Thesis no: MSCS-2016-04

Comparison of A*, Euclidean and Manhattan distance using Influence Map in Ms. Pac-Man

Sudip Karki
Hari Sagar Ranjitkar

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona Sweden

This thesis is submitted to the School of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Computer Science. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author(s):
Sudip Karki
E-mail: sukc10@student.bth.se

Hari Sagar Ranjitkar
E-mail: hara10@student.bth.se

University advisor(s):
Professor Lars Lundberg
Associate Prof. Stefan Johansson
Department of Computer Science and Engineering

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

Abstract

Context An influence map and potential fields are used for finding path in domain of Robotics and Gaming in AI. Various distance measures can be used to find influence maps and potential fields. However, these distance measures have not been compared yet.
Objectives In this paper, we have proposed a new algorithm suitable to find an optimal point in parameters space from random parameter spaces. Finally, comparisons are made among three popular distance measures to find the most efficient.
Methodology For our RQ1 and RQ2, we have implemented a mix of qualitative and quantitative approach and for RQ3, we have used quantitative approach.
Results A* distance measure in influence maps is more efficient compared to Euclidean and Manhattan in potential fields.
Conclusions Our proposed algorithm is suitable to find optimal point and explores huge parameter space. A* distance in influence maps is highly efficient compared to Euclidean and Manhattan distance in potentials fields. Euclidean and Manhattan distance performed relatively similar whereas A* distance performed better than them in terms of score in Ms. Pac-Man (See Appendix A).

Keywords: Ms. Pac-Man, algorithm, influence maps, potential fields, distance measure, A*, Euclidean, Manhattan, optimal parameter space.

Acknowledgements

We would like to thank Stefan J. Johansson and Lars Lundberg for their incredible support and guidance to fulfill the task successfully. We want to express our gratitude towards Blekinge Institute of technology for providing all the necessary support and resources so as to complete the thesis fruitfully.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Pac-Man

Pac-Man is a popular arcade game originally produced by Midway and developed by Toru Iwatani for Namco Company in year 1981 [1]. The best variant of the game is Ms. Pac-Man was released later in 1981. It introduced a female character, new maze designs and several gameplay changes[10]. Screenshots of the game are shown in Figure A.1 and Figure A.2 in the appendix: Ms. Pac-Man moves around the maze, eating pills for points while trying to avoid the four ghosts (Blinky,Pinky,Inky and Sue) who strive to eat Ms. Pac-Man[10]. The four power-pills that appear randomly at the corners of the maze allow Ms. Pac-Man to eat the ghosts for a specified time period to gain additional points, during this specified time the ghosts turn blue.

1.2 Potential Field and Influence Maps

Potential field and influence maps are the techniques that address similar kinds of problems using the similar ideas [8]. Both of them are based on attractive and repelling forces.

In potential fields, a destination attracts its surroundings by reflecting a field with a strength that is a function of e.g. the Euclidean distance between the source and the destination. This facilitates the source to choose among a number of lookahead positions to move towards destination and the choice is made in terms of highest potential position[13].

Influence map is such a property in which the adjacent tiles are influenced by object and the adjacent tiles are respectively influenced by the neighboring tiles forming a propagation of influences in the source which is far away [13]. The source which is moving towards the destination then compares the influences among the possible look ahead positions and picks the path having highest influence. This means the chosen path is such that the source is nearest to the

destination and influence is decreasing gradually with the distance.

There are some differences between potential fields and influence maps. The propagation in potential field is not terrain sensitive i.e. the propagation is not blocked when there is a wall between source and destination. Whereas, the propagation of influence maps is terrain sensitive i.e. the propagation can be blocked when there is a wall that hinders the motion resulting to follow the maze path. In this case, the shortest path algorithm (A*) can be used to compute influence since the influence follow the paths of a maze and do not propagate through walls whereas Euclidean and Manhattan distance metrics are used to compute potential field.
Although, potential field methods and influence maps are similar techniques that are used for obstacle avoidance applications. Potential field is rapidly used in robots and mobile robots whereas influence map is used in gaming industry [18].

The generic influence map in GO showing black and white stones influence is represented diagrammatically in Figure 1.1. Figure 1.2 shows the intensity map of all positions in a map in ORTS.

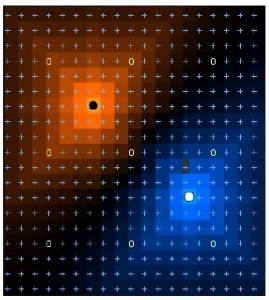


Figure 1.1: A generic influence map in GO showing the black and white stones influence in its surrounding upto 8 tiles further away. Figure with permission from Niklas Hansson [8].

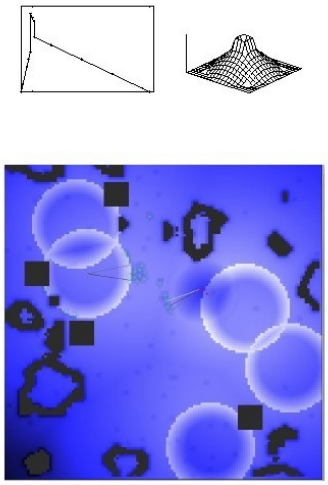


Figure 1.2: The intensity map of all positions in a map in ORTS.Figure with the permission by Hagelback and Johansson [8] .

The figure 1.1 is an example of influence map whereas the figure 1.2 is an example of potential field. Influence map is very important in Go board segmentation [20]. Among the authors many analogies, board domination is the one he is highly interested in. In this board domination, black stone is situated in the

upperleft influences neighboring tiles. Similarly, it influences again its adjacent tiles. In this way, the influence is radiated upto 8 tiles further from the black stone in this particular example. Likewise, white stone that is situated in the bottom right influences the adjacent tiles and this in turn influences its neighbours. The influence of black stone and white stone cancel each other in middle portion of the board and we have a diagonal line there.

It is assumed that a game is in progress and the board position is stored. The position is transferred to a 19X19 integer matrix by placing 50 for each black stone, -50 for each white stone, and 0 elsewhere. Then each point which is positive sends out +1 to each of its four neighbors, and each negative point sends out -1 to each of its neighbors. These numbers are accumulated as this procedure is repeated four times [20].

Other analogies of the author in influence map includes drops of oil and water on a blotted paper and heat conduction. In heat conduction analogy, by altering the thermal conductivity at any point on the plate, we can change the influencing ability.It seems to be a very simple thing when we give a glance but but it is still worthy when we gaze and think of the composite plate consisting of various things added together representing the terrain in the map. For instance, if the certain map section is made up of styrofoam or aluminum, the heat conduction does not take place. So, we need to define about the influence conduction in various terrain type. We can assume that mountainous terrain conduct like styrofoam, plains like aluminum, roads like gold, forest like steel . Jason Kester [4].The heat conduction is analogous to amount of influence which is spread to the neighbors.

The use of influence maps are becoming more popular in gaming industry. An author known as Mat Buckland has referred influence maps for using simple array of data and each element from this data array represents data about the specific world position. Influence maps are regarded as 2D grid overlaid in the world [5]. Influence maps are mostly used in Real time Strategy(RTS) games. In RTS games, influence maps represent areas of various agents. For instance, various players have positive influence over energy sources and negative influence over kill zones. The proper and efficient decision making is possible by querying influence maps by AI techniques [5].

There are some problems with using influence maps. The first is that calculation of influences is expensive. The cost varies according to the types of applications. For instance, some applications are required to update most of the time whereas others require updating only some regions [5].

The advantage of influence maps is that it can be queried for required information regarding the certain position by an AI entity for a particular constant

time. This is greatly important in RTS games in particular since various calculations like finding the safe way out reduces cost to very high extent. The simple and easy visualization of influences is another advantage of influence maps.

In influence maps model various agents in the game maze like normal pills, power pills and ghosts radiate an influence over a certain space and sum of all these influences gives the influence map of that space. So, it is such a property in game world that represents favorable or unfavorable region for motion by calculating the influence maps of that region. If the value is positive, it is favorable to move on in that region otherwise not. In terms of Ms. Pac-Man, the normal pills, power pills, edible ghosts, junctions, teleports emit the positive influences whereas inedible ghosts are the negative influences.

1.3 Euclidean, Manhattan distances and A* search algorithm

Euclidean distance computes the root of square difference between co-ordinates of pair of objects [17]. Mathematically, it can be represented as

$$d_{(x,y)} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Manhattan distance computes the absolute differences between coordinates of pair of objects [17].

Mathematically, it can be represented as

$$d_{(x,y)} = \sum_{i=1}^{n} |(x_i - y_i)|$$

Euclidean and Manhattan distance measures can be graphically represented as follows.

In the above figure the green line represents Euclidean distance whereas red, blue and yellow lines are used to represent Manhattan distances.

A* is a computer algorithm that is widely used in path finding and graph traversal, the process of plotting an efficiently traversable path between points, called nodes. A* uses a best-first search and finds a least-cost path from a given initial node to one goal node (out of one or more possible goals)

A* search algorithm is graphically represented in Fig 1.4.

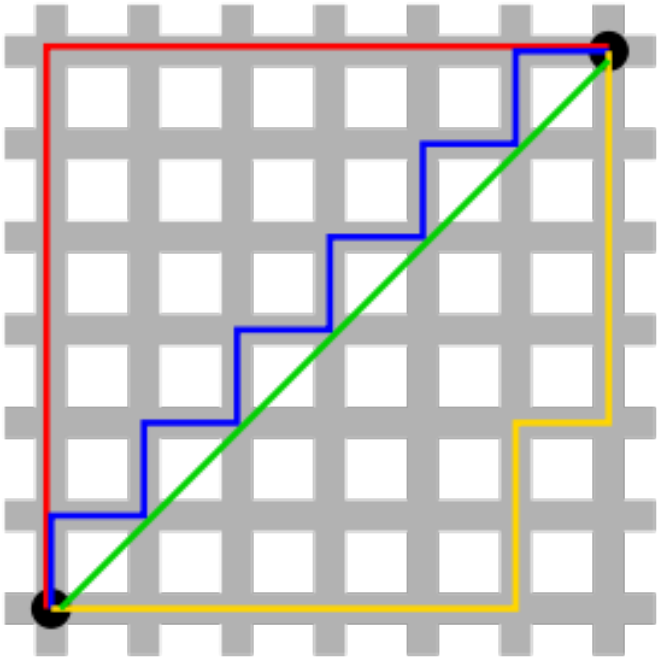


Figure 1.3: Manhattan Distance Representation

1.4 Difference among Euclidean, Manhattan distances and A* search algorithm

Euclidean distance is the shortest path between source and destination which is a straight line as shown in Figure 1.3. but Manhattan distance is sum of all the real distances between source(s) and destination(d) and each distance are always the straight lines as shown in Figure 1.4. So, there can be more than one possible Manhattan distances in a maze and Ms. Pac-Man follows any of the path among them in a maze.

The distance metrics itself can not be considered as terrain sensitive or not. The use of potential field and influence map in the three different distance metrics make them either terrain sensitive or not. Potential field is not terrain sensitive i.e. it does not take terrain of map into account meaning it is unable to recognize the presence of wall between the source and destination. This facilitates the propagation possible through or over the walls. In the other hand, influence map is terrain sensitive i.e. the presence of wall between source and destination affect the propagation of influences [13].

Since we are comparing A* distance metric in influence map and Euclidean and Manhattan in potential field in research question 2, this makes A* terrain sensitive but Euclidean and Manhattan not.

