On a Class of Board Games

Manaswi

M.Tech. Computer Science
Indian Statistical Institute, Kolkata

A dissertation submitted in partial fulllment of the requirements for the degree of
Master of Technology in Computer Science

July 2018

CERTIFICATE

This is to certify that the dissertation titled **"On a Class of Board Games"** submitted by **Manaswi** to Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of **Master of Technology in Computer Science** is a bona fide record of work carried out by him under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.

Arijit Bishnu
Associate Professor,
Advanced Computing and Microelectronics Unit,
Indian Statistical Institute,
Kolkata-700108, INDIA.

Acknowledgements

I am extremely grateful to my advisor **Dr. Arijit Bishnu** for constant support and encouragement. He provided me with exciting problems to think upon and was always available for discussions.

Abstract

Lab-on-Chip (LOC) is a device on which several laboratory functions are integrated on a chip of very small size. LOC can detect and manipulate microorganisms by applying electric field so that they can be collected at desired locations. The manipulation step involves swapping the content of one cell with another. The aim is to separate desired microorganisms from undesired ones by collecting them in their respective receptors.

In this dissertation we abstract the setting of LOC to Board Games and study their efficiency and complexity. We begin identifying the desired chips as red cells or blue cells and chips that we do not care about as black cells. The interpretation may vary depending of context. **Game-0** is used to define the setting for the games we study. We give a proof that **Game-0** is in *PSPACE* and then define **Game-1** and **Game-2** which are our main focus in this thesis.

Game-1 is *NP-Complete* which follows from a reduction from Rectilinear Steiner Tree problem. On approximation side, it admits a Polynomial Time Approximation Scheme (*PTAS*). The ideas used in these proofs are also used to study **Game-1**. We give an $\mathbf{O}(1+\frac{1}{n})$ approximation for **Game-1**. For each of these games we also study their integer linear programs.

Contents

List of Figures

Chapter 1

Introduction

Board Games (Games) are games of complete information played on two dimensional grid. They can serve as model to abstract various practical problems of interest. The Games that we study in this dissertation are motivated by sorting biological cells with Lab-on-Chip [4].

1.1 Biological cell sorting and Lab-on-Chip

Rare cell populations occur is small concentration is samples. Such samples are enriched by magnetic or automatic cell picking followed by manual or automatic cell picking or analysis [6]. There are various applications of this process in stem cell research, cell therapy and cell based diagnosis.

Lab-on-Chip (LOC) is a device on which several laboratory functions are integrated on a chip of very small size. LOCs can handle very small very of fluid and equipped with sensing, processing and actuation functions they can be used for cell sorting purpose. The main principle that allows LOCs to displace and manipulate microorganisms is dielectrophoresis (DEP). Dielectrophoresis refers to the physical phenomena in which a force is exerted on neutral particles when it is subject to non-uniform electric field. Microorganism detection is achieved using DEP cage approach and impedance sensing. Differences in permittivity and conductivity between the particles and the suspending medium is used for detecting and then manipulating the microorganisms. The manipulation is done by applying electric fields using DEP.

Thus LOC can detect and move microorganisms. The microorganisms are moved by applying electric field so that they can be collected at desired locations. The first step is to prepare the sample on a cell array by culturing it with some reagent so that normal or desired samples can be differentiated from undesired ones, thus enabling detection. The manipulation step involves swapping the content of one cell with another. The aim is to

separate desired microorganisms from undesired ones by collecting them in their respective receptors.

We model this problem by representing the array by a matrix where each entry of the matrix can be empty (white), desired cell (red) or undesired cell (blue). There are two receptors: one for the desired cells and one for undesired cells. The aim to collect the the cells at their respective receptors.

1.2 Problem Definition

We define several problems inspired from the above model. The general setup in all these problems is similar to **Game-0** described below.

1.2.1 Game-0

Let $A = (a_{ij})$ be a $m \times n$ matrix. Each entry(or cell) a_{ij} is assigned a color which can be Red(**R**), Blue(**B**), Black(**K**) or White(**W**). The white cell will also be called Empty cell. Such an arrangement of colors on cells of A is called *Configuration of A*. C_0 denotes the *Initial Configuration of A* and C_i is *Configuration of A* at some step $i \in \mathbb{N}$. Two cells at Manhattan Distance one are called *Adjacent Cells*.

