ENTROPY BASED TECHNIQUES WITH APPLICATIONS IN DATA MINING

By

ANTHONY OKAFOR

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2005
This work is dedicated to my family.
ACKNOWLEDGMENTS

I want to thank Professor Panos M. Pardalos for his help and patience in guiding me through the preparation and completion of my Ph.D. I also want to thank Drs. Joseph P. Geunes, Stanislav Uraysev and William Hager for their insightful comments, valuable suggestions, constant encouragement and for serving on my supervisory committee.

I also would to thank my colleagues in the graduate school of the Industrial and Systems Engineering Department especially, Don Grundel.

Finally, I am especially grateful to my wife, my parents, and sister for their support and encouragement as I complete the Ph.D. program.
# TABLE OF CONTENTS

ACKNOWLEDGMENTS .................................................. iv

LIST OF TABLES .......................................................... viii

LIST OF FIGURES .......................................................... x

ABSTRACT ................................................................. xi

CHAPTERS

1 INTRODUCTION .......................................................... 1

1.1 Data Mining ........................................................... 1
    1.1.1 Classification ............................................... 2
    1.1.2 Clustering ...................................................... 2
    1.1.3 Estimation ..................................................... 2
    1.1.4 Prediction ..................................................... 3
    1.1.5 Description .................................................... 3

2 ENTROPY OPTIMIZATION .............................................. 4

2.1 Introduction ........................................................ 4

2.2 A Background on Entropy Optimization ............................. 4
    2.2.1 Definition of Entropy ....................................... 5
    2.2.2 Choosing A Probability Distribution ....................... 6
    2.2.3 Prior Information ........................................... 8
    2.2.4 Minimum Cross Entropy Principle ......................... 9

2.3 Applications of Entropy Optimization ............................. 10

3 DATA MINING USING ENTROPY ....................................... 12

3.1 Introduction ........................................................ 12

3.2 K-Means Clustering ................................................. 13

3.3 An Overview of Entropy Optimization ............................. 15
    3.3.1 Minimum Entropy and Its Properties ..................... 16
    3.3.2 The Entropy Decomposition Theorem .................... 18

3.4 The K-Means via Entropy Model ................................... 19
    3.4.1 Entropy as a Prior Via Bayesian Inference ................ 19
    3.4.2 Defining the Prior Probability ............................ 20
    3.4.3 Determining Number of Clusters ........................... 20
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5 Graph Matching</td>
<td>22</td>
</tr>
<tr>
<td>3.6 Results</td>
<td>25</td>
</tr>
<tr>
<td>3.6.1 Image Clustering</td>
<td>26</td>
</tr>
<tr>
<td>3.6.2 Iris Data</td>
<td>27</td>
</tr>
<tr>
<td>3.7 Conclusion</td>
<td>29</td>
</tr>
<tr>
<td>4 DIMENSION REDUCTION</td>
<td>30</td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>30</td>
</tr>
<tr>
<td>4.1.1 Entropy Dimension Reduction</td>
<td>30</td>
</tr>
<tr>
<td>4.1.2 Entropy Criteria For Dimension Reduction</td>
<td>31</td>
</tr>
<tr>
<td>4.1.3 Entropy Calculations</td>
<td>31</td>
</tr>
<tr>
<td>4.1.4 Entropy and the Clustering Criteria</td>
<td>32</td>
</tr>
<tr>
<td>4.1.5 Algorithm</td>
<td>32</td>
</tr>
<tr>
<td>4.2 Results</td>
<td>32</td>
</tr>
<tr>
<td>4.3 Conclusion</td>
<td>33</td>
</tr>
<tr>
<td>5 PATH PLANNING PROBLEM FOR MOVING TARGET</td>
<td>34</td>
</tr>
<tr>
<td>5.1 Introduction</td>
<td>34</td>
</tr>
<tr>
<td>5.1.1 Problem Parameters</td>
<td>35</td>
</tr>
<tr>
<td>5.1.2 Entropy Solution</td>
<td>36</td>
</tr>
<tr>
<td>5.2 Mode 1</td>
<td>38</td>
</tr>
<tr>
<td>5.3 Mode 2</td>
<td>41</td>
</tr>
<tr>
<td>5.4 Mode 3</td>
<td>41</td>
</tr>
<tr>
<td>5.5 Maximizing the Probability of Detecting a Target</td>
<td>42</td>
</tr>
<tr>
<td>5.5.1 Cost Function. Alternative 1</td>
<td>43</td>
</tr>
<tr>
<td>5.5.2 Generalization</td>
<td>44</td>
</tr>
<tr>
<td>5.5.3 Cost function. Alternative 2 and Markov Chain Model</td>
<td>46</td>
</tr>
<tr>
<td>5.5.4 The Second Order Estimated Cost Function with Markov Chain</td>
<td>47</td>
</tr>
<tr>
<td>5.5.5 Connection of Multistage Graphs and the Problem</td>
<td>48</td>
</tr>
<tr>
<td>5.6 More General Model</td>
<td>51</td>
</tr>
<tr>
<td>5.6.1 The Agent is Faster Than Target</td>
<td>51</td>
</tr>
<tr>
<td>5.6.2 Obstacles in Path</td>
<td>51</td>
</tr>
<tr>
<td>5.6.3 Target Direction</td>
<td>52</td>
</tr>
<tr>
<td>5.7 Conclusion and Future Direction</td>
<td>52</td>
</tr>
<tr>
<td>6 BEST TARGET SELECTION</td>
<td>53</td>
</tr>
<tr>
<td>6.1 Introduction</td>
<td>53</td>
</tr>
<tr>
<td>6.2 Maximize Probability of Attacking the Most Valuable Target</td>
<td>54</td>
</tr>
<tr>
<td>6.2.1 Best Target Strategy</td>
<td>55</td>
</tr>
<tr>
<td>6.2.1.1 Probability of Attacking $j^{th}$ Most Valuable Target</td>
<td>56</td>
</tr>
<tr>
<td>6.2.1.2 Mean Rank of the Attacked Target</td>
<td>58</td>
</tr>
<tr>
<td>6.2.1.3 Mean Number of Examined Targets</td>
<td>58</td>
</tr>
<tr>
<td>6.2.2 Results of Best Target Strategy</td>
<td>59</td>
</tr>
<tr>
<td>6.2.3 Best Target Strategy with Threshold</td>
<td>59</td>
</tr>
</tbody>
</table>
6.3 Maximize Mean Rank of the Attacked Target .......................... 61
  6.3.1 Mean Value Strategy ............................................. 61
  6.3.2 Results of Mean Value Strategy ............................... 63
6.4 Number of Targets is a Random Variable .......................... 63
6.5 Target Strategy with Sampling-The Learning Agent ............... 66
6.6 Multiple Agents ..................................................... 70
  6.6.1 Agents as a Pack ................................................ 70
  6.6.2 Separate Agents ................................................ 73
    6.6.2.1 Separate Agents without Communication .............. 74
    6.6.2.2 Separate Agents with Communication .................. 75
    6.6.2.3 Separate Agents Comparison ............................ 76
  6.6.3 Multiple Agent Strategy-Discussion ......................... 77
6.7 Dynamic Threshold ................................................ 77
6.8 Conclusion .......................................................... 78

7 CONCLUDING REMARKS AND FUTURE RESEARCH ....................... 79
  7.1 Summary .......................................................... 79
  7.2 Future Research .................................................. 79

REFERENCES .............................................................. 81

BIOGRAPHICAL SKETCH .................................................. 86
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3–1</td>
<td>The number of clusters for different values of $\beta$</td>
<td>26</td>
</tr>
<tr>
<td>3–2</td>
<td>The number of clusters as a function of $\beta$ for the iris data</td>
<td>28</td>
</tr>
<tr>
<td>3–3</td>
<td>Percentage of correct classification of iris data</td>
<td>28</td>
</tr>
<tr>
<td>3–4</td>
<td>The average number of clusters for various $k$ using a fixed $\beta = 2.5$</td>
<td>28</td>
</tr>
<tr>
<td>3–5</td>
<td>The average number of clusters for various $k$ using a fixed $\beta = 5.0$</td>
<td>29</td>
</tr>
<tr>
<td>3–6</td>
<td>The average number of clusters for various $k$ using a fixed $\beta = 10.5$</td>
<td>29</td>
</tr>
<tr>
<td>6–1</td>
<td>Significant results of the basic best target strategy</td>
<td>59</td>
</tr>
<tr>
<td>6–2</td>
<td>Empirical results of the best target strategy with threshold at upper 1-percent tail</td>
<td>60</td>
</tr>
<tr>
<td>6–3</td>
<td>Empirical results of the best target strategy with threshold at upper 2.5-percent tail</td>
<td>61</td>
</tr>
<tr>
<td>6–4</td>
<td>Empirical results of the best target strategy with threshold at upper 30-percent tail</td>
<td>61</td>
</tr>
<tr>
<td>6–5</td>
<td>$k$ values to minimize mean rank of attacked targets</td>
<td>62</td>
</tr>
<tr>
<td>6–6</td>
<td>Simulation results of the best target strategy</td>
<td>63</td>
</tr>
<tr>
<td>6–7</td>
<td>Simulation results of the mean target strategy</td>
<td>63</td>
</tr>
<tr>
<td>6–8</td>
<td>Simulation results with number of targets poisson distributed, mean $n$</td>
<td>65</td>
</tr>
<tr>
<td>6–9</td>
<td>Simulation results with number of targets normally distributed, mean $n$ and standard deviation $0.2n$</td>
<td>65</td>
</tr>
<tr>
<td>6–10</td>
<td>Simulation results with number of targets uniformly distributed in $[0.5n, 1.5n]$</td>
<td>66</td>
</tr>
<tr>
<td>6–11</td>
<td>Performance summary of best target strategy and mean value strategy when $n$ varies; values are in percentage drop compared to when $n$ is fixed</td>
<td>66</td>
</tr>
</tbody>
</table>
6–12 Simulation results with expected number of targets, $n$, updated 90-percent into the mission .......................................................... 67

6–13 Simulation results with expected number of targets, $n$, updated near the end of the mission. $n$ may be updated downward, but not upward. . . . 67

6–14 Simulation results of the target strategy with sampling .................. 69

6–15 Simulation results of the target strategy with $m$ agents in a pack. Target values are uniform on $[0,1000]$......................................................... 73

6–16 Simulation results of the target strategy with $m$ agents on separate missions with no communication between them. Target values are uniform on $[0,1000]$. ................................................................. 75

6–17 Simulation results of the target strategy with $m$ agents on separate missions with communication between them. Target values are uniform on $[0,1000]$. ................................................................. 75

6–18 Simulation results of the target strategy with $m$ agents on separate missions with communication between them. Uncommitted agents are allowed to evaluate targets in other unsearched partitions. ............... 76

6–19 Simulation results using the dynamic threshold strategy .......... 78


<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3–1</td>
<td>K-Means algorithm</td>
<td>14</td>
</tr>
<tr>
<td>3–2</td>
<td>Entropy K-means algorithm</td>
<td>22</td>
</tr>
<tr>
<td>3–3</td>
<td>Generic MST algorithm</td>
<td>24</td>
</tr>
<tr>
<td>3–4</td>
<td>Kruskal MST algorithm</td>
<td>25</td>
</tr>
<tr>
<td>3–5</td>
<td>Graph clustering algorithm</td>
<td>25</td>
</tr>
<tr>
<td>4–1</td>
<td>Algorithm for dimension reduction</td>
<td>33</td>
</tr>
<tr>
<td>5–1</td>
<td>Target and agent boundaries</td>
<td>40</td>
</tr>
<tr>
<td>5–2</td>
<td>Region</td>
<td>43</td>
</tr>
<tr>
<td>5–3</td>
<td>Multistage graph representation 1</td>
<td>49</td>
</tr>
<tr>
<td>5–4</td>
<td>Multistage graph representation 2</td>
<td>51</td>
</tr>
<tr>
<td>5–5</td>
<td>Region with obstacles</td>
<td>51</td>
</tr>
<tr>
<td>5–6</td>
<td>Using the 8 zimuths</td>
<td>52</td>
</tr>
<tr>
<td>6–1</td>
<td>Racetrack search of battlespace with $n$ targets</td>
<td>55</td>
</tr>
<tr>
<td>6–2</td>
<td>Plots of probabilities of attacking the $j^{th}$ best target for two proposed strategies, $n=100$</td>
<td>62</td>
</tr>
<tr>
<td>6–3</td>
<td>$m$ agents performing racetrack search of battlespace with $n$ targets</td>
<td>70</td>
</tr>
<tr>
<td>6–4</td>
<td>$m$ agents on separate missions of a equally partitioned battlespace performing racetrack search for $n$ targets</td>
<td>74</td>
</tr>
</tbody>
</table>
Many real word problems in engineering, mathematics and other areas are often solved on the basis of measured data, given certain conditions and assumptions. In solving these problems, we are concerned with solution properties like existence, uniqueness, and stability. A problem for which any one of the above three conditions is not met is called an ill-posed problem. This problem is caused by incomplete and/or noisy data, where noise can be referred to as any discrepancy between the measured and true data. Equally important in the data analysis is the effective interpretation of the results. Data data sets to be analyzed usually have several attributes and the domain of each attribute can be very large. Therefore results obtained in these high dimensions are very difficult to interpret.

Several solution methods exist to handle this problem. One of these methods is the maximum entropy method. We present in this dissertation entropy optimization methods and give applications in modelling real life problems, specifically in mining numerical data. Best target selection and the application of entropy in modelling path planning problems are also presented in this research.
CHAPTER 1
INTRODUCTION

Many real word problems are often solved on the basis of measured data, given certain conditions and assumptions. Several diverse areas where data analysis is involved include the government and military systems, medicine, sports, finance, geographical information systems, etc. [1, 35, 31]. Solutions to these problems involve in most cases the understanding of the structural properties (pattern discovery) of the data set. In pattern discovery, we look for a model that reflects the structure of the data which we hope will reflect the structure of the generating process. Thus given a data set, we want to extract as much essential structure as possible without modelling any of its accidental structures (e.g., noise and sampling artifacts). We want to maximize the information content of all parameters. A method for achieving the objectives above is entropy optimization [8] in which entropy minimization maximizes the amount of evidence supporting each parameter, while minimizing the uncertainty in the sufficient statistic and the cross entropy between the model and the data.

1.1 Data Mining

Data mining or knowledge discovery in databases (KDD) is a non-trivial process that seeks to identify valid, useful and ultimately understandable patterns in data [19]. KDD consists of several steps. These steps include preparation of data, pattern search, knowledge evaluation and refinement. Data mining is a very important process in the KDD process since this is where specific algorithms are employed for extracting patterns from the data. Data mining can therefore be considered as a set of extraction processes of knowledge starting from data contained in a base of data [9].

Data mining techniques include a variety of methods. These methods generally fall into one of two groups: predictive methods and descriptive methods. The
predictive methods involve the use of some variables to predict unknown or future values of other variables. They are usually referred to as classification. The descriptive methods seek human-interpretable patterns that describe the data and are referred to as clustering. However, authors including [6], have phrased these methods in terms of six tasks: classification, estimation, prediction, clustering, market basket analysis and description. These different tasks of data mining are described below.

1.1.1 Classification

Classification is the process of classifying the categories for an unknown data set. The data set to be classified is divided into two parts, a training set and a testing set. A characteristic of classification is that there is a well defined set of categories or classes. Classification is sometimes referred to as supervised learning. The machine algorithm to applied is trained using the training set (pre-classified examples) until the error bound is decreased to some threshold. This process is done iteratively, and repeated many times with different parameter values in some randomized order of the input. Once an optimal process design is obtained, the testing data unknown to the algorithm are used on the algorithm.

1.1.2 Clustering

Clustering is concerned with the grouping of unlabelled feature vectors into clusters such that samples within a cluster are more similar to each other than samples belonging to different clusters. The clustering problem can be stated as follows: given a set of \(n\) data points \((x_1, \ldots, x_k)\) in \(d\) dimensional space \(R^d\) and an integer \(k\), partition the set of data into \(k\) disjoint clusters so as to minimize some loss function.

1.1.3 Estimation

Estimation is a task of data mining that is generally applied to continuous data. Similar to classification, estimation has the feature that the data record is rank ordered, thereby making it easy to work with a part of the data that has the desired attributes. Neural network methods are well suited for estimation.
1.1.4 Prediction

This data mining task sometimes grouped with classification allows objects to be classified based on predicted future behavior or values. In prediction, historical data are often used to build the model that predicts future behaviors.

1.1.5 Description

The ability to understand complicated data bases is the descriptive task of data mining. Data description gives insight on the different attributes of the data.

The remainder of the thesis proceeds as follows. Chapter 2 discusses entropy optimization. Definition of entropy is given and a discussion of the rationale for its use in the area of data mining is provided. Chapter 3 develops entropy tools for data mining and apply them to data clustering. Problems of high dimensionality in data mining is the focus of chapter 4. An entropy dimension reduction method is given in this chapter. In chapter 5 and 6, we apply the predictive ability of our entropy optimization methods to model a path planning problem and best target problem selection respectively. Chapter 7 summarizes the work and proposes extensions.
CHAPTER 2
ENTROPY OPTIMIZATION

2.1 Introduction

Many real world problems are often solved on the basis of measured data, given certain conditions and assumptions. Several diverse areas where data analysis is involved include the government and military systems, medicine, sports, finance, geographical information systems, etc. [1, 35, 31]. Solutions to these problems involve in most cases the understanding of the structural properties (pattern discovery) of the data set. In pattern discovery, we look for a model that reflects the structure of the data which we hope will reflect the structure of the generating process. Thus given a data set, we want to extract as much essential structure as possible without modelling any of its accidental structures (e.g., noise and sampling artifacts). We want to maximize the information content of all parameters. A method for achieving the objectives above is entropy optimization [8] in which entropy minimization maximizes the amount of evidence supporting each parameter, while minimizing the uncertainty in the sufficient statistic and the cross entropy between the model and the data.

This chapter is organized as follows. In the next section, we provide some background on entropy and provide some rationale for its use in the area of data mining.

2.2 A Background on Entropy Optimization

The concept of entropy was originally developed by the physicist Rudolf Clausius around 1865 as a measure of the amount of energy in a thermodynamic system as cited in Fang at al.[15]. This concept was later extended through the development of statistical mechanics. It was first introduced into information theory in 1948 by Claude Shannon as cited in shore et al.[45].
2.2.1 Definition of Entropy

Entropy can be defined as a measure of the expected information content or uncertainty of a probability distribution. It is also defined as the degree of disorder in a system or the uncertainty about a partition [45, 29].

