An algorithm for the shifting checkers problem

Daxin Zhu[1, a], Xiaodong Wang[2, b,*]

[1]Quanzhou Normal University, 362000 Quanzhou, Fujian, China.

[2]Fujian University of Technology, Fuzhou, 350108 Fujian, China.

[a]email: dex@qztc.edu.cn, [b]wangxd@139.com, [*]Corresponding author

Keywords: Moving Checkers; optimal algorithm; Fibonacci number

Abstract. In this paper, we study Moving Checkers Game, an interesting shifting checkers game consisting of *n* black checkers and 1 white checkers. We have proved that the minimum number of steps needed to play the game for general n is 2*n*+1. We have also presented an optimal algorithm to generate all of the optimal solutions in linear time for very large size. The number of solutions for the game of size n is the (*n*+2)th Fibonacci number.

Introduction

Moving Checkers Game is an interesting shifting the checkers game consisting of n black checkers and 1 white checkers, where the aims is to find a solution in the smallest number of steps [1,2,3,10]. The $n+2$ positions of the row are numbered $1,\cdots,n+2$. Initially, the n black checkers are put in the position $1,\cdots,n$, and the white checker is put in the position $O(1)$. The position $n+1$ is initially vacant. In the final state of the game, the leftmost position is occupied by the white checker, and the right most n positions numbered $3,\cdots,n+2$ are occupied by black checkers, leaving the position 2 vacant.

If we denote a black checker by b, a white checker by w, and the vacant position by O, then any status of the checker board can be specified by a sequence consisting of characters b,w and O. The problem is then equivalent to transforming the initial sequence $\overbrace{b\cdots b}^{n}Ow$ to the sequence $wO\overbrace{b\cdots b}^{n}$ in the minimum number of steps. There are only two permissible types of moves. A move of the game consists of sliding one checker into the current vacant position, or jumping over the adjacent checker into the current vacant position. The goal of the game is to make a small number of moves to reach the final state of the game. We are interested in algorithms which, given an integer n, generate the corresponding move sequences to reach the final state of the game with the smallest number of steps. In this paper, we present an optimal algorithm to generate all of the optimal move sequences of the game consisting of n black checkers and 1 white checker.

Preliminaries

In this section, we will investigate some properties of the Moving Checkers Game. We will discuss the game in a more general setting [4-9]. In a general moving checkers game, there are n black checkers and m white checkers put on a table from left to right in a row. The $n+m+1$ positions of the row are numbered $1,\cdots,n+m+1$. Initially, the n black checkers are put in the position $1,\cdots,n$, and the m white checkers are put in the position $n+2,\cdots,n+m+1$. The position $n+1$ is initially vacant. In the final state of the game, the left most m positions numbered $1,\cdots,m$ are occupied by white checkers, and the right most n positions numbered $m+2,\cdots,m+n+1$ are occupied by black checkers, leaving the position $m+1$ vacant.

If we denote a black checker by b, a white checker by w, and the vacant position by O, then any status of the checker board can be specified by a sequence consisting of characters b,w and

O. The problem is then equivalent to transforming the initial sequence $\overbrace{b\cdots b}^{n}O\overbrace{w\cdots w}^{m}$ to the sequence $\overbrace{w\cdots w}^{m}O\overbrace{b\cdots b}^{n}$ in the minimum number of steps[5]. There are only two permissible types of moves. A move of the game consists of sliding one checker into the current vacant position, or jumping over the adjacent checker into the current vacant position.

In a row of checkers of the game, if two checkers have different colors and the black checker is on the left of the white checker, then the two checkers are called an inversion pair.