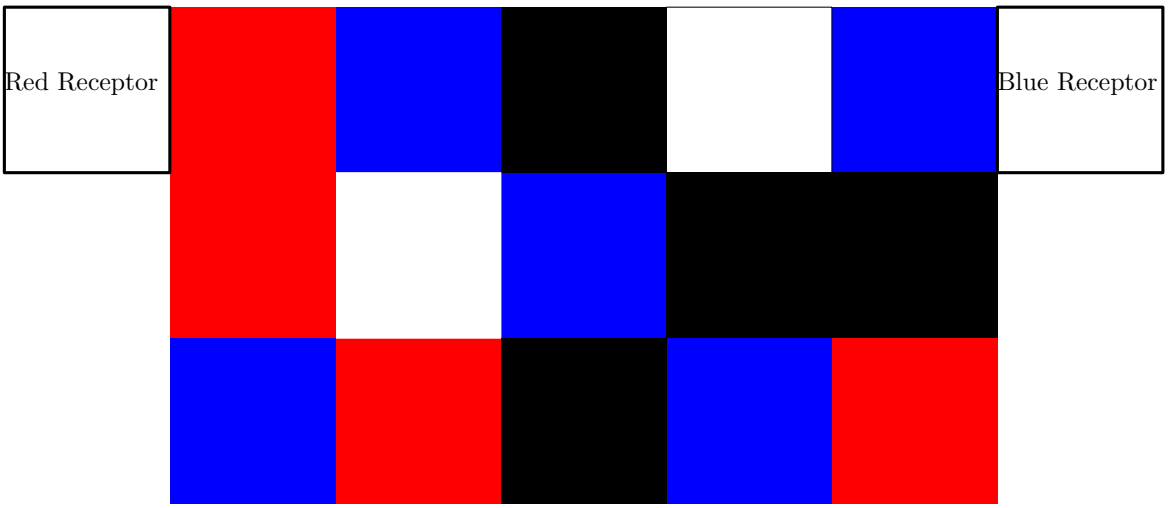

Figure 1.1 Instance of **Game-0**

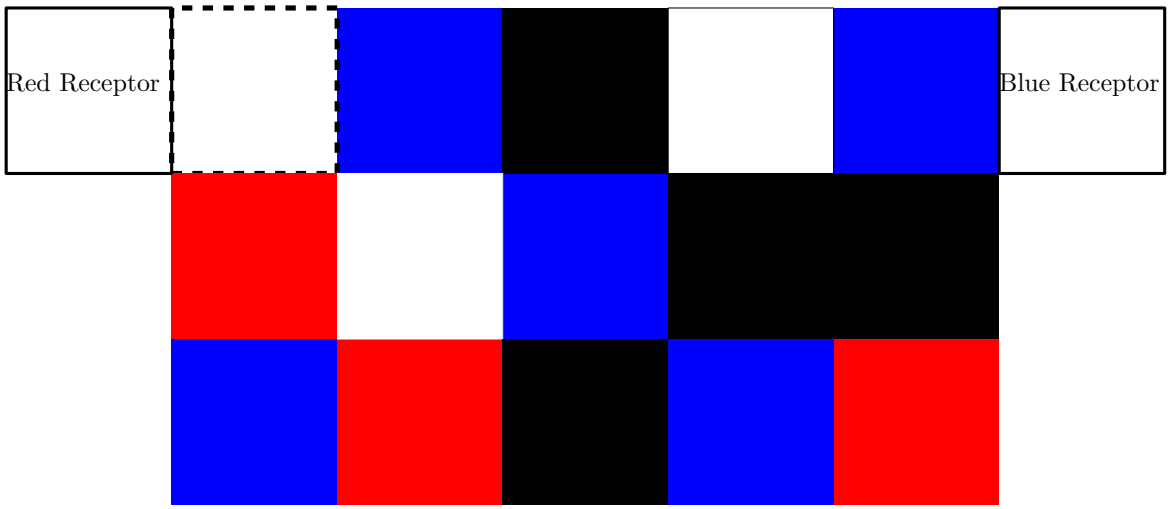

Figure 1.2 Rule 1 of **Game-0** applied to Figure 1.1

There are two receptors, one for red cells and another for Blue cells. The Red Receptor is placed adjacent to (left) one of the entries of first column of the matrix while the Blue receptor is placed adjacent to (right) one of the entries of the last column of the matrix. Once the Receptors are initialized, their position does not change for the rest of the game i.e. for all configurations C_i the position of Red and Blue Receptor is the same. In the next step one of three things can happen. This changes the configuration of **Game-0** from C_i to C_{i+1}.

1. if a red cell is adjacent to the Red Receptor, in one step, the red cell can be "swapped out" by Red Receptor and replaced by an empty cell

2. if a blue cell is adjacent to the Blue Receptor, in one step, the blue cell can be "swapped out" by Blue Receptor and replaced by an empty cell.

3. a white cell can swap its position with a non-White cell adjacent to it

Thus all swaps happen between adjacent cells and each such swipe takes one step. Also each swap involves swap with an adjacent white cell or creation of a new white cell.

Let C_F denote the configuration (or a set of configurations) of the matrix **Game-0** which we want to reach from a given initial configuration C_0. We say the configuration C_0 *is* ***Solvable*** *for the configuration* C_F if there exists a number $k \in \mathbb{N}$ such that there exists sequence of configurations $C_0, \ldots, C_i, C_{i+1}, \ldots, C_k$ where $C_k = C_F$ ($C_k \in C_F$). k is the number of steps or *time* taken to solve C_0. The smallest k for which such a sequence exists is called *Optimum Solution of* C_0 and denoted by $OPT(C_0)$(OPT when clear from context).

To make analysis easier, we define the notion of a **"move"**. A *Contiguous Section* of white cells is a set of cells in a configuration C_i such that any two cells are connected by

a Manhattan path. When a contiguous section of white cells are present, a non-white cell adjacent to this section can freely move to any position inside the section. We call such movements **"move"** and count such movements to take 1 unit of time. If contiguous sections are not present then move and step denote the same thing. **Objective** of **Game-0** is to clear the Red and Blue chips in minimum number of **moves**. We do not care about the black chips in any of the games. They act as obstacles and can be swapped out from either Red or Blue Receptor.

The notions defined in **Game-0** serve as the basis for defining other games that follow. Our analysis is mainly for the next two games and results we obtain there are applicable to **Game-0** as well.

Figure 1.1 shows a configuration of a 3×5 matrix A where each cell is Red, Blue, Black or Empty (White). The Red Receptor is adjacent to $a_{1,1}$ and the Blue Receptor is adjacent to $a_{1,5}$. One of the possible next steps is the red cell located at $a_{1,1}$ getting swapped out by Red receptor and empty cell is created at $a_{1,1}$ as shown in Figure 1.2.

1.2.2 Game-1

Let $A = (a_{ij})$ be an $m \times n$ matrix. Each entry (or cell) a_{ij} is assigned a color which can be Red, Black or White(Empty). There is a single receptor for the Red cells which is placed to the left of one of the entries of first column of the matrix. All other things are same as **Game-0**, in particular, both Red and Black cells can be swapped out by the Red receptor.

Let $\mathscr{C} = \{C \mid C \text{ is a configuration of } \textbf{Game-1} \text{ and } C \text{ has no red cells}\}$. An instance of **Game-1** is given by an initial configurations C_0 and the game is solvable if there exists a number $k \in \mathbb{N}$ such that there exists a sequence of configurations $C_0, \ldots, C_i, C_{i+1}, \ldots, C_k$ such that $C_k \in \mathscr{C}$. The **objective** of **Game-1** is to clear all the Red chips in minimum number of steps.

Figure 1.3 shows a 2×3 matrix A where each cell is Red, Black or Empty (White). There is a single Red Receptor adjacent to $a_{2,1}$. One of the possible next steps is the Black cell located at $a_{2,1}$ getting swapped out by Red receptor and empty cell is created at $a_{2,1}$ as shown in Figure 1.4.

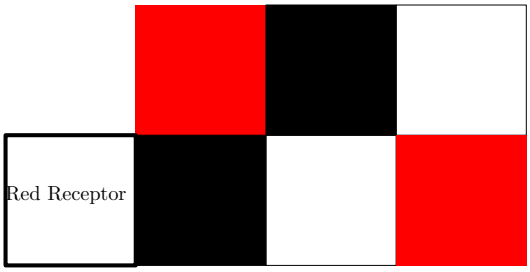

Figure 1.3 Instance of **Game-1**

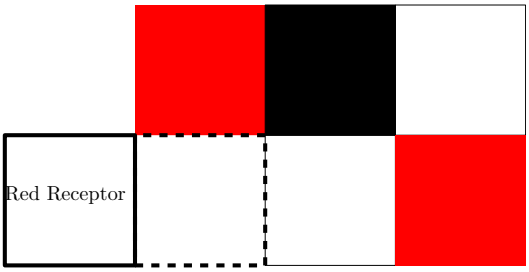

Figure 1.4 A possible next configuration

1.2.3 Game-2

Let $A = (a_{ij})$ be an $m \times n$ matrix. Each entry (or cell) a_{ij} is assigned a color which can be Red, Blue or White. There are two receptors, one for Red cells and another for Blue cells. The Red receptor is placed to the left of one of the entries of first column of the matrix while the Blue receptor is placed to the right of one of the entries of the last column of the matrix. A Red chip can only be swapped out by the Red receptor and a Blur chip can be swapped out only by the Blue Receptor in one step.