It should be noted that the distance traveled by Ms. Pac-Man from source (s)

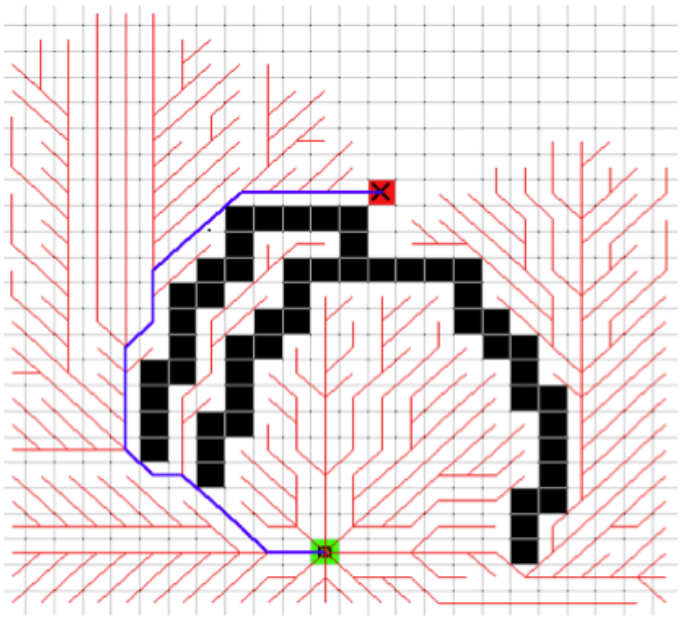


Figure 1.4: A* Search algorithm Representation

to destination (d) may be substantially greater than Euclidean distance between s and d since Euclidean distance calculates the shortest path between s and d [2].

1.5 Related works

Many studies have been carried out regarding potential field in the area of spatial navigation and obstacle avoidance. This method was really helpful to avoid simple obstacles and the ability to avoid complicated obstacles was possible when it was used in combination with autonomous navigation approach.

Artificial potential field was introduced by Ossama Khatib in 1985 while he was looking for real time obstacle avoidance approach for manipulators and mobile robots [6]. Potential field in multi-agent system is providing good results. Howard et. al developed the mobile sensor like robot soccer using potential field.

Computational Intelligence and Games (CIG) conference is a competition that is held yearly since 2008. It has come up with many fruitful and increased level research activities with various solutions in Ms. Pac-Man controller [13].

The early attempts to construct Pac-Man controllers include the work of Koza, Gallagher and Ryan. Koza used genetic programming based approach as a remedy to static ghost movement since Pac-Man shows the deterministic behaviour

of the movement of the ghosts. Gallagher and Ryan applied the weighted set of rules which is dependent on the results of the previous set of games.

Robles and Lucas applied the tree-based search and got a very good result in terms of score by reaching a depth of forty moves ahead in the search. This work was further carried by Samothrakis et al. by using Monte Carlo tree search with certain tree depth allowing to find the paths better than the earlier one.

In 2008, Wirth and Gallagher presented a solution stating the use of influence maps. Their model comprised of three main parameters that have intuitional relation with the behaviour of agents. They have used the greedy algorithm, random, systematic and global exploration method to show the experimental results that explores the model performance over its parameter space [18]. The model of Wirth and Gallagher stems to three dimensional optimization problem. They used greedy algorithm and it took steps in random directions in the parameter space. The step size was slowly deducted in a certain period of search. They have used the Euclidean distance measure in their proposed equation. They constructed the influence map model only by taking accounts of dots, ghosts and edible ghosts but not the fruits and other attributes of maze like junctions and lairs.

Despite Wirth and Gallagher states the use of Influence map in their paper, Svensson in "Influence-map based controllers for Ms. Pac-Man and the Ghosts" claims the use of potential field in this paper. Johan Svensson and Stefan J. Johansson have implemented influence map based controller both for the Ms. Pac-Man and the ghosts. They have used A* distance measure algorithm for calculating the influence [13].

Svensson claimed that Wirth Gallagher has not implemented influence map rather it is a potential field. Moreover, they said that their implementation is influence map. Although, there is slight difference between potential field and influence map but both of them can be represented by same equation. The major differences between Wirth Gallagher and Svensson are they have used different pacman controller , different distance measures i.e. Wirth Gallagher has used Euclidean but Svensson has used A*. They have used different algorithm to find optimal space. These distances are not compared in the same environment with same controller and with the same algorithm. Despite of having less differences the same equation is used for both potential field and influence map. Moreover, influence map is more commonly used in gaming field. In rest of our thesis , we will be representing both potential field and influence map as influence map. Also, we will have single equation which will derive potential field and influence map based on the distance measure used.

Chapter 2

Problem Description and Statement

2.1 Problem and Research gap

Wirth and Gallagher have only used Euclidean distance whereas Johan Svensson and Stefan J. Johansson have only used A* distance measure . As discussed in Section 1.4, the controllers used in their experiments are different and the algorithm used to find optimal space are also different.

The three distance measures Euclidean, Manhattan and A* have not been compared

1. Together.
2. With the same controller.
3. With the same algorithm.

We will be using Ms. Pac-Man vs Ghost controllers which provides interface to calculate euclidean, manhattan and A* distance for given two points.

Ms. Pac-Man Maze has a grid structure i.e. the travel path used by ghost and pacman is either a straight line, right angle or numerous combination of right angle. It is relevant to compare Euclidean distance and Manhattan distance.

Moreover, Ms. Pac-Man maze has numerous routes to the destination. As A* find the least cost path between numerous routes therefore it is worth to compare three way comparison between Euclidean, Manhattan and A*. A* is costly in terms of calculation therefore it may not be suitable in real time scenario however Ms. Pac-Man vs Ghost controller provides pre calculated A* distance measure which loads during the start of the game.

The three way comparison is important because distance measure is not only the factor to improve the highest score, in addition to distance measure various strategy can be applied in Ms. Pac man game which will result more score. i.e.

one can implement euclidean distance measure with good gaming strategy and yield more score than A* implementation.

2.2 Aims and objectives

The main aim of this paper is to find the efficient distance measure algorithm in Ms. Pac-Man for influence map. To achieve the aim of the paper following objectives have to be fulfilled.

2.3 Research Questions

1. what is the increment of score from random parameter space in Ms. Pac-Man to optimal parameter space in Ms. Pac-Man by the algorithm ?
2. Is the algorithm able to perform better than random optimal parameter space ?
3. Which distance measure among A*, Euclidean and Manhattan will provide highest score when run in same algorithm, controller and environment ?

2.4 Research Methodology

2.4.1 Literature Review

Table 2.1 Search results of literature review

Keywords	IEEE	Inspec	ACM
pacman	43	114	22
Ms. Pac-Man	50	53	54
Ms. Pac-Man and Artificial intelligence	32	31	7
Ms. Pac-Man and Potential field	2	3	0
Ms. Pac-Man and Potential field	2	1	0
Ms. Pac-Man and potential field and influence map	1	2	0

For the literature review, studies related to Ms. Pac-Man, artificial intelligence and finally path finding solutions like potential field and influence maps were considered.

We performed search in six different stages where we have used different sets of keywords each time. The choice of keyword were initially chosen as Pacman and this yielded many research papers. Then, the search is narrowed down by

focusing on the advanced version of Pacman i.e. Ms. Pac-Man. After this, search was performed gradually by focusing on one of the modern approaches known as artificial intelligence. Similarly we targeted our search to potential field and influence maps since these two terminologies are the focus areas in our research. The table shows the results and the list of keywords used for searching the different databases. The databases used were IEEE, Inspec and ACM.

By the literature review that we have conducted 271 Journals and Conference papers were obtained. These results were filtered by an exclusion criterion to contain only full text. This resulted to only 157 results. After this, this result was refined further so that the articles are after 1999 A.D. or last 15 years which reduced the result to 70 results. The titles and abstract of all these 70 papers were reviewed and only 15 papers were selected that seem to be relevant for our thesis. Most of the papers were excluded because they focused on decision tree search algorithm. Instead the papers that dealt with hill climbing algorithm, potential field and influence maps were chosen from a lot of papers. Similarly, Papers implementing neural networks, genetic programming are not selected since artificial intelligence approach is selected for calculating potential fields and influence maps in our thesis. The reference papers of the selected papers were also taken into consideration while selecting the papers.

Inclusion and exclusion criteria

Inclusion criteria

1. Papers discussing about artificial intelligence, potential fields and influence maps are taken into consideration.
2. The latest 15 years papers (Journals and Conference papers only) starting from 1999 A.D. were taken into consideration.

Exclusion criteria

1. Papers that were published in language other than English are excluded.
2. To maintain the quality standard the papers other than Journals and Conference papers were excluded.
3. The papers lacking full text while searching in database are excluded. This means those papers which needs to be purchased are excluded.

2.4.2 Methodology

For our RQ1 and RQ2, we will implement a mix of qualitative and quantitative approach. Firstly, we conduct our review through various types of literatures such

as books, magazines, articles etc. Among different online databases, we choose ACM, IEEE explorer and Engineering village for searching literatures. This will help us to find how distance measures are used in Ms. Pac-Man using AI.

In the second step, we gather information collected in the RQ1 to design, construct and synthesize qualitative solutions for implementations and conduct experiment.

For RQ3, we will use quantitative approach. The design in influence model will be used with various distance metrics algorithm and will be implemented in Ms. Pac-Man to play with the controller. The result generated from the game play will be used to analyze and show performance. The performance is shown by pointing how efficient is distance measure using mean value of score from the game.

2.4.3 Outcomes

The possible outcome we expect to achieve is a complete game that can be played with the controller. Graphs show the performance of the algorithms used in Ms. Pac-Man. The outcome results in list of individual graphs for each distance measures, distribution of weights used, ghosts, edible ghosts, power pills and normal pills, frequency graphs and density graphs. Finally, tabular results include numeric values like scores, optimal points that can be gathered from games. These collected values will be used for statistical analysis to analyze the results.

Chapter 3

Ms. Pac-Man Implementation

3.1 Influence Map description

In high level perspective, influences generated by object in influence map propagates in map decreasing its influence in each iteration of the tiles in the map. The influence propagates in this fashion and fades away in distance based on the weight of the influences the object radiates. The influence map of certain region in a maze is a sum of all the local influences. For Ms. Pac-Man, normal pills, power pills, randomly appearing fruits and edible ghosts radiate positive influences whereas inedible ghosts emit negative influences.

A calculation for this type of radiation from many objects is resource intensive where the calculation should be made in every tile. In our design, we use a simple calculation to make it effective where we do not care all influences made along a pathway to origin of the object. Rather, we take the influence in fading point or instead of influences completely fading away. We calculate the influence of objects until it reaches near the adjacent tiles of Ms. Pac-Man Lookahead position.

Lookahead position is adjacent tiles of Ms. Pac-Man. Moreover, one stop Lookahead position is the adjacent tiles with current position of Ms. Pac-Man. Ms. Pac-Man has at most four one stop lookahead position. These positions can also be referred as the directions for Ms. Pac-Man to move, ie. UP, DOWN, LEFT, RIGHT.

