國立交通大學

資訊科學與工程研究所

碩士論文

硬幣移動問題的計算複雜度
On the Complexity of the Linear Sliding-Coin Puzzle

研 究 生：林庭宇

指導教授：蔡錫鈞　教授

中華民國 九十九年六月

硬 幣 移 動 問 題 的 計 算 複 雜 度
On the Complexity of the Linear Sliding-Coin Puzzle

研　究　生：林庭宇　　　　Student：Ting-Yu Lin

指導教授：蔡錫鈞　　　　Advisor：Shi-Chun Tsai

國 立 交 通 大 學
資 訊 科 學 與 工 程 研 究 所
碩 士 論 文

A Thesis
Submitted to Institute of Computer Science and Engineering
College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Computer Science

June 2010

Hsinchu, Taiwan, Republic of China

中華民國九十九年六月

On the Complexity of the Linear Sliding-Coin Puzzle

Ting-Yu Lin

June 2010

摘要

硬幣移動問題的初始盤面是一列硬幣，有 n 個五分錢硬幣接連排著 n 個一分錢硬幣。玩家必須重新排序這列硬幣，讓五分錢的硬幣和一分錢的硬幣交錯排列。每一回合，玩家可以把 k 個相鄰的硬幣移動到新的位置。在移動過程中，這 k 個硬幣的相對順序不能調動。

我們證明至少需要 n 回合，才能解決此問題，更針對參數為 $k = 2$ 和 $k = 3$ 的問題，設計了能產生最佳移法的演算法。此外，我們提出一套建構最佳解答的辦法，並成功套用於參數為 $k = 4$ 和 $k = 5$ 的問題。

Abstract

Consider a line of n nickels and n pennies with all nickels arranged to the left of all pennies, where $n \geq 3$. The puzzle asks the player to rearrange the coins such that nickels and pennies alternate in the line. In each move, the player is allowed to slide $k \geq 2$ adjacent coins to a new position without rotating.

We prove that it takes at least n moves to solve the puzzle, and present algorithms to generate the optimal solutions for $k = 2$ and $k = 3$. We also propose a framework to extend solutions, and apply it successfully to construct optimal solutions for $k = 4$ and $k = 5$.

誌謝

感謝我的指導老師蔡錫鈞教授，在我研究的過程及撰寫論文時，給了我許多有用的建議。在我面臨眾多人生未來可能的道路時，也爲我點出方向。感謝實驗室的各位學長姐，在研究、課業以及助教工作上的協助。感謝所有在新竹遇到的同學及朋友，有你們，才有我這些年精采的人生。最後，感謝我的父母全心全意地支持我，讓我毫無後顧之憂，完成我的學業。

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Preliminary: an old puzzle

Sliding-coin puzzle is a solitary game, which has numerous variations. The goal is to rearrange a configuration of coins into an other via a sequence of moves. The moves should be as few as possible. A puzzle is defined by a pair of initial and final configurations associated with a moving rule.

Let's begin with a particular puzzle. Consider a configuration on the left of Figure 1.1, with three nickels to the left of three pennies. We use white stone and black stone to denote the nickel and penny, respectively. The player is allowed to slide a pair of adjacent coins to unoccupied positions (without rotating) in each move. The goal is to reach either configuration on the right of Figure 1.1, alternate nickels and pennies in the line. The final configuration may not completely stay at the same positions along the line, i.e. it may shift to the left or right.

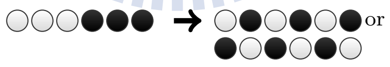


Figure 1.1: Rearrange the left configuration into either right configuration by sliding a pair of adjacent coins in each move. We use white stone and black stone to denote the nickel and penny, respectively.

This puzzle can be solved in three moves. We demonstrate a solution step by step in Figure 1.2. We write $(i \to j)$ to denote a move such that the two coins are slid from i and $i+1$ to j and $j+1$, respectively. Note that the player cannot rotate the pair of coins in transit. For example, we can slide 3 into 7 and 4 into 8, but we cannot slide 3 into 8 while 4 goes into 7.

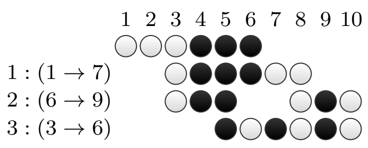


Figure 1.2: A three-move solution to the puzzle described in Figure 1.1.

Since all the coins are arranged in the line, we call this "vanilla" *linear sliding-coin puzzle*. In fact, this is an old puzzle. It has been illustrated as "Sheep and goats" by Mott-Smith [1], and as "empty and filled wine glasses" by Gardner [2]. Both puzzle books were published over half a century ago.

Actually, there are many sliding-coin puzzles whose configurations of coins are arranged on the plane. Two of the classic puzzles presented by Demaine et al. [4] are shown in Figure 1.3 and Figure 1.4. For more sliding-coin puzzles, please refer to [3, 6, 7, 8].

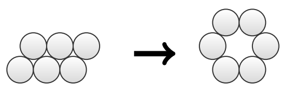


Figure 1.3: Rearrange the rectangle into the circle in three moves. In each move, a coin can be slid to a position adjacent to two other coins.

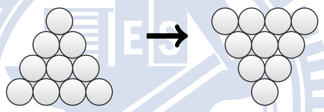


Figure 1.4: Turn the pyramid upside down in three moves. In each move, a coin can be slid to a position adjacent to two other coins.

Unlike the two-dimensional sliding-coin puzzles shown above, the linear one shown in Figure 1.1 can be easily generalized by adding the same number of white and black coins to the configuration or allowing the player to slide more coins in each move. These generalizations then form a family of puzzles. In this thesis, we study the linear sliding-coin puzzle family in an algorithmic view. We will define the problem formally in the next section.

1.2 Problem definition and notions

This section defines the linear sliding-coin puzzle family and other related concepts.

A *board* is a sequence of positions indexed with integers. A *position* can hold either a nickel or a penny, or be *unoccupied*. A *configuration* is a set of coin type and position pairs. We usually associate a configuration with its image, i.e. the arrangement of the coins on the board. The position of a configuration is the position of the leftmost coin in the configuration.

A *valid move* changes the positions of k adjacent coins to unoccupied positions without rotating. More specifically, a *valid move* slides k adjacent coins on source positions $i, i+1, \ldots, i+k-1$ to destination positions $j, j+1, \ldots, j+k-1$, respectively. This move is denoted $(i \to j)$. When the context is clear, a valid move is just called a "move."

The problem we study is generalized from the puzzle in Figure 1.1. The initial configuration is on the left of Figure 1.5, with n nickels to the left of n pennies for $n \geq 3$. We use white stone and black stone to denote the nickel and penny, respectively. There are two final configurations on the right of Figure 1.5, with alternate nickels and pennies in the line. The final configuration is called *white type* if the first coin is a nickel (white stone). Similarly, the final configuration is called *black type* if the first coin is a penny (black stone).

The goal of the puzzle is to rearrange the initial configuration into either final configuration via a sequence of valid moves. The final configuration does not need to stay at the same positions as the initial configuration along the line.

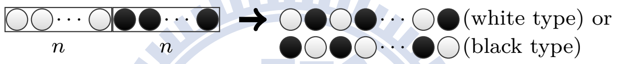


Figure 1.5: Rearrange the left configuration into either right configuration by sliding a fixed number $k \geq 2$ adjacent coins in each move.

k and n are two parameters of the puzzle. Once both k and n are fixed, a concrete puzzle is defined. For example, the puzzle shown in Figure 1.1 is defined by $k = 2$ and $n = 3$. For a solvable puzzle, it is necessary that $2n > k$, i.e. the number of coins in the configuration must be greater than the number of coins which can be slid in each move.

A *solution* is a sequence of valid moves which rearranges the line of coins from initial configuration to final configuration. An *n-move* solution is a solution consisting of n valid moves. Given a solution and the position of the initial configuration, an *image* of moving history can be generated, which consists of rows of configurations. For example, Figure 1.2 is an image of a solution for $k = 2$ and $n = 3$. We often refer to a solution by its image. Therefore when we refer to a coin in a solution, we mean the coin in the image of the solution.

A coin becomes *stationary* in a configuration of a solution if it is not slid by the rest of the moves anymore, i.e. it has reached the correct position as in the final configuration. For example, in Figure 1.2, the coin 5 in the initial configuration and the coin 8 in the second configuration are both stationary. If we say a coin is stationary in a solution, it is stationary from the beginning (in the initial configuration).

A solution is *optimal* if there is no other solution with fewer moves. It is common for a puzzle to have many optimal solutions.

A k-*block* is k adjacent positions in a configuration. A *pattern* is the k adjacent coins slid in a move. There are 2^k patterns.

1.3 Results

To the best of our knowledge, linear sliding-coin puzzle has not been solved algorithmically. Some primitive results are illustrated without proof of optimality [5].

In this thesis, we study the lower bound on the number of moves required to solve the puzzle. We divide the linear sliding-coin puzzle family by k and solve each sub-family for any n algorithmically.

Chapter 2 studies some general properties which are independent of k including the lower bound and the properties of n-move solutions. Chapter 3 studies a recursive algorithm for $k = 2$ which generates optimal solutions. Chapter 4 studies two algorithms for $k = 3$, one for odd n and the other for even n, since the behaviors of the two cases are different. In Chapter 5, we propose a framework to construct optimal solutions, and apply it successfully for $k = 4$ and $k = 5$. Chapter 6 concludes with some open problems.

Chapter 2

Lower bound and properties

This section studies some general properties which are independent of k, the number of coins allowed to slide in each move.

2.1 Lower bound

We show the minimum number of moves required to solve a puzzle. We define a *pair* to be a pair of adjacent white and black stones, i.e. ○● or ●○.

Theorem 2.1.1. *For $n \geq 3$, it takes at least n moves to solve any linear sliding-coin puzzle.*

Proof. We prove it by contradiction. Without loss of generality, suppose a puzzle can be solved in $n-1$ moves.

Observe that in the initial configuration, there is only one pair, i.e. the nth white stone and the $(n+1)$th black stone. The first move contributes at most one pair, and each following move creates at most two more such pairs. Thus, in $n-1$ moves, at most $1+1+2(n-2) = 2n-2$ pairs can be created. However there are $2n-1$ such pairs in either final configuration, which is a contradiction. □

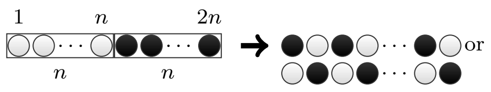


Figure 2.1: The initial configuration has one pair of alternate coins. Either final configuration has $2n-1$ pairs of alternate coins.

2.2 Necessity of n-move solutions

From Theorem 2.1.1, we know n moves are the minimum number of moves possible to solve the puzzle. Actually, in many cases, the puzzle is often solvable in optimal n moves.

We will see this in the following sections. We are interested in the properties of an n-move solution.

For example, suppose we are allowed to move k adjacent coins in each move. Then in an n-move solution to a puzzle of size n, there are $2n$ coins, and the total number of coins moved during the rearrangement is kn. Therefore each coin is moved $kn/2n = k/2$ times on average.

The rest of this section studies some necessary conditions for an n-move solution. In other words, if none of the conditions holds for a solution, the solution must have at least $n + 1$ moves. We first investigate the pairs created by moves in an n-move solution.

Proposition 2.2.1. *If a solution to a puzzle of size n has exactly n moves, then one of the two following conditions holds:*

1. *The first move creates 0 pair, and each of the rest $n - 1$ moves creates 2 pairs.*

2. *Two moves (including the first move) each creates 1 pair, and each of the rest $n - 2$ moves creates 2 pairs.*

Proof. We know that the initial configuration has 1 pair and the final configuration has $2n - 1$ pairs. If any moves destroys pairs, it must take more than n moves to solve the puzzle. Therefore if the solution has exactly n moves, it has no move which destroys pairs. According to the number of pairs created by the first move, there are two cases.

Case 1. Suppose the first move creates 0 pair. To achieve $2n - 1$ pairs in n moves, each of the rest $n - 1$ moves must create $((2n - 1) - 1 - 0)/(n - 1) = 2$ pairs.

Case 2. Suppose the first move creates 1 pair. Then $(2n - 1) - 1 - 1 = 2n - 3$ pairs must be created in $n - 1$ moves. Only one move is allowed to create 1 pair. Each of the rest $n - 2$ move must create 2 pairs to achieve to goal. □

Next, we investigate the k-blocks used by an n-move solution. Let M be the $2n$-block occupied by the $2n$ coins in the initial configuration.

Proposition 2.2.2. *In an n-move solution to a puzzle of size n, there is no gap between M and the k-blocks used by moves during the rearrangement. (See Figure 2.2.)*

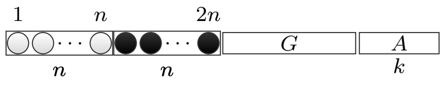


Figure 2.2: The gap in the configuration described in Proposition 2.2.2.

Proof. We prove it by contradiction. Let A be a k-block to the right of M, and there is a gap G between M and A, which consists of at least one positions. All the positions in

G are not used during the rearrangement. In the following paragraphs, the conditions to which we refer are in Proposition 2.2.1.

Without loss of generality, assume A is used by the first move. Then the first move creates 0 pair. There are the following two cases.

Case 1. Suppose at least two k-blocks are used during the rearrangement (including A). Then the first time the move using the second k-block (other than A) creates at most 1 pair, violating condition 1.

Case 2. Suppose A is the only k-block used during the rearrangement. Then in the final configuration, the $2n$ alternate coins must be at M. A cannot be a destination more than once, otherwise there are more than one moves which create 0 pair, violating condition 1. Let C be the k coins slid by the first move. C must be coins in alternating order in the initial configuration, otherwise C is unable to fit into the final configuration.

For $k = 2$, C must be the pair at n and $n + 1$, i.e ○●. However after moving C to A, all the pairs in M are of the same color. The second move creates at most one pair, violating the condition 1. For $k \geq 3$, there are no coins in alternating order in the initial configuration. □

Proposition 2.2.3. *An n-move solution to a puzzle of size n uses one of the following k-block set during the rearrangement (see Figure 2.3):*

1. R_1
2. L_1
3. R_1 *and* L_1
4. R_1 *and* R_2
5. L_1 *and* L_2

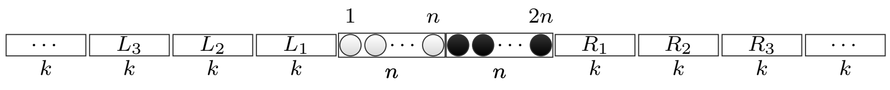


Figure 2.3: The k-blocks in the initial configuration.

Proof. From Proposition 2.2.2, we know that there is no gap between M and the k-blocks used during the rearrangement. Therefore if the solution uses at most two k-blocks, it can use only one of the five k-block set listed above.

Then we show that the solution cannot use more than two k-blocks. Without loss of generality, assume it uses three k-blocks, say R_1, R_2 and L_1. No matter what order the first time the three k-block are used in the solution, neither of the conditions in Proposition 2.2.1 is satisfied. □

2.3 Symmetry of solutions

The initial configuration of linear sliding-coin puzzle is symmetric. What happens if the moves in a solution are performed symmetrically?

