An Analysis of Voting Algorithm in Games

By Yuichiro Sato

A project paper submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Professor Hiroyuki Iida

March, 2013

An Analysis of Voting Algorithm in Games

By Yuichiro Sato (1110030)

A project paper submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Professor Hiroyuki Iida

and approved by
Professor Hiroyuki Iida
Associate Professor Kokoro Ikeda
Associate Professor Shinobu Hasegawa

February, 2013 (Submitted)

Acknowledgments

The author would like to express their appreciation to Prof. Iida who provided carefully considered feedbacks and helpful comments. The author would also like to thank Prof. Cincotti whose comments made enormous contribution to my work. I would like to express my gratitude to my family for their moral support and warm encouragements.

Contents

Chapter 1

Introduction

In game engine research, improving performance by multiple choice systems has been researched. In 1985 Althöfer started 3-Hirn with a seminal experiment in the game of chess [1]. 3-Hirn is a system such that "one or more programs compute a clear handful of candidate solutions and a human chooses amongst these candidates" [2]. In chess, the 3-Hirn consists of two different strong chess engines and one human weak chess player. When the human plays a game by choosing a move from moves which the engines suggest as candidates, his/her performance is improved and overcome the each individual engine. This result is surprising because if the weakest player chooses a move from moves which are suggested by the other stronger human players, the outcome is expected to be the opposite.

After Althöfer's delightful success, Obata et al. reported consultation algorithm where many game engines choose one move by simple majority rule, which improves the performance on Shogi game. This is "a method where a machine chooses a move automatically without human intervention" [3]. They also reported optimistic consultation algorithm [4]. The consultation algorithm adopts consultation between many individual engines. To make many engines, they apply noises on an evaluate function which BONANZA, a strong engine has. Also, they reported consultation of three strong Shogi programs: YSS, GPS, and BONANZA plays better games than any of the three individual programs.

This algorithm works well in other games like chess and Go [5, 6]. In 2010, AKARA, which is a game engine uses consultation algorithm defeated a top player in the Ladies Professional Players Group [7].

Even though the advantage of multiple choice systems is clear in practice, the reason why these systems work well is not clear. In 3-Hirn, both engines are stronger than the human who decides which candidate to play. Also, in some consultation algorithm, weak engines which are made from a strong original engine with random numbers suggest candidate moves. In other words, contribution of weak engines improves total output. This is a paradox. In this thesis, we report a reasonable explanation of this paradox.

In chapter 2, we introduce a mathematical representation of these systems, especially for consultation with random numbers. In chapter 3, we introduce an analysis of these systems. This is an extension of the discussion in [3] and a reasonable explanation of

the paradox. In chapter 4, we apply our representation to 3-Hirn and give a reasonable explanation for it. In chapter 5, we discuss our result. In chapter 6, we conclude the paper.

Chapter 2

Mathematical representation of the general consultation algorithm with majority rule

2.1 Mathematical representation of game engine and its votes

In this section we introduce a mathematical representation of consultation algorithm. Then, in the next chapter, we apply this representation to analyze experiments of consultation which are reported in [3]. This representation is enough powerful, therefore we are able to make a biggest framework for the analysis of consultation.

To analyze the consultation algorithm, the most important point of view is that an engine is a program which chooses a move in a position in a deterministic way, except if a random algorithm is adopted. In other words, if the same position appears many times during the game, the engine suggests the same move every time. This suggestion is fixed and never changes. Therefore, we can represent an engine as a mapping from positions to moves.

To treat positions and moves as numbers, an order for positions and moves is needed. This order indexes positions and moves. We do not need to specify this order. The only restriction is that this order needs to be total, *i.e.*, all the positions and moves are needed to be indexed in this order.

An example of desirable indexing is as follows. For positions, one can fix the blank position which has no marks nor pieces on the board as 0. Then, one changes the blank position by minimum changes which is describable by the language of games and makes a list of positions. Some positions are legal and the others might be illegal, we do not care about it. If the position is illegal for the game, the position never appears in real games. Therefore, it never affects to analysis of real games. One can index positions in the list as he/she want. Then, one can change positions in the list by minimum changes, and index again. In this way, one can index all the positions. This is the example of order for positions. For moves, almost the same strategy is available.

After indexing, we can represent an engine as a function from indexes of positions to indexes of moves. This is a function from natural numbers to natural numbers. Therefore, it is possible to find a continuous function that equals to the function of engine at natural numbers and has various values at the others. If a function is continuous, a derivative exists. This property is desirable in some cases. In this way, we can treat an engine as a function from natural numbers to natural numbers or from real numbers to real numbers.

Let us denote the set of all positions of a game as P and the set of all moves as M. These sets are indexed by natural numbers. An engine decides a move in each position, and this choice is expressed as deciding an index of move for each index of position, *i.e.*, a function. This representation of an engine makes our discussion clear and nothing important is missed for analysis. We represent an engine as a function from a set of natural numbers of 1 to $|P|$ to a set of natural numbers of 1 to $|M|$ in this thesis. We do not distinguish a set of positions and moves to a set of index of them in following discussion. There exist bijections, it is not necessary to distinguish them.

Also, an engine must have an evaluation function f. An evaluation function is a function which evaluates the advantage of a move in a position. If an engine does not have any evaluation function, it is impossible to decide which move is better. As a result, it has no choice and must return a random move. Random engines are not suitable for our purpose. Therefore, we omit systems which do not have any evaluation function.

The evaluation function in this thesis has a different meaning from usual using in artificial game engine research. Usually, an evaluation function decides an evaluate value of a position. Then, an engine calculates evaluate values of legal moves using a search technique. However, in this thesis, an evaluation function can evaluate moves in a position directly. This means it includes search process. If the search algorithm is deterministic, it choses the same move for the same position. Therefore, an evaluation of each move is static, we are able to describe this situation by a function.

An evaluation function is a mapping from Cartesian products of positions and moves to real numbers.

$$f : P \times M \to R \tag{2.1}$$

This mapping also becomes a function if positions and moves are indexed. Therefore, an evaluation function is a function which is defined on 2-dimensional lattice points. To evaluate each move chosen by engine, suppose there exists a perfect player, and let us denote its evaluation function as f^*. This perfect player perfectly evaluates all the moves in a position. Not only winning or losing moves, but also how easy to win or lose. The perfect player of course plays perfectly. Therefore, all positions are classified with 2 class, win or lose. However, real player is not perfect. It makes a mistake in some case. Therefore, if a move derives win for perfect player, real player could not follow the path to win as like as the perfect player. These effects are needed to be included.

