JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 30, 701-712 (2014)

DOI:10.1688/JISE.2014.30.3.10

Analysis of Performance of Consultation Methods in Computer Chess

KUNIHITO HOKI[1], SEIYA OMORI[2] AND TAKESHI ITO[3]
[1]*The Center for Frontier Science and Engineering*
[2]*Department of Communication Engineering and Informatics*
[3]*Department of Information and Communication Engineering*
University of Electro-Communications
Chofu, Tokyo 182-8585, Japan

The performance of consultation methods, *i.e.*, majority voting, the optimistic selection rule, and the pseudo-random number (PRN) ensemble method, are examined in computer chess using 2180 chess problems. Here, the optimistic selection rule selects a program that returns the highest search value, and the PRN ensemble consists of multiple individual copies of one base program, and each copy is diversified by adding random numbers to the evaluation function of the base program. We carried out empirical experiments by using state-of-the-art chess-program Crafty as the base program. We found that the percentage of correct answers increased from 55.6 to 57.1% using optimistic selection from the PRN ensemble. The experimental results indicated that the consultation methods allowed simple yet effective distributed computing in chess.

Keywords: computer chess, chess problem, consultation method, majority voting, optimistic selection, pseudo-random number (PRN) ensemble

1. INTRODUCTION

The old proverb of "two heads are better than one", means that an ensemble of people may be able to solve a problem that an individual cannot. This explains our original motivation for designing consultation methods in computer games. That is, an ensemble of game programs may be able to play a better move than an individual program can (see Fig. 1). Although much successful work with ensemble based systems in computer science has been done [1], designing an ensemble in computer games still remains difficult.

Thus far, three simple techniques of consultation have been reported in shogi, which is a Japanese chess variant. The first is *majority voting* to select a single move from multiple moves provided by the ensemble of programs. This selection rule had first been examined in 5×5 shogi, a shogi variant that uses a small board and a limited set of pieces [2]. Subsequently, majority voting was examined in standard shogi [3]. The second technique is the *optimistic selection rule*. A shogi program usually returns a search value in addition to a move to play. This selection rule makes use of the search value, and selects a program that returns the highest search value [4]. The third technique is the *pseudo-random number* (*PRN*) *ensemble*. Here, the ensemble is built using multiple copies of one base program, and each copy is diversified by adding random numbers to the evaluation function of the base program [3, 4].

The primary goal of the consultation methods is to strengthen base programs by composing an ensemble of these programs. In addition to the primary goal, these meth-

Received February 28, 2013; accepted June 15, 2013.
Communicated by Hung-Yu Kao, Tzung-Pei Hong, Takahira Yamaguchi, Yau-Hwang Kuo, and Vincent Shin-Mu Tseng.

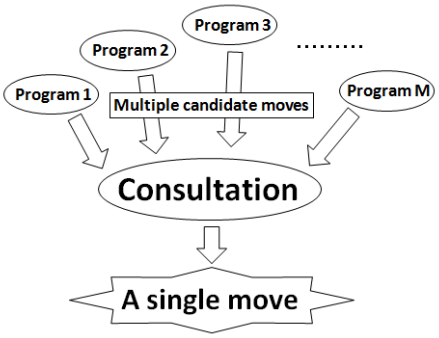


Fig. 1. Methods of consultation and their goal. Best opinion is selected from multiple opinions.

ods can be expected to provide a fail-safe distributed computing system. That is, the breakdown of a component in the entire system merely means the number of voters is reduced by one.

We assessed the performance of these consultation methods, *i.e.*, majority voting, the optimistic selection rule, and the PRN ensemble, in chess, which we report in this paper. We used Crafty as a fair implementation of the chess program [5]. It was developed by Dr. R. Hyatt and it finished second in both the 2010 Fifth Annual Americas' Computer Chess Championships (ACCA) and the 2010 World Computer Rapid Chess Championships. We evaluated its performance by using 2180 chess problems.