For example, in Figure 2.4, the solution on the upper right is for $k = 2$ and $n = 3$. Imagine there is a mirror at the dashed line. The solution on the upper left is a mirror image of the solution on the upper right. If we flip the type of the coins in the upper left solution, it becomes a symmetric solution for $k = 2$ and $n = 3$ on the bottom left. Note that both final configurations on the upper right and the bottom left are of the same type. The symmetric solution does not rearrange the coins into the other type of finial configuration. This interesting property holds for any solution.

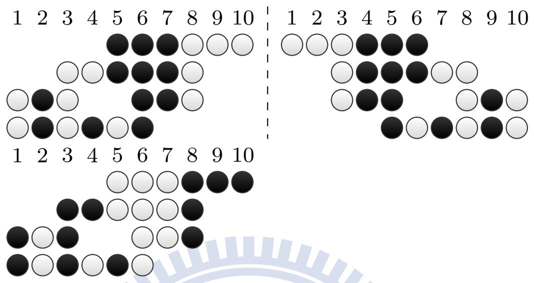


Figure 2.4: The solution on the upper left is a mirror image of the one for $k = 2$ and $n = 3$ on upper right. The symmetric solution on the bottom left is obtained from the one on the upper left by flipping the type of the coins.

Chapter 3

Sliding two adjacent coins in each move

In this section we explore linear sliding-coin puzzle in an algorithmic view. We start from the family with $k = 2$.

We first show the optimal solutions for some small n in Section 3.1. Then we develop a recursive algorithm which generates optimal solutions in Section 3.2. The idea of the algorithm is to reduce the size of the puzzle by 4 and then solve the subproblem recursively.

3.1 Special cases and base cases

The solution for $n = 3$ has already been presented in Section 1.1, which is a special case. The solutions for $n = 4, 5, 6$ and 7 are shown in the following theorem. Each solution has exactly n moves, which is optimal.

Theorem 3.1.1. *For $k = 2$, the following sequences are the optimal solutions for each corresponding n.*

- *For $n = 3$:* $(1 \to 7)$, $(6 \to 9)$, $(3 \to 6)$.
- *For $n = 4$:* $(2 \to 9)$, $(5 \to 2)$, $(8 \to 5)$, $(1 \to 8)$.
- *For $n = 5$:* $(2 \to 11)$, $(8 \to 2)$, $(5 \to 8)$, $(10 \to 5)$, $(1 \to 10)$.
- *For $n = 6$:* $(2 \to 13)$, $(8 \to 2)$, $(4 \to 8)$, $(9 \to 4)$, $(12 \to 9)$, $(1 \to 12)$.
- *For $n = 7$:* $(2 \to 15)$, $(11 \to 2)$, $(5 \to 11)$, $(10 \to 5)$, $(7 \to 10)$, $(14 \to 7)$, $(1 \to 14)$.

Proof. It is easy to check the correctness and the optimality of the solutions in Figures 1.2 and 3.1. □

The optimal solutions for $n = 4, 5, 6$ and 7 will be used as base cases in the algorithm. Note that their final configurations are black type and shifted to the right by two positions. This property is useful to develop the algorithm.

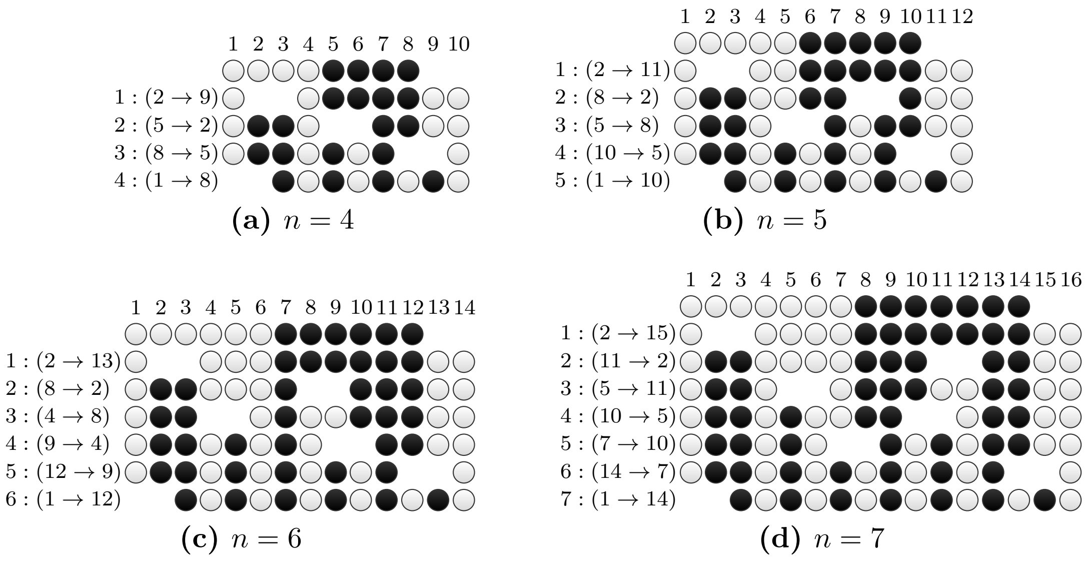


Figure 3.1: The optimal solutions for $k = 2$ and each corresponding n.

3.2 Algorithm

The following is the recursive algorithm which generates optimal solutions for $k = 2$.

```
1  function SlidingCoinK2(start, n)
2  begin
       // start is the starting position of the configuration.
       // n is the number of each type of the coins.
3      if $n < 3$ then
4          print "It is infeasible!"
5      else if $n = 3, 4, 5, 6$ or $7$ then
6          foreach $(p \to q)$ in the corresponding solution in Theorem 3.1.1 do
7              print "$(p + start - 1 \to q + start - 1),$"
8          end
9      else
10         print "$(start + 1 \to 2n + start),$"
11         print "$(2n + start - 4 \to start + 1),$"
12         SlidingCoinK2(start+4, n-4)
13         print "$(2n + start - 1 \to start + 4),$"
14         print "$(start \to 2n + start - 1),$"
15     end
16 end
```

Algorithm 3.1: SlidingCoinK2 *generates optimal solutions for* $k = 2$.

Theorem 3.2.1. SlidingCoinK2$(1, n)$ *generates the optimal solutions correctly for* $n \geq 3$. *Moreover, each solution shifts the black type final configuration to the right by two positions except for* $n = 3$.

Proof. We prove it by induction on n. By Theorem 3.1.1, it is clear that the algorithm generates the optimal solutions correctly for the special case $n = 3$ and the base cases $n = 4, 5, 6$, and 7.

Suppose the algorithm generates an optimal solution for size n. We prove that it generates an optimal solution as well for size $n + 4$. To further generalize the proof, assume that the initial configuration for $n + 4$ starts at position i as shown in row (1) of Figure 3.2. After performing the two moves generated by Step 10 and 11 of the algorithm, the configuration becomes (2) and (3), respectively.

Now the shaded sub-configuration in (3) is a subproblem of size n and starts at position $i+4$. By induction hypothesis, the recursive call in Step 12 of the algorithm generates an n-

move solution for size n. Since the n-move solution shifts the black type final configuration to the right by two positions, the shaded sub-configuration can be rearranged as shown in (4).

Finally, by performing the two moves generated by Step 13 and 14 of the algorithm, the black type final configuration is reached as shown in (6). It takes $n+4$ moves to solve the puzzle of size $n+4$, and it is optimal. $\square$

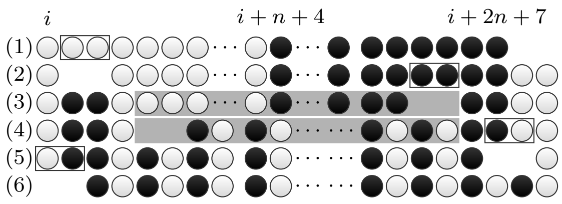


Figure 3.2: The inductive step in the proof of Theorem 3.2.1. The problem size is $n+4$ and the initial configuration starts at position i. Each row becomes the next row after moving the coins in the frame.

The algorithm generates an n-move solution to the puzzle of size n, and each move can be generated in constant time. Since the problem size is reduced by 4 in each recursion, the running time of `SlidingCoinK2` is $O(n)$, which is proportional to the number of coins in the puzzle.

Since the recursion depth of `SlidingCoinK2` is $O(n)$, the space complexity is $O(n \lg n)$. However it can be reduced to $O(\lg n)$ by removing the recursive step. Algorithm 3.2 is an iterative version of `SlidingCoinK2`. The variable i and n initiated at Step 6 and 7 are used to mimic *start* and *size_n* respectively as on each recursion of Step 12 of the recursive algorithm. The first while loop repeatedly does the job of Step 10 and 11 of the recursive algorithm, and reduces the problem size. Once the problem size matches the size of the base cases, the iterative algorithms print the solution of the base cases. Similarly, the second while loop does the job of Step 13 and 14 of the recursive algorithm.

```
function SlidingCoinK2Iter(start, size_n)
begin
    // start is the starting position of the configuration.
    // size_n is the number of each type of the coins.
    if $n < 3$ then
        print "It is infeasible!"
    else
        $i \leftarrow start$
        $n \leftarrow size\_n$
        while $n > 7$ do
            print "$(i + 1 \rightarrow 2n + i)$"
            print "$(2n + i - 4 \rightarrow i + 1)$"
            $i \leftarrow i + 4$
            $n \leftarrow n - 4$
        end
        if $n = 3, 4, 5, 6$ or $7$ then
            foreach $(p \rightarrow q)$ in the corresponding solution in Theorem 3.1.1 do
                print "$(p + start - 1 \rightarrow q + start - 1),$"
            end
        end
        while $n \leq size\_n$ do
            $i \leftarrow i - 4$
            print "$(2n + i - 1 \rightarrow i + 4)$"
            print "$(i \rightarrow 2n + i - 1)$"
            $n \leftarrow n + 4$
        end
    end
end
```

Algorithm 3.2: *An iterative version of* SlidingCoinK2.

Chapter 4

Sliding three adjacent coins in each move

In this section we explore the family with $k = 3$ algorithmically. Our approach is similar to the one in Chapter 3. We first show the optimal solutions for some small n in Section 4.1. However, we observe that the shapes of the solutions for odd n and for even n are quite different. Therefore we develop two algorithms separately, one for odd n and the other for even n, in Section 4.2.

4.1 Special cases and base cases

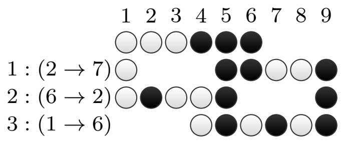
The solution for $n = 3$ will be used as a base case in the algorithm for odd n. As for even n, $n = 4, 6$ and 8 are special cases while $n = 10, 12$ and 14 are base cases in the algorithm for even n. Note that the initial configuration for even n starts at position 4.

Theorem 4.1.1. *For $k = 3$, the following sequences are the optimal solutions for each corresponding n.*

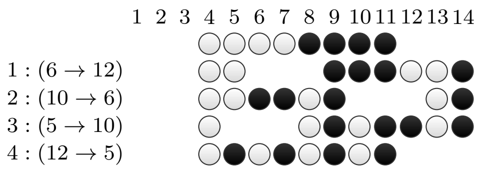
- *For $n = 3$:* $(2 \to 7)$, $(6 \to 2)$, $(1 \to 6)$.
- *For $n = 4$:* $(6 \to 12)$, $(10 \to 6)$, $(5 \to 10)$, $(12 \to 5)$.
- *For $n = 6$:* $(8 \to 16)$, $(5 \to 8)$, $(14 \to 5)$, $(10 \to 14)$, $(6 \to 10)$, $(9 \to 6)$, $(16 \to 9)$.
- *For $n = 8$:* $(10 \to 20)$, $(6 \to 10)$, $(14 \to 6)$, $(18 \to 14)$, $(12 \to 18)$, $(7 \to 12)$, $(10 \to 7)$, $(5 \to 10)$, $(20 \to 5)$.
- *For $n = 10$:* $(19 \to 1)$, $(10 \to 19)$, $(20 \to 24)$, $(15 \to 10)$, $(3 \to 20)$, $(22 \to 15)$, $(8 \to 3)$, $(1 \to 8)$, $(9 \to 22)$, $(4 \to 9)$.

- *For* $n = 12$*:* $(23 \to 1)$, $(12 \to 23)$, $(24 \to 28)$, $(19 \to 12)$, $(3 \to 24)$, $(26 \to 19)$, $(14 \to 26)$, $(8 \to 14)$, $(15 \to 3)$, $(1 \to 15)$, $(13 \to 8)$, $(4 \to 13)$.
- *For* $n = 14$*:* $(27 \to 1)$, $(10 \to 27)$, $(28 \to 32)$, $(21 \to 10)$, $(14 \to 21)$, $(3 \to 28)$, $(8 \to 3)$, $(19 \to 8)$, $(23 \to 19)$, $(1 \to 23)$, $(21 \to 14)$, $(30 \to 21)$, $(9 \to 30)$, $(4 \to 9)$.