Let C_E = Configuration of **Game-2** with all empty cells. An instance of **Game-2** is given by the initial configuration C_0 and the game is solvable if there exists a number $k \in \mathbb{N}$ such that there exists sequence of configurations $C_0, \ldots, C_i, C_{i+1}, \ldots, C_k$ such that $C_k = C_E$. The **objective** of **Game-2** is to clear both the Red and Blue chips in minimum number of steps.

1.3 Game-0 is in *PSPACE*[1]

We show **Game-0** is in *PSPACE*. Given a configuration C_i, the next possible configurations depend on white cells and their location, or if a new white cell can be created by the Red/Blue Receptor. Let the possible next configurations be $C_{i1}, C_{i2} \ldots, C_{ik}$. Given C_i and C_{ij} we want to determine $C_{i(j+1)}$ such that $C_{i(j+1)} \notin \{C_{i1}, C_{i2} \ldots, C_{ij}\}$. To do this we first fix an enumeration scheme, given configurations C_i:

1. Swap the red cell out of the Red Receptor to get C_{i1}
2. Swap the blue cell out of the Blue Receptor to get C_{i2}
3. Pick the left most and top most unmarked cell. Swap the cell in order up, down, left and right to obtain next configurations. After enumerating all possible configurations attainable from this cell, mark the cell

[1] This section is based on preliminary work on the problem. The results in the next chapters make this result infructuous

4. Repeat Step 3 to get all possible configurations

Thus given C_i and C_{ij}, we can obtain $C_{i(j+1)}$ using polynomial space.

In section 3.3 we see that optimum is upper bounded by a polynomial in size of the input, which puts a polynomial bound on the number of configurations, say K. So we start enumerating from C_0 and follow the first line of computation for at most K steps i.e. $C_0, C_{01}, C_{011} \ldots, C_{01\ldots1_{(K-1)}}$. If all red and blue cells are removed in K steps, we stop. Note that using only C_0 and C_{01} we can find C_{02}. Thus it is possible to enumerate all possible paths on computation in polynomial space, as we have to store only the last sequence of computation which is at most polynomial in the length of input.

1.3.1 Related Work

Problems in similar setting have been analyzed in [7] and [2].

The paper [7] defines a problem $GMP1R$ where the input is a connected, undirected graph G on n vertices with a mobile robot on one of the vertices labelled s. The other vertices may contain an obstacle or may be free.The robot or the obstacle can move along the edge to an adjacent empty vertex in one step. A target vertex t for the robot is also given. The aim is to move from s to t in minimum number of moves. The paper shows the problem to be *NP-Complete* for general graphs as well as planar graphs. A generalization of the problem $GMP1R$ is to $GMP1kR$ where there are k robots and each has their destination vertices. For $k = n-1$ on a grid, $GMPkR$ reduces to *15-Puzzles Problem* which is similar to the problems discussed in this thesis. It is known that *15-Puzzles Problem* on 4×4 grid admits a solution if and only if the given permutation is an even permutation [5]. Also, the $(N \times N)$ extension of the *15-Puzzles Problem* are known to be *NP-Complete* [8].

The results in [2] are more closely related to our problem. The paper studies the problem of *Motion Planning* on graphs and grids. Let G be a graph and V and V' be two subsets of vertices of the graphs. A move is defined as shifting a chip from v_1 to v_2 ($v_1, v_2 \in V(G)$) on a path formed by edges of G so that no intermediate vertices are occupied. The problem is studied for the case when G is a graph or grid and when the chips are labelled or un-labelled, giving four variants of the problem. Of these, the problem U-GRID-RP, i.e. motions planning on grid with un-labelled chips is directly related to our work. The paper shows the problem to be *NP-Complete*.

1.4 Organization of Thesis

We started the problem definitions with **Game-0** which helped us to define other games in a general setting but we are mainly interested in complexity of **Game-1** and **Game-2**. The results in these games translate to **Game-0** as well.

In Chapter 2 we study the decision and optimization versions of **Game-1** i.e. the problems **D-Game-1** and **O-Game-1** respectively. We study the complexity, integer linear program and approximation algorithm for these problems. The setting and techniques used in this chapter translate to Chapter 3 where we study integer linear program and approximation algorithm for **Game-2**.

Chapter 2

Game-1

2.1 Problem

The general setting of the problem is same as defined in section 1.2.2. The **input** of the problem is initial configurations C_0. The objective is to clear all red cells. Next we define the corresponding Decision and Optimization Problem. The Decision Problem asks if all the Red cells can by cleared using the rules of the game in at most k moves, $k \in \mathbb{N}$. The Optimization Problem asks for the minimum number of steps to clear all the red cells.

2.1.1 Decision Problem(D-Game-1)

Given **Game-1** as input C_0 and integer $k \in \mathbb{N}$, is it possible to remove all red cells in C_0 in at most k moves.

2.1.2 Optimization Problem(O-Game-1)

Given **Game-1** as input C_0, find the smallest positive integer $k \in \mathbb{N}$ such that in k moves, all red cells of C_0 can be removed.

2.2 D-Game-1

2.2.1 Solvability of Game-1

We first prove that **Game-1** is always solvable i.e. given any configuration C_0 of **Game-1** it is always possible to reach a configuration C_F with no red cells in finite number of steps,

following the rules of the game. The reason for this is that we do not care about black cells as they can be swapped out from Red receptor.

The algorithm to do this is trivial. If there is a red/black cell adjacent to Red Receptor then swap it out replacing with a white cell. If this is not the case then a white cell is adjacent to Red Receptor, say at $(1,1)$. Consider a red cell located at position (i,j). Moving the white cell along the path: $(1,1),(1,j),(i,j),(i,1),(1,1)$ decreases the Manhattan Distance of every Red cell in the path $(i,j),(i,1),(1,1)$ by 1. Thus repeatedly cycling the white cell at $(1,1)$ at most $(i+j))$ times brings at least one red cell adjacent to the Red Receptor. Each such cycle takes $2\times(i+j)$ steps. Thus, in $\mathbf{O}(nm(n+m)^2)$ moves we can clear all Red cells. This is illustrated in Figure 2.1.

