A submission for *ICS 2002:* C. Workshop on Algorithms and Computational Molecular Biology

An Optimal Strategy for 2×n AB Games

Shan-Tai Chen[1], and Sheng-Hsuan Hsu[2], Shun-Shii Lin[3]

[1]*Department of Information and Computer Education,*
National Taiwan Normal University, Taipei, Taiwan, R.O.C.

[2]*Department of Information Management,*
Lunghwa Univeristy of Science and Technology, Taoyuan, Taiwan.

[3]*Graduate Institute of Computer Science and Information Engineering,*
National Taiwan Normal University,
No. 88, Sec. 4, Ting-Chow Rd.,
Taipei, Taiwan, R.O.C.
Tel: 886-2-29322411,29322421
Fax:886-2-29322378
Email: linss@csie.ntnu.edu.tw

Abstract

This paper presents new and systematic strategies for *2×n AB* games. We invent a graphic model to represent the game-guessing process. From this representation, we find some symmetric and recursive structures in the process. This not only reduces the size of the search space but also helps us to derive the optimum strategies more efficiently. By using this novel approach, we develop optimal strategies for *2×n AB games* in the expected and worst cases, and are able to derive the following new results:

(1) $\lceil n/2 \rceil+1$ guesses are necessary and sufficient for 2×n AB games in the worst case.

(2) The minimum number of guesses required for 2×n AB games in the expected case is $(4n^3+21n^2-76n+72)/12n(n-1)$ if n is even, and is $(4n^3+21n^2-82n+105)/12n(n-1)$ if n is odd.

AMS Subject Classification: 68W05, 68Q25, 68W25

Key Words: AB game, Algorithms, Game tree, Mastermind, Search strategies.

Corresponding author: E-mail address: linss@csie.ntnu.edu.tw (Shun-Shii Lin).

1. Introduction

The game of Mastermind is a deductive game for 2 players a codemaker and a codebreaker. The codemaker chooses a secret code consisting of four pegs out of six possible colors. Repeated colors are allowed, so the set of possible codes is 6^4=1296. The codebreaker will then try to guess the code. After each guess, the codemaker responds with a hint that consists of black and white pegs; a black peg means that a peg in the codebreaker's guess is correct in both position and color; a white peg means that a peg in the guess is correct in color but not in position; and finally, no pegs means that there are pegs in the guess which are of no correct color. The purpose of the game is to solve the code (i.e. get four black pegs) in the smallest number of guesses.

Over the past three decades, there has been much research about this kind of games. Knuth [1] demonstrated a strategy for the Mastermind game that requires at most five guesses in the worst case and 4.478 in the expected case. The strategy used in [1] is to choose the guess that minimizes the maximum number of remaining possibilities at every stage. Later, Irving [3] and Nerwirth [4] used sophisticated heuristic strategies to improve the bounds in the expected case to 4.369 and 4.364 respectively. Finally, Koyama and Lai [2] used a recursive backtracking method to determine the Mastermind's optimal strategy, for which the expected number of guesses is 4.34. Also, variants of the Mastermind game have been studied by [6], [10], and [11]. Furthermore, in [8] and [9], they used evolutionary algorithms and genetic algorithms for solving the related problems. More recently, Roche [7] analyzed the generalized Mastermind and got asymptotical bounds under some conditions. Kabatianski and Thorpe [5] investigated the Mastermind game and its related applications based on coding theorem. However, because the complexity of the Mastermind game grows at an exponential rate, no optimal strategy for the Mastermind game with higher dimensions (i.e. the numbers of digits and colors are more than 4 and 6 respectively) has yet been found.

Another well-known deductive game in England and Asia, called " Bulls and Cows"[1] or AB game, is a variant of the Mastermind game. The difference is that all digits of the code in the game must be distinct; but any digits 0 through 9 are allowed. Hence the set of possible codes is the number of permutations P(10,4)=10*9*8*7=5040. Now we restate the game with more precise description. The codemaker chooses a secret code (s_1, s_2, s_3, s_4). After each guess (g_1, g_2, g_3, g_4) by codebreaker, the codemaker responds with a pair of numbers [A, B], where A is the number of "direct hits," i.e., the number of positions j such that $s_j=g_j$ and B is the number of "indirect hits," i.e., the number of positions j such that s_j g_j but $s_j=g_k$ for some position k j. For example, if the secret code is (1, 2, 3, 4) and the guesses are (3, 1, 5, 4) and (3, 1, 4, 5), then the responses are [1, 2] and [0, 3], respectively. The goal of the codebreaker is, based on the responses, to minimize the number of guesses needed and to find the secret code.

For describing and comparing the variants of these games, we briefly introduce the notations defined in [6]. The Mastermind game is denoted MM4×6, signifying four digits and six symbols with repetition of symbols allowed. The AB game is denoted MM4×10N, signifying four digits and ten

symbols with repetition of symbols prohibited. Likewise, MM2×nN signifies two digits and arbitrary n symbols with repetition of symbols prohibited.

In this paper, we develop systematic methodology to discover the optimal strategies for general AB games with 1 digit and 2 digits—1×n and MM2×nN games. This paper is organized as follows. In Section 2, we introduce some properties of game trees and use binary search technique to determine the optimum strategy for the simple 1×n game. In Section 3 we present the graphic model by means of a MM2×5N game. The optimal strategy for 2×n AB games is developed in Section 4. Section 5 contains our concluding remarks.

2. An optimum strategy for 1×n deductive games

In this section, we deal with a simple example of deductive games, 1×n game. By means of some properties of game trees, we will show how to determine the minimum numbers of guesses required both in the expected and the worst case for the game. By this comparatively simple work, we present some fundamental concepts that can be applied to develop optimal strategy for 2×n AB games in Section 3 and Section 4.

In 1×n deductive games, the codemaker chooses a *secret number S*, $S \in \{0, 1, 2, \ldots, n-1\}$. After each guess G_i by the codebreaker, the codemaker responds with a hint H_i , $H_i \in \{<, =, >\}$, three elements in which refer to $S<G_i$, $S=G_i$, and $S>G_i$, respectively. The goal of the codebreaker is, based on the hints, to minimize the number of guesses required and to find the secret number. Obviously, the guessing process for this game can be translated to a search problem. We can obtain the optimum strategy for this game by using *binary search* technique, which is shown in Theorem 1. In order to demonstrate how to calculate the number of guesses required in the worst and expected cases for 1×n games, we illustrate our strategy by way of a 1×16 game, the game tree for which is shown in Figure 1. Notice that the game trees built by binary search techniques are always *full binary trees*.

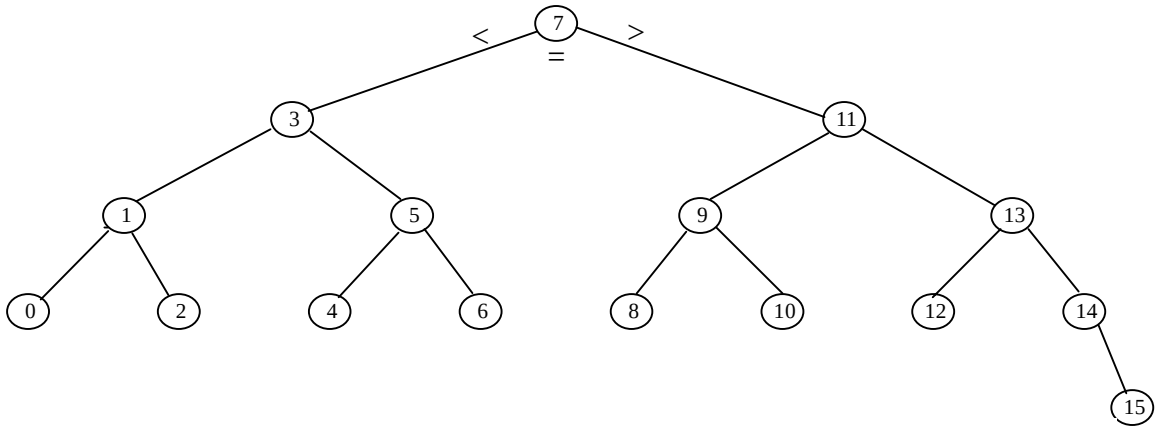

Figure 1. A game tree for 1*16 game where the binary search strategy is used.

In Figure 1, the number in each node represents each guess G_i. We start by comparing the secret

number S to the middle key of the possible secret numbers, which is 7 in the root in this case. According to the hint given by the codemaker, we decide which subtree should be chosen to guess next, and the same procedure can be used again until S is *hit*. More precisely, if the secret code S equals to G_i, then we *hit* the code and the game is finished. If $S<G_i$, then our strategy follows the left subtree; similarly if $S>G_i$, the right subtree is used. For example, if S=5, our guessing sequence will be 7, 3, and then 5 because S<7, S>3, and S=5, where 3 guesses are required to hit the secret number. On the other hand, if S=15, then our guessing sequence will be 7, 11, 13, 14, and then 15, where 5 guesses are required to finish the game. By doing so in Figure 1, we can easily obtain the following 2 Observations.