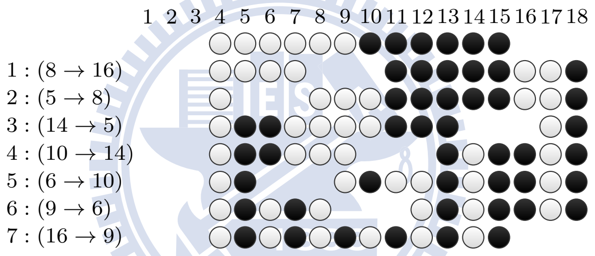
Proof. For $n = 3, 4, 10, 12$ and 14, it is easy to check the correctness and the optimality of the solutions in Figures 4.1. For $n = 6$ and 8, we have performed an exhaustive search for the 6-move and 8-move solutions, respectively, but failed. The solutions in Figures 4.1(c) and 4.1(d) are two of the best solutions that can be found. $\square$



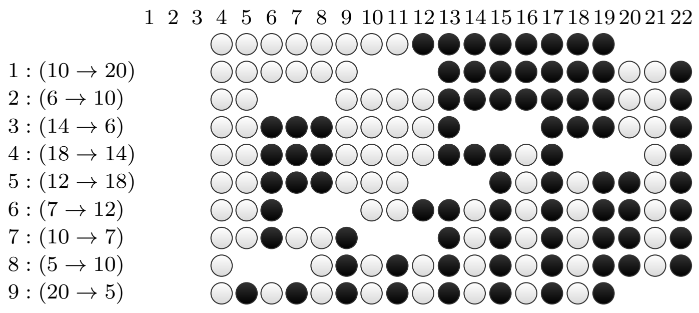
(a) $n = 3$



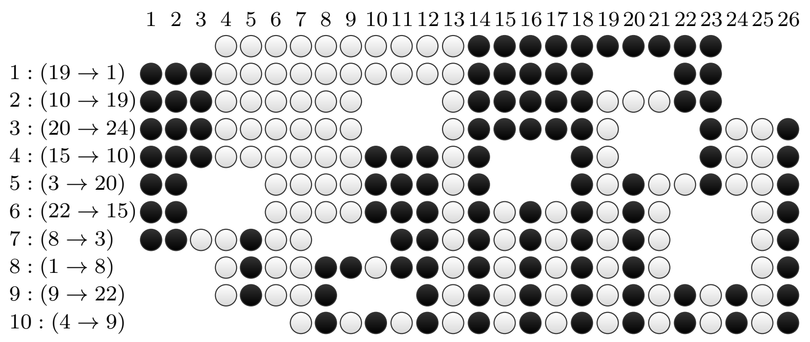
(b) $n = 4$



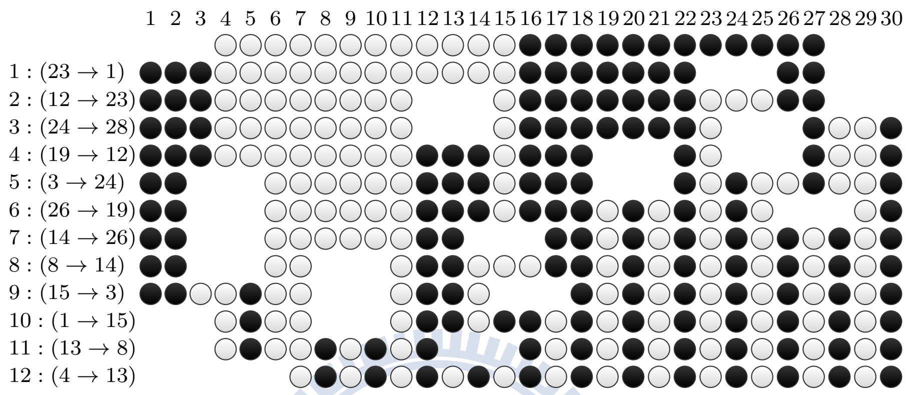
(c) $n = 6$



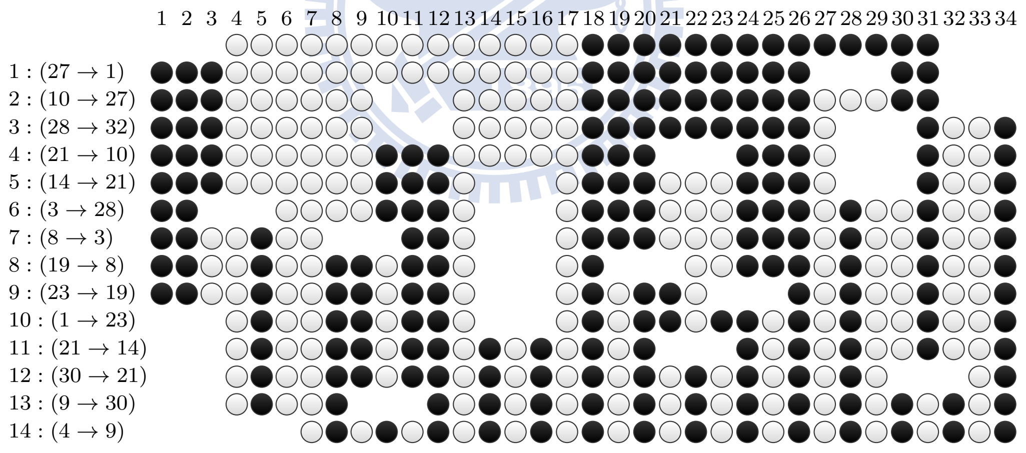
(d) $n = 8$



(e) $n = 10$



(f) $n = 12$



(g) $n = 14$

Figure 4.1: The optimal solutions for $k = 3$ and each corresponding n.

4.2 Algorithms

4.2.1 Algorithm for odd n

The following is the recursive algorithm which generates optimal solutions for $k = 3$ and odd n.

```
function SlidingCoinK3Odd($n$, $gap$)
begin
    // $n$ is the number of each type of the coins.
    // $gap$ controls the number of positions separating the $(n+1)$th
       coin and the $(n+2)$th coin in the configuration of size $n$.
    if $n < 3$ then
        print "It is infeasible!"
    else if $n = 3$ then
        print "$(2 \to n + gap + 4)$"
        print "$(n + gap + 3 \to 2)$"
        print "$(1 \to n + gap + 3)$"
    else
        print "$(n - 1 \to 2n + gap + 1)$"
        print "$(2n + gap - 1 \to n - 1)$"
        SlidingCoinK3Odd($n$-2, $gap$+2)
    end
end
```

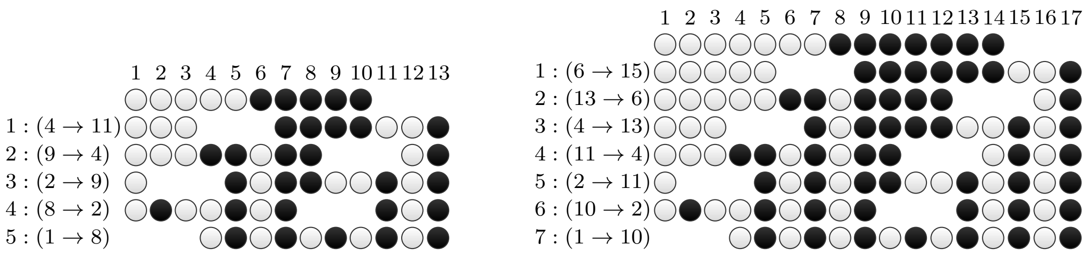
Algorithm 4.1: SlidingCoinK3Odd *generates optimal solutions for $k = 3$ and odd n.*

Theorem 4.2.1. *For odd $n \geq 3$,* SlidingCoinK3Odd$(n, 0)$ *generates the optimal solutions correctly. Moreover, each solution has the following three properties:*

1. *Each solution shifts the white type final configuration to the right by three positions.*
2. *The two coins at positions n and $n + 1$ are both stationary after the second move.*
3. *The $(n + 2)$th coin (the second penny) is stationary.*

Before proving the correctness of the algorithm, let's observe two more optimal solutions generated by SlidingCoinK3Odd$(n, 0$ for $n = 5$ and $n = 7$ as shown in Figure 4.2. The two solutions and the one for $n = 3$ in Figure 4.1(a) all satisfy the three properties in Theorem 4.2.1.

Suppose there is a gap between the $(n+1)$th coin and the $(n+2)$th coin in the initial configuration. For example, there is gap between the first penny (at positions 6) and the second penny (at position 7) in the initial configuration of Figure 4.2(a). Since the $(n+2)$th coins is stationary, we can still apply the solution to the new configuration by properly shifting all the coins after the $(n+2)$th coin to the right by the size of the gap. The gap in the configuration will not affect the moves in the solution. Thus the coins in the final configuration are still in alternating order (but in two parts) as if the gap does not exist.



(a) $n = 5$ **(b)** $n = 7$

Figure 4.2: Two optimal solutions generated by `SlidingCoinK3Odd`.

Proof of Theorem 4.2.1. We prove it by induction on odd n. By Theorem 4.1.1, it is clear that the algorithm correctly generates the optimal solution for the base case $n = 3$ (with $gap = 0$), and the solution satisfies the three properties.

Suppose the algorithm generates an optimal solution for size $n-2$ which satisfies the three properties. We prove that it generates an optimal solution correctly for size n which satisfies the three properties as well. Assume the initial configuration for n starts at positions 1 as shown in row (1) of Figure 4.3. After performing the two moves generated by Step 10 and 11 of the algorithm, the configuration becomes row (2) and row (3), respectively.

Now the shaded sub-configuration in row (2) is a subproblem of size $n-2$ in two parts, separated by a gap of size 2 (positions n and $n+1$). By the assumption and previous observation, the shaded sub-configuration can be solved in $n-2$ moves even if there is a gap between the first penny and the second penny of the shaded sub-configuration. Thus after the recursive call in Step 12, the shaded sub-configuration becomes white type final sub-configuration and shifts to the right by three positions as shown in row (4). The solution uses total n moves, which is optimal. The two coins ●○ at positions n and $n+1$ and the two ○● at positions $2n+2$ and $2n+3$ integrate with the shaded sub-configuration. Thus the $2n$ coins are rearranged correctly into the white type final configuration and shifted to the right by three positions. The second property is satisfied since two coins at positions n and $n+1$ are not moved by the solution to the subproblem of size $n-2$. The third property is also satisfied since the second penny is not moved by the first two moves and it is also the second penny of the subproblem of size $n-2$. □

The running time of `SlidingCoinK3Odd` is $O(n)$. The analysis of the time complexity is similar to the analysis for `SlidingCoinK2` on page 12.

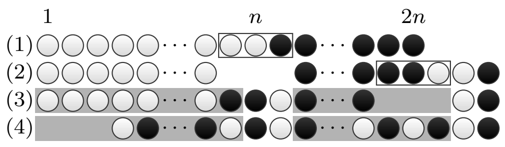


Figure 4.3: The inductive step in the proof of Theorem 4.2.1. The problem size is n and the initial configuration starts at position 1. Each row becomes the next row after moving the coins in the frame.

4.2.2 Algorithm for even n

Although the difference between odd n and even n seems little, the shapes of the solutions are quite different. The puzzle family for even n is not as easy as odd n, which can not be solved by simple recursions. We need to use a different trick to tackle the puzzle for even n.

Consider an auxiliary puzzle of size 6, where the solution is shown in Figure 4.4. Note that the final configuration is different from the original puzzle, which is black type at positions 1 to 6 and is white type at positions 7 to 12. This solution is specially designed such that we can combine it with a known solution S which satisfies the conditions shown in Lemma 4.2.2, resulting a solution to a puzzle of larger size.

Actually, the approach here is an application of a general framework. The ideas of the framework will be explained later in Chapter 5. Let's first accept the solution in Figure 4.4, and see how to use it to construct a solution to a puzzle of larger size. The detail of the construction is shown in the proof of Lemma 4.2.2, which implies an algorithm for even n shown in Algorithm 4.2.

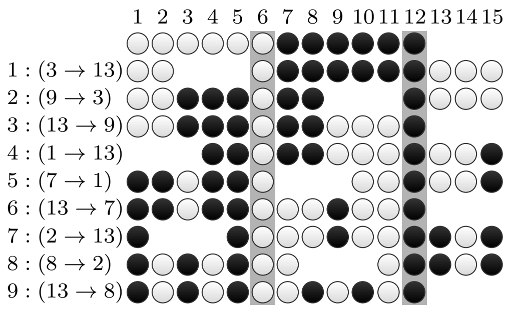


Figure 4.4: A solution to an auxiliary puzzle which can be used to derive the solutions for even n. The coins in the shade are stationary.

Lemma 4.2.2. *Given a solution S to a puzzle of size n which satisfies the following conditions:*

1. *There exists a nickel and a penny such that both coins are stationary.*
2. *Three patterns ○○○, ○○● and ●○● in this order are slid by three moves in S. (The three patterns are the ones showing up during the rearrangement in the 3-block at positions 13 to 15 in Figure 4.4.)*

Then S can be extended to be a solution S' to a puzzle of size $n + 6m$, $m \geq 1$.

Proof. Let S be a solution to a puzzle of size n which satisfies the conditions. Let C be the initial configuration for a puzzle of size n. Let a and b be the positions of the stationary nickel and penny in C, respectively. We give a construction which extends S to be a solution S' to a puzzle of size $n+6$, and we argue that S' still satisfies the two conditions.

First, we add six nickels and six pennies to the right of a and b in C, respectively. Let C' be the modified initial configuration. Let a' and b' be the positions in C' corresponding to the coins a and b in C, respectively. The added coins are shaded as shown in row (Init) in Figure 4.5.

Let S' be derived from S such that the moves consisting of the positions after a' and b' are properly shifted right accordingly as in C'. By condition 1, the coins at positions a' and b' are both stationary of C' when performing S'. For now, after performing the moves of S', the original coins can be slid to reach the final configuration without touching the added coins. Next, we need to further modify the moves of S' such that the added coins are in alternating order and integrated into the final configuration.

Let $(x_1 \to y_1)$ be the move in S sliding ○○○, which is guaranteed by condition 2. Let $(x'_1 \to y'_1)$ be the corresponding move in S'. We replace $(x'_1 \to y'_1)$ with the following three moves as shown in Figure 4.5.

$$\begin{aligned}
&\text{(A-1)}: \ (a'+3 \to y'_1)\\
&\text{(A-2)}: \ (b'+3 \to a'+3)\\
&\text{(A-3)}: \ (x'_1 \to b'+3).
\end{aligned}$$

The net effect of the three moves on the original coins is the same as $(x'_1 \to y'_1)$, i.e. ○○○ is slid to y'_1, and x'_1 becomes unoccupied. The sub-configuration of the added coins is shown in Figure 4.5.

Similarly, let $(x'_2 \to y'_2)$ be the move in S' sliding ○○●, and we replace $(x'_2 \to y'_2)$ with

$$\begin{aligned}
&\text{(B-1):}\ (a'+1 \to y'_2)\\
&\text{(B-2):}\ (b'+1 \to a'+1)\\
&\text{(B-3):}\ (x'_2 \to b'+1).
\end{aligned}$$

Also, let $(x'_3 \to y'_3)$ be the move in S' sliding ●○●, and we replace $(x'_3 \to y'_3)$ with

$$\begin{aligned}
&\text{(C-1):}\ (a'+2 \to y'_3)\\
&\text{(C-2):}\ (b'+2 \to a'+2)\\
&\text{(C-3):}\ (x'_3 \to b'+2).
\end{aligned}$$

After the (C-3) move, the added nickels $a'+1, \dots, a'+6$ and pennies $b'+1, \dots, b'+6$ are in alternating order as shown in Figure 4.5. Since S is a correct solution, after the last move of S', the coins at $a'+7$ and $b'+7$ are a penny and a nickel, respectively. The added coins fit into the final configuration of S'. Hence, S' is a correct solution as well, and has exactly six more moves than S.

Observe Figure 4.5, the nickel at position $a'+6$ and the penny at position $b'+6$ are both stationary after performing S'. Three patterns ○○○, ○○● and ●○● in this order are slid by the three move (A-1), (B-1) and (C-1) in S'. Therefore, S' still satisfies the two conditions. The construction can be applied repeatedly. Hence S can be extended to be a solution to a puzzle of size $n + 6m$, $m \geq 1$. □

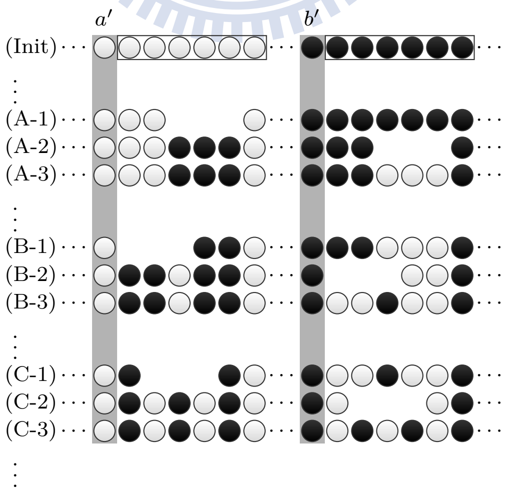


Figure 4.5: The modifications of the three moves in condition 2 in the proof of Lemma 4.2.2. The coins at a' and b' are stationary (in the shade). The added coins are in the frame.

