best child of CUT is ALL and the rest are pruned.
- All children of ALL are CUT.
Basic idea

- Accuracy of move ordering is different between nodes.

Which to inspect?

Probabilities to be the best child:

- 98% 2%
- 60% 40%
Basic idea

- Accuracy of move ordering is different between nodes.

Probabilities to be the best child:
- 98% chance of the left node being the best child.
- 2% chance of the right node being the best child.
- 60% chance of the left node being the best child.
- 40% chance of the right node being the best child.
Basic idea

- Accuracy of move ordering is different between nodes.

![Diagram showing accuracy of move ordering between nodes.](image)

**CUT**

98% 2%

60% 40%

Probabilities to be the best child
Proposed method

- Calculates probability $P_n$ of each node $n$ to be searched as PV, CUT or ALL.
  - $P_n$ is the probability that $n$ is not pruned.

$$P_n = P_{n}^{PV} + P_{n}^{CUT} + P_{n}^{ALL}$$

- Searches a node $n$ with the largest $P_n$. 

![Diagram with probabilities]
Proposed method

- Calculates probability $P_n$ of each node $n$ to be searched as PV, CUT or ALL.
  - $P_n$ is the probability that $n$ is not pruned.
  
  $$P_n = P_{n}^{PV} + P_{n}^{CUT} + P_{n}^{ALL}$$

- Searches a node $n$ with the largest $P_n$. 
Proposed method

- Calculates probability $P_n$ of each node $n$ to be searched as PV, CUT or ALL.
  - $P_n$ is the probability that $n$ is not pruned.

$$P_n = P_{n}^{PV} + P_{n}^{CUT} + P_{n}^{ALL}$$

- Searches a node $n$ with the largest $P_n$. 

![Diagram](image-url)
Proposed method

- Calculates probability $P_n$ of each node $n$ to be searched as PV, CUT or ALL.
  
  $P_n$ is the probability that $n$ is not pruned.

\[ P_n = P_{n}^{PV} + P_{n}^{CUT} + P_{n}^{ALL} \]

- Searches a node $n$ with the largest $P_n$. 
How to calculate $P_n$ for each node $n$

- For the root node,

$$P^{PV}_{\text{root}} = 1.0 \quad \text{and} \quad P^{\text{CUT}}_{\text{root}} = P^{\text{ALL}}_{\text{root}} = 0.0 .$$

- Probabilities of children are calculated recursively from the probability of the parent.

$$(P^{PV}_{n_p}, P^{\text{CUT}}_{n_p}, P^{\text{ALL}}_{n_p}) \rightarrow (P^{PV}_{n_i}, P^{\text{CUT}}_{n_i}, P^{\text{ALL}}_{n_i})$$

- $n_p$ denotes a parent node.
- $n_i$ denotes a $i$-th child of the parent.
Best probability $b_i$ and first-cut probability $c_i$

To calculate $P_{n_i}$, two probabilities are used.

- **Best probability $b_i$**
  A probability that $n_i$ is the best child among the children

- **First-cut probability $c_i$**
  A probability that pruning occurs at $n_i$ and doesn’t occur at the elder brothers
Estimating $b_i$ and $c_i$

- By making histograms statistically
  - Best probability $b_i$ is calculated as follows:
    \[
    b_i = \frac{\text{(Frequency that } n_i \text{ is the best child)}}{\text{(Frequency of searching } n_i)}
    \]

- Used features
  - realization probability
  - search depth
Outline

- Introduction
- Proposed method
- Evaluation
- Conclusion
Experiments

Simulations with artificial trees
+estimating best probabilities and first-cut probabilities

Simulations with game trees of shogi
+considering real processing time

Real parallel $\alpha \beta$ search for shogi
Simulations with artificial trees

- Why artificial trees are used?
  - To evaluate when best probabilities $b_i$ and first-cut probabilities $c_i$ are given

- Why simulations are used?
  - To consider time only on processing leaves
  - To ignore overheads of the proposed method
How to simulate parallel tree search

- Simulations of parallel tree search by $N$ processes are done by iterating the following operations.
  - Choosing $N$ leaf nodes based on priorities
  - Processing those leaves and getting their results
  - Reflecting the results in the game tree
- The number of iterations is the execution steps.
How to simulate parallel tree search

- Simulations of parallel tree search by $N$ processes are done by iterating the following operations.
  - Choosing $N$ leaf nodes based on priorities
  - Processing those leaves and getting their results
  - Reflecting the results in the game tree
- The number of iterations is the execution steps.
How to simulate parallel tree search

- Simulations of parallel tree search by $N$ processes are done by iterating the following operations.
  - Choosing $N$ leaf nodes based on priorities
  - Processing those leaves and getting their results
  - Reflecting the results in the game tree

- The number of iterations is the execution steps.
How to simulate parallel tree search

- Simulations of parallel tree search by $N$ processes are done by iterating the following operations.
  - Choosing $N$ leaf nodes based on priorities
  - Processing those leaves and getting their results
  - Reflecting the results to the game tree
- The number of iterations is the execution steps.
Methods compared

- Proposed method
  - Nodes are searched in decreasing order of $P_n$.

- Previous method
  - Nodes in which $P_n = 1$ are searched in advance.
  - Next, other nodes are searched left to right.

