Exploring Exploitation and Exploration Algorithms in Ms. Pac-Man

by

Sanyat Hoque

A Thesis submitted to the Faculty of Graduate Studies of
The University of Manitoba
in partial fulfilment of the requirements of the degree of

Master of Science

Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg, Manitoba

ABSTRACT

Video game technology has grown to include the development of more sophisticated game play by including computer controlled agents or Non-Player Characters (NPCs). Moreover, game environments provide ideal benchmarking tools to examine new and more sophisticated game related algorithms. Ideal testbeds are games like Pac-Man and Ms. Pac-Man. Ms. Pac-Man is a stochastic game played in a complex maze environment. In this thesis, a novel probabilistic tree search algorithm loosely based on the simulated annealing algorithm is used by the ghost agents that pursue Ms. Pac-Man in order to minimize the score of Ms. Pac-Man (the opponent). In general, simulated annealing is a non-deterministic search optimization algorithm to search for a global maximum/minimum in a non-convex discrete search space. A major advantage of simulated annealing over the tree search models used in past research (Ant Colony Optimization and Monte Carlo Tree Search) as applied to the ghost agents in Ms. Pac-Man is the time and memory complexity. Although simulated annealing is still computationally intensive, it is less so in comparison to the search algorithms mentioned above. Simulated annealing and variants are also excellent sources of easy-to-understand analogs. This thesis investigates the trade-off between exploration and exploitation phases of search algorithms used by ghost agents as they attempt to contain or constrain the movements of Ms. Pac-Man. New algorithms are developed and presented to better control the NPCs within Ms. Pac-Man. The proposed models or algorithms are evaluated on two different experimental setups; 1) maze navigation to test the convergence efficiency with respect to the number of iterations, and 2) within the game of Ms. Pac-Man itself to determine the effectiveness of attacking strategies such as flanking and blocking.   These new algorithms may have application within similar types of problems.

ACKNOWLEDGEMENTS

I would like to acknowledge the support of my advisor, Dr. Robert McLeod, for introducing me to Machine Learning and encouraging me to pursue this project for my Master’s thesis.

TABLE OF CONTENTS

LIST OF FIGURES & TABLES

CHAPTER 1

INTRODUCTION

The video game, graphic and virtual reality industry has seen an exponential increase in economic impact and development [1]. However, there has not been a lot of development in the evolution of game bots or Non-Player Characters (NPCs). Developing sophisticated gameplay has a greater potential for lifelike behavior and strategies of game bots within the game environment. Artificial Intelligence (AI) has an untapped potential in providing very sophisticated behaviors of game bots to make video gaming more fascinating and/or engaging. Artificial Intelligence algorithms include a wide variety of discrete optimization algorithms.

The gaming industry has a desire to generate competitive behavior among computer controlled opponents, bots, or NPCs. In recent years, stochastic video games have co-evolved to become a proving ground for AI algorithms. Probabilistic games such as the Game of Go, Chess, Pac-Man and Ms. Pac-Man provide excellent environments to test AI algorithms since these game have simple rules from which emerge sophisticated strategies associated with play and winning [2][3][4]. Competitions on the use of AI algorithms within the Ms. Pac-Man game environment include: the Ms. Pac-Man competition, where inputs come from screen capture [5] and Ms. Pac-Man vs. Ghost competition where agents are interfaced with a game engine [6].

In a random, dynamic game environment, AI bots are beginning to be used for making game play decisions. The game of Ms. Pac-Man is widely accepted as a suitable test bed for evaluating AI techniques along with Go and Chess. The annual competition of Ms. Pac-Man vs. Ghost Team along with a dedicated course "CS188 Intro to AI" at UC Berkley has facilitated the growth of AI into gaming environments. Participants can use Machine Learning (ML) algorithms such as Artificial Neural Networks (ANN), Monte Carlo Tree Search, and Adversarial Search in ghost teams as well as for Ms. Pac-Man, giving rise complex tactical behaviour of the agents.

In previous research, stochastic search algorithms such as Ant Colony Optimization (ACO) have also achieved very good results producing NPCs in games such as Ms. Pac-Man [7][8]. Reinforcement learning techniques such as Monte Carlo Tree Search used by ghost agents [9][10] won first place in the Ms. Pac-Man vs Ghost Team competition of 2011 [6]. Liberatore et al. constructed ghost agents with an evolutionary algorithm using Lexicographic Ranking that provided good flocking behavior among ghost agents [11]. Tan et al. created ghost agents with hill-climbing and simple feedforward neural networks [12]. The work presented here explores trade-offs between exploration and exploitation phases that arise in algorithms derived from simulated annealing search.

I. Problem Definition:

Artificial Intelligence algorithms ranging from Ant Colony optimization, Monte Carlo Tree Search have been applied in Ms. Pac-Man as well ghost agents. In all cases those reported and tested proved to be computationally expensive. In spite of good performance, these techniques require high computational power, memory and time complexity [7][8].

Since memory complexity has proven to be a difficulty with the algorithms mentioned, Simulated Annealing (SA) was conjectured to be a highly promising alternative in stochastic games such as Ms. Pac-Man. Simulated Annealing is simple and is possibly computationally cheaper with lower memory complexity while achieving reasonable NPC behaviours. To date Ms. Pac-Man and associated ghost agent behaviour does not appear to have been evaluated using SA. Therefore in this thesis, SA is used and evaluated within ghost agents to play against Ms. Pac-Man. From here, variants of SA are developed based on improving the ghost agent's objectives using notions of acceptance probabilities similar to those of SA.

II. Research Aim:

The aim of this thesis is to evaluate the behavior of ghost agents using Simulated Annealing and another algorithm inspired or motivated by SA to portray the following behaviors:

(1) The efficiency of ghost agents in navigating through a game environment to attack their adversary in different parts of the maze environment.

(2) Develop attacking mechanisms such as flanking and blocking maneuvers of ghost agents by means of the proposed SA inspired model in the process of converging towards their enemy.

Due the movement restrictions of ghost agents in game simulators used in annual competitions such as CEC 2011 in reference [6] and CIG 2016 in reference [13], a new simulator model was developed for this thesis to avoid all restrictions and provide an algorithm development and evaluation platform.

III. Contribution:

Contributions of this thesis are as follows:

(1) The introduction and innovation of a stochastic tree search algorithm variation based on SA in the Ms. Pac-Man game environment.

(2) Development of attacking behaviors of ghost agents in dealing with their adversary resulting from the use of the proposed SA inspired algorithm.

(3) Development and application of a new a Ms. Pac-Man vs. Ghost Agents simulator to address the ghost movement limitations such as the restriction of turning back in a tunnel and computation time of 40 ms that were present in previous simulators in references [6] and [13]. The simulator also serves as the algorithm development environment that was used for the mSA algorithm.

This chapter will introduce the game environment of Ms. Pac-Man and the simulator used in this thesis. Furthermore, this chapter will also discuss the approach used in this thesis to apply SA based AI in ghost agents in the game environment and evaluate its performance.

IV. Ms. Pac-Man Game Simulator:

The game of Ms. Pac-Man has attracted the attention of the AI research community because of its stochastic nature, thereby providing different outcomes, and it being familiar, thereby making it easy to understand various agent objectives. The game of Ms. Pac-Man is probabilistic in nature, i.e. ghost agents behave in a random manner which makes it impossible to master the game in contrast to a deterministic environment. Competitions are held annually where different machine learning (ML) algorithms are used to generate behaviors of ghost agents and of Ms. Pac-Man, where these algorithms are evaluated and compared against other machine learning algorithms within particular game engines or simulators. The game starts with Ms. Pac-Man navigating through a complex maze trying to maximize her score by eating pills while four ghost agents are trying to minimize her score by terminating her. Ms. Pac-Man has four set of actions to move vertically or horizontally if there are no walls or obstacles. Like any search tree, each vertex has a list of neighboring vertices depending on its position in the maze, for example, an L-shaped junction vertex will have two neighboring nodes, T-shaped junction node will have three neighboring nodes, while a crossroad will have four neighboring nodes connected with edges.

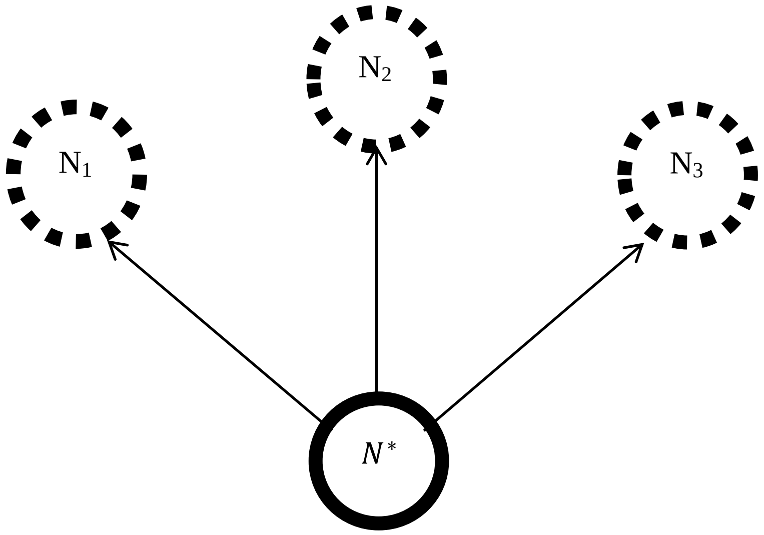


Figure 2: Example of a search tree. N^* is the current state and N_1, N_2 and N_3 are neighboring states.

Similar to past annual competitions such as CEC 2011 [6] and CIG 2016 [13], the speed of all the agents in the game environment are same. In the simulators used in the annual competitions as well the simulator used in this thesis, the maze environment is modelled as vertices linked with edges.

In contrast to the simulator or game engine used in Ms. Pac-Man vs. Ghost Team CEC 2011 competition [5], the simulator used for this work does not have any power-pills in the corner of the maze environment by which Ms. Pac-Man gains additional points. Moreover in the previous simulators used in the competition [6] and [13], while Ms. Pac-Man does not have any restrictions in its movement through the maze, there are some restrictions in the movement of ghost agents. These restrictions in references [6] and [13] include restricting a ghost from turning back and only being able to choose its direction in T-shaped, L-shaped or crossroad junctions, in addition to a probability of global reversal of 0.005. Global reversal is the term used when all ghost agents change their direction together and travel in a reverse direction. Ghosts' inability to turn back in a tunnel is contrary to the rules of the ghost agents used in the simulator developed in this thesis. In the simulator developed for this thesis, ghosts are able to turn back without any restrictions allowing ghosts the ability to calculate their decisions at every unit distance it travels throughout the maze instead of waiting to come across a junction by developing heuristic search that calculates to find the shortest path to Ms. Pac-Man anywhere in the maze including the junctions which is discussed in more details in chapter 2. This is not viewed as a deficiency as the simulator developed here still allows for comparison of various AI strategies.

In Ms. Pac-Man vs. Ghost Team CEC 2011 competition [6], each level is limited to 2000 game ticks, after which, the game moves on to the next level. No such limitations exist in the simulator used for this thesis as it was deemed as a somewhat arbitrary constraint. The simulator used in the above mentioned competition is written in Java while simulator used in this thesis report is developed in Matlab. In contrast to the Ms. Pac-Man vs. Ghost Team competition simulator [6] that limits each move of an AI controlled agent to 40 ms, the Matlab based simulator does not have any time restrictions. Since annual competitions are held on various game simulators, it is not possible to use those algorithms directly for performance evaluation in this thesis. Rather, a qualitative comparison of behaviours is made (face validity). In this thesis, the ghost agents are in a partially observable environment consisting of the visibility of single unit distance similar to the partially observable environment in Ms. Pac-Man vs. Ghost Team [13]. The objective of building a Ms. Pac-Man simulator was to directly aid in the process of algorithm development and evaluation, as opposed to providing a direct comparison.

V. Approach:

Chapters 2, 3, 4 and 5 discuss the heuristic search methods that are used to construct neighboring nodes in tree search algorithms, then baseline algorithms denoted Hill Climbing (HC) and a simple version of SA denoted the Vanilla Simulated Annealing (vSA) algorithm is discussed. Another stochastic tree search algorithm called Ant Colony Optimization (ACO) algorithm is also discussed along with its proposed model in [7] and [8] as presented in previous literature. The ACO algorithm is discussed as it displays qualitatively interesting behaviours associated with flanking and blocking of Ms. Pac-Man. The ACO approach is algorithmically more complicated than the algorithms developed in this thesis.

Chapters 6 and 7 discuss the development of the proposed modified Simulated Annealing (mSA) algorithm and its performance evaluation in two separate experimental tests as follows:

(1) A comparison of the proposed model along with baseline algorithms are used in ghost agents to navigate their way to their enemy.

(2) Test the behavior of multiple ghost agents using the mSA approach illustrating attack strategies such as flanking and blocking movements in reducing the score of their enemy (Ms. Pac-Man).

Directly, comparing the above two experiments with other existing algorithms that took part in previous competitions such as [6] and [13] was not be possible since the simulators used for this thesis and the above mentioned competitions are different. However, it is possible to demonstrate the desired behaviours of ghost agents using the mSA approach. The evaluation methods above support the face macro validity of using the mSA as compared with results published.

VI. Detailed Thesis Outline:

Chapter 2 introduces the concept of heuristic search and its subsequent use in other tree search algorithms in this thesis such as Hill Climbing (HC) and the proposed model denoted modified SA (mSA). The evolution of the development of mSA is presented, motivated by trade-offs in algorithm exploration and exploration phases.

Chapter 3 provides brief introduction of a baseline HC algorithm as a non-stochastic algorithm which is used to compare performance of maze navigation in the game environment.

Chapter 4 presents another baseline algorithm called vanilla SA (vSA) which uses heuristics that generate neighboring candidate nodes guided stochastically. The vSA algorithm is also used to compare the efficiency of maze navigation. The vSA algorithm is a traditional SA algorithm that starts with a high temperature control parameter supporting exploration of the search space followed by transitions to lower temperatures. At lower temperatures, vSA supports exploitation and behaves similar to a greedy HC algorithm.

Chapter 5 briefly discusses another stochastic search algorithm called Ant Colony Optimization [7][8] (a proposed model used in recent literature). This approach is algorithmically more complex and demonstrates desirable gaming behaviours of the NPCs. Specifically, the developers of the ACO approach illustrate behaviours of flanking and blocking as a desirable strategy when approaching the adversary. This model could not be compared with proposed model used in this thesis because each model was programmed in different simulators and by different authors. The discussion is included for completeness and afford a qualitative comparison of strategies that emerge.

Chapter 6 evaluates the performance of the proposed method mSA on two separate setups – maze navigation in Ms. Pac-Man and use of mSA for ghost behaviors to flank and surround Ms. Pac-Man. The development of the algorithm is derived by noting that for maze navigation, it is more desirable to exploit initially followed by exploration. Here, a brief discussion of essentially running SA in reverse is used to demonstrate this trade-off. As opposed to just having SA run from low to high temperature (exploit to explore), a feedback mechanism is introduced that facilitates exploitation when a ghost is far away from its adversary, and exploration when a ghost is near its adversary, and most importantly re-establishes these phases based on the distance the ghost is to its adversary. In effect, the major modification is that the traditional temperature parameter used in the probabilistic acceptance function is associated with the distance the ghost is to its adversary. To the author’s knowledge, this is the first algorithm developed of this type in a general sense and in terms of ghost navigation within Ms. Pac-Man in a specific sense.

Chapter 7 summarizes the thesis and discusses advantages, limitations and future directions of the proposed algorithm.

CHAPTER 2

HEURISTIC SEARCH

I. Heuristic Search:

Heuristic search is an informed search where knowledge of the problem or game environment is used to obtain an acceptable but likely suboptimal solution. Heuristic search is often a trade-off between obtaining a global solution and efficiency of the algorithm. Heuristic search can be used to help point other search optimization algorithms to the problem solution. Having information about problem solutions helps one to search the problem space more efficiently to find the global solution or an acceptable solution. Knowledge about a global maximum as well as the cost of transforming from one state to another can be defined by terms of a heuristic value or a function. This function can be called a heuristic function or evaluation function [14][15].

II. Heuristic Function:

A heuristic function can be thought of as an estimate of the cost from the current state of the algorithm to its goal state or problem solution. Heuristic search is used when other conventional mathematical optimization algorithms cannot be used or are not suitable. Therefore a heuristic function, $h(n)$ can be defined as the evaluation function or cost between the current state of search agent to the global maxima or solution state of the problem (if known). The heuristic function may also just be a quantity that one wishes to maximize or minimize. As the search algorithms used here navigate through the maze from the current state and approaches a better solution, the value of heuristic function typically decreases.

In the path planning of ghost agents in the game of Ms. Pac-Man, the ghost agent uses the positional information of itself and the positional information of Ms. Pac-Man to calculate the Manhattan distance between them in order to converge on to Ms. Pac-Man's position. Using the positional coordinates from the grid of Ms. Pac-Man and the ghost agent, difference along the x-axis and the differences along the of y-axis are taken into account. In terms of distance measures calculations used here the effect is similar to the knowing the Manhattan between a ghost agent and Ms. Pac-Man. Depending on the nature of these differences the displacement a decision from a subset of possible moves $\in \{up, down, left, right\}$ is taken by the ghost agent. Ghosts actually use knowledge of the octant Ms. Pac-Man is in as well in making decisions as to the next node to select. This does not actually provide too much more additional information as a ghost agent would equally well be able to infer this from calculations of the Manhattan distances from adjacent nodes. As such, in Ms. Pac-Man, the heuristic function, $h(n)$ is defined as the "Manhattan" distance from search agent or ghost agent to its adversary (i.e. Ms. Pac-Man) with a slight modification of the ghost being aware of the octant Ms. Pac-Man is in with respect to the ghost's position.

Ghost agents have a visibility of a single unit distance from their position and can only interact locally up to one unit distance from their position. Therefore, the ghost can only sense whether it is in a tunnel or an L-shaped/T-shaped or a crossroad junction by sensing its surrounding obstacles and does not have any other information other than that of the coordinate of Ms. Pac-Man's position in the grid of the game environment, but no information about the maze topology is known by the ghost agents.

In this thesis, an investigation is carried out using ghost agents that use heuristic search to generate neighboring candidate nodal solutions or states in conjunction to an optimization algorithm in order to converge towards their adversary. These heuristics are referred in this report as memory based search or memoryless search algorithms. Ghost agents use these heuristics to develop path planning strategies by means of discrete search optimization algorithms that controls ghost agents in an effort to converge to the position of Ms. Pac-Man. To the author's knowledge, this is the first time a memoryless heuristic search strategy is used where ghost agents can calculate its decision in the middle of a tunnel and therefore are able to turn back if necessary in order to converge in its target. This notion of using these two heuristics search referred in this thesis as memory based and memoryless search heuristic is unique to this work.

III. <u>Memory based Heuristic Search:</u>

In the first simulation scenario video, ghosts manoeuvre through the maze obstacles using the Manhattan distance from Ms. Pac-Man and its related x-axis and y-axis only at T-shaped, L-shaped or crossroad junctions (to calculate their next best move) to travel on the shortest path to the position of Ms. Pac-Man. In this path planning strategy, ghosts calculate their next best move only when they face an obstacle or if the ghost is in a T-shaped, L-shaped or a crossroad junction. As a result, ghost agents cannot turn back in a tunnel until meeting a junction or node. In other words, the ghost agent uses its memory of its last best move it calculated to travel unless it encounters another obstacle or intersection node.

The memory based node selection path planning strategy is shown to swiftly maneuver through the maze without getting stuck in any suboptimal solution as shown in simulation video. The strategy's time taken to reach its global optima in a complex maze environment is less than that of a memoryless path planning process as demonstrated in figure 3.