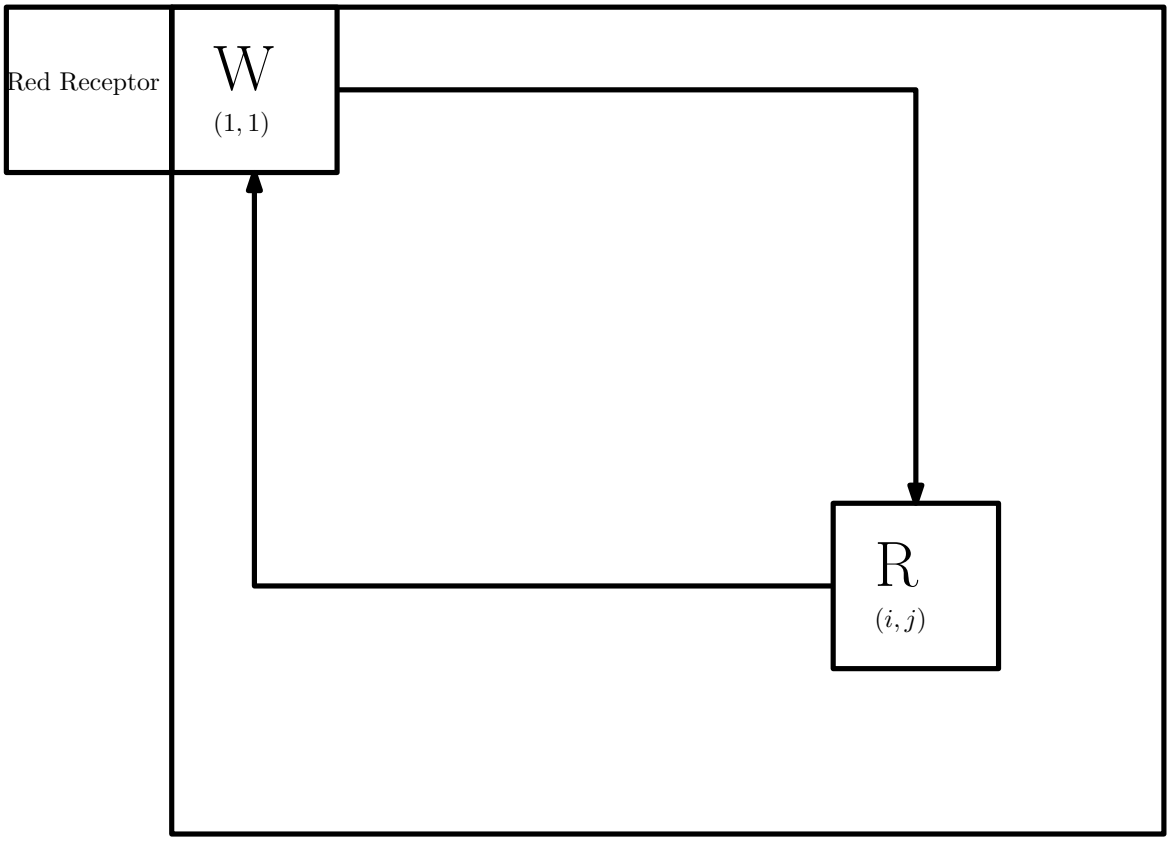

Figure 2.1 Cycling white cell to remove any red cell

2.2.2 Bound on Optimum

Using the algorithm in section 2.2.1 we get an upper bound on the number of steps needed to solve the problem, i.e. an upper bound for **O-Game-1**. Since in maximum

$2nm(n+m)^2$ steps we can clear entire board, the optimum for **O-Game-1** must be less than $2nm(n+m)^2$.

2.2.3 D-Game-1 is in *NP*

Input of **D-Game-1** is the initial configuration C_0 and a positive integer k. We now show **D-Game-1** is in *NP* by making a simple certificate. The certificate consists of k' configurations $C_1, \ldots, C_{k'}$. A deterministic Turing machine checks:

1. k' should be at most k i.e. $k' \leq k$
2. C_{i+1} can be obtained from C_i following the rules of the game
3. the configuration $C_{k'}$ has no Red cells

Clearly, the certificate is polynomial in size of input with k configurations each of size nm. Checking if C_{i+1} can be obtained from C_i following the rules of the game can be done in $\mathbf{O}(nm)$ time. Each move of the game consists of either swapping out a Red/Black cell from the Red receptor or swapping one of the White cells with its adjacent cell or moving a Red/Black chip in contiguous section of white cells. There are at most 4 adjacent cell of each white cell and at most nm white cells. Other two checks can also be done in polynomial time. Thus the certificate is polynomial in size of input and can be verified in polynomial time.

2.2.4 D-Game-1 is *NP-Complete*

We have already shown that **D-Game-1** is in *NP*. Next we show that it is *NP-Hard*. We do this by reducing **Rectilinear Steiner Tree** to **D-Game-1**. This proof closely follows [2].

The **Rectilinear Steiner Tree** (**R-Steiner**) is a Decision Problem defined as follows. We are given n points in $\mathbb{R}^2$ and the aim is to find a tree, consisting of only horizontal and vertical segments (in $\mathbb{R}^2$), that connects all points of length at most k. We can assume that points have integer coordinates and further we can assume that all coordinates are given in unary since **R-Steiner** is strongly *NP-Hard* [3]

Consider an instance of **R-Steiner** i.e. a set of n points $\{p_1, \ldots, p_n\}$ with integer coordinated in $\mathbb{R}^2$ such that p_1 is the left-most point. Let δ be the smallest axis parallel rectangle enclosing all points. Now we construct **Game-1** corresponding to given instance of **R-Steiner**. We map δ to a board of same dimensions where corresponding to each point p_i we put a red cell, the remaining cells are colored black. The Red Receptor is placed adjacent

to p_1. An instance of **R-Steiner** is shown in Figure 2.2 and Figure 2.3 which is taken from [9].

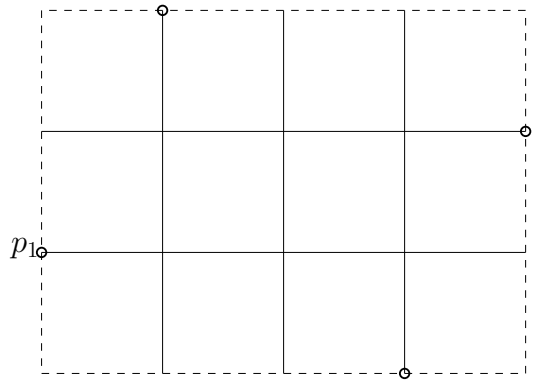

Figure 2.2 Set of points in a grid

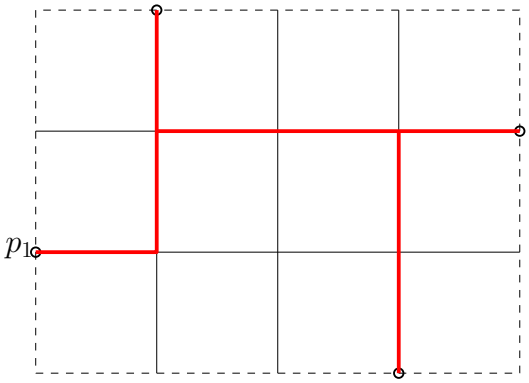

Figure 2.3 RST corresponding to Figure 2.2

The corresponding reduction to **Game-1** is shown in Figure 2.4.