Observation 1. The number of guesses required in the worst case for a game is H+1, where H is the height of the game tree, i.e., the length of a longest path from the root to a leaf in the game tree. For example, H=4 in Figure 1.

Observation 2. The number of guesses required in the expected case for a game is D/n+1, where D is the *total path length* of the game tree; i.e., the sum of distances from the root to each node in the game tree. For example, in Figure 1 D=0*1+1*2+2*4+3*8+4*1=38 and the number of guesses required in the expected case is D/n+1=38/16+1=3.375.

In Theorems 1 and 2, we will demonstrate an optimum strategy based on binary search techniques and derive the minimum numbers of guesses required in the worst case and expect case. Before that, we look at a property [13] about minimum total path length in a binary tree, which is important for us to prove Theorem 1, as shown in Lemma 1.

Lemma 1. A binary tree has minimum total path length if and only if its external nodes all occur on at most two adjacent levels.

Proof [13]: Omitted. ■

Theorem 1. $\lfloor \log_2 n \rfloor+1$ guesses are necessary and sufficient for 1×n games in the worst case.

Proof. The necessary and sufficient conditions can be proven by the following two facts: Firstly, the game trees built by binary search techniques are always *full*, which means that the games trees always have minimum height. Secondly, the height of a *full* binary tree with n nodes is $\lfloor \log_2 n \rfloor$. By Observation 1, we obtain the result. ■

Theorem 2. The minimum number of guesses required for 1×n games in the expected case is
$((n+1)(\lfloor \log_2 n \rfloor-1)+2)/n+1$, if $n=2^k-1$, where $k\in Z$; and is
$((n+1)\lfloor \log_2 n \rfloor-2^{\lfloor \log_2 n \rfloor+1}+2)/n+1$, if $n\neq 2^k-1$, where $k\in Z$.

Proof: *Necessary.* Since the game trees built by binary search techniques are always *full*, which means that the external nodes of the game trees all occur on at most two adjacent levels. Therefore, by Lemma 1, the necessary condition can be proven.

Sufficient. We divide into two cases to prove this property.

Case 1. If $n=2^k-1$, then the game tree for the game is a *complete binary tree*. In this case $D = \sum_{i=1}^{\lfloor \log_2 n \rfloor} i * 2^i = (\lfloor \log_2 n \rfloor-1)*2^{(\lfloor \log_2 n \rfloor+1)}+2=(n+1)(\lfloor \log_2 n \rfloor-1)+2$. By Observation 2, the number of guesses required is $((n+1)(\lfloor \log_2 n \rfloor-1)+2)/n+1$.

Case 2. If $n \neq 2^k-1$, then the game tree for the game is full but not complete. The number of nodes at the bottom level is only $n-(2^{\lfloor \log_2 n \rfloor}-1)$ rather than $2^{\lfloor \log_2 n \rfloor}$ in a complete binary tree, and the distance from the root to each node in this level equals $\lfloor \log_2 n \rfloor$. Therefore, we should subtract the quantity $\lfloor \log_2 n \rfloor*(2^{\lfloor \log_2 n \rfloor}-(n-(2^{\lfloor \log_2 n \rfloor}-1))/n=\lfloor \log_2 n \rfloor*(2^{\lfloor \log_2 n \rfloor+1}-n-1)/n$ from the formula obtained in Case 1. That is, $D= (n+1)(\lfloor \log_2 n \rfloor-1)+2-(\lfloor log_2 n \rfloor*(2^{\lfloor log_2 n \rfloor+1}-n-1))= (n+1) \lfloor \log_2 n \rfloor -2^{\lfloor \log_2 n \rfloor+1}+2$. This completes the proof. ■

We apply the results of Theorems 1 and 2 to 1*15 game, the game tree for which is complete, and 1*16 game, the game tree for which is not complete. For 1*15 game, the numbers of guesses required in the worst and expected cases are 4 and $((15+1)(\lfloor \log_2 15 \rfloor-1)+2)/15 +1=(16*2+2)/15 +1=3.2667$, respectively. For 1*16 game, the numbers of guesses required in the worst and expected cases are 5, and $((1*2+2*4+3*8+4*16)-(4*15))/16 +1=3.375$, respectively.

3. The graphic model

In this section, we invent a graphic model to represent the guessing process of the deductive games. By using this methodology, we are able to develop an optimal strategy for 2×n AB games in the next section. At first, we introduce the graphic model by means of a 2×5 AB game for simplicity. We start with some definitions. The game tree for the 2×5 AB game is shown in Figure 2. Any moment of the game is expressed by a node, represented by a *game graph* $G_i=<V_i, E_i>$, in the game tree. At first, the root $G_0=<V_0, E_0>$ of the game tree is a complete directed graph with 5 vertices and 20 edges. We map the game graphs to the AB game as follows:

Vertex: Each vertex in a game graph G_i corresponds to a *symbol* (digit) in the AB game, for example 0, 1, 2, 3, and 4 are five symbols in 2×5 AB game. Notice that we use the term “node” in the game tree and “vertex” in the game graphs.

Edge: Each directed edge in a game graph G_i refers to a possible codeword, so there are 5×4=20 edges in the root G_0 of the game tree. In the medium stages, the edges in a game graph G_i, i≥1, refer to *remaining candidates*; that is, unidentified codewords.

Partition edge: the edge codebreaker chooses to partition the game graph; for instance, the dashed arrow (0,1) shown in Figure 2, which refers to the first guess (0,1) in the 2×5 AB game.

Each *class*, represented as a node in the game tree, in level 2 refers to one response [A, B] after guessing (0,1) in the game. The five classes at level 2, that is, [2,0], [1,0], [0,1], [0,0] and [0,2], partition the set of 20 edges in the complete directed graph G_0 in the root node. Moreover, only one edge (0,1) remains in G_1, which refers to class [2,0]. It means that the edge (0,1) was *hit.* On the other hand, if there is only one edge in a class, except the class [2,0], then this edge is called *identified*, and it requires one more guess to hit it. For example, there is only one edge (1,0) in G_5. We need one more guess (1,0) to finish the AB game.

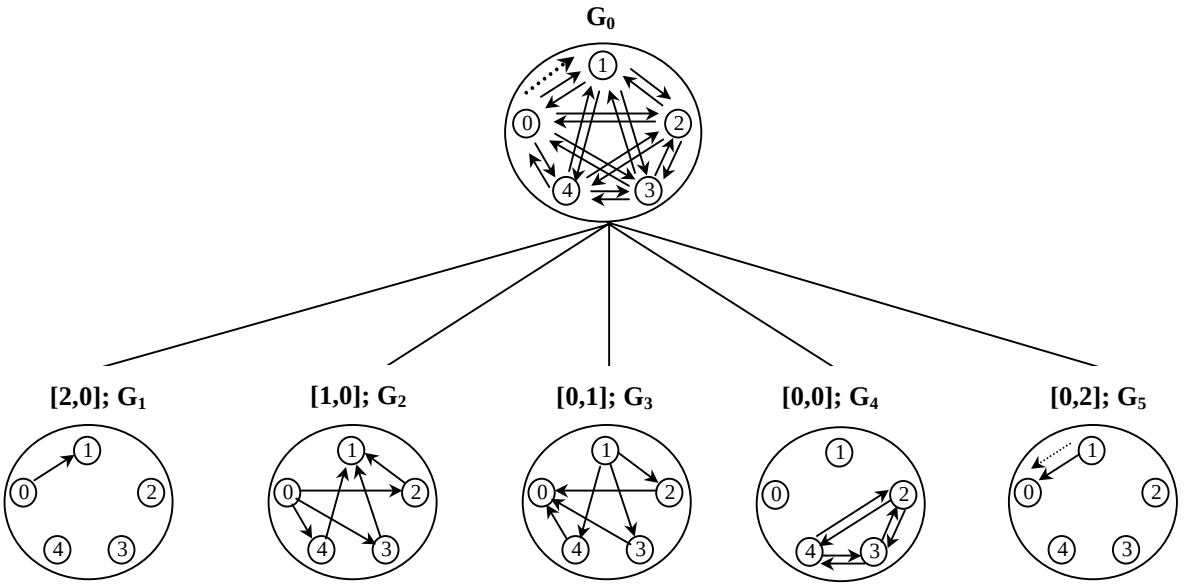

Figure 2. A graphic model to represent the results after the first guess for 2×5 AB game, where (0, 1) is the partition edge.

Now, let us present how to play the 2×5 AB game on the graph. Remember that a game graph G_i is a directed graph. If we have a partition edge (j, k), then vertex j is named *origin vertex* and vertex k is named *destination vertex*. We can partition the game graph G_i by the following simple criteria:

(1) The outgoing edges (j, m) from the origin vertex j and the incoming edges (m, k) to destination vertex k are classified into [1,0], $m \neq k$, $m \neq j$.

(2) The outgoing edges (k, m) from the destination vertex k and the incoming edges (m, j) to origin vertex j are classified into [0,1] , $m \neq j$, $m \neq k$.

(3) The edges that are not adjacent to the origin and destination vertices are classified into [0,0].

(4) The edge (j, k) that is both an outgoing edge from the origin vertex j and an incoming edge to destination vertex k is classified into [2,0].

