Int. Computer Symposium, Dec. 15-17, 2004, Taipei, Taiwan.

Novel Algorithms for Deductive Games

Shan-Tai Chen
Department of Computer Science, Chung Cheng Institute of Technology, National Defense University, Tao-Yuan, Taiwan, R.O.C.
Email: stchen@ccit.edu.tw

Shun-Shii Lin
Graduate Institute of Computer Science and Information Engineering, National Taiwan Normal University, No. 88, Sec. 4, Ting-Chow Rd. *Taipei, Taiwan, R.O.C.*
Email: linss@csie.ntnu.edu.tw

Abstract- *This paper presents two novel algorithms for deductive games. First, a k-way-branching algorithm, taking advantage of a clustering technique, is able to efficiently obtain an optimal strategy in the worst case and a near-optimal strategy in the expected case for a typical deductive game "Bulls and Cows." Second, a pigeonhole-principle-based backtracking algorithm has been successfully applied to efficiently reduce the search space for the game. By using the algorithms, we not only obtain the lower bound on number of guesses required for the game in the worst case, but also derive the main theorem: 7 guesses are necessary and sufficient for the "Bulls and Cows" in the worst case. This is the first paper to prove the exact bound of this problem.*

1. Introduction

Deductive games are *two-player* games of *imperfect information*. Player I, called the *codemaker*, chooses a secret code. Player II, called the *codebreaker*, does not know the choice player I made and has to guess the secret code. After each guess, the codebreaker gets a hint about the accuracy of the guess from the codemaker. The goal of codebreaker is to discover the secret code, based on the hints, in the smallest number of guesses. Merelo et al. [10] formulated the problem as a combinatorial optimization problem. It bears some resemblance to other combinatorial problems, such as *circuit testing*, *differential cryptanalysis*, *on-line models with equivalent queries*, and *additive search problems*. Consequently, any conclusion of this kind of deductive game may be applied, although probably not directly, to any of these problems, as well as to any other combinatorial optimization problem.

Over the past three decades, much research has been done on this kind of game. Knuth [1] stressed two games of this kind, Mastermind and "Bulls and Cows," and demonstrated a strategy for the Mastermind game that requires at most five guesses in the worst case and 4.478 in the expected case. Later, Irving [2] and Nerwirth [3] used sophisticated heuristic strategies to improve the bounds in the expected case to 4.369 and 4.364, respectively. Finally, Koyama and Lai [4] used a recursive backtracking method to determine the optimal strategy for Mastermind, where the expected number of guesses is 4.34. Also, variants of the Mastermind game have been studied in [5, 6], and [7]. Furthermore, in [8, 9] and [10], the authors used evolutionary algorithms and genetic algorithms to solve related problems. Roche [11] analyzed the generalized Mastermind and obtained asymptotical bounds under some conditions. Kabatianski and Thorpe [12] investigated the Mastermind game and its related applications based on coding theory. More recently, a graph-partition approach was introduced to determine the optimal strategies for various games with two-digit secret code [14, 15, 16].

The rule of "Bulls and Cows" [1] (or called AB game in Asia) is described as follows. The codemaker chooses a secret code, e.g., (s_1, s_2, s_3, s_4), consisting 4 digits out of 10 symbols, whereas repetition of symbols is prohibited. Hence, the set of possible codes is the number of permutations, P(10,4)= 10*9*8*7=5040. After each guess (g_1, g_2, g_3, g_4) made by the 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 $s_j \neq g_j$ but $s_j=g_k$ for some position $k \neq j$. For example, if the secret code is (1, 2, 3, 4), then the responses for the guesses (3, 1, 5, 4), (3, 1, 4, 5) are [1, 2], [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.

Since the search space for "Bulls and Cows" is extremely large, in the past no optimal strategy for the game has yet been found. In this paper, we develop new and systematic optimization approaches, *pigeonhole-principle-based backtracking* and *k-way-branching approaches*, to discover an optimal strategy for the game in the worst case and a near-optimal strategy in the expected case.

This paper is organized as follows. In Section 2 we describe the properties of the deductive games. Section 3 develops the *k*-way-branching algorithm for the “Bulls and Cows.” Also, we apply the *k*-way-branching algorithm to the Mastermind game and compare our results with previously published results. In Section 4, we introduce an *extended pigeonhole principle* and demonstrate the *pigeonhole-principle-based backtracking* approach to determine the minimal number of guesses required for “Bulls and Cows” in the worst case. Section 5 presents our conclusions.