	B	*R*	B	B	B
	B	B	B	B	*R*
Red Receptor	*R*	B	B	B	B
	B	B	B	*R*	B

Figure 2.4 Reduction from Fig 2.2

Theorem 2.2.1. There is a rectilinear Steiner tree of length at most k if and only if the corresponding **Game-1** can be solves in at most k moves.

Proof 2.2.1. If there is a rectilinear Steiner tree of length k, then first remove p_1 in one move. This creates a white cell and in next move another Red/Black square can be removed along the tree and the process continues inductively. As our construction ensures the number of Red/Black squares to be exactly equal to the length of rectilinear Steiner tree, we can remove all Red cells in at exactly k steps.

Conversely, the minimum number of steps that p_i must take to reach the Red Receptor is the Manhattan distance between p_i and p_1. There are several paths that can be taken by p_i to reach the Red Receptor. Let V_i denote the set coordinates taken by p_i in the optimum solution. The union $V = \bigcup_i^n V_i$ are all the squares that move. $|V|$ is minimum when v_i's follow Steiner tree. This is because each v_i leaves the board from v_1 so V represents connected graph say G. Further we can remove loops while preserving path length of all v_i thus giving a tree. The length of the tree which corresponds to the number of moves of the game is minimum when the length of the tree is minimum. This happens when V represents (rectilinear) Steiner tree.

We have proved **D-Game-1** is *NP-Complete*. Now we look at the optimization version of the problem **O-Game-1**.

2.3 O-Game-1

2.3.1 Integer Linear Program

In this section we give an ILP to solve the problem. Our goal here is to minimize the number of steps.

2.3.2 Variables

Let us first define some sets and variables.

1. $I = \{1,2,3,\ldots,mn\}$. I is the set of cells i.e. I is labeling of cells of the board. This is illustrated in Figure 2.5

Red Receptor	1	2	3	4
	5	6	7	8
	9	10	11	12

Figure 2.5 Labeling of tiles of the game

2. $R = \{1,2,\ldots,r\}$ is the set of Red cells
3. $W = \{1,2,\ldots,w\}$ is the set of White cells
4. $K = \{1,2,\ldots,k\}$ is the set of Black cells
5. T is total time for which we allow the game to run which should an upper-bound on OPT. We obtained an upper bound on OPT in section 2.2.2. Thus we set $T = \{1,2,3,\ldots,2mn(m+n)^2\}$
6. $PosRed(r,t)$ is position of Red cell labeled r at time t
7. $PosWhite(w,t)$ is position of White cell labeled w at time t
8. $PosBlack(k,t)$ is position of Black cell labeled r at time t
9. $UpWhite(w,t) = 1$ if wth white cell moves up at time t. It is 0 otherwise
10. $DownWhite(w,t) = 1$ if wth white cell moves down at time t. It is 0 otherwise
11. $LeftWhite(w,t) = 1$ if wth white cell moves left at time t. It is 0 otherwise
12. $RightWhite(w,t) = 1$ if wth white cell moves right at time t. It is 0 otherwise

13. $x(w,t)$ is the *x-coordinate* of white cell w at time t
14. $y(w,t)$ is the *y-coordinate* of white cell w at time t
15. $Z(x,t) = 1$ if some cell x (which may be red/black/white) moves right at time t. It is 0 otherwise

2.3.3 Constraints

1. We first set the *Pos* variables according to the initial board configuration: $PosRed(r,1)$, $PosBlack(k,1)$ and $PosWhite(w,1)$ for all red, black and white chips. Also in this step we initialize the x and y coordinated of each white chip.
2. We set constraints relating coordinates and position on board for each white chip w and at each time t:

$$PosWhite(w,t) = x(w,t) + (n \times y(w,t)) + 1 \quad \forall w \in W \;\; \forall t \in T$$

3. Set final configuration:

$$PosWhite(w,T) \geq 1 \quad \forall w \in W$$
$$PosRed(r,T) = 1 \quad \forall r \in R$$

   These equations describe the final state of the game we want. We want all the red cells to be at cell 1 when game ends. This is enforced by $PosRed(r,T) = 1$. We put no constrains on the final position of Black cells.

4. Each white cell is allowed only one move:

$$UpWhite(w,t) + DownWhite(w,t) + LeftWhite(w,t) + RightWhite(w,t) = Z(w,t)$$

   . This should hold for all white cells $w \in W$ and for all time $t \in T$

5. Only one white cell moves in each step:

$$\sum_{w} Z(w,t) = 1 \quad \forall w \in W \;\; \forall t \in T$$

6. Update the position of all white cells after each step:

$$PosWhite(w,t+1) = PosWhite(w,t) - (n \times UpWhite(w,t)) + (n \times DownWhite(w,t)) - LeftWhite(w,$$

This should hold for all white cells $w \in W$ and for all time $t \in (T \setminus \{2mn(n+m)^2\})$. This ensures proper updates for all white cells at all time steps.

The above expression tells us that if the white cell moves left or right then we just need to update its position by subtracting or adding 1, respectively. If the cell moves up or down we do just the same thing but take into account the structure of the board, which adds a factor of n.

7. Restrictions on movement of white cells:

   (a) White square at bottom edge:

   $$PosWhite(w,t) + n \times DownWhite(w,t) \leq mn$$

   (b) White square at top edge:

   $$PosWhite(w,t) - n \times UpWhite(w,t) \geq 1$$

   (c) White square at left edge:

   $$x(w,t) - LeftWhite(w,t) \geq 0$$

   (d) White square at right edge:

   $$x(w,t) + RightWhite(w,t) \geq n-1$$

8. The cell 1 always has α cells on it where α is some number larger than length of Steiner tree of red cells and smaller than nm. Rest all cells have one chip. Thus the following should hold at all time $t \in T$:

   $$\sum_{w \in W} PosWhite(w,t) + \sum_{r \in R} PosRed(r,t) + \sum_{k \in K} PosBlack(k,t) = (\alpha \times 1) + 2 + \cdots + mn$$

   This step ensures every cell is assigned a position at all time t.