Chess is probably one of the most popular board games in the world and has several similarities to shogi in that (i) it is a two-player turn-based board game, (ii) it has the same origin as chaturanga, which was played before the 6th century in India, (iii) its main objective is to checkmate the opponent's king by forcing it under inevitable threat of capture, (iv) it has a promotion rule, and (v) the best algorithms of tree searches for its strength are $\alpha\beta$-search variants as far as is known.

There are also some differences between these two games. One of the clearest differences is the absence of the dropping rule in chess, *i.e.*, in shogi, a captured piece can be dropped back onto the board. The absence of this dropping rule makes the number of choices smaller than in shogi [6, 7]. Therefore, it would be an interesting study to measure the performance of these consultation methods in chess.

The remaining part of this paper is structured as follows. The next section reviews research related to the consultation methods in chess. The third section presents our results on the chess program, where the consultation methods are empirically evaluated. The last section is the conclusion.

2. RELATED WORK

Althöfer and Snatzke proposed 3-Hirn in two player games [8], where two programs make one proposal move, and a human selects a single move from these two and controls the time. They reported that human selection strengthens programs in the team by about 200 Elo points in chess, where the human has 1900 Elo points and is weaker than these programs. Because the consultation methods remove human intervention,

these can be regarded as a reduced form of 3-Hirn. An attempt to eliminate human intervention from 3-Hirn can also be found in 2-Hirn [9]. Here, a sub-computer in addition to the main computer was used to obtain human-like playing styles.

Obata *et al.* evaluated the performance of majority voting in shogi [3]. They observed performance increments from 50-100 Elo points in their experiments on three state-of-the-art shogi programs. Sugiyama *et al.* also assessed the performance of the optimistic selection rule in shogi [4], and reported that it outperformed majority voting. Majority voting has also been applied to a *go* program that carries out Monte-Carlo tree searches [10]. Here, its performance was evaluated using a state-of the-art go program called Fuego [11], and it outperformed a method of standard root parallelization in distributed-memory environments. These methods did not perform better than tree-search parallelization in shared-memory environments in these studies on majority voting and the optimistic selection rule.

Computer players that use majority voting can be found in recent computer shogi games, because consultation methods offer easy use of massive-scale distributed environments for strength, make players safer against failures in the system, and they can be combined with different parallel-search methods. Monju won third prize in the 19th computer shogi championship in 2009 [12]. Here, the PRN ensemble was built using Bonanza, whose source codes are available online [7]. Moreover, Akara 2010, which defeated one of the top female shogi players, Ichiyo Shimizu, employed majority voting and consisted of four different computer programs, Gekisashi, GPS Shogi, Bonanza, and YSS [13]. All four programs have won the computer shogi championships in the past 10 years.

Because consultation methods use distributed computing environments to obtain better moves to play, they can be regarded as the simple distributed computing of game-tree searches. The concepts and mechanisms for the consultation methods differ from the common tree parallelization of chess programs in which each processor takes responsibility for part of the game tree to conduct searches. One well-known method that has been applied to parallel searches in distributed environments is the young brothers wait concept (YBWC) [14]. Young brothers are child nodes expanded after the first child node and they are searched in parallel. YBWC assumes that the game tree is well ordered, *i.e.*, the best child will be expanded first for most of the tree search. Note that a well-crafted $\alpha\beta$-search program satisfies this assumption because a sequential $\alpha\beta$-search is effective when this property is satisfied. A speedup of about 35 was obtained for 24 test positions for a distributed system with 64 processors. This work used a transputer (Inmos transputer T800), which is hardware intended for parallel computing and a chess program called Zugzwang that ran on a single processor that only visited about 350 nodes per second. This hardware and software had totally different characteristics from those used today.

Himstedt proposed the extended use of pondering methods for distributed computing in chess [15]. The expected move of an opponent in ordinary pondering methods is considered as already played, and a computer player starts searching during the opponent's thinking time to avoid the computer from becoming idle while his/her opponent is thinking. The reported method starts searching speculatively ahead on the basis of the expected move sequence during both the player's and opponent's thinking times. The playing strength of GridChess in this study increased by 51 Elo points by using eight

workers. Himstedt also reported the performance of distributed tree searches using modern hardware and software on the basis of YBWC [15]. He built a parallelized version of Gaksch's Toga, which is based on Letouzey's Fruit, and observed an Elo difference of about 52 points between four- and one-node clusters, where each node contained four cores. Because the number of games was only 70, the Elo difference was not statistically significant. This work replicated a part of hash-table entries between all cluster nodes to keep the game tree well ordered. This replication incurred network communication, and high bandwidth and low latency interconnection networks were advantageous.