2. Properties of deductive games

In this section, we introduce some properties of deductive games by a simple *number guessing game*, denoted 1×*n* games. We will show how to determine the numbers of guesses required in the expected and the worst case for the game. By means of this comparatively simple work, we present some fundamental concepts that can be applied to develop optimal strategies for generalized deductive games.

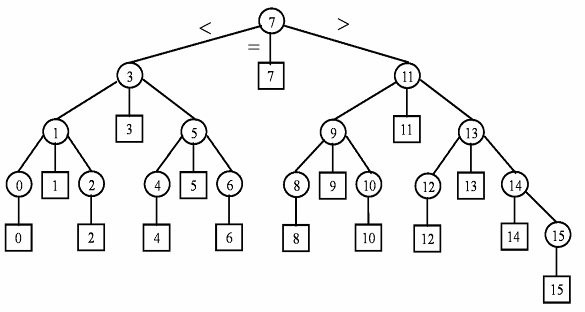


Fig. 1. A game tree for a 1×16 game, where the binary search strategy is used.

In the 1×*n* games, the codemaker chooses a *secret number S*, $S \in \{0, 1, 2, \ldots, n-1\}$. After each guess g_i made by the codebreaker, the codemaker responds with a hint H_i, $H_i \in \{<, =, >\}$, three elements of 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 into a search problem. We can obtain the optimum strategy for this game by using the *binary search* technique. In order to demonstrate how to calculate the number of guesses required in the worst and expected cases for 1×n games, we illustrate the binary search strategy by means of a 1×16 game, the game tree for which is shown in Fig. 1. From this strategy, we can easily obtain the following two observations, which can be applied to analyze arbitrary deductive games.

Observation 1. The number of guesses required in *the worst case* for a game is *H*, 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*=5 in Fig. 1.

Observation 2. The number of guesses required in *the expected case* for a game is *L*/n, where *L* is the *external path length* [13] of the game tree, i.e., the sum of the distances from the root to each leaf in the game tree. For example, in Fig. 1, $L= 1*1 +2*2 +3*4 +4*8+5*1=54$ and the number of guesses required in the expected case is $L/n=54/16 =3.375$.

From Observations 1 and 2, we should minimize *the height H* and *the external path length L* of the game tree to find the optimum strategy for a game in the worst and expected cases, respectively.

3. k-way-branching (KWB) approaches

For a game tree with height *H* and branching factor *b*, the search space is b^H. As the value *b* becomes larger, the game tree will become too huge to deal with. The fundamental idea of our *k*-way-branching algorithm is to reduce the search space from b^H to k^H. For example, the search space for “Bulls and Cows” is reduced from $(14*5040)^7$ to $(14*k)^7$.

The KWB algorithm can be implemented by a *modified exhaustive depth-first search* on the game tree. The main modification to depth-first search is that at each visited node we consider only *k* guesses that are most likely to obtain the optimal strategy, instead of all possible guesses. We use a heuristic procedure, shown in Fig. 2, to select the *k* guesses. That is, we only explore the *k* “most likely” best guesses and ignore the other (5040 – *k*) guesses at each stage. Depending on the execution time and space allowed, we can increase *k* to obtain results that are getting closer to the optima.

The heuristic procedure in Fig. 2 is based on the heuristics used in [1], which chooses the guess that minimizes the maximum number of remaining candidates at each stage. However, our strategy improves on the heuristics in two ways. First, we consider the distribution of all remaining candidates in each class, rather than only the largest class. Second, we expand *k* most-likely-best guesses at each stage to get more promising results. To choose the “*k* most likely” best guesses, we introduce an efficient clustering technique, *hash collision group (HCG)*.

Hash collision group (HCG). We classify the guesses into different HCGs by a given *hash function*. As an example of our procedure, if two guesses result in the same distribution of the numbers of remaining candidates for the 14 response classes, they will be

Table 1. The numbers of the remaining candidates of the 14 response classes after the first guess.