<table>
<thead>
<tr>
<th>$P_n$</th>
<th>1.0</th>
<th>0.3</th>
<th>1.0</th>
<th>0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>proposed</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>previous</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>
Simualtion settings

- With the ternary trees and depth 16
- Average execution steps of 100 trees
- Ideal case and realistic case are simulated.

ideal

realistic

95% nodes 100% 0% 0%
5% nodes 73% 20% 7%

93% 5% 2%
73% 20% 7%

probabilities to be the best child
Simulation results with artificial trees

- In the ideal case

<table>
<thead>
<tr>
<th>the number of processes</th>
<th>1</th>
<th>16</th>
<th>256</th>
<th>4096</th>
</tr>
</thead>
<tbody>
<tr>
<td>previous (std. dev.)</td>
<td>15055</td>
<td>1061</td>
<td>88.2</td>
<td>36.6</td>
</tr>
<tr>
<td>proposed (std. dev.)</td>
<td>15055</td>
<td>1061</td>
<td>73.0</td>
<td>6.7</td>
</tr>
</tbody>
</table>
Simulation results with artificial trees

- In the realistic case

<table>
<thead>
<tr>
<th>the number of processes</th>
<th>1</th>
<th>16</th>
<th>256</th>
<th>4096</th>
</tr>
</thead>
<tbody>
<tr>
<td>previous (std. dev.)</td>
<td>25295</td>
<td>1888</td>
<td>479</td>
<td>177</td>
</tr>
<tr>
<td>(std. dev.)</td>
<td>4605</td>
<td>358</td>
<td>162</td>
<td>55.8</td>
</tr>
<tr>
<td>proposed (std. dev.)</td>
<td>25295</td>
<td>1885</td>
<td>336</td>
<td>82.9</td>
</tr>
<tr>
<td>(std. dev.)</td>
<td>4605</td>
<td>356</td>
<td>116</td>
<td>16.5</td>
</tr>
</tbody>
</table>
Implementation of the parallel shogi program

- With the master-worker model
  - The master performs simple $\alpha\beta$ search without sophisticated techniques.
  - Workers call a search function of Gekisashi.
Simulations with game trees of shogi

- To evaluate the performance including estimation of best probability $b_i$ and first-cut probability $c_i$
- Communication time and overheads of proposed methods are ignored.
- 30 game positions are searched.
Simulation results with actual trees

- Geometric mean of the execution steps over the 30 positions

<table>
<thead>
<tr>
<th>the number of workers</th>
<th>1</th>
<th>16</th>
<th>256</th>
<th>4096</th>
</tr>
</thead>
<tbody>
<tr>
<td>previous</td>
<td>14338</td>
<td>1030</td>
<td>87.2</td>
<td>22.0</td>
</tr>
<tr>
<td>proposed</td>
<td>14338</td>
<td>984</td>
<td>75.5</td>
<td>20.3</td>
</tr>
</tbody>
</table>

- The difference between the two methods is smaller than that with artificial trees.
Real parallel game tree search for shogi

- Evaluates the performance including overheads of the proposed method.
- The 30 game positions are searched.
- The experiment environment is as follows.
  - Each node has two quad core Xeon E5530 processors.
    - 8 MB cache, 2.40 GHz, 5.86 GT/s QPI
  - 24 GB memory
  - 10 Gbps Ethernet
Results of real search

- Average time over 30 trials for each position
- Geometric mean of the average time over 30 positions

<table>
<thead>
<tr>
<th>the number of workers</th>
<th>1</th>
<th>16</th>
<th>64</th>
<th>256</th>
<th>976</th>
</tr>
</thead>
<tbody>
<tr>
<td>previous</td>
<td>84.89</td>
<td>20.33</td>
<td>11.13</td>
<td>7.10</td>
<td>5.76</td>
</tr>
<tr>
<td>(std. dev.)</td>
<td>3.44</td>
<td>1.57</td>
<td>0.80</td>
<td>0.52</td>
<td></td>
</tr>
<tr>
<td>proposed</td>
<td>84.89</td>
<td>19.22</td>
<td>11.61</td>
<td>7.18</td>
<td>5.78</td>
</tr>
<tr>
<td>(std. dev.)</td>
<td>2.92</td>
<td>1.65</td>
<td>0.78</td>
<td>0.56</td>
<td></td>
</tr>
</tbody>
</table>

- There is no difference in performance.
Summary of the evaluation

Simulations with artificial trees
The proposed method is effective.

Simulations with game trees of shogi
The proposed method is slightly effective.

Real parallel $\alpha \beta$ search for shogi
The effectiveness is not observed.
Why isn’t the proposed method effective with actual game trees?

- Aren’t tasks executed speculatively?
  → We verified that a sufficient number of speculative tasks are executed.

- Are overheads of the proposed method large?
  → We verified that the overheads aren’t so large.
Accuracy of probability estimation

- The difference in performance is small also in simulations with actual trees.
  → Estimation of probabilities can be imperfect.
  → The differences of best probability $b_i$ and first-cut probability $c_i$ between nodes aren’t large.
  → Find better features for estimating $b_i$ and $c_i$. 
Outline

- Introduction
- Proposed method
- Evaluation
- Conclusion
Summary

- We estimate probabilities that nodes are necessary to be searched.
- We utilize the probabilities as priorities for parallel $\alpha\beta$ search.
- We show effectiveness of the proposed method for artificial trees.
- However, the effectiveness is not observed for actual trees.
Future work

- To consider better features for estimation of best probability $b_i$ and first-cut probability $c_i$
  - The number of children
  - Estimated evaluation value
  - Estimated threshold for pruning

- To improve the implementation of the parallel program
  - Optimization for calculation of the probabilities
Thank you for your listening!