Other than chess, Brockington and Schaeffer proposed a distributed $\alpha\beta$-search, called asynchronous parallel hierarchical iterative deepening (APHID) [16]. Their method was based on a master/slave model of communication, where the master divided the game tree into fixed subsets and each slave took responsibility for some of the subtrees. It used the APHID table to control the work of all slaves. The Othello program Keyano sped up a 12-ply search of 74 test positions by around eight by using APHID on 16 processors. Kaneko and Tanaka also proposed a distribute $\alpha\beta$-search method using a master/slave model in shogi [13, 17]. Their method divided the game tree on the basis of move ordering and the program GPS Shogi increased the winning percentage from 50 to 70%. Their tree parallelization was successfully combined with majority voting by four different programs. Table 1 summarizes the details between different distributed-computing methods in chess and shogi. Kishimoto and Schaeffer proposed transposition-table driven scheduling alpha-beta (TDSAB) [18], which combined two different methods of transposition-table driven scheduling (TDS) of single-agent searches and a variant of $\alpha\beta$ search MTD(f). Performance in their study was assessed with Awari and Amazons.

Table 1. Distributed computing using high-performance chess and shogi programs. The forth column shows our rough estimate of effectiveness. Note that each method has been demonstrated by using the different game and computational resources.

Method	Game	Workers	Effectiveness
3-Hirn [8]	Chess	A human and different programs	High
Optimistic pondering [15]	Chess	Same programs	Low
Distributed tree search based on YBWC [15]	Chess	Same programs	Medium
Majority voting [13]	Shogi	Different programs	Medium
Majority voting using PRN ensemble [3]	Shogi	Same programs	Low
Optimistic selection using PRN ensemble [4]	Shogi	Same programs	Low
Distributed tree search based on Kaneko and Tanaka [13]	Shogi	Same programs	Medium

Consultation methods also have similarities to parallel dovetailing methods and parallel random search algorithms in single-agent puzzles and AND/OR-game solvers, where multiple solvers are executed for speedup [19-22]. Because the output of these solvers is the 'game is solved' or the 'game is not solved yet' rather than move candi-

dates with evaluation values, the selection rules for these candidates are not necessary in dovetailing methods.

3. EXPERIMENT

We assessed the performance of consultation methods using Crafty-23.4 whose source codes are available online [5]. It uses state-of-the-art techniques such as the principal variation search (PVS) [23-25], a capture search as a quiescence search, transposition tables indexed by Zobrist hashing [26], static exchange evaluation [27], killer and history heuristics [28, 29], null move pruning [30, 31], futility pruning [32, 33], and late move reductions [7, 34]. The base program in our research (i) receives a chess position, (ii) searched a game tree rooted at the position, and (iii) output a move with a search value. We allocated 1024 megabytes for the transposition table used by the tree search, which was sufficient for all the experiments.

The PRN ensemble consisted of M base programs, and determined which move to play by selecting one from M programs. The evaluation function to diversify the outputs of the base programs was modified by adding pseudo-random numbers from an approximated normal distribution $N(0, \sigma^2)$, where σ is the standard deviation. The values were generated on the basis of a central limit theorem that generated 12 uniform random numbers in intervals of $[0, \sigma]$, added them all up, and subtracted six. As a result, the values were in intervals of $[-6\sigma, 6\sigma]$. We pre-computed 4096 random numbers in this way from $N(0, \sigma^2)$ and a random number and a game position were paired up with the Zobrist hashing key of the position [26], so that the modified evaluation function with the random number always returned the same value for the same position. This modification was so simple that this hardly had any impact on the search speed.