Class	[4,0]	[3,0]	[2,2]	[2,1]	[2,0]	[1,3]	[1,2]	[1,1]	[1,0]	[0,4]	[0,3]	[0,2]	[0,1]	[0,0]	Total
Size	1	24	6	72	180	8	216	720	480	9	264	1260	1440	360	5040

1. Perform each of all the 5040 possible guesses on the visited node.
Then, all the possible candidates in the visited node will be partitioned into 14 response classes after each guess. Let the numbers of remaining candidates of the 14 *response classes* resulted from each of the guesses be $C_{g,1}$, $C_{g,2}$, …, $C_{g,14}$, $1 \le g \le 5040$.

2. Hash each of the 5040 sequences, $C_g = \langle C_{g,1}, C_{g,2}, \ldots, C_{g,14} \rangle$, $1 \le g \le 5040$, to a nonincreasing sequence, $C'_g = \langle C'_{g,1}, C'_{g,2}, \ldots, C'_{g,14} \rangle$, and classify the 5040 C'_gs into *Hash Collision Groups* (HCGs).

3. Choose k representative HCGs with higher average distribution.
That is, select k HCGs with the smallest $|C'_g|$s, where $|C'_g|$ denotes a measurement function of C'_g; i.e., $|C'_r|=|C'_s|$ denotes $C'_{r,i} = C'_{s,i}$, $1\le i \le 14$, and $|C'_r|<|C'_s|$ denotes $C'_{r,1} < C'_{s,1}$,or ($C'_{r,i} = C'_{s,i}$ and $C'_{r,j} < C'_{s,j}$), $1\le i < j$ for some $j \le 14$.

4. Choose a representative guess from each representative HCG, by the following rule:
If a guess *hits* one of possible candidates, then choose it as the representative guess, else arbitrarily choose a guess from each HCG.

Fig. 2. A heuristic procedure to choose k best guesses at each node in a game tree

classified as the same HCG. More specifically, let the numbers of remaining candidates of the 14 response classes after a guess g be $C_g = \langle C_{g,1}, C_{g,2}, \ldots, C_{g,14} \rangle$. We define the *hash function*:

$$Hash\ (C_g = \langle C_{g,1}, C_{g,2}, \ldots, C_{g,14} \rangle) = (C'_g = \langle C'_{g,1}, C'_{g,2}, \ldots, C'_{g,14} \rangle), \text{ where } C'_{g,1} \ge C'_{g,2} \ge \ldots \ge C'_{g,14}.$$

That is, the hash function sorts the original sequence C_g into a nonincreasing sequence C'_g. For example, Table 1 shows the number of remaining candidates of the 14 response classes after the first guess g = (0, 1, 2, 3). We have $C_g = \langle 1, 24, 6, 72, 180, 8, 216, 720, 480, 9, 264, 1260, 1440, 360 \rangle$ and $C'_g = \langle 1440, 1260, 720, 480, 360, 264, 216, 180, 72, 24, 9, 8, 6, 1 \rangle$. If two nonincreasing sequences C'_r and C'_s are the same, that is, $C'_{r,1}=C'_{s,1}$, $C'_{r,2}=C'_{s,2}$, …, and $C'_{r,14}=C'_{s,14}$, then guess r and guess s are classified into the same HCG.

Experimental Results. To estimate how well the k-way-branching algorithm can perform, we first apply it to the game of Mastermind, which is a smaller game, since the optimum strategy for the game has been determined in [4]. Then the experimental results for the game of interest, "Bulls and Cows," will be given.

A. Mastermind. For the Mastermind game, Knuth [1] used a heuristic strategy that chooses the guess that minimizes the maximum number of the remaining possibilities at every stage, for which the number of guesses required in the expected case is 4.478. Neuwirth [3] used a sophisticated heuristic method to improve the bound to 4.364, which is the best "heuristic" strategy in literature. Finally, Koyama and Lai [4] used a recursive backtracking method to determine the optimum result, 4.34. However, its run time could be several days but has not been reported.

A comparison of the previous results and our results is shown in Table 2. Our results were obtained by running our program on a Pentium III 750 MHz computer. As seen in Table 2, when k=1, our result is within 97.38 % of optimum and is better than the result [1] in the expected case. Moreover, when k=40, our results outperform the best results of previous heuristic strategy [3]. In addition, observing the bottom row of Table 2, even when k=40, the execution time is still acceptable, so our KWB algorithm is quite efficient.