(5) The edge (k, j) that is both an outgoing edge from the destination vertex k and an incoming edge to origin vertex j is classified into [0,2].

As depicted in Figure 2, at the initial stage, there is a complete directed graph G_0 with 20 edges, which refer to 20 possible codewords. The codebreaker repeatedly chooses one partition edge to partition these edges into five *classes*, that is, [2,0], [1,0], [0,1], [0,0] and [0,2]. For example, the partition edge (0,1) in G_0 partitions the 20 edges into five classes. The outgoing edges from origin vertex 0, i.e., (0,2), (0,3), and (0,4), and incoming edges to destination vertex 1, i.e., (2,1), (3,1), and (4,1), are classified into class [1,0], i.e., G_2. The edge (0,1) is both an incoming edge to vertex 1 and an outgoing edge from vertex 0, so it is classified into class [2,0], i.e., G_1. The other classes can be got in the same way.

Now we describe our goals of this paper and show how to achieve them by the graphic model. The goals are to reduce or minimize the number of guesses in the worst case or the expected case. The

number of guesses in the worst case is the largest number of guesses required to *hit* any one among all candidates. In addition, to obtain the number of guesses in the expected case, we first have to calculate *the total number of guesses required to hit all candidates*, and then divide it by the number of candidates.

By the partition criteria given above, we can translate the game-guessing process to a sequence of graph partition and tree traversal procedures. At first, we translate the problem "the number of guesses for a secret code (candidate)" to "the length of the path from the root to the *leaf* which contains the secret code (the remaining edge) *in the game tree*". Notice that all leaves in the game tree are "hits nodes", i.e., one candidate is *hit* in each of the leaf nodes. This property is a significant difference between 2×n AB games and 1×n games described in Section 2, where *hit nodes* are internal nodes in a game tree. Hence, the problem "the total number of guesses required to hit all candidates" can be translated to "*the external path length L* of the game tree [12]", i.e., the sum of the lengths of the paths from the root to each of all leaves in the game tree. Similarly, the problem of "the number of guesses required in the worst case" refers to "the height *H* of the game tree". Here the height H is defined as the length of a longest path from the root to a leaf. By these translations, our goals are thus to minimize the height H of the game tree for the worst case and to minimize the external path length L of the game tree for the expected case. The key to achieve these goals is simply to choose the best partition edge to partition the remaining edges at each stage, like playing the real game.

Now let us present a strategy to play 2×5 AB game on the graphic model and show how to calculate the external path length L and the height H of the game tree. By this work, we can develop sophisticated strategies for higher dimension games. In Lemma 2, we show how to calculate the total number of guesses required to hit 2 possible candidates (or 2 remaining edges) in a class (or a game graph). This lemma can be applied to m×n AB games with arbitrary m, n.

Lemma 2. *If a game graph that is the root node of a game tree contains only 2 remaining edges, then the minimum possible values for the external path length L and the height H of the game tree are 3 and 2, respectively.*

Proof: *Sufficient:* We can choose one of the two remaining edges as the partition edge. Then, this edge will be hit and the other edge will be identified and one more guess is required. Therefore, a possible value for the external path length L of the game tree is 1+2=3 and that for the height H of the game tree is 2.

Necessary: In the situation of two remaining edges in a game graph, there are only three possibilities to choose the partition edge.

Case 1. Choose one of these two remaining edges as the partition edge. As described in the sufficient condition, the external path length L of the game tree is 1+2=3 and the height H of the game tree is 2.

Case 2. Choose an edge adjacent to at least one of the two remaining edges. The best result of these guesses is able to identify the two remaining edges simultaneously, each of which requires one more guess to be hit. So the external path length L of the game tree is 2+2=4 and the height H

of the game tree is 2.

Case 3. Choose an edge not adjacent to the two remaining edges. The result of this guess has no contribution to further guesses, since the game graph after the guess is the same as the one before the guess. Thus we can omit this possibility.

Therefore, to hit 2 remaining edges in a game graph (or a class), the external path length L of the game tree is at least 3 and the height H of the game tree is at least 2. In other words, the total number of guesses to hit 2 remaining candidates is at least 3. Hence, the number of guesses in the expected case is 3/2=1.5. In addition, the number of guesses in the worst case is at least 2. ■

Observing the first guess in Figure 2, we can choose any edge, (0,1) for example, to be the partition edge, since the game graph G_0 is symmetric and complete. After the first guess, there is only one edge (0,1) remaining in G_1 (class [2,0]), i.e., 'hits'. Notice that Figure 2 shows only one leaf G_1 and the length from the root G_0 to G_1 is 1. Therefore, edge (0,1) only requires one guess to hit it. Although the edge (1,0) is also the only one edge in G_5 (class [0,2]), one more guess is required to 'hit' the edge (1,0) so that it requires 2 guesses. Furthermore, it is easy to show that the game graphs G_2, G_3 for classes [1,0] and [0,1] are *isomorphic*; that is, if we exchange vertex 0 and vertex 1 in the game graph G_2 for class [1,0], then it will be equivalent to the game graph G_3 for class [0,1]. Intuitively, they have the same number of guesses in both the worst case and the expected case. We deal with classes [1,0] and [0,0] in Figure 3 and Figure 4, respectively. (The process of class [0,1] is similar to class [1,0].)

In the following paragraphs, we describe the general procedures to calculate the number of guesses for 2×5 AB game in the worst and expected cases.

In Figure 3, the partition edge (2,3) partitions the remaining 6 edges into 3 nonempty classes, [1,0], [0,1], and [0,0], each of which has 2 remaining edges. By Lemma 2, the two remaining edges can be hit in one and two more guess respectively. Therefore, the total number of guesses (or the external path length L_2) for the 6 edges in G_2 is L_2= 2+3+2+3+2+3(or =1+2+1+2+1+2+6)=15; thus the expected number of guesses for G_2 is L_2/6=2.5.

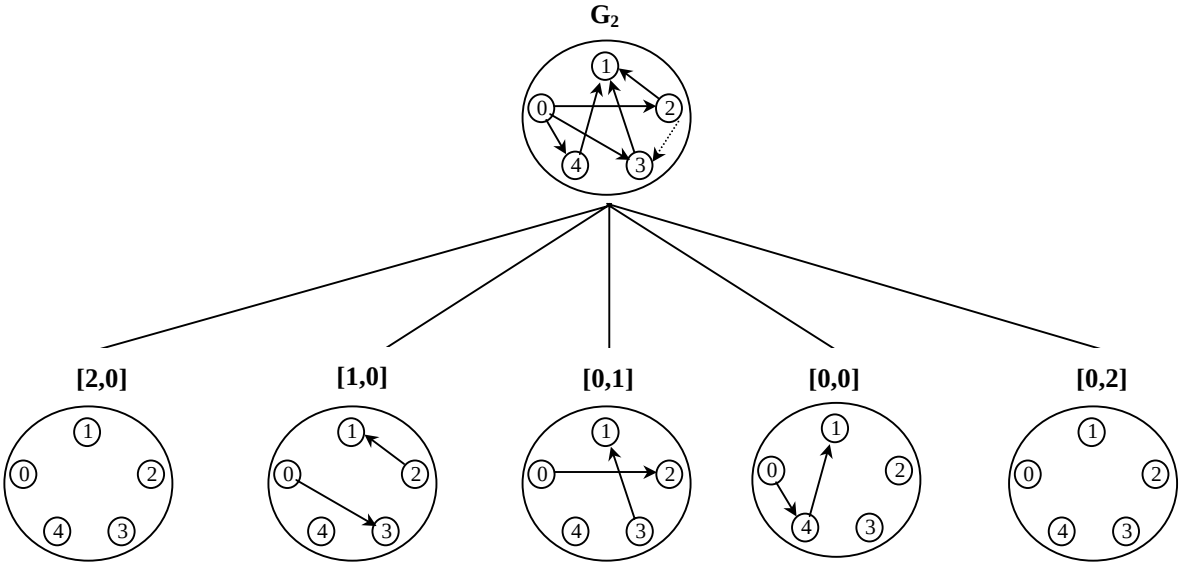

Figure 3. The game tree for G_2, where we choose (2,3) as the partition edge.

Now we consider how to calculate *the total number of guesses* (or the external path length L_4) for 2×3 AB game in the expected case. In G_4 of Figure 4, there is a complete subgraph with 3 vertices and 6 directed edges, which represents 2×3 AB game. Because it is a complete graph, we can choose any edge, (4,2) for example, as the partition edge. By using our graphic representation, as shown in Figure 4, we need 1, 2, 3, 2, 3, and 2 guesses to hit edges (4,2), (4,3), (3,2), (3,4), (2,3), and (2,4), respectively. Therefore, the total number of guesses (or the external path length L_4) for the six edges in G_4 is L_4=1+2+3+2+3+2= 13; thus the expected number of guesses for the game graph G_4 is L_4/6=13/6.

