

# Computer Analysis, Learning and Creation of Physical Arrangements of Information

by

Michael Alan Kahan

Submitted to the Department of Electrical Engineering and Computer  
Science

in partial fulfillment of the requirements for the degree of

Masters of Engineering in Electrical Engineering and Computer  
Science

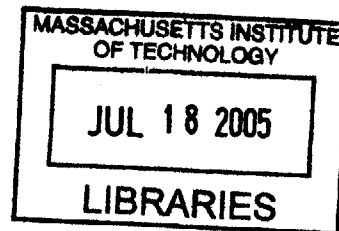
at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2004

© Michael Alan Kahan, MMIV. All rights reserved.

The author hereby grants to MIT permission to reproduce and  
distribute publicly paper and electronic copies of this thesis document  
in whole or in part.



Author .....  
Department of Electrical Engineering and Computer Science  
August 17, 2004

Certified by .....  
Patrick H. Winston  
Ford Professor of Artificial Intelligence and Computer Science  
Thesis Supervisor

Accepted by ....  
Arthur C. Smith  
Chairman, Department Committee on Graduate Students



# Computer Analysis, Learning and Creation of Physical Arrangements of Information

by

Michael Alan Kahan

Submitted to the Department of Electrical Engineering and Computer Science  
on August 17, 2004, in partial fulfillment of the  
requirements for the degree of  
Masters of Engineering in Electrical Engineering and Computer Science

## Abstract

Humans' ability to arrange the individual pieces of a set of information is paramount to their understanding of the set as a whole. The physical arrangement of pieces of information yields important clues as to how those pieces are related. This thesis focuses on computer analysis of physical arrangements and use of perceived physical relations, such as horizontal and vertical alignment, in determining which pieces of information are most likely related. The computer program described in this thesis demonstrates that once a computer can deduce physical relations between pieces of information, it can learn to order the information as a human would with great accuracy. The information analysis methods presented in this thesis are of benefit to projects that deal with user collaboration and the sorting of data based on relative importance, such as the Electronic Card Wall (EWall) project.

Thesis Supervisor: Patrick H. Winston

Title: Ford Professor of Artificial Intelligence and Computer Science



## Acknowledgments

First, I would like to thank Professor Winston for introducing me to the field of AI, encouraging my interest in the EWall project and acting as my advisor in more than just my academic endeavors.

I thank Paul Keel, head of the EWall project, for giving me the opportunity to work with him for over two years, for providing constructive criticism in the design of the EWall software and algorithms, and for helping me revise this thesis.

I also thank all of my past teachers, without whose efforts I would not have had all the wonderful opportunities I have taken advantage of in college. In particular, I wish to thank Mrs. Rachel Egan for introducing me to Science Olympiad and Mr. Alan Gomez for developing my interest in Engineering.

Finally, I would like to thank my friends and family for providing support and encouragement to help me finish this thesis.

The EWall Project and this research is sponsored in part by the Office of Naval Research (ONR) Cognitive, Neural and Biomolecular Science and Technology Division Grant N000140210132.



# Contents

<b>1</b>	<b>Introduction</b>	<b>17</b>
1.1	Motivations . . . . .	18
1.2	Steps to Accurate Information Ordering . . . . .	20
1.3	Overview . . . . .	21
<b>2</b>	<b>Relationships</b>	<b>23</b>
2.1	Physical Relationships . . . . .	23
2.1.1	Horizontal Relationships . . . . .	26
2.1.2	Vertical Relationships . . . . .	27
2.1.3	Close Proximity Relationships . . . . .	29
2.2	Historical Relationships . . . . .	30
2.3	The Electronic Card Wall (EWall) Project . . . . .	30
<b>3</b>	<b>Order Producing Algorithm</b>	<b>33</b>
3.1	Relationship Matrix . . . . .	33
3.1.1	Relation Weights . . . . .	34
3.2	Order Production . . . . .	34
3.2.1	Decay Factor . . . . .	35
3.2.2	Order . . . . .	36
3.3	Score . . . . .	39
<b>4</b>	<b>Simulated Annealing Algorithm</b>	<b>43</b>
4.1	Algorithm Overview . . . . .	43

4.2	Changing Relation Weights . . . . .	45
4.3	Cooling Schedule . . . . .	46
4.4	Probability of Success . . . . .	46
4.5	Updating the Maximum Weight Change . . . . .	47
4.5.1	Alpha . . . . .	48
4.5.2	Beta and Average Successful Change . . . . .	48
<b>5</b>	<b>Results</b>	<b>49</b>
5.1	Expected Score of a Random Ordering . . . . .	49
5.2	Experiment Scenarios . . . . .	50
5.2.1	Experiment Scenario 1: Where to Eat Lunch? . . . . .	51
5.2.2	Experiment Scenario 2: Paul's Ordering . . . . .	51
5.2.3	Experiment Scenario 3: Thesis Organization . . . . .	51
5.2.4	Experiment Scenario 4: Baseball Standings . . . . .	51
5.3	Choice of Parameter Values . . . . .	56
5.3.1	Alpha . . . . .	56
5.3.2	Beta . . . . .	56
5.3.3	Decay Factor . . . . .	57
5.4	Relation Weights . . . . .	57
<b>6</b>	<b>Discussion</b>	<b>61</b>
6.1	Why Physical Positioning Analysis Works . . . . .	61
6.2	Why Simulated Annealing is Appropriate . . . . .	62
6.3	Application of Weights to Similar Arrangements . . . . .	64
<b>7</b>	<b>Contributions</b>	<b>65</b>
<b>A</b>	<b>Tables</b>	<b>67</b>