Of the two types of checker moves, we can further list 12 different cases of the moves into a table, as shown in Table 1. Sliding a black checker right into the current vacant position is denoted as $slide(b,r)$. The other three moves $slide(b,l)$, $slide(w,r)$, and $slide(w,l)$ are defined similarly. Jumping a black checker right over the adjacent white checker into the current vacant position is denoted as $jump(b,w,r)$. The other 7 moves $jump(b,w,l)$, $jump(w,b,r)$, $jump(w,b,l)$, $jump(b,b,r)$, $jump(b,b,l)$, $jump(w,w,r)$, and $jump(w,w,l)$ are defined similarly. These 12 cases of moves are numbered from 1 to 12.

The column Inversions of Table 1 denote the inversion increment of the checker row when the corresponding case of moves applied. Similarly, the column V-Inversions of Table 1 denotes the vacant inversion increment of the checker row when the corresponding case of moves applied.

It is not difficult to verify the following facts on the optimal solutions to play the game.

Lemma 1 *Any optimal solution for playing the game of shifting the checkers with minimum number of moves consists of only the classes of moves numbered from 1 to 4 in Table 1.*

Table 1: All cases of checker moves

No.	*Move*	*Change*	*Inversions*	*V-Inversions*
1	$slide(b,r)$	● □ → □ ●	0	-1
2	$slide(w,l)$	□ ○ → ○ □	0	-1
3	$jump(b,w,r)$	● ○ □ → □ ● ○	-1	0
4	$jump(b,w,l)$	□ ● ○ → ○ ● □	-1	0
5	$jump(w,b,r)$	○ ● □ → □ ● ○	1	0
6	$jump(w,b,l)$	□ ○ ● → ● ○ □	1	0
7	$slide(b,l)$	□ ● → ● □	0	1
8	$slide(w,r)$	○ □ → □ ○	0	1
9	$jump(w,w,r)$	○ ○ □ → □ ○ ○	0	2
10	$jump(b,b,l)$	□ ● ● → ● ● □	0	2
11	$jump(w,w,l)$	□ ○ ○ → ○ ○ □	0	-2
12	$jump(b,b,r)$	● ● □ → □ ● ●	0	-2

From Lemma 1, we can conclude that the following theorem holds.

Theorem 1 *For the general game of shifting the checkers consisting of n black checkers and m white checkers, it needs at least $nm+n+m$ steps to reach the final state of the game from its initial state.*

Since there are only 4 possible moves $slide(w,l)$, $slide(b,r)$, $jump(b,w,l)$, and $jump(b,w,r)$, we can simplify our notation for these 4 moves to $slide(l)$, $slide(r)$, $jump(l)$, and $jump(r)$ in the following discussions. According to Theorem 1, if we can find a move sequence to reach the final state of the game with $nm+n+m$ steps, then the sequence will be an optimal move sequence, since no move sequence can reach the final state of the game in less than $nm+n+m$ steps.

An Optimal Algorithm

In the following sections, we will study the moving checkers game for the special case of $m=1$. In this case we need at least $2n+1$ steps to reach the final state $wO\overbrace{b\cdots b}^{n}$ from its initial state $\overbrace{b\cdots b}^{n}Ow$. There are only two different optimal solutions for the special case of $n=1$.

In general cases of n, we can denote our problem $shift(n)$ as the problem to transform the initial game board $\overbrace{b\cdots b}^{n}Ow$ to the goal game board $wO\overbrace{b\cdots b}^{n}$ according to move rules. For the general problem $shift(n)$, there are only 2 choices of the first move from the initial game board. If $slide(l)$ is chosen as the first move, then the next move $jump(r)$ must be mandatory according to Lemma 1. In this case, the game board is changed to $\overbrace{b\cdots b}^{n-1}Owb$, or equivalently, our problem is reduced to $shift(n-1)$. If $slide(r)$ is chosen as the first move, then the following 3 steps $jump(l)$, $slide(r)$ and $jump(r)$ are mandatory according to Lemma 1. In this case, the game board is changed to $\overbrace{b\cdots b}^{n-2}Owbb$, or equivalently, our problem is reduced to $shift(n-2)$. After $2n$ moves, the game board can be changed to $Owb\overbrace{\cdots b}^{n}$. One more move $slide(l)$ will reach the goal game board $wO\overbrace{b\cdots b}^{n}$. The total number of moves is $2n+1$, and thus the move sequences are optimal.