Let $E_i$ stand for an event and $p_i$ the probability that event $E_i$ occurs. Let there be $n$ such events $E_1, ..., E_n$ with probabilities $p_1, ..., p_n$ adding up to 1. Since the occurrence of events with smaller probability yields more information since they are least expected, a measure of information $h$ should be a decreasing function of $p_i$. Claude Shannon proposed a log function $h(p_i)$ to express information. This function is given as

$$h(p_i) = \log_2 \frac{1}{p_i}$$

(2-1)

which decreases from infinity to 0, for $p_i$ ranging from 0 to 1. This function reflects the idea that the lower the probability of an event to occur, the higher the amount of information in the message stating that the event occurred.

From these $n$ information values $h(p_i)$, the expected information content $H$ called entropy is derived by weighting the information values by their respective probabilities.

$$H = -\sum_{i=1}^{n} p_i \log_2 p_i$$

(2-2)

Since $-p_i \log_2 p_i \geq 0$ for $0 \leq p_i \leq 1$ it follows from (2-2) that $H \geq 0$, where $H = 0$ iff one of the $p_i$ equals 1; all others are then equal to zero. Hence the notation $0 \ln 0 = 0$.

Definition 2.1. Given a discrete random variable $X$ taking on values in the finite set $\{x_1, ..., x_n\}$ with probabilities $\mathbf{p} = (p_1, ..., p_n)$, we define the Shannon entropy to be

$$H(X) = H(\mathbf{p}) = -k \sum_{i=1}^{n} p_i \ln p_i$$

(2-3)
where \( k \) depends on the unit used and is usually set to unity. The convention \( 0 \ln 0 \) applies also.

The Shannon entropy has the following desirable properties [15]:

1. Shannon measure is nonnegative and concave in \( p_1, ..., p_n \).
2. The measure does not change with inclusion of a zero-probability outcome.
3. The entropy of a probability distribution representing a completely certain outcome is 0 and the entropy of any probability distribution representing uncertain outcome is positive.
4. Given a fixed number of outcomes, the maximum possible entropy is that of the uniform distribution.
5. The entropy of the joint distribution of two independent distributions is the sum of the individual entropies.
6. The entropy of the joint distribution of two dependent distributions is no greater than the sum of the two individual entropies.
7. Since entropy only depend on the unordered probabilities and not on \( X \), it is invariant to both shift and scale i.e. \( H(aX + b) = H(X) \) for \( a \neq 0 \) and for all \( b \).

**Definition 2.2.** The differential entropy of a continuous random variable \( X \) with probability density function \( p(x) \), is

\[
H(X) = -\int p(x) \ln p(x) dx
\]  

(2-4)

Again \( 0 \ln 0 \) is taken to be 0.

The differential entropy does not retain all of the useful properties of the discrete entropy. The differential entropy is not invariant to transform. Its value could also be negative.

### 2.2.2 Choosing A Probability Distribution

E.T. Jaynes in 1957 [18] introduced the principle of maximum entropy. The Maximum Entropy Principle (MaxEnt) is stated as follows:

*Out of all possible distributions that are consistent with available information (constraints), choose the one that has maximum entropy.*

Using this principle, we give an entropy formulation for an associated problem. Let \( X \) denote a random variable with \( n \) possible outcomes \( x_1, ..., x_n \). Let \( p = (p_1, ..., p_n) \)
denote their respective probabilities, respectively.

Let \( r_1(X), \ldots, r_m(X) \) be \( m \) functions of \( X \) with known expected values \( E(r_1(X)) = a_1, \ldots, E(r_m(X)) = a_m \). The MaxEnt formulation is as follows:

\[
\begin{align*}
\max & \quad H(X) = -\sum_{i=1}^{n} p_i \ln p_i \\
\text{s.t.} & \quad \sum_{i=1}^{n} (p_i r_j(x_i)) = a_j, j = 1, \ldots, m \\
& \quad \sum_{i=1}^{n} p_i = 1 \\
& \quad p_i \geq 0, i = 1, \ldots, n
\end{align*}
\]

This is a concave optimization problem with linear constraints. The solution to this optimization problem is obtained by applying the method of Lagrange multipliers. The form of the solution is exponential. The Lagrangian of the optimization problem is

\[
L(\lambda, p) = -\sum_{x \in X} (p(x) \ln p(x)) + \lambda_0 (\sum_{i=1}^{n} p(x) - 1) + \sum_{i=1}^{m} \lambda_i (\sum_{x \in X} p(x) r_i(x) = a_i) \tag{2–5}
\]

Taking the gradient with respect to \( p(x) \) we get

\[
\frac{\partial L(\lambda, p)}{\partial p(x)} = \ln p(x) - 1 + \lambda_0 + \sum_{i=1}^{m} \lambda_i r_i(x) \tag{2–6}
\]

\[
\Rightarrow p(x) = e^{-1+\lambda_0+\sum_{i=1}^{m} \lambda_i r_i(x)}, \text{for all } x \in X
\]

where \( \lambda_0, \lambda_1, \ldots, \lambda_m \) are chosen so that the constraints are satisfied. In absence of the moment constraints, that is

\[
\begin{align*}
\max & \quad H(X) = -\sum_{i=1}^{n} p_i \ln p_i \\
& \quad \sum_{i=1}^{n} p_i = 1 \\
& \quad p_i \geq 0, i = 1, \ldots, n
\end{align*}
\]
The distribution with the maximum entropy is the uniform distribution with \( p_i = 1/n \).

Example:

Suppose you are given data on 3 routes from A to B that you usually take to work. The cost of each route in dollars is 1, 2, and 3. The average cost is $1.75. What is the maximum entropy distribution describing your choice of route for a particular day?

The solution to the above example can be formulated and solved as follows.

\[
\begin{align*}
\max & \quad -(p_1 \ln p_1 + p_2 \ln p_2 + p_3 \ln p_3) \\
\text{s.t.} & \quad 1 \ln p_1 + 2 \ln p_2 + 3 \ln p_3 = 1.75 \\
& \quad p_1 + p_2 + p_3 = 1 \\
& \quad p_1 \geq 0, p_2 \geq 0, p_3 \geq 0
\end{align*}
\]

The range of values for \( p_i \) is

\[
\begin{align*}
0 & \leq p_2 \leq 0.75 \\
0 & \leq p_3 \leq 0.375 \\
0.25 & \leq p_1 \leq 0.625
\end{align*}
\]

The maximum entropy solution is \( p_1 = 0.466, p_2 = 0.318, p_3 = 0.216 \).

### 2.2.3 Prior Information

Suppose that in addition to the moment constraints, we have a priori probability distribution \( p^0 \) that we think our probability distribution \( p \) should be close to. How close should \( p \) be to \( p^0 \)? A measure of this closeness or deviation is the Kullback-Liebler distance or the measure of relative (cross) entropy. This distance measure was introduced in 1951 by S. Kullback and R.A. Leibler \[29]\.

With relative entropy as a measure of deviation, the Kullback-Leibler minimum entropy principle, or MinEnt is stated as follows:
Out of all possible distributions that are consistent with available information (constraints), choose the one that minimizes the cross-entropy with respect to the a priori distribution.

The MinEnt Formulation is as follows:

\[
\min D(p||p^0) = H(p) = \sum_{i=1}^{n} p_i \ln \frac{p_i}{p_i^0}
\]

s.t. \[\sum_{i=1}^{n} (p_i)r_j(x_i) = a_j, j = 1, ..., m\]
\[\sum_{i=1}^{n} p_i = 1\]
\[p_i \geq 0, i = 1, ..., n\]

If no a priori distribution is given, then we can use the maximum entropy distribution. This leads to the uniform distribution \(u\) as the a priori distribution. We then obtain:

\[
D(p||u) = \sum_{i=1}^{n} p_i \ln \frac{p_i}{1/n} = \ln n + \sum_{i=1}^{n} p_i \ln \frac{p_i}{p_i^0}
\]

MaxEnt is a special case of MinEnt. Thus minimizing the cross-entropy with respect to the uniform distribution is equivalent to maximizing entropy.

2.2.4 Minimum Cross Entropy Principle

Several authors [49, 51] have explored the use of cross-entropy and have shown rigorously that the Jaynes principle of maximum entropy and Kullback’s principle of minimum cross-entropy provides a correct method of inductive inference when new information is given in the form of expected value. Given a distribution \(p^0\) and some new information in the form of constraints:

\[
\int p(x)c_k(x)dx \geq 0, k = 1, 2, ..., m
\]

then the new distribution \(p(x)\), which incorporates this information in the least biased way one and which is arrived at in a way that does not not lead to any
inconsistencies or contradictions, is the one obtained from minimizing

\[ \int p(x) \ln \frac{p(x)}{p^0(x)} dx \]  

(2–8)

This is the minimum cross-entropy principle \([49, 50, 51]\). These authors also showed that maximum entropy principle is a special case of minimum cross-entropy based as outlined below. Suppose that we are trying to estimate the probability of finding a system in state \(x\). If we know that only \(n\) discrete states are possible, then we already know the some information about the system. This information is expressed by \(p_i^0 = 1/n \forall i\). If we obtain more information in the form of the inequality given in 2–7, then the correct estimate of the probability of the system being in state \(i\) is given by minimizing:

\[ D(p||p^0) = p_i \ln \frac{p_i}{p_i^0} = p_i \ln p_i - \ln n \]

which is equivalent to maximizing the entropy

\[ H = - \sum_{i=1}^{n} p_i \log_2 p_i \]

This principle is used in developing models in chapters five and six.

### 2.3 Applications of Entropy Optimization

Entropy optimization has successfully been applied in many scientific and engineering problems. Example applications of entropy optimization include transportation planning (Fang and Tsao, 1993)\([16]\), regional planning (Wilson, 1970) \([55]\), investment portfolio optimization (Kapur et al., 1989)\([29]\), image reconstruction (Burch et al., 1984)\([21]\), and pattern recognition (Tou and Gonzalez, 1974)\([53]\)

#### Rationale for Using Entropy Optimization

Data to be clustered are usually incomplete. Solution using the data should incorporate and be consistent with all
relevant data and maximally noncommittal with regard to unavailable data. The solution may be viewed as a procedure for extracting information from data. The information comes from two sources: the measured data and the assumption about the unavailable ones because of data incompleteness. Making an assumption means artificially adding information which may be true or false. Maximum entropy implies that the added information is minimal. A maximum entropy solution has the least assumption and is maximally noncommittal.

In the next chapter we develop an entropy minimization method and apply it to data clustering.
3.1 Introduction

Data clustering and classification analysis is an important tool in statistical analysis. Clustering techniques find applications in many areas including pattern recognition and pattern classification, data mining and knowledge discovery, data compression and vector quantization. Data clustering is a difficult problem that often requires the unsupervised partitioning of the data set into clusters. In the absence of prior knowledge about the shape of the clusters, similarity measures for a clustering technique are hard to specify. The quality of a good cluster is application dependent since there are many methods for finding clusters subject to various criteria which are both ad hoc and systematic [28].

Another difficulty in using unsupervised methods is the need for input parameters. Many algorithms, especially the K-means and other hierarchical methods [26] require that the initial number of clusters be specified. Several authors have proposed methods that automatically determine the number of clusters in the data [22, 29, 25]. These methods use some form of cluster validity measures like variance, a priori probabilities and the difference of cluster centers. The obtained results are not always as expected and are data dependent [54]. Some criteria from information theory have also been proposed. The Minimum Descriptive Length (MDL) criteria evaluates the compromise between the likelihood of the classification and the complexity of the model [48].

In this chapter, we develop a framework for clustering by learning from the structure of the data. Learning is accomplished by randomly applying the K-means algorithm via entropy minimization (KMEM) multiple times on the data. The
(KMEM) enables us to overcome the problem of knowing the number of clusters \textit{a priori}. Multiple applications of the KMEM allow us to maintain a similarity measure matrix between pairs of input patterns. An entry $a_{ij}$ in the similarity matrix gives the proportion of times input patterns $i$ and $j$ are co-located in a cluster among $N$ clusterings using KMEM. Using this similarity matrix, the final data clustering is obtained by clustering a sparse graph of this matrix.

The contribution of this work is the incorporation of entropy minimization to estimate an approximate number of clusters in a data set based on some threshold and the use of graph clustering to recover the expected number of clusters.

This chapter is organized as follows: In the next section, we provide some background on the K-means algorithm. A brief discussion of entropy that will be necessary in developing our model is presented in section 3.3. The proposed K-Means via entropy minimization is outlined in section 4. The graph clustering approach is presented in section 5. The results of our algorithms are discussed in section 3.6. We conclude briefly in section 3.7.

\section*{3.2 K-Means Clustering}

The K-means clustering [33] is a method commonly used to partition a data set into $k$ groups. In the K-means clustering, we are given a set of $n$ data points (patterns) $(x_1, ..., x_k)$ in $d$ dimensional space $R^d$ and an integer $k$ and the problem is to determine a set of points (centers) in $R^d$ so as to minimize the square of the distance from each data point to its nearest center. That is find $k$ centers $(c_1, ..., c_k)$ which minimize:

\begin{equation}
J = \sum_k \sum_{x \in C_k} |d(x, c_k)|^2
\end{equation}

where the $C$'s are disjoint and their union covers the data set. The K-means consists of primarily two steps:

1) The assignment step where based on initial $k$ cluster centers of classes, instances
are assigned to the closest class.

2) The re-estimation step where the class centers are recalculated from the instances assigned to that class.

These steps are repeated until convergence occurs; that is when the re-estimation step leads to minimal change in the class centers. The algorithm is outlined in figure 3–1.

<table>
<thead>
<tr>
<th>The K-means Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input:</td>
</tr>
<tr>
<td>$P = { p_1,..., p_n }$ (points to be clustered)</td>
</tr>
<tr>
<td>$k$ (number of clusters)</td>
</tr>
<tr>
<td>Output:</td>
</tr>
<tr>
<td>$C = { c_1,..., c_k }$ (cluster centers)</td>
</tr>
<tr>
<td>$m : P \rightarrow {1,...,k}$ (cluster membership)</td>
</tr>
<tr>
<td>Procedure K-means:</td>
</tr>
<tr>
<td>1. Initialize $C$ (random selection of $P$).</td>
</tr>
<tr>
<td>2. For each $p_i \in P$, $m(p_i) = \text{arg min}_{j \in k} \text{distance}(p_i, c_j)$.</td>
</tr>
<tr>
<td>3. If $m$ has not changed, stop, else proceed.</td>
</tr>
<tr>
<td>4. For each $i \in {1,...,k}$, recompute $c_i$ as a center of ${ p \mid m(p) = i }$.</td>
</tr>
<tr>
<td>5. Go to step 2.</td>
</tr>
</tbody>
</table>

Figure 3–1: K-Means algorithm

Several distance metrics like the Manhattan or the Euclidean are commonly used. In this research, we consider the Euclidean distance metric. Issues that arise in using the K-means include: shape of the clusters, choosing the number of clusters, the selection of initial cluster centers which could affect the final results and degeneracy.

There are several ways to select the initial cluster centers. Given the number of clusters $k$, you randomly select $k$ values from the data set. (This approach was used in our analysis). You could also generate $k$ seeds as the initial cluster centers, or manually specify the initial cluster centers. Degeneracy arises when the algorithm is
trapped in a local minimum thereby resulting in some empty clusters. In this paper we intend to handle the last threes problem via entropy optimization.

3.3 An Overview of Entropy Optimization

The concept of entropy was originally developed by the physicist Rudolf Clausius around 1865 as a measure of the amount of energy in a thermodynamic system [15]. This concept was later extended through the development of statistical mechanics. It was first introduced into information theory in 1948 by Claude Shannon [45]. Entropy can be understood as the degree of disorder of a system. It is also a measure of uncertainty about a partition [45, 29].

The philosophy of entropy minimization in the pattern recognition field can be applied to classification, data analysis, and data mining where one of the tasks is to discover patterns or regularities in a large data set. The regularities of the data structure are characterized by small entropy values, while randomness is characterized by large entropy values [29]. In the data mining field, the most well known application of entropy is information gain of decision trees. Entropy based discretization recursively partitions the values of a numeric attribute to a hierarchy discretization. Using entropy as an information measure, one can then evaluate an attribute’s importance by examining the information theoretic measures [29].

Using entropy as an information measure of the distribution data in the clusters, we can determine the number of clusters. This is because we can represent data belonging to a cluster as one bin. Thus a histogram of these bins represents cluster distribution of data. From entropy theory, a histogram of cluster labels with low entropy shows a classification with high confidence, while a histogram with high entropy shows a classification with low confidence.
3.3.1 Minimum Entropy and Its Properties

Recall that the Shannon Entropy is defined as

$$H(X) = -\sum_{i=1}^{n} (p_i \ln p_i)$$  \hspace{1cm} (3–2)$$

where $X$ is a random variable with outcomes $1, 2, ..., n$ and associated probabilities $p_1, p_2, ..., p_n$.

Since $-p_i \ln p_i \geq 0$ for $0 \leq p_i \leq 1$ it follows from (5–6) that $H(X) \geq 0$, where $H(X) = 0$ iff one of the $p_i$ equals 1; all others are then equal to zero. Hence the notation $0 \ln 0 = 0$. For continuous random variable with probability density function $p(x)$, entropy is defined as

$$H(X) = -\int p(x) \ln p(x) dx$$  \hspace{1cm} (3–3)$$

This entropy measure tells us whether one probability distribution is more informative than the other. The minimum entropy provides us with minimum uncertainty, which is the limit of the knowledge we have about a system and its structure [45]. In data classification, for example the quest is to find minimum entropy [45]. The problem of evaluating a minimal entropy probability distribution is the global minimization of the Shannon entropy measure subject to the given constraints. This problem is known to be NP-hard [45].

Two properties of minimal entropy which will be fundamental in the development of KMEM model are concentration and grouping [45]. Grouping implies moving all the probability mass from one state to another, that is, reduce the number of states. This reduction can decrease entropy.

**Proposition 3.1.** Given a partition $\beta = [B_a, B_b, A_2, A_3, ..., A_N]$, we form the partition $\bar{\beta} = [A_1, A_2, A_3, ..., A_N]$ obtained by merging $B_a$ and $B_b$ into $A_1$, where $p_a = P(B_a)$, $p_b = P(B_b)$ and $p_i = P(A_i)$, we maintain that

$$H(\bar{\beta}) \leq H(\beta)$$  \hspace{1cm} (3–4)$$
Proof. The function $\varphi(p) = -p \ln p$ is convex. Therefore for $\lambda > 0$ and 

\[ p_1 - \lambda < p_1 < p_2 < p_2 + \lambda \]

we have

\[ \varphi(p_1 + p_2) < \varphi(p_1 - \lambda) + \varphi(p_2 + \lambda) < \varphi(p_1) + \varphi(p_2) \]  \hspace{1cm} (3–5)

Clearly,

\[ H(\beta) - \varphi(p_a) - \varphi(p_b) = H(\bar{A}) - \varphi(p_a + p_b) \]

because each side equals the contribution to $H(\beta)$ and $H(\bar{A})$ respectively due the to 

common elements of $\beta$ and $\bar{A}$.