# List of Figures

2-1	A newly created card in the EWall software environment. . . . .	24
2-2	A sample arrangement of four cards: A, B, C and D. Each card can contain any text or be placed anywhere in the EWall work area. This arrangement simply represents one way to arrange four different cards.	25
2-3	Cards A and B in this figure are considered horizontally related to one another. . . . .	26
2-4	Cards A and B in this figure are still considered horizontally related to one another, even though card B is slightly offset from card A. This figure demonstrates the fudge factor built into the relationship finding algorithms. . . . .	26
2-5	Cards A and B in this figure are not considered horizontally related to one another because the difference in their vertical positioning has just exceeded the fudge factor built into the relationship finding algorithms.	27
2-6	Cards A and B in this figure are still considered vertically related to one another given the fudge factor built into the relationship finding algorithms. . . . .	28
2-7	Cards A and B in this figure are not considered vertically related to one another because the difference in their horizontal positioning has just exceeded the fudge factor built into the relationship finding algorithms.	28
2-8	A collection of tiles, sorted by shape. Each square is probably more likely related to every other square than to any of the circles or triangles. This type of relationship between objects is embodied by a Close Proximity relationship in the EWall software. . . . .	29

3-1	An example Relationship Matrix detailing the overall strength of relationships between any two cards. . . . .	35
3-2	An order production tree for the example Relationship Matrix in figure 3-1. Each path through the tree from the head node to a leaf represents a chain of cards one can walk to evaluate the leaf node's overall strength of relation to the target card (the head node). . . . .	36
3-3	An order production tree with decayed link values. . . . .	37
3-4	An order production tree with the values for one card computed at each level of the tree. . . . .	38
3-5	A target ordering of cards to be compared against an ordering created by the Order Producing Algorithm. . . . .	40
3-6	A sample ordering of cards to be compared against the target ordering in figure 3-5. Cards A and B are both displaced from their spots in the target ordering by one place, thus the score of this ordering is $1 + 1 = 2$ . . . . .	40
3-7	A sample ordering of cards to be compared against the target ordering in figure 3-5. Card A is displaced from its spot in the target ordering by 3 places, card B is displaced by 1 place and card D is displaced by 2 places, thus the score of this ordering is $3 + 1 + 2 = 6$ . . . . .	41
3-8	A sample ordering of cards to be compared against the target ordering in figure 3-5. All cards are in their correct places, thus the score of this ordering is 0, the best score possible. . . . .	41
3-9	A sample ordering of cards to be compared against the target ordering in figure 3-5. This ordering is exactly the reverse of the target ordering and produces the maximum score possible for an ordering of five cards: $4 + 2 + 0 + 2 + 4 = 4 * 2 + 2 * 2 + 0 = 12$ . As a quick proof that such an ordering produces the maximum score possible, note that switching any two (or more) cards will produce a score that is lower or equal to this score, thus this arrangement must have produced the maximum score. . . . .	41

5-1	This figure includes cards representing the thoughts you might have when you are trying to decide what to eat for lunch. The cards are grouped somewhat by topic. The numbers on the cards represent the order in which they were created. This order is used to determine what historical relations exist between the cards. . . . .	52
5-2	The arrangement of cards created by Paul Keel, the EWall project coordinator, when first attempting to use the software I created for this thesis. The numbers on the cards represent the order in which they were created. This order is used to determine what historical relations exist between the cards. . . . .	53
5-3	This arrangement of cards represents a first draft of my thesis organization, created long before I actually wrote my thesis. The numbers on the bottom part of the cards are the order in which they were created. This order is used to determine what historical relations exist between the cards. . . . .	54
5-4	This arrangement of cards represents the teams in the American League of Major League Baseball and the overall standings of that league. The numbers on the bottom part of the cards are the order in which they were created. This order is used to determine what historical relations exist between the cards. . . . .	55



# List of Tables

A.1 Number of cards in an ordering compared to the expected score of a random ordering of that size and the maximum score of an ordering of that size. . . . . 68

A.2 Experiment Scenario 1: Where to Eat Lunch? This table records the scenario results for the  $\alpha$  parameter. All results were produced with the following settings for the other parameters: *InitialTemperature* = 1000, *FinalTemperature* = 1, *TemperatureDecayFactor* = 0.95,  $\beta$  = 0.9, *DecayFactor* = 0.9, *depth* = maximum for each scenario. For each row of the table, the I ran the Simulated Annealing algorithm 20 times and recorded the minimum score for each run. The data in each row represents the mean and standard deviation of those minimum scores. I list the maximum score and expected score of a random arrangement (as calculated in section 5.1) for reference. . . . . 68