The ensemble in our research behaved as follows: (i) M programs received the same chess position, (ii) M programs searched the same position individually using different series of pseudo-random numbers, (iii) M programs output multiple moves and corresponding search values, and (iv) a single move was selected based on majority voting or the optimistic selection rule. Majority voting is one of the most popular decision rules. When three programs output move A and two output B, the majority of move A is selected. However, the optimistic selection rule selects a program that returns the highest search value. An auxiliary rule for random selection is used in both cases to break a tie.

We evaluated the performance of programs by the percentages of correct answers to 2180 chess problems from the Encyclopedia of Chess Middlegame (the second section of 879 problems) [35], Win at Chess (300 problems) [36], and 1001 Winning Chess Sacrifices (1001 problems) [37].

3.1 Controlling Computational Resources through Nodes

We limited the computational resources for all tree searches by the number of nodes. All programs, *i.e.*, the base programs in the ensemble and the original program, searched 50, 100, and 200 kilonodes. The first reason as to why we limited computational resources by the number of nodes was the reasonable correlation between searched nodes and search time. The second reason was that the search results were irrelevant to com-

puter hardware specifications, so that a set of experiments could be conducted using multiple different computers in a laboratory. Moreover, because network communication between M base programs was only caused at the beginning and the end of game-tree searches, the communication overhead was not taken into account.

We first examined the setup for the experiments and the performance of the diversified base program by using standard deviation σ. Table 2 lists the dependence of standard deviation on the percentage of correct answers. Note that $\sigma = 0$ means that there were no modifications to the original program and the results for $\sigma > 0$ were averaged over 16 sets of experiments using different series of pseudo-random numbers. We can see that the original program that searched 50,000, 100,000, and 200,000 nodes gave respective correct answers to 54.6, 60.5, and 66.1% of problems. We can also see that the random numbers in the evaluation function did not decrease the performance of the base program if the standard deviation σ was less than 10 centipawns. Note that the base program valued one pawn at 100.

Table 2. Percentage $P(\sigma)$ of correct answers using base program, where σ is standard deviation for random numbers.

Nodes	$P(\sigma = 0)$	$P(1)$	$P(2)$	$P(4)$	$P(8)$	$P(12)$	$P(25)$	$P(50)$	$P(100)$
50,000	54.6	54.7	54.6	54.6	54.5	54.3	53.9	53.0	51.2
100,000	60.5	60.6	60.7	60.7	60.8	60.5	59.8	59.3	57.6
200,000	66.1	66.0	65.9	65.9	65.8	65.4	65.1	64.4	63.1

Table 3. Diversity of base program, where $D(\sigma)$ is percentage to output different answer from original and σ is standard deviation for random numbers.

Nodes	$D(\sigma = 0)$	$D(1)$	$D(2)$	$D(4)$	$D(8)$	$D(12)$	$D(25)$	$F(50)$	$D(100)$
50,000	0.0	6.5	10.2	14.1	16.9	19.5	22.1	26.4	32.6
100,000	0.0	6.6	9.7	12.4	14.5	16.0	18.4	22.4	27.5
200,000	0.0	6.5	9.0	10.8	12.8	13.6	16.2	19.0	23.1

These results indicate that such a small standard deviation suffices for the base program to diversify its outputs by a sufficient amount. Table 3 summarizes the diversity of the base program, where the results for $\sigma > 0$ were also averaged over 16 sets of experiments using different series of pseudo-random numbers. The base program with $\sigma = 8$ in this table output a different answer from the original with probabilities of 10% or more. Moreover, there was a preferred tendency, *i.e.*, the deeper the base program searched the game tree the less the random numbers diversified the output of the base program. That is, the increment in the number of nodes from 50,000 to 200,000 at $\sigma = 8$ increased the percentage of correct answers from 54.5 to 65.8% and decreased the percentage to output a different answer from the original from 16.9 to 12.8%.

Table 4 lists the percentages of correct answers using majority voting in M base programs. We can see from this table that the more M increases, the more the percentage of correct answer increases. However, the increment in the percentage of correct answers was very small and majority voting in 16 base programs with 200,000 nodes improved the percentage from 66.1 to 66.7%. This means that majority voting in the PRN ensemble was not effective in solving chess problems.