Finally, let us combine Figure 2, 3, and 4 all together. If we choose (2,3) and (4,2) as the partition edges in G_2 and G_4, respectively, then the height of the game tree is 4; that is, by using our analysis technique the number of guesses in the worst case is 4. The total number of guesses L_0 for G_0 can be computed as follows: L_0= (L_1+ L_2+ L_3+ L_4+ L_5)+ (the total number of edges in G_0) = (L_1+1) + (L_2+6) + (L_3+6) + (L_4+6) + (L_5+1) = (0+1)+(15+6)+(15+6)+(13+6)+(1+1)=64. Where L_1+1 is for G_1, L_2+6 is for G_2, and so on. Therefore, the expected number of guesses for G_0=64/20=3.2.

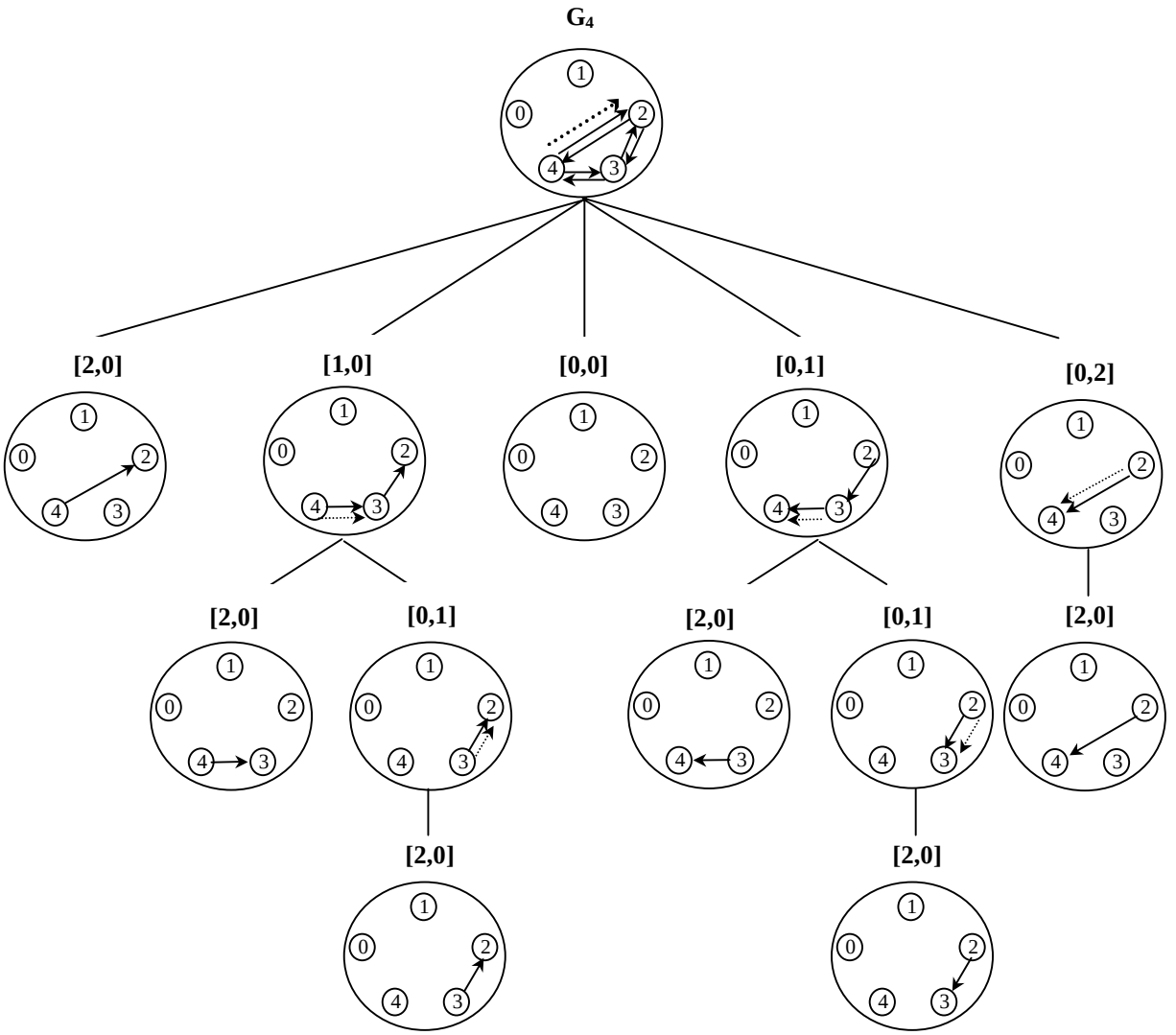

Figure 4. The game tree for G_4, which represents 2*3 AB game. The total number of guesses for the six edges is 13; the expected number of guesses for the graph is 13/6.

4. An optimum strategy for 2×n AB games

In this section, the graphic model described in Section 3 is used to develop an optimum strategy for 2×n AB games. In Figure 5, we simplify the graphic representation and define three functions for 2×n AB games. The rectangle shown in Figure 5(a) and the ellipse shown in Figure 5(b), denoted "i～j" inside, refer to a graph with only j-i+1 separate vertices named i, i+1, …, j and a complete directed graph with j-i+1 vertices named i, i+1, …, j, respectively. Let *T(n)*, $T_1(n)$, and $T_2(n)$ denote the total number of guesses required to hit all candidates for three types of game graph with n vertices, as shown in Figure 5(b), 5(c), and 5(d). Notice that the thick edges in Figure 5(c) and 5(d) refer to "one to all" or "all to one" edges, each of which connects the vertex outside the rectangle to all the vertices inside the rectangle. Hence there are n-1 and 2(n-2) edges in Figure 5(c) and 5(d), respectively.

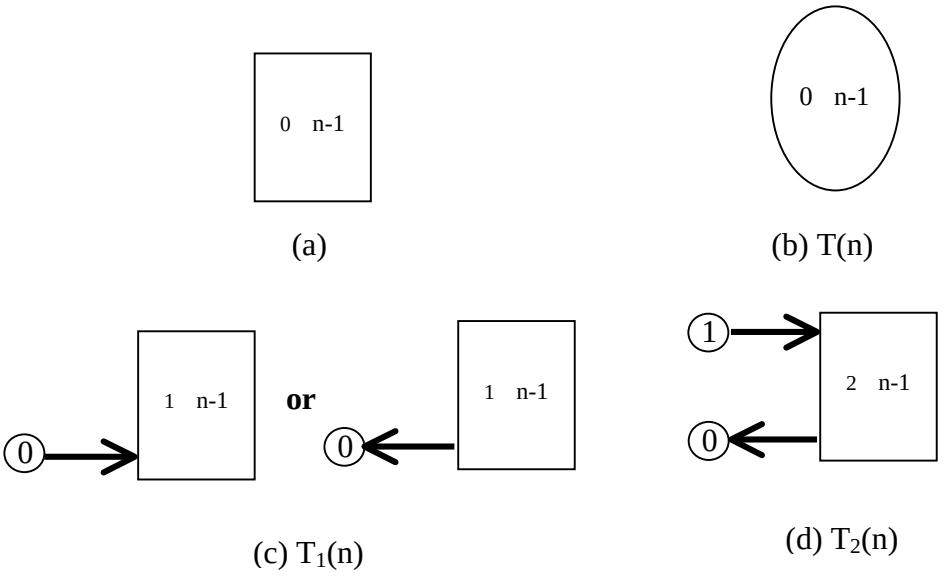

Figure 5. (a) A graph with n vertices and no edges. (b) A complete directed graph with n vertices and n(n-1) edges. T(n) is its total number of guesses. (c) Two types of graphs with n-1 edges. $T_1(n)$ is its total number of guesses. (d) A graph with 2(n-2) edges. $T_2(n)$ is its total number of guesses.

Now, we will demonstrate our procedure to minimize the external path length L and the height H of the game tree for 2×n AB games, by which we can obtain the minimum number of guesses in the expected and worst cases. First of all, we choose (0,1) as the partition edge at the initial stage. The game tree after the first guess is shown in Figure 6. The numbers of further guesses required for the classes [2,0], [1,0], [0,1], [0,0], and [0,2] (the external path lengths of subtrees whose roots are the nodes for classes [2,0], [1,0], [0,1], [0,0], and [0,2]) are 0, $T_2(n)$, $T_2(n)$, T(n-2), and 1, respectively. In addition, since the number of guesses for each remaining candidate (the length form root to each leaf) will be increased by one after the first guess, we have to add the quantity n(n-1) for computing T(n), where n(n-1) is the number of edges before the guess(i.e. in the root node). Hence, the total number of

guesses (the external path length of the game tree) $T(n) = 0+ T_2(n)+ T_2(n)+ T(n-2) +1+n(n-1)= T(n-2)+2T_2(n)+ n^2-n+1$.