A.3 Experiment Scenario 2: Paul’s Ordering. This table records the scenario results for the  $\alpha$  parameter. The settings for the other parameters in this experiment are the same as those described in Table A.2 . . . 69

A.4 Experiment Scenario 3: Thesis Organization. This table records the scenario results for the  $\alpha$  parameter. The settings for the other parameters in this experiment are the same as those described in Table A.2 . . . . . 69

A.5	Experiment Scenario 4: Baseball Standings. This table records the scenario results for the $\alpha$ parameter. The settings for the other parameters in this experiment are the same as those described in Table A.2 . . . . .	69
A.6	Experiment Scenario 1: Where to Eat Lunch? This table records the scenario results for the $\beta$ parameter. All results were produced with the following settings for the other parameters: <i>InitialTemperature</i> = 1000, <i>FinalTemperature</i> = 1, <i>TemperatureDecayFactor</i> = 0.95, $\alpha$ = 0.9, <i>DecayFactor</i> = 0.9, <i>depth</i> = maximum for each scenario. For each row of the table, the I ran the Simulated Annealing algorithm 20 times and recorded the minimum score for each run. The data in each row represents the mean and standard deviation of those minimum scores. I list the maximum score and expected score of a random arrangement (as calculated in section 5.1) for reference. . . . .	70
A.7	Experiment Scenario 2: Paul’s Ordering. This table records the scenario results for the $\beta$ parameter. The settings for the other parameters in this experiment are the same as those described in Table A.6 . . .	70
A.8	Experiment Scenario 3: Thesis Organization. This table records the scenario results for the $\beta$ parameter. The settings for the other parameters in this experiment are the same as those described in Table A.6 . . . . .	71
A.9	Experiment Scenario 4: Baseball Standings. This table records the scenario results for the $\beta$ parameter. The settings for the other parameters in this experiment are the same as those described in Table A.6 . . . . .	71

A.10 Experiment Scenario 1: Where to Eat Lunch? This table records the scenario results for the $\gamma$ decay factor parameter. All results were produced with the following settings for the other parameters: <i>InitialTemp</i> = 1000, <i>FinalTemp</i> = 1, <i>TempDecayFactor</i> = 0.95, $\alpha = 0.9$ , $\beta = 0.9$ , <i>depth</i> = maximum for each scenario. For each row of the table, the I ran the Simulated Annealing algorithm 20 times and recorded the minimum score for each run. The data in each row represents the mean and standard deviation of those minimum scores. I list the maximum score and expected score of a random arrangement (as calculated in section 5.1) for reference. . . . .	72
A.11 Experiment Scenario 2: Paul’s Ordering. This table records the scenario results for the $\gamma$ decay factor parameter. The settings for the other parameters in this experiment are the same as those described in Table A.10 . . . . .	72
A.12 Experiment Scenario 3: Thesis Organization. This table records the scenario results for the $\gamma$ decay factor parameter. The settings for the other parameters in this experiment are the same as those described in Table A.10 . . . . .	73
A.13 Experiment Scenario 4: Baseball Standings. This table records the scenario results for the $\gamma$ decay factor parameter. The settings for the other parameters in this experiment are the same as those described in Table A.10 . . . . .	73
A.14 This table presents an overall summary of the experiment results for all four scenarios. Average score and standard deviation reported for a sample size of 100 trials with the parameter values detailed in section 5.4. . . . .	74



# Chapter 1

## Introduction

Building systems that can identify, sort and present related pieces of information can potentially save us vast amounts of time and money. An important concept in building such systems is the idea that humans' ability to physically arrange information greatly aids them in categorizing and understanding that information. For example, suppose I give you an assortment of colored tiles, each in the shape of a circle, square or triangle. If I ask you to tell me how many red squares there are, you would probably group all the red tiles together, then group the squares from those red tiles, and finally, count the red squares. This task would obviously be much harder if you were not allowed to first rearrange the tiles. If a computer could look at the final arrangement of tiles, ideally it would be able to tell which tiles are related to one other (the red tiles, the red squares, etc.).

In this thesis, I describe a system that achieves the goal of being able to look at an arrangement of information and determine which pieces are most likely related to each other. This determination is accomplished in a two step process. The first step is to identify relationships between physically arranged objects, such as any horizontal or vertical alignments. The second is to use those relationships in learning to order the objects by relevance to one particular object (e.g. given one red square, find all the others like it). After analyzing the results of my system, I demonstrate that:

- Relationships identified between physically arranged pieces of information are

enough to generate orderings of relevance between those pieces of information. The generated orderings of relevance are close to what a human would list for the same pieces of information and are up to 70 percent more accurate than randomly generated orderings.

- A Simulated Annealing algorithm can be used to learn which types of relationships are most important in each arrangement of information. The relative weights placed on each type of relationship can then be applied to similar arrangements of information in the future to produce accurate orderings of that information.
- The formulas used to generate orderings of pieces of information only need to operate on lists of relationships between those pieces of information. These formulas are structured in such a way that allows them to be evaluated extremely quickly, a necessity for algorithms that complete many iterations before reaching a solution, like Simulated Annealing algorithms or Genetic algorithms.
- The formulas used to score generated orderings are simple to evaluate and can be applied to different orderings, regardless of the type of algorithm used to generate those orderings.