B. "Bulls and Cows": By using the efficient KWB algorithm, we obtain a near-optimal strategy for the game, where the number of guesses is $26605/5040 \fallingdotseq 5.279$ in the expected case and 7 in the worst case when k=5. (The program was run on a Pentium III 750 MHz computer for about 40 minutes.) That is, the program can build a game tree with *height H*=7 and *external path length L*=26605.

Table 2. Comparison of our results and previous results for the Mastermind game.

Methods		Total number of guesses for all candidates (external path length L)	Number of guesses in the expected case ($L/6^4$)	% of optimum (**5625**/L)	Execution time (seconds)
Knuth [1]		5803	4.4776	96.93%	N/A
Neuwirth [3]		5656	4.3642	99.45%	N/A
Koyama & Lai [4]		**5625**	4.3403	100%	N/A
Our method	k=1	5776	4.45679	97.39%	5
	k=5	5748	4.435185	97.86%	27
	k=10	5711	4.406636	98.49%	179
	k=20	5669	4.374228	99.22%	656
	k=30	5659	4.366512	99.40%	1508
	k=40	5654	4.362654	99.49%	2637

4. Pigeonhole-principle based backtracking

In this section, we will demonstrate a computer-aided verification method to solve the problem: *the minimum number of guesses required for "Bulls and Cows" in the worst case.* To make the backtracking algorithms more efficient, we introduce an *extended pigeonhole principle* in Lemma 1. The principle can be applied to the situation where pigeonholes have different volumes. Let the set of the volumes $V=\{v_1, v_2, v_3, \ldots, v_n\}$ and a proper subset of V, $V'=\{v'_1, v'_2, v'_3, \ldots, v'_k\}$, where $1\le k<n$, i.e., $V' \subset V$. We have the following lemma.

Lemma 1. (Extended pigeonhole principle) If m pigeons occupy n pigeonholes with different volumes and $m > \sum_{i=1}^{k} v'_i$, then there exists one pigeonhole with at least $\lceil (m-\sum_{i=1}^{k} v'_i)/(n-k) \rceil$ pigeons roosting in it, where $1 \le k < n$.

Proof. Given $m > \sum_{i=1}^{k} v'_i$, we divide the proof into two cases:

Case 1. If pigeons fill up the *k* pigeonholes whose respective volumes belong to *V'*, then $(m-\sum_{i=1}^{k} v'_i)$ remaining pigeons have to be distributed among the other (*n-k*) pigeonholes.

Case 2. Otherwise, there are more than $(m-\sum_{i=1}^{k} v'_i)$ remaining pigeons that have to be distributed among the other (*n-k*) pigeonholes.

Therefore, by *the generalized pigeonhole principle*, in both cases there exists one pigeonhole with at least $\lceil (m-\sum_{i=1}^{k} v'_i)/(n-k) \rceil$ pigeons roosting in it. This completes the proof. ■

We now derive the total number of guesses required to hit two remaining candidates in Lemma 2. This lemma not only can be applied to arbitrary games of this kind but also is useful to derive the exact bound for "Bulls and Cows."

Lemma 2. If a subtree remains only two candidates, then the minimum external path length *L* and the minimal height *H* of the game tree will be increased by 3 and 2, respectively.

Proof. We will illustrate the result with an example, and omit the detailed proof here. If there are only two candidates in a subtree, it is obvious that the optimal strategy is to guess one of the candidates first and then guess the other. For example in Fig.1, if there are only two remaining candidates, say, 14 and 15, in the subtree that contains nodes 14 and 15, an optimal strategy is to guess 14 first and then 15. Hence, the *L* will be increased by 1 (for guessing 14) + 2 (for guessing 15) = 3, and *H* will be increased by 2 (for guessing 15). ■

At the first guess for "Bulls and Cows," without losing the generality, we can guess any one, say (g_1, g_2, g_3, g_4), of the 5040 possible codes. All of these guesses result in the same distribution of the remaining candidates since all of the guesses are equivalent. The distribution is shown in Table 1 of Section 3, where the size of each class can easily be calculated. For example, after the first guess (0,1,2,3), the size of class [3,0] is 24 because the remaining candidates in class [3,0] are (0,1,2,y), (0,1,y,3), (0,y,2,3), and (y,1,2,3), where y ∈ {4,5,6,7,8,9}, i.e., unused digits. Note that the size of each class shown in Table 1 is the maximum possible size during the game-guessing process because we obtained it from partitioning all 5040 possible candidates. Obviously, during the following game-guessing process, at any stage each class [3,0] will not have more than 24 remaining candidates. Therefore, the sizes of the 14 classes can be considered to be the volumes of the corresponding 14 pigeonholes. Now we will apply *the extended pigeonhole principle* to prove that there exists one response class that contains at least some amount of the remaining candidates at each stage.