Table 4. Percentage $P(\sigma)$ of correct answers using majority voting, where σ is standard deviation for random numbers and M is size of ensemble players.

50,000 nodes (percentage using original program is 54.6%)								
M	$P(\sigma=1)$	$P(2)$	$P(4)$	$P(8)$	$P(12)$	$P(25)$	$P(50)$	$P(100)$
4	54.6	55.0	54.6	55.0	55.5	55.3	54.4	52.6
8	54.4	54.9	54.9	55.7	55.4	55.3	54.4	53.7
16	54.7	54.6	55.1	55.6	55.9	55.3	55.0	54.8
100,000 nodes (percentage using original program is 60.5%)								
M	$P(\sigma=1)$	$P(2)$	$P(4)$	$P(8)$	$P(12)$	$P(25)$	$P(50)$	$P(100)$
4	60.7	60.9	61.1	61.2	61.1	61.4	59.9	60.3
8	60.6	60.9	61.2	61.5	61.7	61.1	61.1	60.4
16	60.6	61.0	61.1	61.6	61.1	61.0	61.2	60.6
200,000 nodes (percentage using original program is 66.1%)								
M	$P(\sigma=1)$	$P(2)$	$P(4)$	$P(8)$	$P(12)$	$P(25)$	$P(50)$	$P(100)$
4	66.4	66.1	66.2	66.0	65.6	66.1	65.3	64.9
8	66.2	66.1	66.2	66.3	66.1	66.3	65.6	64.9
16	66.1	66.0	66.3	66.6	66.7	66.2	66.0	65.4

Table 5. Percentage $P(\sigma)$ of correct answers using optimistic voting, where σ is standard deviation for random numbers and M is size of ensemble players.

50,000 nodes (percentage using original program is 54.6%)								
M	$P(\sigma=1)$	$P(2)$	$P(4)$	$P(8)$	$P(12)$	$P(25)$	$P(50)$	$P(100)$
4	55.3	55.6	55.6	56.3	56.1	56.2	55.7	54.3
8	55.3	56.1	56.0	56.5	56.4	56.6	56.0	54.7
16	55.3	56.2	56.2	57.1	56.8	57.0	57.4	55.0
100,000 nodes (percentage using original program is 60.5%)								
M	$P(\sigma=1)$	$P(2)$	$P(4)$	$P(8)$	$P(12)$	$P(25)$	$P(50)$	$P(100)$
4	61.3	62.1	61.7	62.1	62.3	62.2	61.0	59.9
8	61.5	62.6	62.8	63.2	62.4	62.4	62.0	60.8
16	61.5	62.7	63.0	63.3	63.3	62.8	62.3	60.8
200,000 nodes (percentage using original program is 66.1%)								
M	$P(\sigma=1)$	$P(2)$	$P(4)$	$P(8)$	$P(12)$	$P(25)$	$P(50)$	$P(100)$
4	66.8	66.8	66.6	66.1	66.7	66.9	66.0	65.0
8	66.8	67.4	67.2	66.8	67.3	67.0	66.4	65.7
16	66.9	67.7	67.6	67.4	67.6	67.4	67.0	66.1

Table 5 summarizes the percentages of correct answers using the optimistic selection rule. Here, we observed greater improvements to performance than those with majority voting. The ensemble player of 16 base programs with 50,000 nodes improved the percentage from 54.6 to 57.1%, that with 100,000 nodes improved the percentage from 60.5 to 63.3%, and that with 200,000 nodes improved the percentage from 66.1 to 67.7%. Although the optimistic selection rule sufficiently improved performance, this method did not seem to outperform the original program that searched twice more nodes of the game tree. Increments in the nodes from 50,000 to 100,000 improved the percentage in the original program from 54.6 to 60.5%.