When p is a position, m is a move and x is a real number.

$$f^*(p, m) = x \tag{2.2}$$

This f^* is used to calculate the exact advantage of the consultation algorithm. M includes the all moves that the game has, therefore m could be an illegal move in p. If it is, define

the evaluation value as the minimum. The discussion becomes clear if evaluations of illegal moves are 0 and evaluations of legal moves have positive value. However, one could need to use minus evaluation for some moves. In any case, the minimum evaluation is enough for illegal moves.

Now, we can start making the mathematical representation of the consultation algorithm. Let us denote $p \in P = \{1, 2, \cdots, |P|\}$ as the position in which the engine needs to play and $m \in M = \{1, 2, \cdots, |M|\}$ as the move which the engine chooses in that position, then the engine is defined as follows.

$$AI(p) = m \tag{2.3}$$

If you want an analytical function, use a polynomial function as like $AI(p) = \sum_{i=1}^{|P|} \lambda_i p^i = m$. It is possible to choose λ_i to mimic the target engine's decision, because this engine is deterministic. The important point is that this mathematical function returns completely the same move as a real engine which is written as a program. If there are n engines, let us denote them as AI_1, AI_2, $\cdots$, AI_n.

To analyze the consultation algorithm, we need to make a matrix as

$$M_{ij}(p) = AI_i(p) - AI_j(p) \tag{2.4}$$

$M_{ij}(p) = 0$ if and only if $AI_i(p) - AI_j(p) = 0$, *i.e.*, the matrix element is 0 if and only if corresponding engines choose the same move. To convert this matrix to an easy to use one, let us use the function $\delta(x)$ which returns 1 if $x = 0$ and 0 if $x \neq 0$. Then,

$$\begin{aligned} V_{ij}(p) &= \delta(M_{ij}(p)) \\ &= \begin{cases} 1 & (AI_i(p) = AI_j(p)) \\ 0 & (AI_i(p) \neq AI_j(p)) \end{cases} \end{aligned} \tag{2.5}$$

hence if $AI_i(p) = AI_j(p)$ then $M_{ij}(p) = 0$ and if $AI_i(p) \neq AI_j(p)$ then $M_{ij}(p) \neq 0$. Let us call this the voting matrix. Then,$\sum_{j=1}^{n} V_{ij}(p)$ is the number of engines who agree with AI_i. This is greater than or equals to 1, because AI_i always chooses the same candidate as AI_i.

Let us introduce an example of voting matrix. Suppose there exist 3 engines. Also, suppose $P = \{1, 2, 3, 4, 5\}$ and $M = \{1, 2, 3\}$ and AI_i are as follows.

$$AI_1(1) = 1 \tag{2.6}$$
$$AI_1(2) = 2 \tag{2.7}$$
$$AI_1(3) = 3 \tag{2.8}$$
$$AI_1(4) = 1 \tag{2.9}$$
$$AI_1(5) = 2 \tag{2.10}$$
$$AI_2(1) = 1 \tag{2.11}$$
$$AI_2(2) = 1 \tag{2.12}$$
$$AI_2(3) = 1 \tag{2.13}$$

$$AI_2(4) = 2 \tag{2.14}$$
$$AI_2(5) = 2 \tag{2.15}$$
$$AI_3(1) = 3 \tag{2.16}$$
$$AI_3(2) = 2 \tag{2.17}$$
$$AI_3(3) = 1 \tag{2.18}$$
$$AI_3(4) = 3 \tag{2.19}$$
$$AI_3(5) = 2 \tag{2.20}$$
$$\tag{2.21}$$

Then, the voting matrix becomes as follows.

$$\begin{pmatrix} V_{11}(1) & V_{12}(1) & V_{13}(1) \\ V_{21}(1) & V_{22}(1) & V_{23}(1) \\ V_{31}(1) & V_{32}(1) & V_{33}(1) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.22}$$

$$\begin{pmatrix} V_{11}(2) & V_{12}(2) & V_{13}(2) \\ V_{21}(2) & V_{22}(2) & V_{23}(2) \\ V_{31}(2) & V_{32}(2) & V_{33}(2) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \tag{2.23}$$

$$\begin{pmatrix} V_{11}(3) & V_{12}(3) & V_{13}(3) \\ V_{21}(3) & V_{22}(3) & V_{23}(3) \\ V_{31}(3) & V_{32}(3) & V_{33}(3) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \tag{2.24}$$

$$\begin{pmatrix} V_{11}(4) & V_{12}(4) & V_{13}(4) \\ V_{21}(4) & V_{22}(4) & V_{23}(4) \\ V_{31}(4) & V_{32}(4) & V_{33}(4) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.25}$$

$$\begin{pmatrix} V_{11}(5) & V_{12}(5) & V_{13}(5) \\ V_{21}(5) & V_{22}(5) & V_{23}(5) \\ V_{31}(5) & V_{32}(5) & V_{33}(5) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{2.26}$$

In the consultation algorithm, a weight vector $\vec{w}$ is used. This vector represents a priority of each engine. Heavily weighted engines have more priority than lightly weighted ones. For example, in the simple majority consultation algorithm, all the elements of weight vector are 1. In the consultation algorithm with a leader, the leader is weighted as 1.5 and the others are 1. In this way, voting vector $\vec{v}(p)$ is calculated as follows.

$$\begin{pmatrix} V_{11}(p) & V_{12}(p) & \dots & V_{1n}(p) \\ V_{21}(p) & V_{22}(p) & \dots & V_{2n}(p) \\ \vdots & \vdots & \ddots & \vdots \\ V_{n1}(p) & V_{n2}(p) & \dots & V_{nn}(p) \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} \tag{2.27}$$

The index of the max coordinate in $\vec{v}(p)$ represents the accepted engine in the consultation algorithm.

On the example above, suppose $\vec{w} = (1.5, 1, 1)$. Then, voting vector is as follows.

$$\vec{v}(1) = \begin{pmatrix} 2.5 \\ 2.5 \\ 1 \end{pmatrix} \tag{2.28}$$

$$\vec{v}(2) = \begin{pmatrix} 2.5 \\ 1 \\ 2.5 \end{pmatrix} \tag{2.29}$$

$$\vec{v}(3) = \begin{pmatrix} 1.5 \\ 2 \\ 2 \end{pmatrix} \tag{2.30}$$

$$\vec{v}(4) = \begin{pmatrix} 1.5 \\ 1 \\ 1 \end{pmatrix} \tag{2.31}$$

$$\vec{v}(5) = \begin{pmatrix} 3.5 \\ 3.5 \\ 3.5 \end{pmatrix} \tag{2.32}$$

You can see how consultations work in this example. In position 3, leader's suggestion is rejected by the others. Also, in position 4, leader's suggestion is accepted to resolve a conflict.