3.2 Generic Influence Map Function

The generic influence map function we use is based on the following definition of Influence maps.

$$I_{foO}(O, w_o, lap_i) = \max_{lap_i \in LAP} \sum_{k=1}^{n} \frac{w_o}{d_m(lap_i, O_k)}$$

where O is the object that radiates positive or negative influence. O can be

power pills, normal pills, ghost , edible ghost , lairs. w_o is the strength of the influence of the object O, lap_i is the lookahead position of adjacent neighbour of Ms. Pac-Man where $i >= 1$ and $i <= 4$. d_m is the distance measure between the lap_i and O where m is the various distance measure through:

$$m \in M(Eu, M, A*)$$

Eu, M and A* are Euclidean, Manhattan and A-Star distance measures respectively. M is the collection of Distance measures where m is one of the members of the Objects.

3.3 Categorizing Influence

To calculate the Influences of Ms. Pac-Man, we divide the Influences into 2 distinct categories based on their properties.

1. Positive Influence : Those objects which attract the agent. These objects radiate a positive value in the maze.
2. Negative Influence : Those objects which repel the agent. These objects radiate a negative value.

3.4 Influence of Ms. Pac-Man

From Ms. Pac-Man perspective, all the objects in a game can be placed into 2 categories. They are:

1. Positive Influence : Normal Pills, Power Pills, Junctions, Edible Ghosts
2. Negative Influence : Ghosts

We further divide these categorised objects into fields where we discuss how the individual objects has its influences.The actual Influence map functions used from the generic functions are:

3.4.1 The Fields of Power Pills

The field of power pill is positive when all four inedible ghosts are near to Ms. Pac-Man with distance d. If the inedible ghosts are far away from Ms. Pac-Man, the power pill radiates negative value. Moreover,when ghosts are at edible state, the field of power pill is negative. So, Ms. Pac-Man will be repelled towards the power pill and starts to chase the ghosts. Ms. Pac-Man is awarded 50 points for

eating a power pill. The field of power pill can be represented by

$$I_{foP}(P, w_P, lap_i) = \max_{lap_i \in LAP} \sum_{k=1}^{4} \frac{w_P}{d_m(lap_i, P_k)}$$

where $I_{foP}(P, w_P, lap_i)$ is field of power pills, P is power pill where the maximum number of power pill is four and w_P is weight of power pill.

3.4.2 The Fields of Normal Pills

The field of normal pill is positive when pills are remained to be eaten and inedible ghosts are far away from Ms. Pac-Man. As soon as the game starts total number of pills is 250 and eating each pill gives four points. The condition of local optima occurs when the positive and negative fields are equal in certain position of the maze. In such condition, she can not move in either directions. Moreover, at the end of game, when there are fewer number of pills, a small weight is not enough to attract Ms. Pac-Man in long distance. Therefore, in the beginning of the game, weight of the normal pills should be less. As the number of pills decreases, the weight of the normal pills should increase.Here, Np in following equation is used to prevent these conditions.

The fields of normal pills can be represented by

$$I_{foNp}(Np, w_{Np}, lap_i) = \max_{lap_i \in LAP} \sum_{k=1}^{250} \frac{w_{Np}}{d_m(lap_i, Np_k)}$$

where $I_{foNp}(Np, w_{Np}, lap_i)$ is the field of normal pills, Np is number of normal pills and its value ranges from 1 to 250, w_{Np} is the weight of normal pills and dm is the distance between the look ahead position and the normal pill.

3.4.3 The Fields of Edible Ghosts

When Ms. Pac-Man eats the power pills, field of edible ghosts come into existence. This field emits the positive value. Eating a first edible ghost gives Ms. Pac-Man value 200 and it successively doubles if other remaining three ghosts are also eaten by her. So, the total points she is awarded by eating all four ghosts is 200+400+800+1600=3000. If Ms. Pac-Man eats another power pill when the ghost are in edible state the value is reset to 200 again.The edible ghosts starts to blink after certain time after eating power pills and it emits negative influence to Ms. Pac-Man. The period of transforming from edible state of ghost to inedible state decreases gradually in higher levels of game. The field of edible ghosts can be represented by

$$I_{foEg}(Eg, w_{Eg}, lap_i) = \max_{lap_i \in LAP} \sum_{k=1}^{4} \frac{w_{Eg}}{d_m(lap_i, Eg_k)}$$

where $I_{foEg}(Eg, w_{Eg}, lap_i)$ is field of edible ghost, Eg is the edible ghost where maximum number of edible ghost is four and w_{Eg} is the weight of edible ghost.

3.4.4 The Fields of Junctions

The junctions in a maze emit a little positive influence.The junctions are used to escape from ghosts and they are very good escaping points for Ms. Pac-Man if two ghosts are coming from opposite directions. The ghosts travel in slightly lower speed in the junctions as compared to the other positions of maze. The number of junctions is different in different levels of game. When Ms. Pac-Man is trapped by two or more ghosts, the nearest junction between Ms. Pac-Man and ghosts is calculated. The field of junction can be represented by

$$I_{foJ}(J, w_J, lap_i) = \max_{lap_i \in LAP} \sum_{k=1}^{n} \frac{w_J}{d_m(lap_i, J_k)}$$

where $I_{foJ}(J, w_J, lap_i)$ is the field of junction, n is the number of junctions and its value ranges from 1 to n and n is the highest number of junctions of the different mazes, w_J is the weight of the junction which has fixed value of two.

3.4.5 The Fields of Ghosts

The ghosts emit the negative influence to Ms. Pac-Man. The field of ghost is very important because the ghosts are the ones that cause Ms. Pac-Man to die. The field of ghost can be represented by

$$I_{foG}(G, w_G, lap_i) = \max_{lap_i \in LAP} \sum_{k=1}^{4} \frac{w_G}{d_m(lap_i, G_k)}$$

where $I_{foG}(G, w_G, lap_i)$ is the field of ghost, w_G is the weight of ghost, d_m is the distance between the ghost and Ms. Pac-Man in a maze.

3.4.6 The Total Influence

The total calculation of all Influences made in the lookahead positions is sum of all the Influences generated in the fields.

$I_m(lap) = I_m(Np, 250, W_{Np}, lap) + I_m(Eg, 4, W_{Eg}, lap) + I_m(G, 4, W_G, lap) + I_m(P, 4, W_P, lap) + I_m(J, n, W_J, lap)$
The $I_m(lap)$ is calculated for all the look ahead positions. Finally, among all

the $I_m(lap)$ the one with the highest value will be chosen as a direction for Ms. Pac-Man.

3.5 Optimal Parameter Space

The algorithm is relatively close to hill climbing algorithm, which is designed to find the optimal space for five parameters. These five parameters are Power pills, Normal pills, Ghosts, Edible ghosts and Junctions.

The algorithm starts by initializing random values for all five parameters. The range for random values for Ghosts, Edible ghosts and Power pills are between 1 and 400. The range of random values of Normal pills are between 1 to 40. A random value of Normal pills is not more than 40 because high value for normal pills leads to local minimum problem. The value of Junction is fixed and set to 2.

The proposed algorithm starts with one of the random values by decrementing the value as long as the score is improved. Once the score is no longer improved the algorithm starts incrementing the random value as long as the score is improved. when the value is both incremented and decremented the algorithm makes selection between incremented and decremented value based on the highest score.

This process is repeated with rest of the random values and selection is made. The final result will be selection of five optimized values that yielded the highest score. This whole process is referred as single iteration. The algorithm then moves to second iteration but instead of random values optimized values of previous iteration are used.

The algorithm completes each iteration when all the five values are both incremented and decremented. The first iteration starts with random values whereas the consecutive iteration starts with optimized values from previous iteration. The iteration stops when the score can be no longer be improved. The result will be final optimized values that yielded the highest score among all the iterations.

The pseudo code of algorithm is as follows.

```
INITIALIZE decrementWeight[5] WITH 4 random number AND 1 constant number
SET incrementWeight[5] TO decrementWeight[5]
currentIterationAvgScore

SET decrementScore TO ZERO
SET incrementScore TO ZERO
SET highestAverageScore TO ZERO
SET currentIterationAvgScore TO ZERO

DO
        IF highestAverageScore <= currentIterationAvgScore THEN
                SET highestAverageScore  to currentIterationAvgScore
                SET decrementWeight[5] TO decrementWeight[5]
        ENDIF

        SET averageScore TO ZERO
        SET Counter TO ZERO
        DO
                DO
                        IF currentIterationAvgScore <= averageScore THEN
                                SET currentIterationAvgScore to averageScore
                        ENDIF

                        IF Counter MOD 2 == 0 THEN
                                SUBTRACT 1 FROM decrementWeight[Counter/2]
                                SET decrementScore to currentIterationAvgScore
                        ELSE
                                ADD 1 TO incrementWeight[Counter/2]
                                SET incrementScore to currentIterationAvgScore
                        ENDIF

                        FOR i = 0 to 9
                                ADD RETURN gameScore() to Score
                        ENDFOR

                        SET averageScore to Score/10;
                        WRITE decrementWeight[5],incrementWeight[5],averageScore
                WHILE currentIterationAvgScore <= averageScore

                INCREMENT Counter

                IF Counter MOD 2 == 0 AND Counter > 0 THEN
                        IF decrementScore > incrementScore THEN
                                SET decrementWeight[(Counter/2)-1] to (decrementWeight[(Counter/2)-1]) + 1;
                                SET incrementWeight[(Counter/2)-1] to decrementWeight[(Counter/2)-1];

                        ELSE
                                SET incrementWeight[(Counter/2)-1] to (incrementWeight[(Counter/2)-1]) - 1 ;
                                SET decrementWeight[(Counter/2)-1] to incrementWeight[(Counter/2)-1] ;
                        ENDIF
                ENDIF

        WHILE Counter < 10
        INCREMENT iteration
WHILE highestAverageScore < currentIterationAvgScore
```

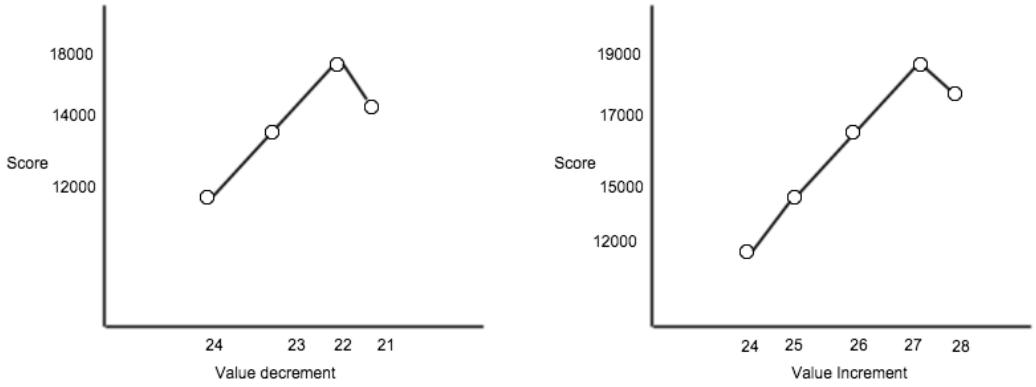


Figure 3.1: Decrement & Increment of single value

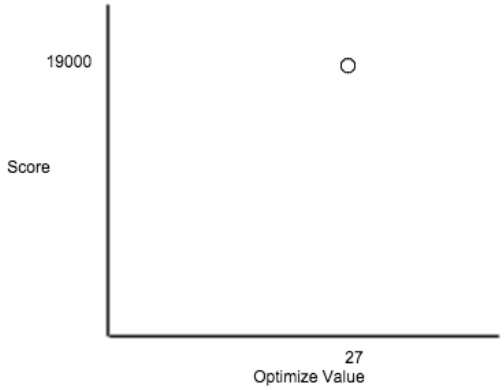


Figure 3.2: Selection of value based on score

The Figure 3.1 is an example which shows decrementing and incrementing of single value. A random value 24 is decremented to 23 and 22 as long as the score is increased i.e. highest score 18000. When the score is no more improved the value 24 is incremented to 25, 26 and 27 as long as the score is increased i.e. highest score 19000.