In the memory based heuristic search, once a ghost makes a decision to travel in a specified direction, the ghost would keep heading in that direction until it comes to another crossroad, L-shaped or T-shaped junction, and it would not be able to turn back in a tunnel similar to the restrictions of the simulation in [6]. Therefore, the ghost displays a 'memory' of the decision it made previously and thus is referred as a memory based search heuristic in this thesis.

IV. Memoryless Heuristic Search:

In the second scenario, ghost agents travel through the maze using a Manhattan distance from Ms. Pac-Man without necessarily being at a T-shaped, L-shaped or crossroad junctions while calculating their path planning decisions. Ghost agents still use the simple heuristic based on the Manhattan distance between ghost agents and Ms. Pac-Man and its related x-axis and y-axis. The heuristic search method calculates the best move at every node or unit distance as ghost agents travel along their routes regardless of whether they sense that they are at T-shaped, L-shaped or crossroad junctions.

This process of calculating the shortest path by means of simple heuristics is repeated at every node that the ghost travels towards Ms. Pac-Man. As such, a ghost does not ‘remember’ its last best move it calculated a single unit distance earlier and calculates the shortest path every time it traverses a unit distance. The ghost does not display any ‘memory’ of its last move.

The memoryless search heuristic is similar to the above that includes ghost interactions with their local environment within the range of a single unit distance of their surrounding environment. The process of next node generation is repeated at every unit distance the ghost travels or at every iteration regardless of whether it is in a tunnel, T/L-shaped or crossroad junction. Therefore, the memoryless search heuristic search process does not display any ‘memory’ of its last move calculated. Using the memoryless search heuristic in conjunction with a search algorithm would enable a decision bot or agent to generate neighboring nodes even at tunnels in the maze environment. In contrast to references in [6] and [13], ghost agents are now able to turn back in a tunnel that further increases the effectiveness of search algorithm used for maze navigation and eventual cornering of Ms. Pac-Man.

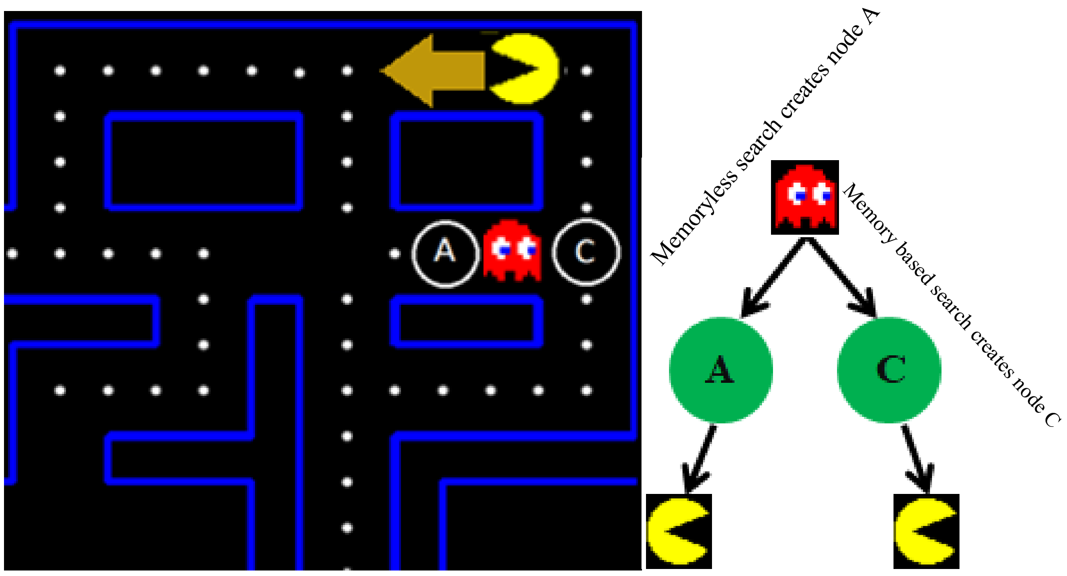

A
C
Memoryless search creates node A
Memory based search creates node C
A
C

Figure 3: A scenario of path planning heuristic search (left). Tree structure of heuristics search when the ghost creates two neighboring candidate nodes where memory based search selects vertex C and memoryless search selects vertex A.

An example of a particular scene is shown in figure 2. In this scenario, a ghost agent creates two neighboring nodes as candidates to visit (move) next while it travels through the tunnel. In general, the ghost agent's journey in a maze navigation is towards its enemy, both heuristic searches generate neighboring nodes by calculating the shortest path to the adversary.

Initially, a ghost agent's current position in the maze shown in the screenshot, both heuristics (memory based and memoryless) generate node C by calculating the shortest path. However, if Ms. Pac-Man shifts its position and starts moving westward to evade the ghost agent as indicated by the arrow, the traditional memory based search heuristic still creates Node C (as it can only calculate a new decision at a junction) while the memoryless heuristic search would calculate its next best move at every unit distance it travels. This memoryless search heuristic would generate another neighboring vertex A westward by calculating the shortest path considering the new positional coordinates of Ms. Pac-Man.

The memoryless search heuristic pseudocode can be written as:

```
Algorithm 1: Heuristic Search
1:   Compute Manhattan distance ($x - axis + y - axis$) between ghost and Ms. Pac-Man
2:   while Manhattan distance $> 0$ do
3:       Evaluate which Axis is bigger, if $y - axis > x - axis$ do
4:           if $y - axis\ is\ positive$ then
5:               Compute tentative nextMove ← $Up$
6:           else $y - axis\ is\ negative$ then
7:               Compute tentative nextMove ← $Down$
8:           end
9:       else $x - axis > y - axis$ then
10:          if $x - axis\ is\ positive$ then
11:              Compute tentative nextMove ← $Left$
12:          else $x - axis\ is\ negative$ then
13:              Compute tentative nextMove ← $Right$
14:          end
15:      end
16:      Compute $intersect\ (tentative\ nextMove, possible\ Moves)$ in current grid
17:      if $intersect\ (tentative\ nextMove, possible\ Moves) = 1$ then
18:          Ghost's nextMove ← tentative nextMove
19:      else $intersect\ (tentative\ nextMove, possible\ Moves) = 0$ then
20:          Ghost's nextMove ← Go back to step 4
21:          if $y - axis\ is\ positive$ then
22:              Compute tentative nextMove ← $Left$
23:          else $y - axis\ is\ negative$ then
24:              Compute tentative nextMove ← $Right$
25:          end
26:      else $x - axis > y - axis$ then
27:          if $x - axis\ is\ positive$ then
28:              Compute tentative nextMove ← $Up$
29:          else $x - axis\ is\ negaive$ then
30:              Compute tentative nextMove ← $Down$
31:          end
32:      end
33:  Until Manhattan Distance becomes 0.
```

A demonstration of these two path planning strategies is shown in two separate simulation scenarios (www.youtube.com/watch?v=dGYPm8Uy3PA and www.youtube.com/watch?v=ArNfSK-s_kE).

In these simulation scenes, each heuristic was used in two simulation scenarios to find Ms. Pac-Man at two separate locations. As can be seen, the ghost agents manoeuvre through the maze using the memoryless search heuristic in the first two simulation scenarios and a memory based search heuristic in the second two simulation scenarios. These videos illustrate the use of memory based and memoryless heuristic search methods as applied to path planning strategies. In these scenarios with a stationary Ms. Pac-Man, the memory based search heuristic appears to be more effective in guiding ghosts to Ms. Pac-Man. In general though, when Ms. Pac-Man is also attempting to avoid the ghost agents, it is anticipated that the memoryless heuristic would be more effective.

In figure 3, a comparison of the efficiency of the above two path planning heuristics is shown that consists of the memory based path planning and the memoryless path planning strategies in converging towards the position of the global optima. The graph demonstrates the time taken to reach the position of their stationary adversary is much less for the memory based shortest route planning strategy in contrast to the memoryless strategy in calculating the shortest route.

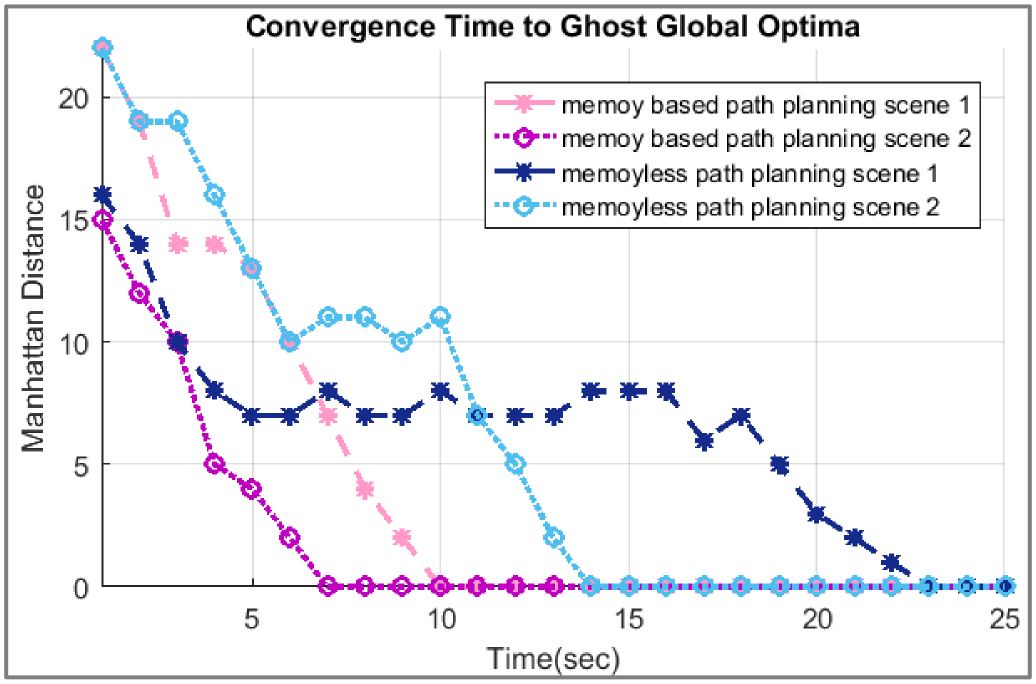


Figure 4: A comparison of CPU convergence time of path planning strategies at different locations of maze as shown in the simulation videos above.

A heuristic function is used often in tree search algorithms to produce neighboring candidate node solutions that help search algorithms look for a global maximum in the solution space. The heuristic referred in this report as the memoryless search heuristic generates one neighboring candidate node while the memory based search heuristic generates a different neighboring candidate depending on the location of ghost agent in the maze. These heuristic search methods are used in subsequent algorithms described in this thesis in the case of baseline HC algorithm, vSA, and the mSA (Hill Climbing, vanilla SA, and modified SA respectively). For proper comparison, all the algorithms use the same heuristic search strategies (ie. both memoryless and memory based search heuristic) that plays a significant role in the search process within the solution space as ghost agents are able to turn back in a tunnel and able to make a decision anywhere in the maze instead of waiting come across a junction contrary to previous research [6][7][8] [9][10] [11].

Contrasting these search heuristics are random search which would just select an adjacent node at random. This is more common in traditional SA based algorithms.

CHAPTER 3

HILL CLIMBING

HC is an iterative algorithm that improves any given solution in a search tree based on discrete vector space by making gradual adjustments to the solution and at the same time, measuring the fitness of the solution [16][17]. If the newly changed solution is better than the previous one or has a better fitness value, then the new solution is accepted and the last solution is discarded. This process is iterated until new solution's fitness value does not improve in comparison with the current solution of the algorithm. Hill Climbing is an ideal algorithm for convex optimization scenarios, since it greedily accepts a new candidate neighboring solution without exploring any other neighboring solutions. A greedy algorithm may become trapped in a local optima in the case of non-convex optimization scenarios.

Therefore as heuristic search generates a possible solution or a leaf node among a tree of nodes, the hill climbing (HC) algorithm iterates by evaluating its destination state from its current state where an evaluation function is used to measure whether the leaf node represents a state that is closer to the goal solution or not. If the evaluation function returns a value of true, the algorithm makes the generated leaf node its destination state. If the evaluation function returns a value of false, the algorithm rejects the generated leaf node. Algorithm iteration from the current state to its next best state is done via accepting a better solution with a probability of 1 or exploiting a better node in a greedy based manner without exploring any other options or neighboring nodes or options. The HC algorithm simply exploits and never explores. The overall procedure is repeated until the algorithm reaches the global optima or gets stuck in a local optima in a non-convex solution space tree of nodes. The absence of accepting a worse solution or a worse neighboring node than its current state or node in a discretized non-convex search tree typically makes the algorithm get trapped in a local optima. A local optima is a generated leaf node or a possible solution from where the heuristic cannot find a better solution state than it current state and hence gets stuck. The algorithm cannot proceed any further.

For instance, in the classic travelling salesman problem (TSP), heuristics are used to find a solution by distorting the initial solution via randomly swapping cities. Then, if the new solution is better than the current one - the solution is kept. If the new solution is worse than the current one – the solution is discarded. In other words, HC iterates towards the neighboring candidate solution and greedily accepts it without exploring any other solution. The algorithm proceeds until it gets trapped in a local minima where the swapping of cities only makes candidate solutions worse.

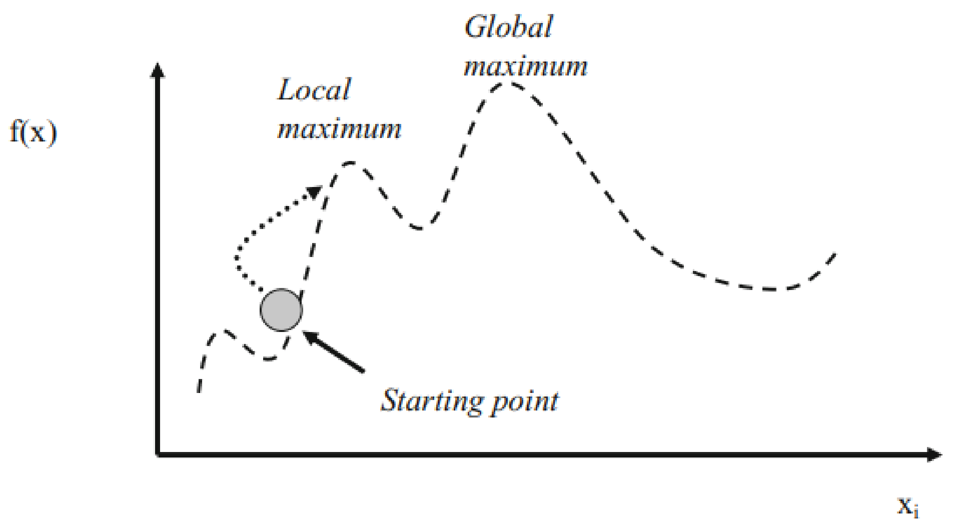


Figure 5: The HC algorithm iterating locally into a local maximum point from where it cannot iterate further. Once it gets trapped in suboptimal solution, it cannot converge into the global maxima due to its greedy selection decisions.

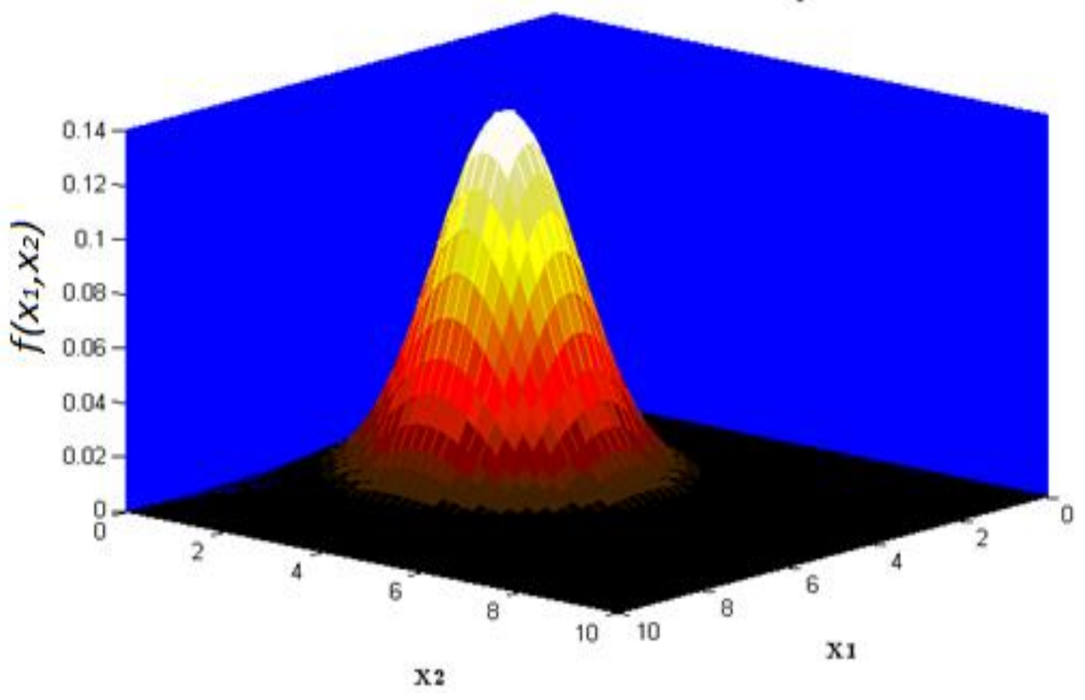


Figure 6: Shows a convex vector search space with only one optimal solution.

I. Local Maxima:

The greedy selection of the HC algorithm while choosing its neighboring vertex solution in general does not lead to the optimal solution for problems where the search space is non-convex. In other words, HC locally iterates in to better solutions in a greedy manner. In a non-convex space, HC may iterate to a vertex from where it does not detect any neighboring candidate solutions which are better than its current state while better solutions or a global maximum exists somewhere else in the search space. So, the HC algorithm gets trapped in local maxima which is not the optimal solution and potentially not even an acceptable local maxima.