Of course, there is the case where two or more different candidates get the same number of votes. For example, in a 5 engines consultation, the leader is alone, 2 engines choose some candidate and the other 2 engines choose another candidate. In this case, we need to decide which candidate to play. Therefore, a conflict resolution is needed. To do so, resolution function r is used. Then, a consultation algorithm with majority rule is represented as follows

$$C(p) = AI_{r(\vec{v})}(p) \tag{2.33}$$

where

$$r(\vec{v}) = r\left(\begin{pmatrix} \sum_{j=1}^{n} w_1 \delta(AI_1(p) - AI_j(p)) \\ \sum_{j=1}^{n} w_2 \delta(AI_2(p) - AI_j(p)) \\ \vdots \\ \sum_{j=1}^{n} w_n \delta(AI_n(p) - AI_j(p)) \end{pmatrix}\right) \tag{2.34}$$

One example of r is random choose from candidates which have a conflict. This r is used in [3]. Another is to use the evaluation function f^s of the strongest engine. In this case, the index of engine which is used in a consultation algorithm is written as follows

$$Max[f^s(p, AI_k(p))] \tag{2.35}$$

where k is an index of v_k as $v_k \in \{v_j | \forall j.v_j \leq v_i\}$ and $v_i = \sum_{j=1}^{n} w_j V_{ij}(p)$.

2.2 Mathematical representation of the consultation algorithm

In the previous section, we finished the mathematical representation of the consultation algorithm with majority rule. Now, We analyze the conditions in which the consultation algorithm works well. Therefore, we need a reasonable and quantitative definition of working well. Otherwise, nobody can say the consultation algorithm works well compare to another algorithm. The following definition is one of the suitable definition.

Definition 1. *The consultation algorithm absolutely works well if and only if*

$$\forall i. \forall p. f^*_{AI_i}(p) \leq f^*_C(p)$$

where $f^*_{AI_i}(p) = f^*(p, AI_i(p))$ and $f^*_C(p) = f^*(p, C(p))$ and f^* is the evaluation function of the perfect player.

This definition is too strict in practice. What we expect is an average improvement. In other words, what we expect from the consultation algorithm is choosing a better move for most of the positions, but not necessarily for all of the positions. Then, the practical definition is as follows.

Definition 2. *The consultation algorithm works well under a distribution of positions D if and only if*

$$\forall i. \sum_{p=1}^{|P|} Pr(p) f^*_{AI_i}(p) < \sum_{p=1}^{|P|} Pr(p) f^*_C(p) \Leftrightarrow \forall i. Ave_D f^*_{AI_i}(p) < Ave_D f^*_C(p)$$

where $Pr(p)$ is the probability of occurrence of position p under distribution D and Ave_D is an average.

If the engine is stochastic, we need an extended definition as follows.

Definition 3. *The consultation algorithm is expected to work well under distribution of positions D and distribution of moves D' if and only if*

$$\forall i. \sum_{p=1}^{|P|} Pr(p) \sum_{m=1}^{|M|} \pi(p,m) f^*_{AI_i}(p) < \sum_{p=1}^{|P|} Pr(p) \sum_{m=1}^{|M|} \Pi(p,m) f^*_C(p)$$

$$\Leftrightarrow \forall i. Ave_D \sum_{m=1}^{|M|} \pi(p,m) f^*_{AI_i}(p) < Ave_D \sum_{m=1}^{|M|} \Pi(p,m) f^*_C(p)$$

$$\Leftrightarrow \forall i. Ave_D Ex[f^*_{AI_i}(p)] < Ave_D Ex[f^*_C(p)]$$

where $\pi(p,m)$ and $\Pi(p,m)$ are the probabilities that AI_i and C choose move m in position p under distribution D', and $Ex[\cdot]$ is an expectation value. The sum of m is taken on all moves, therefore it could contain illegal moves. If a move m is illegal in p, $\pi(p,m)$ and $\Pi(p,m)$ are 0. Every evaluation value has a specific value, therefore, if probability is 0, it does not affect the expectation value.

Proposition 1. *Suppose the consultation algorithm is expected to work well and engines contribute to the consultation are deterministic, then the consultation algorithm works well.*

Proof. If the consultation algorithm is expected to work well,

$$\forall i. Ave_D Ex[f^*_{AI_i}(p)] < Ave_D Ex[f^*_C(p)]. \tag{2.36}$$

Also, if engine is deterministic, $\pi(p, m)$ and $\Pi(p, m)$ is 1 for a move and 0 for the others in a position p. Therefore,

$$\forall i. Ave_D Ex[f^*_{AI_i}(p)] < Ave_D Ex[f^*_C(p)] = \forall i. Ave_D f^*_{AI_i}(p) < Ave_D f^*_C(p). \tag{2.37}$$

This is the definition of the consultation works well. □

Therefore, the definition of the consultation is expected to work well is a generalization of the definition of the consultation works well. Under these definitions, we continue to analyze the consultation algorithm in this thesis.

Chapter 3

Mathematical analysis of the consultation algorithm with random numbers

3.1 Introducing noise function

In the previous chapter we formed a mathematical representation of the consultation algorithm and defined when the consultation algorithm works well. They are explicit and formal, therefore they make discussions clear. Now, it is time to analyze reported experimental results and give a reasonable explanation of the consultation algorithm. We analyze consultation with random numbers as reported in [3]. This consultation is the most simple one and good target for the early study.

Obata et al. reported experiments of consultation algorithm with random numbers. In these experiments, they prepared many engines which are generated with random numbers in the evaluation function of the original strong Shogi engine, BONANZA. Let us denote the original engine as AI_O and derived noisy engine as AI'. Then, the difference of these engines defines a noise function N_O. Even though the explicit analytical expression is unknown, $N_O(p)$ is a function which represents the difference of the original engine and noisy one. In other words, a prepared engine is represented as follows.

$$AI'(p) = AI_O(p) + N_O(p) \tag{3.1}$$

$N_O(p)$ is a function which is generated by a fixed list of random numbers, once $N_O(p)$ is generated, it is fixed. Therefore, AI' is still deterministic. $N_O(p)$ depends on p and the original engine, because some positions or engines could be sensitive for noises, some could be not. If noises did not affect a move of engine in a position p, $N_O(p)$ is 0, else it is not 0 and changes the index of move to play.