Hence, (3–4) follows from (3–5).

Concentration implies moving probability mass from a state with low probability to 

a state with high probability. Whenever this move occurs, the system becomes less 

uniform and thus entropy decreases.

**Proposition 3.2.** Given two partitions $\beta = [b_1, b_2, A_3, A_4, \ldots A_N]$ and 

$\bar{A} = [A_1, A_2, A_3, \ldots A_N]$ that have the same elements except the first two.

We maintain that if 

\[ p_1 = P(A_1), \ p_2 = P(A_2) \] \text{ with } \ p_1 < p_2 \text{ and } (p_1 - \lambda) = P(b_1) \leq (p_2 + \lambda) = P(b_2),

then

\[ H(\beta) \leq H(\bar{A}) \]  \hspace{1cm} (3–6)

Proof. Clearly,

\[ H(\bar{A}) - \varphi(p_1) - \varphi(p_2) = H(\beta) - \varphi(p_1 - \lambda) - \varphi(p_2 + \lambda) \]

because each side equals the contribution to $H(\beta)$ and $H(\bar{A})$ respectively due to the 

common elements of $\bar{A}$ and $\beta$

Hence, (3–6) follows from (3–5).
3.3.2 The Entropy Decomposition Theorem

Another attractive property of entropy is the way in which aggregation and disaggregation are handled [18]. This is because of the property of additivity of entropy. Suppose we have $n$ outcomes denoted by $X = \{x_1, ..., x_n\}$, with probability $p_1, ..., p_n$. Assume that these outcomes can be aggregated into a smaller number of sets $C_1, ..., C_K$ in such a way that each outcome is in only one set $C_k$, where $k = 1, ..., K$.

The probability that outcomes are in set $C_k$ is

$$p_k = \sum_{i \in C_k} p_i$$  \hfill (3–7)

The entropy decomposition theorem gives the relationship between the entropy $H(X)$ at level of the outcomes as given in (5–6) and the entropy $H_0(X)$ at the level of sets. $H_0(X)$ is the between group entropy and is given by:

$$H_0(X) = - \sum_{k=1}^{K} (p_k \ln p_k)$$  \hfill (3–8)

Shannon entropy (5–6) can then be written as

$$H(X) = - \sum_{i=1}^{n} p_i \ln p_i$$

$$= - \sum_{k=1}^{K} \sum_{i \in C_k} p_i \ln p_i$$

$$= - \sum_{k=1}^{K} p_k \sum_{i \in C_k} \frac{p_i}{p_k} (\ln p_i + \ln \frac{p_k}{p_i})$$

$$= - \sum_{k=1}^{K} (p_k \ln p_k) - \sum_{k=1}^{K} p_k \sum_{i \in C_k} \frac{p_i}{p_k} \ln \frac{p_i}{p_k}$$

$$= H_0(X) + \sum_{k=1}^{K} p_k H_k(X)$$  \hfill (3–9)

where

$$H_k(X) = - \sum_{i \in C_k} \frac{p_i}{p_k} \ln \frac{p_i}{p_k}$$  \hfill (3–10)
A property of this relationship is that \( H(X) \geq H_0(X) \) because \( p_k \) and \( H_k(X) \) are nonnegative. This means that after data grouping, there cannot be more uncertainty (entropy) than there was before grouping.

### 3.4 The K-Means via Entropy Model

In this section we outline the K-means via entropy minimization. The method of this section enables us to perform learning on the data set, in order to obtain the similarity matrix and to estimate a value for the expected number of clusters based on the clustering requirements or some threshold.

#### 3.4.1 Entropy as a Prior Via Bayesian Inference

Given a data set represented as \( X = \{x_1, \ldots, x_n\} \), a clustering is the partitioning of the data set to get the clusters \( \{C_j, j = 1, \ldots, K\} \), where \( K \) is usually less than \( n \). Since entropy measures the amount of disorder of the system, each cluster should have a low entropy because instances in a particular cluster should be similar. Therefore our clustering objective function must include some form of entropy. A good minimum entropy clustering criterion has to reflect some relationship between data points and clusters. Such relationship information will help us to identify the meaning of data, i.e. the category of data. Also, it will help to reveal the components, i.e. clusters and components of mixed clusters. Since the concept of entropy measure is identical to that of probabilistic dependence, an entropy criterion measured on a posteriori probability would suffice. The Bayesian inference is therefore very suitable in the development of the entropy criterion.

Suppose that after clustering the data set \( X \), we obtain the clusters \( \{C_j, j = 1, \ldots, K\} \) by Bayes rule, the posterior probability \( P(C_j|X) \) is given as;

\[
P(C_j|X) = \frac{P(X|C_j)P(C_j)}{P(X)} \propto P(X|C_j)P(C_j)
\]

(3–11)

where \( P(X|C_j) \) given in (3–12) is the likelihood and measures the accuracy in clustering the data and the prior \( P(C_j) \) measures consistency with our background.
knowledge.

\[
P(X|C_j) = \prod_{x_i \in C_j} P(x_i|C_j) = e^{\sum_{x_i \in C_j} \ln p(x_i|C_j)}
\]  

(3–12)

By the Bayes approach, a classified data set is obtained by maximizing the posterior probability (3–11). In addition to three of the problems presented by the K-means which we would like to address: determining number of clusters, selecting initial cluster centers and degeneracy; a fourth problem is, the choice of the prior distribution to use in (3–11). We address these issues below.

### 3.4.2 Defining the Prior Probability

Generally speaking, the choice of the prior probability is quite arbitrary [56]. This is a problem facing everyone and no universal solution has been found. For our our application, we will define the prior as an exponential distribution, of the form:

\[
P(C_j) \propto e^{\beta \sum_{i=1}^{k} p_i \ln p_i}
\]  

(3–13)

where \( p_j = |C_j|/n \) is the prior probability of cluster \( j \), and \( \beta \geq 0 \) refers to a weighting of the \textit{a priori} knowledge. Hence forth, we call \( \beta \) the entropy constant.

### 3.4.3 Determining Number of Clusters

Let \( k^* \) be the final unknown number of clusters in our K-means algorithm (KMEN). After clustering, the entropy

\[
H(X) = - \sum_{i=1}^{k^*} p_i \ln p_i
\]

will be minimum based on the clustering requirement. From previous discussions, we know that entropy decreases as clusters are merged. Therefore if we start with some large number of clusters \( K > k^* \), our clustering algorithm will reduce \( K \) to \( k^* \) because clusters with probability zero will vanish. Note that convergence to \( k^* \) is guaranteed because the entropy of the partitions is bounded below by 0. A rule of thumb on the value of initial number of clusters is \( K = \sqrt{n} \) [17].
The KMEM Model

The K-Means algorithm works well on a data set that has spherical clusters. Since our model (KMEM) is based on the K-means, we make the assumption that the each cluster has Gaussian distribution with mean values $c_j, i = (1, ..., k)$ and constant cluster variance. Thus for any given cluster $C_j$,

$$P(x_i|C_j) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i - c_j)^2}{2\sigma^2}}$$

(3–14)

Taking natural log and omitting constants, we have

$$\ln P(x_i|C_j) = -\frac{(x_i - c_j)^2}{2\sigma^2}$$

(3–15)

Using equations (3–12) and (3–13), the posterior probability (3–11) now becomes:

$$P(C_j|X) \propto \exp \sum_{x_i \in C_j} \left( \ln p(x_i|C_j) \right) \exp \left[ \beta \sum_{i=1}^{k^*} p_i \ln p_i \right] \propto \exp(-E)$$

(3–16)

where $E$ is written as follows:

$$E = - \sum_{x_i \in C_j} \ln p(x_i|C_j) - \beta \sum_{i=1}^{k^*} p_i \ln p_i$$

(3–17)

If we now use equation (3–14), equation (3–17) becomes

$$E = \sum_{i=1}^{k^*} \sum_{x_i \in C_j} \frac{(x_i - c_j)^2}{2\sigma^2} - \beta \sum_{i=1}^{k^*} p_i \ln p_i$$

(3–18)

or

$$E = \sum_{i=1}^{k^*} \sum_{x_i \in C_j} \frac{(x_i - c_j)^2}{2\sigma^2} + \beta H(X)$$

(3–19)

Maximizing the posterior probability is equivalent to minimizing (5–20). Also, notice that since the entropy term in (5–20) is nonnegative, equation (5–20) is minimized if entropy is minimized. Therefore (5–20) is the required clustering criterion.

We note that when $\beta = 0$, $E$ is identical to the cost function of the K-Means clustering algorithm.
The Entropy K-means algorithm (KMEM) is given in figure 3–2. Multiple runs of KMEM are used to generate the similarity matrix. Once this matrix is generated, the learning phase is complete.

---

**Entropy K-means Algorithm**

1. Select the initial number of clusters $k$ and a value for the stopping criteria $\varepsilon$.

2. Randomly initialize the cluster centers $\theta_i(t)$, and the *a priori* probabilities $p_i$, $i=1,2,...,k$, $\beta$, and the counter $t=0$.

3. Classify each input vector $x_j$, $j=1,2,...,n$ to get the partition $C_i$ such that for each $x_j \in C_i$, $r=1,2,...,k$

   
   \[
   [x_j - \theta_i(t)]^2 - \frac{\beta}{n} \ln(p_i) \leq [x_j - \theta_i(t)]^2 - \frac{\beta}{n} \ln(p_i)
   \]

4. Update the cluster centers

   \[
   \theta_i(t+1) = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j
   \]

   and the *a priori* probabilities of clusters

   \[
   p_i(t+1) = \frac{|C_i|}{n}
   \]

5. Check for convergence; that is see if

   \[
   \max_i |\theta_i(t+1) - \theta_i(t)| < \varepsilon
   \]

   if it is not, update $t=t+1$ and go to step 3.

---

Figure 3–2: Entropy K-means algorithm

This algorithm iteratively reduces the numbers of clusters as some empty clusters will vanish.

### 3.5 Graph Matching

The rationale behind our approach for structure learning is that any pair of patterns that should be co-located in a cluster after clustering must appear together in the same cluster a majority of the time after $N$ applications of KMEM.

Let $G(V,E)$ be the graph of the similarity matrix where each input pattern is a vertex of $G$ and $V$ is the set of vertices of $G$. An edge between a pair of patterns $(i,j)$ exists if the entry $(i,j)$ in the similarity matrix is non-zero. $E$ is a collection of all the edges of $G$. Graph matching is next applied on the maximum spanning tree of the sparse graph $G'(V,E) \subset G(V,E)$. The sparse graph is obtained
by eliminating inconsistent edges. An inconsistent edge is an edge whose weight is less than some threshold \( \tau \). Thus a pattern pair whose edge is considered inconsistent is unlikely to be co-located in a cluster. To understand the idea behind the maximum spanning tree, we can consider the minimum spanning tree which can be found in many texts, for example [3] pages 278 and 520. The minimum spanning tree (MST) is a graph theoretic method, which determines the dominant skeletal pattern of points by mapping the shortest path of nearest neighbor connections [40]. Thus given a set of input patterns \( X = x_1, ..., x_n \) each with edge weight \( d_{i,j} \), the minimum spanning tree is an acyclic connected graph that passes through all input patterns of \( X \) with a minimum total edge weight. See section 3.5. The maximum spanning tree on the other hand is a spanning with a maximum total weight. Since all of the edge weight in the similarity matrix are nonnegative, we can negate these values and then apply the minimum spanning tree algorithm given in figure 3–4.

**Minimum Spanning Tree**

Minimum spanning trees are used in solving many real world problems. For example, consider a case of a network with \( V \) nodes with \( E \) undirected connections between nodes. This can be represented as a connected, undirected graph \( G = (V; E) \) containing \( V \) vertices and \( E \) edges. Now suppose that all the edges are weighted, i.e., for each edge \((u; v) \in E\) we have an associated weight \( w(u; v) \). A weight can be used to represent real world quantities such as cost of a wire, distance etc between two nodes in a network. A spanning tree is defined as a acyclic graph that connects all the vertices. A minimum spanning tree is a spanning tree with the minimum weight. Suppose we represent the spanning tree as \( T \subseteq E \), which connects all the vertices, and whose total length is \( w(T) \), then the minimum spanning tree is defined as,

\[
\min(w(T)) = \sum_{(u,v) \in T} w(u; v)
\]

(3–20)
Algorithm 1
1. $S \leftarrow 0$
2. while $S$ does not form a spanning tree
3. do find a safe-edge $(u, v)$ which can be added to $S$
4. $S \leftarrow S \cup (u, v)$
5. return $S$

Figure 3–3: Generic MST algorithm.

Generic MST algorithm. The book by Cormen et al. [11] gives a supported analysis of minimum spanning tree algorithms. The MST algorithm falls in the category of greedy algorithms. Greedy Algorithms are algorithms that make the best choice at each decision making step. In other words, at every step, greedy algorithms make the locally optimum choice and hope that it leads to a globally optimum solution. The greedy MST algorithm builds the tree step-by-step, incorporating the edge that causes minimum increase in the total weight at each step, without adding any cycles to the tree. Suppose there is a connected, undirected graph $G = (V; E)$ with the weight function $w$. While finding the minimum spanning tree for graph $G$, the algorithm manages at each step an edge-set $S$ which is some subset of the MST. At each step, edge $(u; v)$ is added to subset $S$ such that it does not violate the MST property of $S$. This makes $S \cup (u; v)$ a subset of the Minimum Spanning Tree. The edge which is added at each step is termed a ”safe edge”. The generic algorithm is given in figure 3–3.

There are two popular algorithms for computing the Minimum Spanning Tree, Prim’s algorithm and Kruskal’s algorithm (refer [11]). We used the Kruskal’s Algorithm in our analysis. Its description follows.

Kruskal’s algorithm for MST. Kruskal’s algorithm is an extension of the generic MST algorithm described in the preceding sub-section above. In the Kruskal’s algorithm the set $S$, which is a subset of the minimum spanning tree, is a forest. At each step, the Kruskal’s Algorithm finds the safe edge to be added as the edge with the minimum weight that connects two forests. Initially, the edges are sorted in the decreasing order
Algorithm 2
1. \( S \leftarrow 0 \)
2. for each vertex \( v \in V[G] \) do MAKE-SET\( (v) \)
3. sort the edges \( E \) by non-decreasing weight \( w \)
4. for each edge \((u, v) \in E\), in order of nondecreasing weight do if \( \text{FIND-SET}(u) \neq \text{FIND-SET}(v) \) then \( S \leftarrow S \cup (u, v) \)
5. UNION\( (u, v) \)
6. return \( S \)

Figure 3–4: Kruskal MST algorithm.

Algorithm 3
1. input: \( n \) \( d \)-dimensional patterns, initial number of clusters \( k \), the number of clustering \( N \), the threshold \( \tau \) and the \( \beta \)
2. output: clustered patterns
3. initialize the similarity matrix \( M \) to null \( n \times n \) matrix and the number of iterations \( \text{iter} = 0 \)
4. apply the KMEM algorithm to produce the partition \( C \)
5. update the \( M \); for each input pattern \((i, j) \in C\) set \( a(i, j) = a(i, j) + 1/N \)
6. if \( \text{iter} < N \) go to step 2
7. obtain final clustering by applying the MST and removing inconsistent edges \((a(i, j) < \tau)\)

Figure 3–5: Graph clustering algorithm.

of their weights. At each step, one finds the minimum edge in the graph not already present in the minimum spanning tree, connects two forests together. This process is repeated until all the vertices are included in the graph. The algorithm is given in figure 3–4. In steps 1-3, the subset \( S \) is initialized to null and \( V \) number of forests each with a single vertex are created. Step 4 sorts the edge set \( E \) in a non decreasing order of weight. In steps 5-8, an edge \((u; v)\) is found such that the endpoint \( u \) belongs to one forest and endpoint \( v \) belongs to other forest. This edge is incorporated in the subset \( S \). The algorithm stops when all vertices are included in the tree.

The algorithm given in figure 3–5 is used to generate the final clustered data.

3.6 Results

The KMEM and the graph matching algorithms were tested on some synthetic image and data from the UCI data repository [7]. The data include the Iris data,
Table 3–1: The number of clusters for different values of $\beta$

<table>
<thead>
<tr>
<th>$\beta$</th>
<th>Images</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>test1</td>
</tr>
<tr>
<td>1.0</td>
<td>10</td>
</tr>
<tr>
<td>1.5</td>
<td>6</td>
</tr>
<tr>
<td>3.5</td>
<td>5</td>
</tr>
<tr>
<td>5.5</td>
<td>4</td>
</tr>
</tbody>
</table>

wine data and heart disease data. The results for the synthetic images and iris data are given in 6.1 and 6.2. The KMEM algorithm was run 200 times in order to obtain the similarity matrix and the average number of clusters $k_{\text{ave}}$.

3.6.1 Image Clustering

For the synthetic images, the objective is to reduce the complexity of the grey levels. Our algorithm was implemented with synthetic images for which the ideal clustering is known. Matlab and Paint Shop Pro were used for the image processing in order to obtain an image data matrix. A total of three test images were used with varying numbers of clusters. The first two images, test1 and test2, have four clusters. Three of the clusters had uniformly distributed values with a range of 255, and the other had a constant value. Test1 had clusters of varying size while test2 had equal sized clusters. The third synthetic image, test3, has nine clusters each of the same size and each having values uniformly distributed with a range of 255. We initialized the algorithm with the number of clusters equal to the number of grey levels, and the value of cluster centers equal to the grey values. The initial probabilities ($p_i$) were computed from the image histogram. The algorithm was able to correctly detect the number of clusters. Different clustering results were obtained as the value of the entropy constant was changed, as is shown in Table 3–1. For the image test3, the correct number of clusters was obtained using a $\beta$ of 1.5. For the images test1 and test2, a $\beta$ value of 5.5 yielded the correct number of clusters. In Table 3–1, the optimum number of clusters for each synthetic image are bolded.
3.6.2 Iris Data

Next we tested the algorithm on the different data obtained from the UCI repository and got satisfactory results. The results presented in this section are on the Iris data. The Iris data are well known [12, 27] and serves as a benchmark for supervised learning techniques. It consists of three types of Iris plants: *Iris Versicolor*, *Iris Virginica*, and *Iris Setosa* with 50 instances per class. Each datum is four dimensional and consists of a plants’ morphology namely *sepal width*, *sepal length*, *petal width*, and *petal length*. One class *Iris Setosa* is well separated from the other two. Our algorithm was able to obtain the three-cluster solution when using the entropy constant $\beta$’s of 10.5 and 11.0. Two cluster solutions were also obtained using entropy constants of 14.5, 15.0, 15.5 and 16.0 Table 3–2 shows the results of the clustering.