The figure 3.2 is an example which shows selection of optimal value based on the highest score. In the example, 27 is selected as an optimal value based on the highest score. This process is repeated for all the set of values.

2000 random set of weights will be processed through the algorithm which will yield 2000 final optimized values.

Example is provided in appendix where single set of random weights is processed by algorithm through consecutive iterations and the end result is a set of final optimized values.

A running Java source code is provided in appendix.

Chapter 4

Experiments

4.1 Goal of the experiment

The goal of experiment is to compare various distance measures and study behavior when different distance measure algorithms are implemented with influence map in Ms. Pac-Man. To verify whether the algorithm applied to find the optimal weight is effective or not and which distance measure algorithm is efficient.

4.2 Motivation

Quantitative approach is known for fetching quantitative data from controlled experiments which promotes comparison and statistical analysis [19].

Previously, the distance measures were used independently under different scenarios and environment within influence map. We don't have a comparison of various distance measures under similar experiment. Our implementation will be used to compare and study the behavior, performance and impact of these distance measure algorithm in Ms. Pac-Man maze. To carry out a comparison using statistical analysis among the distance measures, we used quantitative analysis to show our final comparisons.

4.3 Experiment Planning

1. Context Selection: We selected random values for five different parameters such as normal pills, power pills, ghosts, edible ghosts and junctions and optimized the parameter space by our algorithm. To represent huge population we choose 2,000 random points for the experiment [19].

2. Hypothesis Formulation

   To be sure our proposed algorithm works better than random parameters used to find optimal value we have constructed hypothesis
   "Random parameters is efficient in finding optimal values than proposed algorithm to find optimal values"

   (a) Hypothesis I
   The hypothesis test is related to comparison of score obtained from optimized algorithms with score obtained from random parameter space in Ms. Pac-Man. Our sample size is equal whereas the variance is assumed to be unequal. Therefore, we choose Welch's t-test which is intended for use with two sample of equal or unequal size with possibly unequal variances.

   Null Hypothesis Ho:
   $\mu_Random \geq \mu_Optimized$

   Alternate Hypothesis Ha:
   $\mu_Random < \mu_Optimized$

   where the t value can be calculated as

$$t = \frac{\overline{X_R} - \overline{X_O}}{\sqrt{\frac{S_R^2}{N_R} + \frac{S_O^2}{N_O}}}$$

$$\text{Degrees of freedom } (f) = \frac{(\frac{S_R^2}{N_O} + \frac{S_O^2}{N_O})^2}{\frac{S_R^4}{N_R^2(N_R-1)} + \frac{S_O^4}{N_O^2(N_O-1)}}$$

   where X is mean, S is standard deviation and N is number of samples in the experiment.

   The decision outcomes of hypothesis testing
   Type-I-Error: Ho will be correctly accepted with a significance (or confidence) level (1-α) and falsely rejected with a type I error probability α.

   Type-II-Error: If the null hypothesis is false, it will be correctly rejected with a power of the test (1-β) and will be falsely accepted with a type II error probability β.

3. Variable Selection: we selected different dependent and independent variables that are needed to carry out experiments.

Dependent variables: Values for normal pills, power pills, ghosts, edible ghosts.

Independent variable: Junction value which is fixed for iteration.

4. Subject Selection: we used random sampling technique for subject selection since we choose values for the five different parameters from programming functions. We used this sampling technique since only the random values for various parameters will search huge parameter space to find optimal point in space. The generalization error is supposed to be very less because we check for other possible better optimal space nearby the chosen random value to check if there are more optimal space. 2000 sample points and 4 parameters ensure that sample is representative of whole population.

4.4 Experiment Instrumentation

1. Object: Object needed for conducting experiment is a computer with Java installed to run the software. Ms. Pac-Man controllers apply AI logic on top of it. We use 8 core processor computer to conduct the experiment so that different distance measures can be run in parallel.

2. Guidelines: As the controller plays the game itself, we only need to verify and check if the software gets aborted abnormally. Data is collected and appended in every run.

3. Measurement Instruments: The data collected is post processed for measurements. Scripted tools in conjunction with R programming is used for post measurements of the collected data.

4. Data Collection: The implementation code of Ms. Pac-Man is written to collect the numeric variables (Score, Level, Number of Normal pills eaten, Number of Ghosts Eaten) provided by Ms. Pac-man controller interface. Furthermore, few values like the position where Ms. Pac-Man dies are added in core level of the program to record the event. Data are gathered in multiple files with column wise format for further interpretation.

5. Experimental Environment: Similar experiments are executed in parallel to minimize the whole execution time. Second experiment is conducted with interval to analyze the first experiment results.

   The experiments are conducted using Ms. Pac-Man game controller. Ms. Pac-Man game controller is written purely in Java for the "Ms Pac-Man vs Ghosts Competition". Ms. Pac-Man controller allows to develop AI controllers on top. This controller also provides read facility of most of the state of the game. Moreover, distance measures like A*, Euclidean and

Manhattan are already implemented in the controller.

The influence map calculation and Algorithm discussed in implementation phases are used to develop the AI controller and tested further on.

The experiment is run against starter ghost program, which is default ghost program provided. The starter ghost is one of the aggressive ghost program. The major behaviors of these ghosts are: -if edible or Ms. Pac-Man is close to power pill, run away from Ms. Pac-Man. -If non-edible, attack Ms. Pac-Man with certain probability, - else choose random direction.

Based on the implementation, a series of experiments are performed. The experiment is conducted in two phases i.e phase 1 and phase 2. Phase 1 of the experiment is to find the optimized parameter of objects. This helps to further move towards the phase 2 of the experiment. The phase 2 of the experiment is processed using the results from the experiment conducted in phase 1.

4.4.1 Experiment Phase 1

We have 5 different objects as parameters, which radiates influences. The objects are Ghosts, Edible Ghosts, Power pills, Normal pills and Junctions. These objects radiate positive and negative influence based on the description in Section 3.4 of Implementation. Our first task is to find the optimized weight of these objects. The experiment is conducted to find the optimized value of parameters that yields maximum score. This experiment is performed for all distance measures because weight will vary depending on the type of distance measure used. Thus three individual experiments are performed.

The Algorithm described in Section 3.5 is implemented in this experiment to find optimal parameter space. The random values between 1 to 400 are chosen for objects i.e. Power pill, Ghosts and Edible Ghosts whereas random values between 1 to 40 is chosen for Normal Pills and Junctions has fixed value of 2.

Each score is obtained from average of 10 games. Number of Iterations are run for every random weights. 2,000 optimized weights are collected for each distance measures. The log parameters recorded during this experiments are the weights of the objects and the scores earned using the parameters.

4.4.2 Experiment Phase 2

The optimal parameter space from the result of the experiments in phase 1 is used to conduct the phase 2 of the experiments. In phase 2 of the experiment, optimal parameters are set for the objects and a series of games are run to collect

data for analysis.

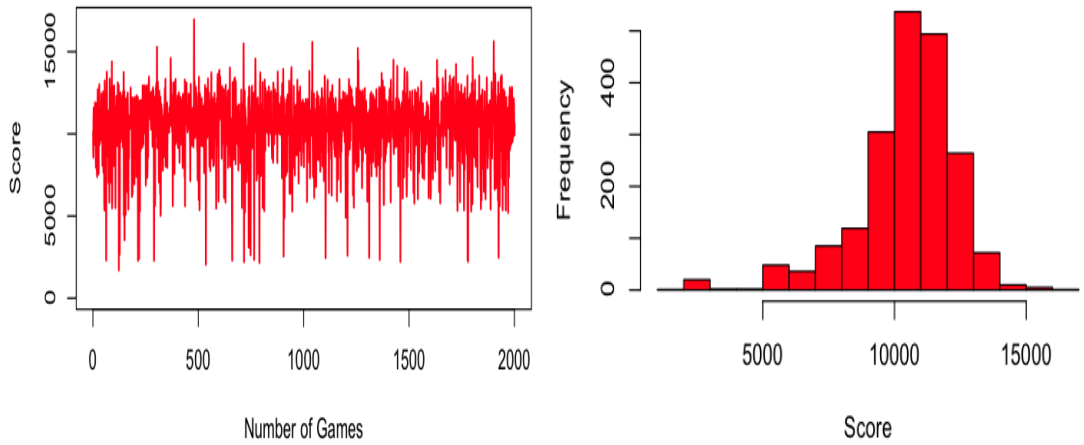
Altogether three different experiments are conducted for A*, Euclidean and Manhattan distance measure. 10,000 game trials are run for each individual experiment.

The log parameters recorded during this experiments are the score, number of normal pills eaten, number of ghosts eaten, number of power pills eaten. These recorded parameters belong to each individual game.

Chapter 5

Result and Analysis

5.1 Result of Experiment 1



(a) Unsorted high score

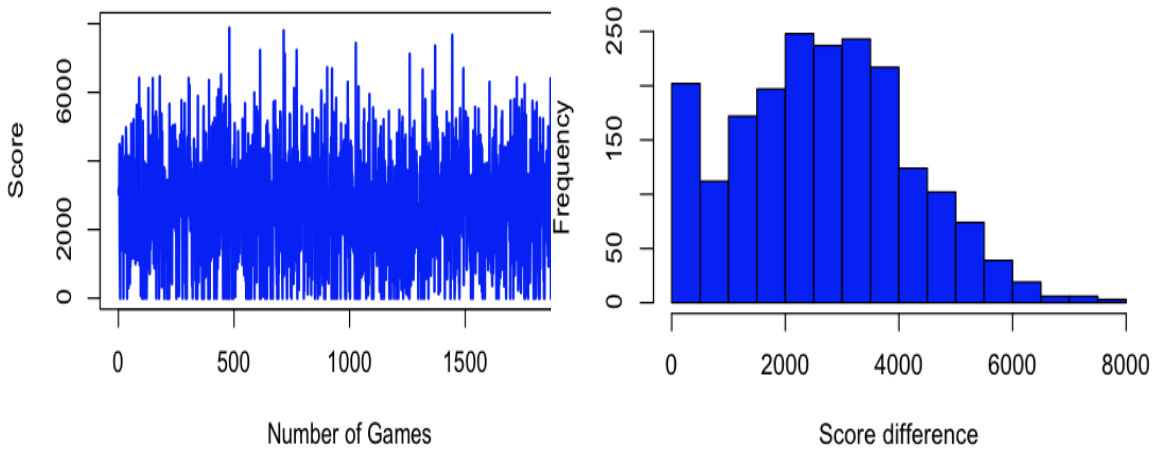
(b) Highest score frequency

Figure 5.1: Results of the highest scores using A*

The Figure 5.1 (a) and 5.1 (b) are obtained using A* distance measure. The Figure 5.1 (a) shows the unsorted highest score obtained when implementing algorithm for every random weights in search space. X axis is number of games whereas y axis is final high score obtain after applying algorithm. The maximum score in the graph is 16,934 whereas the minimum score obtained is 2071. The Figure 5.1 (b) shows highest average score frequency, the frequency score is grouped with 1000 interval.