See the solution for $n = 16$ in Figure 4.6 for a example. It is generated by the construction shown in the proof of Lemma 4.2.2, which uses the solution for $n = 10$ in Figure 4.1e and the solution to the auxiliary puzzle in Figure 4.4. In Figure 4.1e, the coins at 7 and 18 are stationary. The three moves 2, 3 and 9 move the three patterns ○○○, ○○● and ●○●, respectively. The moves in A, B and C in Figure 4.6 are extended from the move 2,3 and 9 in Figure 4.1e.

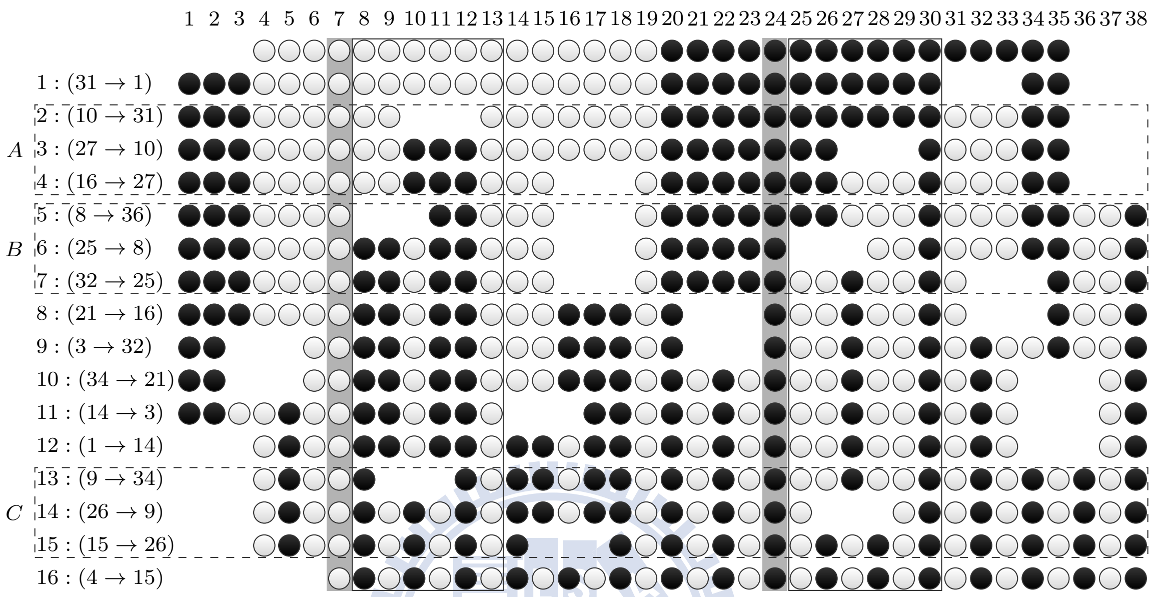


Figure 4.6: An optimal solution for $k = 3$ and $n = 16$, which is extended from the solutions for $n = 10$ in Figure 4.1e by using the construction in Lemma 4.2.2. The added coins are in the solid frame. The moves in A, B and C (in the dashed frame) are extended from the move 2,3 and 9 in Figure 4.1e, respectively.

Theorem 4.2.3. *For even $n \geq 10$, there exists an optimal solution which solves the puzzle in exactly n moves.*

Proof. For $n = 10, 12$ and 14, it is clear that each has an optimal solution by Theorem 4.1.1. It is easy to check that each solution satisfies the two conditions in Lemma 4.2.2 from Figure 4.1e, 4.1f and 4.1g. The construction in the proof of Lemma 4.2.2 implies that if S has n moves, then S' has $n + 6m$ moves, $m \geq 1$. Hence the solutions for $n = 10, 12$ and 14 can be used as base cases, and can be extended to be the solutions for all even $n \geq 10$. $\square$

Algorithm 4.2 is a direct implementation of Theorem 4.2.3. Note that the initial configuration starts at position 4. Since the solutions for base cases $n = 10, 12, 14$ are known, the extended moves to the solutions can be calculated in constant time. Thus, it takes

$O(n)$ time for the algorithm to generates a solution for even n.

```
function SlidingCoinK3Even($n$)
begin
    // $n$ is the number of each type of the coins.
    if $n < 3$ then
        print "It is infeasible!"
    else if $n = 4, 6, 8, 10, 12$ or $14$ then
        foreach $(p \to q)$ in the corresponding solution in Theorem 4.1.1 do
            print "$(p \to q)$,"
        end
    else if $n = 10 + 6m, m \geq 1$ then
        Apply the construction in Lemma 4.2.2 by using the solution for $n = 10$ in
        Figure 4.1e.
    else if $n = 12 + 6m, m \geq 1$ then
        Apply the construction in Lemma 4.2.2 by using the solution for $n = 12$ in
        Figure 4.1f.
    else if $n = 14 + 6m, m \geq 1$ then
        Apply the construction in Lemma 4.2.2 by using the solution for $n = 14$ in
        Figure 4.1g.
    end
end
```

Algorithm 4.2: SlidingCoinK3Even *generates optimal solutions for* $k = 3$ *and even* n.

Chapter 5

A framework for general cases

In this section we propose a framework to find the needed components which can be used to develop algorithms generating optimal solutions. The framework can be regarded as a meta-algorithm or a heuristic method to construct optimal solutions for k and sufficiently large n.

Actually, the algorithm for $k = 3$ and even n is a successful application of the framework. We also apply the framework successfully to construct the optimal solutions for the family with $k = 4$ and 5.

We introduce an auxiliary puzzle of size $2k$ as shown in Figure 5.1. The initial configuration has $2k$ nickels to the left of $2k$ pennies. In each move, k adjacent coins can be slid. Note that the goal is different from the original puzzle. The first half of the final configuration is black type while the second half is white type.

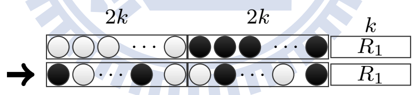


Figure 5.1: The initial and final configuration of an auxiliary puzzle of size $2k$. k adjacent coins can be slid in each move.

Suppose we deal with the puzzle family with some k. Let S_n be an optimal solution to the puzzle of size n. The following is the framework. We explain the details of each phase in the following section.

```
1 repeat
2     Phase 1. Perform exhaustive search for a solution $A$ to the auxiliary puzzle of
      size $2k$ which satisfies Requirement 5.1.1.
3     Phase 2. Perform exhaustive search for $2k$ solutions $S_n, \ldots, S_{n+2k-1}$ to the
      original puzzle of size in the range of $n$ to $n + 2k - 1$, for some $n$. Each solution
      must satisfy Requirement 5.1.2.
4 until $A$ and $S_n, \ldots, S_{n+2k-1}$ are found.
5 Phase 3. $S_n$ can be extended to be an optimal solution $S_{n+2km}$, where $m \geq 1$.
  Thus $S_n, \ldots, S_{n+2k-1}$ solve all the puzzles of size at least $n$.
```

Algorithm 5.1: *The framework*

The idea behind the framework is as follows. The solution A to the auxiliary puzzle can be seen as a building block of size $2k$, which can be piled onto S_n (with some modifications). The result is an optimal solution S_{n+2k}. The requirements in Phase 1 and 2 are used to guarantee that A and S_n can be integrated successfully.

5.1 The framework

5.1.1 Phase 1

In Phase 1, we perform exhaustive search for a solution A to the auxiliary puzzle of size $2k$. A must satisfy the following requirements.

Requirement 5.1.1.

1. *Both the initial configuration and final configuration start at the same positions.*

2. *Only the k-block R_1 is used during the rearrangement.*

3. *Both the kth and $2k$-th coins are stationary.*

4. *The $(3i+1)$th move must slide a pattern from the first $2k$-block to R_1, and the $(3i+3)$th move must slide a pattern from R_1 to the second $2k$-block, where $0 \leq i \leq k - 1$.*

5. *It has $3k$ moves.*

By Requirement 1 and 2, A can be regarded as a quasi-solution to the original problem of size $2k$, which can be split into three parts, including the first $2k$-block, the second $2k$-block, and a temporary space R_1. R_1 can be viewed as a buffer such that A uses it in order to rearrange the coins in the first $2k$-block and second $2k$-block into alternating order.

By Requirement 3, the number of coins which can be slid in the first $2k$-block is $2k-1$, so is it in the second $2k$-block. Thus, by Requirement 4, the $(3i+2)$th move must slide a pattern from the second $2k$-block to the first $2k$-block, and the patterns slid in the $(3i+1)$th move and in the $(3i+3)$th move are of the same. Let p and p' be the patterns slid by $(3i+1)$th move and $(3i+2)$th move, respectively. Note that p' is the complement to p, i.e. p' is the pattern by flipping the coin types in p.

By Requirement 5, there are k patterns which are moved to R_1 in A. Let $P = \{p_1, \ldots, p_k\}$ be the sequence of the k patterns. Each pattern p_i is different, and number of pairs in each pattern p_i is increasing. P can be viewed as the necessary intermediate output to the buffer R_1 for A.

For example, Figure 4.4 is a solution to the auxiliary puzzle for $k = 3$. It satisfies all the requirements. The three patterns in P are ○○○, ○○● and ●○●.

5.1.2 Phase 2

After Phase 1, we have P the sequence of k patterns, which will be used as one of the requirements in Phase 2. In this phase, we also perform exhaustive search for the $2k$ optimal solutions $S_n, \ldots, S_{n+2k-1}$ to the original puzzles of size in the range of n to $n+2k-1$, for some n. In addition, each solution S_i must satisfy the following requirements.

Requirement 5.1.2.

1. *There exists a nickel and a penny such that both coins are stationary in S_i.*

2. *The patterns $p_1, \ldots, p_k$ are moved by some k moves $m_1, \ldots, m_k$ in S_i, respectively.*

The $2k$ optimal solutions are the bases, and each of which can be integrated with A by the "connector" P in Phase 3.

For example, Lemma 4.2.2 is an instance of Requirement 5.1.2 for $k = 3$ with respect to the solution to the auxiliary puzzle of size 6 in Figure 4.4.

Note that the search in Phase 2 depends on the solution A derived from Phase 1. Therefore if the $2k$ optimal solutions can not be found, we go back to Phase 1 and conduct a new search for another A'.

5.1.3 Phase 3

In this phase, we already have all the needed components: A and the $2k$ optimal solutions $S_n, \ldots, S_{n+2k-1}$. The following theorem generalizes Lemma 4.2.2 to any k, which can be used to implement the algorithms generating optimal solutions.

Theorem 5.1.1. *Given a solution A to the auxiliary puzzle of size $2k$ satisfying Requirement 5.1.1 and an optimal solution S_n satisfying Requirement 5.1.2, then S_n can be extended to be an optimal solution S_{n+2km}, $m \geq 1$.*

Proof. The theorem generalizes Lemma 4.2.2 which is an instance for $k = 3$. For a concrete result of the construction, please see Figure 4.6. We briefly review the idea as follows.

Let a and b be the positions of the stationary nickel and penny in S, respectively. We add $2k$ nickels and $2k$ pennies to the right of a and b, respectively. The positions after a and b in S should be properly shifted to the right by $2k$ and $4k$, respectively.

Next, we modify the k moves $m_1, \ldots, m_k$ which moves $p_1, \ldots, p_k$ in S. The moves in A can be divided into k groups, say $M_1, \ldots, M_k$ such that each group M_i has three consecutive moves $(3i+1), (3i+2)$ and $(3i+3)$, $0 \leq i \leq k$. We replace each m_i in S with M_i with some modifications. Without loss of generality, we show how to modify m_1. Suppose m_1 slides p_1 from position x to position y, denoted as $x \overset{p_1}{\to} y$. We regard m_1 as that x becomes unoccupied and y gets p_1. Let the three moves in M_i be

$$g \overset{p_1}{\to} R_1$$
$$h \overset{p_1'}{\to} g$$
$$R_1 \overset{p_1}{\to} h,$$

where p_1' is the complement pattern of p_1. We regard the three moves as that g has p_1 and h has p_1', and they exchange p_1 and p_1' by using the buffer R_1. Then we replace m_1 by the following three moves such that the needs of x, y, g and h are all fulfilled:

$$g \overset{p_1}{\to} y$$
$$h \overset{p_1'}{\to} g$$
$$x \overset{p_1}{\to} h.$$

$m_2 \ldots, m_k$ are modified similarly. Then S_n can be extended to be an optimal solution S_{n+2k} which also satisfies Requirement 5.1.2. Therefore this technique can be applied repeatedly, resulting an optimal solution S_{n+2km}, $m \geq 1$. □

By Theorem 5.1.1, the $2k$ optimal solutions $S_n, \ldots, S_{n+2k-1}$ can be extended to be optimal solutions to the puzzles of size at least $n+2k$. Therefore they solve all the puzzles of size at least n. For the puzzles of size less than n, we can perform exhaustive search for the optimal solutions.

5.2 Application for $k = 4$ and $k = 5$

To show that the framework actually applies for $k = 4$ and $k = 5$, we present the valid data found in Phase 1 and Phase 2.

5.2.1 $k = 4$

Phase 1: a solution A. P contains ○○○○, ●○○○, ●●○●, ○●○●.

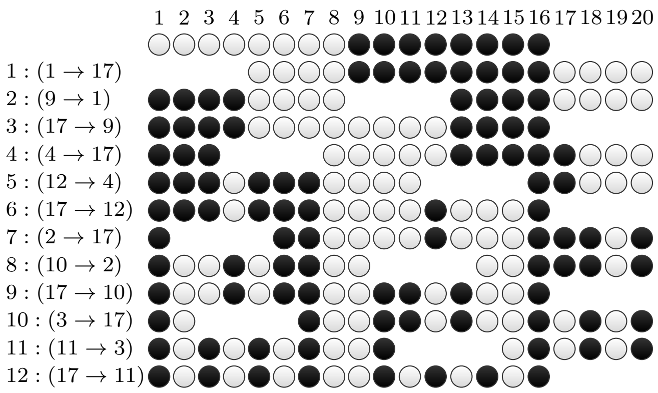


Figure 5.2: A solution to an auxiliary puzzle of size 8.

Phase 2: the $2k$ optimal solutions to the puzzles of size in the range of 13 to 20. Note that each initial configuration starts at position 5. For the figures of the following solutions, please see Appendix A.1 on page 34. By Theorem 5.1.1, we have the optimal solutions for $n \geq 13$.

- For $n = 13$: $(6 \to 31)$, $(17 \to 1)$, $(23 \to 6)$, $(30 \to 17)$, $(16 \to 23)$, $(3 \to 16)$, $(11 \to 3)$, $(4 \to 30)$, $(28 \to 4)$, $(5 \to 11)$, $(13 \to 28)$, $(22 \to 13)$, $(1 \to 22)$.
- For $n = 14$: $(6 \to 33)$, $(23 \to 6)$, $(12 \to 23)$, $(25 \to 1)$, $(4 \to 12)$, $(20 \to 25)$, $(11 \to 20)$, $(32 \to 11)$, $(13 \to 4)$, $(26 \to 13)$, $(1 \to 26)$, $(27 \to 32)$, $(14 \to 27)$, $(5 \to 14)$.
- For $n = 15$: $(6 \to 35)$, $(19 \to 1)$, $(24 \to 6)$, $(34 \to 19)$, $(18 \to 24)$, $(29 \to 18)$, $(12 \to 29)$, $(4 \to 12)$, $(27 \to 4)$, $(11 \to 34)$, $(3 \to 11)$, $(17 \to 3)$, $(5 \to 27)$, $(28 \to 17)$, $(1 \to 28)$.
- For $n = 16$: $(6 \to 37)$, $(22 \to 6)$, $(11 \to 22)$, $(25 \to 1)$, $(31 \to 11)$, $(12 \to 25)$, $(19 \to 31)$, $(4 \to 19)$, $(18 \to 12)$, $(36 \to 18)$, $(20 \to 4)$, $(26 \to 20)$, $(1 \to 26)$, $(17 \to 36)$, $(28 \to 17)$, $(5 \to 28)$.