To evaluate the performance of our algorithm, we determined the percentage of data that were correctly classified for three cluster solution. We compared it to the results of direct K-means. Our algorithm had a 91% correct classification while the direct K-means achieved only 68% percent correct classification, see Table 3–3. Another measure of correct classification is entropy. The entropy of each cluster is calculated as follows

$$H(C_j) = - \sum_{j=1}^{k} \frac{n_j}{n_i} \frac{n_i^j}{n_j} \ln \frac{n_i^j}{n_j}$$

(3–21)

where $n_j$ is the size of cluster $j$ and $n_i^j$ is the number of patterns from cluster $i$ that were assigned to cluster $j$. The overall entropy of the clustering is the sum of the weighted entropy of each cluster and is given by

$$H(C) = \sum_{j=1}^{k} \frac{n_j}{n} H(C_j)$$

(3–22)

where $n$ is the number of input patterns. The entropy is given in table 3–3. The lower the entropy the higher the cluster quality.
Table 3–2: The number of clusters as a function of $\beta$ for the iris data

<table>
<thead>
<tr>
<th>$\beta$</th>
<th>10.5</th>
<th>11.0</th>
<th>14.5</th>
<th>15.0</th>
<th>15.5</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k$</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 3–3: Percentage of correct classification of iris data

<table>
<thead>
<tr>
<th>$k$</th>
<th>3.0</th>
<th>3.0</th>
<th>2.0</th>
<th>2.0</th>
<th>2.0</th>
<th>2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>%</td>
<td>90</td>
<td>91</td>
<td>69</td>
<td>68</td>
<td>68</td>
<td>68</td>
</tr>
<tr>
<td>$Entropy$</td>
<td>0.31</td>
<td>0.27</td>
<td>1.33</td>
<td>1.30</td>
<td>1.28</td>
<td>1.31</td>
</tr>
</tbody>
</table>

We also determined the effect of $\beta$ and the different cluster sizes on the average value of $k$ obtained. The results are given in tables 3–4, 3–5 and 3–6. The tables show that for a given $\beta$ and different $k$ value the average number of clusters converge.

Table 3–4: The average number of clusters for various $k$ using a fixed $\beta = 2.5$ for the iris data

<table>
<thead>
<tr>
<th>$k$</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>30</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_{ave}$</td>
<td>9.7</td>
<td>14.24</td>
<td>18.73</td>
<td>27.14</td>
<td>42.28</td>
</tr>
</tbody>
</table>
Table 3–5: The average number of clusters for various $k$ using a fixed $\beta = 5.0$ for the iris data

<table>
<thead>
<tr>
<th>$k$</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>30</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_{ave}$</td>
<td>7.08</td>
<td>7.10</td>
<td>7.92</td>
<td>9.16</td>
<td>10.81</td>
</tr>
</tbody>
</table>

Table 3–6: The average number of clusters for various $k$ using a fixed $\beta = 10.5$ for the Iris Data

<table>
<thead>
<tr>
<th>$k$</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>30</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_{ave}$</td>
<td>3.25</td>
<td>3.34</td>
<td>3.36</td>
<td>3.34</td>
<td>3.29</td>
</tr>
</tbody>
</table>

3.7 Conclusion

The KMEM provided good estimates for the unknown number of clusters. We should point out that whenever the clusters are well separated, the KMEM algorithm is sufficient. Whenever that was not the case, further processing by the graph clustering produced the required results. Varying the entropy constant $\beta$ allows us to vary the final number of clusters in KMEM. However, we had to empirically obtain values for $\beta$. Further research work was necessary in order to find a way of estimating the value of $\beta$ based on the some properties of the data set. Our approach worked on the data that we tested, producing the required number of clusters. While our results are satisfactory, we observed that our graph clustering approach sometimes matched weakly linked nodes, thus combining clusters. Therefore, further work will be required to reduce this problem. Such a result would be very useful in image processing and other applications.
CHAPTER 4
DIMENSION REDUCTION

4.1 Introduction

Data mining often requires the unsupervised partitioning of the data set into clusters. It also places some special requirements on the clustering algorithms including: data scalability, non-presumable assumptions of any canonical data distribution, and insensitivity to the order of the input record \[4\]. Equally important in data mining is the effective interpretability of the results. Data sets to be clustered usually have several attributes and the domain of each attribute can be very large. Therefore results obtained in these high dimension are very difficult to interpret. High dimensionality poses two challenges for unsupervised learning algorithms. First the presence of irrelevant and noisy features can mislead the clustering algorithm. Second, in high dimensions data may be sparse (the curse of dimensionality), making it difficult for an algorithm to find any structure in the data. To ameliorate these problems, two basic approaches to reducing the dimensionality have been investigated: feature subset selection (Agrawal et al., 1998; Dy and Brodley, 2000) and feature transformations, which project high dimensional data onto "interesting" subspaces (Fukunaga, 1990; Chakrabarti et al., 2002). For example, principle component analysis (PCA), chooses the projection that best preserves the variance of the data. It therefore is important to have clusters represented in lower dimensions in order to allow effective use of visual techniques and better result interpretation. In this chapter, we address dimensionality reduction using entropy minimization.

4.1.1 Entropy Dimension Reduction

In the previous two chapters and in \[41\], we have shown that entropy is a good measure of the quality of clustering. We therefore propose an entropy method to
handle the problem of dimension reduction. As with any clustering algorithm, certain requirements such as sensitivity to outliers, shape of the cluster, efficiency, etc play vital roles in how well the algorithm performs. In the next, we outline the different criteria necessary to handle these problem via entropy.

4.1.2 Entropy Criteria For Dimension Reduction

Given two clustering of different data sets, how do we determine which cluster is better? Since these clusters may be in different dimensions, we need some criteria that will be robust. We propose the following measures: good data span or coverage; a dimensional space that has well defined clusters will tend to have good data span than one that is closed to random, high density; whereas two two distribution can have have the same data span, one may be more dense and therefore qualify as a cluster. Given these criteria, we some metric that is capable of measuring these criteria simultaneously. A reduced dimension with good clustering should score high on this metric at some level of a threshold. This metric is entropy and we outline the approach in the following sections.

4.1.3 Entropy Calculations

Each dimension is divided into intervals of equal length thus partitioning the high dimension to form a grid. The density of each cell can be found by counting the number of points in the cell. If we denote the set of all cells by $\chi$ and $d(x)$ the density of cell $x$, we can define the entropy of the data set as:

$$H(X) = \sum_{x \in \chi} d(x) \ln d(x) \quad (4-1)$$

When the data points are uniformly distributed, we are most uncertain where a particular point would lie. In this case entropy is highest. When the data points are closely packed in a small cluster, we know that a particular point to fall within a small area of the cluster, and so the entropy will be low. The size of the partition when each is divided should be carefully selected. If the interval is too small, there
will be many cells making the average number of points in each so small, similarly if
the interval size is too large, it may be difficult to capture the differences in density
in different regions of the space. Selecting at least 30 points in each is recommended.
We follow the approach outlined by Chen et al. [10].

4.1.4 Entropy and the Clustering Criteria

Entropy is used to relate the different criteria outline for clustering. As the
density of dense units increases, the entropy decreases. Hence entropy can be
used to relate the measurement of density in clustering. Problem of correlated
variable can also be handled by entropy. Entropy can easily detect independence in
variable through the following relationships. \( H(X_1, ..., X_n) = H(X_1) + ... + H(X_n) \) if
\( X_1, ..., X_n \) are independent. This property will be necessary in our algorithm

4.1.5 Algorithm

The algorithm for dimension reduction consist of two main steps:
1. Find out reduced dimension with good clustering by entropy method.
2. Identify clusters in the dimensions found
To identify good cluster, we set a threshold \( \tau \). A reduced dimension has good
clustering if its entropy is below the threshold. This proposed approach uses a
bottom-up approach. It starts by finding a large one-dimensional space with good
clustering, this is the used to generate candidate 2-dimensional spaces which are
checked against the data set to determine if they have good clustering. The process
is repeated with increasing dimensionality until no more spaces with good clustering
are found. The algorithm is given in 4–1.

4.2 Results

We evaluated the algorithm using both synthetic and real data. For synthetic
data, we generated data of fixed dimensions and also the dimensions that contained
clusters. The algorithm was able to identify the lower dimensional spaces that had
cluster. We next used the algorithm on the breast cancer data which can found in
Algorithm 1
1. \( k = 1 \)
2. Let \( C_k \) be one dimensional space
3. For each space \( c \in C_K \) do
4. \( f_c(.) = \text{density}(c) \)
5. \( H(c) = \text{entropy}(f_c(.)) \)
6. if \( H(c) < \tau \) then
7. \( S_k = S_k \cup c \)
8. else
9. \( NS_k = NS_k \cup c \)
10. End For
11. \( C_{k+1} = \text{cand}(NS_k) \)
12. If \( C_{k+1} = 0 \), goto 15
13. \( k = k + 1 \)
14. goto step 3
15. Result = \( \cup_{k} S_k \)

Figure 4–1: Algorithm for dimension reduction.

[7] in order to reduce the 38 feature space. The algorithm performed well when using only the numerical features.

4.3 Conclusion

In this chapter, we provided an entropy method that can be used for dimension reduction of high dimensional data. This method uses data coverage, density and correlation to determine the reduced dimension that have good clustering. While this method does not cluster the data, it provided a subspace the has good clustering and whose results are easy to interpret.
CHAPTER 5
PATH PLANNING PROBLEM FOR MOVING TARGET

5.1 Introduction

Path planning is concerned with creating an optimal path from point A to point B while satisfying constraints imposed by the path like obstacles, cost, etc. In our path planning problem, we are concerned with planning a path for an agent such that the likelihood of the agent being co-located with the target at some time in its trajectory is maximized. An assumption here is that the agent operates in receding-horizon optimization framework where the optimization considers the likely position of target up to sometime time in the future and is repeated at every time step. The information about the target is contained in a stochastic process, and we assume that the distribution is known at every time in the future.

We consider the problem for path planning of single agent. The basic formulation of the problem is to have the agent move from an initial dynamic state to moving target. In the case of stationary target, several methods have been proposed by other researchers [46, 52]. Here, we assume that the velocity of the vehicle is fixed and it is higher than the maximum velocity of the target. In each fixed length of time interval (even it’s not constant, proposed methods can work), the agents can have the information about the positions of the targets at that time. There will be several modes depending on the prediction of the targets’ move.

1. the prediction of the targets’ move is unknown.
2. the prediction of the targets’ move is given by a probability distribution.

We also consider the case of planning a path for an agent such that its likelihood of being co-located with a target at some time in its trajectory is maximized. We assume
that the agent operates in a receding-horizon optimization framework, with some fixed planning horizon and a reasonable of re-planning. When the future location of the target is expressed stochastically, we derive condition under which the planning horizon of the agent can be bounded from above, without sacrificing performance.

We assume that the target employs a receding horizon approach, where the optimization considers the likely position of the target up to some fixed time in the future and where the optimization is repeated at every time step.

5.1.1 Problem Parameters

We define a discrete two-dimensional state-space $X \subset \mathbb{Z} \times \mathbb{Z}$, where $\mathbb{Z}$ denotes the set of integers. We also denote by the discrete index $t \in \mathbb{Z}$. An agent’s position at time $t = t_0$ is denoted by $x(t_0) \in X$, and a trajectory of an agent can be defined as follows:

**Definition 5.1.** A state $x_i \in X$ is said to adjacent to the state $x_j \in X$ if

$$||x_i - x_j|| \leq 1$$

**Definition 5.2.** The path $p$ for an agent is a sequence of states $x(t) \in X$:

$$p = \{x(t_0), x(t_0 + 1), ..., x(t_0 + T)\},$$

such that $x(t)$ is adjacent to $x(t + 1)$ for all $t \in [t_0 + 1, t_0 + T]$. We say that such a path has length $T$.

The agent is assumed to have information regarding the future position of the target, and this information is contained in a stochastic process, $M(t) \in \mathbb{R}^{N \times N}$. Thus $M(.)$ is a sequence of $N \times N$ matrices whose element at a particular time constitute a probability mass function

$$\sum_i \sum_j M_t(i, j) = 1,$$

$$M_t(i, j) \leq 1,$$
where \( M_t(i,j) \) denotes the \((i,j)^{th}\) element of the matrix \( M(t) \). We assume that there exists a stationary mapping of the elements of \( M(t) \) to the state space \( X \) for all \( t \), and we will use the notation that the probability that the target is at state \( y \in X \) at time \( t = t_0 \) is \( M_{t_0}(y) \).

The receding horizon problem optimization problem is to find a path \( p \) of fixed length \( T \), such that the likelihood of being co-located with the target in at least one state on the path is maximized. One way of estimating this is the following cost function:

\[
J(p) = \sum_{t=t_0}^{t_0+T} M_t(x(t))
\]

In the following sections, we examined the different solution methods we proposed to solve this problem.

### 5.1.2 Entropy Solution

In constructing the optimal path, we will make use of the concept of information theory, specifically information gain. Entropy in information theory measures how predictable a distribution is. Specifically, information gain is the change in entropy of a system when new information relating to the system is gained. Suppose the \( \{X_n, Y_n\} \) is a random process taken on values in a discrete set. Shannon introduced the notion mutual information between the two process:

\[
I(X,Y) = H(X) + H(Y) - H(X,Y),
\]

the sum of two entropies minus the entropy of their pair. Average mutual information can also be defined in terms of conditional entropy

\[
H(X|Y) = H(X,Y) - H(Y)
\]

and hence
\[ I(X, Y) = H(X) - H(X|Y) = H(Y) - H(Y|X). \]  \hspace{1cm} (5–2)

In this form the mutual information can be interpreted as the information contained in one process minus the information contained in the process when the other process is known [20]. Recall that the entropy of a discrete distribution over some set \( X \) is defined as

\[ H(X) = -\sum_{x \in X} p(x) \ln p(x) \]  \hspace{1cm} (5–3)

Given the value \( v \) of a certain variable \( V \), the entropy of a system \( S \) defined on \( X \) is given by

\[ H(S|V = v) = -\sum_{x \in X} p(x|v) \ln p(x|v) \]  \hspace{1cm} (5–4)

Now suppose that we gain new information \( V \) about the system in the form a distribution over all possible values of \( V \), we can define the conditional entropy of the system \( S \) as

\[ H(S|V) = -\sum_{v \in V} p(v) \sum_{x \in X} p(x|v) \ln p(x|v) \]  \hspace{1cm} (5–5)

Thus for some system and some new knowledge \( V \), the information gain is according to equation \((5–2)\) is

\[ I(S, V) = H(S) - H(S|V) \]  \hspace{1cm} (5–6)

Incorporating information gain in our path planning, a strategy that leads to maximum decrease in conditional entropy or maximum information gain is desirable. This strategy will move us to a state of low entropy as quickly as possible, given our current knowledge and representation of the system. We can therefore use entropy measure in which a state with low entropy measure correspond to the solution of the
path. Our proposed entropy method is used to build an initial path. Local search method is then used to improve the path. The results generated by our simulator is then used on the different modes outlined below and on the cost function given in 5–14. Several other methods and cost functions are also discussed.

5.2 Mode 1

We propose the following method. This method can be applied for the both modes. We describe the method for single vehicle and single target. The vehicle move along the shortest path between the vehicle’s current position and the target’s current position until getting the updated information. We can give the following Proposition for this mode.

**Proposition 5.1.** Let $S_t$ be length of the shortest path between the vehicle and the target at time $t$. Then there exists an positive integer number $n_0$ such that

$$ S_t \leq v_m t_0 \text{ for all } t \geq n_0 t_0, $$

where $t_0$ is the length of the time interval the vehicle can get the information of the target’s position and $v_m$ is the maximum velocity of the target.

**Proof.** Consider $S_{kt_0}$, the length of the shortest path between the target and the vehicle at time $kt_0$. We can write down the following inequality for $S_{kt_0}$ and $S_{(k+1)t_0}$.

$$ S_{kt_0} - S_{(k+1)t_0} \geq (u - v_m) t_0 $$

(5–7)

(explanation and a figure)

Indeed, the length of the shortest path between the current position of the vehicle and the next position (in time $t_0$) of the target is less than the sum of the $S_{kt_0}$ and $L_k$, which is the length of the trajectory the target move in time interval $[kt_0, (k + 1)t_0]$. Moreover, the following inequality holds:

$$ L_k \leq v_m t_0. $$
Along the shortest path, the vehicle moves for distance of $ut_0$ in time interval $[kt_0, (k+1)t_0]$.

Since in each time interval, the length of shortest path between the vehicle and the target is reduced by at least fixed amount of $(u - v_m)t_0$, after time $n_0t_0$, where

$$n_0 \leq \left\lceil \frac{S_0}{(u - v_m)t_0} \right\rceil$$  \hspace{1cm} (5-8)

Which completes the proof.

**Definition 5.3.** $D_T = D_T(t_1, x^0) \subseteq \mathbb{R}^2$, the set the target could reach in time $t_1$ from the current point $x^0$, is called the reachable set of the target at time $t_1$.

**Definition 5.4.** $D_A = D_A(t_1, x^0) \subseteq \mathbb{R}^2$, the set the agent could reach in time $t_1$ from the current point $x^0$, is called the reachable set of the agent at time $t_1$.

Since the agent can not predict the move of the target, we will assume that the direction of the target move is the uniform distribution over $[-\pi, \pi]$ at every time $t \in [0, t_1]$. Since for each move of the target, there exist a corresponding move which is exactly vice direction of the move. According to this assumption, the reachable set is a circle with its inside. Moreover, the density function values at points on the circle which has the same center as the reachable set. At the next time, the distance between the target and the agent is a random variable. Our goal is to find a target which minimizes the expectation of the distance between the agent and the target.

The following proposition gives us the answer for single target and single agent.

**Proposition 5.2.** The optimal trajectory is the line segment which connects the current position of the agent and the current position of the target.