The suitability of the algorithm is measured in four factors. They are score optimization, number of iterations used, distribution of weights used and comparison with equal sample using random set of values. These four factors are described in detail below.

5.1.1 Score optimization



(a) Score optimization ***(b)*** Frequency of score optimized

Figure 5.2: Optimization of scores using algorithm in A*

The Figure 5.2 (a) shows optimization of scores when algorithm is applied. Optimization of score is difference of first score obtained from set of random weights and final highest score obtained using the proposed algorithm. The collected optimized score consist result from both increment and decrement weight. The algorithm is able to optimize score maximum upto 8000.

Figure 5.2 (b) shows frequency of optimized score, the frequency is grouped with 500 interval. The average score the algorithm is able to optimize is between 2000 to 4000. The score 0 means algorithm is not able to optimize initial score obtain from random weight. The number of inital score that the algorithm is not able to increase is 129 out of 2000 random set of weights, in percentage number of 0 obtained is 6.45%.

5.1.2 Iteration used

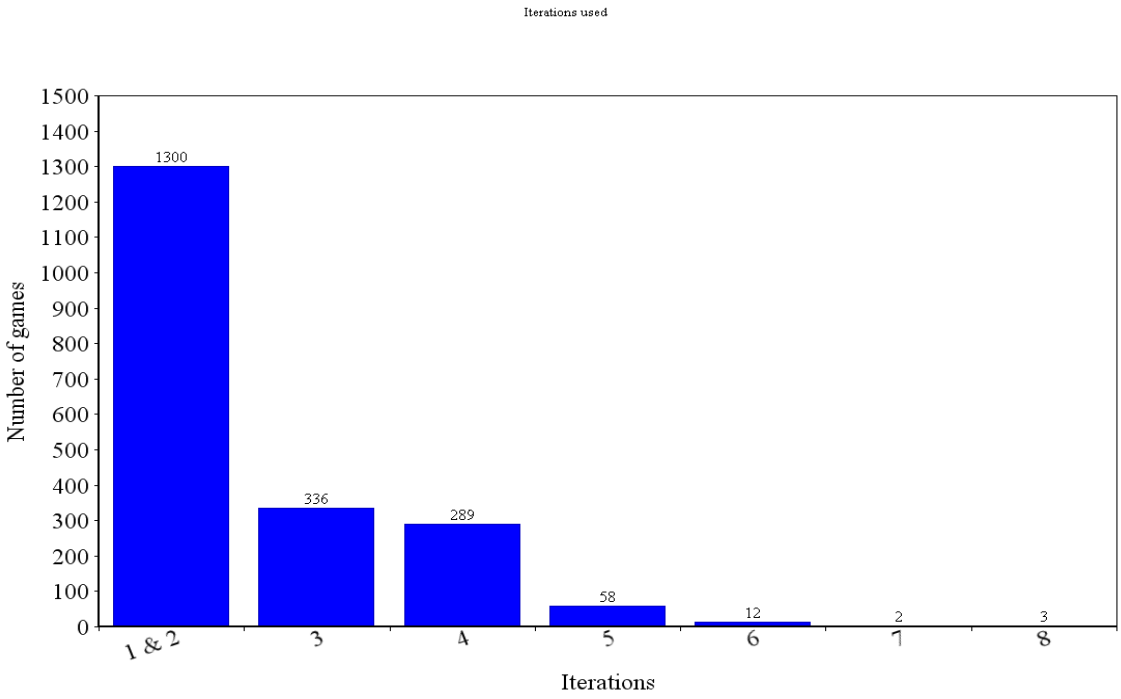


Figure 5.3: Frequency of set of weights in iterations using A*

The Figure 5.3 shows the number of sets of values used in every iterations.Chapter 3.5 describes the iteration in detail. The algorithm starts with random sets of values and will move to further iterations if there is possibility to optimize more. The algorithm will always go to iteration 2 from iteration 1.

The histogram shows 1300 random set of values are able to reach iteration 2nd but were unable to move to iteration 3 (The iteration 1 will always proceed to iteration 2). 336 random sets of values are able to reach iteration 3, 289 random sets of values are able to reach iteration 4, 58 random sets of values are able to reach iteration 5, 12 random sets of values are able to reach iteration 6, 2 random sets of values are able to reach iteration 7 and 3 random sets of values are able to reach iteration 3.

In every iteration the set of values which are able to score better than previous iteration are used in next iteration. Figure 5.3 shows 35% of random set of values are able to pass through iteration 2. The maximum number of iteration the algorithm has reached is 8.

5.1.3 Set of weights from highest score after applying algorithm

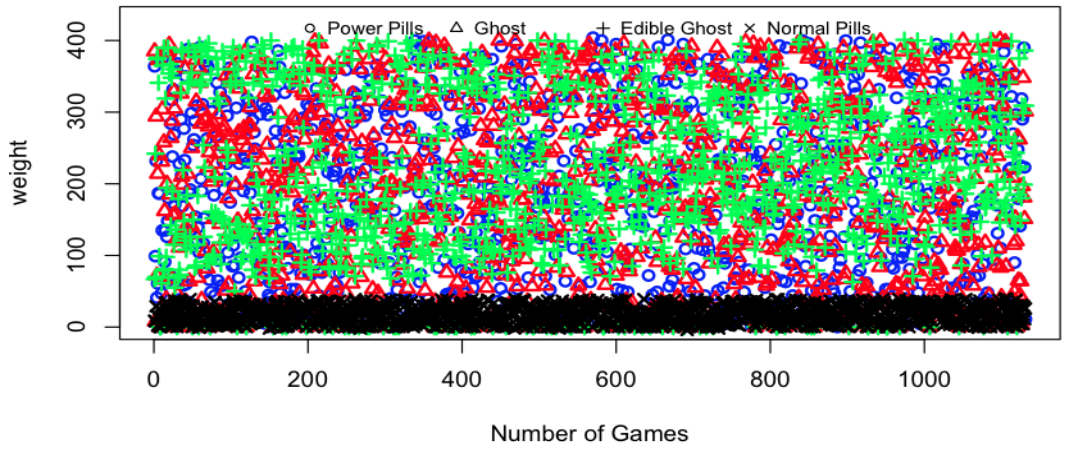


Figure 5.4: Set of weights between high score 10,000 and 12,000 using A*

From Figure 5.1 (b) it is clear that high frequency of score after implementation of algorithm is between 10,000 and 12,000. Therefore, the weights that yielded score between 10,000 and 12,000 are used to plot Figure 5.5 for finding the distribution of weights. The Figure 5.4 shows the weights are randomly distributed within the range of 0 to 400. This shows that the algorithm is able to explore randomly in all the parameter space.

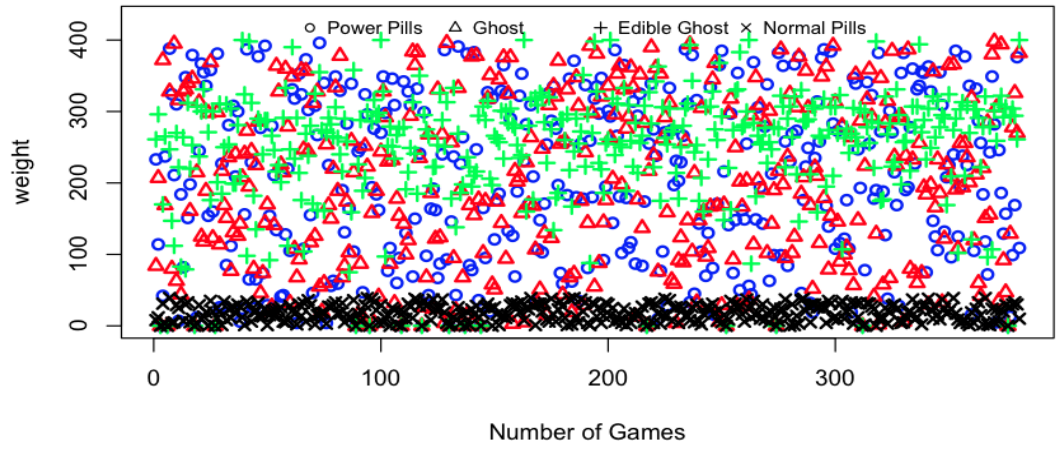


Figure 5.5: Set of weights greater than high score 12000 using A*

From Figure 5.5 the weights are found randomly distributed from which it is unable to find optimal space in clustered area. The Figure 5.5 is plotted with weights that yielded score greater than 12,000. The result was similar to above

distribution where the weights were randomly distributed.

5.1.4 Comparison with equal sample using random set of weights

The proposed algorithm is compared without optimizing weights and running equal number of games using only random set of weights to obtain individual high score.

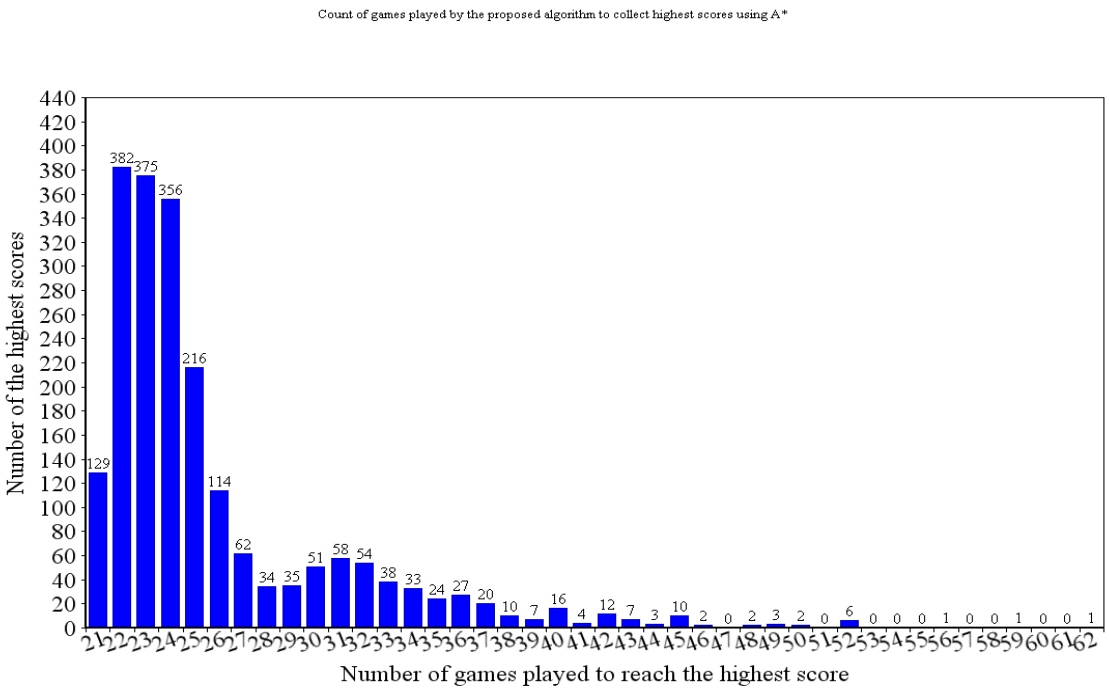


Figure 5.6: Count of games played in algorithm to collect high score using A*

2000 random set of values are used to get 2000 final optimized set of values. These 2000 final optimized set of values are the result of the highest score obtained from different number of games.