- For $n = 17$: $(6 \to 39)$, $(21 \to 1)$, $(26 \to 6)$, $(38 \to 21)$, $(11 \to 26)$, $(3 \to 11)$, $(19 \to 3)$, $(32 \to 19)$, $(4 \to 38)$, $(20 \to 4)$, $(27 \to 20)$, $(12 \to 27)$, $(17 \to 32)$, $(33 \to 12)$, $(5 \to 17)$, $(18 \to 33)$, $(1 \to 18)$.

- For $n = 18$: $(6 \to 41)$, $(20 \to 1)$, $(25 \to 6)$, $(40 \to 20)$, $(13 \to 25)$, $(35 \to 13)$, $(22 \to 35)$, $(3 \to 40)$, $(32 \to 3)$, $(4 \to 22)$, $(19 \to 32)$, $(14 \to 19)$, $(27 \to 14)$, $(11 \to 4)$, $(17 \to 11)$, $(1 \to 17)$, $(14 \to 27)$, $(5 \to 14)$.

- For $n = 19$: $(6 \to 43)$, $(23 \to 1)$, $(28 \to 6)$, $(42 \to 23)$, $(11 \to 28)$, $(37 \to 11)$, $(18 \to 37)$, $(3 \to 18)$, $(25 \to 3)$, $(4 \to 42)$, $(12 \to 4)$, $(31 \to 12)$, $(15 \to 25)$, $(36 \to 31)$, $(30 \to 15)$, $(5 \to 30)$, $(27 \to 36)$, $(14 \to 27)$, $(1 \to 14)$.

- For $n = 20$: $(6 \to 45)$, $(22 \to 1)$, $(27 \to 6)$, $(44 \to 22)$, $(11 \to 27)$, $(39 \to 11)$, $(24 \to 39)$, $(3 \to 44)$, $(12 \to 3)$, $(29 \to 12)$, $(4 \to 24)$, $(13 \to 4)$, $(36 \to 29)$, $(31 \to 13)$, $(2 \to 36)$, $(16 \to 2)$, $(21 \to 16)$, $(1 \to 31)$, $(30 \to 21)$, $(5 \to 30)$.

5.2.2 $k = 5$

Phase 1: a solution A. P contains ○○○○, ○●●●●, ○○●○○, ○●○○● and ●○●○●.

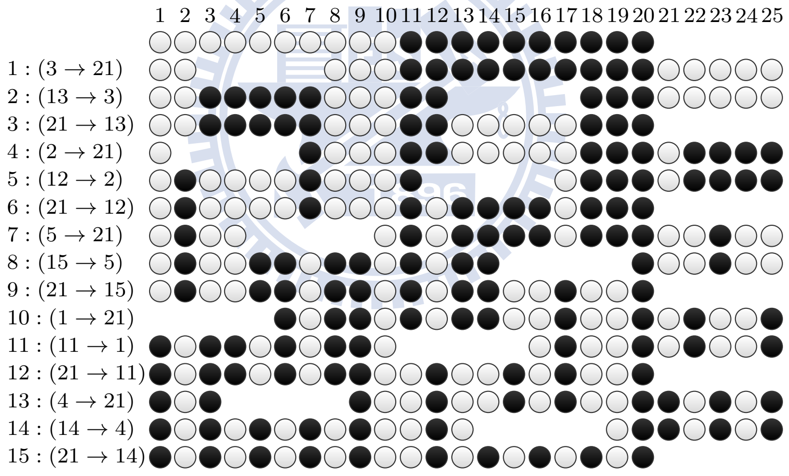


Figure 5.3: A solution to an auxiliary puzzle of size 10.

Phase 2: the $2k$ optimal solutions to the puzzles of size in the range of 14 to 23. Note that each initial configuration starts at position 6. For the figures of the following solutions, please see Appendix A.2 on page 40. By Theorem 5.1.1, we have the optimal solutions for $n \geq 14$.

- For $n = 14$: $(21 \to 1)$, $(9 \to 21)$, $(25 \to 34)$, $(3 \to 25)$, $(26 \to 9)$, $(18 \to 26)$, $(27 \to 3)$, $(33 \to 18)$, $(10 \to 27)$, $(16 \to 10)$, $(1 \to 16)$, $(24 \to 33)$, $(17 \to 24)$, $(6 \to 17)$.

- For $n = 15$: $(22 \to 1)$, $(12 \to 22)$, $(26 \to 36)$, $(4 \to 26)$, $(27 \to 12)$, $(18 \to 27)$, $(33 \to 18)$, $(24 \to 33)$, $(34 \to 4)$, $(2 \to 34)$, $(15 \to 2)$, $(6 \to 15)$, $(35 \to 24)$, $(14 \to 35)$, $(1 \to 14)$.

- For $n = 16$: $(24 \to 1)$, $(16 \to 24)$, $(28 \to 38)$, $(35 \to 16)$, $(5 \to 28)$, $(12 \to 5)$, $(3 \to 12)$, $(26 \to 35)$, $(13 \to 26)$, $(29 \to 3)$, $(23 \to 29)$, $(36 \to 23)$, $(26 \to 13)$, $(1 \to 36)$, $(37 \to 26)$, $(6 \to 37)$.

- For $n = 17$: $(31 \to 1)$, $(14 \to 31)$, $(22 \to 40)$, $(4 \to 22)$, $(23 \to 14)$, $(16 \to 23)$, $(24 \to 4)$, $(37 \to 16)$, $(30 \to 24)$, $(12 \to 30)$, $(2 \to 12)$, $(26 \to 37)$, $(39 \to 2)$, $(13 \to 26)$, $(6 \to 39)$, $(38 \to 13)$, $(1 \to 38)$.

- For $n = 18$: $(25 \to 1)$, $(12 \to 25)$, $(29 \to 42)$, $(5 \to 29)$, $(33 \to 5)$, $(4 \to 12)$, $(22 \to 4)$, $(39 \to 22)$, $(15 \to 33)$, $(32 \to 39)$, $(41 \to 32)$, $(31 \to 15)$, $(3 \to 31)$, $(18 \to 3)$, $(1 \to 18)$, $(28 \to 41)$, $(19 \to 28)$, $(6 \to 19)$.

- For $n = 19$: $(26 \to 1)$, $(12 \to 26)$, $(27 \to 44)$, $(42 \to 27)$, $(28 \to 12)$, $(14 \to 28)$, $(32 \to 42)$, $(4 \to 32)$, $(33 \to 14)$, $(16 \to 33)$, $(34 \to 4)$, $(2 \to 34)$, $(35 \to 16)$, $(24 \to 35)$, $(41 \to 24)$, $(23 \to 2)$, $(6 \to 23)$, $(22 \to 41)$, $(1 \to 22)$.

- For $n = 20$: $(27 \to 1)$, $(12 \to 27)$, $(31 \to 46)$, $(43 \to 12)$, $(4 \to 31)$, $(23 \to 43)$, $(37 \to 23)$, $(19 \to 4)$, $(13 \to 19)$, $(45 \to 13)$, $(5 \to 37)$, $(27 \to 5)$, $(3 \to 45)$, $(44 \to 27)$, $(20 \to 44)$, $(34 \to 20)$, $(21 \to 3)$, $(1 \to 21)$, $(17 \to 34)$, $(6 \to 17)$.

- For $n = 21$: $(28 \to 1)$, $(12 \to 28)$, $(34 \to 12)$, $(20 \to 34)$, $(38 \to 48)$, $(45 \to 20)$, $(4 \to 38)$, $(31 \to 45)$, $(16 \to 31)$, $(29 \to 16)$, $(17 \to 4)$, $(2 \to 17)$, $(46 \to 2)$, $(39 \to 29)$, $(5 \to 39)$, $(13 \to 5)$, $(26 \to 13)$, $(6 \to 46)$, $(47 \to 26)$, $(32 \to 47)$, $(1 \to 32)$.

- For $n = 22$: $(29 \to 1)$, $(12 \to 29)$, $(35 \to 12)$, $(19 \to 35)$, $(37 \to 50)$, $(5 \to 37)$, $(41 \to 5)$, $(3 \to 19)$, $(26 \to 3)$, $(49 \to 26)$, $(20 \to 41)$, $(39 \to 20)$, $(13 \to 39)$, $(4 \to 13)$, $(28 \to 49)$, $(48 \to 4)$, $(36 \to 48)$, $(47 \to 28)$, $(1 \to 36)$, $(16 \to 47)$, $(37 \to 16)$, $(6 \to 37)$.

- For $n = 23$: $(30 \to 1)$, $(12 \to 30)$, $(36 \to 12)$, $(18 \to 36)$, $(40 \to 52)$, $(51 \to 18)$, $(4 \to 40)$, $(34 \to 51)$, $(16 \to 34)$, $(26 \to 16)$, $(41 \to 26)$, $(28 \to 41)$, $(42 \to 4)$, $(49 \to 42)$, $(17 \to 49)$, $(2 \to 17)$, $(13 \to 28)$, $(25 \to 2)$, $(50 \to 13)$, $(6 \to 25)$, $(43 \to 50)$, $(24 \to 43)$, $(1 \to 24)$.

Chapter 6

Conclusion and open problems

We proved that n moves is the minimum number of moves required to solve a puzzle, showed the properties of the n-move solutions, and designed the algorithms generating optimal solutions for the puzzle families with $k = 2$ and 3. Furthermore, we proposed a framework to extend solutions, and applied it successfully to construct the optimal solutions for $k = 4$ and 5. However, there are many issues which remain unclear. We state them as open problems.

1. Does the framework proposed in Chapter 5 apply to any $k \geq 6$?

2. Can the puzzle be solved in n moves for any k and sufficiently large n?

3. What does the complexity class solving the puzzle belong given k and n as parameters?

4. What is the minimum moving distance to solve a puzzle given k and n?

5. Can an arbitrary initial configuration of equal number of nickels and pennies be rearranged to alternate coins? Can it be done in n moves?

Bibliography

[1] Geoffrey Mott-Smith, *Mathematical Puzzle, for Beginners and Enthusiasts*, page 7, Dover Publications, 1954.

[2] Martin Gardner, *Mathematical Puzzles of Sam Loyd*, page 70, Dover Publications, 1959.

[3] Edward Hordern, *Sliding Piece Puzzles*, Oxford University Press, 1987.

[4] Erik D. Demaine, Martin L. Demaine, and Helena A. Verrill, "Coin-Moving Puzzles," *More Games of No Chance*, pages 405-— 431, Cambridge University Press, 2002.

[5] C.-C. Chu, M.-Y. Lee, and C.-S. Tsai, "Sliding-Coin Puzzle," Unpublished manuscript (in Chinese), 2003. `http://tinyurl.com/34o3vpe`

[6] Erik D. Demaine, and Martin L. Demaine, "Puzzles, Art, and Magic with Algorithms," Theory of Computing Systems, volume 39, number 3, pages 473–481, 2006.

[7] M. Abellanas, S. Bereg, F. Hurtado, A. G. Olaverri, D. Rappaport, and J. Tejel, "Moving Coins", *Computational Geometry*, volume 34, issue 1, pages 35–48, 2006.

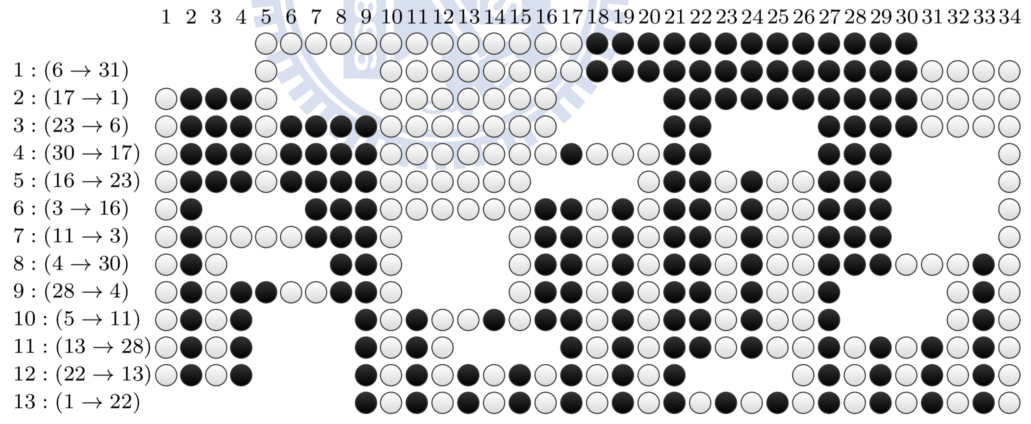
[8] "Sliding Coin Puzzles and Games." `http://www.alethis.net/amusements/coin/coin.html`