**Proof.** Let $A_k$ be the current position of the agent and $B_k$ be the current position of the target at time $kt_0$. Consider the circle with center of $B_k$ and a radius of $v_mt_0$; and the circle $w_2$ with center of $A_k$ and a radius of $ut_0$. Then those circles represent the boundaries of the reachable sets for targets and the agents. We prove
that the optimal trajectory for the agent is $A_k A_{k+1}$ (optimal position for the agent at time $(k + 1)t_0$ is $A_{k+1}$). Without loss of generality, the optimal position for the agent is $A'_{k+1}$ which is a point inside of $w_2$ at time $(k + 1)t_0$. Consider a polar coordinate system $(\rho, \alpha)$. Let $h(\rho, \alpha)$ be the distance between point $(\rho, \alpha)$ and $A_{k+1}$. Then the expectation of the distance between $A_{k+1}$ and the next position of the target at time $(k + 1)t_0$ is

$$
\int_0^{v_m t_0} \int_0^{2\pi} h(\rho, \alpha) f(\rho, \alpha) d\alpha d\rho,
$$

(5–9)

where $f(\rho, \alpha)$ is the density function of the next position of the target at time $(k+1)t_0$.

Consider a new polar coordinate system $(\rho', \alpha')$ for the angle $\theta$,

$$
\alpha' = \alpha - \theta
$$

$$
\rho' = \rho.
$$

Consider the expectation of the distance between $A'_{k+1}$ and the next position of the target at time $(k + 1)t_0$. If we denote $h'(\rho', \alpha')$ by the distance between point $(\rho', \alpha')$
and $A'_{k+1}$, the expectation is
\[ \int_0^{v_m t_0} \int_0^{2\pi} h'(\rho', \alpha') f(\rho', \alpha') d\alpha' d\rho', \]
(5–10)
since the function $f(\rho, \alpha)$ has rotation symmetry. If we count the fact that
\[ h'(\rho', \alpha') \geq h(\rho, \alpha) \]
when $\rho' = \rho$ and $\alpha' = \alpha$,
then
\[ \int_0^{v_m t_0} \int_0^{2\pi} h_2(\rho', \alpha') f(\rho', \alpha') d\alpha' d\rho' \geq \int_0^{v_m t_0} \int_0^{2\pi} h(\rho, \alpha) f(\rho, \alpha) d\alpha d\rho. \]
Since the rotation keeps distances, our proposition is proved.

5.3 Mode 2

Suppose that the agent is given the target move as a probability distribution. Moreover, we assume that we know the density function, say $p(x, y)$, of the next position of target in the reachable set. Our goal is to find the trajectory, which minimizes the expected distance between the agent and the target at time $t_1$. Let $D_A$ be the reachable set, a set of finite points, of the agent at time $t_1$.
Let $x^0 = (x^0, y^0)$ be the current position of the agent and $D_T = \{x^i = (x^i, y^i), i = 1, 2, ..., l\}$ be the reachable set for the target at $t$.

\[ \min \sum_{i=1}^{l} p_i \|x - x^i\| \]
(5–11)
\[ s.t. \quad x \in D_A. \]

Without obstacles the reachable set $D_A$ for the agent is a circle with its inside.

5.4 Mode 3

Another interesting question is how to find the optimal path if we do not know the exact time when the next information will be received by the agent. In other words, suppose that the time when the agent receive the next information about the position of the target at is given according to some probability distribution $g(t)$,
Given $t \in \{0 = t_0 < t_1 < t_2 < ... < t_T = t_1\}$. Without loss of generality, $t = 0, 1, ..., T$ and $g(t) = g_t$, the probability which the next information is received by the agent at time $t$. Let $D_T(t) = \{x^i_t = (x^i_t, y^i_t), i = 1, 2, ..., l\}$, $t = 0, 1, 2, ..., T$ be the reachable sets of the target. One we can propose the following model.

$$\min \sum_{t=0}^{T} \left( \sum_{i=1}^{l_t} \|x_t - x^i_t\| p_t^i - h^*(t) \right) g_t$$

(5–12)

s.t. $\|x_{t+1} - x_t\| \leq u$, $t = 0, 1, 2, ..., T - 1$

where $x_t = (x_t, y_t)$, $t = 0, 1, 2, ..., T$, constitute the agent’s path, and $u$ is the velocity of the agent and $h^*(t)$ is a solution to Problem (5–11). Consider the following expression.

$$\sum_{i=1}^{l_t} \|x_t - x^i_t\| p_t^i - h^*(t)$$

This give us the error amount when the next information is received at $t$. Then objective function of Problem (5–12) is the expected error of the agent regarding the time when information is received.

**Proposition 5.3.** The constraint set of Problem (5–12) is convex.

*Proof.* Consider the function $f(x^1, x^2) = \|x - y\|$. This is a convex function. Thus

$$f(\lambda x^1 + (1 - \lambda)y^1, \lambda x^2 + (1 - \lambda)y^2) \leq \lambda f(x^1, x^2) + (1 - \lambda)f(y^1, y^2)$$

$$\lambda u \triangle + (1 - \lambda)u \triangle = uu \triangle$$

Since the problem is convex, it can be solved using one of the existing gradient methods.

### 5.5 Maximizing the Probability of Detecting a Target

The modes we discussed in the previous sections can work for the case which the distance between the agent and the target is far enough and the time interval between
consecutive two information is short. For long time and short distance, those modes are not very efficient. From now, we discuss the problem to find an optimal path \( p \), of fixed length \( T \), for the agent such that likelihood of being co-located with the target in at least one point on the path is maximized. For continuous time the problem is very expensive to solve. Thus, we will have the following assumption. The vehicle and the target move among a finite set of cells in discrete time. At the beginning of each time period the agent and the target can move only to the adjacent cells or can stay the same cells as they were staying in the previous time period. Moreover, when the agent and the target are in the same cell, then the agent can detect the target with probability 1. We are looking for a path such that the probability of detecting the target in a fixed number of time periods, say \( T \), is maximized.

\[
\text{Figure 5–2: Region}
\]

### 5.5.1 Cost Function. Alternative 1

We find the probability of co-locating at least one point in a fixed number of time for a path \( x \). Let us define the indicator random variables \( I^i_x, i = 0, 1, \ldots, T \) for path \( x \) by

\[
I^i_x = \begin{cases} 
1 & \text{if } x(t_0 + i) \text{ is a co-located point} \\
0 & \text{otherwise.}
\end{cases}
\]

The probability that the agent and the target are co-located at least one point on the agent’s \( T \)-length path is

\[
J(x) = 1 - P \left( \bigcap_{t=0}^{T} I^t_x = 0 \right).
\]
where \( P \left( \bigcap_{t=0}^{T} I_x^t = 0 \right) \) can be written as follows

\[
P \left( \bigcap_{t=0}^{T} I_x^t = 0 \right) = \prod_{t=0}^{T} P \left( I_x^t = 0 \big| \bigcap_{j=0}^{t-1} I_x^j = 0 \right).
\]

Our optimization problem becomes

\[
\max_x J(x) \quad (5\text{–}15)
\]
or

\[
\min_x \prod_{t=0}^{T} P \left( I_x^t = 0 \big| \bigcap_{j=0}^{t-1} I_x^j = 0 \right).
\]

Taking natural logarithm from this, the objective function takes the form

\[
\min_x \sum_{t=0}^{T} \ln \left( P \left( I_x^t = 0 \big| \bigcap_{j=0}^{t-1} I_x^j = 0 \right) \right).
\]

Since \( I_x^t, t = 0, \ldots, T \) random variables are dependent on each other, we need a huge and complete information regarding the target’s motion which contains ..... (I mean it’s much more difficult because we are not talking about \( I_x^t = 1 \)'s). One way to handle this problem is an approximation method. If we assume that the dependence of \( I_x^k = 0 \) and \( I_x^j = 0 \)'s, \( j = 0, 1, 2, \ldots, k - 1 \) and \( k = 1, 2, \ldots, T \), is weak, then one can take

\[
\min_x \sum_{t=0}^{T} \ln (P(I_x^t = 0)),
\]
or

\[
\min_x \sum_{t=0}^{T} \ln (1 - P(I_x^t = 1)). \quad (5\text{–}16)
\]

However, the consequent error of using the assumption depends on a model of target’s motion. For some case, it can give us optimal path. The last optimization problem is much more easier than (5–15). We will discuss this problem in a later section.

5.5.2 Generalization

For more reality, we should accept that an agent’s ability to detect a target in the same cell could be not perfect. If the agent and the target are in cell \( j, j=1,\ldots,N, \) at
the beginning of a time period, then the agent can detect the target with probability $q_j$. If they are in different cells, the agent cannot detect the target during the current time period.

Let us introduce the following indicator random variables $D^i_x$, $i = 0, 1, \ldots, T$ for path $x$ by

$$D^i_x = \begin{cases} 
1 & \text{if the agent detects the target at } x(t_0 + i) \\
0 & \text{otherwise.}
\end{cases}$$  \hspace{1cm} (5–17)

The probability of detecting the target in a fixed time $T$ for given agent’s $T$ length path path $x$ is

$$J(x) = 1 - P\left(\bigcap_{t=0}^{T} D^t_x = 0\right),$$

where $P\left(\bigcap_{t=0}^{T} D^t_x = 0\right)$ can be written as follows

$$P\left(\bigcap_{t=0}^{T} D^t_x = 0\right) = \prod_{t=0}^{T} P\left(D^t_x = 0 \bigg| \bigcap_{j=0}^{t-1} D^j_x = 0\right).$$

After taking natural logarithm as we did before, the problem becomes

$$\min_x \sum_{t=0}^{T} \ln\left(P\left(D^t_x = 0 \bigg| \bigcap_{j=0}^{t-1} D^j_x = 0\right)\right).$$

We propose an approximate method for the problem changing the cost function as follows:

$$\min_x \sum_{t=0}^{T} \ln(P(D^t_x = 0)),\hspace{1cm} \text{or}$$

$$\min_x \sum_{t=0}^{T} \ln(1 - P(D^t_x = 1)), \hspace{1cm} (5–18)$$

where $P(D^t_x = 1)$ can be expressed as follows:

$$P(D^t_x = 1) = P(D^t_x = 1 | I^t_x = 1)P(I^t_x = 1) + P(D^t_x = 1 | I^t_x = 0)P(I^t_x = 0) = P(D^t_x = 1 | I^t_x = 1)P(I^t_x = 1) = q_j P(I^t_x = 1).$$
Here $j = x(t)$.

5.5.3 Cost function. Alternative 2 and Markov Chain Model

In this subsection we discuss a problem which is a case of the problem, so called the path constrained search problem, James N.Eagle introduced in [13, 14]. They assumed that the target moves according to a Markov chain model.

**Definition 5.5.** *Short definition of Markov chain.*

In more precisely, we assume that the target in cell $j$ moves to cell $k$ with probability $p_{jk}$ in one time period. The transition matrix, $P = (p_{jk})$ is known to the agent. Under this assumption, here we find the exact probability of detecting the target for a given agent’s $T$ length path $x$.

The cost function (5–14) can be written in another form as follows:

$$ J(x) = P \left( \bigcup_{i=0}^{T} I_x^i = 1 \right) $$

(5–19)

The right hand side of the last equation is extracted using the ....’s identity.

$$ P \left( \bigcup_{i=0}^{T} I_x^i = 1 \right) = \sum_{i=0}^{T} P(I_x^i = 1) - \sum_{i<j} P(I_x^i = 1, I_x^j = 1) $$

$$ + \sum_{i<j<k} P(I_x^i = 1, I_x^j = 1, I_x^k = 1) $$

$$ - \cdots + (-1)^{T+1} P(I_x^0 = 1, I_x^1 = 1, ..., I_x^T = 1) $$

We show that we are able to simplify each component of the summation.

**Definition 5.6.** $P^{(n)} = P \cdot P \cdots P = P^n$ is called $n$-step transition probabilities matrix.

Note that

$$ P(I_x^{i_1} = 1, I_x^{i_2} = 1, ..., I_x^{i_k} = 1) = \prod_{j=1}^{k} P \left( I_x^{i_j} = 1 \right| \bigcap_{l=1}^{j-1} I_x^{i_l} = 1 \right). $$
Using Markovian property the above can be simplified as follows:

\[ P(I_x^i = 1, I_x^{i_2} = 1, ..., I_x^{i_k} = 1) = \prod_{j=1}^{k} P(I_x^{i_j} = 1 | I_x^{i_j-1} = 1) \]

where \( P(I_x^{i_j} = 1 | I_x^{i_j-1} = 1) = P_{s_{r^{i_j}}, s^{i_j}} \), \( s = x(i_{j-1}) \), \( r = x(i_j) \) and \( P(I_x^{i_1} = 1) = P_{s_0 p} \), \( p = x(i_1) \) and \( s_0 \) is the initial cell the target was at the beginning of time period 0.

### 5.5.4 The Second Order Estimated Cost Function with Markov Chain

Using the cost function (5–16) could give undesirable big error. For this we can propose another cost function.

\[ J(x) = 1 - \prod_{i=1}^{T} P(I_x^i = 0 | I_x^{i-1} = 0) \]

(5–20)

Of course, we should assume that \( P(I_x^0 = 0) = 1 \) or initial states of the agent and the target are not the same otherwise there is nothing to solve. Using the fact that

\[
\begin{align*}
P(\bar{A} | \bar{B}) &= 1 - P(A | \bar{B}) \\
&= 1 - \frac{P(A, \bar{B})}{P(\bar{B})} \\
&= 1 - \frac{P(\bar{B} | A) P(A)}{1 - P(B)} \\
&= 1 - \frac{(1 - P(B | A)) P(A)}{1 - P(B)} \\
&= 1 - \frac{(1 - \frac{P(AB)}{P(A)}) P(A)}{1 - P(B)} \\
&= 1 - \frac{P(A) - P(AB)}{1 - P(B)} \\
&= 1 - \frac{P(A) - P(A | B) P(B)}{1 - P(B)} \quad (5–21)
\end{align*}
\]

(assuming \( P(B) \neq 1 \))

the problem becomes

\[
\min_x J(x) = \prod_{i=1}^{T} \left( 1 - \frac{P(I_x^i = 1) - P(I_x^i = 1 | I_x^{i-1} = 1) P(I_x^{i-1} = 1)}{1 - P(I_x^{i-1} = 1)} \right)
\]
or
\[
\min_x J(x) = \sum_{i=1}^{T} \ln \left( 1 - \frac{P(I_x^i = 1) - P(I_x^i = 1 | I_x^{i-1} = 1) P(I_x^{i-1} = 1)}{1 - P(I_x^{i-1} = 1)} \right). \tag{5–22}
\]

5.5.5 Connection of Multistage Graphs and the Problem

Definition 5.7. A multistage graph \( G = (V, E) \) is a directed graph in which the vertices are partitioned into \( k \geq 1 \) disjoint sets \( V_i, 0 \leq i < k \). In addition, if \((u, v)\) is an edge in \( E \), then \( u \in V_i \) and \( v \in V_{i+1} \) for some \( i, 0 \leq i < k \). The sets \( V_0 \) and \( V_k \) are such that \( |V_0| = |V_k| = 1 \). The vertices in \( V_0 \) and \( V_k \) are called the source and the sink nodes.

Let \( c(i, j) \) be the cost of edge \((i, j)\) \( \in E \). The cost of a path from \( s \) to \( t \) is the sum of the costs of the edges on the path. The multistage graph problem is to find a minimum-cost path from the source node to the sink node.

A dynamic programming formulation for a \( k \)-stage graph problem is obtained by first noticing that every path from the source node to the sink node is the result of a sequence of \( k - 1 \) decisions. The \( i \)th decision involves determining which vertex in \( V_{i+1}, 0 \leq i \leq k - 2 \), is to be on the path. Let \( p(i, j) \) be a minimum-cost path from the source node to a vertex \( j \) in \( V_i \). Let \( \text{cost}(i, j) \) be the cost of \( p(i, j) \). Then the following is true.

\[
\text{cost}(i, j) = \min_{l \in V_{i-1}} \{ \text{cost}(i - 1, l) + c(l, j) \}.
\]

Next, we explain how our problem with cost functions (5–14), (5–22) and (??) can be expressed as multistage graph problems.

Let \( \{1, 2, \ldots, N\} \) be the cells that represent the map of the region. \( V_i, i = 1, 2, \ldots, T \) consists of \( N \) nodes that represents \( N \) cells of the agent (or target) at time \( i \). \( V_0 \) or source node represents the agent’s initial position which is a cell. We have additional one node, the sink node in \( V_{T+1} \). We connects nodes in \( V_i \) to nodes \( V_{i+1} \).
which the agent (or the target) can move in one time step (accessible or adjacent cells). This cost function works for not only Markov chain but also general case. In other words, each edge which connects to a node in \( V_i, i = 1, 2, \ldots, T \) has the same length. For instance, let \( j \in V_i \). Then all of edges from \( V_{i-1} \) to \( j \in V_i \) has length \( \ln(1 - P^i_j) \). Here \( P^i_j \) is the probability that the target would be in cell \( j \) at time \( i \). The sink node must be connected to the all nodes in \( V_T \). We assign cost of 0 to those edges. For instance, let us consider a \( 3 \times 3 \) map. Let us assume that the agent can move from one node to only its adjacent nodes (diagonally adjacent nodes cannot be included) in one time period. Then the multistage graph representation can be shown as in Figure 2.

![Figure 5-3: Multistage graph representation 1](image-url)
Here the construction of multistage graph is the same as (5–16). The only difference is the length of the edges. We should assign cost of

$$\ln \left( 1 - \frac{P^i_{soq} - P^{i-1}_{pq}P^i_{soq}}{1 - P^{i-1}_{pq}P^i_{soq}} \right),$$

where $s_0$ is the initial position or cell of the target, to edge $(p, q)$, where $p \in V_{i-1}$ and $q \in V_i$, $i=1,2,...,T$.

We give an explanation how to use the multistage graph problem to this problem. $V_0$ consists of one node which represents the initial state of the target. $V_1$ consists of all possible cells of the map but we add only admissible edges and assign costs same as we did in the second order estimated cost function. $V_k$, $k = 2,3,...,T$, consists of a number of nodes. Each of them represents a couple of states, say $(p, q)$. Moreover edge $(p, q)$ must be an admissible edge in sense of real path. We connect node $(p, q)$ in $V_2$ to node $p$ in $V_1$ by an edge and assign to the edge cost of

$$1 - \frac{P^2_{soq} - P_{pq}P_{soq}}{1 - P_{pq}P_{soq}}.$$

We connect node $(p, q)$ in $V_k$ to any nodes in $V_{k+1}$ which the first of the two states of the node is $q$. For example node $(p, q)$ in $V_k$ and node $(q, r)$ in $V_{k+1}$ must be connected by an edge. We give cost of

$$1 - \frac{P^k_{soq} - P_{qr}P^k_{soq} - P^k_{pr}P^{k-1}_{soq} + P_{qr}P_{pq}P^{k-1}_{soq}}{1 - P^k_{soq} - P^k_{pr}P^{k-1}_{soq} + P_{pq}P^{k-1}_{soq}}$$

to the edge. Consider node $(4,5)$ in $3 \times 3$ region. The agent can move from one node to only its adjacent nodes (diagonally adjacent cannot be included) in one time period. Then $(4,5)$ in $V_k$ can be connected to only nodes $(5,2)$, $(5,4)$, $(5,5)$, $(5,6)$ and $(5,8)$ (see Figure 3.).
5.6 More General Model

5.6.1 The Agent is Faster Than Target

It can be arranged changing the transition probabilities matrix, i.e., the probability of staying the same state is increased.