Lemma 3. (Lower bound Lemma) The height of the game tree for "Bulls and Cows" is at least 6.

Proof. As shown in Table 1, after the first guess, class [0,1] contains 1440 remaining candidates. After the second guess, the numbers of these 1440 candidates classified to the six classes [4,0], [3,0], [2,2], [2,1], [1,3], and [0,4] are at most 1, 24, 6, 72, 8, and 9, respectively, as shown in Table 1. Hence, at least one class other than the six classes contains at least $\lceil [1440-(1+24+6+8+9+72)]/(14-6) \rceil = 165$ remaining candidates after the second guess.

In a similar way, we can obtain the result that at least one class contains more than $\lceil [165-(1+6+8+9)]/(14-4) \rceil = \lceil 14.1 \rceil = 15$ remaining candidates after the third guess.

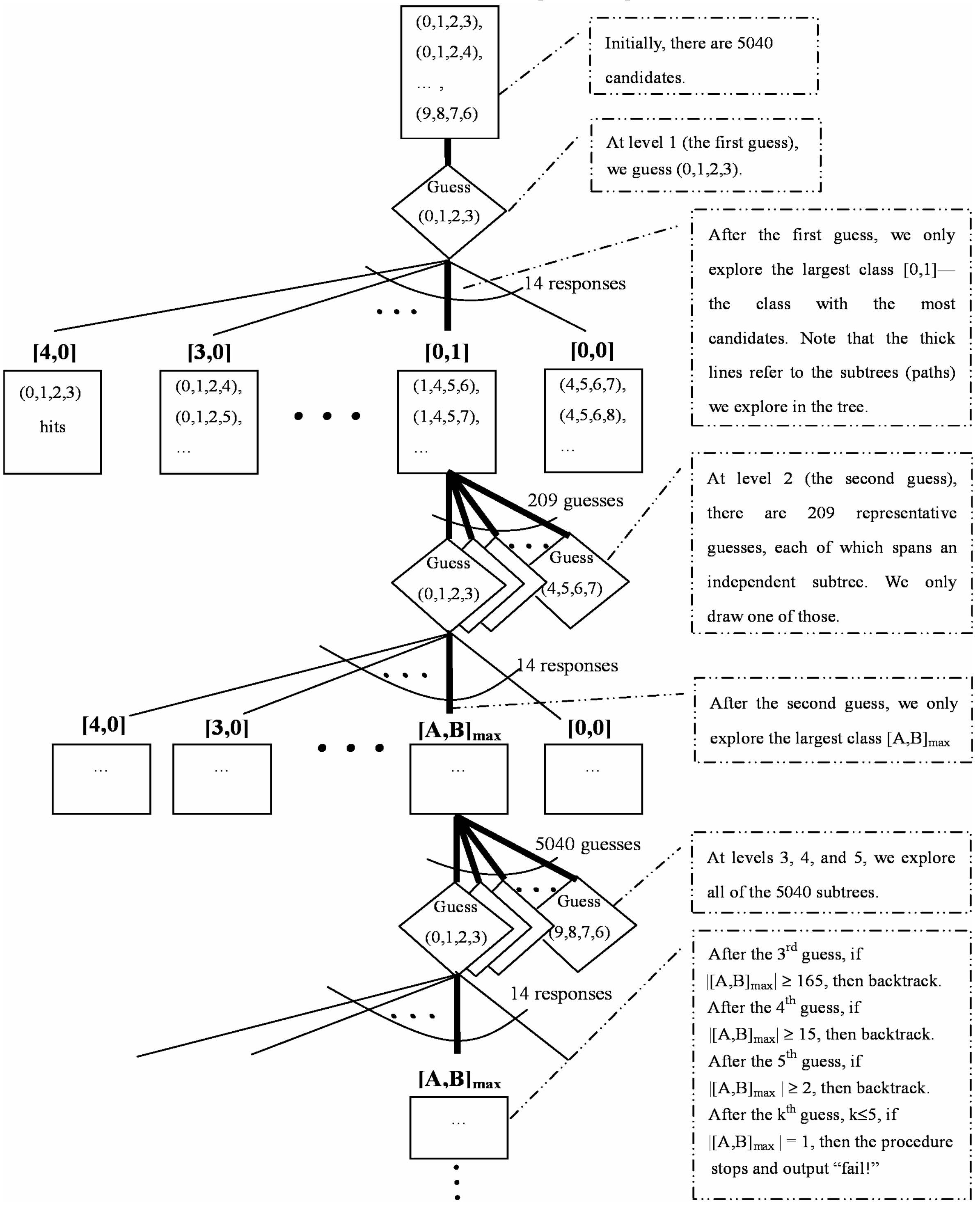
After the fourth guess, since there are 14 classes, at least one class contains at least $\lceil 15/14 \rceil = 2$ remaining candidates.