Based on the discussions above, we can design a recursive algorithm to generate all optimal move sequences of problem $shift(n)$.

The optimal solution found by the algorithm shift$n$ can be presented by a vector x. For $i=1,2,\cdots,2n+1$, the step i of the optimal move sequence is given by x_i. This means that the checker located at position x_i will be moved in step i to the current vacant positions and leaving the positions x_i the new vacant positions. This can also be viewed that x is a function of i, which is called a move function. In the next section we will discuss the explicit expression of function x.

If we denote $x_0 = n+1$ and

$$d_i = x_{i-1} - x_i, 1 \le i \le 2n+1 \qquad (1)$$

Then the vector d will be a move direction function of the corresponding move sequence.

A related function t can then be defined as $t_i = \sum_{j=1}^{i} d_j, 1 \le i \le 2n+1$.

Since

$$t_i = \sum_{j=1}^{i} d_j = \sum_{j=1}^{i} (x_{j-1} - x_j) = x_0 - x_i = n+1-x_i$$

We have

$$x_i = n+1-t_i, 1 \le i \le 2n+1 \qquad (2)$$

Therefore, our task is equivalent to compute the function x or t efficiently.

Denote the number of different optimal move sequences of problem $shift(n)$ as $\rho(n)$, then according to Algorithm 3.1 we have,

$$\rho(n) = \begin{cases} 1 & n = 0 \\ 2 & n = 1 \\ \rho(n-1) + \rho(n-2) & n > 1 \end{cases} \qquad (3)$$

The solution of this recurrence is

$$\rho(n) = F_{n+2} \qquad (4)$$

where F_n is the nth Fibonacci number $\frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$.

For all of these different optimal solutions, we have to label them to identify each individual solution. In the recursion tree of the algorithm 3.1, whenever we have two choices *slide*(l) or *slide*(r), the tree edge corresponding to *slide*(l) is labeled 0, and the other tree edge corresponding to *slide*(r) is labeled 1. Every leaf node of the recursion tree corresponds to a different optimal move sequence of the game. The concatenation of the edge label on the path from the root to each leaf node is a binary string of digits 0 and 1. This binary string can be seen as an integer in its binary expression. In this manner, all of the different optimal solutions can be labeled with a unique integer.

It is obvious that the integer labels in $L(n)$ in sorted order are generally not consecutive. But we can show that

$$L(n) = \{k \mid 0 \le k \le 2^{n-1}, a(k) + \lfloor \log(k) \rfloor \le n\} \qquad (5)$$

where $a(k)$ is the number of 1's in the binary expansion of integer k, and $1+\lfloor \log(k) \rfloor$ is the length of integer k in its binary expansion.

For each label $k \in L(n)$, let its binary expansion be $k = \sum_{i=0}^{\lfloor \log(k) \rfloor} b_i 2^i$. In our algorithm, the meaning of bit b_i is two folds. If $b_i = 0$, then a *slide*(l) is chosen as the first move for the problem *shift*($n-i$), and then the problem is reduced to *shift*($n-i-1$). On the other hand, if $b_i = 1$, then a *slide*(r) is chosen as the first move for the problem *shift*($n-i$), and then the problem is reduced to *shift*($n-i-2$). Since the game will be finished when its size reduced to 0, we must have, $\sum_{i=0}^{\lfloor \log(k) \rfloor}(1+b_i) \le n+1$. It is equivalent to $a(k) + \lfloor \log(k) \rfloor \le n$. It is obvious that the largest label in $L(n)$ must be 2^{n-1}. Therefore, $L(n) \subseteq \{k \mid 0 \le k \le 2^{n-1}, a(k) + \lfloor \log(k) \rfloor \le n\}$.