5.6.2 Obstacles in Path

Obstacles can be arranged assigning cost of $-M$ (or $M$ depending on the problem), where $M$ is a big positive number.
5.6.3 Target Direction

We can consider the current direction of the target. It can be arranged extending the Markov chain or the transition matrix. If we use the 8 azimuths, we have an $8N \times 8N$ transition probabilities matrix instead of an $N \times N$ matrix. For each cell, there are 8 different states which is shown in Figure 4.

![Figure 5-6: Using the 8 azimuths](image)

If the target is in cell $i$ and heads to North East then the probability that it will be in cell $j$ and will head to South East is $P_{(i,NE),(j,SE)}$.

5.7 Conclusion and Future Direction

Given a path we have developed and coded an exact method of calculating the probability of intercept. We have also built a simulator to test our path planning algorithm. The simulator builds agent and target paths and determines the number of times the target and agent paths intersect. The different models proposed in this chapter were tested against our coded simulation. The future direction in this research is to compare our model against competing path planning algorithms and also consider impacts of path planning horizon.
CHAPTER 6
BEST TARGET SELECTION

6.1 Introduction

Weaponized unmanned systems can be expected to search for and attack targets in a battlespace where there is a variety of targets with different values. An obvious goal of war fighters is to attack the most valued targets. This goal can be challenging in an uncertain battlespace where the number of detectable targets and their values are most likely not known with certainty at the time of launch. However, a best target strategy is suggested to maximize the probability of a weaponized unmanned system attacking the most valued target within the battlespace. The strategy fits well with autonomous systems making decisions of action. The results of calculations and simulations show the strategy is robust to variability’s and uncertainties. The strategy is extended to multiple unmanned systems searching for targets in the same battle space. Information sharing between unmanned systems is considered as way of improving results. This can be achieved through mutual information or cross entropy.

The future battlespace will consist of automatous agents either working alone or cooperatively in an effort to locate and attack enemy targets. An obvious goal is that agents should attack the most valuable targets. Although value is subjective and can change as the battlespace changes, one may assume the value of targets can be assessed given a time and situation. Some basic strategies are presented that maximize the probability of attacking the most valuable targets. Performance measures of these strategies are:

1. Probability of attacking the most valuable target.
2. Probability of attacking the $j^{th}$ most valuable target. In particular, probability of attacking the worst target.
3. Mean rank of the attached target. Rank of 1 is best of all \( n \) targets.

4. Average number of targets examined before a decision to attack is made. A smaller number means the agent is exposed to a hostile environment for less time.

This chapter is organized into eight sections. Section 6.2 discusses a strategy to maximize the probability of attacking the most valuable target. Section 6.3 discusses a strategy that increases the mean value of attacked targets after many missions. The impact of a variable number of targets is discussed in Section 6.4. A novel approach based on learning agents is presented in Section 6.5. This learning could be This strategy provides some exciting results that encourage further research where entropy or mutual information could be applied. Multiple agents is discussed in Section 6.6. Multiple agents can be used as a pack or separately in against a battlespace. Lastly, a strategy based on a dynamic threshold is presented in Section 6.7.

6.2 Maximize Probability of Attacking the Most Valuable Target

Consider a situation as represented in Figure 6–1 where an agent is to find and attack 1 of \( n \) distinct targets that are uniformly distributed across a battlespace. The agent carries a single weapon. In random order the agent detects and classifies each target one at a time and the agent must decide to attack or move on. If the agent moves on, then it cannot return to attack (assume the target conceals itself or moves upon detection). In this situation assume the locations and values of the targets within the battlespace are unknown ahead of time. As mentioned above, a goal is to attack the most valuable target. If the agent makes a decision to attack a particular target soon into the mission, then it may not see a more valuable target later. If the agent attacks late in the mission, the agent may have passed over more valuable targets.
6.2.1 Best Target Strategy

To maximize the probability of attacking the most valuable target a decision strategy to consider is that described by [58] where the agent will examine the first $k$, $1 \leq k < n$, targets then attack the first target after $k$ that is more valuable than any of the first $k$. The probability of attacking the best target is shown as

$$P_k(\text{best}) = \frac{k}{n} \sum_{i=k}^{n-1} \frac{1}{i}.$$

$k$ is selected such that the agent maximizes the probability of attacking the best target. The value of $k$ that maximizes this probability is $k = \frac{n}{e}$ rounded to nearest integer. When $n$ is large this probability is $\frac{1}{e} \approx 0.368$. Many may find this a surprisingly high probability of attacking the most valued target among a large number of targets given that there is only one most valued target and that value and the value of all other targets are unknown beforehand.

As an example of this strategy, consider 5 distinct targets with distinct values. Allow $k = 2$. Assume the targets would be presented in the following order of their ranked value (1 being best): 4, 3, 1, 5, and 2. The agent passes first two targets and
records the values (best of first \( k \) is 3). After the first two targets pass, the agent is ready to attack the first target with value better than 3 which is 1 in this case. In the situation of \( n = 5 \) and \( k = 2 \), the probability of attacking the most valuable target using this strategy is 0.433.

If the most valuable target is among the first \( k \) targets, then the strategy would result in attacking the \( n^{th} \) target, regardless of value. Even if the most valuable target is not among the first \( k \) targets, the strategy may result in attacking other than most valuable target. For example, if the third most valuable target is the most valuable among the first \( k \) targets and, if after \( k \), the second most valuable target is encountered before the most valuable target, then the second most valuable target will be attacked. Therefore, this brings to mind the questions in the introduction.

6.2.1.1 Probability of Attacking \( j^{th} \) Most Valuable Target

To develop the relation for the probability of attacking the \( j^{th} \) most valuable target we begin by examining the probability of attacking the second most valuable target, then third and finally the worst. This development will lead to a generalized closed form expression for \( P_k(j^{th} \text{best}) \).

By using conditional probability where \( X \) is the position of the second most valuable target and ensuring \( 0 < k \leq n - 2 \) an expression is developed as follows

\[
P_k(\text{2nd best}) = \sum_{i=1}^{n} P_k(\text{2nd best}|X = i)P(X = i),
\]

\[
= \sum_{i=1}^{n} P_k(\text{2nd best}|X = i) \frac{1}{n}.
\]

To complete (6–1), expressions are needed for \( P_k(\text{2nd best}|X = i) \) which follow
\[ P_k(2\text{nd best}|X = i) = \begin{cases} P_k(2\text{nd best}) = 0 & \text{if } 1 \leq i \leq k, \\ \frac{k}{i} \cdot \frac{n-i}{n-1} & \text{if } k < i < n, \\ \frac{k}{n-1} & \text{if } i = n. \end{cases} \] (6–2a)

If the second best target is among the first \( k \), then there is no chance of attacking it; therefore, (6–2a) holds. Equation (6–2b) holds because it is the probability of the best of the first \( i-1 \) is among the first \( k \) and the overall best target is after \( i \). Finally, (6–2c) follows because it is the probability of the overall best is encountered among the first \( k \).

Note that if \( k = n - 1 \) the only way the second best target is attacked is if the overall best target is among the first \( k \) and the second best target is located at the \( n \)th position. The probability of this occurring is

\[ P_k(2\text{nd best}) = \frac{1}{n} \cdot \frac{k}{n-1} \text{ if } k = n - 1. \] (6–3)

Inserting (6–2) back into (6–1) and incorporating (6–3) results in

\[ P_k(2\text{nd best}) = \begin{cases} \frac{1}{n} \cdot \frac{k}{n-1} + \frac{1}{n} \sum_{i=k+1}^{n-1} \frac{k}{i} \cdot \frac{n-i}{n-1} & \text{if } 1 \leq k < n-2, \\ \frac{1}{n} \cdot \frac{k}{n-1} & \text{if } k = n-1. \end{cases} \] (6–4)

Similarly, if we ensure \( 0 < k \leq n - 3 \) an expression for the probability of attacking the third best target can be developed and is

\[ P_k(3\text{nd best}) = \frac{1}{n} \cdot \frac{k}{n-1} + \frac{1}{n} \sum_{i=k+1}^{n-2} \frac{k}{i} \cdot \frac{n-i}{n-1} \cdot \frac{n-i-1}{n-2}. \] (6–5)

The event of attacking the worst target occurs if the worst target is located at the \( n \)th position and the overall best position is located within the first \( k \). The
probability of this occurring is

\[ P_k(\text{worst}) = \frac{1}{n} \cdot \frac{k}{n-1}. \] (6–6)

A generalized relationship for the probability of attacking the \( j^{th} \) best target, \( j > 1 \), may be developed from the above relationships and is as follows

\[
P_k(\text{jth best}) = \begin{cases} 
\frac{1}{n} \cdot \frac{k}{n-1} & \text{if } 1 \leq k \leq n - j, \\
\frac{1}{n} \sum_{i=k+1}^{n-j} \frac{k}{n-1} \cdot \frac{n-i-1}{n-2} \cdot \ldots \cdot \frac{n-i-j+2}{n-j+1} & \text{if } n - j < k \leq n - 1.
\end{cases}
\]

6.2.1.2 Mean Rank of the Attacked Target

Let the \( j^{th} \) best target be considered the rank of that target, \( 1 \leq j \leq n \), where \( j = 1 \) is the highest value and \( j = n \) is the worst value. The mean rank of the attacked target is

\[
E[j] = \sum_{j=1}^{n} j \cdot P_k(\text{jth best}).
\] (6–7)

As a general rule a higher mean rank is desirable.

6.2.1.3 Mean Number of Examined Targets

The mean number of targets examined before an attack may be developed by considering the probability of attacking the \( i^{th} \) examined target, \( 1 \leq i \leq n \). The probability of attacking the \( i^{th} \) target seen where \( 0 < i \leq k \) is, of course, 0. The probability of attacking the \( i^{th} \) target where \( k < i < n \) is the probability the best of the \( i-1 \) is within \( k \) and the best of first \( i \) values is in position \( i \). The result is

\[
P(\text{attacking } i^{th} \text{examined target}) = \frac{k}{i-1} \cdot \frac{1}{i} \quad \text{if } k < i < n.
\]

The probability of attacking the very last target, \( i = n \), is the probability the overall best is found within \( k \) or the best of the \( n-1 \) is within \( k \) and the best of the
\( n \) is in position \( n \). This results in

\[
P(\text{attacking last target}) = \frac{k}{n} + \frac{k}{i-1} \cdot \frac{1}{i} \quad \text{if } i = n.
\]

Using the rules of expectation, the mean number of examined targets simplifies to

\[
E(\text{number of examined targets}) = k \left( \frac{n}{n - 1} + \sum_{i=k+1}^{n-1} \frac{1}{j-1} \right).
\]

### 6.2.2 Results of Best Target Strategy

Table 6–1 shows the results of the Best Target Strategy against 10, 100, and 1000 targets where the values of the targets and even the distribution of the values of the targets are unknown ahead of time. The \( k \)-value for each is optimum for the Best Target Strategy in that it maximizes \( P_k(\text{best}) \). The results in Table 6–1 were verified empirically by simulations with target values from either uniform or normal distributions. Note for even large \( n \) of targets with values that are completely unknown beforehand, the probability of attacking the most valuable target remains very favorable. Mean rank appears acceptable. For example, for \( n = 100 \) mean rank indicates that on average missions result in attacking targets within the top 21 in value. The number of targets examined appears relatively large with about 75-percent of all targets examined before a decision to attack is made.

Table 6–1: Significant results of the basic best target strategy

<table>
<thead>
<tr>
<th>( n )</th>
<th>( k )</th>
<th>( P_k(\text{best}) )</th>
<th>( P_k(\text{worst}) )</th>
<th>Mean Rank</th>
<th>Mean Examinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>4</td>
<td>0.398</td>
<td>0.044</td>
<td>3.3</td>
<td>7.98</td>
</tr>
<tr>
<td>100</td>
<td>37</td>
<td>0.371</td>
<td>0.004</td>
<td>20.01</td>
<td>74.10</td>
</tr>
<tr>
<td>1000</td>
<td>368</td>
<td>0.368</td>
<td>0.0003</td>
<td>185.54</td>
<td>736.20</td>
</tr>
</tbody>
</table>

### 6.2.3 Best Target Strategy with Threshold

Next consider a situation similar to that just studied; however, instead of skipping the first \( k \) targets, the agent will attack a target that meets or exceeds some threshold.
For example, a target of significant value may appear within the first $k$. If so, then attack the target immediately. If no target meets or exceeds the threshold value within the first $k$, then follow the basic Best Target Strategy as outlined above which is to attack the first target after $k$ whose value exceeds the maximum value found within the first $k$. Obviously, the higher the threshold value the smaller the probability that any target(s) in the battle space exceeds the threshold. A threshold set too high simply defaults the strategy to the Best Target Strategy. Too small of a threshold may result in attacking relatively lower valued targets compared to those available in the battlespace thereby lowering the probability of attacking the most valuable target. Setting a threshold relies on some knowledge of the target values which violates the assumption of no knowledge of target values or their distributions. However, use of a relatively high threshold can have improved results by increasing probability of attacking the most valuable target and decreasing the number of targets examined.

Tables 6–2, 6–3 and 6–4 show, respectively, the results of simulations with threshold values set at the upper 1-percent, 2.5-percent and 30-percent tail of uniform distributions of target values. Other simulations using normal distributions had almost identical results. These results indicate a high threshold can increase the probability of attacking the best target and decrease the number of examined targets. However, as shown in Table 6–4 the probability of attacking the best target can decrease much with a threshold set too low.

Table 6–2: Empirical results of the best target strategy with threshold at upper 1-percent tail

<table>
<thead>
<tr>
<th>$n$</th>
<th>$k$</th>
<th>$P_k$(best)</th>
<th>$P_k$(worst)</th>
<th>Mean Rank</th>
<th>Mean Examinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>4</td>
<td>0.435</td>
<td>0.0403</td>
<td>3.11</td>
<td>7.69</td>
</tr>
<tr>
<td>100</td>
<td>37</td>
<td>0.515</td>
<td>0.0014</td>
<td>8.41</td>
<td>51.11</td>
</tr>
</tbody>
</table>
Table 6–3: Empirical results of the best target strategy with threshold at upper 2.5-percent tail

<table>
<thead>
<tr>
<th>$n$</th>
<th>$k$</th>
<th>$P_k$(best)</th>
<th>$P_k$(worst)</th>
<th>Mean Rank</th>
<th>Mean Examinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>4</td>
<td>0.474</td>
<td>0.0355</td>
<td>2.89</td>
<td>7.33</td>
</tr>
<tr>
<td>100</td>
<td>37</td>
<td>0.433</td>
<td>0.0004</td>
<td>3.83</td>
<td>34.53</td>
</tr>
</tbody>
</table>

Table 6–4: Empirical results of the best target strategy with threshold at upper 30-percent tail

<table>
<thead>
<tr>
<th>$n$</th>
<th>$k$</th>
<th>$P_k$(best)</th>
<th>$P_k$(worst)</th>
<th>Mean Rank</th>
<th>Mean Examinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>4</td>
<td>0.387</td>
<td>0.0014</td>
<td>2.15</td>
<td>3.11</td>
</tr>
<tr>
<td>100</td>
<td>37</td>
<td>0.036</td>
<td>$\sim 0$</td>
<td>15.26</td>
<td>3.38</td>
</tr>
</tbody>
</table>

6.3 Maximize Mean Rank of the Attacked Target

In the preceding section a strategy is presented that maximizes the probability of attacking the most valuable target. In the long run, however, one may wish to minimize the mean rank (or maximize mean value) of the attacked targets over many missions. This section introduces a Mean Value Strategy which is superior to the Best Target Strategy in several ways.

6.3.1 Mean Value Strategy

Consider the same battlespace situation as described above. Again, the agent will examine the first $k$, $1 \leq k < n$, targets then attack the first target after $k$ that is more valuable than any of the first $k$. Equation $(6–7)$ still applies for mean rank and rank=1 is the most valued target. Instead of fixing $k$ at a value that maximizes $P_k$(best) consider fixing $k$ that minimizes mean rank (or maximizes mean value). The values of $k$ that minimize the mean rank for different values of $n$ were determined empirically and are listed in Table 6–5. To interpret the results in Table 6–5 consider $n = 100$. The mean rank of 9.6 means that on average, the Mean Value Strategy will attack $10^{th}$ best target out of 100 targets. For large $n$ mean rank is minimum at approximately $\sqrt{n}$. These results might be surprising in the sense that for large $n$,
say $n = 3000$, this strategy will result, on average, in attacking at least the 55th best target.

Table 6–5: $k$ values to minimize mean rank of attacked targets

<table>
<thead>
<tr>
<th>$n$</th>
<th>$k$</th>
<th>Mean Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2</td>
<td>2.9</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>4.2</td>
</tr>
<tr>
<td>50</td>
<td>6</td>
<td>6.7</td>
</tr>
<tr>
<td>100</td>
<td>9</td>
<td>9.6</td>
</tr>
<tr>
<td>200</td>
<td>13</td>
<td>13.7</td>
</tr>
<tr>
<td>500</td>
<td>21</td>
<td>21.9</td>
</tr>
<tr>
<td>1000</td>
<td>31</td>
<td>31.2</td>
</tr>
<tr>
<td>3000</td>
<td>54</td>
<td>54.3</td>
</tr>
</tbody>
</table>

Of course change in $k$ from that used in the Best Target Strategy reduces the probability of attacking the best target. As Figure 6–2 shows however, the Best Target Strategy results in $P_k(j^{th} \text{best})$ dropping off much faster than for the Mean Value Strategy as $j$ goes to $n$. The result is that the mean value is typically higher using the Mean Value Strategy.

Figure 6–2: Plots of probabilities of attacking the $j^{th}$ best target for two proposed strategies, $n=100$
6.3.2 Results of Mean Value Strategy

Tables 6–6 and 6–7 are the results of simulations (10^6 runs per experiment) using the Best Target Strategy and Mean Value Strategy against targets with values that are uniform in [1,1000] and normal (μ = 1000, σ = 250). Note the improvement in mean target values and reduction of mean number of target examinations when using the Mean Value Strategy.