## 1.1 Motivations

David Kirsh’s work [2] on complementary strategies shows us that humans use physical arrangements to make sense of information and ease their cognitive load when performing tasks with this information. Kirsh defines a complementary strategy as an organizing activity (such as pointing, arranging the position and orientation of nearby objects, writing things down, etc.) that reduces cognitive loads by recruiting external elements. Kirsh’s experiments [2] describe counting coins under two conditions, without using hands to manipulate the coins, and with the use of hands. Not surprisingly, people were slower at computing the sum of the coins and made more errors in the “no hands” condition than in the “hands allowed” condition. Kirsh

also observed such organizing strategies as people grouping coins by denomination and pointing to coins while computing the sum. The ideas of using complementary strategies fueled the beginnings of the Electronic Card Wall (EWall) project [4] and have provided a base for the development of basic physical relations to look for among objects, such as groupings and a measure of proximity to other objects.

The EWall project is based on the work of William Peña and Steven Parshall. From their book, *Problem Seeking: an Architectural Programming Primer* [5], the EWall project obtained its outline for several guiding principles. One such guiding principle is the importance of the ability to easily record and rearrange information. The EWall project chose to rely on the use of portable, electronic “Post-It” notes to allow users to record and rearrange information. Peña and Parshall’s work also stresses the importance of vertical alignment of related information to allow humans to easily categorize it. The EWall project and this thesis extend this work and use other spatial relations (such as horizontal alignment) to better categorize and relate pieces of information. I will discuss the EWall project in greater detail in section 2.3.

A fair amount of research has been performed on how humans construct and perceive relations among spatially arranged objects. Max Wertheimer’s work, detailed in his book *Productive Thinking* [7], provides evidence that the recognition of relations between parts and wholes helps humans process spatial arrangements. Wertheimer’s research on perceptual organization [8] provides valuable insights into how humans perceive relations among spatially arranged objects by suggesting a set of principles for detecting object relations based on concepts such as size, proximity, similarity and continuation. This thesis takes into account several of these principles, such as size and proximity, when developing its physical arrangement analysis algorithms. From these bodies of work and many others, it is clear that if we are to create a system for analyzing how pieces of information are related, we need to explore the physical relationships between those pieces whenever possible.

## 1.2 Steps to Accurate Information Ordering

To create a system that accurately orders pieces of information by relevance to one target piece, we need first to analyze the physical arrangement of that information. In this analysis, we will identify any physical relationships that exist between the pieces of information, such as pieces that are aligned horizontally or vertically, or pieces that are close together. In addition to physical relationships, we will also look for historical links between pieces of information, such as pieces that are created close together in time.

Once the search for relationships between the pieces of information is complete, we will use these relationships to construct a matrix of how strongly each pair of pieces is related. Since certain types of relations may be more pertinent to determining which pieces of information are actually related, we will scale each relation by a certain weight before we construct this “relationship strength” matrix. After construction of the matrix, we will be ready to produce an ordering of the information in relation to one target piece.

To produce the ordering of information, we will use our relationship strength matrix and a method of taking into consideration transitive links between pieces of information<sup>1</sup>. Once we have created an ordering relative to one target piece, we need to evaluate this ordering for accuracy against a known ordering. Finally, after evaluating the ordering our algorithm produced, we will slightly tweak the weights on the types of relationships to produce a better ordering and then iterate this weight-tweaking/evaluation process. In this way, we can produce a highly accurate ordering of the pieces of information in relation to the target piece and can deduce which types of relationships are most important for this particular physical arrangement of information.

---

<sup>1</sup>For example, if piece A is strongly related to piece B, which is strongly related to piece C, we can deduce that piece C is fairly strongly related to piece A.

## 1.3 Overview

Chapter 2 details the types of relationships we will look for in analyzing a physical arrangement of information and covers this thesis' involvement with the EWall project. Chapter 3 explains the Relationship Matrix and the Order Producing algorithm. The Order Producing algorithm creates an ordering of pieces of information in relation to one target piece. Chapter 4 describes the scoring metric and Simulated Annealing algorithm that I used to deduce the best arrangement of information. The Simulated Annealing algorithm works by iteratively changing the relation weights and re-running the Order Producing algorithm from chapter 3. I report the results of my algorithms in chapter 5 and discuss these results in chapter 6. Finally, I list the contributions of this thesis in chapter 7.



# Chapter 2

## Relationships

As described in section 1.1, analyzing physical and historical relationships between pieces of information can give us a good starting point in attempting to produce an accurate ordering of that information. This chapter discusses the physical and temporal relationships I used in my Order Producing algorithm, including horizontal, vertical, close proximity and historical relationships. After introducing the various types of relationships, I detail how this thesis fits within and complements the EWall project.