On the other hand, for each integer $j \in \{k \mid 0 \le k \le 2^{n-1}, a(k) + \lfloor \log(k) \rfloor \le n\}$, if the binary expansion of j is used to guide the moves of the game, then the moves are well defined and the game will be finished since $a(j) + \lfloor \log(j) \rfloor \le n$. Therefore, we have, $L(n) \supseteq \{k \mid 0 \le k \le 2^{n-1}, a(k) + \lfloor \log(k) \rfloor \le n\}$.

Finally, we conclude that, $L(n) = \{k \mid 0 \le k \le 2^{n-1}, a(k) + \lfloor \log(k) \rfloor \le n\}$.

The Explicit Solutions to the Problem

For any fixed label $k \in L(n)$, let the optimal solution of the problem *shift*(n) labeled k be $x^k = (x_1^k, x_2^k, \cdots, x_{2n+1}^k)$. In this section, we will find an explicit solution to compute x_j^k, for all $k \in L(n), 1 \le j \le 2n+1$.

For the problem *shift*(n), let $k = \sum_{i=0}^{\lfloor \log(k) \rfloor} b_i 2^i$ and $1 \le j \le 2n+1$. If we define $b_i = 0$ for all $i > \lfloor \log(k) \rfloor$ then we have $k = \sum_{i=0}^{n} b_i 2^i$. In our algorithm, bit b_i is followed by $2(1+b_i)$ moves. So, for integer j, we have to find an integer $i = \alpha(k, j)$ such that in the solution labeled k the move j follows bit b_i. It is readily seen that $\alpha(k, j) = \min_{0 \le t \le n}\left\{t \middle| \sum_{s=0}^{t} 2(1+b_s) \ge j\right\}$. It is equivalent to

$$\alpha(k,j)=\min_{0\le t\le n}\left\{t\left|\sum_{s=0}^{t}b_s\ge j/2-t-1\right.\right\} \qquad (6)$$

Let

$$\beta(k,j)=2\sum_{s=0}^{j}(1+b_s)=2\left(1+j+\sum_{s=0}^{j}b_s\right) \qquad (7)$$

Then, from bit b_0 to bit b_j, a total of $\beta(k,j)$ moves have been played.

By using these two functions, we can give an explicit expression of x_j^k for all $k\in L(n),1\le j\le 2n+1$ as follows.

Theorem 2 *For the general game of shifting the checkers shift*(n) *consisting of* n *black checkers and 1 white checkers, there are total of* F_{n+2} *different optimal solutions. The optimal solutions can be labeled by* $L(n)$ *as shown in (5). For any* $k\in L(n)$, *let the solution labeled* k *be* $x^k=(x_1^k,x_2^k,\cdots,x_{2n+1}^k)$, *then for any* $k=\sum_{i=0}^{\lfloor\log(k)\rfloor}b_i2^i\in L(n),1\le j\le 2n+1$, x_j^k *can be computed by*

$$x_j^k=\begin{cases}\begin{cases}t/2-1 & : \quad j=t+1\\ t/2+1 & : \quad j>t+1\end{cases} & : \quad b_i=0\\ \begin{cases}t/2+1 & : \quad j=t+1\\ t/2-1 & : \quad j=t+2\\ t/2 & : \quad j=t+3\\ t/2+2 & : \quad j>t+3\end{cases} & : \quad b_i=1\end{cases} \qquad (8)$$

where $i=\alpha(k,j)$ *and* $t=\beta(k,i-1)$.

Proof.