Table 6–6: Simulation results of the best target strategy

<table>
<thead>
<tr>
<th>n</th>
<th>k</th>
<th>P_k(best)</th>
<th>P_k(worst)</th>
<th>Mean Value</th>
<th>Mean Examinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>4</td>
<td>0.397</td>
<td>0.045</td>
<td>697.4</td>
<td>7.99</td>
</tr>
<tr>
<td>100</td>
<td>37</td>
<td>0.357</td>
<td>0.004</td>
<td>795.1</td>
<td>74.6</td>
</tr>
</tbody>
</table>

Table 6–7: Simulation results of the mean target strategy

<table>
<thead>
<tr>
<th>n</th>
<th>k</th>
<th>P_k(best)</th>
<th>P_k(worst)</th>
<th>Mean Value</th>
<th>Mean Examinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>4</td>
<td>0.399</td>
<td>0.045</td>
<td>1178.2</td>
<td>7.98</td>
</tr>
<tr>
<td>100</td>
<td>37</td>
<td>0.370</td>
<td>0.004</td>
<td>1358.4</td>
<td>74.1</td>
</tr>
</tbody>
</table>

6.4 Number of Targets is a Random Variable

Both the Best Target Strategy and the Mean Value Strategy described in the earlier sections rely on knowing n. n may, of course, be some random variable. Consider the case where an agent has a fixed flight time, t. Assume for example, the agent runs out of fuel after t time into the mission. If we make a further assumption that the time between targets is exponentially distributed with some mean, λ, then the number of targets becomes a Poisson random variable with mean \( \frac{t}{\lambda} \). Of course other
distributions for the number of targets may be appropriate. This section discusses the impacts of \( n \) being some random variable.

Consider three possible outcomes for \( n \). Let \( n_{\text{expected}} \) be the expected number of targets and let \( n_{\text{realized}} \) be the actual number of targets in the battlespace. If \( n_{\text{realized}} = n_{\text{expected}} \) then no impact on strategy. If \( n_{\text{realized}} > n_{\text{expected}} \) then the agent would default to attacking \( n_{\text{expected}} \) if no acceptable target is found before then. If \( n_{\text{realized}} < n_{\text{expected}} \) then it is possible no target is attacked.

Tables 6–8, 6–9 and 6–10 are the results of simulations (10\(^6\) runs per experiment) using the Best Target Strategy and Mean Value Strategy against targets with values that are uniform in \([1,1000]\) and normal (\(\mu = 1000, \sigma = 250\)). The number of targets in Table 6–8 are Poisson distributed with mean \( n \). The number of targets in Table 6–9 are approximately normally distributed (non-negative and rounded to nearest integer) with mean \( n \) and standard deviation \(0.2n\). And finally, the number of targets in Table 6–10 are uniformly distributed in \([0.5n, 1.5n]\). In these results, worst case is not attacking any target and the value of not attacking a target is zero.

A review of this data reveals the Mean Value Strategy is more robust to a variable \( n \). A summary of performance of the two strategies for uniform target values (results are similar for normally distributed target values) is given in Table 6–11. The summary shows how much average performance changes when \( n \) varies compared to a fixed \( n \). Table 6–11 shows the average percentage drop in probability of attacking the most valuable target, the mean value of attacked target and the mean number of examinations before an attack is made. The results also indicate that both strategies tend to be less impacted by a varying \( n \) as \( n \) gets large.

The preceding results can be improved if during the mission the agent updates its expected value of \( n \). As an example of this, we assume we know \( n \); however, due to atmospheric conditions, sensor performance may be either degraded or enhanced. This results in the agent detecting fewer or more targets than expected. The
Table 6–8: Simulation results with number of targets poisson distributed, mean $n$

<table>
<thead>
<tr>
<th>$n$</th>
<th>$k$</th>
<th>$P_k$ (best)</th>
<th>$P_k$ (no target)</th>
<th>Mean Value</th>
<th>Mean Examinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform Target Values, [1,1000]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>0.348</td>
<td>0.135</td>
<td>651.1</td>
<td>5.33</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>0.328</td>
<td>0.265</td>
<td>542.0</td>
<td>7.42</td>
</tr>
<tr>
<td>100</td>
<td>9</td>
<td>0.212</td>
<td>0.050</td>
<td>875.6</td>
<td>31.1</td>
</tr>
<tr>
<td>100</td>
<td>37</td>
<td>0.343</td>
<td>0.202</td>
<td>687.8</td>
<td>72.9</td>
</tr>
<tr>
<td>Normal Target Values, $\mu = 1000, \sigma = 250$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>0.348</td>
<td>0.135</td>
<td>1055.4</td>
<td>5.33</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>0.330</td>
<td>0.264</td>
<td>892.9</td>
<td>7.42</td>
</tr>
<tr>
<td>100</td>
<td>9</td>
<td>0.218</td>
<td>0.048</td>
<td>1370.2</td>
<td>30.8</td>
</tr>
<tr>
<td>100</td>
<td>37</td>
<td>0.356</td>
<td>0.197</td>
<td>1150.8</td>
<td>72.6</td>
</tr>
</tbody>
</table>

Table 6–9: Simulation results with number of targets normally distributed, mean $n$ and standard deviation $0.2n$

<table>
<thead>
<tr>
<th>$n$</th>
<th>$k$</th>
<th>$P_k$ (best)</th>
<th>$P_k$ (no target)</th>
<th>Mean Value</th>
<th>Mean Examinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform Target Values, [1,1000]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>0.363</td>
<td>0.132</td>
<td>657.2</td>
<td>5.40</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>0.361</td>
<td>0.261</td>
<td>552.2</td>
<td>7.47</td>
</tr>
<tr>
<td>100</td>
<td>9</td>
<td>0.213</td>
<td>0.057</td>
<td>868.7</td>
<td>30.6</td>
</tr>
<tr>
<td>100</td>
<td>37</td>
<td>0.327</td>
<td>0.232</td>
<td>660.5</td>
<td>71.0</td>
</tr>
<tr>
<td>Normal Target Values, $\mu = 1000, \sigma = 250$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>0.365</td>
<td>0.131</td>
<td>1064.8</td>
<td>5.39</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>0.362</td>
<td>0.262</td>
<td>902.4</td>
<td>7.47</td>
</tr>
<tr>
<td>100</td>
<td>9</td>
<td>0.220</td>
<td>0.055</td>
<td>1359.2</td>
<td>30.3</td>
</tr>
<tr>
<td>100</td>
<td>37</td>
<td>0.338</td>
<td>0.227</td>
<td>1105.3</td>
<td>70.6</td>
</tr>
</tbody>
</table>

Knowledge of number of targets encountered by some time into the mission can be used to update the expected value of $n$. This should result in fewer non-attacks because the number of targets was overestimated. It may increase the mean value of the attacked target because the agent may examine more targets if they exist. Indeed, Table 6–12 shows the simulation results if, at about 90-percent into the mission time, the agent updates the expected number of targets. This simulation assumes exponentially distributed interarrival times with some mean, $\lambda$, for encountering targets. The results show that the mean value of the attacked targets increases by a notable amount.
Table 6–10: Simulation results with number of targets uniformly distributed in $[0.5n, 1.5n]$

<table>
<thead>
<tr>
<th>$n$</th>
<th>$k$</th>
<th>$P_k$(best)</th>
<th>$P_k$(no target)</th>
<th>Mean Value</th>
<th>Mean Examinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2</td>
<td>0.349</td>
<td>0.136</td>
<td>650.1</td>
<td>5.29</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>0.323</td>
<td>0.271</td>
<td>535.9</td>
<td>7.25</td>
</tr>
<tr>
<td>100</td>
<td>9</td>
<td>0.214</td>
<td>0.064</td>
<td>862.0</td>
<td>30.0</td>
</tr>
<tr>
<td>100</td>
<td>37</td>
<td>0.306</td>
<td>0.261</td>
<td>631.2</td>
<td>68.7</td>
</tr>
</tbody>
</table>

Table 6–11: Performance summary of best target strategy and mean value strategy when $n$ varies; values are in percentage drop compared to when $n$ is fixed

<table>
<thead>
<tr>
<th>$n$</th>
<th>$P_k$(best)</th>
<th>Mean Value</th>
<th>Mean Examinations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>best</td>
<td></td>
</tr>
<tr>
<td>Best Target Strategy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>15.0</td>
<td>22.1</td>
<td>7.6</td>
</tr>
<tr>
<td>100</td>
<td>8.9</td>
<td>17.0</td>
<td>5.0</td>
</tr>
<tr>
<td>Mean Value Strategy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2.9</td>
<td>10.6</td>
<td>5.7</td>
</tr>
<tr>
<td>100</td>
<td>0.5</td>
<td>3.7</td>
<td>2.7</td>
</tr>
</tbody>
</table>

If the probability of attacking no target can be reduced, this should increase both probability of attacking the best target and the mean value of the attacked target. One way to do this is update $n$ only when $n$ is overestimated. Simply put, allow $n_{\text{expected}}$ to be reduced and not increased. Table 6–13 shows the desired results are obtained when expected number of targets is allowed to be lowered but not increased. The optimum results are obtained when the update is made at about the 95-percent point into the mission time (in other words near the end of the mission).

### 6.5 Target Strategy with Sampling-The Learning Agent

Obviously, the Best Target Strategy and Mean Value Strategy use a very simple piece of information available to make a decision. That is they use only the maximum
Table 6–12: Simulation results with expected number of targets, $n$, updated 90-percent into the mission

<table>
<thead>
<tr>
<th>$n$</th>
<th>$k$</th>
<th>$P_k$(best)</th>
<th>$P_k$(no target)</th>
<th>Mean Value</th>
<th>Mean Examinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2</td>
<td>0.333</td>
<td>0.094</td>
<td>678.0</td>
<td>5.19</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>0.334</td>
<td>0.185</td>
<td>595.3</td>
<td>7.43</td>
</tr>
<tr>
<td>100</td>
<td>9</td>
<td>0.210</td>
<td>0.031</td>
<td>884.9</td>
<td>29.1</td>
</tr>
<tr>
<td>100</td>
<td>37</td>
<td>0.344</td>
<td>0.125</td>
<td>726.8</td>
<td>70.0</td>
</tr>
</tbody>
</table>

Uniform Target Values, $[1,1000]$

<table>
<thead>
<tr>
<th>$n$</th>
<th>$k$</th>
<th>$P_k$(best)</th>
<th>$P_k$(no target)</th>
<th>Mean Value</th>
<th>Mean Examinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2</td>
<td>0.334</td>
<td>0.095</td>
<td>1116.8</td>
<td>5.19</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>0.334</td>
<td>0.183</td>
<td>1012.9</td>
<td>7.33</td>
</tr>
<tr>
<td>100</td>
<td>9</td>
<td>0.218</td>
<td>0.030</td>
<td>1388.6</td>
<td>28.9</td>
</tr>
<tr>
<td>100</td>
<td>37</td>
<td>0.356</td>
<td>0.122</td>
<td>1229.0</td>
<td>69.6</td>
</tr>
</tbody>
</table>

Normal Target Values, $\mu = 1000, \sigma = 250$

<table>
<thead>
<tr>
<th>$n$</th>
<th>$k$</th>
<th>$P_k$(best)</th>
<th>$P_k$(no target)</th>
<th>Mean Value</th>
<th>Mean Examinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2</td>
<td>0.332</td>
<td>0.087</td>
<td>680.7</td>
<td>5.18</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>0.335</td>
<td>0.166</td>
<td>603.8</td>
<td>7.24</td>
</tr>
<tr>
<td>100</td>
<td>9</td>
<td>0.210</td>
<td>0.018</td>
<td>891.1</td>
<td>30.0</td>
</tr>
<tr>
<td>100</td>
<td>37</td>
<td>0.340</td>
<td>0.072</td>
<td>751.9</td>
<td>71.4</td>
</tr>
</tbody>
</table>

Table 6–13: Simulation results with expected number of targets, $n$, updated near the end of the mission. $n$ may be updated downward, but not upward.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$k$</th>
<th>$P_k$(best)</th>
<th>$P_k$(no target)</th>
<th>Mean Value</th>
<th>Mean Examinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2</td>
<td>0.332</td>
<td>0.087</td>
<td>680.7</td>
<td>5.18</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>0.335</td>
<td>0.166</td>
<td>603.8</td>
<td>7.24</td>
</tr>
<tr>
<td>100</td>
<td>9</td>
<td>0.210</td>
<td>0.018</td>
<td>891.1</td>
<td>30.0</td>
</tr>
<tr>
<td>100</td>
<td>37</td>
<td>0.340</td>
<td>0.072</td>
<td>751.9</td>
<td>71.4</td>
</tr>
</tbody>
</table>

value of the targets within $k$. A strategy modification to consider is to use more information about the values of the targets encountered within $k$. For example, although the values (or their distributions) of the targets are unknown beforehand, the agent learns more about the target-value distribution as examinations are made. Information on mean and standard deviation of the sample may be useful in the decision making process. Suppose that instead of using the maximum value within $k$, the agent uses mean of the sample plus some factor of the standard deviation as a decision threshold. In addition, suppose that after $k$, as potential targets are
encountered but not attacked, their values are used to update the sample mean and sample standard deviation. In turn, the sample mean and standard deviation can be used to further refine the threshold used in the decision making process. This threshold will be developed shortly based on order statistics.

Using the idea of order statistics and given a uniform distribution in $[0,1]$ of $n$ variables, it can be shown the expected value of the maximum value is

$$E[\text{maximum value}] = \frac{n}{n+1}.$$ 

For example, if given 10 variables drawn from a uniform distribution in $[0,1]$, the expected value of the highest value is $\frac{10}{11}$.

A potential decision strategy for attacking a target is to set the decision threshold at the value of the expected highest value of the remaining targets. In the example of $n$ targets whose values are uniformly distributed in $[0,1]$, an agent will decide to attack the $i$th target if

$$\text{ith Target Value} \geq \begin{cases} \frac{n+1-i}{n+2-i}, & \text{if } 1 \leq i \leq n-1, \\ 0, & \text{if } i = n. \end{cases} \quad (6-8)$$

The noticeable flaw with the above strategy is the distribution of the targets is unknown. We can use entropy to overcome this flaw. We develop using the result from (6-8) to some threshold. We propose a strategy where the agent will examine the first $k$, $1 \leq k < n$, targets then attack the first target after $k$ that is equal or more valuable than threshold, $T$, such that

$$T = \begin{cases} H(p)(\frac{n+1-i}{n+2-i})(\text{max} - \text{min}) + \text{min}, & \text{if } k < i \leq n-1, \\ 0, & \text{if } i = n, \end{cases}$$
where $H(p)$ is a scaling factor and $max$ and $min$ are calculated as follows

$$
max = \eta + \gamma \varsigma,
$$
$$
min = \eta - \gamma \varsigma,
$$

where $\eta$ is sample mean, $\varsigma$ is sample standard deviation, and $\gamma$ is some scaling factor for the sample standard deviation. Through simulations, applicable values for $\rho$ and $\gamma$ were found to be 0.95 and 2.0, respectively. The sample mean and standard deviation are determined from the values of targets up to but not including $i^{th}$ target.

The results of simulations ($10^6$ runs per experiment) of this strategy are summarized in Table 6–14. We found the best value for $k$ is the mean of the optimum values of $k$ for the Best Target Strategy and Mean Value Strategy. Referring back to Tables 6–6 and 6–7, one may note much improvement for the mean value of the attacked targets (over 5-percent improvement in the case of $n = 100$ and normally distributed target values). A drawback of this strategy compared to the Mean Value Strategy is the number of target examinations is higher.

Table 6–14: Simulation results of the target strategy with sampling

<table>
<thead>
<tr>
<th>$n$</th>
<th>$k$</th>
<th>$P_k$(best)</th>
<th>$P_k$(worst)</th>
<th>Mean Value</th>
<th>Mean Examinations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Uniform Target Values, [1,1000]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>0.350</td>
<td>0.023</td>
<td>752.7</td>
<td>7.08</td>
</tr>
<tr>
<td>100</td>
<td>23</td>
<td>0.196</td>
<td>0.001</td>
<td>921.1</td>
<td>63.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Normal Target Values, $\mu = 1000, \sigma = 250$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>0.381</td>
<td>0.023</td>
<td>1226.1</td>
<td>7.08</td>
</tr>
<tr>
<td>100</td>
<td>23</td>
<td>0.210</td>
<td>~ 0</td>
<td>1497.5</td>
<td>42.7</td>
</tr>
</tbody>
</table>

This strategy may be interpreted as a “learning-agent” strategy since as the agent examines targets it learns more about the target-value distributions. Because complete knowledge about the target distributions should result in the best decisions, we believe more research is warranted for this strategy.
6.6 Multiple Agents

Consider the situation in Figure 6–3 where multiple agents, each carrying one weapon, are to search a battlespace and attack separate targets (although not examined in this paper, more than one agent attacking the same target may be appropriate where given agent $y$ attacks target $z$, the probability of agent $y$ disabling target $z$, $p_{yz}$, is less than 1). Again, the goal is to attack the most valuable targets. Two search and attack strategies will be considered in the following subsections. The first strategy is that of agents searching together as a pack. A pack may be advantageous because as a pack, multiple looks at a single target (coordinated sensing) can result in a higher probability of a correct target classification [36]. The second strategy is that of agents searching separately with and without communication. Agents searching separately obviously results in more ground being covered in a shorter amount of time.

Figure 6–3: $m$ agents performing racetrack search of battlespace with $n$ targets

6.6.1 Agents as a Pack

As depicted in the upper left of Figure 6–3 a strategy proposed here is $m$ agents, $m \leq n$, examine the first $k$, $1 \leq k < n$, targets then the first agent attacks the first
target after \( k \) that is more valuable than any of the first \( k \) and the next agent attacks the next target that is more valuable than any of the first \( k \), and so on. \( k \) should be set equal to a value that maximizes \( P_k(\text{best}) \) or maximizes the mean value of the attacked targets. If the most valuable target is within \( k \) then the pack of \( m \) agents will attack the last \( m \) targets regardless of value. We expect the following outcomes when using multiple agents as a pack

- \( P_k(\text{best})_m \) should be higher
- The mean value of all attacked targets will go down because additional agents will obviously attack targets of lesser value than the best
- The mean value of the best target attacked will be higher because there is a higher chance of attacking the best target when using more than one agent
- The number of target examinations should go up since it will take longer for all agents to find an appropriate target

where \( P_k(\text{best})_m \) is the probability of attacking the best target with \( m \) agents. To develop the relation for the probability of attacking the most valuable target when using multiple agents as a pack we begin by examining the probability of attacking the most valuable target when using just two agents, then three and finally \( m \). This development will lead to a generalized closed form expression for \( P_k(\text{best})_m \).