Finally, we assessed how the number of correct answers increased due to the optimistic selection rule. Table 6 lists two kinds of numbers. The first is the number of chess problems where the original program returned a correct answer and the ensemble program returned an incorrect answer (denoted as Correct→Incorrect). The second is the number of chess problems where the original program returned an incorrect answer and the ensemble player returned a correct answer (denoted as Incorrect→Correct). These results indicate that the changes between correct and incorrect answers are stochastic. That is, the performance of the base program was improved by optimistic selection for a specific chess problem with a certain probability. However, this method also had the probability of performance worsening for another specific chess problem.

Table 6. Changes in number of correct and incorrect answers, where M is size of ensemble players and each base program searches 50,000 nodes for chess problem with standard deviation for random numbers, $\sigma = 12$.

Majority voting		
	$M = 4$	$M = 16$
Incorrect → Correct	71	99
Correct → Incorrect	44	55
Optimistic Voting		
	$M = 4$	$M = 16$
Incorrect → Correct	83	129
Correct → Incorrect	41	72

3.1 Controlling Computational Resources by Depth

We limited the computational resource for tree searches by using the nominal depth of the programs. All programs, *i.e.*, the base programs of the ensemble and the original program, search with a depth of eight plies. Note that Crafty employs some methods of search-depth extensions and reductions for particular moves. Therefore, the nominal depth specified in this experiment was not equal to the depth of the actual game tree.

Table 7. Percentage $P(\sigma)$ of correct answers using base program, where σ is standard deviation of random numbers and nominal depth is eight.

	$\sigma = 0$	$\sigma = 100$	$\sigma = 1000$
$P(\sigma)$	59.0	59.2	58.2
Average nodes searched	65511	84951	173010

We examined the setup for the experiments and the performance of the diversified base program by using standard deviation σ. Table 7 lists the dependence of standard deviation on the percentage of correct answers and the number of averaged nodes. We can see that the random values from $N(0,1000^2)$ decreased the percentage from 59.0 to 58.2%. This decrement is unusually small, considering that the standard deviation equals a value of 10 pawns (corresponding to two rooks).

Table 8. Percentage $P(\sigma)$ of correct answers with nominal depth of eight, where σ is standard deviation for random numbers and M is size of ensemble players. Percentage using original program is 59.0%.

Majority voting				
M	$P(\sigma = 12)$	$P(25)$	$P(50)$	$P(100)$
4	59.5	60.1	59.9	61.2
8	59.8	60.2	59.8	61.6
16	59.7	60.2	60.6	61.3
Optimistic voting				
M	$P(\sigma = 12)$	$P(25)$	$P(50)$	$P(100)$
4	61.6	62.3	62.8	64.7
8	62.4	63.7	64.0	66.2
16	63.4	64.7	65.6	68.1

We can also see that the random values increased nodes that had to be searched with the depth of eight plies. This means that the random values decreased the amount of tree pruning and increased the size of the search space. This is why the percentage of correct answers did not decrease so much with a σ of two rooks. These results indicate that even when the computational resources were limited by the number of nodes the random values also diversified the search space from the original, and the PRN-ensemble player covered a larger search space than the original program did.

Table 8 summarizes the percentages of correct answers. Note that $\sigma = 0$ means that there are no modifications from the original program. When computational resources were limited by the search depth, we observed an unusual property in that the percentages increased even when σ was close to 100. Because the number of searched nodes increased with σ, these results do not tell us about the performance of consultation methods. As we observed in the previous subsection, the more M increases, the more the percentages increase. Also, the optimistic selection rule was better than majority voting.

5. CONCLUSION

We evaluated the performance of consultation methods, *i.e.*, majority voting, the optimistic selection rule, and the PRN ensemble, in chess. We found that optimistic selection from the PRN ensemble increased the percentages of correct answers by a sufficient amount, whereas majority voting of the PRN ensemble did not. We also found that the PRN ensemble not only diversified the evaluation values but also the search space for the chess problems. As a result, the PRN-ensemble player covered a larger search space than the original did.