### 2.1 Physical Relationships

The EWall project, which I will discuss further in section 2.3, uses digital “Post-It” notes (hereafter known as “information cards,” or simply “cards”) as its means of representing distinct pieces of information. Figure 2-1 is an example of a newly created card in the EWall software.

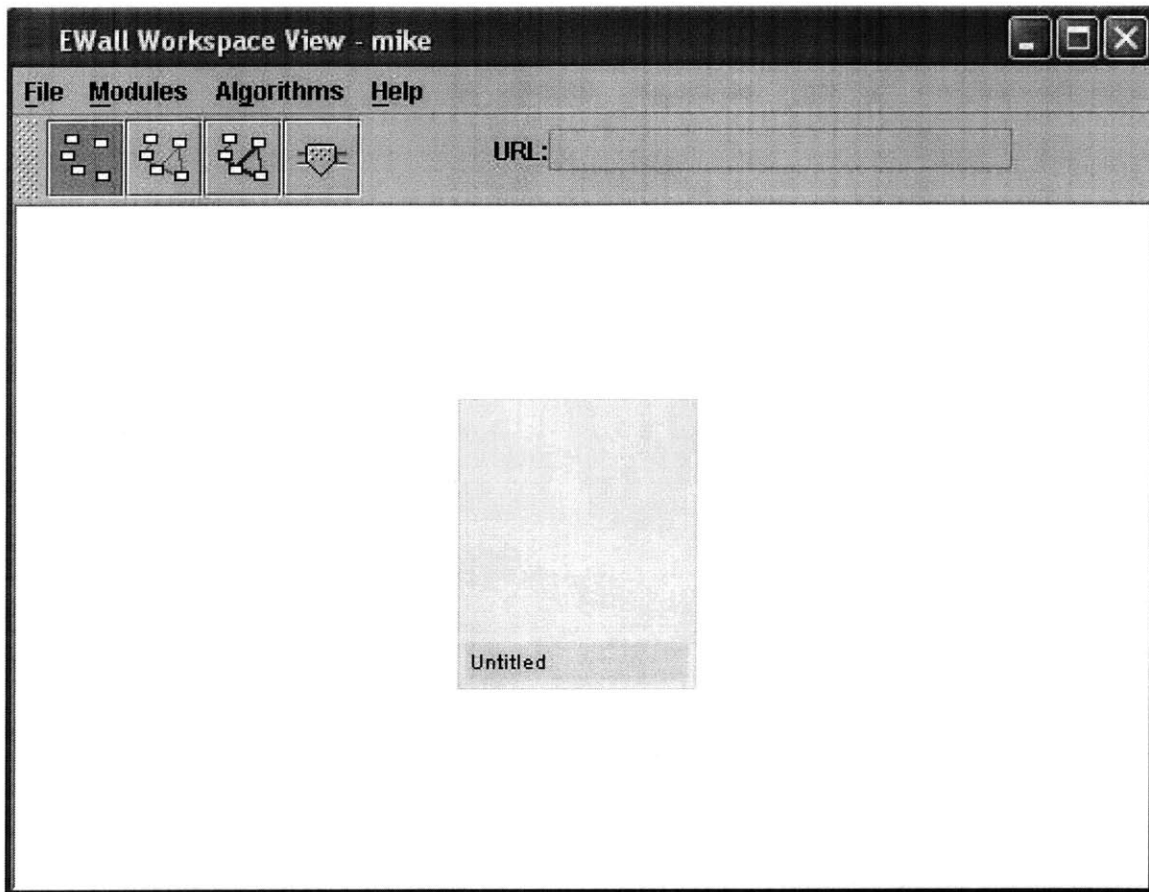


Figure 2-1: A newly created card in the EWall software environment.