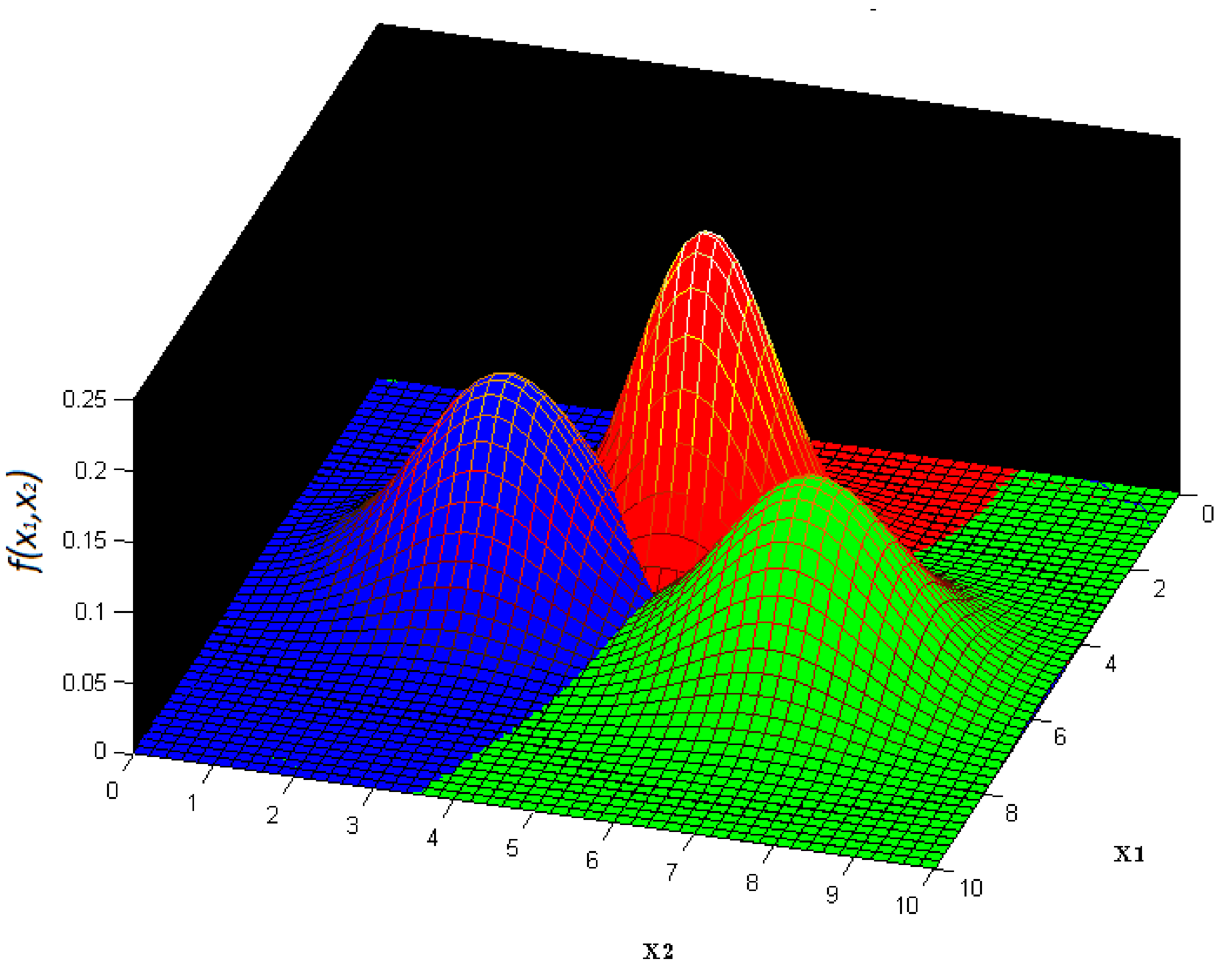


Figure 7: Demonstrates a non-convex search space with a local maxima or a suboptimal solution on a non-convex search space. Depending on the initial iteration point, HC often gets trapped in the suboptimal solution.

In a discrete search space tree of nodes connected with the edges of a graph, the HC algorithm would search for a solution at a neighboring state or node for a candidate new solution. The algorithm greedily accepts the new vertex as long as the candidate solution is better (to maximize) or worse (to minimize) than the current state or vertex.

The HC algorithm is non-stochastic greedy search that unconditionally accepts a better neighboring state or solution than its current state without exploring any other neighboring nodes. This is why this algorithm is ideal for convex search spaces where it will not be trapped in a suboptimal solution. For the proposed stochastic vSA and mSA algorithms, HC is a good baseline algorithm for comparison in a maze navigation experimental test as it only exploits and never explores.

CHAPTER 4

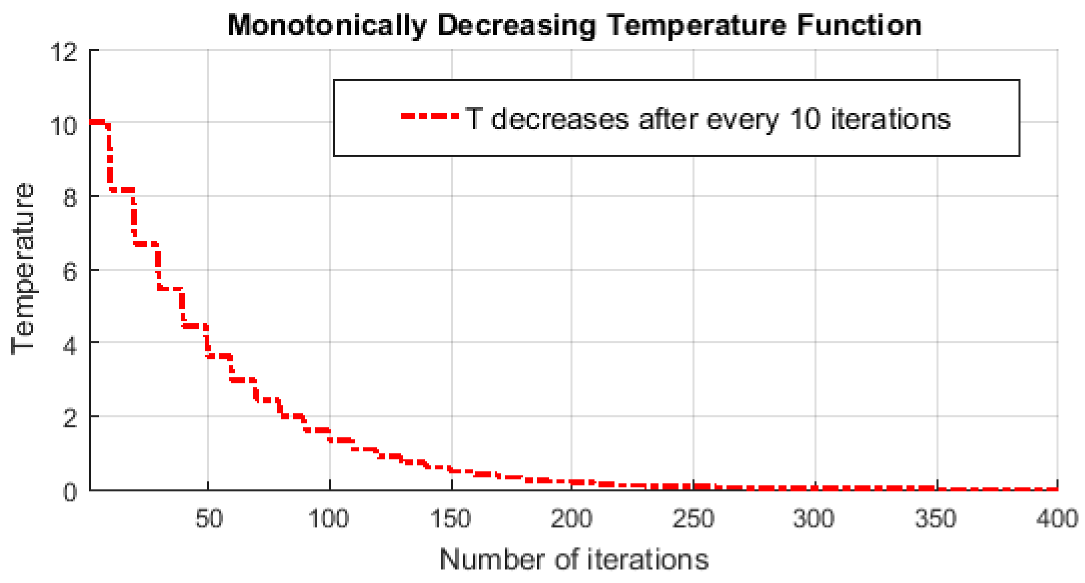
VANILLA SIMULATED ANNEALING

I. Introduction:

A conventional tree search approach among ghost agents in the game of Ms. Pac-Man includes path planning and calculating next best nodes/states only at T-shaped, L-shaped or crossroad junctions. In this thesis, the introduction of a new heuristic path planning search strategy called memoryless path planning of ghost agents was introduced in an earlier chapter 2. In this path planning strategy, ghost agents calculate their best move (or best node) at every unit distance they travel without waiting to come across a junction. Using heuristics of both traditional tree search of nodes at T-shaped, L-shaped or crossroad junctions in conjunction with heuristics of non-traditional tree search or the memoryless heuristic generates neighboring nodes or candidate solutions to calculate best nodes for every unit distance the ghost travels. Here, a probabilistic optimization search algorithm called simulated annealing is developed for ghost agents in the game of Ms. Pac-Man to select their destination nodes or states to maneuver through the maze in order to converge to their adversary. The underlying search algorithm is denoted vanilla Simulated Annealing (vSA). The vSA algorithm is explored as an extension of the HC algorithm. In the vSA algorithm, an exploration phase is followed by an exploitation phase thereby extending HC which is limited to exploitation.

II. Vanilla Simulated Annealing:

Simulated annealing is a stochastic optimization technique to search for global maxima in a non-convex discrete space, where it iterates through exploration (in the initial phase of algorithm's progression) and exploitation (in the later phase). Initially, conventional simulated annealing explores worse neighboring candidate solutions in order to thoroughly explore the local tree graph. This phase of algorithm iteration is called exploration. After thoroughly searching the sample space, it gradually stops exploring worse neighboring candidate solutions and greedily accepts better neighboring candidate solutions with a probability that approaches 1. This phase of algorithm iteration is called exploitation. It is a metaheuristic optimization method to find an approximate solution in large non-convex search space. A metaheuristic algorithm is a higher level heuristic used to obtain an approximate solution in a stochastic optimization method. A metaheuristic algorithm may incorporate properties of exploration and exploitation during the search process (a trade-off between randomization and local search). The probability distribution associated with selecting poorer quality solutions within traditional simulated annealing algorithm is defined by the Boltzmann function where probability of acceptance (Pr_A) depends on a global control variable called temperature as demonstrated in figure 7. The temperature, in turn, typically depends on the number of iterations the algorithm has performed. Heuristics generate candidate neighboring nodes or vertices where possible moves by ghost agents are made to reach their destination. The algorithm accepts a worse candidate solution or a negative energy valued nodes with a probability based on the Boltzmann's probability distribution (Pr_A) which depends on the temperature. Acceptance of worse nodes with a probability in simulated annealing algorithm makes it somewhat immune to being trapped in less acceptable local optima relative to HC. As the algorithm iteration progresses, the temperature is reduced (e.g. typically exponentially). As the temperature approaches 0, simulated annealing acts more like a deterministic greedy search algorithm where it only accepts better candidate solutions, or a higher energy valued neighboring node, and rejects a worse candidate solution with a high probability [18][19][20][21].



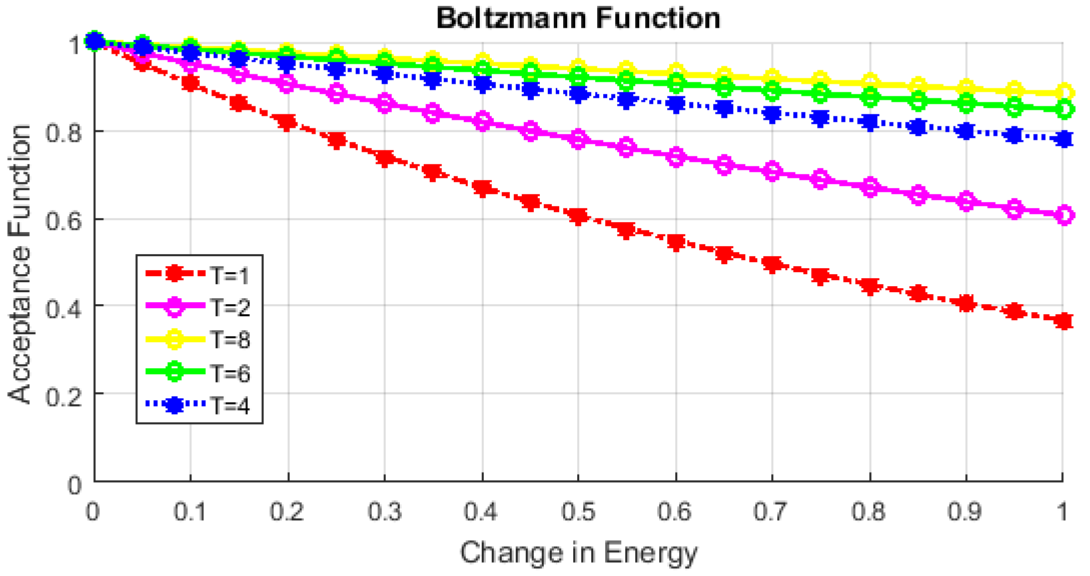


Figure 8: Shows monotonically decreasing temperature function (top) and its corresponding acceptance probability that is defined by the Boltzmann function (bottom). The change in energy or difference between the current node and a neighboring candidate node is always 1.

In figure 7, Temperature is a monotonically decreasing function,

$T = T_0 \times (0.9800)^{iteration}$,

where T_0 is initial temperature and temperature, T decreases after every 10 iterations in vSA

Its corresponding Boltzmann's probability distribution function is,

Acceptance Probability, $Pr_A(N^*, N, T) = \exp\left(\frac{\Delta E}{Temperature}\right)$,

Or, Acceptance Probability, $Pr_A(N^*, N, T) = \exp\left(\frac{N^* - N}{Temperature}\right)$,

where N^* is the Manhattan distance from Current state to the Global Solution, N is the Manhattan distance from Neighboring Candidate state to Global Solution, and T is Temperature. Difference between the current state to a candidate solution state, ie. $|N^* - N|$ is always 1,

The pseudocode for vSA can be expanded and written as:

```
Algorithm 2: Vanilla Simulated Annealing
1:   Initialize iteration ← 0
2:   Initialize Initial Temperature, $T_0$ = 10
3:   while Manhattan distance $> 0$ and$\Delta E < 0$ do
4:     $iteration \leftarrow iteration + 1$
5:     if $iteration$ mod 10 == 0
6:       Temperature ← $T_0 \times (0.98)^{iteration}$
7:     end
8:     if $\nabla E \in \{-1,1\} > 0$ then
9:       Accept node greedily
10:    else $\nabla E \in \{-1,1\} < 0$ then
11:      Calculate Boltzmann Distribution, $Pr_A(N^*, N, T) = exp\,(\Delta E\ /\ Temperature)$
12:      if $Pr_A(N^*, N, T)$ > Random_Numb_Gen then
13:        Accept node
14:      else
15:        Reject Node
16:      end
17:    end
18:  Until Manhattan distance is 0.
```

III. <u>Vanilla simulated annealing during high the temperature phase:</u>

Traditional SA consists of a mathematical model where the algorithm that starts at very high temperature and cools down after a sufficient number of iterations as shown in figure 7. Figure 7 (bottom) shows probability distribution of Boltzmann's acceptance function as its shape changes along with temperature. When the temperature variable is high, the shape of probability function $Pr_A(N^*, N, T)$ is almost a straight line regardless of change in energy of nodes (ΔE). In this phase of high temperature, Boltzmann's function has a high probability of acceptance Pr_A value regardless of change in energy value of its surrounding nodes. At high temperature, SA ignores differences in the energy between the current state or node(N^*) with destination state or node(N). In other words, SA does not take into account a change in energy (ΔE) between the two nodes. This makes the search algorithm choose its nodes in a random manner where it does not differentiate between a better state or positive energy valued nodes and worse state or negative energy valued with respect to its current state. Therefore, SA initially goes through an exploration phase where it searches the space thoroughly (in its adjacent neighbourhood) regardless of positive or negative energy value of adjacent nodes which encourages the algorithm to only explore its surrounding space while the temperature is high.

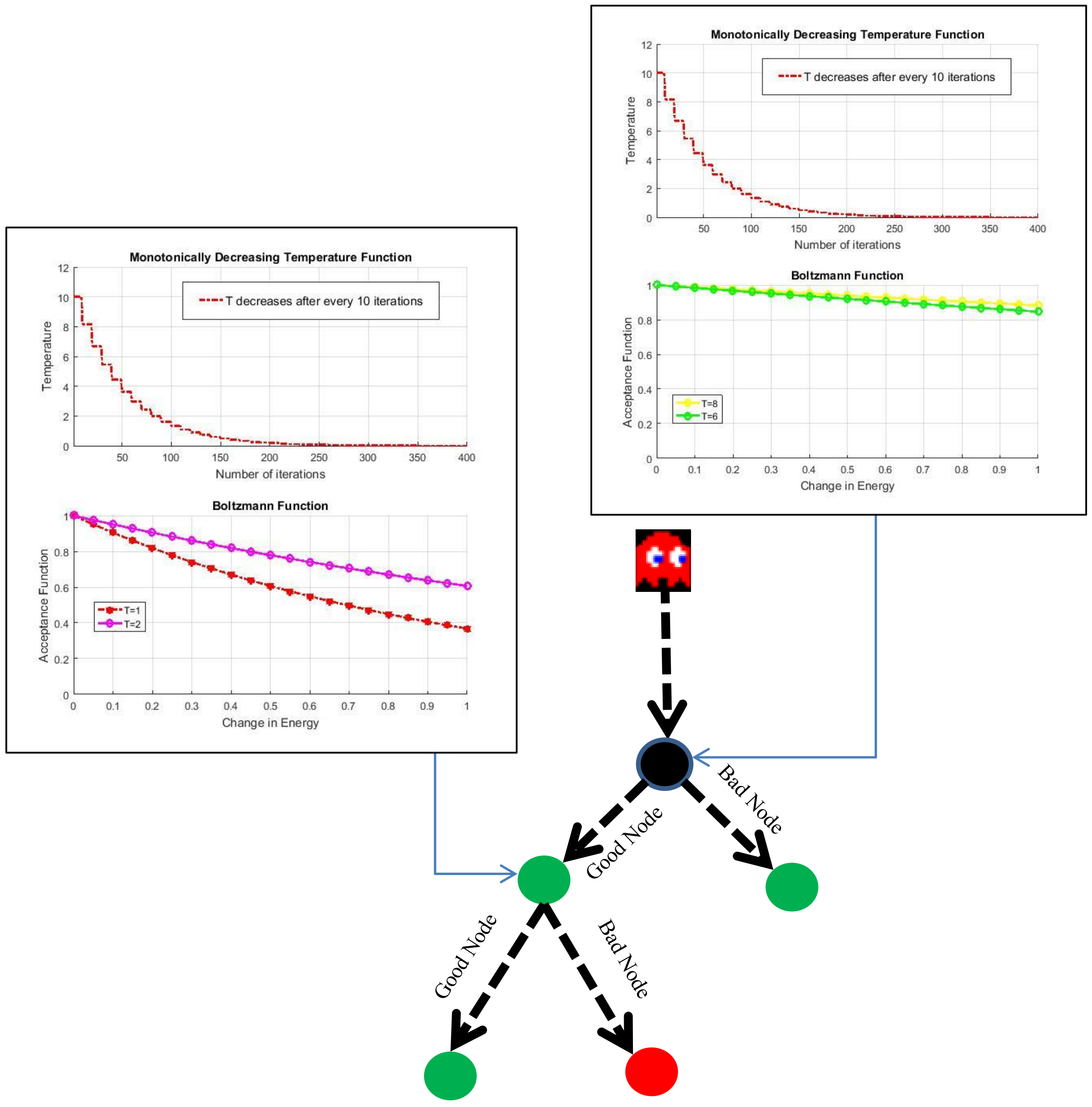


Figure 9: Demonstration of acceptance probabilities at respective nodes. Acceptance probability, Pr_A of a bad node is high when temperature, T is high (right graph) but, the acceptance probability, Pr_A of a bad node is low when temperature, T is low (left graph).

IV. Vanilla simulated annealing during low temperature phase:

The acceptance probability $Pr_A(N^*, N, T)$ eventually changes with respect to ΔE as the temperature decreases exponentially as shown in figure 7 (top). When the temperature is low, the acceptance probability $Pr_A(N^*, N, T)$ changes with respect to change in energy (ΔE) between the current state (N^*) and the neighboring candidate state or node (N). This plays a critical role in the search process as the algorithm would have a lower acceptance probability (Pr_A) of choosing a worse state or node (N) value than its current node (N^*). The candidate node (N) or solution that has the lower energy than the current node (N^*) will have very low probability of being accepted $Pr_A(N^*, N, T)$ as the next state or next node. This makes the algorithm act more like a greedy algorithm as the temperature drops, where it readily exploits the better solution or the positive energy node but avoids a node with negative energy value or a worse solution with a high probability. At low temperature, the probability of accepting a worse node is almost equal to 0. Therefore, SA begins with the exploration phase and ends with an exploitation phase. This makes SA unable to exploit better solutions unless it explores all possible solutions or states in its initial search phase at least for the maze traversals of ghost agents within Ms. Pac-Man. This approach of reducing the temperature exponentially works well for problems such as the TSP, placement, or graph partitioning but may not be ideal for developing behaviours or strategies of maze traversals for ghost agents with an objective to display collective behaviours of flanking and blocking.

Although unconventional, an idea explored here was reversing the temperature to increase exponentially from low to high. This appeared to work better in a Ms. Pac-man game environment as ghost agent using this algorithm converged to its target more efficiently (within fewer iterations) than traditional vSA where the temperature decreases exponentially as the algorithm iteration occurs as shown in chapter 5.

CHAPTER 5

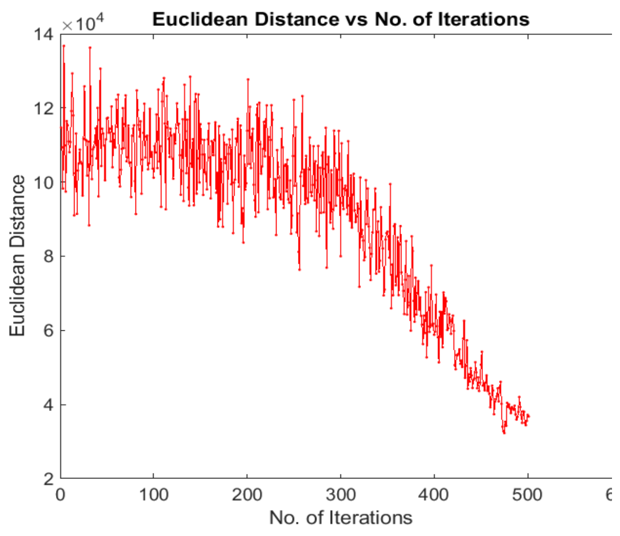
PROPOSED MODEL: MODIFIED SIMULATED ANNEALING

This chapter introduces an algorithm that was motivated by the ideas associated with varying exploitation and exploration phases of search. As it is motived by simulated annealing a brief discussion will include a more familiar problem domain for algorithms like SA.