The algorithm starts with 4 random weights where one of the weights for lair i.e. junction is fixed. Although, the junction is fixed meaning it was not given any random weight like other four parameters but the value is incremented and decremented for the junction as well. So, initial game is played with the random weights then the weights are increased and decreased in Iteration 1 and 2.

In iteration 1 all the 5 weights are at least incremented and decremented once. i.e. 10 games are played when 5 weights are incremented and decremented once. Therefore, in iteration 2 also at least 10 games are played when 5 weights are incremented and decremented once. The algorithm will proceed to iteration 2 even if the random weights in iteration 1 yields highest score.

Thus, 21 games are at least played (Initial game plus 10 games in iteration 1 plus 10 games in iteration 2).

In Figure 5.6 , X axis shows the number of games played and Y axis shows the number of optimized score. example: X axis = 21 and Y axis = 129, i.e. 129 optimized scores are collected where each optimized score is highest from 21 games. X axis = 22 and Y axis = 382, i.e. 382 optimized scores are collected where each optimized score is highest from 22 games.

In algorithm the initial weights are increased and decreased with 1 every time, what about instead of increasing and decreasing weights random sets are used every time ? To compare algorithm with only random set of weights data from Figure 5.6 is used.

Result shown in Figure 5.6 is used to collect same amount of high score with equal number of games only using random set of weights. example X axis = 21 and Y axis = 129., i.e. 21 games are played with 21 random sets and the highest score is recorded, this process is repeated until 129 highest scores are collected. X axis = 22 and Y axis = 382 i.e. 22 games are played with 22 random sets and the highest score is recorded, this process is repeated until 382 games are collected.

In this way to compare with random sets similar distribution of games are played only with random sets as described in above paragraph. The result of this comparison is shown in Figure 5.7

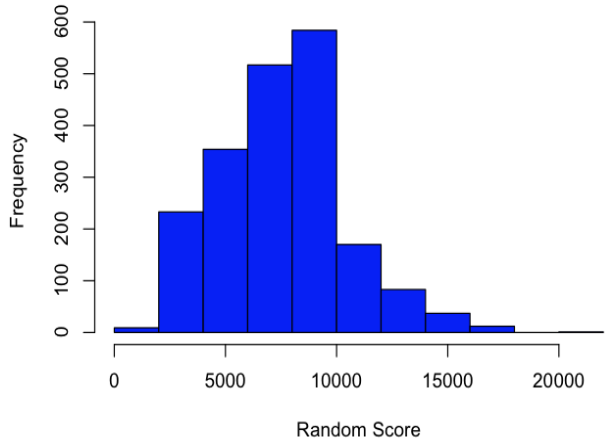


Figure 5.7: Random score frequency for A*

Figure 5.7 is the high score obtained with only random set of weights for A*.

Figure 5.7 is obtained only with random sets with similar distribution of games based on Figure 5.6. In Figure 5.7, X axis shows the score obtained and Y axis shows the frequency. While comparing Figure 5.7 (Highest score obtain with random sets) with Figure 5.1 (b) (Highest score obtain applying algorithm) it shows the algorithm is performing better compare to only using random sets.

In conclusion of experiment 1 result, the algorithm is able to score maximum value of 16934, 10426 and 8923 for A*, Euclidean and Manhattan respectively. This shows that the algorithm is able to explore space that results high value. Figure 5.4 and Figure 5.5 shows the algorithm is able to search through huge parameter space. Thus we take the global maxima, as the optimal point in parameter space because the highest score obtained is result of number of games with average of 10 games. Finally, we move to second experiment with the optimal point yielded in first experiment. Figure 5.8 below shows the optimal point used in second experiment.

Distance			**Optimal Weights**			
	Power Pills	Ghost	Edible Ghost	Normal Pills	Junction	Highest Score
A*	109	382	400	10	2	16934
Euclidean	279	106	292	26	2	10426
Manhattai	43	123	273	33	2	8923

Figure 5.8: Optimal points of individual distance.

Table 5.2 t-test

5.1.5 Hypothesis I Random optimized Score and A* Optimized Score

Input	Two independent samples of score : x1,x2,x3..Xn where n=2000 and y1,y2,y3..Ym where m=2000
Ho	$\mu_R \geq \mu_O$ i.e. the expected mean values of R is greater than O $t = \frac{\overline{X_R} - \overline{X_O}}{\sqrt{\frac{S_R^2}{N_R} + \frac{S_O^2}{N_O}}}$ $t_0 = \frac{7594.925 - 10889.76}{\sqrt{\frac{2809.3^2}{2000} + \frac{1300.4^2}{2000}}} = -47.5985$
Criteria	Two sided $(H_1 : \mu_R \leq \mu_O)$: reject $H_0 if \lvert t_0 \rvert > t_\alpha, df$ Degrees of freedom $(f) = \frac{(\frac{S_R^2}{N_R} + \frac{S_O^2}{N_O})^2}{\frac{S_R^4}{N_R^2(N_R - 1)} + \frac{S_O^4}{N_O^2(N_O - 1)}}$ $= \frac{(\frac{2809.3^2}{2000} + \frac{1300.4^2}{2000})^2}{\frac{2809.3^4}{2000^2(2000-1)} + \frac{1300.4^4}{2000^2(2000-1)}} =$ 2818.041 t_α, df $t_=$1.960 $\lvert -47.5985 \rvert > 1.960$ and hence H_0 is rejected

5.1.6 Hypothesis II Random optimized Score and Euclidean Optimized Score

Input	Two independent samples of score : x1,x2,x3..Xn where n=2000 and y1,y2,y3..Ym where m=2000
Ho	$\mu_R \geq \mu_O$ i.e. the expected mean values of R is greater than O $t = \frac{\overline{X_R} - \overline{X_O}}{\sqrt{\frac{S_R^2}{N_R} + \frac{S_O^2}{N_O}}}$ $t_0 = \frac{5473.14 - 7182.854}{\sqrt{\frac{2047.1^2}{2000} + \frac{1193.2^2}{2000}}} = -32.26925$
Criteria	Two sided $(H_1 : \mu_R \leq \mu_O)$: reject $H_0 if \lvert t_0 \rvert > t_\alpha, df$ Degrees of freedom $(f) = \frac{(\frac{S_R^2}{N_R} + \frac{S_O^2}{N_O})^2}{\frac{S_R^4}{N_R^2(N_R - 1)} + \frac{S_O^4}{N_O^2(N_O - 1)}}$ $= \frac{(\frac{2047.1^2}{2000} + \frac{1193.2^2}{2000})^2}{\frac{2047.1^4}{2000^2(2000-1)} + \frac{1193.2^4}{2000^2(2000-1)}} =$ 3216.73 t_α, df $t_=$1.960 $\lvert -32.26925 \rvert > 1.960$ and hence H_0 is rejected

Table 5.3 t-test

5.1.7 Hypothesis III Random optimized Score and Manhattan Optimized Score

Input	Two independent samples of score : x1,x2,x3..Xn where n=2000 and y1,y2,y3..Ym where m=2000
Ho	$\mu_R \geq \mu_O$ i.e. the expected mean values of R is greater than O $t = \frac{\overline{X_R} - \overline{X_O}}{\sqrt{\frac{S_R^2}{N_R} + \frac{S_O^2}{N_O}}}$ $t_0 = \frac{3972.84 - 5666.367}{\sqrt{\frac{3052.3^2}{2000} + \frac{1724.1^2}{2000}}} = -21.60468$
Criteria	Two sided $(H_1 : \mu_R \leq \mu_O)$: reject $H_0 if \lvert t_0 \rvert > t_\alpha, df$ Degrees of freedom $(f) = \frac{(\frac{S_R^2}{N_R} + \frac{S_O^2}{N_O})^2}{\frac{S_R^4}{N_R^2(N_R-1)} + \frac{S_O^4}{N_O^2(N_O-1)}}$ $= \frac{(\frac{3052.3^2}{2000} + \frac{1724.1^2}{2000})^2}{\frac{3052.3^4}{2000^2(2000-1)} + \frac{1724.1^4}{2000^2(2000-1)}} =$ 3156.74 t_α, df $t_=$1.960 $\lvert -21.60468 \rvert > 1.960$ and hence H_0 is rejected

t critical value 1.960 is chosen for distribution of freedom (df) larger than 2000 with significance level 0.05 from t distribution table.

The Bonferroni correction sets the significance cut-off at α/n where α is the significance level and n is number of tests. In the above 3 tests with significance level 0.05, the Bonferroni correction of p value is $0.05/3 = 0.01666666666667$. The p value of each individual hypothesis is less than .00001 i.e the p value of each individual hypothesis is lower than 0.01666666666667 therefore null hypothesis is rejected.

5.2 Experiment 2 Results

The result shows null hypothesis is rejected that means the alternative hypothesis is accepted which means random parameter does not perform better than proposed algorithm.

The optimal weights from Figure 5.8 are used to conduct experiment 2. In experiment 2, 10,000 game trials are run for each of the three experiments (A*,Euclidean and Manhattan) where for each distance measure with optimal weights fixed for all the 10,000 games. The experiment 2 has been carried out to compare mean score of A* with Euclidean and Manhattan. We have presented frequency and

density of score from A*, Euclidean and Manhattan.

5.2.1 Euclidean and A*

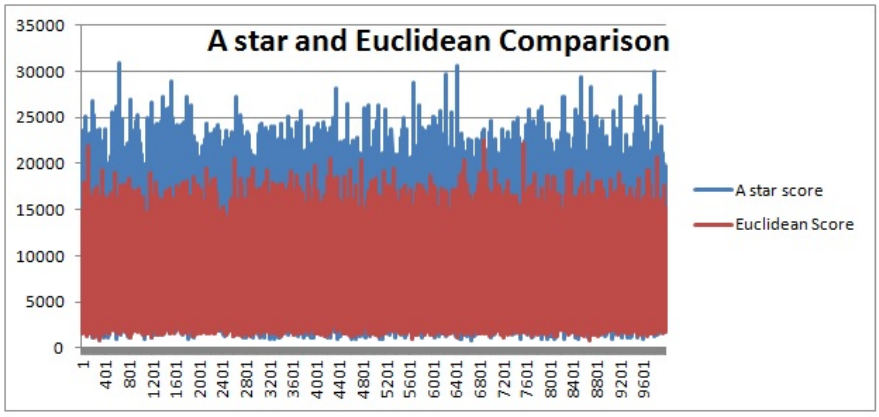


Figure 5.9: Euclidean vs A* Comparison.

5.2.2 Manhattan and A*

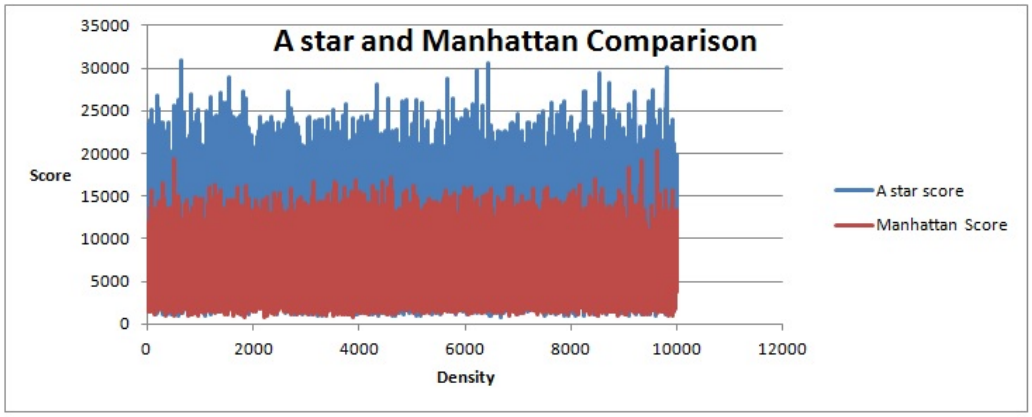


Figure 5.10: Manhattan vs A* Comparison.