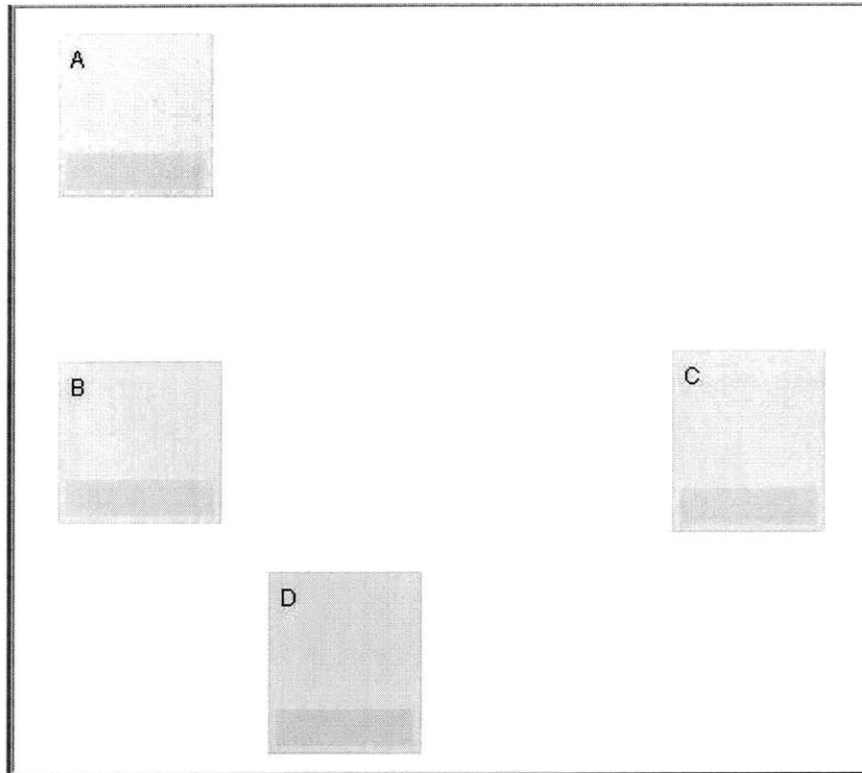


Figure 2-2: A sample arrangement of four cards: A, B, C and D. Each card can contain any text or be placed anywhere in the EWall work area. This arrangement simply represents one way to arrange four different cards.

In any arrangement of pieces of information, physical relationships of some kind exist between those pieces. For example, figure 2-2 shows a sample arrangement of four cards: A, B, C and D. Card A is above card B. Card B, in turn, is to the left of card C and close to a card D.

There are potentially an infinite number of physical relationships that could be exploited in attempting to figure out how cards are related to one another. For simplicity and efficiency, I chose to focus on card context rather than content. To this end, I explored the use of basic physical relationships, such as horizontal and vertical alignment and close proximity.

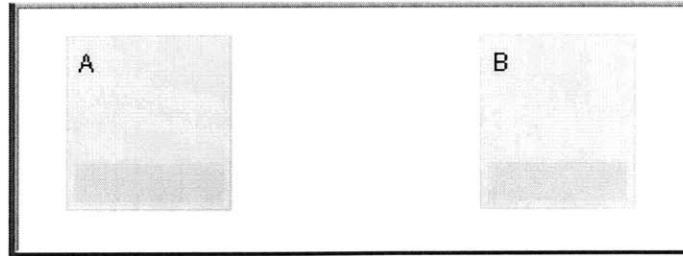


Figure 2-3: Cards A and B in this figure are considered horizontally related to one another.

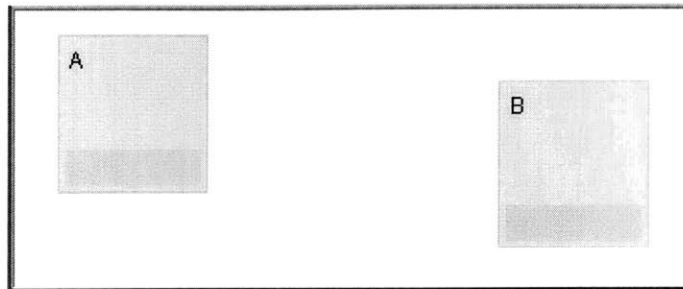


Figure 2-4: Cards A and B in this figure are still considered horizontally related to one another, even though card B is slightly offset from card A. This figure demonstrates the fudge factor built into the relationship finding algorithms.

### 2.1.1 Horizontal Relationships

In the EWall software, a horizontal relationship exists between two cards when they are roughly aligned in a horizontal row. For example, the EWall software would consider cards A and B in figure 2-3 to be horizontally related.

Because humans do not perfectly align cards when working with the EWall software, the software includes a tolerance of about half the average height of the cards in its search for horizontal relationships. With this tolerance, the software would consider the cards in figure 2-4 to be horizontally related, but not the cards in figure 2-5.

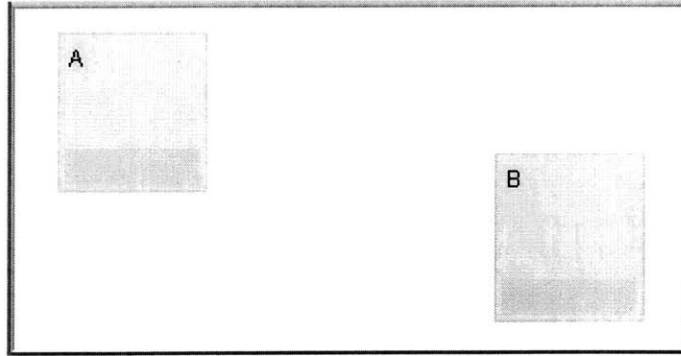


Figure 2-5: Cards A and B in this figure are not considered horizontally related to one another because the difference in their vertical positioning has just exceeded the fudge factor built into the relationship finding algorithms.

### 2.1.2 Vertical Relationships

A vertical relationship exists between two cards when they are roughly aligned in a vertical column. As with horizontal relations, the EWall software has a similar tolerance for cards not in exact vertical alignment. Thus the EWall software would consider cards A and B in figure 2-6 to be vertically related, but not those cards in figure 2-7.