I. Travelling Salesman Problem:

Travelling salesman problem (TSP) is a NP-complete problem or a class of decision problems that falls under both the non-deterministic polynomial time problems (NP) and the non-deterministic polynomial time hard problems (NP-hard) used for benchmarking combinatorial optimization algorithms [27][28]. Combinatorial optimization algorithms search for a global solution out of an objective function that has a large, discrete solution space. TSP problem is defined by a connection of nodes or vertices are connected with edges where each edge has a cost function (typically distance). The goal or the global solution is to find the least cost tour by visiting all the nodes without any loop or travelling the same node twice. A non-deterministic polynomial time is a problem where no fixed heuristic is known to solve the problem in a non-exponential time complexity.

Application of vanilla Simulated Annealing to the travelling salesman problem - where the visited nodes are randomly swapped to define a new neighboring solution – gives good results as shown in the figure 9 below. The temperature control parameter of vSA is reduced exponentially with increasing number of algorithm iterations. The vSA algorithm accepts worse neighboring solutions at the beginning of iterations (as the temperature is high) that creates the exploration phase, but the vSA algorithm stops accepting worse solutions as the vSA algorithm progresses with increasing number of iterations (as the temperature reduces) which creates the exploitation phase in the later stages of vSA algorithm's iteration.



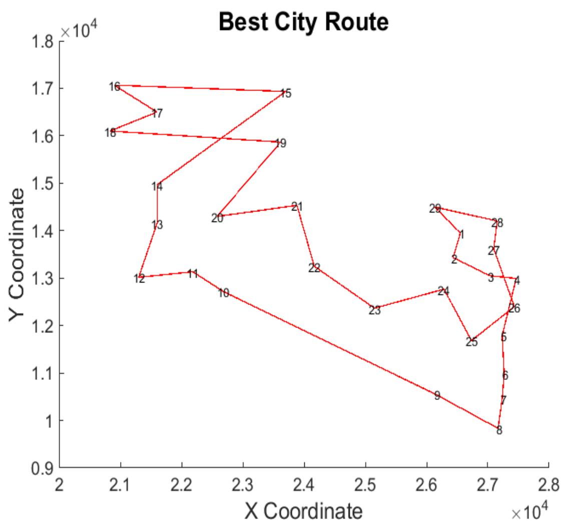


Figure 10: Demonstration of vSA in the travelling salesman problem where length of the tour reduces as the vSA algorithm starts from exploration at high temperature and ends at exploitation at low temperature and provides a good solution (left). This figure shows the most efficient tour generated by the vSA algorithm (right).

To better understand exploration and exploitation, the temperature parameter in vSA was reversed, i.e. as the vSA algorithm progresses the temperature was increased exponentially as shown in figure 10. As expected, applying the algorithm in travelling salesman problem gives poor results as compared to the previous scenario where the temperature reduces exponentially as the vSA algorithm iteration progresses as demonstrated in figure 11 (left).

In this vSA algorithm where the temperature increases exponentially instead of being reduced exponentially as the algorithm progresses shown in figure 10 below, the algorithm does not accept worse neighboring solutions in the beginning of the iteration marking its exploitation phase. This makes the vSA algorithm accept only better neighboring solutions and reject worse neighboring solutions during the beginning of the algorithm's iterations.

As the algorithm progresses, the temperature increases exponentially after sufficient number of iterations making the algorithm choose worse neighboring candidate solutions to marking its exploration phase in later stages of the algorithm's iterations. This exploration phase at the end of the algorithm iteration gives poor results of route selection as shown in figure 11 (right) below.

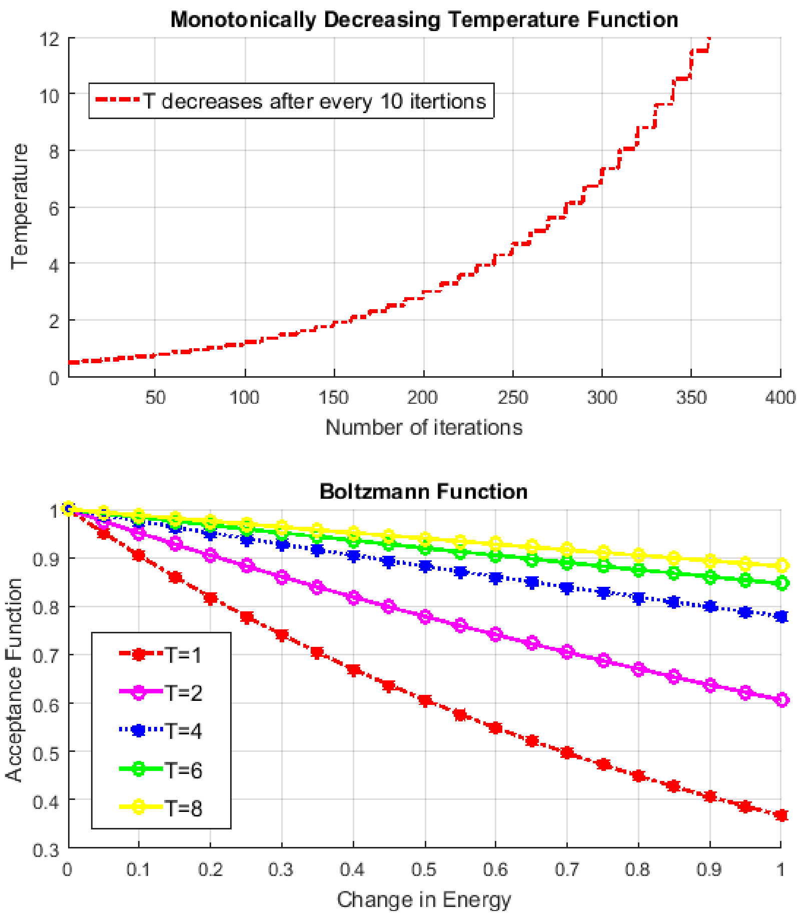


Figure 11: Demonstration of a monotonically increasing temperature function (top) and its corresponding acceptance probability that is defined by the Boltzmann function (bottom). The difference between a current node and its neighboring candidate node is always 1.

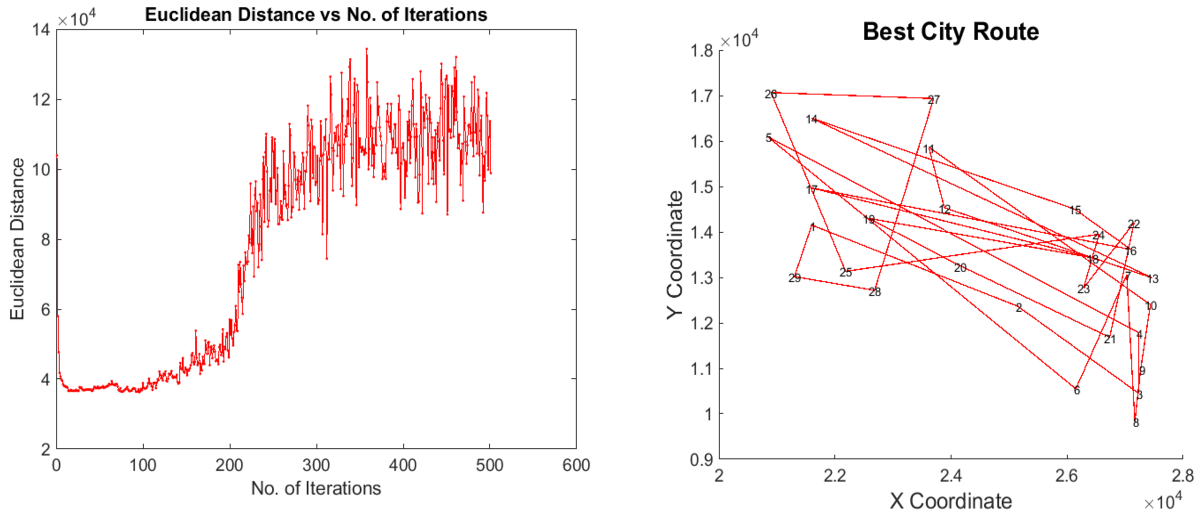


Figure 12: Demonstrates use of vSA where temperature increases exponentially as the algorithm iterates in this travelling salesman problem and the length of tour increases as the vSA algorithm starts from exploitation phase at low temperature and ends at exploration phase at high temperature (left). This figure shows the last accepted solution taken by vSA algorithm with increasing temperature (right).

In the game environment of Ms. Pac-man, both the above mentioned vSA algorithm (ie. vSA with temperature increasing exponentially and vSA with temperature reducing exponentially as the algorithm progresses) is tested with that of the proposed mSA where acceptance probability depends on Manhattan distance and the results are shown below in figure 12 (mSA is discussed in more detail subsequently). In order to properly compare, all the above mentioned algorithms are using heuristics that create neighboring candidate nodes randomly. In figure 12, reversing the temperature in the vSA proved to give better results in comparison to conventional vSA as the ghost iterates towards its enemy more efficiently within fewer iterations in contrast to conventional vSA where it takes more time to reach its adversary. But as the iteration progresses, the reverse vSA algorithm diverge away from the solution as the temperature increases exponentially. The mSA algorithm is dictated by an adaptive sigmoid function reaches close to the solution, due to its exploitation phase in the beginning (not in a strictly greedy fashion), in fewer iterations compared to reverse vSA. Then, mSA algorithm switches to an exploration phase as the mSA algorithm comes closer to its solution that enables the algorithm to stay close to its solution. Unlike reverse vSA algorithm which diverges away as the algorithm iteration progresses, the mSA algorithm does not diverge away from solution as the mSA algorithm uses an adaptive acceptance probability that switches to exploitation phase as Manhattan distance becomes higher or as the mSA algorithm starts to diverge.

Observations of vSA and reverse SA within Ms. Pac-Man lead to developing an algorithm that could take advantage of the qualities of exploration and exploitation. This provided a good reason as to consider the use of an adaptive acceptance probability based on a sigmoid function that depends on the Manhattan distance between the ghost agent and Ms. Pac-Man. In this manner mSA was developed that begins from exploitation during high distance and switches to an exploration phase during low distance similar to reverse vSA that has a non-adaptive temperature that starts from low to high which provides better results in convergence in comparison to conventional vSA. Unlike vSA, the mSA algorithm developed here does not remain exploitive in its later stages but is adaptive, with its probability of accepting nodes being exploitive or explorative depending upon the Manhattan distance measure.

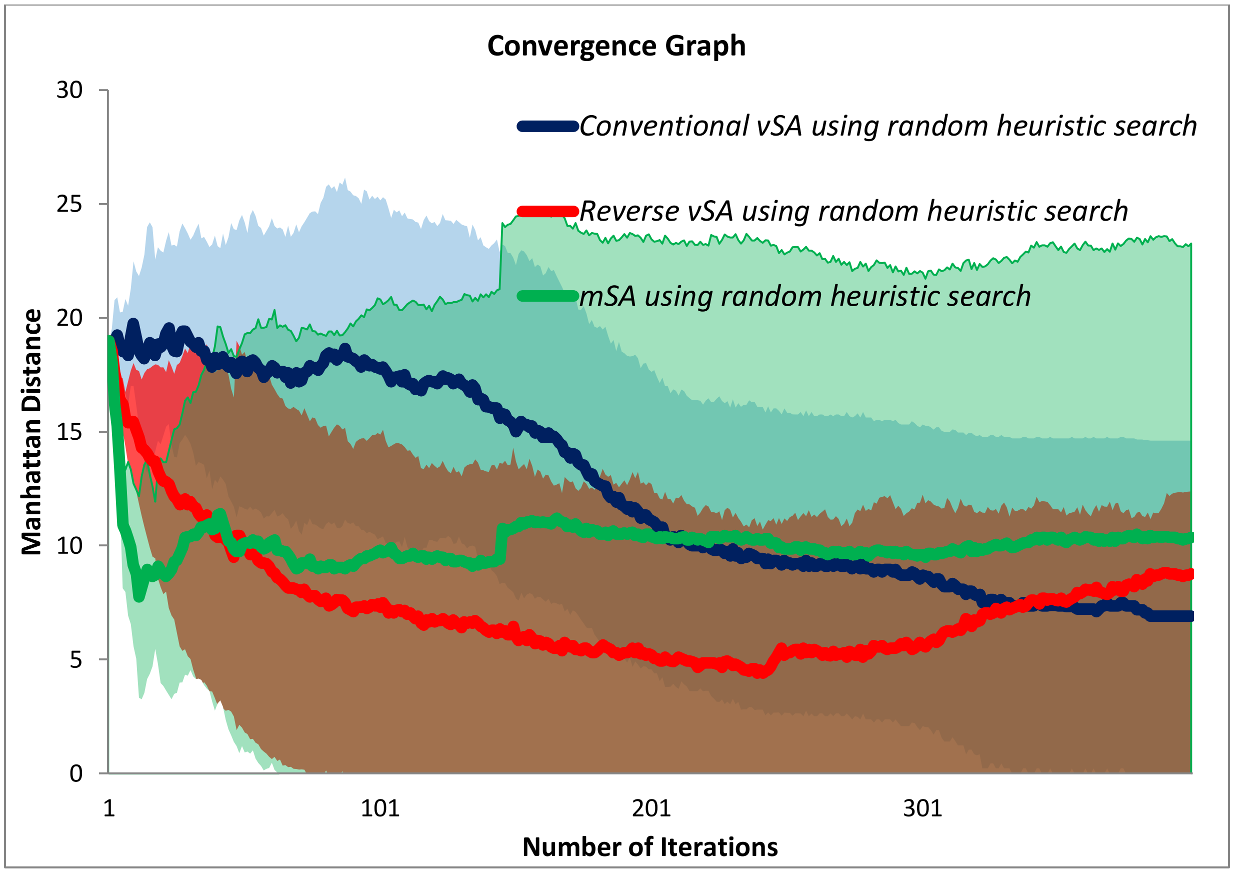


Figure 13: Convergence rate shows the convergence of vanilla SA with exponentially reducing temperature (blue) and vanilla SA with exponentially increasing temperature (red) as they navigate through maze environment to a stationary adversary.

This insight into a trade-off between exploitation and exploration is further developed or explored through a new approach to probabilistic acceptance that could continuously explore or exploit as the situation warranted.

Contrary to conventional vSA or reverse vSA, mSA accepts a better solution when $\Delta E > 0$ with a probability $Pr_A(N^*)$ depending on random number generator throughout its search starting from high Manhattan distance from ghost to Ms. Pac-man. As ghost agent converges to its adversary. As value of node N^* reduces (node $N^*$is heuristically defined by Manhattan distance between ghost agent and Ms. Pac-Man), temperature of the mSA algorithm can be thought of as increasing exponentially in Ms. Pac-man game environment.

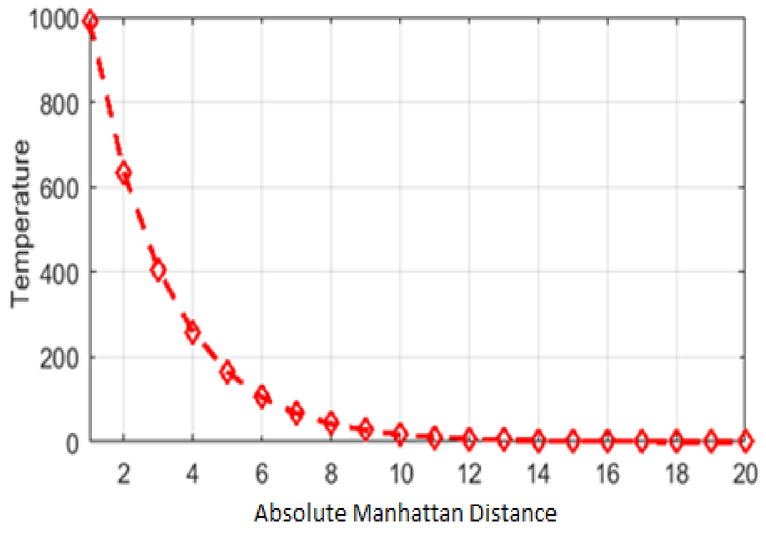
To accomplish this, mSA makes use of a sigmoid function as an acceptance probability that changes with respect to the Manhattan distance, N^*. For the discussion here, N^* may either refer to the node N^* or its distance to its adversary. This absolute distance measure is better suited for this particular problem scenario instead of using a Boltzmann function that depended on value of $N^* - N$. This idea of using a single parameter N^* instead of using two parameters $N^* - N$ is better suited for this particular simulation scenario where value of $|N^* - N|$ is always 1.

II. Proposed Model: Modified Simulated Annealing:

Traditional simulated annealing algorithm has a probability of acceptance that depends on the change in energy ΔE or Manhattan distance between the current state and its neighboring candidate state, $N^* - N$. In the game of Ms. Pac-man, the difference in the Manhattan distance (referred to as difference in Energy, ΔE) between energy of current state of ghost agent (N^*) to the neighboring candidate state or node (N) is either a positive or negative unit distance, ie. $\Delta E \in \{-1, 1\}$. Since ΔE is always constant for this maze navigation problem, only the Manhattan distance (N^*) parameter determines the acceptance probability of a neighboring candidate solution node. A modified Simulated Annealing (mSA) is used for this problem scenario where acceptance probability depends only on Manhattan distance of current state to its adversary, i.e. N^*, in contrast to traditional simulated annealing where the change in Energy, $\Delta E = N^* - N$ between two states is used to calculate acceptance probability.

Moreover, (for both vSA and mSA) the difference between the current node and neighboring candidate node is constant. The proposed mSA only sees the candidate solution's feature of positive ($\Delta E > 0$) or negative value ($\Delta E < 0$) and Manhattan distance from the agent to its adversary (N^*) in order to calculate acceptance probability, Pr_A. As briefly mentioned above, another modification investigated here is to use the sigmoid function as an acceptance function. This is another novel feature of the proposed mSA used here. Therefore, the acceptance probability of mSA is defined by a sigmoid function where the probability of acceptance shown in figure 13 is $Pr_A(N^*)$ when $\Delta E > 0$ and probability of acceptance is $1 - Pr_A(N^*)$ when $\Delta E < 0$. Although rather radical departures from traditional SA, these modifications in mSA maintain the notions of acceptance probabilities and notions of exploration and exploitation phases. As the mSA is described, another fairly radical departure is noted and used to help justify the mSA proposed here.

Finally, ghost agents using mSA algorithm locally interact with each other by maintaining a separation of a single unit distance that further helps in achieving the objective of this thesis: to show flanking and blockade maneuver that allows bots to portray a flanking maneuver and allows other bots to restrict its enemy's movement. Separation of ghosts ensures that they block at different nodes surrounding Ms. Pac-Man (instead of the possibility of sticking together and blocking a single node or junction) and at the same time increasing the probability of ghosts to mount a flanking maneuver. The use of separation among ghost agents to achieve their overall objective of flanking and blocking is discussed in detail in chapter 7, section II.