The experimental results indicated that these consultation methods allow simple yet effective distributed computing in chess. However, although optimistic selection from the PRN ensemble sufficiently improved performance, these methods did not seem to outperform the original program that searched twice as many nodes of the game tree. We concluded from these observations that the selection rules discussed in this paper were rather unsophisticated. Human selection from two programs is capable of improving

performance by about 200 Elo points and doubling search time would not achieve this improvement [8], which implies the existence of more sophisticated methods for better ensemble systems. Building such systems in games would be an interesting problem in artificial intelligence. We would like to leave this problem for future work.

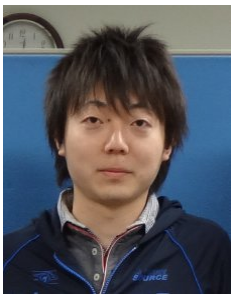
REFERENCES

1. R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits and Systems Magazine*, Vol. 6, 2006, pp. 21-45.
2. M. Hanawa and T. Ito, "The council system in brain game," in *Proceedings of Cognitive Science and Entertainment Symposium*, 2009, pp. 72-75.
3. T. Obata, T. Sugiyama, K. Hoki, and T. Ito, "Consultation algorithm in computer shogi – A move decision by majority –," in *Computer and Games*, J. van den Herik, H. Iida, and A Plaat, eds., LNCS 6515, Springer, 2010, pp. 156-165.
4. T. Sugiyama, T. Obata, K. Hoki, and T. Ito, "Optimistic selection rule for ensemble approach to improving strength of shogi program," in *Computer and Games*, eds., J. van den Herik, H. Iida, and A. Plaat, LNCS 6515, Springer, 2010, pp. 156-165.
5. R. Hyatt, Crafty 23.4, ftp://ftp.cis.uab.edu/pub/hyatt, last access, 2012.
6. H. Matubara, H. Iida, and R. Grimbergen, "Chess, shogi, go, a natural development in game research," *ICCA Journal*, Vol. 19, 1996, pp. 103-112.
7. K. Hoki and M. Muramatsu, "Efficiency of three forward-pruning techniques in shogi: Futility pruning, null-move pruning, and Late Move Reduction (LMR)," *Entertainment Computing*, Vol. 3, 2012, pp. 51-57.
8. I. Althöfer and R. G. Snatzke, "Playing games with multiple choice systems," in *Proceedings of Computers and Games*, J. Schaeffer, LNCS 2883, 2002, pp. 142-153.
9. K. Shibahara, T. Goto, N. Inui, and Y. Kotani, "2-Hirn", in *Proceedings of the 8th Game Programing Workshop*, 2003, pp. 59-66.
10. Y. Soejima, A. Kishimoto, and O. Watanabe, "Evaluating root parallelization in go," *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 2, 2010, pp. 278-287.
11. M. Enzenberger, M. Müller, B. Arneson, and R. Segal, "FUEGO – an open-source framework for board games and Go engine based on Monte-Carlo tree search," *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 2, 2010, pp. 259-270.
12. T. Ito, "Consultation player "Monju" – A majority rule to increasing the winning rate," *IPSJ Magazine*, Vol. 50, 2009, pp. 887-894.
13. K. Hoki, T. Kaneko, D. Yokoyama, T. Obata, H. Yamashita, Y. Tsuruoka, and T. Ito, "A system-design outline of the distributed-shogi-system Akara 2010," in *Proceedings of the 14th IEEE/ACIS International Conference on Software Engineering*, *Artificial Intelligence*, *Networking and Parallel/Distributed Computing*, 2013, pp. 466-471.
14. R. Feldmann, P. Mysliwietz, and B. Monien, "A fully distributed chess program," in *Advances in Computer Chess 6*, 1991, pp. 1-27.
15. K. Himstedt, "GridChess: combining optimistic pondering with the young brothers wait concept," *ICGA Journal*, Vol. 35, 2012, pp. 67-79.