Finally, by Lemma 2, we need one and two extra guesses to hit these 2 remaining candidates, respectively. Therefore, at least 6 guesses are required in the worst case, and hence the height of the game tree for "Bulls and Cows" is at least 6. ■

The above lemma only shows a looser lower bound 6 for this problem. Now we demonstrate the pigeonhole-principle-based backtracking for verifying that the height of the game tree is at least 7. Fig. 3 shows the verification procedure, the correctness of which will be proven in Lemma 4.

```
1. Do a depth-first search of the worst-first search
   tree as illustrated in Fig. 4.
2. After each guess, the procedure backtracks to the
   node's parent, or stops and outputs "fail!",
   according to the following 4 cases, where the
   notation $|[A,B]_{max}|$ denotes the number of the
   remaining candidates in $[A,B]_{max}$.
     Case 1. After the 3rd guess, if $|[A,B]_{max}| \ge 165$,
             then backtrack.
     Case 2. After the 4th guess, if $|[A,B]_{max}| \ge 15$,
             then backtrack.
     Case 3. After the 5th guess, if $|[A,B]_{max}| \ge 2$,
             then backtrack.
     Case 4. After the kth guess, $k \le 5$, if $|[A,B]_{max}| = 1$,
             then the procedure stops and outputs
             "fail!".
3. If the procedure terminates and Case 4 has never
   happened, then outputs "pass!"
```

Fig. 3. The verification procedure.



Initially, there are 5040 candidates.

At level 1 (the first guess), we guess (0,1,2,3).

After the first guess, we only explore the largest class [0,1]— the class with the most candidates. Note that the thick lines refer to the subtrees (paths) we explore in the tree.

At level 2 (the second guess), there are 209 representative guesses, each of which spans an independent subtree. We only draw one of those.

After the second guess, we only explore the largest class $[A,B]_{max}$

At levels 3, 4, and 5, we explore all of the 5040 subtrees.

After the 3rd guess, if $|[A,B]_{max}| \geq 165$, then backtrack.
After the 4th guess, if $|[A,B]_{max}| \geq 15$, then backtrack.
After the 5th guess, if $|[A,B]_{max}| \geq 2$, then backtrack.
After the kth guess, $k \leq 5$, if $|[A,B]_{max}| = 1$, then the procedure stops and output "fail!"

Fig. 4. The worst-first search tree

Lemma 4. *If the output of the verification procedure is "pass!", then seven guesses are necessary for "Bulls and Cows" in the worst case.*

Proof. The main idea to prove this lemma is that the height of the game tree is at least 7. The four conditions used in Step 2 of the verification procedure are from Lemma 3. That is, only for Case 4, the height of the game tree would be less than or equal to 6. In this situation, the procedure would stop and output "fail!" On the other hand, after the fifth guess, if there exists a class with 2 remaining candidates, then the height of the game tree would be 5+2=7. In the same way, we can derive the constraints for Cases 1, 2, and, 3. If all strategies are constrained by Cases 1, 2, and 3, then the height of the game tree would be at least 7. That is, the number of guesses is at least 7 for “Bulls and Cows” in the worst case. ■

Now, we can obtain the exact bound of the minimal number of guesses required for the “Bulls and Cows” in the worst case, as shown in the following Theorem.