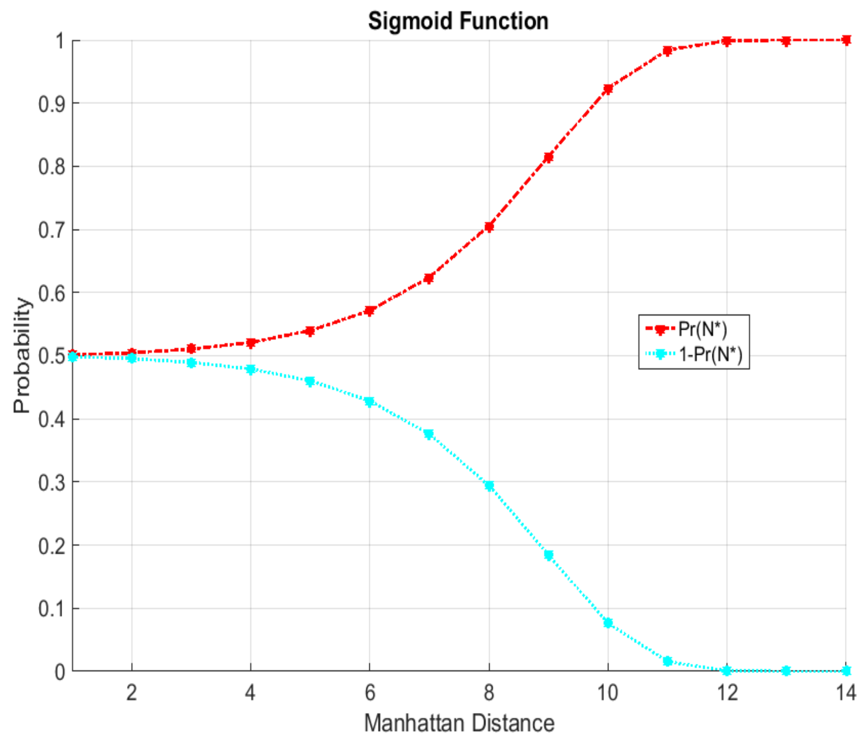


Figure 14: Shows a monotonically decreasing temperature function (top) and its corresponding acceptance probability function of its neighboring candidate node which is defined by a sigmoid function (which varies with Manhattan distance between ghost agent and its adversary) (bottom).

In figure 13, Temperature is a monotonically decreasing function, where

$T(N^*) = T_0(0.98)^{2^{N^*}},$

where T_0 is initial temperature and N^* is a heuristic function defined by Manhatan distance from ghost to Ms. Pac-Man, their corresponding Probability Distribution of Sigmoid function and,

$Acceptance\ Probability, Pr_A(N^*) = \{1 + exp\,(-6 \times N^* / Temperature)\}^{-1}$;

where N^* is a heuristic function defined by Manhatan distance from ghost to Ms. Pac-Man, their corresponding Temperature depends on Manhattan distance from ghost to Ms. Pac-man.

The pseudocode of the above mentioned algorithm can be written as:

Algorithm 3: Modified Simulated Annealing

```
1:   Initialize Temperature = 1000
2:   while Manhattan distance $> 0$ do
3:     Calculate $Acceptance\ Probability, Pr_A(N^*) = \{1 + exp\,(-6 \times N^* / T)\}^{-1}$
4:     if $\Delta E > 0$ then
5:       if $Pr_A(N^*)$ > Random_Number_Generator then
6:         Accept node
7:       else
8:         Reject Node
9:       end
10:    else $\Delta E < 0$ then
11:      if $1 - Pr_A(N^*)$ > Random_Number_Generator then
12:        Accept node
13:      else
14:        Reject Node
15:      end
16:    end
18:  Until Manhattan Distance is 0.
```

i. *Modified simulated annealing at low temperature during high Manhattan distance:*

The mSA technique developed here as ghost agents search for their adversary, mSA goes through iterations that start from exploitation phase with an acceptance function $1 - Pr_A(N^*)$ when $\Delta E < 0$, which provides very low probability of acceptance Pr_A of accepting worse candidate solutions or negative energy valued nodes(N^*) in the initial phase of algorithm while the Manhattan distance is high.

During low temperature at high Manhattan distance, acceptance function $Pr_A(N^*)$ when $\Delta E > 0$ has high probability of acceptance of a better candidate solution while the acceptance function of accepting a worse candidate solution, $1 - Pr_A(N^*)$ when $\Delta E < 0$, gives the algorithm a probability of close to 0 that marks the exploitation phase shown in figure 14 (right).

ii. *<u>Modified simulated annealing during high temperature at low Manhattan distance:</u>*

The modified simulated annealing algorithm reaches a high temperature as the Manhattan distance between ghost agents and Ms. Pac-Man reduces in order for ghost agent to converge to its global maxima. During this phase of high temperature at low Manhattan distance, the acceptance function when it sees a better candidate solution (ie. when $\Delta E > 0$) is $Pr_A(N^*) = 0.5$ and acceptance function when it sees a worse candidate solution (ie. when $\Delta E < 0$) is $1 - Pr_A(N^*) = 0.5$ which makes mSA choose a neighboring vertex or a node regardless of value of N^*which marks the algorithm's exploration phase demonstrated in figure 14 (left). In mSA, if the analogy to temperature continued, the temperature is inversely proportional to the Manhattan distance between ghost agent and its adversary.

As described above, the mSA algorithm switches to an exploration phase later as the Manhattan distance is reduced. Therefore, if the Manhattan distance becomes very low and consequently increasing mSA algorithm temperature, sigmoid function $Pr_A(N^*)$ when $\Delta E > 0$ and sigmoid function $1 - Pr_A(N^*)$ when $\Delta E < 0$ becomes equal at 0.5 that makes mSA algorithm to explore among good $(\Delta E > 0)$ neighboring candidate solutions as well as among bad $(\Delta E < 0)$ neighboring candidate solutions space.

The adaptive acceptance function is defined by the sigmoid function where its threshold changes with changes in the Manhattan distance between a ghost agent and Ms. Pac-Man (N^*). This change in threshold of the sigmoid function that depends in the parameter N^* makes a ghost agent exploit neighboring candidate nodes when Manhattan distance is high and to explore neighboring candidate nodes to explore when Manhattan distance is low. The mSA should not be confused with simply running SA from a low temperature to a high as would seem to be the case as they go through opposite phases of exploration and exploitation. As an illustration, the TSP problem was used to demonstrate that although possible, in general the solution would as expected become considerably worse if that were the case. A simple example in this chapter illustrates with results as expected are shown in figure 9, 10 and 11. In figure 9, conventional SA referred in this thesis as vSA is applied in travelling salesman problem that gives optimal results in terms of optimal route selection. Figure 10 demonstrates the variation of temperature of reverse vSA with and its change in Boltzmann function with number of iterations and figure 11 demonstrates the application of reverse vSA in travelling salesman problem that gives poor results of optimal route selection. Figure 12 shows the use of vSA, vSA in reverse. and mSA in the game environment of Ms. Pac-Man.

The mSA algorithm for this problem is designed for exploitation during initial iterations at high Manhattan distance (but not in a strictly greedy fashion) and then followed by exploration. The probability of acceptance defined by a sigmoid function for mSA is shown in figure 13 (bottom). The reasoning for this behaviour of exploitation followed by exploration is as follows; when the distance is far (N^* high), the probability of accepting a better state is also high and the probability of accepting a worse state is low; when the distance is close, then there is a trade-off between exploration and exploitation where the probability of acceptance of better as well as worse neighboring nodes are equal and converges to ½. This variation prevents the algorithm from ever really getting trapped, nor having a ghost wander off. If the distance were to increase, there would be a concomitant increase in exploitation. Therefore, mSA algorithm allows a dynamic balance between exploration and exploitation.

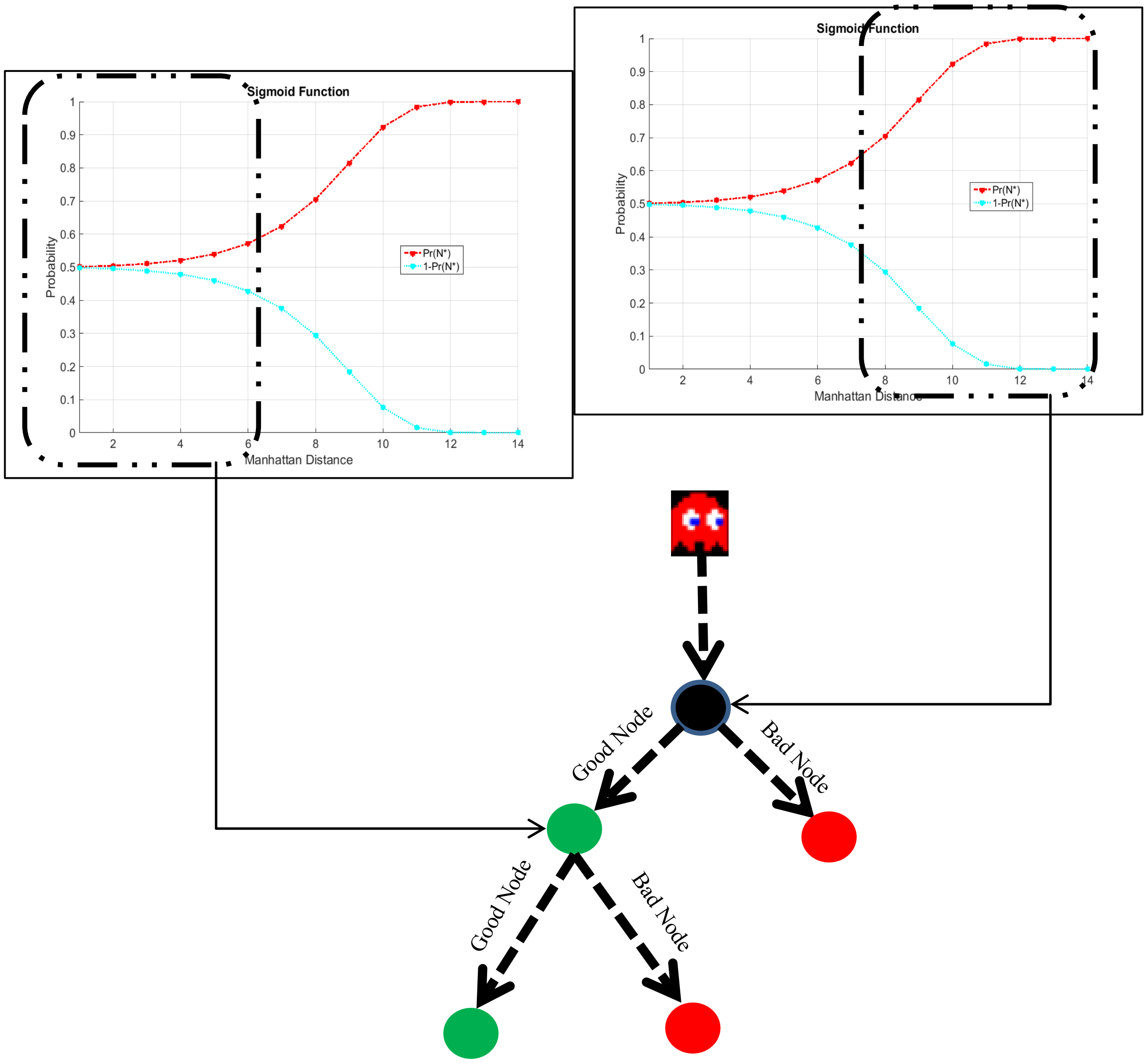


Figure 15: Demonstration of acceptance function at respective nodes. During high Manhattan distance N^* when $\Delta E > 0$ acceptance function becomes $Pr_A(N^*) \approx 1$, and when $\Delta E < 0$, acceptance function becomes $1 - Pr_A(N^*) \approx 0$ that marks exploitation phase (right graph). During low Manhattan distance N^* when $\Delta E > 0$, acceptance function becomes $Pr_A(N^*) \approx 0.5$, and when $\Delta E < 0$, acceptance function becomes $1 - Pr_A(N^*) \approx 0.5$ that marks exploration phase among good and bad neighboring candidate solution nodes (left graph).

III. Differences between mSA and vSA:

1. The traditional simulated annealing algorithm (vSA) has a cooling rate that varies with respect to number of iterations as shown in figure 7, in contrast to the mSA algorithm whose cooling rate depends on Manhattan distance from the ghost agent to Ms. Pac-Man as demonstrated in figure 13. In mSA, the temperature parameter has an exponential relation with the Manhattan distance between ghost agent and its adversary, in contrast to vSA where temperature depends on number of iterations as algorithm iteration progresses. This is perhaps not the best description but included here to make reference to vSA.

2. When the temperature is low and its related Manhattan distance is high, the sigmoid function that defines the acceptance function of mSA has a very high acceptance threshold value as shown in figure 13. At this point, probability of acceptance $\mathrm{Pr}_{\mathrm{A}}(\mathrm{N}^*)$ when the mSA algorithm sees a neighboring positive node ($\Delta E > 0$) or a better candidate solution, the acceptance probability becomes $Pr_A(N^*) = 1$. In contrast to the situation when it sees a neighboring negative node ($\Delta E < 0$) or a worse candidate solution, its acceptance probability is $1 - Pr_A(N^*) = 0$. This creates an exploitation strategy where algorithm has a high chance of greedily accepting a good node and a high chance of rejecting a bad node.

3. When the temperature is high and its related Manhattan distance low, the sigmoid function has a low threshold probability of acceptance of 0.5 shown in figure 13. At this stage, the probability of acceptance becomes $\mathrm{Pr}_{\mathrm{A}}(\mathrm{N}^*) = 0.5$ when algorithm sees a neighboring

positive node or a better candidate solution ($\Delta E > 0$), and the probability of acceptance becomes $1 - Pr_A(N^*) = 0.5$ when algorithm sees a neighboring negative node or a worse candidate solution ($\Delta E < 0$). This creates an exploration strategy where algorithm has an equal chance of accepting or rejecting a good node and an equal chance of accepting or rejecting a bad node demonstrated in figure 13.

4. The acceptance function of traditional simulated annealing depends on the difference energy, $\Delta E = |N^* - N|$ while the acceptance function for mSA depend only on N^*. Since the difference between the current node and a neighboring candidate node is always constant, ie. $|N^* - N| = 1$, the mSA algorithm sees candidate solution being positive ($\Delta E > 0$) or negative ($\Delta E < 0$) and only uses the Manhattan distance (N^*) from agent to its adversary to calculate its acceptance probability(Pr_A), i.e. when $\Delta E > 0$, acceptance function becomes $Pr_A(N^*)$ but when ($\Delta E < 0$), acceptance function becomes $1 - Pr_A(N^*)$ demonstrated in algorithm 3.

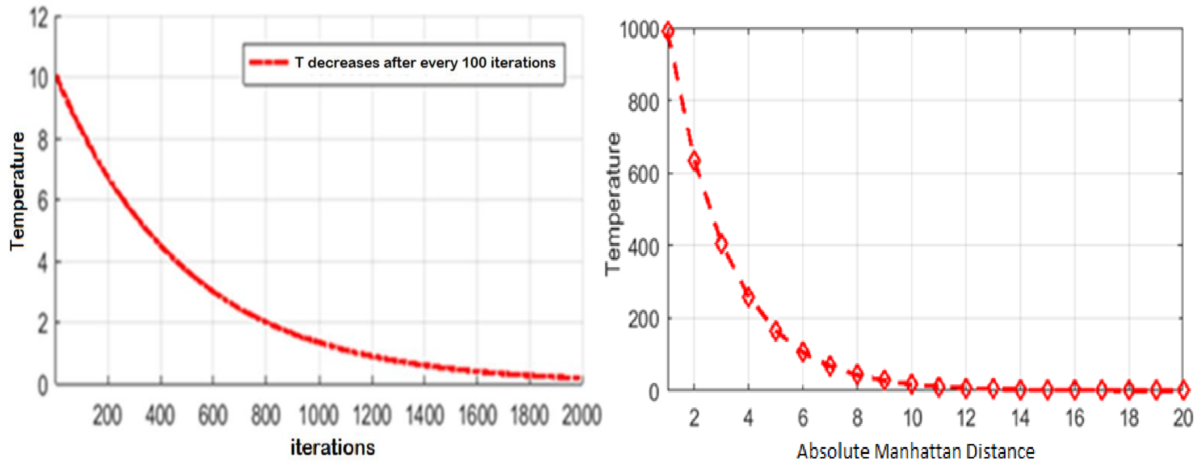


Figure 16: Demonstrates a comparison of temperature rate that decreases with number of iterations in the case of vSA which is in contrast to the temperature rate of mSA algorithm that decreases with respect to Manhattan distance.

5. The mSA algorithm explores among better nodes with acceptance probability $Pr_A(N^*)$ as well as worse nodes with acceptance probability $Pr_A(N^*)$ in contrast to the traditional simulated annealing algorithm where it exploits any better candidate solution with probability of 1 but explores only within worse nodes with an acceptance probability $Pr_A(N^*)$.

6. In other words, the modified simulated annealing algorithm accepts better neighboring candidate solutions as well as worse neighboring candidate solutions with a probability, in contrast to traditional simulated annealing where it greedily accepts a better candidate or vertex without exploring any other nodes. Exploration among better candidate solutions leads to a more robust search in the optimization process at least for the objectives of NPCs.

For evaluation of performance in the simulator used in this thesis, which enables ghost agents to calculate the shortest path at every unit distance they travel without having to wait to come across a junction, the proposed mSA algorithm is compared with a traditional vSA algorithm using a ghost agent that maneuvers through the maze to converge on Ms. Pac-Man both of which use heuristic search referred as memoryless and memory based. Furthermore, the proposed mSA algorithm is compared with the HC algorithm (a deterministic greedy search algorithm) using the same heuristics for node generation as the mSA and vSA in order to compare the convergence efficiency in navigating through maze (results shown in chapter 7). Comparisons of these algorithms are demonstrated at different scenarios where ghost adversary is located at different parts of the maze as shown in figures 17, 18 and 19 in chapter 7.

HC algorithm is a deterministic greedy search algorithm used by a ghost agent that readily accepts a potentially better solution or node without exploring any other neighboring node/vertices or potential candidate solutions while rejecting any worse candidate nodes with a probability of 1 while vSA algorithm starts from high temperature and ends at low temperature as algorithm iteration progresses. These two algorithms make them good baseline algorithms from which to assess mSA.

CHAPTER 6

ANT COLONY OPTIMIZATION

This section is included a review of other algorithmic techniques that attempt to improve NPC behaviours The reason for placing this section here is to present research of past algorithm that attempts to improve NPC behaviours. This particular example is one that attempts to utilize Swarm or Ant Colony analogs to replicate artificially intelligent behaviours. The ACO discussion is included here to illustrate the development of algorithms by others that have similar objectives of improving NPC behavious. ACO has shown satisfactory results when used in ghost agents to play against Ms. Pac-Man in the Ms. Pac-Man versus Ghost Team competitions CEC 2011 [7] and CIG 2016 [8].

I. Introduction:

Ant colony optimization (ACO) is a stochastic or probabilistic technique for finding a global solution or global optima in a discrete space in a graph. ACO is a bio inspired algorithm adopted from the technique of ant behavior or ant foraging of resources and their indirect communication among themselves through use of chemical called pheromones. This process is called stigmergy which is the indirect communication between agents by modifying their external environment. In

ACO, search agents, inspired by ants, lay down pheromones as they travel through the space and keep memory of their whereabouts and the number of nodes they visited, as well as the fitness of possible candidate solutions. ACO is a population based metaheuristic algorithm that is a branch of swarm intelligence where the search algorithm uses a population or a multiple of search agents inspired by collective behaviors of insects or ants' foraging methods. Usage of multiple search agents speeds up the exploration through the graph, where number of search agents depends on the population size with which the algorithm initiates the search. Different candidates contribute jointly to explore new solutions [22][23][24][25][26].

.