If one makes n of engines which contribute to the consultation, let us denote them as $AI_i(p) = AI_O(p) + N_O^i(p)$. Each $N_O^i(p)$ is made from different random numbers, therefore

they are different each other. Then, a voting matrix becomes as follows

$$V(p) = \delta \left(\begin{pmatrix} N_O^1(p) - N_O^1(p) & N_O^1(p) - N_O^2(p) & \dots & N_O^1(p) - N_O^n(p) \\ N_O^2(p) - N_O^1(p) & N_O^2(p) - N_O^2(p) & \dots & N_O^2(p) - N_O^n(p) \\ \vdots & \vdots & \ddots & \vdots \\ N_O^n(p) - N_O^1(p) & N_O^n(p) - N_O^2(p) & \dots & N_O^n(p) - N_O^n(p) \end{pmatrix} \right) \tag{3.2}$$

because for each $AI_i(p)$, $AI_O(p)$ is common. As you see, the voting matrix depends only on noise functions. The original engine is common for all engines, therefore a difference $N_O^i(p)$ and $N_O^j(p)$ is only caused from random numbers which are used to create them. The probability of the same value a is shared in N_O^i and N_O^j are calculated as a product of the probability of $N_O^i = a$ and $N_O^j = a$. This is a constant $c(v)$. Therefore, the sum of all $c(v)$ is the probability of $AI_i(p)$ and $AI_j(p)$ choosing the same move. This is a constant C. Therefore, the expectation value of voting matrix is as follows.

$$Ex[V(p)] = \delta \left(\begin{pmatrix} 0 & C & \dots & C \\ C & 0 & \dots & C \\ \vdots & \vdots & \ddots & \vdots \\ C & C & \dots & 0 \end{pmatrix} \right) \tag{3.3}$$

N_O depends on positions and the original engine. This dependency is important for an improvement of performance. If N_O depends on the original engine, adopting several engines in consultations could affect the consultation result due to the difference of sensitivity of noises. Also, N_O depends on the position, and if it is easily affected when the original engine chooses a bad move, and not easily affected when it chooses a good move, noises could improve the consultation result.

3.2 Explicit solutions for the consultation algorithm with random numbers

In the experiments reported in [3], all engines which join in the consultation are weaker than the original engine.

$$\forall i. Ave_D f^*_{AI_i}(p) < Ave_D f^*_{AI_O}(p) \tag{3.4}$$

This equation means random noise makes the decision worse. This is reasonable, because random choice is not expected to be better than careful choice. Even though no engine is stronger than the original engine, the majority of them choose better candidates compared to the original one. This seems a paradox. However, there exists a reasonable explanation.

As definition, if consultation algorithm works well, the average of the evaluation values by the perfect player becomes better. In these experiments, consultation worked well. Therefore, the following condition must be satisfied.

$$\forall i. Ave_D f^*_{AI_i}(p) < Ave_D f^*_C(p) \tag{3.5}$$

Additionally, they reported that the consultation algorithm is stronger than the original engine. Therefore,

$$\forall i. Ave_D f^*_{AI_i}(p) < Ave_D f^*_{AI_O}(p) < Ave_D f^*_C(p) \tag{3.6}$$

must be satisfied. This equation could be satisfied for some noise function N^i_O. However, N^i_O are generated randomly. Therefore, whether consultation algorithm works well or not is stochastic in this case. Therefore, we need to treat probability explicitly.

Before discuss about it, we need a partition of M which is made according to a classification as follows.

$$\begin{aligned} b(p) &= \{m | f^*(p,m) > f^*_{AI_O}(p)\} && (3.7)\\ e(p) &= \{m | f^*(p,m) = f^*_{AI_O}(p)\} && (3.8)\\ w(p) &= \{m | f^*(p,m) < f^*_{AI_O}(p)\} && (3.9) \end{aligned}$$

In $b(p)$, the moves have a better evaluation value than the original value, in $e(p)$ and $w(p)$, they do not. This partition depends on p because better or worse is the only relative property. Illegal moves have the minimum as its evaluate value, therefore they are classified in $w(p)$. What we expect for the consultation algorithm is that the majority of noisy engines choose a candidate from $b(p)$.

At position p, engines which join in the consultation have three behaviors. One is to choose an equivalent candidate of the original engine. The other is to choose a better or worse candidate compared to the original one. Therefore, any stochastic change on AI_i by noise are classified as three types, *i.e.*, going on $b(p)$ or $w(p)$, and staying on $e(p)$.

Suppose there exists exact probabilities of AI_O changing its move from $AI_O(p)$ to m. Let us denote this probability as $\pi(p,m)$. We assume this probability is common for all AI_i. Then, from experiment which is reported in [3],

$$Ave_D Ex[f^*_{AI_i}(p)] < Ave_D Ex[f^*_{AI_O}(p)] \tag{3.10}$$

$$\Leftrightarrow Ave_D \sum_{m=1}^{|M|} \pi(p,m) f^*(p,m) < Ave_D f^*_{AI_O}(p) \tag{3.11}$$

M is divided into a partition by evaluation as $b(p)$, $e(p)$ and $w(p)$. Therefore, a summation on M is divided into a summation on $b(p)$, $e(p)$ and $w(p)$. Let us denote such a summation as $\sum^{b(p)}$, $\sum^{e(p)}$ and $\sum^{w(p)}$. Then,

$$\begin{aligned} \sum_{m=1}^{|M|} \pi(p,m) f^*(p,m) &= \sum^{b(p)} \pi(p,m) f^*(p,m) \\ &+ \sum^{e(p)} \pi(p,m) f^*(p,m) + \sum^{w(p)} \pi(p,m) f^*(p,m) \\ = \sum^{b(p)} \pi(p,m) f^*(p,m) &+ \sum^{e(p)} \pi(p,m) f^*_{AI_O}(p) + \sum^{w(p)} \pi(p,m) f^*(p,m) \end{aligned} \tag{3.12}$$

because in $e(p)$, $f^*(p,m) = f^*_{AI_O}(p)$.