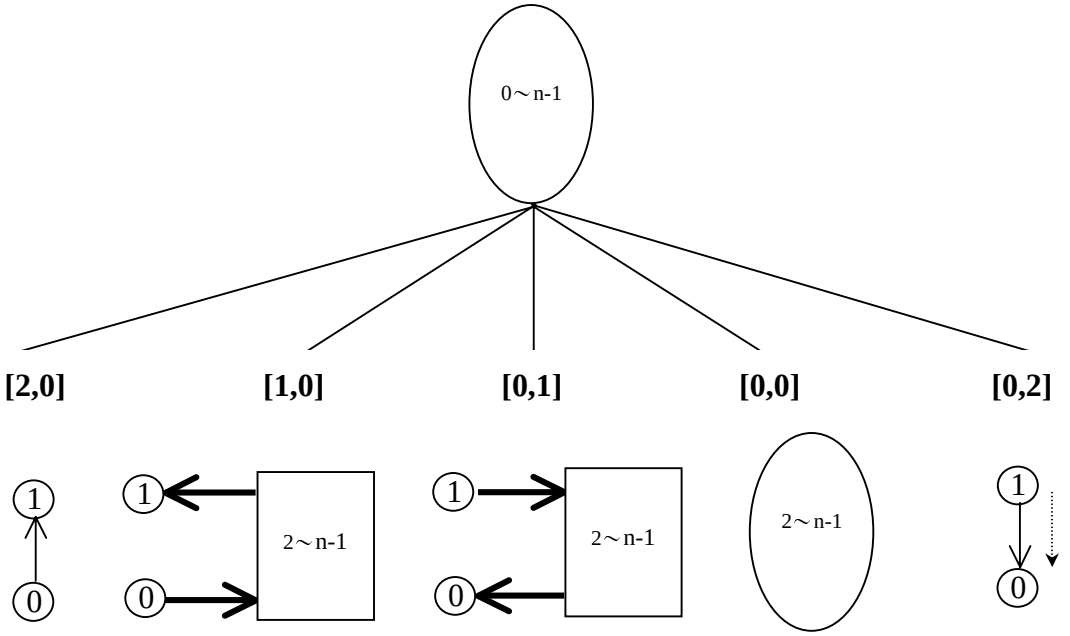

Figure 6. The game tree for 2×n AB game, where we choose (0,1) as the partition edge. $T(n)=0+ T_2(n)+ T_2(n)+T(n-2)+1+n(n-1)= T(n-2)+2 T_2(n)+ 1+n(n-1)$.

In Figure 6, since the game graphs for the classes [1,0] and [0,1] are isomorphic, and the game graph for class [0,0] can be solved recursively, we can only pay our attention to deal with the class [1,0]. The four ways to partition the class [1,0] and their recurrences are shown as follows:

1. Choose (0, y) or (y, 1) as the partition edge, where $y \in \{2, 3, 4, \ldots, n-1\}$. The game graph after the first guess is shown in Figure 7, where we choose (0,2) as the partition edge. Now, The numbers of further guesses required for the classes [2,0], [1,0], [0,1], and [0,0] are 0, $T_1(n-2)$, 1, and $T_1(n-2)$, respectively. In addition, we have to add the quantity 2(n-2) for computing $T_2(n)$, where 2(n-2) is the number of edges before the guess (0,2). Therefore, $T_2(n) = 0 + T_1(n-2)+ 1+T_1(n-2) +2(n-2) =2 T_1(n-2)+ 2n- 3$.
2. Choose (y, 0) or (1, y) as the partition edge, where $y\in\{2, 3, 4, \ldots, n-1\}$. For example, in Figure 8, we choose (2,0) as the partition edge. The numbers of further guesses required for the classes [2,0], [1,0], [0,1], [0,0] and [0,2] are 0, 1, $T_1(n-2)$, $T_1(n-2)$, and 1, respectively. Therefore, $T_2(n)= 1+ T_1(n-2) +T_1(n-2) +1 +2(n-2)=2 T_1(n-2)+ 2n-2$. Choosing (1,2) as the partition edge will get the same result by the similar analysis, so we omit it here.
3. Choose (y_1, y_2) as the partition edge, where $y_1, y_2 \in \{2, 3, 4 \ldots, n-1\}$ and $y_1 \quad y_2$. For example, in Figure 9, we choose (2,3) as the partition edge. Now, $T_2(n)=3+3+T_2(n-2)+2(n-2) = T_2(n-2)+2n +2$.
4. Choose (0,1) or (1,0) as the partition edge. As shown in Figure 10, if we choose (0,1) as the partition edge, there is only one nonempty class [1,0], which contains all the 2(n-2) edges; similarly, if (1,0) is chosen, the only one nonempty class is [0,1]. That is, we get no clue from this guess and also have to

add the quantity 2(n-2) for computing $T_2(n)$. Therefore, $T_2(n)= T_2(n)+ 2(n-2)$.

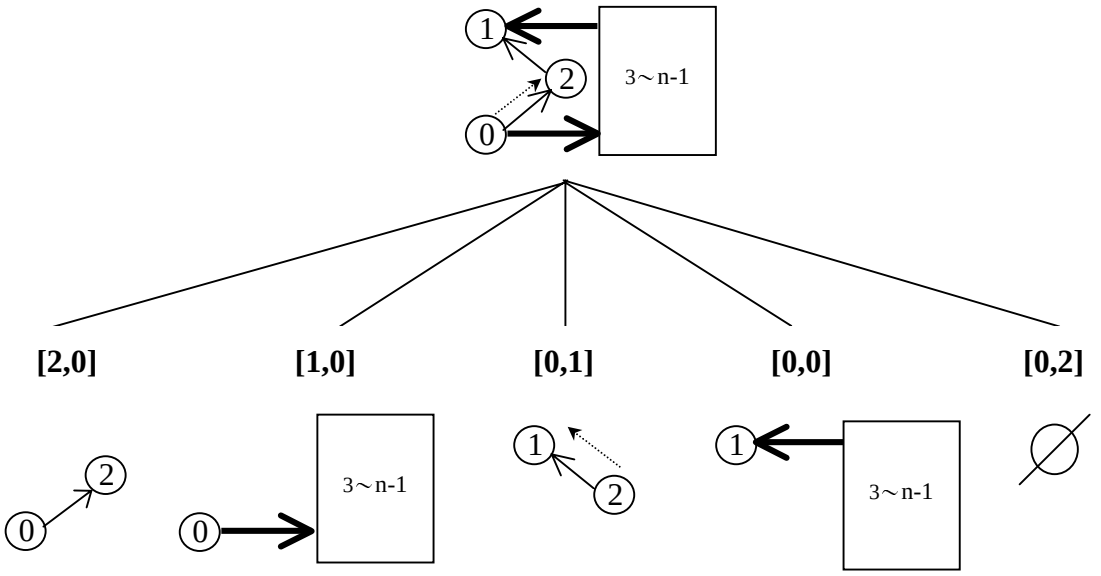

Figure 7. The game tree of graph $T_2(n)$, where we choose (0,2) as the partition edge. The notation "∅" in the class [0,2] refers to an empty set. $T_2(n)=0+T_1(n-2)+1+T_1(n-2)+2(n-2)=2\ T_1(n-2)+2n-3$.

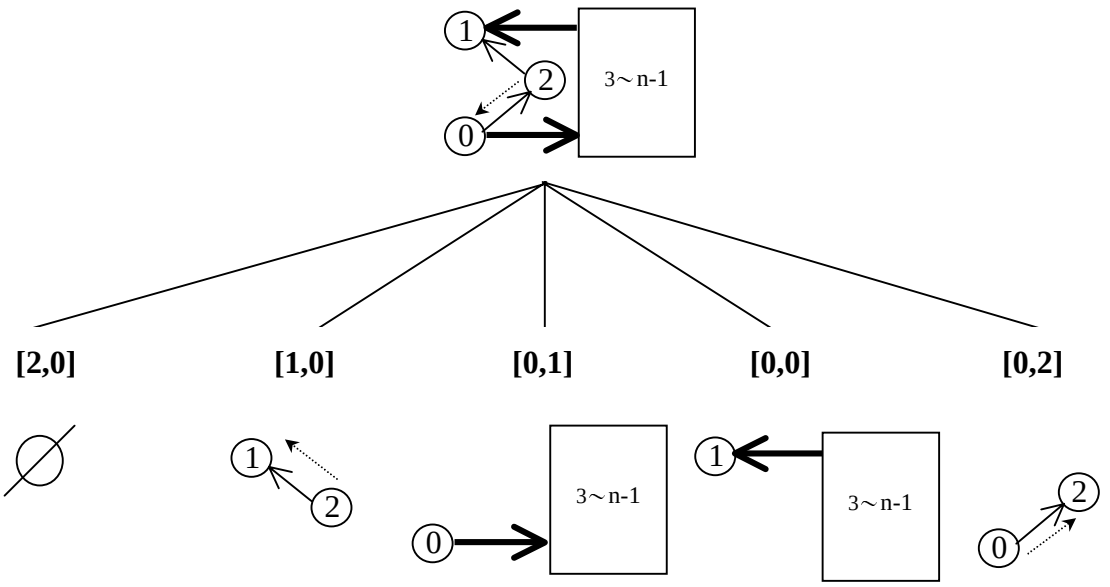

Figure 8. The game tree of graph $T_2(n)$, where we choose (2,0) as the partition edge.
$T_2(n)=0+1+T_1(n-2)+T_1(n-2)+1+2(n-2)=2\ T_1(n-2)+2n-2$.