In ACO, a search agent builds a possible candidate solution node or solution state through the use of heuristics at every iteration of the algorithm. Ghosts use bio inspired ant search agents to accept new possible solution states or nodes using a probability function that depends on the heuristic function and pheromone levels of an edge. The value of transition probability of moving from current state to its next best solution state depends on the level of attractiveness or visibility of the node or state which is a heuristic function that requires a priori knowledge of that particular node. The value of the probability also depends on the level of pheromones of the edge leading to that vertex which is a posteriori term. As explorer ants visit nodes through edges to find the global optima, pheromone levels of edges are locally updated in real time as the explorer ant passes every edge. Evaporation rate of pheromone level helps to prevent immature convergence by explorer ants by reducing pheromone level in respective edges so other ants from the same ghost agent are less likely (reduce acceptance probability) to accept the same edge that helps in more exploration of the tree graph and prevents immature convergence by preventing all other ants taking the same route.

After ants visit the edges by laying down pheromones throughout all the routes, ants stop traveling through edges via some stopping criteria. After ants stop their individual tours, only the pheromone levels laid down in the current iteration by the best explorer ant (ie. ant with the shortest route) is then globally updated. In the game of Ms. Pac-Man, ghost agents try to minimize the score of Ms. Pac-Man by finding the optimal and suboptimal paths to their adversary from all sides. For this objective, two different tasks are carried out [7][8]:

(1) Find the shortest path to Ms. Pac-Man in order to chase it.

(2) Find additional suboptimal paths in order to corner or flank its adversary.

In the first case of chasing Ms. Pac-Man, the objective is to minimize the distance between ghost agents and their adversary. Explorer Ants are used to estimate Ms. Pac-Man's move by finding an optimal path and other suboptimal paths (to try to flank it) by releasing these explorer ants or search agents from adjacent vertex locations of Ms. Pac-Man to find the nearest ghost (ACO algorithm finds the nearest ghost to compute the first move out of all other ghosts). In reference [7] and [8], the task to flank or corner Ms. Pac-Man is done via two kind of ants, ie. explorer ants and hunter ants.

II. <u>Explorer ants help to find the nearest ghost agent via local and global update:</u>

Explorer Ants are used to find the closest ghost to their adversary so that nearest ghost can converge to its adversary via the shortest route, while other ghosts use hunter ants can converge from different directions in order to flank it. Explorer Ants converge on locations of the nearest ghost from the location of Ms. Pac-Man while locally updating the pheromone level within the edges. The pheromone level of that route deposited by the best explorer ant (ie. the ant with the shortest route) in the search graph is updated via global update equation. As explorer ants choose the optimal or shortest route, global pheromone levels are updated in that route by the best explorer ant with the shortest route after all ants stop travelling which is governed by a stopping condition. After explorer ants find the shortest path to the nearest ghost, search agents or hunter ants are released from the adjacent locations of the nearest ghost. As the nearest ghost's hunter ants uses the pheromones laid down by explorer ants, nearest ghosts follow the shortest path to its adversary.

III. <u>Reduction of pheromone levels in global update by hunter ants:</u>

As soon as the nearest ghost is located, the ghost releases hunter ants or search agents from its location node. These hunter ants would be guided by explorer ants pheromone level with same heuristics as to find the node nearest to Ms. Pac-Man. As a result the first ghost or the nearest ghost finds the shortest path as it uses the pheromones laid down by explorer ants as well their own pheromones. In reference [7] and [8], the evaporation of pheromones or reducing the explorer ant's pheromone level to a minimum in the global update iteration (after nearest ghost computes its shortest route) ensures other ghosts have a much higher probability of following a different route. Note, removing all pheromones from a particular edge reduces pheromone level to 1. Reducing the pheromone level in the edges travelled by the best ghost agent having the optimal route ensures this path to be less attractive for other ghost's hunter ants.

Since ACO based ghost agents are programmed in a different simulator, it could not be used in this thesis for performance evaluation against the proposed model. In the case of ghosts in the game of Ms. Pac-Man, use of ACO in ghost agents as shown in reference [7] and [8] illustrates desirable flanking behaviours by ghost agents as it tries to corner or trap their enemy as illustrated in figure 16.

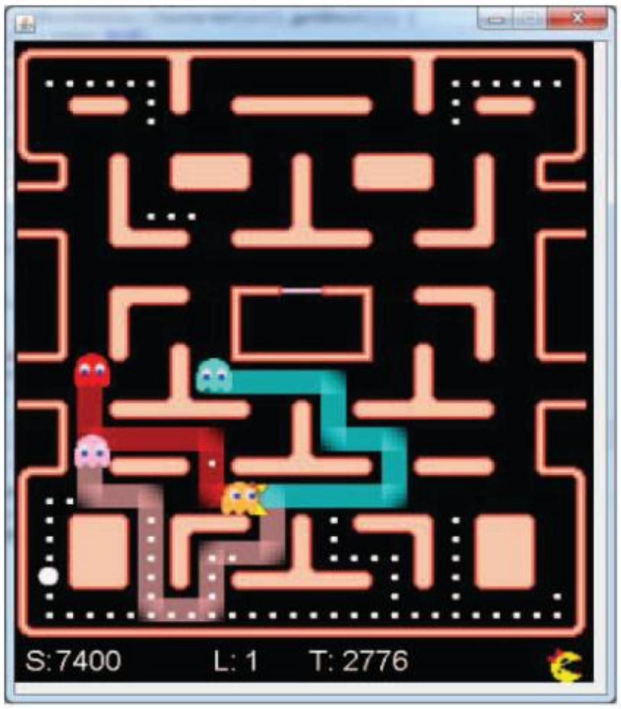


Figure 17: Shows a screenshot of different ghosts choosing different, optimal and sub-optimal paths to converge on the position of Ms. Pac-Man in order to flank it from all sides.

CHAPTER 7

RESULTS & DISCUSSION :

The performance evaluation of the proposed stochastic mSA optimization search model is evaluated by the methods similar to reference [29] that are as follows:

(I) The mSA optimization method of ghost agents in the game of Ms. Pac-Man is applied in order to compare with baseline algorithms such as a deterministic greedy search method called HC along with the traditional SA referred in this experiment as vSA; all of which uses heuristic search described in chapter 2. The performance of the algorithms' respective convergence navigation through different locations of the game environment (where its adversary remains stationary) along with number of iterations is compared as illustrated in figures 17, 18 and 19.

(II) The mSA algorithm using the heuristic developed in this thesis is also used to evaluate attacking strategies of multiple ghost agents as in the Atari 2600 stochastic game of Ms. Pac-Man where the adversary of the ghost agents, ie. Ms. Pac-Man is controlled by a human player. As ghosts use the mSA in order to flank or restrict their enemy, the ghosts also locally interact among themselves by maintaining a level of separation between themselves. In each simulation, Ms. Pac-Man evades the ghost agent until getting trapped by the artificially intelligent ghost agents.

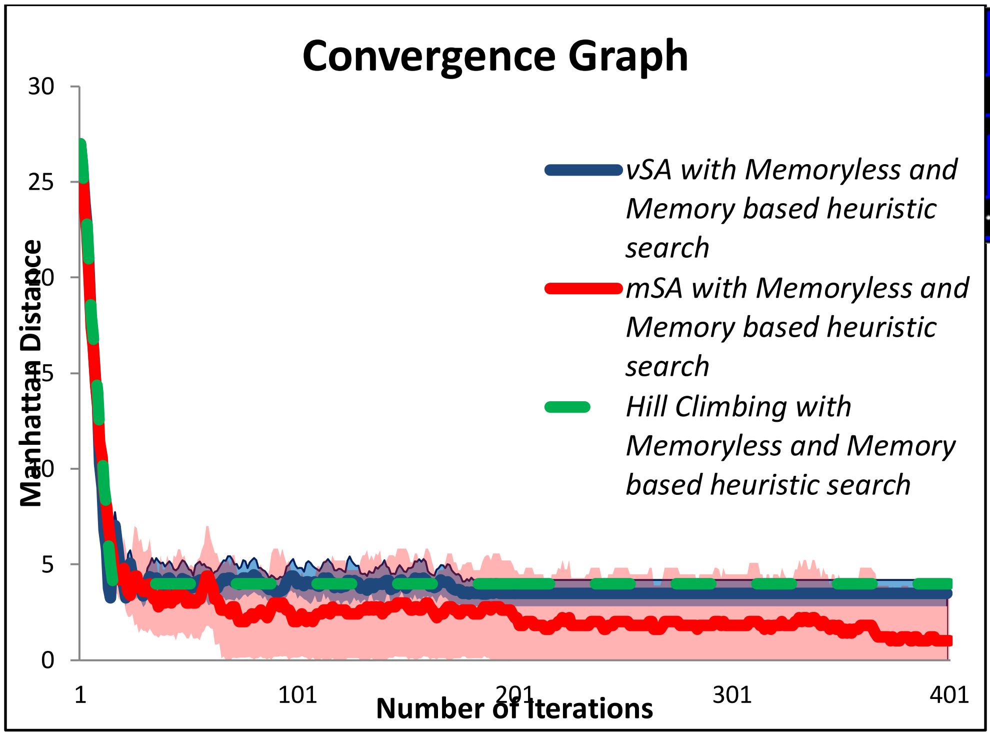
I. <u>Comparison of the proposed mSA against vSA and HC in the maze navigation experiment:</u>

Figures 17, 18 and 19 demonstrate three different scenarios of maze navigation where the adversarial agent remains stationary at three different locations. In each of the scenarios, ghost agents start from an initial starting position and navigate through the complex maze environment. The modified SA is compared with deterministic greedy HC search algorithm where the HC algorithm always chooses a better node on a greedy basis without exploring any other options and completely rejects a worse solution. The modified SA is further compared with vSA that has a monotonically decreasing temperature that decreases with interval of 10 iterations, $T = T_0 \times (0.9800)^{iteration}$, where T_0 is initial temperature.

Performance is measured by the successful reduction of Manhattan distance between ghost agent and its target in the maze environment. The number of iterations is limited to 400 iterations per trial as vSA iteration starts with exploration and end towards exploitation phase within 20 iterations as well mSA algorithm converges towards Ms. Pac-Man within 20 iterations with exploitation in the beginning of the iterations and then switches to exploration as the mSA algorithm converges closer to its solution. All the algorithms used in this experiment use the heuristics referred in this thesis as memoryless and memory based heuristic search that helps the algorithms to point in the best direction in order to find the shortest path or point the better candidate solution which helps the algorithms to converge close to their solution in less than 20 iterations.

Each simulation is carried out twenty times to obtain a mean and standard deviation of algorithm performance to converge in its target by navigating through the maze. The ghost agent uses mSA that depends on a sigmoid function while choosing better solutions (or while choosing better neighboring candidate nodes) as well choosing worse solutions (or choosing worse neighboring candidate nodes) that provides the algorithm a more robust search during exploitation phase. The shaded area represents standard deviation consisting of upper and lower bounds indicating exploration of worse or better candidate node solution based on an acceptance probability. This mSA algorithm is compared with the baseline vSA algorithm where a ghost bot greedily accepts a better solution but explores worse solutions based on an acceptance probability governed by the control parameter (temperature).

Scenario 1:



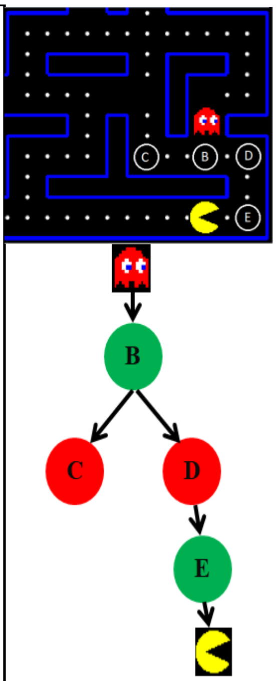


Figure 18: The convergence of Vanilla SA, modified SA and Hill Climbing within ghost agent to converge to its enemy that indicates the mean (marked line) and standard deviation (shaded area) of each algorithms (left). The right shows a snapshot of location of ghost and its adversary where it navigates through the game environment; and illustration of ghost tree structure when it gets trapped in a local maxima, where red colored vertex indicates 'bad' nodes while green colored nodes indicate 'good' nodes. mSA is still able to traverse the maze (bad nodes) while vSA would become trapped in a local maxima after a sufficient number of iterations.

In figure 17, the mSA algorithm outperforms both HC and vSA as it converges to its target to completely reduce Manhattan distance in less than 20 iterations. As the mSA algorithm iteration progresses and the ghosts approach its adversary, the threshold of its acceptance function changes along with Manhattan distance. As modified SA algorithm starts from high Manhattan distance, this high Manhattan distance N^* gives the algorithm high acceptance probability of accepting good or better neighboring solutions ie. $Pr_A(N^*) = 1$ while having low acceptance probability of accepting bad nodes or worse candidate solutions, ie. $1 - Pr_A(N^*) = 0$ that marks the algorithms' exploitation phase. As ghost agent converges towards its target at low Manhattan distance, acceptance probability of accepting better neighboring solutions $Pr_A(N^*)$ becomes equal to acceptance probability of accepting worse neighboring solutions $1 - Pr_A(N^*)$ that gives the algorithm a 50% chance to explore among better solutions while simultaneously giving it a 50% chance to explore worse solutions which marks the algorithm's exploration phase.

This change in acceptance probability to 0.5 helps mSA algorithm to explore the solution space more robustly that also creates high standard deviation when the ghost agent comes close to its solution in its maze navigation process but converges to its solution faster than other algorithms with very little standard deviation due its exploitation phase in the beginning of its iteration process that enables it to explore (in a non-greedy manner) among better neighbour nodes with high acceptance probability and to explore among worse candidate solutions with low acceptance probability.

In figure 17, the vSA algorithm controlled ghost agent uses a heuristic search referred as memoryless search heuristic and memory based search heuristic described in chapter 2 of this thesis. The vSA algorithm selects a neighboring candidate node and deterministically accepts better candidate node solutions but explores worse nodes when the algorithm comes across worse candidate solutions. In the beginning of vSA as the temperature is high, the algorithm has high acceptance probability value of accepting worse candidate node solutions. This makes the algorithm to explore worse neighboring nodes regardless of the energy value of the respective nodes being evaluated that gives ghost agents a high exploration rate in the beginning of its iteration process in simulation trials as they approaches their target.

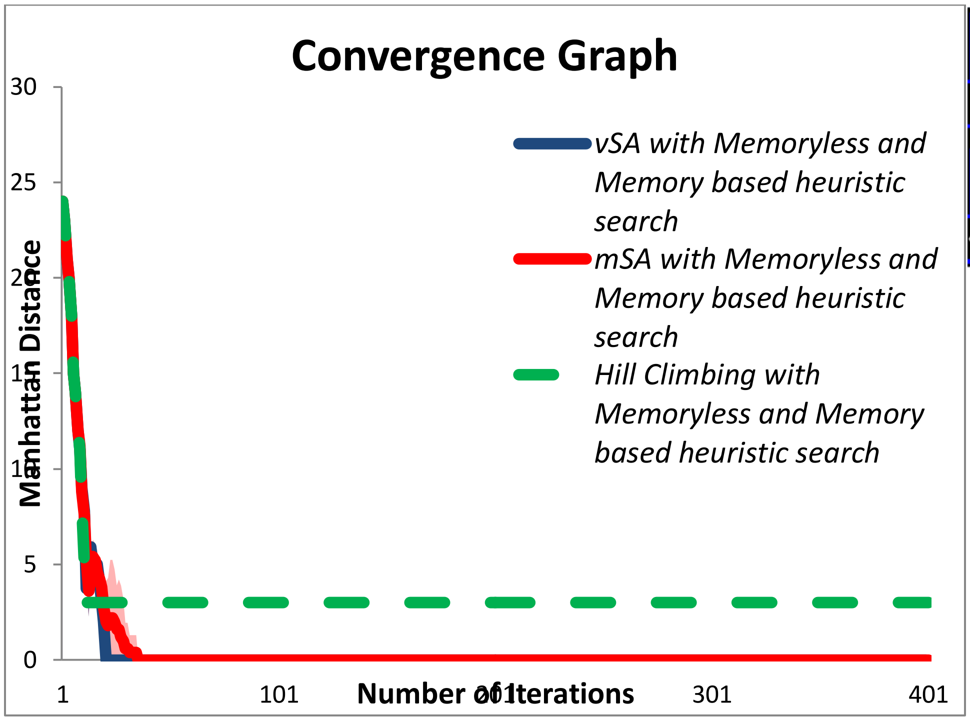
At the end of the vSA algorithm's iteration, temperature reduces and threshold of its corresponding Boltzmann's function reduces that decreases the probability of acceptance of worse neighboring solutions. As the temperature decreases with successive iterations, acceptance probability is reduced and the algorithm possess a low probability of accepting a worse solution. This makes ghost agent to accept good or better neighboring candidate nodes and reject worse candidate solutions with high probability which provides an exploitation phase for the vSA algorithm that makes it converge closer at its target location towards end of the iterations. This also makes vSA susceptible to getting stuck in local optima during low temperature as the number of algorithm iterations progresses.

Due to the heuristics described in chapter 2 referred as memoryless search heuristic and its good starting location in the maze where the algorithm comes across only better neighboring candidate nodes in the beginning of the iteration process leading to the fast convergence towards the solution as the vSA greedily accepts better candidate nodes deterministically. As the ghost agent using the vSA algorithm comes close to its solution, it comes across worse candidate nodes in the maze (after 20 iterations) that leads to small standard deviation. Due to the cooling of its temperature in vSA algorithm shown in figure 7 of chapter 4, the vSA starts acting more like hill climbing after 150 iterations by having very low acceptance probability of accepting a worse solution and greedily accepting better neighbouring candidate nodes that makes the ghost agent using this algorithm to become stationary after 150 iterations.

The ghost agent that uses HC has a fast convergence towards its target in less than 10 iterations due to the heuristics developed in this thesis that helps point towards the shortest path towards the ghost or helps to find the better candidate nodes as the HC greedily accepts better neighbour nodes but rejects worse neighbour nodes.

Ghost agent using HC algorithm fails to reach its target destination in all the trails by getting trapped in a local optimal vertex as the HC is always prone to get trapped in a local optima. Deterministically greedy based selection of a better node solution as ghosts navigate through the maze does not create any variability in their paths as they approaches their target.