Therefore, equation (3.11) is fixed as follows.

$$\begin{aligned} Ave_D \sum_{m=1}^{|M|} \pi(p,m) f^*(p,m) &< Ave_D f^*_{AI_O}(p) \\ \Leftrightarrow Ave_D \sum_{m=1}^{|M|} \pi(p,m) f^*(p,m) &< Ave_D \Big(\sum^{b(p)} \pi(p,m) f^*_{AI_O}(p) \\ &+ \sum^{e(p)} \pi(p,m) f^*_{AI_O}(p) + \sum^{w(p)} \pi(p,m) f^*_{AI_O}(p) \Big) \quad (3.13) \\ \Leftrightarrow Ave_D \sum_{m=1}^{b(p)} \pi(p,m) \{ f^*(p,m) &- f^*_{AI_O}(p) \} \\ &< Ave_D \sum_{m=1}^{w(p)} \pi(p,m) \{ f^*_{AI_O}(p,m) - f^*(p,m) \} \quad (3.14) \end{aligned}$$

because $\sum^{b(p)} \pi(m,p) + \sum^{e(p)} \pi(m,p) + \sum^{w(p)} \pi(m,p) = 1$ and Ave_D is linear. This equation means an average of an expected improvement on $b(p)$ is less than an average of an expected reduction in quality and is a reasonable condition of this experiment. Let us denote the expected improvement of AI', *i.e.*, expected improvement by changing AI_O to AI' as $Ex_i^{AI'}(p)$. Then,

$$Ex_i^{AI_i}(p) = \sum^{b(p)} \pi(p,m) \{ f^*(p,m) - f^*_{AI_O}(p) \} \quad (3.15)$$

Also, let us denote the expected reduction of AI', *i.e.*, expected reduction by changing AI_O to AI' as $Ex_r^{AI'}(p)$. Then,

$$Ex_r^{AI_i}(p) = \sum^{w(p)} \pi(p,m) \{ f^*_{AI_O}(p) - f^*(p,m) \} \quad (3.16)$$

The experimental condition becomes as follows

$$\forall i. Ave_D Ex_i^{AI_i}(p) < Ave_D Ex_r^{AI_i}(p) \quad (3.17)$$

Theorem 1.

$$\forall i. Ave_D Ex[f^*_{AI_i}(p)] < Ave_D Ex[f^*_{AI_O}(p)] \Leftrightarrow \forall i. Ave_D Ex_i^{AI_i}(p) < Ave_D Ex_r^{AI_i}(p)$$

Proof. As above. □

Let denote us the consultation algorithm with random numbers works well if and only if

$$\forall i. Ave_D Ex[f^*_{AI_i}(p)] < Ave_D Ex[f^*_{AI_O}(p)] < Ave_D Ex[f^*_C(p)] \quad (3.18)$$

This means that the consultation algorithm with random numbers works well when the consultation result is better than the original even though the all of engines are weaker than the original. If there is an engine who is stronger than the original, the improvement could be mainly came from the engine, no need to the consultation. If the consultation result is weaker than the original, then the consultation algorithm with random numbers just wasted resources. Therefore, this is the reasonable definition.

Theorem 2. *The consultation algorithm with random numbers works well if and only if*

$$\begin{cases} \forall i. Ave_D Ex_i^{AI_i}(p) < Ave_D Ex_r^{AI_i}(p) \\ Ave_D Ex_r^C(p) < Ave_D Ex_i^C(p) \end{cases}$$

Proof.

$$\begin{aligned} Ave_D Ex[f^*_{AI_O}(p)] &< Ave_D Ex[f^*_C(p)] \\ \Leftrightarrow Ave_D f^*_{AI_O}(p) &< Ave_D \sum_{m=1}^{|M|} \Pi(p,m) f^*_C(p) \end{aligned} \tag{3.19}$$

where $\Pi(p,m)$ is a probability of the consultation algorithm chooses a move m in position p.

$$\begin{aligned} Ave_D f^*_{AI_O}(p) &< Ave_D \sum_{m=1}^{|M|} \Pi(p,m) f^*_C(p) \\ \Leftrightarrow Ave_D f^*_{AI_O}(p) &< Ave_D \Big(\sum^{b(p)} \Pi(p,m) f^*(p,m) \\ &+ \sum^{e(p)} \Pi(p,m) f^*(p,m) + \sum^{w(p)} \Pi(p,m) f^*(p,m) \Big) \end{aligned} \tag{3.20}$$

because a summation on M is divided into $\sum^{b(p)}$, $\sum^{e(p)}$ and $\sum^{w(p)}$.

$$\begin{aligned} Ave_D f^*_{AI_O}(p) &< Ave_D \Big(\sum^{b(p)} \Pi(p,m) f^*(p,m) \\ &+ \sum^{e(p)} \Pi(p,m) f^*(p,m) + \sum^{w(p)} \Pi(p,m) f^*(p,m) \Big) \\ \Leftrightarrow Ave_D \sum^{w(p)} \Pi(p,m) \{ f^*_{AI_O}(p) &- f^*(p,m) \} \\ &< Ave_D \sum^{b(p)} \Pi(p,m) \{ f^*(p,m) - f^*_{AI_O}(p) \} \qquad (3.21) \\ \Leftrightarrow Ave_D Ex_r^C(p) &< Ave_D Ex_i^C(p) \qquad (3.22) \end{aligned}$$

because $f^*_C(p) = f^*_{AI_O}(p)$ in $e(p)$.

$$\forall i. Ave_D Ex[f^*_{AI_i}(p)] < Ave_D Ex[f^*_{AI_O}(p)] \Leftrightarrow \forall i. Ave_D Ex_i^{AI_i}(p) < Ave_D Ex_r^{AI_i}(p) \tag{3.23}$$

is proven in theorem 1. Therefore

$$\begin{cases} Ave_D Ex[f^*_{AI_O}(p)] < Ave_D Ex[f^*_C(p)] \\ \forall i. Ave_D Ex[f^*_{AI_i}(p)] < Ave_D Ex[f^*_{AI_O}(p)] \end{cases} \tag{3.24}$$

$$\Leftrightarrow \begin{cases} \forall i. Ave_D Ex_i^{AI_i}(p) < Ave_D Ex_r^{AI_i}(p) \\ Ave_D Ex_r^C(p) < Ave_D Ex_i^C(p) \end{cases} \tag{3.25}$$

□

This theorem suggests when the consultation algorithm works well.