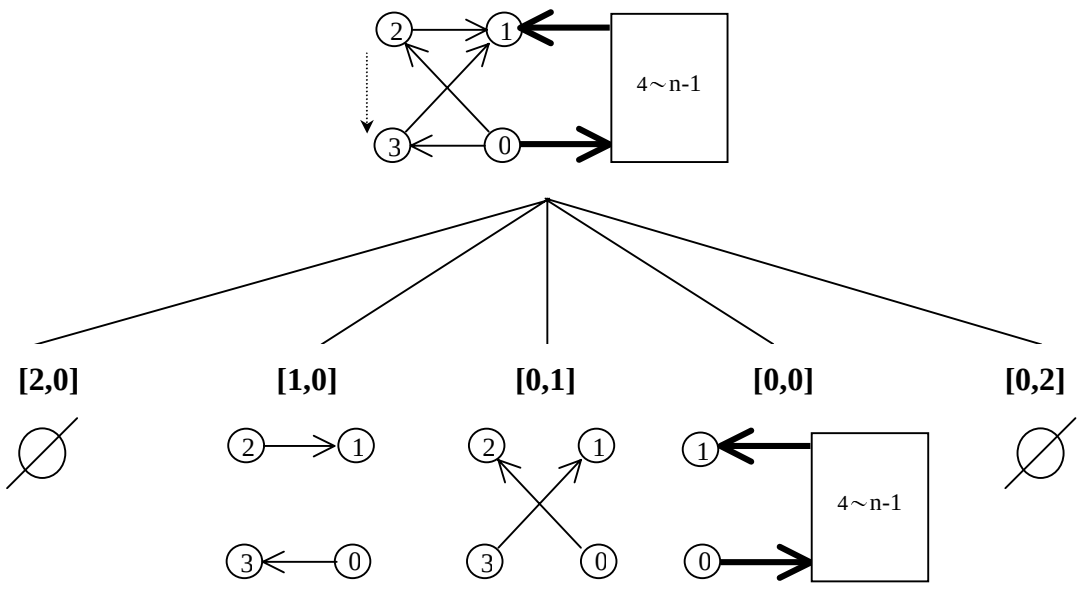

Figure 9. The game tree of graph $T_2(n)$, where we choose (2,3) as the partition edge.

$T_2(n)=3+3+ T_2(n-2)+2(n-2)= T_2(n-2)+2n+2$.

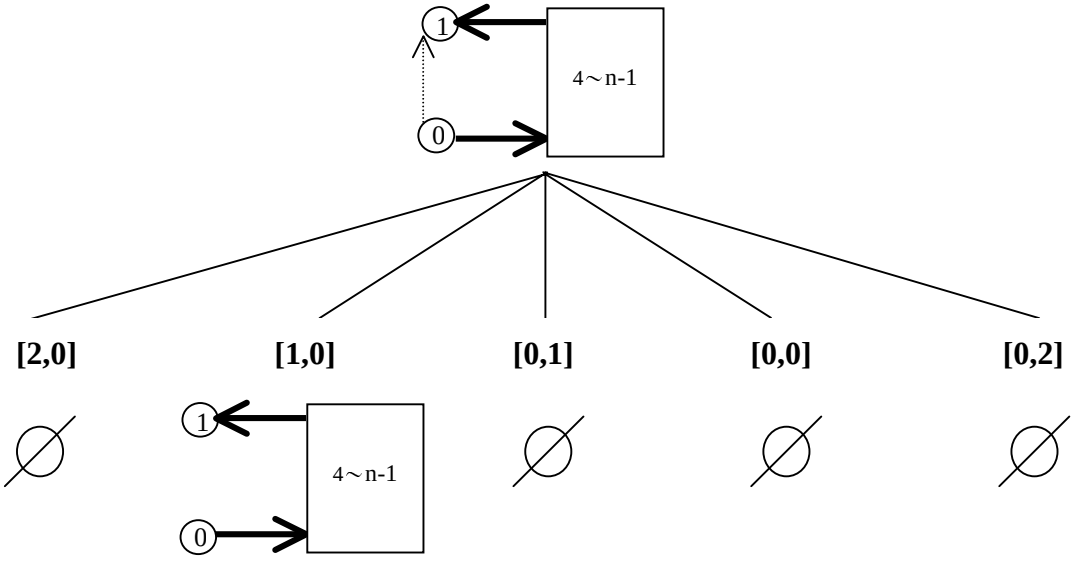

Figure 10. The game tree of graph $T_2(n)$, where we choose (0,1) as the partition edge.

$T_2(n)= T_2(n)+2(n-2)= T_2(n)+2n-4$.

Observing the above recurrences, the total number of guesses for strategy 2 is always one more than that for strategy 1. In addition, we get no clue by using strategy 4. Therefore, we can ignore strategy 2 and strategy 4 here. Now, we can just investigate which is better between strategies 1 and 3. That is, we can determine the optimal strategy for $T_2(n) = \text{Min}(2T_1(n-2)+2n-3, T_2(n-2)+2n+2)$.

Now let's focus on the graph for $T_1(n)$. There are two types of graphs, between which only the edge direction is different, as shown in Figure 5(c). However, we can obtain the same numbers of

further guesses for these two types of graphs by changing the direction of the partition edge. Hence, without losing the generality, we can only consider one type of the graphs. Now we show the only two ways to partition $T_1(n)$ and their recurrences, as follows:

(i) Choose (0, y) as the partition edge, where y∈{1, 2, 3, …, n-1}. In Figure 11(a), we choose (0, 1) as the partition edge, which partitions the graph for $T_1(n)$ into two nonempty classes, [2, 0] and [1, 0]. The numbers of further guesses for these two classes are 0 and $T_1(n-1)$ respectively. Actually, there are n-1 edges in the graph for $T_1(n)$. Therefore, $T_1(n)=0+ T_1(n-1)+n-1= T_1(n-1) +n-1$.

(ii) Choose (y_1, y_2) as the partition edge, where y_1, y_2∈{1, 2, 3, …, n-1} and y_1 y_2. For example, we choose the edge (1, 2) as the partition edge, as shown in Figure 11(b). Now, $T_1(n)=1+ T_1(n-2) +1+n-1= T_1(n-2)+n+1$.

Form the above analysis, we have $T_1(n)=\text{Min}(T_1(n-2)+n+1, T_1(n-1)+n-1)$.

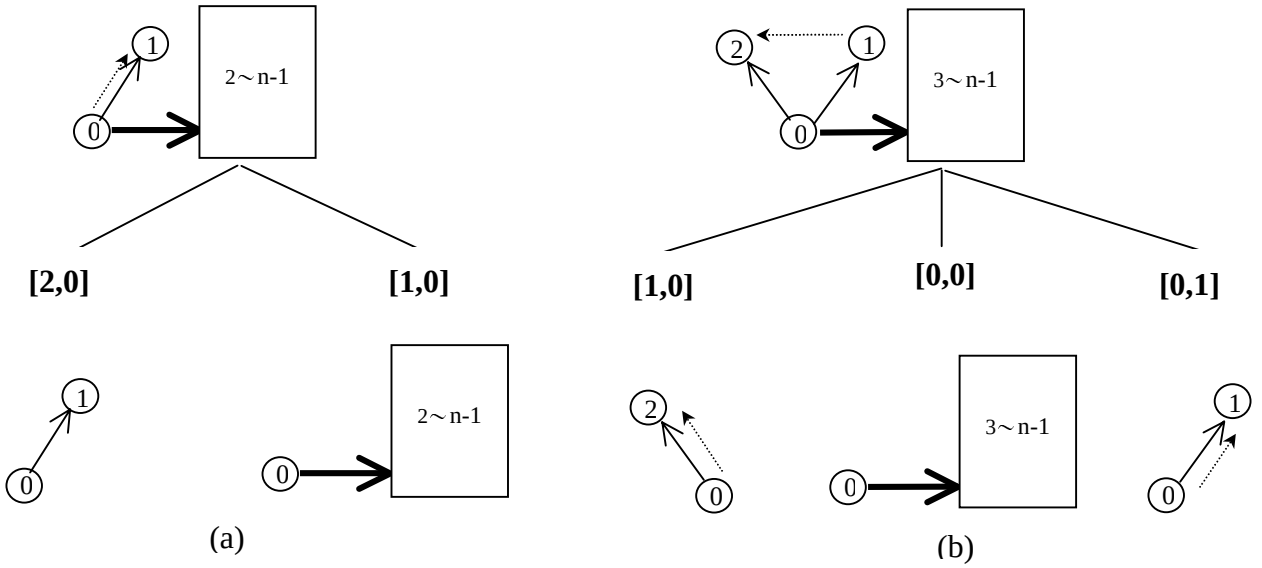

Figure 11. (a) The strategy (i), where (0,1) is the partition edge. $T_1(n)= 0+T_1(n-1)+n-1$.
(b) The strategy (ii), where (1,2) is the partition edge. $T_1(n)=1+ T_1(n-2)+1+n-1$.