Appendices

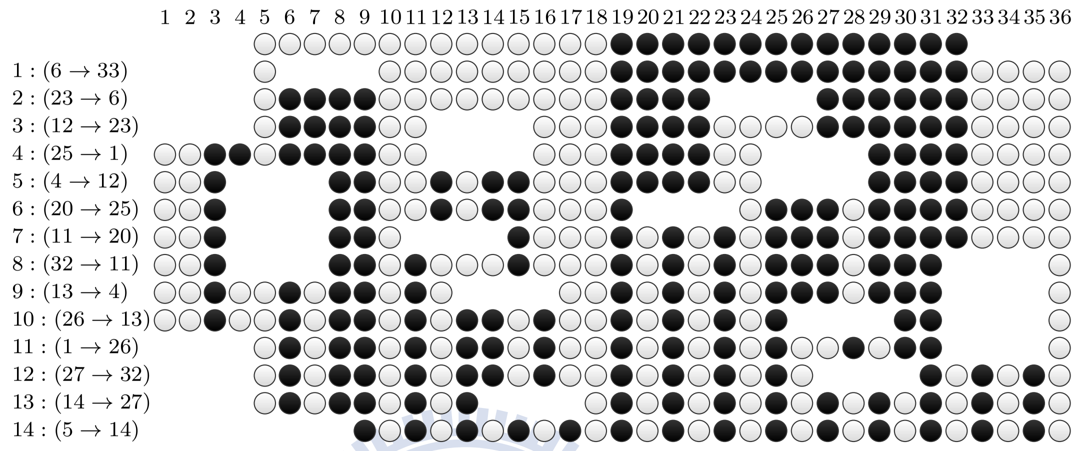
Appendix A

Figures of the application for $k = 4$ and $k = 5$

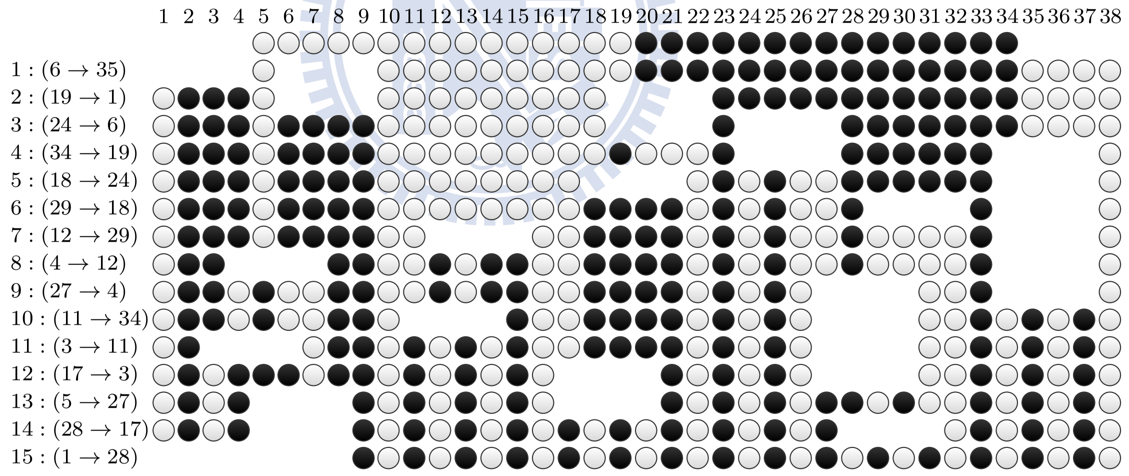
A.1 $k = 4$



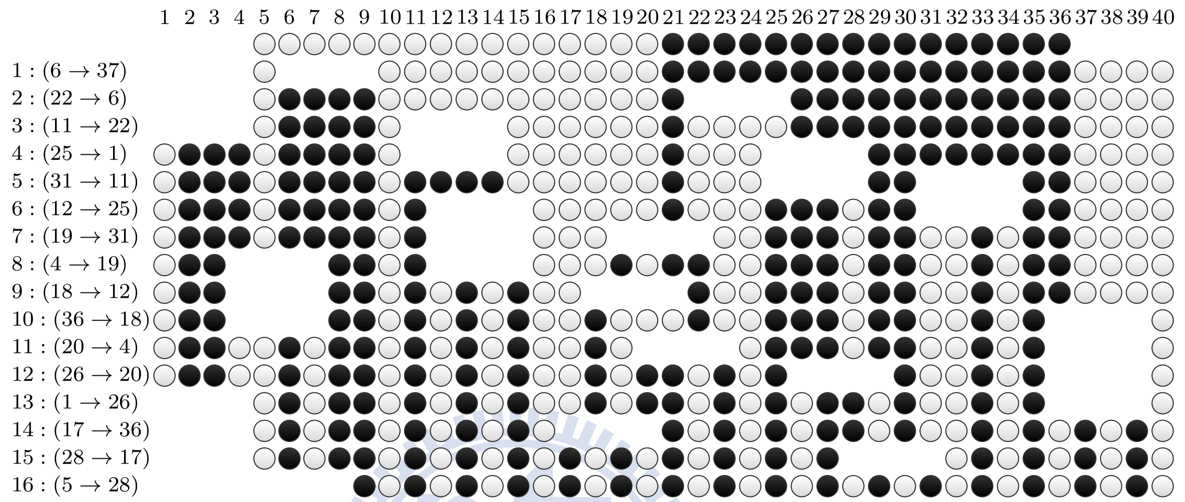
(a) $n = 13$


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
1 : (6 → 33)
2 : (23 → 6)
3 : (12 → 23)
4 : (25 → 1)
5 : (4 → 12)
6 : (20 → 25)
7 : (11 → 20)
8 : (32 → 11)
9 : (13 → 4)
10 : (26 → 13)
11 : (1 → 26)
12 : (27 → 32)
13 : (14 → 27)
14 : (5 → 14)

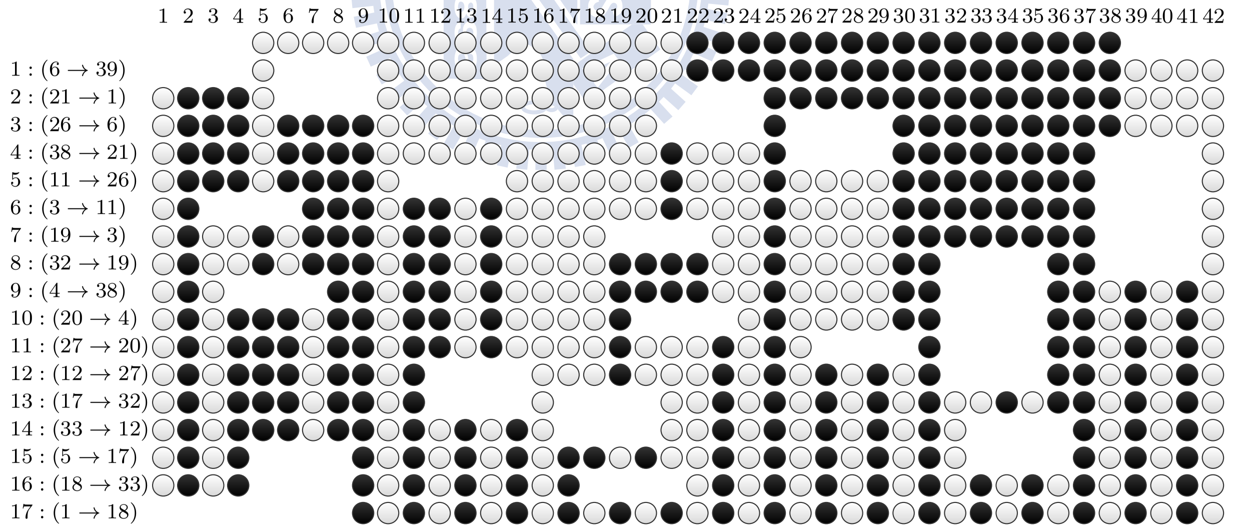
(b) $n = 14$


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38
1 : (6 → 35)
2 : (19 → 1)
3 : (24 → 6)
4 : (34 → 19)
5 : (18 → 24)
6 : (29 → 18)
7 : (12 → 29)
8 : (4 → 12)
9 : (27 → 4)
10 : (11 → 34)
11 : (3 → 11)
12 : (17 → 3)
13 : (5 → 27)
14 : (28 → 17)
15 : (1 → 28)

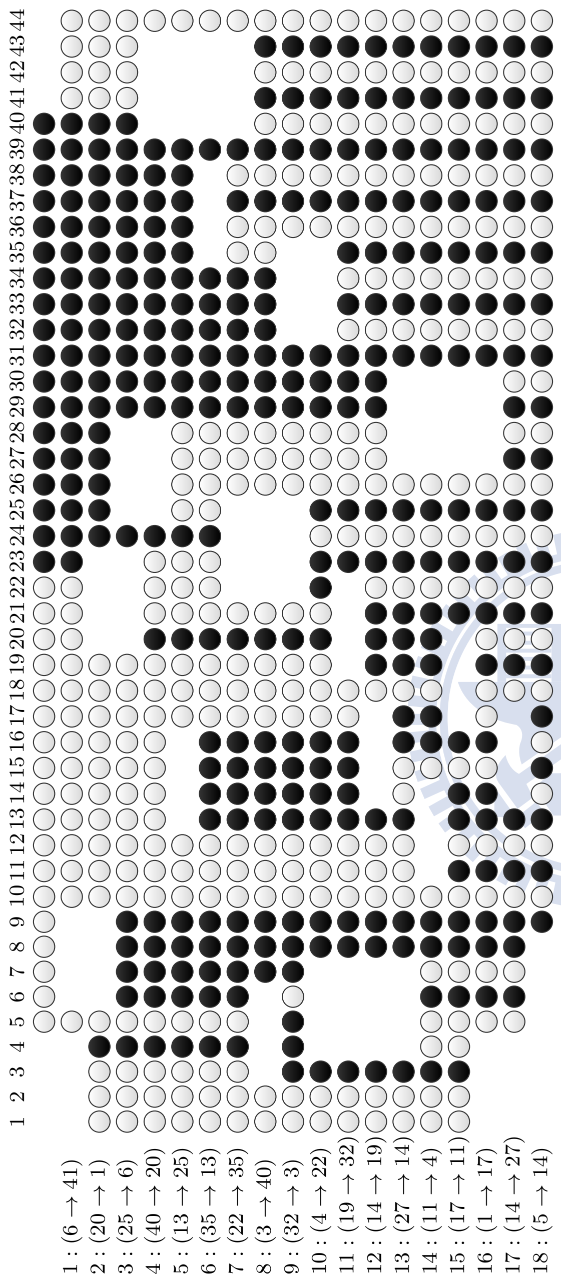
(c) $n = 15$


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
1 : (6 → 37)
2 : (22 → 6)
3 : (11 → 22)
4 : (25 → 1)
5 : (31 → 11)
6 : (12 → 25)
7 : (19 → 31)
8 : (4 → 19)
9 : (18 → 12)
10 : (36 → 18)
11 : (20 → 4)
12 : (26 → 20)
13 : (1 → 26)
14 : (17 → 36)
15 : (28 → 17)
16 : (5 → 28)

(d) $n = 16$

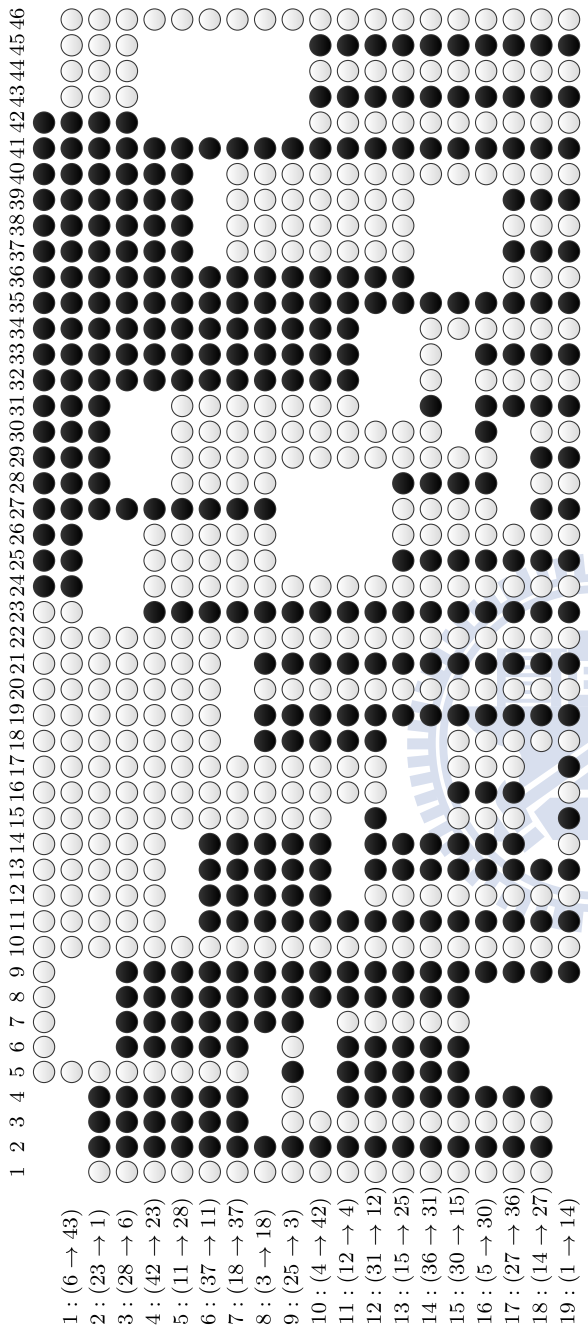

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42
1 : (6 → 39)
2 : (21 → 1)
3 : (26 → 6)
4 : (38 → 21)
5 : (11 → 26)
6 : (3 → 11)
7 : (19 → 3)
8 : (32 → 19)
9 : (4 → 38)
10 : (20 → 4)
11 : (27 → 20)
12 : (12 → 27)
13 : (17 → 32)
14 : (33 → 12)
15 : (5 → 17)
16 : (18 → 33)
17 : (1 → 18)

(e) $n = 17$

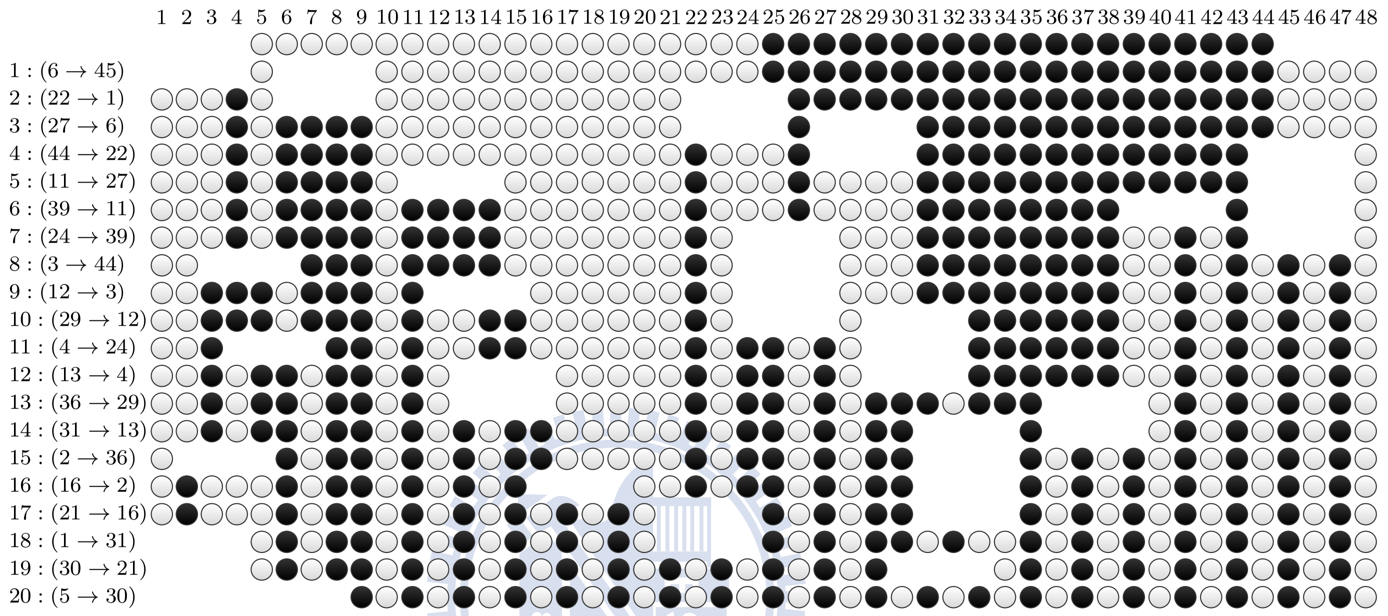


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44
1 : (6 → 41)
2 : (20 → 1)
3 : (25 → 6)
4 : (40 → 20)
5 : (13 → 25)
6 : (35 → 13)
7 : (22 → 35)
8 : (3 → 40)
9 : (32 → 3)
10 : (4 → 22)
11 : (19 → 32)
12 : (14 → 19)
13 : (27 → 14)
14 : (11 → 4)
15 : (17 → 11)
16 : (1 → 17)
17 : (14 → 27)
18 : (5 → 14)