Scenario 2:



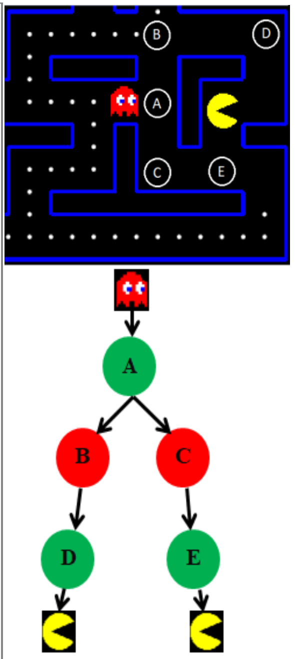
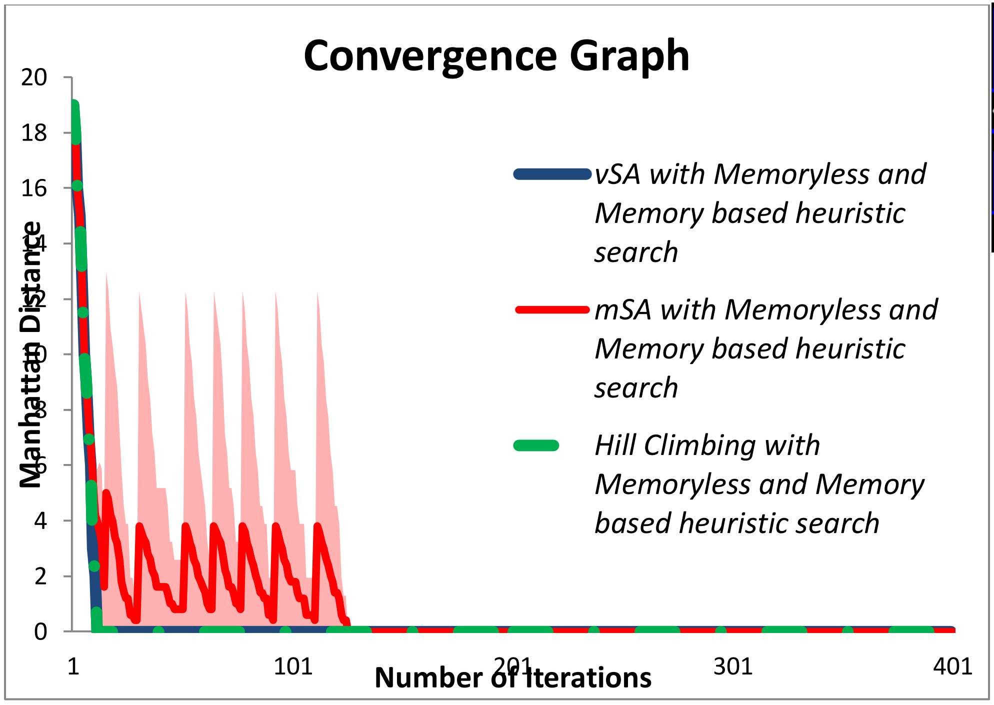


Figure 19: The convergence of vSA, mSA and HC in ghost agents to converge to their enemy in a different location (left). The right shows the location of a ghost, the adversary and part of the maze environment with an instance of the decision tree structure. Red colored vertex indicates 'bad' nodes while green colored nodes indicate 'good' nodes. Both vSA and mSA are still able to traverse the maze (bad nodes) while HC would become trapped in a local maxima.

In figure 18, the deterministic greedy search algorithm of HC converges at the same rate as vSA in less than 10 iterations. The ghost agent controlled by the HC algorithm always gets trapped in a local maximum node from where it cannot detect a better candidate solution in contrast to the vSA algorithm that does not get trapped in a local optima as it can explore with worse candidate solution that creates some standard deviation when the ghost agent comes close to its solution.

Also, the mSA algorithm converges fast to its target solution with no standard deviation in the beginning of its convergence process towards the target solution as it always comes across better neighboring candidate nodes. Its high acceptance probability of accepting better candidate nodes leads to fast convergence towards the target solution. As the ghost agent using mSA algorithm comes close its target solution, ghost agent turns from exploitation to exploration which shows high standard deviation in figure 18 (pink coloured area).

Scenario 3:



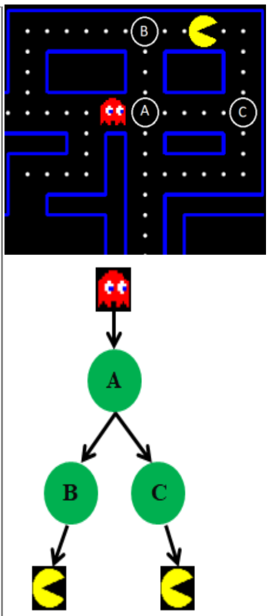


Figure 20: The convergence of vSA, mSA and HC of ghost agents their enemy (left). On the right, the location of a ghost, the adversary and a portion of the maze is shown. The decision tree structure for this particular event of the dynamic tree shows that ghost always comes across 'good' nodes (right).

In figure 19, the deterministic HC greedy search algorithm and the vSA optimization algorithm outperforms mSA or reaches its target faster than mSA in less than 10 iterations. In this scenario, all the algorithms experience only better candidate nodes as they iterate towards their adversary. This makes the vSA and HC to converge faster towards the target as they greedily accept better nodes which is in contrast to the mSA algorithm that explores among better candidate nodes. The exploration among better candidate nodes leads to high standard deviation as it converges close to Ms. Pac-Man at which point it comes across a choice of better neighboring candidate nodes, i.e. vertex B and vertex C. In addition, if vSA were to run indefinitely it would at some point simply resemble HC as the control temperature approaches 0. This would be apparent in a more realistic game environment where a human controlled Ms. Pac-Man also tries to evade the ghost agents.

In the scenarios presented above, it is not particularly obvious which would be best. It is argued that mSA would be a better algorithm to use to control NPCs. mSA in all scenarios converges to its adversary, mSA displays flanking and blocking, and the behaviour of mSA is not dependent upon the number of iterations for its probability of acceptance criteria. More specifically, the behaviour of the NPCs would make for a more engaging game when Ms. Pac-Man were human or computer controlled.

II. Flanking & Blockade Behavior of Ghost Agents:

The objective of this research is to portray complex attacking strategies of ghost agents using a stochastic search method based on the proposed mSA to maneuver through the maze and converge onto their adversary. The stochastic nature of the mSA with its feedback mechanism would ensure ghost agents display a flanking maneuver as well as a blockade maneuver to cut off escape routes of their adversary while other agents converge to their global maxima. The use of individual sigmoid acceptance functions, if used, would also ensure ghost agents have non-unique acceptance probabilities at the same Manhattan distance to Ms. Pac-Man.

Moreover, ghost agents maintain a level of separation or make decisions to impose a separation constraint in the event of their proximity of 1 unit distance as they travel throughout the maze (mentioned in chapter 1, part 4) that further contributes to the convergence of their adversary from all directions. This separation among ghost agents further increases the effectiveness of their attack strategy. The proposed mSA algorithm is used in this investigation to create these blocking and flanking behaviors based solely on their stochastic exploitation and exploration phases. Consequently, these behaviours are not seen with HC or vSA algorithms.

Use of a different and independent threshold of sigmoid functions provides ghost agents with different acceptance probabilities with respect to the Manhattan distance of each ghost to Ms. Pac-Man. This leads to complex overall group interactions with their adversary such as flanking maneuvers and blockades of junctions. Therefore, usage of individual sigmoid decision functions in mSA for ghost agents provides more varied attacking maneuvers such as blockade and flanking strategies.

i. *Ghost Agents with Higher Threshold of Sigmoid Function display Flanking Behavior:*

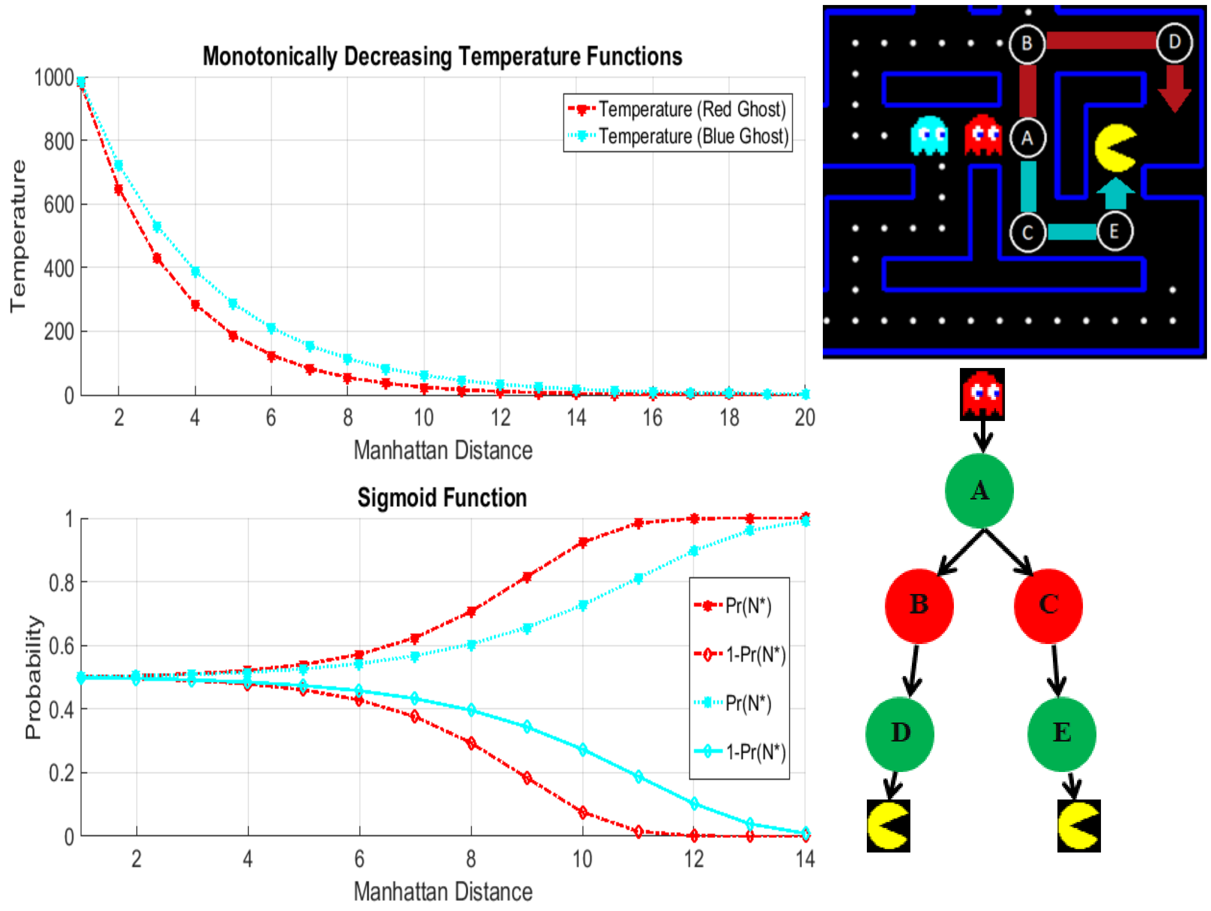


Figure 21:_Shows the relationship between distance and acceptance function which is $Pr_A(N^*)$ for better neighboring solutions and $1 - Pr_A(N^*)$ for worse neighboring solutions for ghost Red and Cyan (left). A scenario of navigation in the game environment of Ms. Pac-man through decision trees (right).

In figure 20, Temperature is a monotonically decreasing function,

$$For\ ghost\ Red, T_1(N^*) = T_0(0.980)^{2^{N^*}},$$

$$For\ ghost\ Cyan, T_2(N^*) = T_0(0.985)^{2^{N^*}},$$

For ghost Red, the sigmoid function is:

$$Acceptance\ Probability, Pr_{A1}(N^*) = \left\{1 + exp\left(-6 \times N^* / T_0(0.980)^{2^{N^*}}\right)\right\}^{-1} \quad ;$$

For ghost Cyan, the sigmoid function is:

$$Acceptance\ Probability, Pr_{A2}(N^*) = \left\{1 + exp\left(-6 \times N^* / T_0(0.985)^{2^{N^*}}\right)\right\}^{-1} \quad ;$$

where N^* is a heuristic function defined by Manhatan distance from ghost to Ms. Pac-man, their corresponding acceptance function is defined by a Sigmoid function.

The pseudocode of the above mentioned algorithm can be written as:

Algorithm 4: Modified Simulated Annealing display Flanking Behavior

```
1:   Initialize Temperature = 1000
2:   while Manhattan distance $> 0$ do
3:     Calculate $Acceptance\ Probability, Pr_A(N^*) = \{1 + exp\ (-6 \times N^*/ T)\}^{-1}$
4:     if $\Delta E > 0$ then
5:       if $Pr_A(N^*)$ > Random_Number_Generator then
6:         Accept node
7:       else
8:         Reject Node
9:       end
10:    else $\Delta E < 0$ then
11:      if $1 - Pr_A(N^*)$ > Random_Number_Generator then
12:        Accept node
13:      else
14:        Reject Node
15:      end
16:    end
19:  Until Manhattan Distance is 0.
```

a. *Acceptance Probability among Better Candidate Solutions:*

In this simulation scenario, ghost agents use a sigmoid function with higher and lower distance thresholds that leads to corresponding lower and higher acceptance probabilities based on a shifted sigmoid function. When the Manhattan distance is high, the acceptance probability, $Pr_A(N^*)$ is high or close to 1, facilitating exploration among better candidate solution nodes with high acceptance probability. When the Manhattan distance is low (or when ghost agents come close to their adversary), the acceptance probability, $Pr_A(N^*)$, decreases to 0.5. This lowering of acceptance probability to 0.5 enables ghost agents to reduce their exploitation of better candidate solutions. As the acceptance probability or $Pr_A(N^*)$ becomes 0.5, the algorithm has an equal chance to accept or reject good nodes.

b. *Acceptance Probability among Worse Candidate Solutions:*

For worse candidate solutions, the acceptance probability $1 - Pr_A(N^*)$ is low or close to 0 when the Manhattan distance is high. This creates low exploration among bad nodes or worse solutions at the beginning of the iteration. As Manhattan distance is reduced (or when ghost agents come close to its adversary), acceptance probability, $1 - Pr_A(N^*)$ increases to 0.5. The increase in acceptance probability to 0.5 enables ghost agents to increase its exploration of worse candidate solutions. As probability or Pr_A becomes 0.5, the algorithm has an equal chance to accept or reject bad nodes.

Having equal probability of 0.5 of accepting good or bad nodes helps multiple ghost agents to accept optimal node by one ghost agent while the other ghost accepts a sub-optimal node thereby portraying a flanking maneuver tactic as mSA agents, ie. ghost agents, choose different routes in their process of convergence towards their adversary. This flanking maneuver of ghost agents helps in minimizing the score of their adversary, ie. Ms. Pac-Man. In addition, ghost agents with different acceptance probabilities tend to maintain a minimum separation distance between themselves that further increases the chance that ghost agents would choose different paths as they converge towards Ms. Pac-Man. In other words, the bots' adversary is attacked from two opposite sides out of its surrounding junctions in order to outflank Ms. Pac-Man via mSA algorithm and the maintenance of separation among ghost agents.

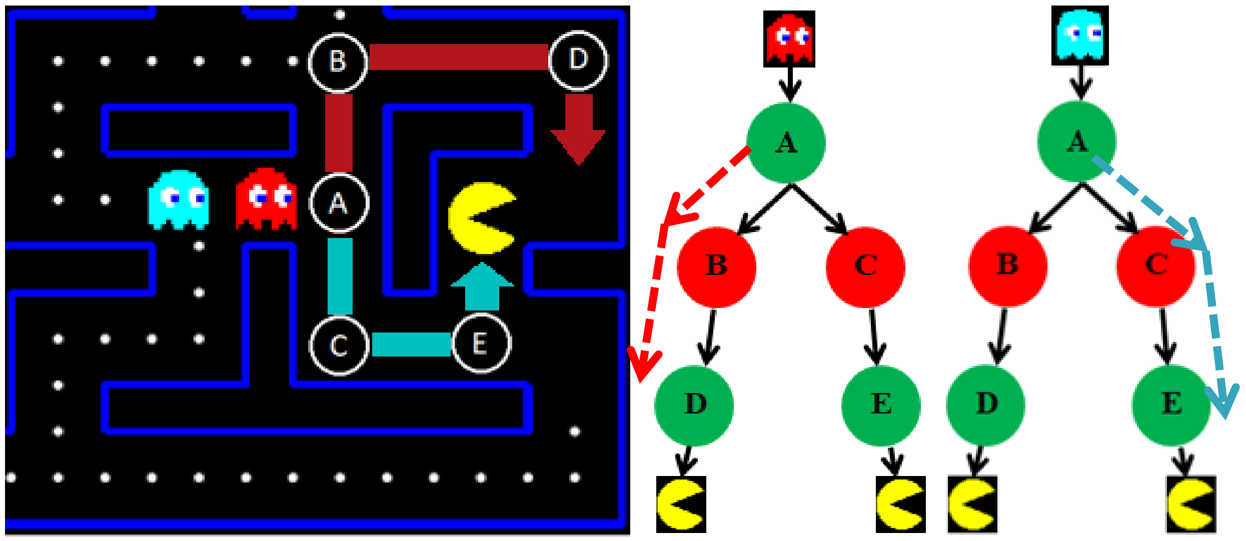


Figure 22: Shows a snapshot of possible flanking behavior by ghost agents when they come across worse nodes (left). The arrows in the tree structure of ghost agents indicate ghosts respective routes and color of nodes indicate better or worse solutions (right).

ii. *<u>Ghost Agents with Low Temperature Rates display Blocking Strategy:</u>*

a. *<u>ϵ Variation</u>*

Adding an additional control parameter of offset ϵ may provide an additional control parameter that can be tuned to promote variations in emergent group behaviour. This section explores the role of an acceptance probability offset that promote blocking or denying in NPS behaviours in games like Ms. Pac-Man. As a result in this simulation scenario, ghost agents block Ms. Pac-Man's surrounding T-shaped or L-shaped junctions denying or restricting Ms. Pacman's movements across the maze which portrays blockade strategy at different locations of maze in the gaming environment . In other words, lowering the threshold of sigmoid function shown in figure 22, leads to the display of a blockade strategy that contributes in reducing score of their adversary.

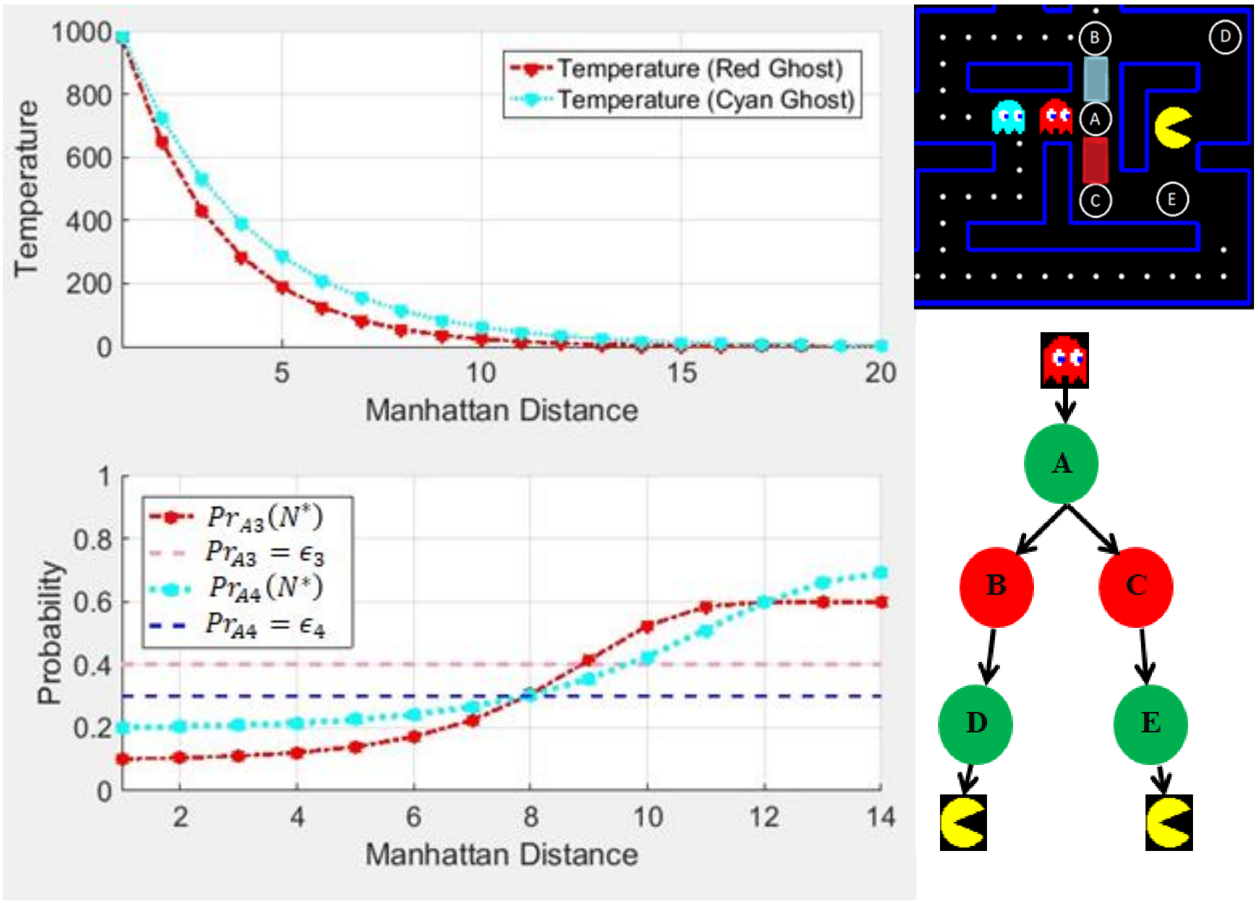


Figure 23: Shows the relationship between distance and acceptance function which is $Pr_{A3}(N^*)$ for better neighboring solutions and $Pr_{A3} = \epsilon_3$ for worse neighboring solutions for ghost Red; $Pr_{A4}(N^*)$ for better neighboring solutions and $Pr_{A4} = \epsilon_4$ for worse neighboring solutions for ghost Cyan (left). A particular scenario of navigation in the game environment of Ms. Pac-Man (upper right) and its corresponding decision tree (lower right).