9. Next we ensure that a Red or Black cell moves each time a white cell moves:

   $$\sum_{w \in W} Z(w,t) = \sum_{r \in R} Z(r,t) + \sum_{k \in K} Z(k,t)$$

Above should be true at all time $t \in T$

10. Also we ensure that if white cell does not move then red/blue cell will also not move:

$$PosRed(r,t) - PosRed(r,t+1) + nZ(r,t) \geq 0 \quad \forall r \in R \ \ \forall t \in T$$

$$PosRed(r,t) - PosRed(r,t+1) - nZ(r,t) \leq 0 \quad \forall r \in R \ \ \forall t \in T$$

Similarly:

$$PosBlack(k,t) - PosBlack(k,t+1) + nZ(k,t) \geq 0 \quad \forall k \in K \ \ \forall t \in T$$

$$PosBlack(k,t) - PosBlack(k,t+1) - nZ(k,t) \leq 0 \quad \forall k \in K \ \ \forall t \in T$$

11. Putting bounds on variables:

$$1 \leq PosWhite(w,t) \leq mn \quad \forall w \in W \ \ \forall t \in T$$

$$1 \leq PosWhite(k,t) \leq mn \quad \forall k \in K \ \ \forall t \in T$$

$$1 \leq PosWhite(r,t) \leq mn \quad \forall r \in R \ \ \forall t \in T$$

$$1 \leq x(w,t) \leq n \quad \forall w \in W \ \ \forall t \in T$$

$$1 \leq y(w,t) \leq m \quad \forall w \in W \ \ \forall t \in T$$

2.3.4 Objective Function

Minimize the following which is the total number of steps:

$$\sum_{t \in T} \sum_{w \in W} Z(w,t)$$

2.4 O-Game-1 is in PTAS

Rectilinear Steiner Tree problem is strongly *NP-Hard*. A **FPTAS** algorithm for **O-Game-1** would imply $P = NP$. Thus we look for **PTAS** algorithm which directly follows from [1] and the reduction described in section 2.2.4.

2.5 Using Game-1 for Biological Cell Sorting

Biological cell sorting in concerned with separation of desired cells which occur in low concentration. **Game-1** can be used as process to increases the concentration of desired cells (red cells).

Chapter 3

Game-2

3.1 Problem

The general setting of the problem is same as defined in 1.2.3. The **input** of the problem is initial configurations C_0. The objective is to clear all red **and** blue cells. We note that the red cells can not be swapped out by the Blue Receptor and the blue cells cant be swapped out by Red Receptor. Next we define the corresponding Decision and Optimization Problems. The Decision Problem asks if all the red and blue cells can by cleared using the rules of the game in at most k moves, $k \in \mathbb{N}$. The Optimization Problem asks for the minimum number of steps to clear all the red and blue cells. Let C_E = Configuration of **Game-2** with all empty cells.

3.1.1 Decision Problem(D-Game-2)

Given **Game-2** as input C_0 and integer $k \in \mathbb{N}$, is C_0 solvable for C_E in at most k moves.

3.1.2 Optimization Problem(O-Game-2)

Given **Game-2** as input C_0, find the smallest positive integer $k \in \mathbb{N}$ such that the configuration C_0 is solvable for C_E in k moves.

3.2 D-Game-2

3.2.1 Solvability of Game-2

We first prove that **Game-1** is solvable if and only if there is a White cell in C_0 or a white cell can be created in C_1 by swapping from Red/Blue Receptor.

If there is no white cell in C_0 but a white cell can be created in C_1, then the proof is same as that of **Game-1**. If there is a red cell adjacent to Red Receptor or a blue cell adjacent to Blue Receptor then swap it out replacing with a white cell. This created a White cell at $(0,0)$ (or at (m,n)). Consider a Red cell located at position (i,j). Moving the white cell along the path: $(0,0),(0,j),(i,j),(i,0),(0,0)$ decreases the Manhattan Distance of every Red cell in the path $(i,j),(i,0),(0,0)$ by 1. Thus in at most $(2(i+j)\times(i+j))$ steps we can bring at least one Red cell adjacent to the Red Receptor. Clearly, in $\mathbf{O}(nm(n+m)^2)$ moves we can clear all red cells. This is illustrated in Figure 2.1. Same argument holds for blue cells.

On the other hand if a white cell can't be created at Red/Blue Receptor but C_0 has a white cell, then any Red/Blue cell can be removed. The argument is similar to the above argument. This is illustrated in Figure 3.1

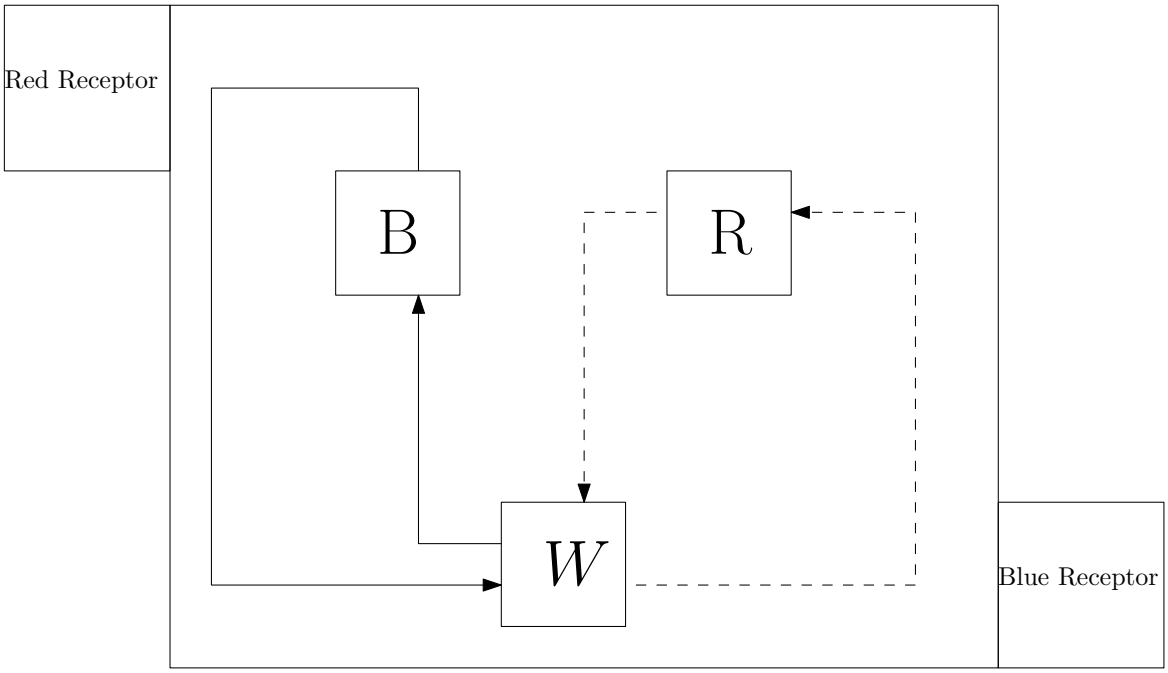

Figure 3.1 Solving **Game-2** using single white cell

On the other hand, if there are no white cells and no white cells can be created in the next move, then there are no legal moves according to the rules of the game.

3.2.2 D-Game-2 is in *NP*

Input of **D-Game-1** is the initial configuration C_0 and a positive integer k. We now show **D-Game-1** is in *NP* by making a simple certificate. The certificate consists of k' configurations $C_1, \ldots, C_{k'}$. A deterministic Turing machine checks:

1. $k' \leq k$
2. C_{i+1} can be obtained from C_i following the rules of the game
3. The configuration C'_k has only White cells i.e. C'_k is C_E

Clearly, the certificate is polynomial in size of input with k' configurations each of size nm. Checking if C_{i+1} can be obtained from C_i following the rules of the game can be done in $\mathbf{O}(nm)$ time. Each move of the game consists of either swapping out a red or blue cell from the Red receptor or Blue Receptor respectively or swapping one of the White cells with its adjacent cell or moving a Red or Blue chip in contiguous section of white cells. There are at most 4 adjacent cell of each white cell and at most nm white cells. Thus the *poly-size* certificate is verifiable in *poly-time*.