To minimize the total number of guesses, we list the recurrences developed above and their solutions in Table 1, whose detailed proofs are shown in Theorem 3.

unctions	Recurrences relations	Solutions
T(n)=	$\begin{cases} 3, & \text{if } n=2 \\ 13, & \text{if } n=3 \\ T(n-2)+2T_2(n)+n^2-n+1, & \text{otherwise} \end{cases}$	$\begin{cases} (4n^3+21n^2-76n+72)/12, & \text{if n is even} \\ (4n^3+21n^2-82n+105)/12, & \text{if n is odd} \end{cases}$
T_2 (n)=	$\begin{cases} 3, & \text{if } n=3 \\ 7, & \text{if } n=4 \\ \text{Min}(2T_1(n-2)+2n-3, T_2(n-2)+2n+2), & \text{otherwise} \end{cases}$	$\begin{cases} (n^2+4n-18)/2, & \text{if n is even} \\ (n^2+4n-19)/2, & \text{if n is odd} \end{cases}$
T_1 (n)=	$\begin{cases} 1, & \text{if } n=2 \\ 3, & \text{if } n=3 \\ \text{Min}(T_1(n-2)+n+1, T_1(n-1)+n-1), & \text{otherwise} \end{cases}$	$\begin{cases} (n^2+4n-8)/4, & \text{if n is even} \\ (n^2+4n-9)/4, & \text{if n is odd} \end{cases}$

Table 1. The recurrence relations and their solutions for T(n), T_2(n), and T_1(n).

Theorem 3. *The minimum number of guesses required for 2×n AB games in the expected case is $(4n^3+21n^2-76n+72)/12n(n-1)$ if n is even, and is $(4n^3+21n^2-82n+105)/12\ n(n-1)$ if n is odd.*

Proof. We begin with solving the recurrences *T(n), T_2(n), and T_1(n)* listed in Table 1.

1. $T_1(n) = \begin{cases} 1, & \text{if } n=2, \\ 3, & \text{if } n=3, \\ \text{Min}(T_1(n-2)+n+1, T_1(n-1)+n-1), & \text{otherwise.} \end{cases}$

At first, we shall prove by induction that

$T_1(n) = T_1(n-2)+n+1$ (i.e., Min($T_1(n-2)+n+1$, $T_1(n-1)+n-1$)= $T_1(n-2)+n+1$).

Basis step:

$T_1(4)$ =Min($T_1(4-2)+4+1$, $T_1(4-1)+4-1$)=Min(6,6), and

$T_1(5)$ = Min($T_1(5-2)+5+1$, $T_1(5-1)+5-1$)=Min(3+5+1,6+5-1)=Min(9,10),

so $T_1(n) = T_1(n-2)+n+1$ is true for n=4,5.

Inductive step:

Assume that $T_1(k) = T_1(k-2)+k+1$ is true for $4 \le k < n$.

We want to prove that $T_1(n) = T_1(n-2)+n+1$ is also true.

$T_1(n)$ = Min($T_1(n-2)+n+1$, $T_1(n-1)+n-1$)

= Min($T_1(n-4)+(n-1)+(n+1)$, $T_1(n-3)+n+(n-1)$)

= Min($T_1(n-6)+(n-3)+(n-1)+(n+1)$, $T_1(n-5)+(n-2)+n+(n-1)$)

$$= \begin{cases} \text{Min}(T_1(2)+5+7+\ldots+(n+1),\ T_1(3)+6+8+\ldots+n+(n-1)), & \text{if n is even,} \\ \text{Min}(T_1(3)+6+8+\ldots+(n+1), T_1(2)+5+7+\ldots+n+(n-1)), & \text{if n is odd.} \end{cases}$$

$$= \begin{cases} \text{Min}((n^2+4n-8)/4,\ (n^2+6n-16)/4), & \text{if n is even,} \\ \text{Min}((n^2+6n-9)/4,\ (n^2+6n-15)/4), & \text{if n is odd.} \end{cases}$$

Since $(n^2+4n-8)/4 \le (n^2+6n-16)/4$ and

$(n^2+4n-9)/4 \le (n^2+6n-15)/4$ for $n \ge 4$,

we conclude that $T_1(n) = T_1(n-2)+n+1$ is true and

$$T_1(n) = \begin{cases} (n^2+4n-8)/4, & \text{if } n \text{ is even,} \\ (n^2+4n-9)/4, & \text{if } n \text{ is odd.} \end{cases}$$

2. $$T_2(n) = \begin{cases} 3, & \text{if } n = 3, \\ 7, & \text{if } n = 4, \\ \text{Min}(2T_1(n-2)+2n-3,\ T_2(n-2)+2n+2), & \text{otherwise.} \end{cases}$$

Now, we shall prove by induction that

$T_2(n) = 2T_1(n-2)+2n-3$.

Basis step:

$T_2(5) = \text{Min}(2T_1(3)+10-3,\ T_2(3)+10+2) = \text{Min}(6+10-3, 3+10+2) = \text{Min}(13,15)$, and

$T_2(6) = \text{Min}(2T_1(4)+12-3,\ T_2(4)+12+2) = \text{Min}(12+12-3, 7+12+2) = \text{Min}(21,21)$,

so $T_2(n) = 2T_1(n-2)+2n-3$ is true for $n=5,6$.

Inductive step:

Assume that $T_2(k) = 2T_1(k-2)+2k-3$ is true for $5 \le k < n$.

We want to prove that $T_2(n) = 2T_1(n-2)+2n-3$ is also true.

$T_2(n) = \text{Min}(2T_1(n-2)+2n-3,\ T_2(n-2)+2n+2)$

$= \text{Min}(2T_1(n-2)+2n-3,\ 2T_1(n-4)+2(n-2)-3+(2n+2))$

$= \text{Min}(2T_1(n-2)+2n-3,\ 2T_1(n-4)+4n-5)$

$$= \begin{cases} \text{Min}\big(2((n-2)^2+4(n-2)-8)/4+2n-3,\ 2((n-4)^2+4(n-4)-8)/4+4n-5\big), & \text{if } n \text{ is even,} \\ \text{Min}\big(2((n-2)^2+4(n-2)-9)/4+2n-3,\ 2((n-4)^2+4(n-4)-9)/4+4n-5\big), & \text{if } n \text{ is odd.} \end{cases}$$

$$= \begin{cases} \text{Min}((n^2+4n-18)/2,\ (n^2+4n-18)/2), & \text{if } n \text{ is even,} \\ \text{Min}((n^2+4n-19)/2,\ (n^2+4n-19)/2), & \text{if } n \text{ is odd.} \end{cases}$$

Therefore, we conclude that $T_2(n) = 2T_1(n-2)+2n-3$ is true and

$$T_2(n) = \begin{cases} (n^2+4n-18)/2, & \text{if } n \text{ is even,} \\ (n^2+4n-19)/2, & \text{if } n \text{ is odd.} \end{cases}$$

3. $$T(n) = \begin{cases} 3, & \text{if } n = 2, \\ 13, & \text{if } n = 3, \\ T(n-2)+2T_2(n)+n^2-n+1, & \text{otherwise.} \end{cases}$$

If $n>3$ is even,

$$
\begin{aligned}
T(n) &= T(n-2)+2T_2(n)+ n^2-n+1 \\
&= T(n-2)+2(n^2 +4n-18)/2+ n^2-n+1 \\
&= T(n-2)+2n^2+3n-17 \\
&= T(2)+ \sum_{i=0}^{(n-4)/2}[2(n-2i)^2 +3(n-2i)-17] \\
&= T(2)+ \sum_{i=0}^{(n-4)/2}[2n^2 +3n -17+8i^2 -8ni-6i] \\
&= 3+[(n-4)/2+1](2n^2+3n-17)+ \sum_{i=0}^{(n-4)/2}[8i^2 -8ni-6i] \\
&= 3+[(n-4)/2+1](2n^2+3n-17)+8\sum_{i=0}^{(n-4)/2} i^2 -(8n+6)\sum_{i=0}^{(n-4)/2} i \\
&= 3+[(n-4)/2+1](2n^2+3n-17)+8[(n^3-9n^2+26n-24)/24] -(8n+6)[(n^2-6n+8)/8] \\
&= (4n^3+21n^2-76n+72)/12.
\end{aligned}
$$

If n is odd,

$$
\begin{aligned}
T(n) &= T(n-2)+2T_2(n)+ n^2-n+1 \\
&= T(n-2)+2(n^2 +4n-19)/2+ n^2-n+1 \\
&= T(n-2)+2n^2+3n-18 \\
&= T(3)+ \sum_{i=0}^{(n-5)/2}[2(n-2i)^2 +3(n-2i)-18] \\
&= (4n^3+21n^2-82n+105)/12.
\end{aligned}
$$

Therefore,

$$
T(n) = \begin{cases} (4n^3 + 21n^2 - 76n + 72)/12, & \text{if } n \text{ is even,} \\ (4n^3 + 21n^2 - 82n + 105)/12, & \text{if } n \text{ is odd.} \end{cases}
$$

The necessary and sufficient conditions can be achieved by the fact that our procedure always chooses the best strategy at each stage. Therefore, the minimum number of guesses required for 2×n AB games is $(4n^3+21n^2 -76n+72)/12$ if n is even, and is $(4n^3+21n^2 -82n+105)/12$ if n is odd. Because the number of possible candidates of 2×n AB games is n(n-1), we have to divide these two numbers by n(n-1) in the expected case. This completes the proof. ■

To determine the number of guesses for 2×n AB games in the worst case, we consider the height instead of the external path length of the game tree. Let H(n), $H_1(n)$, and $H_2(n)$ be the minimum possible value for the height of the game tree for the game graphs in Figure 5(b), 5(c), and 5(d) respectively. We can obtain the recurrences for H(n), $H_1(n)$, and $H_2(n)$ by slightly modifying the recurrences T(n), $T_1(n)$, and $T_2(n)$ shown in Table 1. Observing Figure 6, the height of the game tree will be 1 plus the height of the highest among the five subtrees whose roots are the nodes for the five classes. Therefore, we have $H(n)=\text{Max}(0, H_2(n), H_2(n), H(n-2), 1)+1= \text{Max}(H_2(n), H(n-2))+1$. We can also obtain this recurrence from the recurrence $T(n) = T(n-2) + 2T_2(n) +n^2-n+1$. That is, change the coefficient 2 associated with

the recurrence function $T_2(n)$ into 1, the cost (n^2-n+1) for each iteration into 1, and the *sum operations* between recurrence functions in the right side into *Max function*. The recurrences $H_1(n)$ and $H_2(n)$ can be got by the similar way. The recurrence relations and their solutions for H(n), $H_1(n)$, and $H_2(n)$ are shown in the Table 2. We show how to obtain the solutions in Theorem 4.