In figure 22, Temperature is a monotonically decreasing function,

$$for\ Ghost\ Red, T_3(N^*) = T_0(0.980)^{2^{N^*}},$$

$$for\ Ghost\ Cyan, T_4(N^*) = T_0(0.985)^{2^{N^*}},$$

For Red ghost,

$$Acceptance\ Probability, Pr_{A3}(N^*) = \left\{1 + exp\left(-6 \times N^* / T_0(0.980)^{2^{N^*}}\right)\right\}^{-1} - \epsilon_3;$$

For Cyan ghost,

$$Acceptance\ Probability, Pr_{A4}(N^*) = \left\{1 + exp\left(-6 \times N^* / T_0(0.985)^{2^{N^*}}\right)\right\}^{-1} - \epsilon_4;$$

where N^* is a heuristic function defined by Manhatan distance from ghost to Ms. Pac-man, their corresponding acceptance function is defined by a Sigmoid function.

The pseudo code can be written as:

```
Algorithm 5: Modified Simulated Annealing display Blocking Behavior
1:     Initialize Temperature = 1000
2:     while Manhattan distance $> 0$ do
3:        Calculate $Acceptance\ Probability, Pr_A(N^*) = \{1 + exp\,(-6 \times N^* / T)\}^{-1}$;
4:        if $\Delta E > 0$ then
5:           if $Pr_A(N^*) - \epsilon$ > Random_Number_Generator then
6:              Accept node
7:           else
8:              Reject Node
9:           end
10:       else $\Delta E < 0$ then
11:          if $Pr_A = \epsilon$ > Random_Number_Generator then
12:             Accept node
13:          else
14:             Reject Node
15:          end
16:       end
17:    Until Manhattan Distance is 0.
```

b. *Acceptance Probability among Better Candidate Solutions:*

Since ghost agents are more likely to come across good nodes when the Manhattan distance is high (N^*), having probability function, $Pr_A(N^*) - \epsilon$ while accepting better solution nodes gives ghost agents a high acceptance probability (Pr_A) of accepting good nodes or better solutions in its initial phase of convergence that provides fast convergence towards its adversary. This marks the exploration of better solution nodes when Manhattan distance is high but to a lesser degree than investigated previously. Since ghost agents are more likely to come across better neighboring nodes when Manhattan distance is high, having relatively high threshold of acceptance function enables ghost to quickly converge on to their adversary.

The desired strategy for blockade maneuver is fast convergence towards ghost agent followed by blocking or denying Ms. Pac-Man's movement as the Manhattan distance reduces instead of simply converging via the shortest path. Therefore as ghost agents come close to their adversary, the acceptance probability, $Pr_A(N^*) - \epsilon$ reduces which further reduces acceptance of better nodes. The lowering of the threshold of sigmoid function provides a lower exploration phase among better nodes when it comes close to Ms. Pac-Man that gives the effect of ghost agent to block Ms. Pac-Man's surrounding junctions that help to corner their enemy.

c. *Acceptance Probability among Worse Candidate Solutions:*

The proposed ϵ variation is used as a threshold function for the probability of accepting a worse solution as $Pr_A = \epsilon$, which remains constant regardless of change in Manhattan distance between ghost agent and Ms. Pac-Man. This low threshold of an acceptance probability gives ghost agents a low acceptance probability, Pr_A, which makes them prone to reject worse nodes or have a low probability of accepting a suboptimal solution. A low acceptance probability of worse solutions, $Pr_A = \epsilon$ makes ghost agents to have low exploration of worse neighboring candidate solutions throughout its iteration process, independent of their Manhattan distances to Ms. Pac-Man.

Since ghost agents are more likely to come across bad or worse solution nodes when they are close to their adversary or when Manhattan distance is low, having a low threshold of the acceptance function $Pr_A = \epsilon$ makes ghost agents reject bad nodes and to stand their ground instead of converging towards their adversary as the Manhattan distance reduces. So, using ghost agents with lower acceptance function, $Pr_A = \epsilon$ while selecting bad nodes (worse solutions are more common when Ms. Pac-Man is closer) leads to the lowering of acceptance probability that makes ghost agents to stand their ground or restrict their adversary's movement when they are close to Ms. Pac-Man, which in turn, helps to contribute in the display of ghost agents blockade behavior.

Therefore, the combination of a low threshold of the sigmoid function of acceptance probability, $Pr_A(N^*) - \epsilon$ when ghost agents come across better candidate solution and the threshold function of acceptance probability, $Pr_A = \epsilon$, when ghost agents come across worse solution nodes display a blockade maneuver tactic that denies or restricts Ms. Pac-Man's movements which helps ghost agents minimize the score of their adversary. Although the combination of two acceptance functions mentioned above (ie. $Pr_A(N^*) - \epsilon$ and $Pr_A = \epsilon$) helps to corner their adversary in denying its enemy's movement; the algorithm portraying blockade maneuver does not have any exploration and exploitation property of a metaheuristic algorithm. Moreover, the maintenance of separation of a single unit distance among ghost agents helps to ensure ghost agents to stand ground at different locations of the maze that further helps in restricting Ms. Pac-Man's movements and minimizing its score.

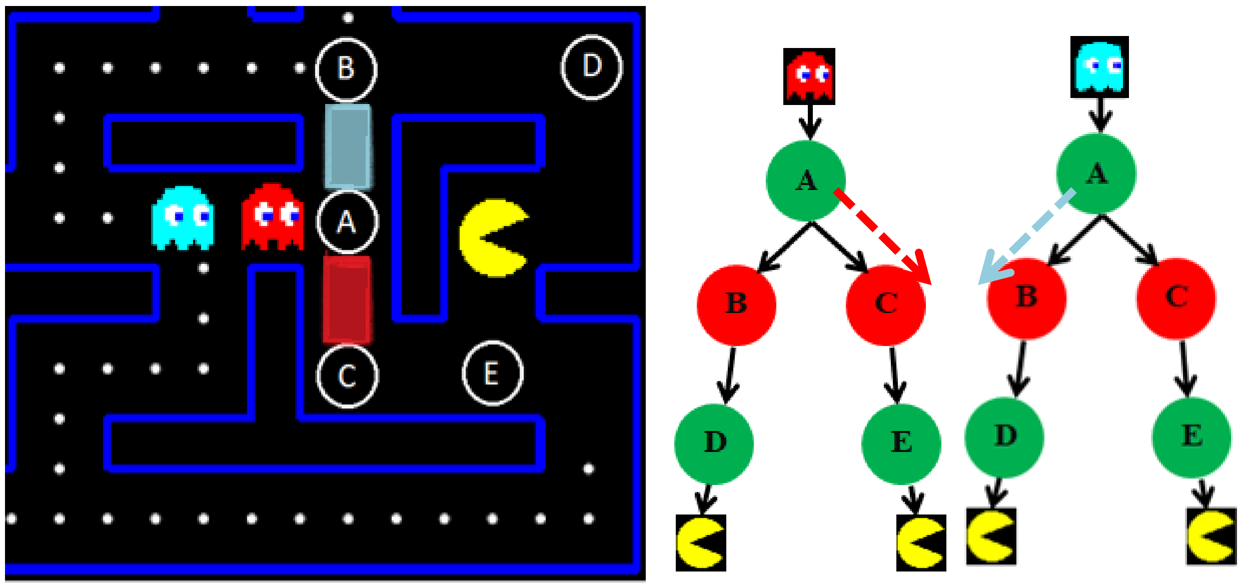


Figure 24: The blockade behavior of ghost agents where ghosts have high probability of rejection of worse neighboring nodes (left). The tree structure of ghost agents where red color of nodes indicate worse solutions and green color of nodes indicate better solutions and arrows indicate ghosts respective paths (right).

The use of different threshold values of acceptance functions such as, $Pr_A(N^*)$, for better neighboring nodes and $1 - Pr_A(N^*)$ for worse neighboring nodes as well as other acceptance functions such as, $Pr_A(N^*) - \epsilon$ for better neighboring nodes and $Pr_A = \epsilon$ for worse neighboring nodes; makes AI controlled ghost agents display flanking maneuvers and blockade maneuvers. Moreover, all artificially controlled ghost agents maintain a separation distance between themselves as they chase their adversary which ensures each blocking agents block at different surrounding junctions of Ms. Pac-Man while other agents flank from other sides towards their adversary. Therefore, all four ghost agents within the maze environment display flanking from surrounding junctions and blocking of their adversary at critical choke points surrounding their adversary. The strategy of blocking of an adversary's critical cross points and at the same time mounting a flanking attack from two opposite sides is called the pincer attack. A demonstration of this pincer attack that consists of blocking of Ms. Pac-Man’s surrounding junctions and creating a flanking attack is demonstrated in this simulation scene at https://www.youtube.com/watch?v=Pvo7TSSghS0.

CHAPTER 8

CONCLUSION

Ms. Pac-Man provides a solid proving arena for programming novel artificial intelligent algorithms and the development of NPCs. This game provides a good opportunity for constructing and evaluating deterministic or non-deterministic tree search algorithms. Throughout the last decade, several artificial intelligence algorithms have been applied to tackle the dynamic probabilistic behavior of Ms. Pac-Man. These include proposed models of swarm intelligence based on ant colony optimization [7][8] and Monte Carlo tree search [9][10]. These algorithms show some efficacy in surrounding their adversary to lower its score in the game environment. Using stochastic search to optimize path selection of ghost agents provides satisfactory results in dealing with the stochastic nature of their adversary, which makes Ant Colony optimization algorithm a good choice in dealing with Ms. Pac-Man.

The work here was interested in developing other novel algorithms that may be simpler to implement while yielding comparable NPC behaviour patterns. Therefore, another non-deterministic algorithm, mSA, was developed and applied by ghost agents and was evaluated against HC and vSA in terms of their performance of ghosts in the game, showing comparable or superior results in terms of the time and proximity of converging upon Ms. Pac-Man. Further, and to the author's knowledge, this is the first time a probabilistic optimization algorithm based upon modifications suggested by simulated annealing was used by ghost agents to demonstrate flanking and blocking behaviors in order to minimize the score of Ms. Pac-Man. These modifications were inspired by trade-offs seen during phases of exploration and exploitation.

One of the limitations of this thesis work is that it was not possible to compare the proposed model’s performance with the algorithms of past research due to the use of a different simulators. The simulator in Ms. Pac-Man Vs. Ghost Team competition is a good benchmarking platform for understanding the fitness of various computational intelligence algorithms used by related agents. However, it was the intent of this work to develop an entirely new algorithm for use within NPC game agents. For this reason, it was desirable to work from within a simulator developed for this purpose, facilitating algorithm exploration as opposed to algorithm implementation. To mitigate this issue, the algorithm developed here could now be constructed and run in the Ghost vs. Ms. Pac-Man simulator, and compared with other algorithms that showed satisfactory performance in the competition.

Future work could extend these concepts in the mSA algorithm proposed here in addition with a degree of local interacting agents to bring out more complex tactical maneuvers in the ghost agents. Subsequently, mSA or its variants could be used in another problem such as game of Go to bring out more complex strategies. The work here is useful as a medium for discussing the relative roles of exploration and exploitation phases in any algorithms that utilize these concepts.

It should be noted that the trade-off between exploration and exploitation within algorithms such the mSA presented here demonstrate that for game NPCs, although simple to implement display apparently complex behaviours. My development of mSA was from simulated annealing. This just happened to be the route I have taken, there are other approaches that may have led to similar interplay of exploration and exploitation. There are also alternative probability of acceptance functions that could have been developed and in the future this may be the case. However, it is the interplay between exploration and exploitation that may yet benefit a variety of search strategies, not being restricted to improving NPCs within games.

BIBLIOGRAPHY

[1] K. Anderton, “The Business of Video Games: A Multi-Billion Dollar Industry [Infographic],” Forbes, April 29, 2017. [Online]. Available: https://www.forbes.com/sites/kevinanderton/2017/04/29/the-business-of-video-games-a-multi-billion-dollar-industry-infographic/#51d7fa046d27

[2] P. Hingston, “A turing test for computer game bots,” *IEEE Trans. Comput. Intell. AI Games,* vol. 1, no. 3, pp. 169–186, 2009.

[3] R. Hunicke and V. Chapman, “AI for Dynamic Difficulty Adjustment in Games,” *Challenges Game Artif. Intell. AAAI Work.*, pp. 91–96, 2004.

[4] P. Spronck, I. Sprinkhuizen-Kuyper, and E. O. Postma, “Difficulty Scaling of Game AI,” *Proc. 5th Int. Conf. Intell. Games Simul. (GAME-ON 2004)*, no. May, pp. 33–37, 2004.

[5] Ms. Pac-Man Competition, Screen Capture Version [Online]. Available: http://dces.essex.ac.uk/staff/sml/pacman/PacManContest.html

[6] P. Rohlfshagen and S. M. Lucas, “Ms Pac-Man Versus Ghost Team CEC 2011 Competition,” *Evol. Comput. (CEC), 2011 IEEE Congr.*, pp. 70–77, 2011.

[7] G. Recio, E. Martin, C. Estebanez, and Y. Saez, “AntBot: Ant colonies for video games,” *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 4, pp. 295–308, 2012.

[8] I. Hunkeler, F. Schar, R. Dornberger, and T. Hanne, “FairGhosts - Ant colony controlled ghosts for Ms. Pac-Man,” *2016 IEEE Congr. Evol. Comput. CEC 2016*, pp. 4214–4220, 2016.

[9] G. Chaslot, S. Bakkes, I. Szita, and P. Spronck, “Monte-Carlo Tree Search : A New Framework for Game AI,” *Artif. Intell.*, no. 612, 2006.

[10] K. Q. Nguyen and R. Thawonmas, “Monte carlo tree search for collaboration control of ghosts in Ms. Pac-Man,” *IEEE Trans. Comput. Intell. AI Games*, vol. 5, no. 1, pp. 57–68, 2013.

[11] F. Liberatore, A. M. Mora, P. A. Castillo, and J. J. Merelo, “Evolving evil: Optimizing flocking strategies through genetic algorithms for the Ghost Team in the game of Ms. Pac-Man,” *Appl. Evol. Comput.*, pp. 313–324, 2014.

[12] T. G. Tan, J. Teo, and P. Anthoni, “A simple heuristic search method for the automatic generation of neural-based game artificial intelligence architectures in Ms. Pac-man,” *10th Int. Conf. Inf. Sci. Signal Process. their Appl. ISSPA 2010*, no. Isspa, pp. 753–756, 2010.

[13] P. R. Williams, D. Perez-Liebana, and S. M. Lucas, “Ms. Pac-Man Versus Ghost Team CIG 2016 competition,” *IEEE Conf. Comput. Intell. Games, CIG*, pp. 1–8, 2017.

[14] P. Abbeel, UC Berkeley, “CS188 SP14 Lecture 3 Informed Search,” Accessed Dec 24, 2018. [Video file]. Available: https://www.youtube.com/watch?v=8pTjoFiICg8&index=3&list=PLIZQvCoJVokgKBNx210mkXEk4FeSOGfWu

[15] Class notes for course AI2, Cardiff School of Comp. Sci., Cardiff University. “Heuristic search.” Accessed Dec 24, 2018. [Online]. Available: http://users.cs.cf.ac.uk/Dave.Marshall/AI2/node23.html

[16] D. Khemani, Department of Computer Science and Engineering, IIT Madras, “Hill Climbing” [Video file]. Available: https://www.youtube.com/watch?v=ZOvRZ7UJMjk

[17] P. Abbeel, UC Berkeley, "CS188 SP14 Lecture 3 Informed Search" [Video file]. Available: https://www.youtube.com/watch?v=8pTjoFiICg8&index=3&list=PLIZQvCoJVokgKBNx210mkXEk4FeSOGfWu

[18] A. Brabazon and M. O'Neil, S. McGarraghy, chapter 23.1, "Simulated Annealing" *Natural Computing Algorithms*, March 2015. [Online]. Available: https://doc.lagout.org/science/0_Computer%20Science/2_Algorithms/Natural%20Computing%20Algorithms%20%5BBrabazon%2C%20O%27Neill%20%26%20McGarraghy%202015-11-14%5D.pdf

[19] D. Khemani, Department of Computer Science and Engineering, IIT Madras, "Simulated Annealing" [Video file]. Available: https://www.youtube.com/watch?v=dg5zUxdAE_E&t=1953s

[20] C. Balaji, Department of Mechanical Engineering, IIT Madras, "Mod-01 Lec-40 Simulated Annealing and Summary" [Video file]. Available: https://www.youtube.com/watch?v=UXUFZYbvQss&t=349s

[21] B. Adhikari, "The simulated annealing algorithm explained with an analogy of a toy" [Video file]. Available: https://www.youtube.com/watch?v=eBmU1ONJ-os&t=452s

[22] A. Brabazon, M. O'Neil, S. McGarraghy, chapter 9.3, "Ant Algorithms for Discrete Optimisation" *Natural Computing Algorithms*, March 2015. [Online]. Available: https://doc.lagout.org/science/0_Computer%20Science/2_Algorithms/Natural%20Computing%20Algorithms%20%5BBrabazon%2C%20O%27Neill%20%26%20McGarraghy%202015-11-14%5D.pdf

[23] M. Dorigo and T. Stützle, *Ant Colony Optimization*. MIT Press, 2004.

[24] D. Khemani, Department of Computer Science and Engineering, IIT Madras, “Population Based Methods II” [Video file]. Available: https://www.youtube.com/watch?v=V-Yipf9GuH0&t=3063s

[25] J. Lettmann, “Ant Colony Optimization” [Video file]. Available: https://www.youtube.com/watch?v=xpyKmjJuqhk&t=531s

[26] A. Mirjalili, “How the ant colony optimization works” [Video file]. Available: https://www.youtube.com/watch?v=783ZtAF4j5g

[27] Lecture slides for Data Structures and Algorithms, Department of Comp. Sci. and Eng., University of Nebraska-Lincoln. “P, NP and NP Complete.” Accessed Dec 24, 2018. [Online]. Available: http://cse.unl.edu/~goddard/Courses/CSCE310J/Lectures/Lecture10-NPcomplete.pdf

[28] School of Comp. Sci., Carnegie University. “Complexity theory.” Accessed Dec 24, 2018. [Online]. Available: http://www.cs.cmu.edu/afs/cs/academic/class/15451-f99/www/lectures/lect1014

[29] H. M. Le, N. Jiang, A. Agarwal, M. Dudík, Y. Yue, and H. Daumé, “Hierarchical Imitation and Reinforcement Learning,” *Int. Conf. Mach. Learn.*, 2018.