Theorem 3. *If $n-1 \leq |M|$, the lower bound of the consultation algorithm chooses a better move, $\Pi_b(p,m)$ and the upper bound of the consultation algorithm chooses a worse move, $\Pi_w(p,m)$ is as follows*

$$\Pi_b(p,m) = \sum_{i=2}^{n} \binom{n}{i} \pi(p,m)^i (1-\pi(p,m))^{n-i} - \sum_{i=2}^{\lfloor n/2 \rfloor} \binom{n}{i} \pi(p,m)^i (1-\pi(p,m))^{n-i}$$
$$\cdot \sum_{j=i}^{n-i} \binom{n-i}{j} \Big(\sum_{k=1}^{|M|} \pi(p,k)^j (1-\pi(p,k)-\pi(p,m))^{n-i-j}$$
$$-\pi(p,m)^j (1-2\pi(p,m))^{n-i-j} \Big)$$
$$\Pi_w(p,m) = \sum_{i=1}^{n} \binom{n}{i} \pi(p,m)^i (1-\pi(p,m))^{n-i} - \sum_{i=1}^{\lfloor n/2 \rfloor} \binom{n}{i} \pi(p,m)^i (1-\pi(p,m))^{n-i}$$
$$\cdot \sum_{j=i}^{n-i} \binom{n-i}{j} \Big(\sum_{k=1}^{|M|} \pi(p,k)^j (1-\pi(p,k)-\pi(p,m))^{n-i-j}$$
$$-\pi(p,m)^j (1-2\pi(p,m))^{n-i-j} \Big)$$

Proof. If consultation algorithm with random numbers chooses a move, the move needs to be the majority of the candidates. If more than half of engines choose the same candidate, the candidate is chosen as a move in any case. Moreover, if some move is chosen by relatively many engines it is majority. Therefore, the sum of these two is the probability of a move being chosen by consultation.

The probability of a better move $m \in b(p)$ being chosen by more than half of the engines is as follows.

$$\sum_{i=\lfloor n/2 \rfloor + 1}^{n} \binom{n}{i} \pi(p,m)^i (1-\pi(p,m))^{n-i} \tag{3.26}$$

The probability of relatively many engines choosing the better move is the sum of the product of two probabilities. One is the probability of i engines choosing a move m. The other is the probability of j such that $i < j$ engines not choosing the same move. If $n-1 \leq |M|$, there exists at least a situation such that $n-i$ engines do not make any group which has greater than i as its size. If $|M| < n-1$, this is not always true. The probability of i engines choosing a move m is as follows.

$$\binom{n}{i} \pi(p,m)^i (1-\pi(p,m))^{n-i} \tag{3.27}$$

The probability of j such that $i < j$ engines not choosing the same move equals to 1 minus the sum of probability of j engines choose the same move except for m. The

probability of j engines in $n-i$ engines choosing the same move except for m is as follows.

$$\binom{n-i}{j}\Big\{\sum_{k=1}^{|M|}\pi(p,k)^j(1 \quad - \quad \pi(p,k)-\pi(p,m))^{n-i-j} \\ - \quad \pi(p,m)^j(1-2\pi(p,m))^{n-i-j}\Big\} \tag{3.28}$$

Therefore, the probability of more than i engines in $n-i$ engines choose the same move except for m is as follows.

$$1-\sum_{j=i}^{n-i}\binom{n-i}{j}\Big\{\sum_{k=1}^{|M|}\pi(p,k)^j(1 \quad - \quad \pi(p,k)-\pi(p,m))^{n-i-j} \\ - \quad \pi(p,m)^j(1-2\pi(p,m))^{n-i-j}\Big\} \tag{3.29}$$

Therefore, the probability of relatively many engines choosing the better move m is

$$\sum_{i=2}^{\lfloor n/2\rfloor}\binom{n}{i}\pi(p,m)^i(1-\pi(p,m))^{n-i} \\ \cdot\Big\{1-\sum_{j=i}^{n-i}\binom{n-i}{j}\Big(\sum_{k=1}^{|M|}\pi(p,k)^j(1 \quad - \quad \pi(p,k)-\pi(p,m))^{n-i-j} \\ - \quad \pi(p,m)^j(1-2\pi(p,m))^{n-i-j}\Big)\Big\} \tag{3.30}$$

The summation of equation 3.26 and 3.30

$$\Pi_b(p,m)=\sum_{i=\lfloor n/2\rfloor+1}^{n}\binom{n}{i}\pi(p,m)^i(1 \quad - \quad \pi(p,m))^{n-i} \\ +\sum_{i=2}^{\lfloor n/2\rfloor}\binom{n}{i}\pi(p,m)^i(1-\pi(p,m))^{n-i} \\ \cdot\Big\{1-\sum_{j=i}^{n-i}\binom{n-i}{j}\Big(\sum_{k=1}^{|M|}\pi(p,k)^j(1 \quad - \quad \pi(p,k)-\pi(p,m))^{n-i-j} \\ - \quad \pi(p,m)^j(1-2\pi(p,m))^{n-i-j}\Big)\Big\}$$

$$=\sum_{i=2}^{n}\binom{n}{i}\pi(p,m)^i(1-\pi(p,m))^{n-i} \quad - \quad \sum_{i=2}^{\lfloor n/2\rfloor}\binom{n}{i}\pi(p,m)^i(1-\pi(p,m))^{n-i} \\ \cdot\sum_{j=i}^{n-i}\binom{n-i}{j}\Big(\sum_{k=1}^{|M|}\pi(p,k)^j(1 \quad - \quad \pi(p,k)-\pi(p,m))^{n-i-j} \\ - \quad \pi(p,m)^j(1-2\pi(p,m))^{n-i-j}\Big) \tag{3.31}$$

is the lower bound of $m \in b(p)$ is chosen by consultation in p. We eliminated the case such that the same number of engines choose a better move and another worse move.

In this case, conflict resolution needed. For example, in the experiments in [3], this is done by randomly choosing a move from the candidates. Therefore, in such a case, the better move is not necessarily chosen. This is the reason why we omit this case. Hence, $\Pi_b(p,m)$ is the lower bound of the probability of the consultation algorithm choosing a better move.

In the same context, the upper bound of $m \in w(p)$ is chosen by consultation at p can be calculated. The probability of a worse move $m \in w(p)$ being chosen by more than half of the engines is as follows.

$$\sum_{i=\lfloor n/2 \rfloor+1}^{n} \binom{n}{i} \pi(p,m)^i (1-\pi(p,m))^{n-i} \tag{3.32}$$

The probability of relatively many engines choosing the better move is the sum of the product of two probabilities. One is the probability of i engines choosing a move m. The other is the probability of j such that $i < j$ engines not choosing the same move.