Functions	Recurrences relations	Solutions
$H(n)=$	$\begin{cases} 2, & \text{if } n=2 \\ 3, & \text{if } n=3 \\ \text{Max}(H(n-2), H_2(n))+1, & \text{otherwise} \end{cases}$	$\lceil n/2 \rceil+1$
$H_2(n)=$	$\begin{cases} 2, & \text{if } n=3 \\ 2, & \text{if } n=4 \\ \text{Min}(H_1(n-2), H_2(n-2))+1, & \text{otherwise} \end{cases}$	$\lceil n/2 \rceil$
$H_1(n)=$	$\begin{cases} 1, & \text{if } n=2 \\ 2, & \text{if } n=3 \\ \text{Min}(H_1(n-2), H_1(n-1))+1, & \text{otherwise} \end{cases}$	$\lceil n/2 \rceil$

Table 2. The recurrence relations and their minimum solutions for H(n), $H_1(n)$, and $H_2(n)$.

Theorem 4. $\lceil n/2 \rceil+1$ guesses are necessary and sufficient for 2×n AB games in the worst case.

Proof. We solve the recurrences H(n), $H_1(n)$, and $H_2(n)$ according to Table 2.

1. Since $H_1(n-2) \le H_1(n-1)$, $H_1(n) = \text{Min}(H_1(n-2), H_1(n-1)) + 1 = H_1(n-2) + 1$ for $n>3$.

$$H_1(n) = H_1(n-2) + 1$$
$$= H_1(n-4) + 1 + 1$$
$$= \ldots$$
$$= \begin{cases} H_1(2) + (n-2)/2 = (n-2)/2 + 1, & \text{if } n \text{ is even,} \\ H_1(3) + (n-3)/2 = (n-3)/2 + 2, & \text{if } n \text{ is odd.} \end{cases}$$
$$= \lceil n/2 \rceil.$$

2. $H_2(n) = \text{Min}(H_1(n-2), H_2(n-2))+1$ for $n>4$.

At first, we shall prove by induction that

$H_2(n) = H_1(n-2)+ 1$.

Basis step:

$H_2(5) = \text{Min}(H_1(3)+1, H_2(3)+1) = \text{Min}(3,3)$, and

$H_2(6) = \text{Min}(H_1(4)+1, H_2(4)+1) = \text{Min}(2+1,2+1) = \text{Min}(3,3)$,

so $H_2(n) = H_1(n-2)+ 1$ is true for $n=5,6$.

Inductive step:

Assume that $H_2(k) = H_1(k-2)+ 1$ is true for $5 \le k < n$.

We want to prove that $H_2(n) = H_1(n-2)+ 1$ is also true.

$H_2(n) = \text{Min}(H_1(n-2), H_2(n-2))+1$

$= \text{Min}(\lceil n/2 \rceil/2, H_1(n-4)+1)+1$

$= \text{Min}(\lceil n/2 \rceil-1, \lceil n/2 \rceil-1)+1.$

Therefore, $H_2(n) = H_1(n-2)+ 1$ is true and

$H_2(n) = \lceil n/2 \rceil$

3. $H(n) = \text{Max}(H(n-2), H_2(n)) + 1$ for $n>3$.

Now, we shall prove by induction that

$H(n) = H_2(n)+ 1$.

Basis step:

$H(4) = \text{Max}(H(2), H_2(4)) +1 = \text{Max}(2,2)+1$, and

$H(5) = \text{Max}(H(3), H_2(5)) +1 = \text{Max}(3,3)+1$,

so $H(n) = H_2(n)+ 1$ is true for $n=4,5$.

Inductive step:

Assume that $H(k) = H_2(k)+ 1$ is true for $4 \le k < n$.

We want to prove that $H(n) = H_2(n)+ 1$ is also true.

$H(n) = \text{Max}(H(n-2), H_2(n)) + 1$

$= \text{Max}(H_2(n-2) + 1, \lceil n/2 \rceil)+1$

$= \text{Max}(\lceil n/2 \rceil, \lceil n/2 \rceil)+1.$

Therefore, *$H(n) = H_2(n)+ 1$ is true and*

$H(n) = \lceil n/2 \rceil + 1.$ ■

We apply the optimum strategy obtained in this section to 2×5 AB game. The height of the game tree for the strategy used in Section 2 for 2×5 AB game is 4, which has already met the best result, $H(n) = \lceil n/2 \rceil+1 = \lceil 5/2 \rceil+1 = 4$, in Theorem 4. Nevertheless, the strategy used in Figure 3, i.e., choosing (2,3) as the partition edge, is not the best choice if we consider the external path length. As proven in Theorem 3, if we use the best strategy, i.e., choosing (0, y) or (y, 1) as the partition edge, where $y \in \{2, 3, 4\}$, then the external path length L_2 of the game tree will be 13 instead of 15 in Figure 3. Thus, the total number of guesses required for 2*5 AB game, $L_0 = (L_1 + L_2 + L_3 + L_4 + L_5)+$ (the total number of edge in G_0)= (0+13+13+13+1)+20=60. Therefore, the minimum number of guesses required for G_0=60/20=3, which meets the result, $(4n^3 + 21n^2 - 82n + 105) / 12n(n-1) = (4*5^3 + 21*5^2 - 82*5 + 105)/(12*5*4)=3$, in Theorem 3.

5. Concluding remarks

We have presented in this paper a systematic methodology to obtain optimal strategies for 2×n AB games in both the worst and expected case. First, we take advantage of properties in game trees, such as the height and the external path length of the game tree, to calculate the numbers of guesses required in the worst and expected cases for a deductive game. Furthermore, we invent a graphic model to

represent the 2×n AB game and discover some recursive and isomorphic properties in the graphic model. By this, we can reduce the search space and obtain the optimal results in both the worst and the expected case. We hope that the methodologies proposed in this paper will prompt researchers to study other related problems.

REFERENCES

[1] Knuth, D. E.: The computer as Mastermind, Journal of Recreational Mathematics, 9:1, 1-6 (1976).

[2] Koyama, K., Lai, T. W.: An optimal Mastermind strategy, Journal of Recreational Mathematics, 25, 251-256 (1993).

[3] Irving, R. W.: Towards an optimum Mastermind strategy, Journal of Recreational Mathematics, 11:2, 81-87 (1978-79).

[4] Neuwirth, E.: Some strategies for Mastermind, Zeitschrift fur Operations Research, 26, 257-278 (1982).

[5] Kabatianski, G., Lebedev, V.: The Mastermind game and the rigidity of the Hamming space, In: Proceedings of the International Symposium on Information Theory IEEE, 2000, 375-375.

[6] Flood, M. M.: Sequential search strategies with Mastermind variants — Part 1, Journal of Recreational Mathematics, 20:2, 105-126 (1988).

[7] Roche, J. R.: The value of adaptive questions in generalized Mastermind, In: Proceedings of the International Symposium on Information Theory IEEE, 1997, 135-135.

[8] Bento, L., Pereira, L., Rosa, A.: Mastermind by evolutionary algorithms, In: Proceedings of the International Symposium on Applied Computing, 1999, 307-311.

[9] Bernier, J. L., et al.: Solving Mastermind using Gas and simulated annealing: a case of dynamic constraint optimization, Proceedings PPSN, Parallel Problem Solving from Nature IV, in Computer Science, 1141, 554-563 (1996).

[10] Ko, K.-I., Teng, S.-C.: On the number of queries necessary to identify a permutation. Journal of Algorithms, 7, 449-462 (1886).

[11] Chen, Z., et al.: Finding a hidden code by asking questions. COCOON'96, Computing and Combinatorics, 50-55 (1996).

[12] Sedgewick, R.: Algorithms, second edition, Addison-Wesley, 1988.

[13] Knuth, D. E.: The art of computer programming- sorting and searching, volume 3, second edition, 1998.