After collecting the experimental data, the descriptive statistics are used to describe and present the data graphically. The Figures 5.9 and 5.10 depict that the average score of A* is more efficient compared to Euclidean and Manhattan distance. The densities of average score of Euclidean and Manhattan tend to be quite similar whereas the density of average score of A* is twice efficient. The mean score from the experiment are 10790, 7469 and 6108 for A*, Euclidean and Manhattan respectively. The highest score obtained from the experiment are 30930 , 22480 and 20260 for A*, Euclidean and Manhattan respectively.

5.3 Conclusion

For RQ1, Section 5.1.1 Figure 5.2 (a) and (b) shows increment of score where the highest score incremented is 8000. Moreover algorithm was able to increment 93.55% of random parameter space and only 6.45% was not able to increase.

For RQ2, Section 5.1.4 and welch t test proves that algorithm was able to perform better than random optimal parameter space.

For RQ3, Experiment 2 clearly shows A* was able to provide highest score 30,930 compared to Euclidean and Manhattan. Moreover, the mean score of A* is 10790 which is greater than mean score 7469 and 6108 of Euclidean and Manhattan. We conclude that A* provides highest score when run in same algorithm, controller and environment.

Chapter 6

Discussion and Validity Threats

6.1 Discussion

There are a lot of algorithms, which we could have chosen for Experiment 1 such as hill climbing, greedy algorithm, weighted algorithm e.t.c. Our requirement was an algorithm, which could explore a huge search space with time constraints. So we made an effort to introduce this algorithm, which could explore a huge parameter space with time constraints.

The results of experiment number 1 show that the proposed algorithm is suitable to find optimal points. From Figure 5.2 raise in score clearly shows the algorithm was able to optimize the random points however we were not satisfied because the samples for random point in space and optimize points in space are from different distribution. To make a comparison we collected random points in space with same distribution, which is based on games played in each iteration.

In Figure 5.6, we can see the score collected from games with parameter space visited i.e. 400 games are run where each individual games have visited at least 21 different parameter space and out of 21 parameter space the highest score is taken. The parameters space visited are from 21 to 62. Using this frequency table the random games are run to obtain data from similar distribution. The output of the result Figure 5.7 shows on using same distribution of sample still the algorithm is able to perform better than random parameters. If we look deeper into the result the algorithm has optimized score greater than 12,000.

The reason for nominal optimization might be of incrementing and decrementing with value 1. There exist local optima, Figure 5.3 shows 65% of random values were unable to pass through second iteration. This may be due to random probability sampling we have used for the value. e.g. low positive value for power pills, low negative value for ghosts, the game ends quickly without moving forward.

This has been noted and suggested for future research to implement superior

version of the algorithm and make comparison with the recent results. Moreover, future research also includes incrementing based on percentage and using float values.

Every random value will go through iteration 2 and the game shows only 35% of the value has crossed iteration 2. 7594 and 10889 are average score obtain from random and optimize algorithm where average increment of score in terms of percentage is 30.25%. To prove further we have run Welch t-tests which disprove our null hypothesis and approve our alternate hypothesis. This strongly suggest no doubt the optimization algorithm is better than random parameter space. We have performed three different t tests for A*, Manhattan and Euclidean distance measures and all of them disprove our null hypothesis and approve our alternate hypothesis.

In the experiment No.2, 10790, 7469 and 6108 are the mean score values of A*, Euclidean and Manhattan from 10,000 runs. The mean value suggest that Euclidean and Manhattan performance is close enough. A* in the other hand has performed far more better than Euclidean and Manhattan. The density graph clearly show without no doubt A* is the most effective among three of them.

6.2 Validity Threats

Validity threats are very important to consider while performing any kind of experiments. These threats deviate the result from being accurate and precise. They are discussed by categorizing into four main different types. They are as follows.

6.2.1 Internal validity

This threat is caused due to confounding variables that change the value of dependent variables. This can be a threat in our experiment since we lock one parameter (e.g junction) and see the result in terms of other parameters like power pills, normal pills, ghosts and edible ghosts. However, in most of the cases it gives good result and junctions have minor influence as compared to other parameters. There may be such situation in the maze in which influence of junction is more significant.

6.3 External validity

External validity are those conditions which limit the possibility to generalize the experimental results and techniques to other games. In our results A* performs better than other distance measure however computational cost of A* is high

compared to Euclidean and Manhattan. In our experiment A* cost, path is pre calculated and cached moreover Ms. Pac Man maze is smaller compared to other huge terrain based games and real world application. There may be situation where due to real time cost computation and size of map, Euclidean and Manhattan may perform better than A*.

Chapter 7

Conclusion

We conclude that the algorithm is suitable to find optimal point in parameter space and the algorithm is able to optimize the random parameter space. However, using suggestion presented in discussions there is space of improvement in optimization of optimal space. We also conclude that there is space of improvement to increase the iteration and find more optimal parameters. The algorithm was able to increase score obtained from random parameter space by 30.25%.

Finally, distance measure is vital element to consider while implementing path finding based on attractive and repelling solutions. To address our second research question, we propose using A* distance measure in influence maps is more efficient compared to Euclidean and Manhattan in potential fields.

Chapter 8

Future Work

Since the efficiency of distance measure used in influence maps and potential fields have been identified in this effort, proposed future work deals with improvement of algorithm used. The algorithm can be improved by using other non probability sampling techniques, using percentage instead of fixed increment of 1 during optimization. Future research can be comparison of algorithm with other algorithms such as advanced hill climbing algorithm in Ms. Pac-Man. The algorithm can be used in other environment to verify how good it is able to evolve optimal parameters. Similar experiment can be run with implementation of Influence map in ghosts. Solutions that have been proposed in discussions against the issues can be worked on for further improvement of algorithm.

References

[1] N. Bell, Xinghong Fang, R. Hughes, G. Kendall, E. O'Reilly, and Shenghui Qiu. Ghost direction detection and other innovations for Ms. Pac-Man. In *2010 IEEE Symposium on Computational Intelligence and Games (CIG)*, pages 465 –472, August 2010. 00006.

[2] Avrim Blum, Prabhakar Raghavan, and Baruch Schieber. Navigating in Unfamiliar Geometric Terrain. In *Proceedings of the Twenty-third Annual ACM Symposium on Theory of Computing*, STOC '91, pages 494–504, New York, NY, USA, 1991. ACM. 00142.

[3] L.L. DeLooze and W.R. Viner. Fuzzy Q-learning in a nondeterministic environment: developing an intelligent Ms. Pac-Man agent. In *IEEE Symposium on Computational Intelligence and Games, 2009. CIG 2009*, pages 162 –169, September 2009. 00013.

[4] P.K. Egbert and S.H. Winkler. Collision-free object movement using vector fields. *IEEE Computer Graphics and Applications*, 16(4):18 –24, July 1996.

[5] Jonas Flensbak. Flock behavior based on influence maps. *Department of Computer Science, University of Copenhagen (DIKU), Denmark, Bachelor thesis*, 2007. 00005.

[6] Johan Hagelbck and Stefan J. Johansson. Using multi-agent potential fields in real-time strategy games. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Volume 2*, AAMAS '08, pages 631–638, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems. 00059.

[7] I.D. Horswill. Lightweight Procedural Animation With Believable Physical Interactions. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(1):39–49, March 2009. 00018.

[8] Stefan J. Johansson. A survey of the use of artificial potential fields and influence maps in game ai research. 2013.

[9] Simon M. Lucas. Ms Pac-Man versus ghost-team competition. In *IEEE Symposium on Computational Intelligence and Games, 2009. CIG 2009*, page 1, September 2009.

[10] P. Rohlfshagen and S.M. Lucas. Ms Pac-Man versus Ghost Team CEC 2011 competition. In *2011 IEEE Congress on Evolutionary Computation (CEC)*, pages 70 –77, June 2011.

[11] S. Samothrakis, D. Robles, and S. Lucas. Fast Approximate Max-n Monte Carlo Tree Search for Ms Pac-Man. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(2):142 –154, June 2011.

[12] N. Shaker, J. Togelius, G.N. Yannakakis, L. Poovanna, V.S. Ethiraj, S.J. Johansson, R.G. Reynolds, L.K. Heether, T. Schumann, and M. Gallagher. The turing test track of the 2012 Mario AI Championship: Entries and evaluation. In *2013 IEEE Conference on Computational Intelligence in Games (CIG)*, pages 1–8, August 2013. 00009.

[13] J. Svensson and S.J. Johansson. Influence Map-based controllers for Ms. PacMan and the ghosts. In *2012 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 257 –264, September 2012.

[14] G. Synnaeve and P. Bessiere. A Bayesian model for RTS units control applied to StarCraft. In *2011 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 190 –196, September 2011.

[15] Tse Guan Tan, J. Teo, and P. Anthony. Uniform versus Gaussian mutators in automatic generation of game AI in Ms. Pac-man using hill-climbing. In *2010 International Conference on Information Retrieval Knowledge Management, (CAMP)*, pages 282 –286, March 2010.

[16] T. Uusitalo and S.J. Johansson. A reactive mutli-agent approach to car driving using artificial potential fields. In *2011 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 203 –210, September 2011.

[17] Dr. S.Karthikeyan V.V.Gomathi. Performance analysis of distance measures for computer tomography image segmentation, 2014.

[18] N. Wirth and M. Gallagher. An influence map model for playing Ms. Pac-Man. In *Computational Intelligence and Games, 2008. CIG '08. IEEE Symposium On*, pages 228 –233, December 2008.

[19] Claes Wohlin, Per Runeson, Martin Hst, Magnus C. Ohlsson, Bjrn Regnell, and Anders Wessln. Experiment Process. In *Experimentation in Software Engineering*, number 6 in The Kluwer International Series in Software Engineering, pages 31–39. Springer US, January 2000.

[20] Albert L. Zobrist. A model of visual organization for the game of GO. In *Proceedings of the May 14-16, 1969, spring joint computer conference*, AFIPS '69 (Spring), pages 103–112, New York, NY, USA, 1969. ACM. 00080.