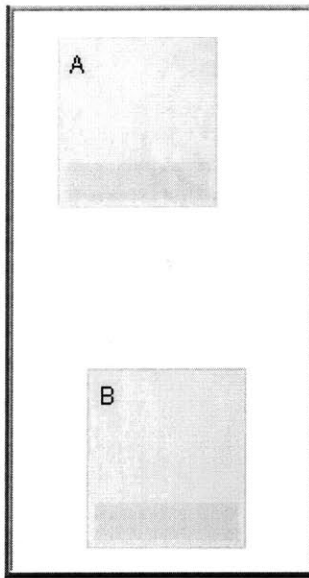


Figure 2-6: Cards A and B in this figure are still considered vertically related to one another given the fudge factor built into the relationship finding algorithms.

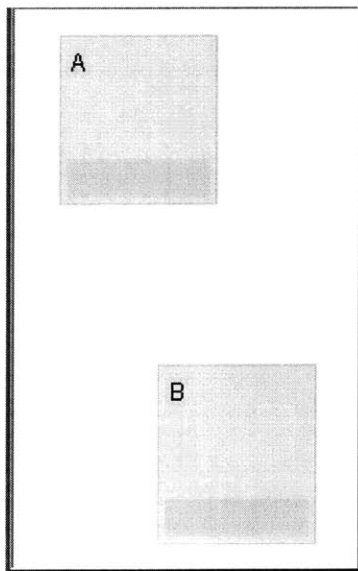


Figure 2-7: Cards A and B in this figure are not considered vertically related to one another because the difference in their horizontal positioning has just exceeded the fudge factor built into the relationship finding algorithms.

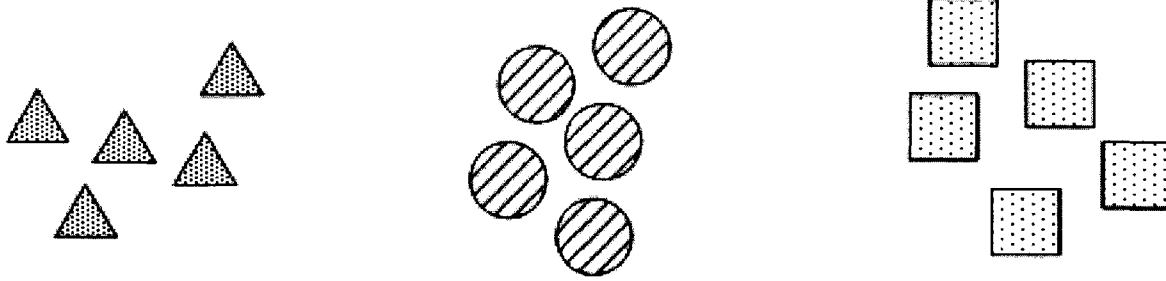


Figure 2-8: A collection of tiles, sorted by shape. Each square is probably more likely related to every other square than to any of the circles or triangles. This type of relationship between objects is embodied by a Close Proximity relationship in the EWall software.

### 2.1.3 Close Proximity Relationships

In addition to horizontal and vertical relationships, another type of relationship I chose to consider is how close together cards are. Logically, we tend to put items that are more related closer together, and those that are less related farther apart. For example, if you were presented with a collection of tiles, each a circle, square or triangle, you would probably group the tiles together by their shape, as in figure 2-8. In this arrangement of the tiles, each square is relatively closer to the other squares than any of the other tiles, thus each square is probably more likely related to every other square than to any of the other tiles. In this case, each pair of squares would have a Close Proximity relationship between them.

#### Base Distance and the Close Proximity Relationship Test

In the EWall software environment, close proximity relationships are calculated by first computing a “base distance.” The base distance is computed in the following manner:

1. Compute the sum of the distances between the centers of each pair of cards.
2. Divide the sum by the number of distinct pairs of cards, thereby computing the average distance between any two cards.
3. Finally, divide the result by a scaling factor of 1.5.

We included division by the scaling factor in the last step because we want our base distance to be smaller than the average distance between any two cards.

Once you have computed the base distance, test each pair of cards against that distance. Pairs of cards whose centers are closer together than the base distance are considered to have a Close Proximity relationship between them.

## **2.2 Historical Relationships**

Besides looking for physical relationships in an arrangement of cards, we can also look for historical relationships between those cards. If we know when the cards of an arrangement were created (or in what order), we can use this information to create relations between pairs of cards created in historical order. In my thesis, I assign a Historical Relationship to a pair of cards created one after the other. Such a relationship between cards embodies the idea that cards created one after another are more likely related than those whose creation is separated by large amount of time or by the creation of several other cards. In this manner, Historical Relationships should prove a valuable tool for analyzing situations where information is created in logical order, such as when jotting down someone's name, address and phone number.

## **2.3 The Electronic Card Wall (EWall) Project**

This thesis was developed in conjunction with the EWall Electronic Card Wall (EWall) Project[4], which is supervised by William Porter and Patrick Winston at M.I.T. The EWall project was undertaken to research improving collaboration strategies between humans in a work setting. The EWall Project's goal is to help problem solvers work faster and produce smarter solutions by easily collecting, organizing and viewing graphical and contextual information. To this extent, my thesis is an integral part of the EWall project, as the ideas presented in this thesis are directly applicable to EWall's goals of showing users relevant information in a timely manner.