By using conditional probability where \( X \) is the position of the most valuable target and ensuring \( 0 < k < n - 2 \) an expression for two agents searching as a pack is developed as follows

\[
P_k(\text{best})_2 = \sum_{i=1}^{n} P_k(\text{best}|X = i) \cdot 2P(X = i),
\]

\[
= \sum_{i=1}^{n} P_k(\text{best}|X = i) \cdot \frac{1}{n}.
\]

\[(6\,9)\]
To complete (6–9), expressions are needed for \( P_k(\text{best}|X = i) \), which follow

\[
P_k(\text{best}|X = i) = 0 \quad \text{if } 1 \leq i \leq k, \quad (6–10a)
\]

\[
= \frac{k}{i - 1} + \frac{k}{i - 1} \cdot \frac{i - 1 - k}{i - 2} \quad \text{if } k < i \leq n. \quad (6–10b)
\]

If the best target is among the first \( k \), then there is no chance of attacking it; therefore, (6–10a) holds. Equation (6–10b) holds because; 1) it is the probability of the best of the first \( i - 1 \) is among the first \( k \) in which case the first agent strikes, or 2) the third best is among the first \( k \) and the second best is among the targets from \( k + 1 \) to \( i - 1 \) in which case the second agent strikes.

Inserting (6–10) back into (6–9)

\[
P_k(\text{best}) = \frac{1}{n} \sum_{i=k+1}^{n} \left( \frac{k}{i - 1} + \frac{k}{i - 1} \cdot \frac{i - 1 - k}{i - 2} \right),
\]

\[
= \frac{k}{n} \sum_{i=k}^{n-1} \left( \frac{1}{i} + \frac{i - k}{i(i - 1)} \right) \quad \text{if } 0 < k < n - 2. \quad (6–11)
\]

Similarly, if we ensure \( 0 < k < n - 3 \) an expression for \( P_k(\text{best})_3 \) can be developed as

\[
P_k(\text{best})_3 = \frac{k}{n} \sum_{i=k}^{n-1} \left( \frac{1}{i} + \frac{i - k}{i(i - 1)} + \frac{(i - k)(i - k - 1)}{i(i - 1)(i - 2)} \right) \quad \text{if } 0 < k < n - 3. \quad (6–12)
\]

A generalized relationship for the probability of attacking the \( P_k(\text{best})_m \) best target may be developed from the above relationships and is as follows

\[
P_k(\text{best})_m = \frac{k}{n} \sum_{i=k}^{n-1} \left( \frac{1}{i} + \frac{i - k}{i(i - 1)} + \frac{(i - k)(i - k - 1)}{i(i - 1)(i - 2)} + \cdots \right.
\]

\[
\left. + \frac{(i-k)(i-k-1) \cdots (i-k-m+2)}{i(i-1)(i-2) \cdots (i-m+1)} \right) \quad \text{if } 0 < k < n - m.
\]

For large \( n \), the following formula may be derived that provides the value of \( k \) that maximizes \( P_k(\text{best})_m \) given pack of \( m \) agents searching \( n \) targets

\[
k = \frac{n}{e^{1 + \frac{m - 1}{m + 2}}}. 
\]
The results of simulations ($10^6$ runs per experiment) of this strategy against targets with uniformly distributed target values are summarized in Table 6–15. Simulations against normally distributed target values are not included for compactness; however, the results are similar. Table 6–15 includes statistics concerning the value of attacked targets. The mean value is the overall mean of the targets attacked by all agents. The mean best statistic is the mean of the value of the best target attacked by the pack of agents. Mean examinations is the average number of target examinations made by the pack of agents. The values for $k$ were selected to either optimize $P_k$(best)$_m$ or mean best. The asterisk placed next to a value in the table indicates it was the parameter optimized by the selection of $k$. As expected the optimal value of $k$ for $P_k$(best)$_m$ reduced as the number of agents increased; however, the $k$ that maximized mean best remained at or near the value that is optimal for the Mean Value Strategy for a single agent.

Table 6–15: Simulation results of the target strategy with $m$ agents in a pack. Target values are uniform on [0,1000].

<table>
<thead>
<tr>
<th>$m$</th>
<th>$n$</th>
<th>$k$</th>
<th>$P_k$(best)</th>
<th>$P_k$(worst)</th>
<th>Mean Value</th>
<th>Mean Best</th>
<th>Mean Examinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>2</td>
<td>0.552</td>
<td>0.067</td>
<td>686.5</td>
<td>808.6*</td>
<td>5.66</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>3</td>
<td>0.562*</td>
<td>0.067</td>
<td>663.6</td>
<td>798.0</td>
<td>7.54</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>2</td>
<td>0.661*</td>
<td>0.134</td>
<td>645.4</td>
<td>843.8</td>
<td>8.64</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>2</td>
<td>0.727*</td>
<td>0.223</td>
<td>608.3</td>
<td>862.4</td>
<td>9.30</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>9</td>
<td>0.345</td>
<td>0.002</td>
<td>881.8</td>
<td>934.3*</td>
<td>45.2</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>30</td>
<td>0.495*</td>
<td>0.009</td>
<td>775.9</td>
<td>883.6</td>
<td>82.0</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>9</td>
<td>0.440</td>
<td>0.005</td>
<td>862.1</td>
<td>950.2*</td>
<td>55.3</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>26</td>
<td>0.573*</td>
<td>0.014</td>
<td>758.0</td>
<td>920.3</td>
<td>86.3</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>9</td>
<td>0.513</td>
<td>0.009</td>
<td>843.4</td>
<td>959.5*</td>
<td>62.9</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>23</td>
<td>0.626*</td>
<td>0.020</td>
<td>744.7</td>
<td>940.2</td>
<td>88.8</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>8</td>
<td>0.550</td>
<td>0.012</td>
<td>832.8</td>
<td>965.9*</td>
<td>65.7</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>21</td>
<td>0.664*</td>
<td>0.025</td>
<td>739.2</td>
<td>953.5</td>
<td>89.8</td>
</tr>
</tbody>
</table>

6.6.2 Separate Agents

The next strategy for multiple agents is that of separate missions as depicted in Figure 6–4 where each agent is assigned to equal partitions of the battlespace.
Two substrategies are considered: 1) without communications, and 2) with communications.

6.6.2.1 Separate Agents without Communication

Consider the situation where agents independently search their respectively partition without any interaction. For example, if \( n = 100 \) and \( m = 2 \), the battlespace is partitioned into two equal parts each with \( n = 50 \) targets. Each agent then performs its mission within its partition. That is each agent agent will examine the first \( k \), \( 1 \leq k < n/m \), targets then attack the first target after \( k \) that is more valuable than any of the first \( k \). We expect similar impacts to the pertinent performance measures as with agents in a pack. That is \( P_k(\text{best})_m \) and average value of the best target attacked should go up as more agents are added. In this model \( P_k(\text{best})_m \) is calculated similarly to a single agent attacking a battlespace using the Best Target Strategy. This is

\[
P_k(\text{best})_m = \frac{k}{n/m} \sum_{i=k}^{n/m-1} \frac{1}{i}.
\]
$k$ is selected such that the agent maximizes the probability of attacking the best target. The value of $k$ that maximizes this probability is $k = \frac{n}{m} \text{ e}$ rounded to nearest integer. Table 6–16 shows the results of simulations ($10^6$ runs per experiment) of this strategy against targets with uniformly distributed target values. Note the mean examinations is based on the total number of targets examined by $m$ agents.

Table 6–16: Simulation results of the target strategy with $m$ agents on separate missions with no communication between them. Target values are uniform on $[0,1000]$.

<table>
<thead>
<tr>
<th>$m$</th>
<th>$n$</th>
<th>$k$</th>
<th>$P_k$(best)</th>
<th>$P_k$(worst)</th>
<th>Mean Value</th>
<th>Mean Best</th>
<th>Mean Examinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>1</td>
<td>0.416</td>
<td>0.051</td>
<td>647.5</td>
<td>792.9*</td>
<td>3.74</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>2</td>
<td>0.431*</td>
<td>0.100</td>
<td>630.6</td>
<td>786.6</td>
<td>4.64</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>9</td>
<td>0.304</td>
<td>0.004</td>
<td>857.2</td>
<td>952.3*</td>
<td>33.0</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>18</td>
<td>0.358*</td>
<td>0.008</td>
<td>781.6</td>
<td>925.7</td>
<td>44.3</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>7</td>
<td>0.355</td>
<td>0.013</td>
<td>795.0</td>
<td>968.6*</td>
<td>23.2</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>9</td>
<td>0.364*</td>
<td>0.016</td>
<td>767.3</td>
<td>966.9</td>
<td>24.1</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>6</td>
<td>0.362</td>
<td>0.017</td>
<td>775.7</td>
<td>970.6*</td>
<td>19.6</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>7</td>
<td>0.367*</td>
<td>0.019</td>
<td>759.9</td>
<td>970.3</td>
<td>19.8</td>
</tr>
</tbody>
</table>

### 6.6.2.2 Separate Agents with Communication

We now consider the same situation as just described, but this time the agents communicate by informing others of their maximum within $k$. Table 6–17 shows the results of simulations ($10^6$ runs per experiment) of this strategy against targets with uniformly distributed target values.

Table 6–17: Simulation results of the target strategy with $m$ agents on separate missions with communication between them. Target values are uniform on $[0,1000]$.

<table>
<thead>
<tr>
<th>$m$</th>
<th>$n$</th>
<th>$k$</th>
<th>$P_k$(best)</th>
<th>$P_k$(worst)</th>
<th>Mean Value</th>
<th>Mean Best</th>
<th>Mean Examinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>1</td>
<td>0.511</td>
<td>0.081</td>
<td>663.8</td>
<td>803.8</td>
<td>4.05</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>4</td>
<td>0.301</td>
<td>0.003</td>
<td>867.7</td>
<td>933.7</td>
<td>24.9</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>15</td>
<td>0.450</td>
<td>0.010</td>
<td>748.4</td>
<td>883.0</td>
<td>43.6</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>2</td>
<td>0.429</td>
<td>0.011</td>
<td>812.7</td>
<td>958.3</td>
<td>18.9</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>6</td>
<td>0.544</td>
<td>0.024</td>
<td>696.4</td>
<td>937.4</td>
<td>24.1</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>2</td>
<td>0.508</td>
<td>0.019</td>
<td>773.4</td>
<td>963.8</td>
<td>17.5</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>4</td>
<td>0.573</td>
<td>0.030</td>
<td>694.9</td>
<td>952.2</td>
<td>19.4</td>
</tr>
</tbody>
</table>
In the situation just described, agents investigate their independent partitions of the battlespace. Consider a case where say agent $A$ in one partition attacks early. This leaves much of $A$’s partition unsearched. If another agent, $B$, is still active, it may be advantageous to allow $B$ investigate the remaining part of $A$’s partition. $B$ will search until it finds an appropriate target or until it reaches the end of $A$’s partition. This can only occur if $A$ and $B$ communicate by telling each other when and where they made their attack. Table 6–18 shows the results of simulations against targets with uniform target values. The first part is where agents do not share the maximum target value found within $k$ and the second part is where agents have full communication – that is both maximum target value found within $k$ and when and where they attack.

Table 6–18: Simulation results of the target strategy with $m$ agents on separate missions with communication between them. Uncommitted agents are allowed to evaluate targets in other unsearched partitions.

<table>
<thead>
<tr>
<th>$m$</th>
<th>$n$</th>
<th>$k$</th>
<th>$P_k$ (best)</th>
<th>$P_k$ (worst)</th>
<th>Mean Value</th>
<th>Mean Best</th>
<th>Mean Examinations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Not sharing information on maximum target value within $k$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>1</td>
<td>0.481</td>
<td>0.036</td>
<td>682.2</td>
<td>810.4</td>
<td>6.73</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>2</td>
<td>0.494</td>
<td>0.086</td>
<td>655.6</td>
<td>799.8</td>
<td>8.77</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>9</td>
<td>0.372</td>
<td>0.003</td>
<td>884.7</td>
<td>957.7</td>
<td>57.0</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>17</td>
<td>0.434</td>
<td>0.006</td>
<td>830.9</td>
<td>937.7</td>
<td>79.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Sharing information on maximum target value within $k$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>1</td>
<td>0.551</td>
<td>0.067</td>
<td>686.3</td>
<td>808.5</td>
<td>7.45</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>2</td>
<td>0.528</td>
<td>0.123</td>
<td>630.9</td>
<td>777.4</td>
<td>9.17</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>4</td>
<td>0.326</td>
<td>0.003</td>
<td>883.3</td>
<td>934.1</td>
<td>42.2</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>14</td>
<td>0.494</td>
<td>0.008</td>
<td>786.9</td>
<td>889.7</td>
<td>79.8</td>
</tr>
</tbody>
</table>

6.6.2.3 Separate Agents Comparison

In comparing the performance of the two strategies (separate agents without communications and separate agents with communications) we observe the best form of communications is when attacking agents inform other agents when and where their attack is made and then allowing uncommitted agents to evaluate other unsearched
partitions. There appears to be no advantage for agents communicating the maximum target value found within their first respective \( k \) evaluations.

When the ratio of agents to targets is higher, communication is beneficial for both \( P_k(\text{best})_m \) and mean best; however, when the ratio is lower, it appears there is no advantage to have communications.

### 6.6.3 Multiple Agent Strategy—Discussion

While operating under the given assumptions and comparing the two strategies of either a pack of agents or separate agents, we observe that when there are few agents and many targets, then a separate-agent strategy with communication is the best strategy. However, when the ratio of agents to targets increases, then the best strategy is agents hunting as a pack.

### 6.7 Dynamic Threshold

In all preceding sections we assume the target values are unknown \textit{a priori}. However, in actuality, the war fighter may have knowledge of the maximum and minimum target values that are possible in the battlespace. This knowledge can come from experience, intelligence or previous missions and can be very useful in increasing both \( P_k(\text{best}) \) and mean value of attacked target. A strategy to consider is that of a dynamic decision threshold [42]. Consider as before, the number of targets, \( n \), is known. Unlike earlier assumptions, we now assume we know the maximum and minimum values of the target-value distribution and we assume the target values are uniformly distributed. A strategy derived by induction that maximizes \( P_k(\text{best}) \) is to set a threshold \( T \) such that the agent attacks a target if its value exceeds \( T \). \( T \) is determined as follows

\[
T = \begin{cases} 
0.5^{i-1}(\text{max} - \text{min}) + \text{min} & \text{if } 1 \leq i \leq n-1, \\
0 & \text{if } i = n,
\end{cases}
\]  

(6–13)
where $\textit{max}$ and $\textit{min}$ are maximum and minimum target values, respectively, that are possible in the battlespace. To summarize the dynamic threshold strategy, an agent will find and examine targets in succession. The agent will attack the first target whose value exceeds $T$ as defined in (6–13). Although (6–13) is derived based on an assumption of uniform target values, we found that it is very effective with other distributions such as the normal distribution where one assumes $\textit{max}$ and $\textit{min}$ are, respectively, two standard deviations above and below the mean. The results of simulations ($10^6$ runs per experiment) of the dynamic threshold strategy are summarized in Table 6–19. These results indicate, as expected, the more an agent knows about the target-value distribution, the better the results.

Table 6–19: Simulation results using the dynamic threshold strategy

<table>
<thead>
<tr>
<th>$n$</th>
<th>$P_k$ (best)</th>
<th>Mean Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Uniform Target Values, [1,1000]</td>
</tr>
<tr>
<td>10</td>
<td>0.574</td>
<td>836.5</td>
</tr>
<tr>
<td>100</td>
<td>0.528</td>
<td>962.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Normal Target Values, $\mu = 1000, \sigma = 250$</td>
</tr>
<tr>
<td>10</td>
<td>0.516</td>
<td>1278.5</td>
</tr>
<tr>
<td>100</td>
<td>0.463</td>
<td>1556.1</td>
</tr>
</tbody>
</table>

6.8 Conclusion

We have shown that an agent will find and examine targets in succession. The agent will attack the first target whose value exceeds $T$ as defined in (6–13). Although (6–13) is derived based on an assumption of uniform target values, we found that it is very effective with other distributions such as the normal distribution where one assumes $\textit{max}$ and $\textit{min}$ are, respectively, two standard deviations above and below the mean. The results of simulations ($10^6$ runs per experiment) of the dynamic threshold strategy are summarized in Table 6–19. These results indicate, as expected, the more an agent knows about the target-value distribution, the better the results.
CHAPTER 7
CONCLUDING REMARKS AND FUTURE RESEARCH

7.1 Summary

The goal of this research is to employ entropy optimization in data mining and feature extraction in data. In this dissertation, we have developed minimum and maximum entropy based approaches that allow us to extract essential information from the data in order to effectively cluster and classify data. Most data when the data set

Several methods for mining data exist, however most of these methods suffer from several limitations especially that of specifying the data distribution which is usually unavailable. Equally important in data mining is the effective interpretability of the results. Data to be clustered often exist in very high dimensions which most of the existing methods do not handle but instead rely on some data preprocessing. The entropy methods developed in this thesis are well suited for handling these problems. Our methods eliminates making distributional assumptions which may or may not exist. Dimension reduction for better data result comprehension are also achieved by the entropy methods developed in this dissertation. We have also successfully applied these entropy methods in best target selection and path planning which are areas of research interest.

7.2 Future Research

There are several important directions to extend the work presented in this dissertation. One such area is finance. In financial analysis, an investor is always interested in a model’s performance. He therefore must evaluate models based on the performance of the strategies that the model suggests. This performance measure of the models can be evaluated using the principles of relative entropy. In macro
econometric modelling and policy analysis, the empirical models that forecast well are typically nonstructural, yet making the kinds of theoretically coherent forecasts policy makers wish to see requires imposing structure that may be difficult to implement and that in turn often makes the model empirically irrelevant. Cross entropy procedure can be used to produce forecasts that are consistent with a set of moment restrictions without imposing them directly on the model.
REFERENCES


BIOGRAPHICAL SKETCH

Anthony Okafor was born in Awgu, Nigeria. He received a bachelor’s degree in mechanical engineering from the University of Nigeria, Nsukka. He later immigrated to the United States where he earned a master’s degree in mathematical sciences from the University of West Florida, Pensacola, and a master’s degree in industrial and systems engineering (ISE) from the University of Florida. He is a Ph.D. student in the ISE department, University of Florida. His research interests include mathematical programming, entropy optimization and operations research.

As for hobbies, Anthony loves table tennis, badminton and tennis. He is also actively involved with home automation and enjoys mentoring.