Appendix A

Appendix

```
Iteration: 1
52 111 88 2 2 ------------------Random set

6586 52 111 88 2 2
8737 51 111 88 2 2 ---------------------- Highest
(optimized value)
8597 50 111 88 2 2
5391 53 111 88 2 2
7535 51 110 88 2 2
6375 51 112 88 2 2
7700 51 111 87 2 2
7515 51 111 89 2 2
6070 51 111 88 1 2
7639 51 111 88 3 2

Iteration: 2
51 111 88 2 2 ----------------------
Optimized value from Iteration 1
8247 50 111 88 2 2
8634 49 111 88 2 2
8125 48 111 88 2 2
9137 52 111 88 2 2
6281 53 111 88 2 2
7658 52 110 88 2 2
7172 52 112 88 2 2
8820 52 111 87 2 2
6111 52 111 89 2 2
6372 52 111 88 1 2
9634 52 111 88 3 2 ------------------- Highest
(optimized value)
7841 52 111 88 4 2

Iteration: 3
52 111 88 3 2 ----------------------
Optimized value from Iteration 2
6763 51 111 88 3 2
9087 53 111 88 3 2
6097 54 111 88 3 2
9574 53 110 88 3 2
8580 53 109 88 3 2
8562 53 112 88 3 2
6772 53 110 87 3 2
6657 53 110 89 3 2
7438 53 110 88 2 2
7837 53 110 88 4 2

final optimized : : 52 111 88 3 2
Score from Random set : 6586
Score from final optimized sert : 9634
```

```
Game game;
Pacman objPacman = new Pacman();
Random random = new Random(0);

/** Temporary array to hold value in both +ve and –ve direction **/

int[] decrementWeight = new int[4];

decrementWeight[0] = (int) (Math.random() * 399) + 3;
decrementWeight[1] = (int) (Math.random() * 399) + 3;
decrementWeight[2] = (int) (Math.random() * 399) + 3;
decrementWeight[3] = (int) (Math.random() * 39) + 1;

int[] incrementWeight = new int[4];
incrementWeight = decrementWeight;

/** This 2 array holds the score yielded from above temporary value to make highest
comparison and selection */

int decrementScore = 0;
int incrementScore = 0;

/** Array used for logging score and value **/

int highestAverageScore = 0;
int currentIterationAvgScore = 0;

do {

    int Counter=0;
    int iteration = 0;
    int averageScore = 0;

    highestAverageScore=currentIterationAvgScore;

    do {
            do{
                     currentIterationAvgScore=averageScore;

    /*** This block will increase and decrease value of the array decrementWeight
       and incrementWeight
    * Stores the score in array decrementScore and incrementScore.
    * Array size is 4 and counter 0 to 7.
    * This block will decrease weight if counter is 0,2,4,6 (Even)
    * and will increase weight if counter is 1,3,5,7 (Odd)
    *
    */

                    if(Counter%2==0)
                    {
                        decrementWeight[Counter/2] =
                        decrementWeight[Counter/2]-1; // Decrement Value
                        objPacman.setWeight(decrementWeight);
                        // Set value in game

                        decrementScore = currentIterationAvgScore;
            // Store score from
```

```
                        printArray = decrementWeight;
// Array for logging

                    }
                    else
                    {
                         incrementWeight[Counter/2] =
                         incrementWeight[Counter/2]+1;
                        // Increment Value

                         objPacman.setWeight(incrementWeight);
                        // Set value in game

                         incrementScore = currentIterationAvgScore;
                        // Store score from

                         printArray = incrementWeight
                        // Array for logging
                    }

    /*** Main section to run game. Game run 10 times due to stochastic behavior.**/

                    int score=0;
                    int trials = 10;

                    for(int i=0;i < trials;i++)
                    {
                        score+=game.getScore();
                //Run the game as we have already set weight with object objPaman
                    }

                     averageScore = score/trials;

                     /** Executed only once in the beginning
                     * This block is needed for logging the CurrentIteration
                     **/
                     if(currentIterationAvgScore==0)
                     {
                        currentIterationAvgScore = averageScore;
                     }

                     functionToLog(printArray,currentIterationAvgScore);
//Function to log for data collection
    /** End of section to run game **/

    /** If we find improvement in score do not increase counter and continue the
loop
    * If the score is not improved while loop will exit and counter will be increased
    **/

                }while(currentIterationAvgScore<=averageScore);

            Counter++;

/*** Selection of Array Value between decrementValue[index] and
incrementValue[index]
 **  Selection is done based on value of decrementScore[index] and
```

```
incrementScore[index]
 ** When the counter is
 **
 **   Counter to index calculation is done by formula
 ** e.g Counter = X , Index = (X/2) - 1
 **                2 ,         (2/2)  - 1 = 0
 **                4 ,         (4/2)  - 1 = 1
 **                6 ,         (6/2)  - 1 = 2
 **                8 ,         (8/2)  - 1 = 3
 **
 **  Counter 2 = Selection between decrementValue[0] and incrementValue[0]
 **  Counter 4 = Selection between decrementValue[1] and incrementValue[1]
 **  Counter 6 = Selection between decrementValue[2] and incrementValue[3]
 **  Counter 8 = Selection between decrementValue[3] and incrementValue[3]
 **/


/** This loop will only execute when counter value is 2,4,6,8 **/

            if(Counter%2==0)
            {

                  if(decrementScore > incrementScore)
                 {
                    /** Selected value is assign to both decrementWeight and
incrementWeight because we want the array to be synced all the time
                    ** When index value is synced we have to sync it so that it
will be same when we work in other array index.
                    ** i.e. index 0 needs to be sync when we work in index 1 ,
1 needs to be sync when we work in index 2, 2 needs to be sync when we work in
index 3
                    **/
                     decrementWeight[(Counter/2)-1] =
(decrementWeight[(Counter/2)-1]) + 1;   // Added +1 to value because the value
was decremented above

// and that did not yield highest socre that is the value before that

// value yielded highest score



                     incrementWeight[(Counter/2)-1] =
decrementWeight[(Counter/2)-1];         // Sync incrementWeight with
decrementWeight
                 }
                 else
                 {
                     incrementWeight[(Counter/2)-1] =
(incrementWeight[(Counter/2)-1]) - 1 ;  // Subtracted -1 to value because the
value was incremented above

// and that did not yield highest socre that is the value before that

// value yielded highest score

                     decrementWeight[(Counter/2)-1] =
```

```
172   incrementWeight[(Counter/2)-1] ;        // Sync incrementWeight with
173   decrementWeight
174                   }
175               }
176
177       /** Run the loop from 0 to 7 where **/
178
179       }while(Counter<8);
180
181           /** In the next iteration instead of using random value we are selecting
182       value that
183       *   yielded highest score and repeating the process
184       */
185        iteration++;
186
187   }while(highestAverageScore<currentIterationAvgScore);
```

A.1 Ms. Pac-Man

Pac-Man is a popular arcade game originally produced by Midway and developed by Toru Iwatani for Namco Company in year 1981 [1]. Pac-Man is a point based game where points are collected by eating pills. The player aims to guide Pac-Man around the maze to collect all the pills without getting trapped by the ghosts.

Maze is a layout of the game which consist of ghosts, lairs, Pac-Man, normal pills, power pills, fruits, junctions and teleports [11]. All these components are described below in detail.

Figure A.1: Ms. Pac-Man Maze.

A.1.1 Ghosts

Ghosts hinder Pac-Man to eat the normal pills, power pills and fruits. When all the normal pills are eaten, the level of the game advances forward with different types of maze. As soon as Pac-Man eats a power pill, ghosts become edible for a short period of time and successive eating of those ghosts accumulates points in exponential manner. Pac-Man achieves the extra points by eating randomly appeared fruits in the maze. Pac-Man loses one of the three lives as soon as she collides with an inedible ghost. When all the three lives are lost by Pac-Man, the game ends. Ms. Pac-Man is the advanced version of Pac-Man. There are four different types of ghosts in Ms. Pac-Man. They are Blinky (red), Pinky (pink), Inky (cyna) and Sue (orange) [11].

A.1.2 Lair

In the start of each level in the game, ghosts are situated in the middle of maze, called lair. The idle time to get outside of lair and chasing Pac-Man decreases with increasing levels.

A.1.3 Pac-Man

Pac-Man is a main agent in maze which can move right, left and up to eat normal pills, power pills fruits and edible ghosts to score points. Pac-Man has three lives and game ends when all the three lives are lost. When Pac-Man scores 10000 points she gets an extra life.

A.1.4 Normal Pills

The small dots in a maze which Pac-Man needs to eat to move forward to another level are called normal pills. Once all the normal pills are eaten by Pac-Man, next level is reached. There are 250 normal pills in a maze.

A.1.5 Power Pills

Power pills are bigger pills in size compared to normal pills which are randomly situated. When Pac-Man eats power pills, ghost changes to edible state for a short period of time. In the edible state, colour of all the ghosts changes to blue and the ghosts start to blink indicating they are becoming active. There are four power pills in a maze. These power pills are situated in different positions in different mazes.

A.1.6 Fruits

Fruits appear randomly in a maze during a game and Pac-Man receives extra points by eating these fruits.

A.1.7 Junctions

Junctions are the places in a maze which have three or four directions to choose. They are the places in the maze where Pac-Man and Ghosts can change the directions (left, right and up).

A.1.8 Teleports

Teleports of a maze are special junctions which are connected to the opposite side of the maze.

There are four different types of mazes which are shown in Fig 1.2. Game starts with maze A and successively moves to B, C and D when level of the game increases.

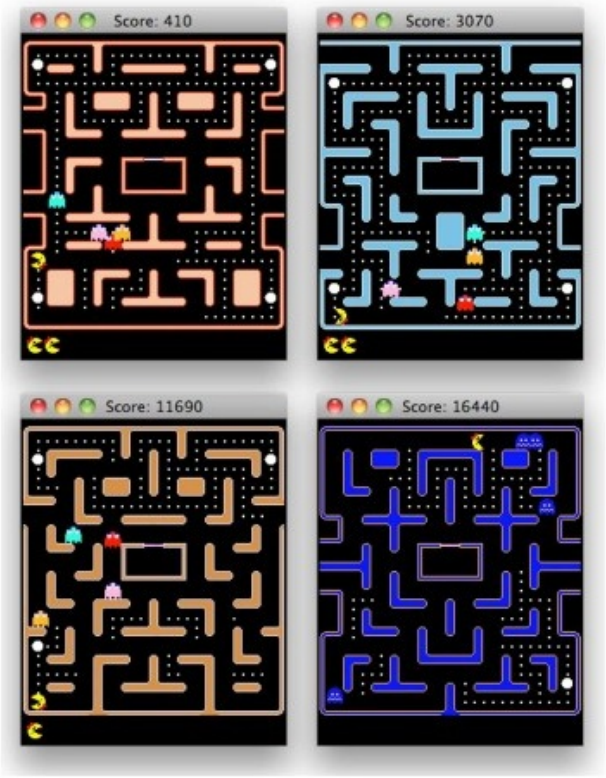


Figure A.2: Mazes.

Ms. Pac-Man is successor to Pac-Man. These two games are very similar, still they have a number of minor differences. In particular, the ghosts in Pac-Man behave deterministically whereas in Ms. Pac-Man the ghosts movement has an element of pseudo-randomness, making it impossible to learn paths and leading to a more challenging game. Ms. Pac-Man also features additional maze layouts compared to Pac-Man [18].

Ms. Pac-Man scores 50 points as soon as she eats any of the power pills. Successive eating of the ghosts after eating a power pill doubles the score each time. So, the optimal score that can be achieved after eating power pills is 3050 (= 50+ 200 + 400 + 800 + 1600). If another power pill is eaten during the edible stage of ghost, the point is reset to 200 again [11].

The game starts in maze A with three lives of Ms. Pac-Man and ghosts ready to come out from lair. Ms. Pac-Man starts to eat normal pills, power pills and

randomly located fruits and ghost starts to chase her. Once any of the power pills are eaten, ghosts become edible for a short period of time.

Ms. Pac-Man is one of the competitions included in yearly competitions track like CIG [9]. The competitions since 2007 are being held every year for implementing Ms. Pac-Man controllers [9, 16].