3.2.3 Integer Linear Program

In this section we give an ILP to solve the problem. Our goal here is to minimize the number of steps.

3.2.4 Variables

Let us first define some sets and variables.

1. $I = \{1, 2, 3, \ldots, mn\}$. I is the set of cells i.e. I is labeling of cells of the board. This is illustrated in Figure 3.2

Red Receptor	1	2	3	4	
	5	6	7	8	
	9	10	11	12	Blue Receptor

Figure 3.2 Labeling of the tiles

2. $R = \{1,2,\ldots,r\}$ is the set of Red cells
3. $W = \{1,2,\ldots,w\}$ is the set of White cells
4. $B = \{1,2,\ldots,b\}$ is the set of Blue cells
5. T is total time given which is an upper-bound on OPT. Thus we set $T = \{1,2,3,\ldots,4n^2 + nm(n+m)\}$. The upper bound used here, $(4n^2 + nm(n+m))$ is derived in section 3.3
6. $PosRed(r,t)$ is position of Red cell labeled r at time t
7. $PosWhite(w,t)$ is position of White cell labeled w at time t
8. $PosBlue(b,t)$ is position of Blue cell labeled b at time t
9. $UpWhite(w,t) = 1$ if wth white cell moves up at time t. It is 0 otherwise
10. $DownWhite(w,t) = 1$ if wth white cell moves down at time t. It is 0 otherwise
11. $LeftWhite(w,t) = 1$ if wth white cell moves left at time t. It is 0 otherwise
12. $RightWhite(w,t) = 1$ if wth white cell moves right at time t. It is 0 otherwise
13. $x(w,t)$ is the *x-coordinate* of white cell w at time t
14. $y(w,t)$ is the *y-coordinate* of white cell w at time t

15. $Z(x,t) = 1$ if some cell x (which may be red/blue/white) moves right at time t. It is 0 otherwise

3.2.5 Constraints

1. We first set the *Pos* variables according to the initial board configuration: $PosRed(r,1)$, $PosBlue(b,1)$ and $PosWhite(w,1)$ for all red, black and white chips. Also in this step we initialize the x and y coordinated of each white chip.

2. We set constraints relating coordinates and position on board for each white chip w and at each time t:

$$PosWhite(w,t) = x(w,t) + (n \times y(w,t)) + 1 \quad \forall w \in W \;\; \forall t \in T$$

3. Set final configuration:

$$\begin{aligned} 2 \leq PosWhite(w,T) \leq mn-1 \quad \forall w \in W \\ PosRed(r,T) = 1 \quad \forall r \in R \\ PosBlue(b,T) = mn \quad \forall b \in B \end{aligned}$$

   These equations describe the final state of the game we want. We want all white chips to be in the board, of which $|R|$ were initially places at position 1 and $|B|$ were initially placed at position mn. This is enforced by the first constraint. We want all the Red cells to be at cell 1 when game ends and all Blue cells to be at position mn. This is enforced by the next two constraints.

4. Each white cell is allowed only one move:

$$UpWhite(w,t) + DownWhite(w,t) + LeftWhite(w,t) + RIghtWhite(w,t) = Z(w,t)$$

   . This should hold for all white cells $w \in W$ and for all time $t \in T$

5. Only one white cell moves in each step:

$$\sum_{w} Z(w,t) = 1 \quad \forall w \in W \;\; \forall t \in T$$

6. Update the position of all white cells after each step:

$$PosWhite(w,t+1) = PosWhite(w,t) - (n \times UpWhite(w,t)) + (n \times DownWhite(w,t)) - LeftWhite(w,$$

This should hold for all white cells $w \in W$ and for all time $t \in (T \setminus \{4n^2 + nm(n+m)\})$. This ensures proper updates for all white cells at all time steps except the last one.

The above expression tells us that if the white cell moves left or right then we just need to update its position by subtracting or adding 1, respectively. If the cell moves up or down we do just the same thing but take into account the structure of the board, which adds a factor of n.

7. Restrictions on movement of white cells:

   (a) White square at bottom edge:

   $$PosWhite(w,t) + n \times DownWhite(w,t) \leq mn$$

   (b) White square at top edge:

   $$PosWhite(w,t) - n \times UpWhite(w,t) \geq 1$$

   (c) White square at left edge:

   $$x(w,t) - LeftWhite(w,t) \geq 0$$

   (d) White square at right edge:

   $$x(w,t) + RightWhite(w,t) \geq n - 1$$

8. The cell 1 always has $|R| = r$ cells on it and the cell mn always has $|B| = b$ cells on it. Rest all cells have one chip. Thus the following should hold at all time $t \in T$:

   $$\sum_{w \in W} PosWhite(w,t) + \sum_{r \in R} PosRed(r,t) + \sum_{k \in K} PosBlue(b,t) = (r \times 1) + 2 + \cdots + (b \times mn)$$

   This step ensures every cell is assigned a position at all time t and tiles 1 and mn have r and b chips respectively.

9. Next we ensure that a Red or Blue cell moves each time a white cell moves:

   $$\sum_{w \in W} Z(w,t) = \sum_{r \in R} Z(r,t) + \sum_{k \in K} Z(k,t)$$

Above should be true at all time $t \in T$

10. Also we ensure that if white cell does not move then red/blue cell will also not move:

$$PosRed(r,t) - PosRed(r,t+1) + nZ(r,t) \geq 0 \quad \forall r \in R \ \ \forall t \in T$$

$$PosRed(r,t) - PosRed(r,t+1) - nZ(r,t) \leq 0 \quad \forall r \in R \ \ \forall t \in T$$

Similarly:

$$PosBlue(b,t) - PosBlue(b,t+1) + nZ(b,t) \geq 0 \quad \forall k \in K \ \ \forall t \in T$$

$$PosBlue(b,t) - PosBlue(b,t+1) - nZ(b,t) \leq 0 \quad \forall k \in K \ \ \forall t \in T$$