(f) $n = 18$


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
1 : $(6 \to 43)$
2 : $(23 \to 1)$
3 : $(28 \to 6)$
4 : $(42 \to 23)$
5 : $(11 \to 28)$
6 : $(37 \to 11)$
7 : $(18 \to 37)$
8 : $(3 \to 18)$
9 : $(25 \to 3)$
10 : $(4 \to 42)$
11 : $(12 \to 4)$
12 : $(31 \to 12)$
13 : $(15 \to 25)$
14 : $(36 \to 31)$
15 : $(30 \to 15)$
16 : $(5 \to 30)$
17 : $(27 \to 36)$
18 : $(14 \to 27)$
19 : $(1 \to 14)$

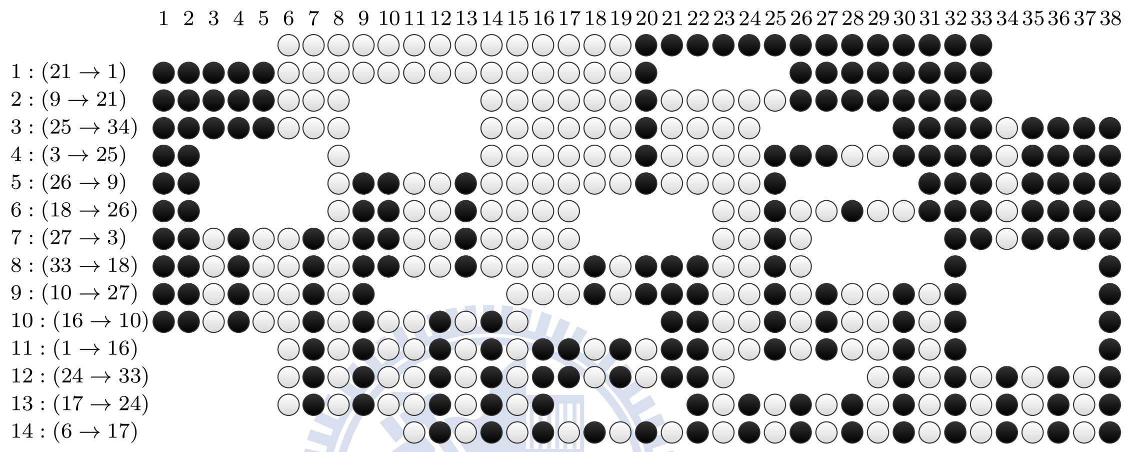
(g) $n = 19$



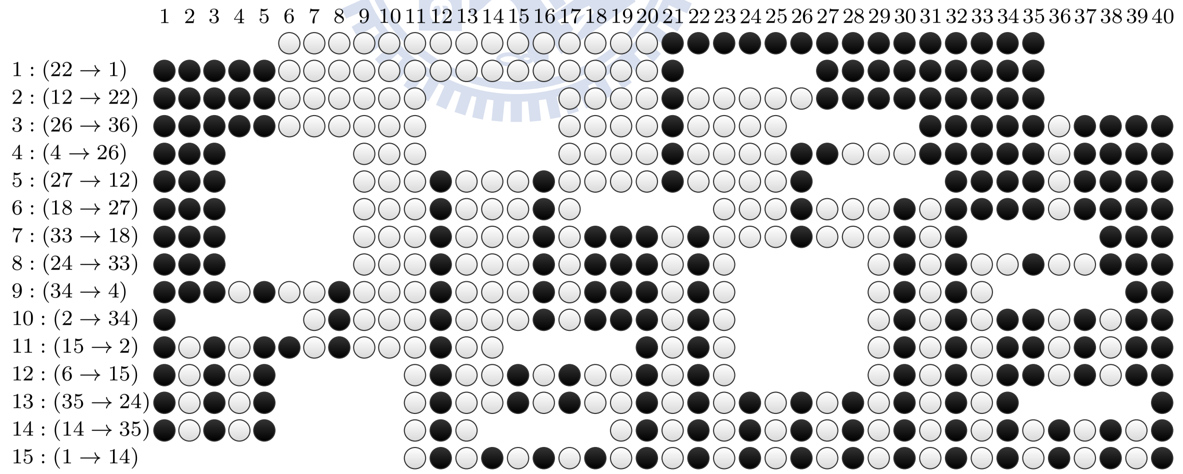
(h) $n = 20$

Figure A.1: The optimal solutions for $k = 4$ and each corresponding n.

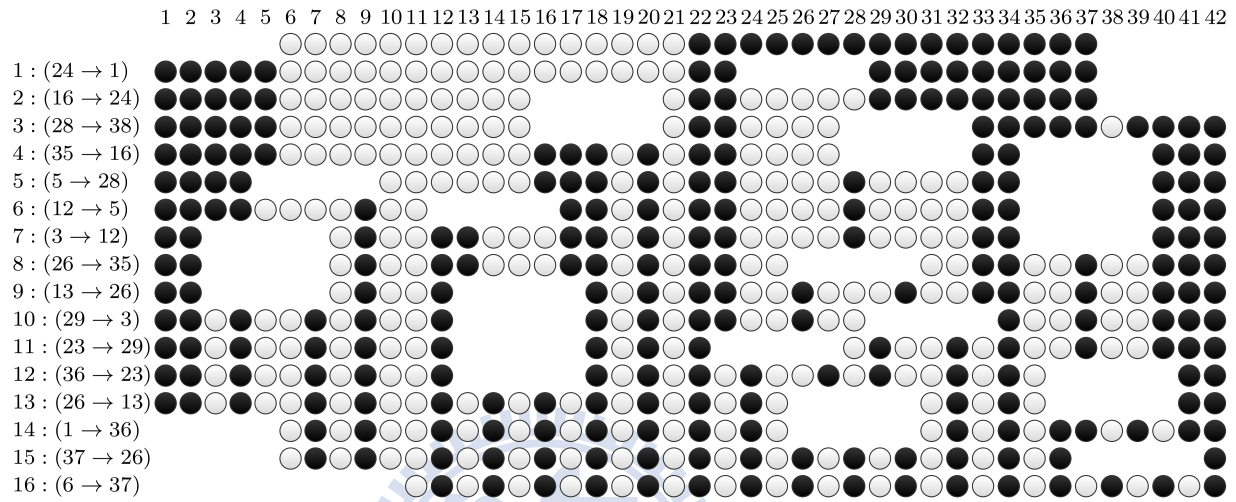
A.2 $k = 5$



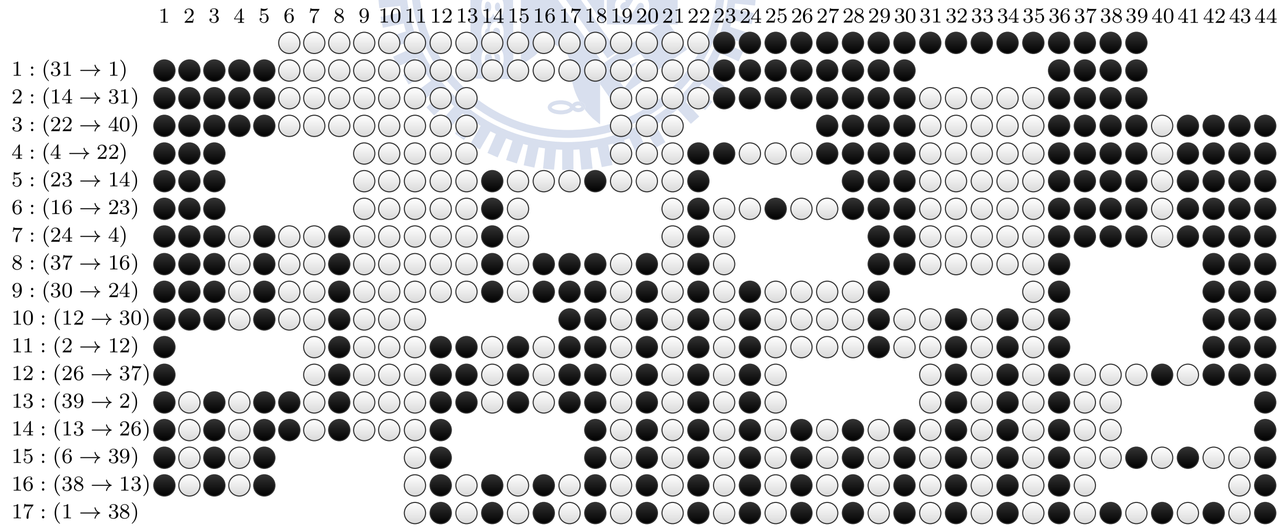
(a) $n = 14$



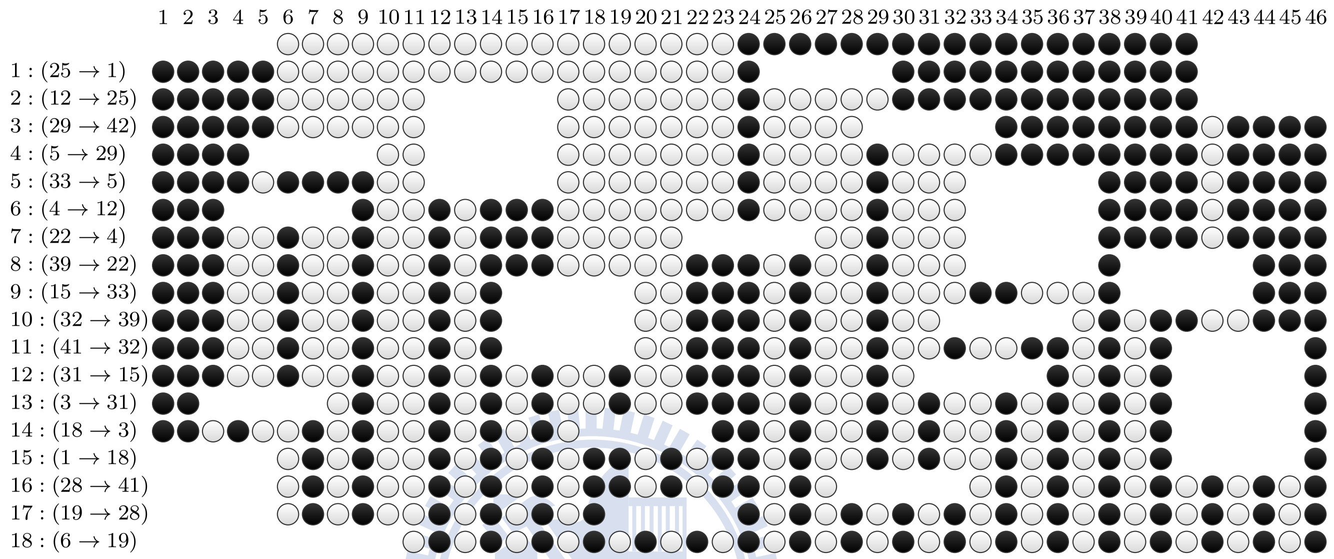
(b) $n = 15$


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42
1 : (24 → 1)
2 : (16 → 24)
3 : (28 → 38)
4 : (35 → 16)
5 : (5 → 28)
6 : (12 → 5)
7 : (3 → 12)
8 : (26 → 35)
9 : (13 → 26)
10 : (29 → 3)
11 : (23 → 29)
12 : (36 → 23)
13 : (26 → 13)
14 : (1 → 36)
15 : (37 → 26)
16 : (6 → 37)