The probability of i engines choosing a move m is as follows.

$$\binom{n}{i} \pi(p,m)^i (1-\pi(p,m))^{n-i} \tag{3.33}$$

The probability of j engines in $n-i$ engines choosing the same move except for m is as follows.

$$\begin{aligned} \binom{n-i}{j} \Big\{ \sum_{k=1}^{|M|} \pi(p,k)^j (1 \quad &- \quad \pi(p,k) - \pi(p,m))^{n-i-j} \\ &- \quad \pi(p,m)^j (1-2\pi(p,m))^{n-i-j} \Big\} \end{aligned} \tag{3.34}$$

Therefore, the probability of more than i engines in $n-i$ engines choose the same move except for m is as follows.

$$\begin{aligned} 1 - \sum_{j=i}^{n-i} \binom{n-i}{j} \Big\{ \sum_{k=1}^{|M|} \pi(p,k)^j (1 \quad &- \quad \pi(p,k) - \pi(p,m))^{n-i-j} \\ &- \quad \pi(p,m)^j (1-2\pi(p,m))^{n-i-j} \Big\} \end{aligned} \tag{3.35}$$

Therefore, the probability of relatively many engines choosing the worse move m is

$$\begin{aligned} &\sum_{i=1}^{\lfloor n/2 \rfloor} \binom{n}{i} \pi(p,m)^i (1-\pi(p,m))^{n-i} \\ &\cdot \Big\{ 1 - \sum_{j=i}^{n-i} \binom{n-i}{j} \Big(\sum_{k=1}^{|M|} \pi(p,k)^j (1 \quad - \quad \pi(p,k) - \pi(p,m))^{n-i-j} \\ &\qquad\qquad - \quad \pi(p,m)^j (1-2\pi(p,m))^{n-i-j} \Big) \Big\} \end{aligned} \tag{3.36}$$

We counted all situations which has a conflict into this probability. In other words, in the conflict resolution, we choose the worse move. Therefore, the obtained probability is the upper bound. The total probability is as follows.

$$\Pi_w(p,m) = \sum_{i=\lfloor n/2\rfloor+1}^{n} \binom{n}{i}\pi(p,m)^i(1 \quad - \quad \pi(p,m))^{n-i}$$
$$+ \sum_{i=1}^{\lfloor n/2\rfloor} \binom{n}{i}\pi(p,m)^i(1-\pi(p,m))^{n-i}$$
$$\cdot\Big\{1 - \sum_{j=i}^{n-i}\binom{n-i}{j}\Big(\sum_{k=1}^{|M|}\pi(p,k)^j(1 \quad - \quad \pi(p,k)-\pi(p,m))^{n-i-j}$$
$$- \quad \pi(p,m)^j(1-2\pi(p,m))^{n-i-j}\Big)\Big\}$$

$$= \sum_{i=1}^{n}\binom{n}{i}\pi(p,m)^i(1-\pi(p,m))^{n-i} \quad - \quad \sum_{i=1}^{\lfloor n/2\rfloor}\binom{n}{i}\pi(p,m)^i(1-\pi(p,m))^{n-i}$$
$$\cdot\sum_{j=i}^{n-i}\binom{n-i}{j}\Big(\sum_{k=1}^{|M|}\pi(p,k)^j(1 \quad - \quad \pi(p,k)-\pi(p,m))^{n-i-j}$$
$$- \quad \pi(p,m)^j(1-2\pi(p,m))^{n-i-j}\Big) \tag{3.37}$$

is the upper bound of $m \in w(p)$ is chosen by consultation in p. □

These probabilities are useful to calculate the lower bound of expected improvement and the upper bound of expected reduction. $Ave_D Ex_r^C(p) < Ave_D Ex_i^C(p)$ is satisfiable under $\forall i. Ave_D Ex_i^{AI_i}(p) < Ave_D Ex_r^{AI_i}(p)$. There exists such situations and it is easy to find them by numerical trial and error.

Theorem 4. *If $|M| \leq n-1$, the lower bound of the consultation algorithm chooses a better move, $\Pi_b(p,m)$ and the upper bound of the consultation algorithm chooses a worse move, $\Pi_w(p,m)$ is as follows*

$$\Pi_b(p,m) = \sum_{i=\lfloor n/2\rfloor+1}^{n}\binom{n}{i}\pi(p,m)^i(1-\pi(p,m))^{n-i} + \sum_{i=\lfloor\frac{n-1}{|M|}\rfloor+2}^{\lfloor n/2\rfloor}\binom{n}{i}\pi(p,m)^i(1-\pi(p,m))^{n-i}$$
$$\cdot\Big\{1-\sum_{j=i}^{n-i}\binom{n-i}{j}\Big(\sum_{k=1}^{|M|}\pi(p,k)^j(1-\pi(p,k)-\pi(p,m))^{n-i-j}$$
$$-\pi(p,m)^j(1-2\pi(p,m))^{n-i-j}\Big)\Big\}$$
$$\Pi_w(p,m) = \sum_{i=\lfloor n/2\rfloor}^{n}\binom{n}{i}\pi(p,m)^i(1-\pi(p,m))^{n-i} + \sum_{i=\lfloor\frac{n-1}{|M|}\rfloor+1}^{\lfloor n/2\rfloor}\binom{n}{i}\pi(p,m)^i(1-\pi(p,m))^{n-i}$$
$$\cdot\Big\{1-\sum_{j=i}^{n-i}\binom{n-i}{j}\Big(\sum_{k=1}^{|M|}\pi(p,k)^j(1-\pi(p,k)-\pi(p,m))^{n-i-j}$$
$$-\pi(p,m)^j(1-2\pi(p,m))^{n-i-j}\Big)\Big\}$$

Proof. If $|M| \leq n-1$, in some case, it is impossible for engines to make relatively majority group. When i engines choose a move, the others can not make a group greater than i as its size. Therefore, $n-i$ engines are needed to be classified in a group less than i as its size for $\Pi_b(p,m)$. A move m is chosen by i engines, therefore, $|M|-1$ moves are leave for the others. If too less moves are given for $n-i$ engines, it is impossible to satisfy the condition. There are $n-i$ engines and the number of empty room for them is $(|M|-1)(i-1)$, therefore limit of i is as follows.

$$n-i = (|M|-1)(i-1)+1 \Leftrightarrow i = \frac{n-1}{|M|}+1 \tag{3.38}$$

Otherwise, there exists a group which size is greater than i. Therefore, if $|M| \leq n-1$, the sum of equation 3.30 is needed to be taken from $\lfloor \frac{n-1}{|M|} \rfloor + 2$ for $\Pi_b(p,m)$ and $\lfloor \frac{n-1}{|M|} \rfloor + 1$ for $\Pi_w(p,m)$. □

These results have no contradiction with the result which is reported in [3]. Suppose there exists only a better move and a worse move, then $M = \{1,2\}$. Let us denote a better move as 1 and a worse move as 2. If more than 3 game engines are in the consultation, it is impossible for relatively many engines to choose the better move. Therefore, the probability of the better move is chosen in consultation algorithm is

$$\sum_{i=\lfloor n/2 \rfloor+1}^{n} \binom{n}{i} \pi(p,1)^i (1-\pi(p,1))^{n-i} \tag{3.39}$$

This is equivalent to the equation which is reported in [3] and a main improvement from our previous work [8].