11. We make sure that any blue chip ever goes to cell 1:

$$PosBlue(b,t) \geq 2$$

Also any red chip should never goes to cell n:

$$PosRed(r,t) \leq mn - 1$$

12. Putting bounds on variables:

$$1 \leq PosWhite(w,t) \leq mn \quad \forall w \in W \ \ \forall t \in T$$

$$1 \leq PosWhite(k,t) \leq mn \quad \forall k \in K \ \ \forall t \in T$$

$$1 \leq PosWhite(r,t) \leq mn \quad \forall r \in R \ \ \forall t \in T$$

$$1 \leq x(w,t) \leq n \quad \forall w \in W \ \ \forall t \in T$$

$$1 \leq y(w,t) \leq m \quad \forall w \in W \ \ \forall t \in T$$

3.2.6 Objective Function

We want to minimize the total number of steps thus we minimize the following:

$$\sum_{t \in T} \sum_{w \in W} Z(w,t)$$

3.3 An Approximation Algorithm for O-Game-2

In this section we give a simple approximation algorithm for the problem. The algorithm is based on the idea of clearing first two rows of the given instance. Let us assume that C_0 is a $m \times n$ matrix and assume we are given a white cell at $(1,1)$. Consider one rotation of the white cell following the path $(1,1),(1,n),(2,1),(2,n),(1,n),(1,1)$ as illustrated in Figure 3.3. In the setting shown in the figure, how many steps does it take to clear the first row? In $2n$ steps the white cell is back to position $(1,1)$. In these moves every cell in the the first row moves one position to right and every cell in second row moves one position to left. At any time, if a red cell is adjacent to Red Receptor we swap it out and if any blue cell is adjacent to Blue Receptor, we swap it out. Thus, repeating the cycle $2n$ times removes all the red and blue cells in both first and second rows. This takes $4n^2$ steps.

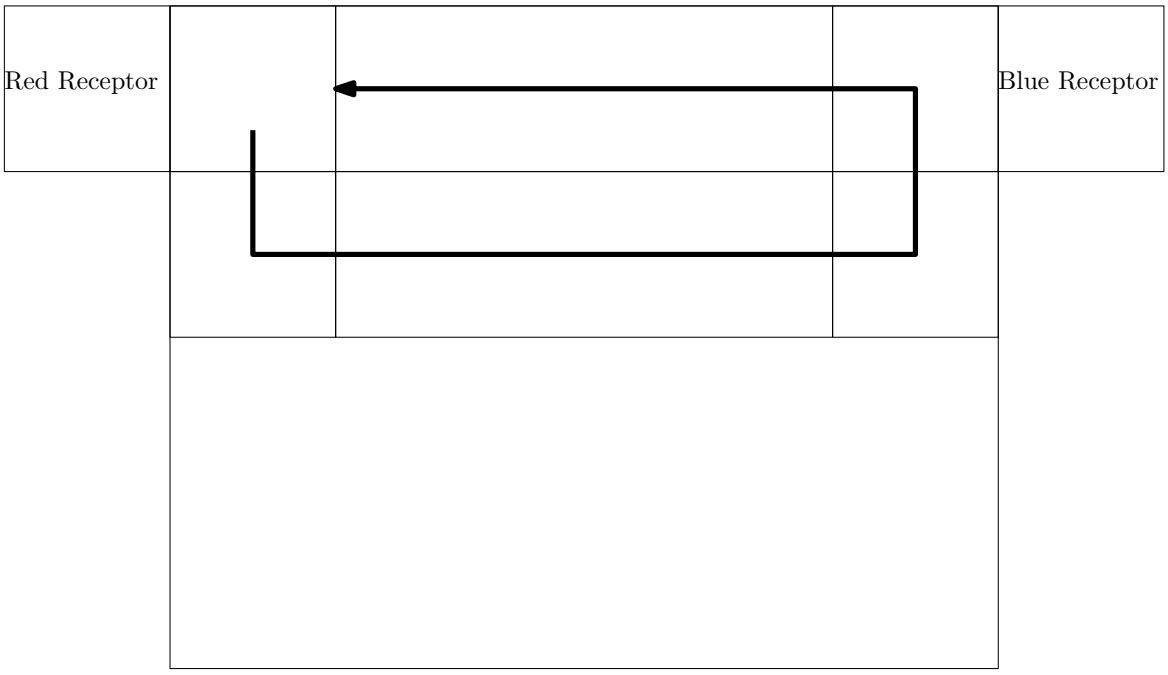

Figure 3.3 Clearing the first two rows

Next we look at lower and upper bound on optimum. The minimum number of steps that a cell must move is the the Manhattan Distance of the cell from the receptor of its color. Thus a trivial lower bound on optimum (OPT) is $\mathbf{O}(nm(n+m))$. In the algorithm described above, it takes $4n^2$ steps to clear the first two rows. After clearing these rows, all other cells can follow one of their Manhattan Paths which is optimum. This will require further $\mathbf{O}(nm(n+m))$. Thus we get the following bounds on OPT, for some constants k_1 and k_2:

$$k_1(nm(n+m)) \leq OPT \leq 4n^2 + k_2(nm(n+m))$$

If we assume C_0 is $n \times n$ board then we get, for some constants k_1' and k_2':

$$k_1' n^3 \leq OPT \leq 4n^2 + k_2' n^3$$

Thus using the algorithm described above we get $\mathbf{O}(1+\frac{1}{n})$ approximation of the optimum. If we consider $n \times m$ board then we get $\mathbf{O}\left(1+\frac{1}{m(1+\frac{m}{n})}\right)$ approximation of optimum.

3.4 Using Game-2 for Biological Cell Sorting

This game is directly related to the problem we want solve as it is related to removing red and blue chips which can correspond to desired and undesired cells. We note that our approximation algorithm performs better when n i.e. number of columns of LoC is smaller than m i.e. number of rows of LoC. This also suggests a design paradigm for LoC.

Bibliography

[1] Arora, S. (1998). Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *Journal of the ACM (JACM)*, 45(5):753–782.

[2] Călinescu, G., Dumitrescu, A., and Pach, J. (2008). Reconfigurations in graphs and grids. *SIAM Journal on Discrete Mathematics*, 22(1):124–138.

[3] Garey, M. R. and Johnson, D. S. (2002). *Computers and intractability*, volume 29. wh freeman New York.

[4] Ghosh, A., Shah, R., Bishnu, A., and Bhattacharya, B. B. (2009). Algorithms for biological cell sorting with a lab-on-a-chip. In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 104–109. IEEE.

[5] Johnson, W. W. and Story, W. E. (1879). Notes on the "15" puzzle. *American Journal of Mathematics*, 2(4):397–404.

[6] Medoro, G., Manaresi, N., Tartagni, M., and Guerrieri, R. (2000). Cmos-only sensors and manipulators for microorganisms. In *Technical Digest - International Electron Devices Meeting*, pages 415 – 418.

[7] Papadimitriou, C. H., Raghavan, P., Sudan, M., and Tamaki, H. (1994). Motion planning on a graph. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 511–520. IEEE.

[8] Ratner, D. and Warmuth, M. (1990). The (n2-1)-puzzle and related relocation problems. *J. Symb. Comput.*, 10(2):111–137.

[9] Robins, G. and Zelikovsky, A. (2008). Minimum steiner tree construction. In *Handbook of Algorithms for Physical Design Automation*.