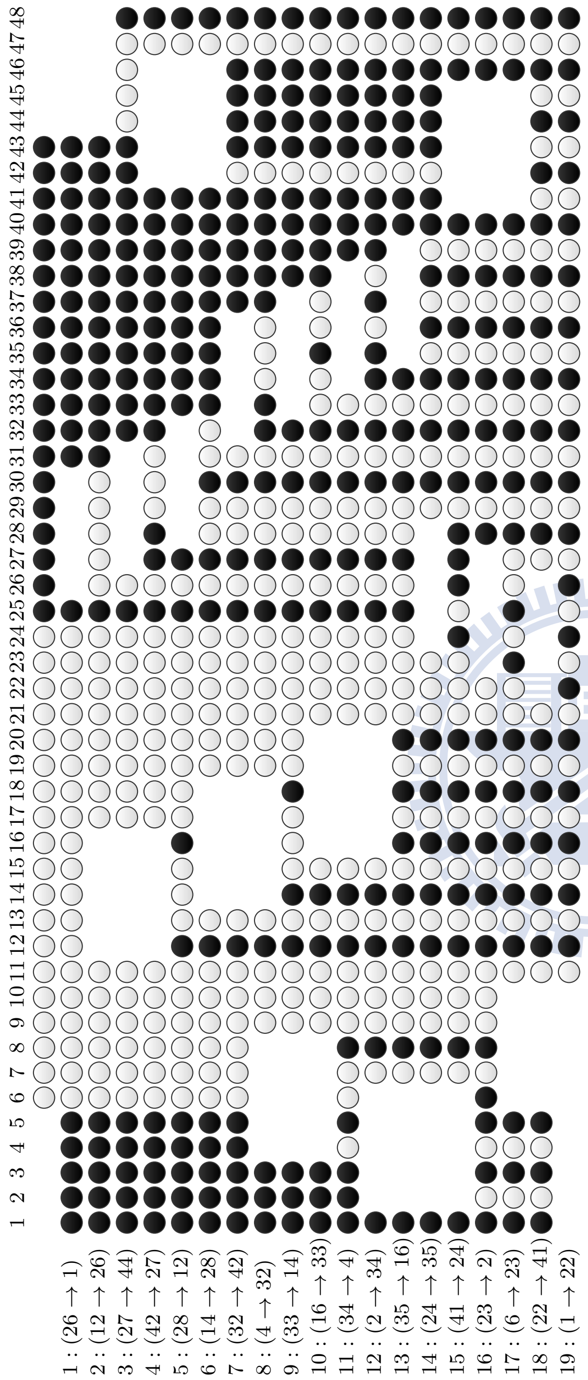
(c) $n = 16$

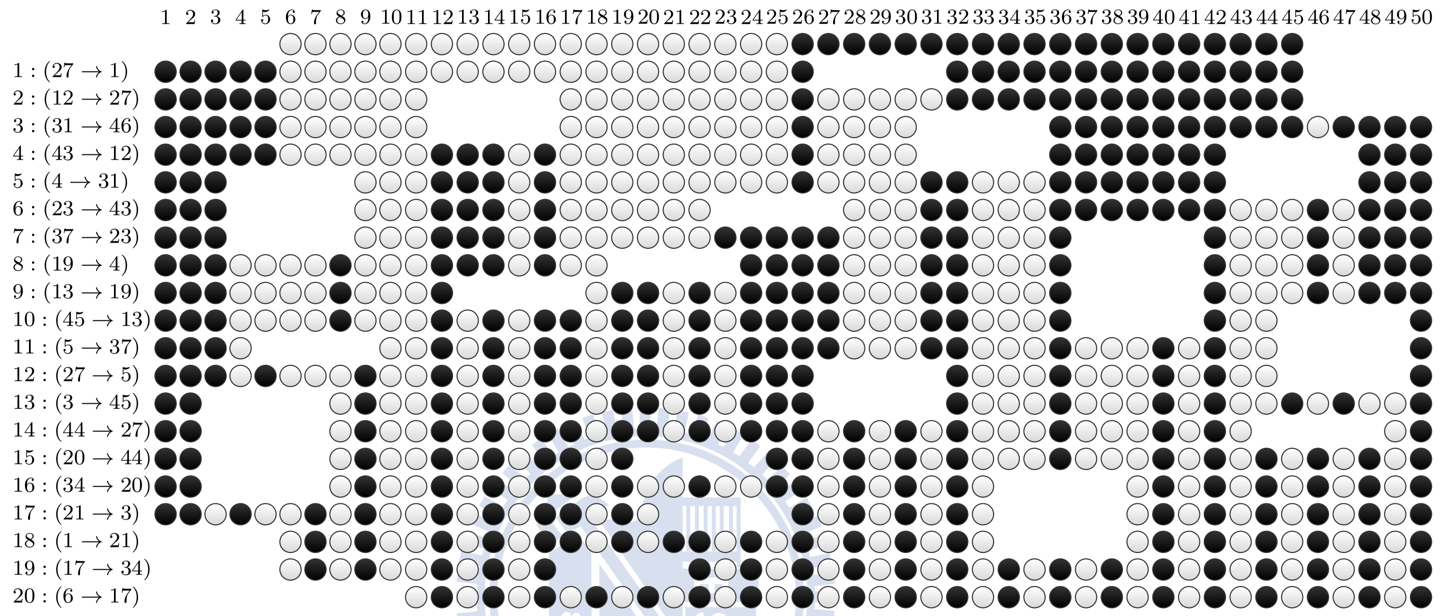

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44
1 : (31 → 1)
2 : (14 → 31)
3 : (22 → 40)
4 : (4 → 22)
5 : (23 → 14)
6 : (16 → 23)
7 : (24 → 4)
8 : (37 → 16)
9 : (30 → 24)
10 : (12 → 30)
11 : (2 → 12)
12 : (26 → 37)
13 : (39 → 2)
14 : (13 → 26)
15 : (6 → 39)
16 : (38 → 13)
17 : (1 → 38)

(d) $n = 17$

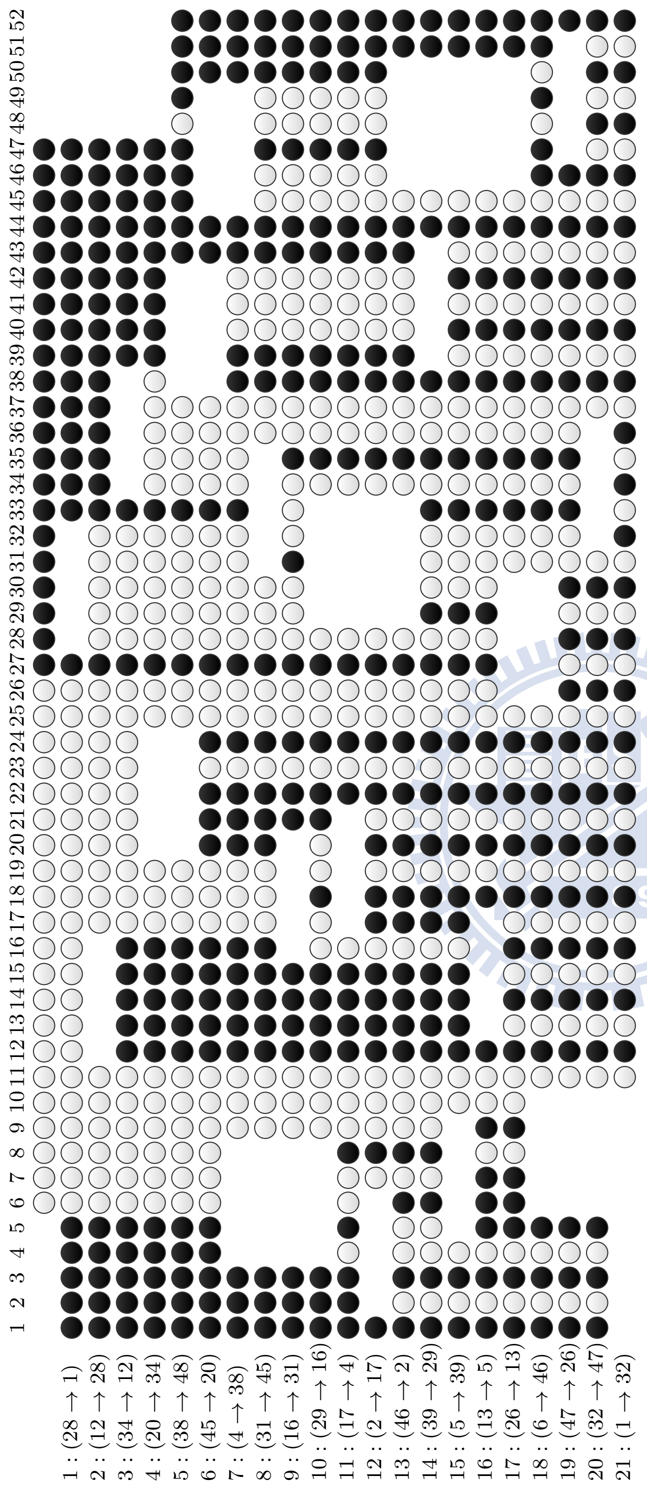

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
1 : (25 → 1)
2 : (12 → 25)
3 : (29 → 42)
4 : (5 → 29)
5 : (33 → 5)
6 : (4 → 12)
7 : (22 → 4)
8 : (39 → 22)
9 : (15 → 33)
10 : (32 → 39)
11 : (41 → 32)
12 : (31 → 15)
13 : (3 → 31)
14 : (18 → 3)
15 : (1 → 18)
16 : (28 → 41)
17 : (19 → 28)
18 : (6 → 19)

(e) $n = 18$

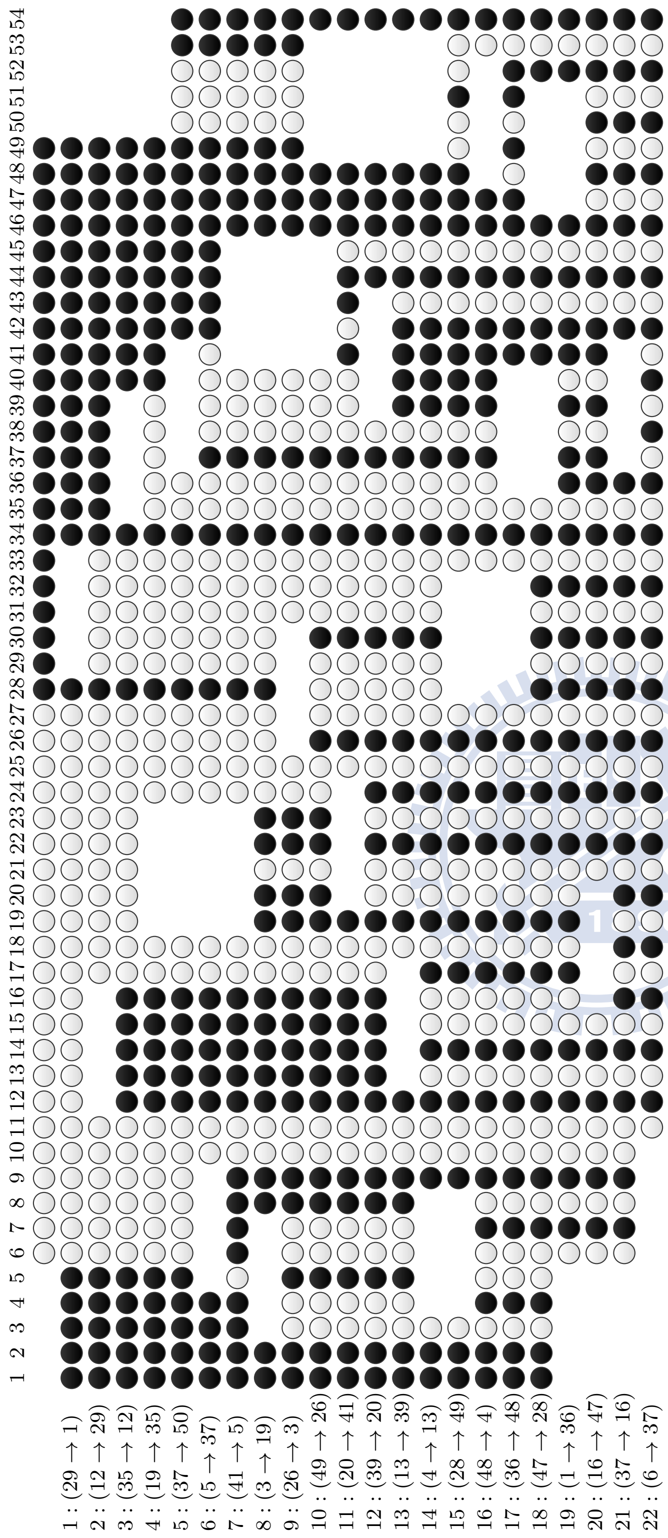

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
1 : (26 → 1)
2 : (12 → 26)
3 : (27 → 44)
4 : (42 → 27)
5 : (28 → 12)
6 : (14 → 28)
7 : (32 → 42)
8 : (4 → 32)
9 : (33 → 14)
10 : (16 → 33)
11 : (34 → 4)
12 : (2 → 34)
13 : (35 → 16)
14 : (24 → 35)
15 : (41 → 24)
16 : (23 → 2)
17 : (6 → 23)
18 : (22 → 41)
19 : (1 → 22)
(f) $n = 19$


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
1 : (27 → 1)
2 : (12 → 27)
3 : (31 → 46)
4 : (43 → 12)
5 : (4 → 31)
6 : (23 → 43)
7 : (37 → 23)
8 : (19 → 4)
9 : (13 → 19)
10 : (45 → 13)
11 : (5 → 37)
12 : (27 → 5)
13 : (3 → 45)
14 : (44 → 27)
15 : (20 → 44)
16 : (34 → 20)
17 : (21 → 3)
18 : (1 → 21)
19 : (17 → 34)
20 : (6 → 17)

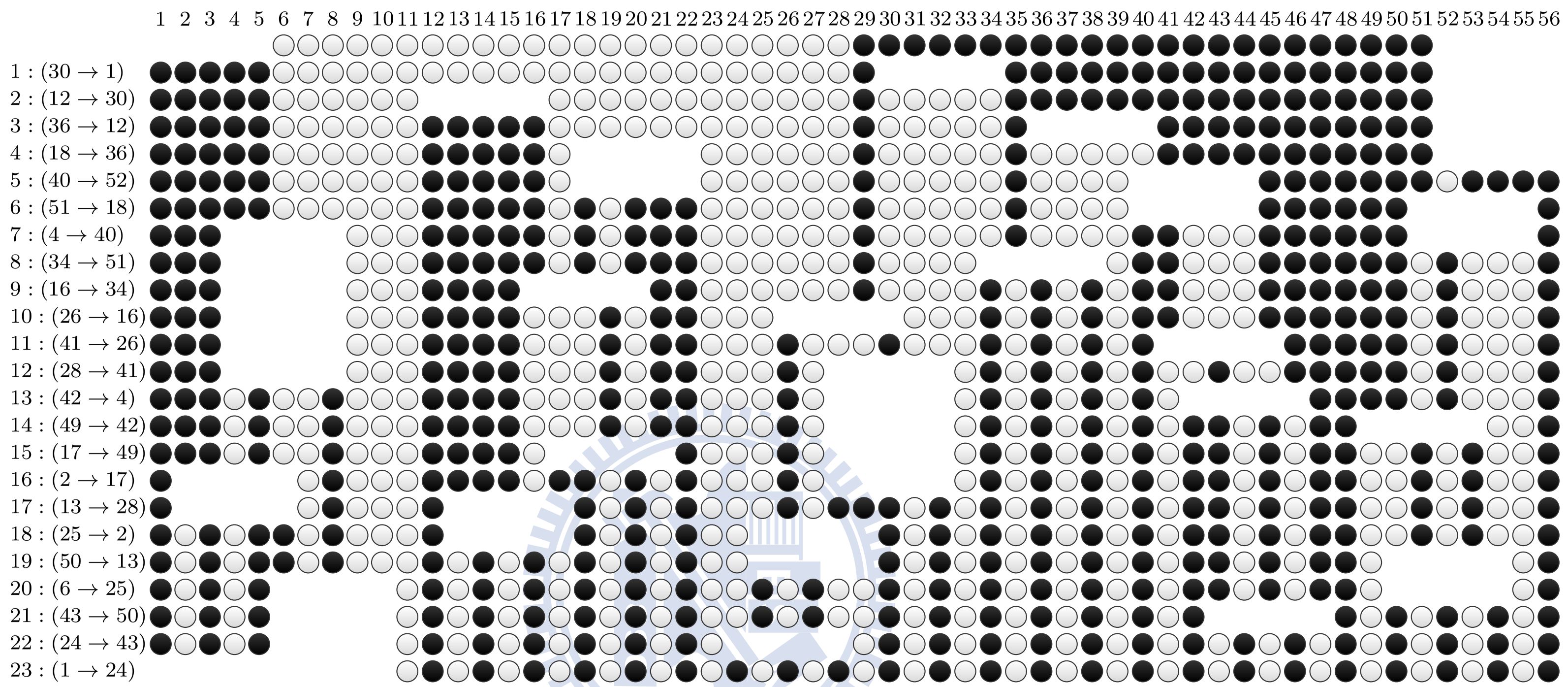
(g) $n = 20$


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52
1 : (28 → 1)
2 : (12 → 28)
3 : (34 → 12)
4 : (20 → 34)
5 : (38 → 48)
6 : (45 → 20)
7 : (4 → 38)
8 : (31 → 45)
9 : (16 → 31)
10 : (29 → 16)
11 : (17 → 4)
12 : (2 → 17)
13 : (46 → 2)
14 : (39 → 29)
15 : (5 → 39)
16 : (13 → 5)
17 : (26 → 13)
18 : (6 → 46)
19 : (47 → 26)
20 : (32 → 47)
21 : (1 → 32)

(h) $n = 21$


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
1 : (29 → 1)
2 : (12 → 29)
3 : (35 → 12)
4 : (19 → 35)
5 : (37 → 50)
6 : (5 → 37)
7 : (41 → 5)
8 : (3 → 19)
9 : (26 → 3)
10 : (49 → 26)
11 : (20 → 41)
12 : (39 → 20)
13 : (13 → 39)
14 : (4 → 13)
15 : (28 → 49)
16 : (48 → 4)
17 : (36 → 48)
18 : (47 → 28)
19 : (1 → 36)
20 : (16 → 47)
21 : (37 → 16)
22 : (6 → 37)

(i) $n = 22$



(j) $n = 23$

Figure A.2: The optimal solutions for $k = 5$ and each corresponding n.