Chapter 4

Reasonable explanation for 3-Hirn

3-Hirn is a system which is made of two engines and a human [1, 2]. In this system, the human chooses a better candidate from candidates which are suggested from engines. The human is weaker than each engine in the game, but this system is stronger than each engine. The weak player's contribution improves the final outcome. It looks unreasonable that this system becomes stronger than any individual engines. However, there is a reasonable explanation of this puzzle in almost the same context as consultation algorithm, as follows.

Let us denote two engines in this system as AI_1 and AI_2 and the evaluation function of the human as f^h. Then the 3-Hirn system is written as follows

$$H_3(p) = \begin{cases} AI_1(p) & (f^h_{AI_1}(p) - f^h_{AI_2}(p) > 0) \\ AI_2(p) & (f^h_{AI_1}(p) - f^h_{AI_2}(p) \leq 0) \end{cases} \tag{4.1}$$

If 3-Hirn works well, the following condition is satisfied

$$\forall i. Ave_D f^*_{AI_i}(p) < Ave_D f^*_{H_3}(p) \tag{4.2}$$

where i is 1 or 2.

This condition is satisfiable. An engine has an evaluation function to improve its search speed. This function tends to be rough compared to the one a human has. Even though the engine has a rough evaluation function, the computer is really powerful, so it is able to cover the weakness by making a vast search tree. For human, the situation is opposite. A strong human player has a really good evaluation function and cuts the search tree efficiently.

Mathematically, this situation is represented as follows

$$Ave_D|f^h(p,m) - f^*(p,m)| < Ave_D|f^{AI_i}(p,m) - f^*(p,m)| \tag{4.3}$$

for almost all of $m \in M$.

This situation is satisfiable. An evaluation function is a function from Cartesian product of positions and moves to real numbers. This means it only depends on the current position, not on search depth. However the strength of a player depends on an evaluation

function and search depth. Therefore, if the human is good at evaluation but not for search, and engines are oppositely good at search but not for evaluation, the situation could arise.

If the above equations are satisfied, the human can choose the best candidate at a glance from candidates which engines found by wide and deep search effort. In other word, this system works well "By combining the gifts and strengths of humans and machines in appropriate ways" as Althöfer mentioned in [1]. This is the reason why 3-Hirn works well.

Chapter 5

Discussion

Now, we have an analysis and reasonable explanation for consultation algorithm with random numbers and 3-Hirn. There exists many situations in which consultation works well. However, it is difficult to derive a property which is common for all cases. Everything depends on the game and the engine. Therefore, consultation algorithm with random numbers is not a general solution. The domain which is available to consultation is limited. A clear explanation of 3-Hirn is obtained in the same context.

The consultation algorithm with random numbers is formally explained by our approach, but we have another type of consultation, *i.e.*, optimistic consultation. Unfortunately, it is impossible to find the origin of an advantage of optimistic consultation in this approach. To analyze optimistic consultation algorithm, we need to analyze detailed structures of an evaluation function. This is difficult.

Also, consultation of different types of engines are reported [3]. This is a slightly different situation from our analysis. It is possible to apply our analysis for this case. This consultation makes a large improvement on the original engine. No longer using a noisy weak engine but different type of engine. Therefore, it could be possible to approximate this consultation using our analysis.

We conclude that the effectiveness of the consultation algorithm depends on the game. In our representation, a game has positions and moves, *i.e.*, current states and decisions to make. Therefore, some human activity is included in our analysis. For example, the case in which one human considers his idea from many points of view has an analogy with our analysis. To get different point of view, one needs to come up with options which is not the first choice. This means generating worse options under his evaluation. If his evaluation is close enough to perfect, this is the same as adding random noise to his thinking. Consultation with many humans could also be approximated by our approach. Solomon describes a result of social epistemology as "If group deliberation does take place, outcomes are better when members of the group are strangers, rather than colleagues or friends." [9]. By taking a group of friends, and adding randomness to the decision process, a group of strangers is formed. This is a case of human activity to which our approach is applicable.

Chapter 6

Conclusions

In this thesis, we clearly explained the origin of an advantage of consultation algorithm with random numbers, and 3-Hirn. The consultation algorithm with random numbers works well if and only if the expected improvement of consultation is greater than the expected reduction of consultation and the expected improvement of each engine is less than the expected reduction of each engine. In this thesis a new definition is derived of the necessary and sufficient condition for consultation algorithm working well. This new definition is an improvement of the existing one because it considers all choices. A explanation of 3-Hirn is given, elaborating on the existing one. Formulas for bounds of probability of consultation algorithm improving/reducing are given. Our analysis has a possibility to be applicable to human activities not only engine.

Bibliography

[1] Althöfer I., and Snatzke, G.R.: Playing Games with Multiple Choice Systems. Lecture Notes in Computer Science 2883, pp. 142-153 (2003)

[2] Althöfer I.: Improved game play by multiple computer hints. Theoretical Computer Science 313, 315-324 (2004)

[3] Obata, T., Sugiyama, T., Hoki, K., and Ito, T.: Consultation Algorithm for Computer Shogi Move Decisions by Majority. Lecture Notes in Computer Science 6515, pp. 156-165 (2011)

[4] Sugiyama, T., Obata, T., Hoki, K., and Ito, T.: Optimistic Selection Rule Better Than Majority Voting System. Lecture Notes in Computer Science 6515, pp. 166-175 (2011)

[5] Omori, S., Hoki, K., and Ito, T.: Consultation Algorithm for Computer Chess. SIG Technical Reports 2011-GI-26(5), 1-7 (2011)

[6] Manabe, K., and Muramatsu, M.: Boosting Approach for Consultaton by Weighted Majority Vote in Computer Go. IPSJ Symposium Series 2011 6, 128-134 (2011)

[7] 清水市代女流王将 vs. あから 2010 速報, http://www.ipsj.or.jp/50anv/shogi/20101012.html (News:a top player in the Ladies Professional Players Group vs. AKARA 2010)

[8] Sato, Y., Cincotti, A., and Iida, H.: An Analysis of Voting Algorithm in Games. Computer Games Workshop at ECAI 2012, 102-113 (2012)

[9] Solomon, M.: Groupthink versus The Wisdom of Crowds: The Social Epistemology of Deliberation and Dissent. The Southern Journal of Philosophy 44, 28-42 (2006)