Theorem 1. Seven guesses are necessary and sufficient for the “Bulls and Cows” in the worst case.
Proof. *Necessary*. “Bulls and Cows” has successfully passed the verification procedure shown in Fig. 3, in which the numbers of backtrackings caused by Cases 1, 2, 3 and 4 were 407528, 3254981544, 59149440, and 0, respectively. The verification program was run on a Pentium III 750 computer for about two days. From Lemma 4, we conclude that seven guesses are necessary for “Bulls and Cows” in the worst case.
Sufficient. In Section 3, the program, a *k*-way-branching, has built a game tree with *height H*=7 and *external path length L*=26605 for “Bulls and Cows”. From Observation 1, we conclude that seven guesses are sufficient for “Bulls and Cows” in the worst case.

5. Conclusions

In this paper, we propose new algorithms for deductive games and have successfully applied it to solve “Bulls and Cows.” By using the *k*-way-branching algorithm, we can obtain an optimal strategy in the worst case and a near-optimal strategy in the expected case. The near-optimal result can approach the optimal solution by increasing the parameter *k*. Furthermore, we introduce the *extended pigeonhole principle* and have applied it to effectively reduce the search space of the problem. Therefore, we are able to determine the minimum number of guesses required for “Bulls and Cows” in the worst case.

We use the concept of *hash collision groups* in our KWB algorithm because the *k* groups can be efficiently obtained and near-optimal results can be achieved even using a small value for *k*. Depending on the characteristics of each problem, the critical issue is how to define the most adaptive *hash collision groups* for the problem in hand. The proposed algorithms can be applied to related problems. We hope this paper will prompt researchers to study other related problems.

Acknowledgments

The research is supported in part by NSC93-2213-E-003-001, R.O.C.

References

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

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

[3] Neuwirth, E. (1982) “Some strategies for Mastermind,” *Zeitschrift fur Operations Research,* 26, pp. 257-278.

[4] Koyama, K. Lai, T. W. (1993). “An optimal Mastermind strategy,” *Journal of Recreational Mathematics,* 25, pp. 251-256.

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

[6] Ko, K.-I. Teng, S.-C.. (1986). “On the number of queries necessary to identify a permutation,” *Journal of Algorithms,* 7, pp. 449-462.

[7] Chen, Zhixiang and Cunha, Carlos (1996). “Finding a hidden code by asking questions,” *COCOON’96, Computing and Combinatorics,* pp. 50-55.

[8] Bento, L. Pereira, L. Rosa, A. (1999). “Mastermind by evolutionary algorithms,” in *Proceedings of the International Symposium on Applied Computing,* pp. 307-311.

[9] Bernier, J. L. Herraiz, C. I., Merelo, J. J., Olmeda, S., and Prieto, A. (1996). “Solving Mastermind using Gas and simulated annealing: a case of dynamic constraint optimization,” in *Proceedings PPSN, Parallel Problem Solving from Nature IV, in Computer Science,* 1141, pp. 554-563.

[10] Merelo, J. J. Carpio, J. Castillo, P. Rivas, V. M. Romero, G. GeNeura Team. (1999). “Finding a needle in a haystack using hints and evolutionary computation: the case of Genetic Mastermind,” *Genetic and Evolutionary Computation Conference late breaking papers books*, pp. 184-192.

[11] Roche, J. R. (1997). “The value of adaptive questions in generalized Mastermind,” in *Proceedings of the International Symposium on Information Theory IEEE,* pp. 135-135.

[12] Kabatianski, G. Lebedev, V. (2000). “The Mastermind game and the rigidity of the Hamming space,” in *Proceedings of the International Symposium on Information Theory IEEE*, pp. 375-375.

[13] Sedgewick, R. (1988). *Algorithms,* second edition, Addison-Wesley.

[14] Chen, S. T., Hsu, S. H., and Lin, S. S. (2002).“An Optimal Strategy for 2×n AB Games,” in *Proceedings of the 2002 International Computer Symposium on Algorithms and Computational Molecular Biology,* C2-2.

[15] Chen, S. T. and Lin, S. S. (2004). “Optimal Algorithms for 2×n AB Games- a Graph-partition Approach.” *Journal of Information Science and Engineering,* **20:1**, pp. 105-126.

[16] Chen, S. T. and Lin, S. S. (2004). “Optimal Algorithms for 2×n Mastermind Games- a Graph-partition Approach.” *Computer Journal*, 47: 5, pp. 602-611.