For any $k\in L(n)$, we have $k=\sum_{i=0}^{n}b_i2^i$, where $b_i=0$ for all $i>\lfloor\log(k)\rfloor$. For any $1\le j\le 2n+1$ we can determine the index $i=\alpha(k,j)$ such that the move j follows bit b_i. Before bit b_i, $t=\beta(k,i-1)$ moves have been played. In bit b_i, move j is the $(j-t)$th move. If $b_i=0$, then the two moves followed are *slide*(l) and *jump*(r). These two moves contribute to move distances -1 and +2 respectively. Therefore, if $j-t=1$, then the move distance is -1. If $j-t=2$, then the move distance is -1+2=1. Similarly, if $b_i=1$, then the 4 moves followed are *slide*(r), *jump*(l), *slide*(r) and *jump*(r). These 4 moves contribute to move distances +1,-2,+1 and +2 respectively. In these cases, the move distances are +1,+1-2=-1,+1-2+1=0, and +1-2+1+2=2 respectively.

Before bit b_i, $t=\beta(k,i-1)$ moves have been played, and a distance $\sum_{s=0}^{i-1}(1+b_s)=t/2$ has moved. We finally conclude that if $b_i=0$, then $x_j^k=t/2-1$ when $j-t=1$, and $x_j^k=t/2+1$ when $j-t=2$. Similarly, if $b_i=1$, the values of x_j^k are $t/2-1,t/2+1,t/2$ and $t/2+2$, when $j-t=1,2,3$ and 4 respectively. ■

Conclusion

We have presented an optimal recursive construction algorithm for *Moving Checkers Game*. The algorithm can produce all of the optimal solutions in linear time for very large size n. The number of solutions for the game of size n is the $(n+2)$ th Fibonacci number $\frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^{n+2}-\left(\frac{1-\sqrt{5}}{2}\right)^{n+2}\right)$. In Section 4, an extremely simple explicit solution for each of the labeled optimal moving sequences of the general game is given. The formula gives for each individual step j, its optimal move in $O(1)$ time.

Another similar game is to reverse the n checkers numbered $1,\cdots,n$ by two permissible types of moves *slide* and *jump*. It is not clear whether our methods presented in this paper can be applied to this game. We will investigate the problem further.

Acknowledgement

This research was financially supported by the Natural Science Foundation of Fujian (Grant No.2013J01247), and Fujian Provincial Key Laboratory of Data-Intensive Computing and Fujian University Laboratory of Intelligent Computing and Information Processing.

References

[1] R. Bird, Pearls of Functional Algorithm Design, 258-274, Cambridge University Press, 2010.

[2] Erik D. Demaine, Playing games with algorithms, Algorithmic combinatorial game theory. Proceedings of the 26th Symposium on Mathematical Foundations in Computer Science, LNCS 2136, 18-32, 2001.

[3] Erik D. Demaine and Martin L. Demaine, Puzzles, Art, and Magic with Algorithms, Theory of Computing Systems, vol. 39, number 3, 473-481, 2006.

[4] J. Kleinberg, E. Tardos. Algorithm Design, 223-238, Addison Wesley, 2005.

[5] D.L. Kreher and D. Stinson, Combinatorial Algorithms: Generation, Enumeration and Search, 125-133, CRC Press, 1998.

[6] A. Levitin and M. Levitin, Algorithmic Puzzles, 3-31, Oxford University Press, New York, 2011.

[7] John S. Gray, The shuttle puzzle ?? A lesson in problem solving, Journal of Computing in Higher Education, Volume 10, Issue 1, 56-70, 1998.

[8] S. Sukparungsee, Y. Areepong, Exact Average Run Length of Double Moving Control Chart, International Journal of Applied Mathematics & Statistics, Vol. 52, No. 2, 152-158, 2014.

[9] T. Yato and T. Seta. Complexity and completeness of finding another solution and its application to puzzles. IEICE Trans. Fundamentals E86-A(5):1052-1060, 2003.

[10] D. Zhu, L. Wang, J. Tian and X. Wang, An Algorithmic Solution for a Single Player Computer Game, International Journal of Applied Mathematics & Statistics, Vol. 52, No. 5, 21-28, 2014.