16. M. G. Brockington and J. Schaeffer, "The APHID parallel alpha-beta search Algorithm," in *Proceedings of the 8th IEEE Symposium of Parallel and Distributed Processing*, 1996, pp. 428-432.
17. T. Kaneko and T. Tanaka, "Distributed game-tree search based on prediction of best moves," *IPSJ Journal*, Vol. 53, 2012, pp. 2517-2524.
18. A. Kishimoto and J. Schaeffer, "Transposition table driven work scheduling in distributed game-tree search", in *Proceedings of the 15th Canadian Conference on Artificial Intelligence*, LANI, Vol. 2338, 2002, pp. 56-68.
19. R. Mehrotra and E. F. Gehringer, "Superlinear speedup through randomized algorithms," in *Proceedings of International Conference on Parallel Processing*, 1985, pp. 291-300.
20. K. Knight, "Are many reactive agents better than a few deliberative ones?" in *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, http://dl.acm.org/citation. cfm?id=1624025.1624086, Vol. 1, 1993, pp. 432-437.
21. R. Valenzano, N. Sturtevant, J. Schaeffer, K. Buro, and A. Kishimoto, "Simultaneously searching with multiple settings: An alternative to parameter tuning for suboptimal single-agent search algorithms," in *Proceedings of ICAPS*, 2010, pp. 177-184.
22. K. Hoki and T. Ito, "Distributed computing of df-pn search using pseudo-random number," in *Proceedings of the 16th Game Programming Workshop*, 2011, pp. 116-119.
23. J. Pearl, "Scout: A simple game-searching algorithm with proven optimal properties," in *Proceedings of the 1st Annual National Conference on Artificial Intelligence*, 1980, pp. 143-145.
24. T. A. Marsland and M. Campbell, "Parallel search of strongly ordered game trees," *ACM Computing Surveys*, Vo. 14, 1982, pp. 533-551.
25. A. Reinefeld, "An improvement to the scout tree search algorithm," *ICCA Journal*, Vol. 6, 1983, pp. 4-14.
26. A. L. Zobrist, "A new hashing method with application for game playing," *ICCA Journal*, Vol. 13, 1990, pp. 69-73.
27. F. Reul, "tatic exchange evaluation with $\alpha\beta$-approach," *ICGA Journal*, Vol. 33, 2010, pp. 3-17.
28. S. Akl and M. Newborn, "The principal continuation and the killer heuristic," in *ACM Annual Conference Proceedings*, 1977, pp. 466-473.
29. J. Schaeffer, "The history heuristic," *ICCA Journal*, Vol. 6, 1983, pp. 16-19.
30. G. M. Adelson-Velskiy, V. L. Arlazarov, and M. V. Donskoy, "Some methods of controlling the tree search in chess programs," *Artificial Intelligence*, Vol. 6, 1975, pp. 361-371.
31. E. A. Heinz, "Adaptive null-move pruning," *ICCA Journal*, Vol. 22, 1999, pp. 123-132.
32. J. Schaeffer, "Experiments in search and knowledge," Ph.D. Thesis, Department of Computing Science, University of Waterloo, Canada, 1986.
33. E. A. Heinz, "Extended futility pruning," *ICCA Journal*, Vol. 21, 1998, pp. 75-83.
34. T. Romstad, "An introduction to late move reductions," http://www.glaurungchess.com/lmr.html, 2012.
35. N. Krogius, A. Livsic, B. Parma, and M. Taimanov, *Encyclopedia of Chess Middlegames: Combinations*, Chess Informant, Belgrade, Serbia, 1980.

36. F. Reinfeld, *Win at Chess* (*Dover Books on Chess*), Dover Publications, NY, 2001.
37. F. Reinfeld, *1001 Winning Chess Sacrifices and Combinations*, Wilshire Book Company, CA, 1969.

Kunihito Hoki received his Ph.D. from the Graduate School of Science of Tohoku University in Miyagi, Japan. He is presently working in the Center for Frontier Science and Engineering at the University of Electro-Communications in Tokyo, Japan. His current research interests include computer games and machine learning.



Seiya Omori received his M.E. from the Department of Communication Engineering and Informatics at the University of Electro-Communications in Tokyo, Japan, in 2013. He intends to start working for Canon Inc. in April 2013.

Takeshi Ito has been associated with the University of Electro-Communications since 1994. He received his Ph.D. from Nagoya University graduated schools in Aichi, Japan in 1994. His research interests include human cognitive processes and learning processes in playing thinking games or solving difficult problems.