The EWall software allows users to create digital cards that can be arranged and

shared in a collaborative work environment. To achieve the goal of helping groups collaborate in a more efficient manner, the EWall software employs several different types of algorithms to analyze information content and arrangement. The class of algorithms relevant to this thesis involve the physical arrangement of cards. The EWall software analyzes the placement of each card and the order in which it was produced to create relationships between the cards. Thus, the relationships presented in this chapter are all implemented with EWall algorithms I wrote over the course of my work on the project.

In a collaborative setting, the EWall software strives to show each user cards from other users that are relevant to those he or she is currently working on. Being able to order cards by their relevance to one card is extremely important when attempting to prioritize cards for users. For example, say five users are collaborating on the same project. Ideally, the EWall software of user A would show him or her only those cards from the rest of the users that are relevant to the cards A is working with. To accomplish this feat, the software first tries to produce an ordering of everyone else's cards with respect to one card from user A, then looks at many such orderings for each of the cards A is working with. The next chapter will detail how I produce orderings of cards based on relevance to a target card.



# Chapter 3

## Order Producing Algorithm

This chapter covers the heart of my thesis: how the software I wrote produces an ordering of cards by their relevance to one target card. In designing this software, I sought to choose processes that roughly mimic human information grouping and argument analysis tactics. In section 3.1, I discuss the construction of the Relationship Matrix, which measures the aggregate strength of relationship between any two cards. During the generation of this matrix, I apply weights to each type of relationships to account for the fact that different people arrange the same information in different ways. In section 3.2, I explain how I use the Relationship Matrix to order cards relative to the target card. Finally, in section 3.3 I discuss how a generated ordering is objectively scored against a target ordering, an idea that will be useful when evaluating whether a given change in the relation weights produces a better ordering.

### 3.1 Relationship Matrix

The software first analyzes a user's arrangement of cards to find relationships between those cards, as previously described in chapter 2. Once all the relationships have been identified, a "Relationship Matrix" is formed. The Relationship Matrix represents the overall strength of relationship between any two cards. Since one type of relation may be more relevant than another for a particular arrangement of information, each type

of relation has a weight associated with it. To produce the overall strength of the relationship between a pair of cards, we multiply the number of each type of relation between those cards by the weight associated with that type of relation. For example, if there is one horizontal relation and one close proximity relation between two cards (say, cards 1 and 2), and the weights of horizontal relations and close proximity relations are 3 and 4 respectively, then the (1,2) entry in the relationship matrix will be  $(1 * 3) + (1 * 4) = 7$ . Since the types of relationships I chose to focus on in this thesis were bidirectional in nature, each entry in the relationship matrix has the same value as the corresponding entry across the diagonal (i.e. the (1,2) and (2,1) entries have the same value).

### 3.1.1 Relation Weights

The “relation weight” associated with each relation represents the relative importance of that type of relation. Since each arrangement of cards carries a different context, it makes sense that each arrangement would have different relation weights for the various types of relations. These weights are adjusted by the Simulated Annealing algorithm, as discussed later in chapter 4. By adjusting the relative weights of the relations, the Simulated Annealing algorithm iteratively produces different ordering outcomes. I will now discuss how the Order Production algorithm produces an ordering of the cards relative to a target card.

## 3.2 Order Production

Armed with the Relationship Matrix, a data structure representing the total strength of the relationship between any pair of cards, the Order Production algorithm produces an ordering of the cards with respect to one target card. The high-level view of how this is accomplished is fairly straight forward and I will illustrate it now in the following example. After analyzing a hypothetical user’s workspace for relationships between cards, the relationship matrix in figure 3-1 is produced in the manner described previously in section 3.1.

	A	B	C	D	E	F
1		Card 1	Card 2	Card 3	Card 4	Card 5
2	Card 1	-	0	2	1	0
3	Card 2	0	-	3	1	2
4	Card 3	2	3	-	1	2
5	Card 4	1	1	1	-	0
6	Card 5	0	2	2	0	-

Figure 3-1: An example Relationship Matrix detailing the overall strength of relationships between any two cards.

In this example, the target card is card 1. One might initially think that the cards most related to the target card would be those with the highest score in card 1's row of the Relationship Matrix, because this row represents each card's direct relationship with card 1. Ranking cards by this method would produce a somewhat biased result because only the direct links to the target card are being considered. For example, according to card 1's row of the Relationship Matrix in figure 3-1 card 2 is not related at all to our target card. Notice, however, that card 2 is highly related to card 3, and card 3 is fairly strongly related to our target card. Simply arranging the cards by only looking at card 1's row of the Relationship Matrix would completely ignore the fact that card 2 is probably indirectly related to our target card through card 3.

To take a more in-depth look at a card's relationship to the target card, we need to explore each card's relationship to every other card. This type of analysis can be envisioned in the tree-like structure of figure 3-2. Each successive layer of the tree represents a connection to the target card through a chain of cards that is getting progressively longer. Logically, when considering the value of a card in relation to the target card, a link between two cards farther away from the target card should be discounted when compared to a direct link to the target card. This idea is captured in the decay factor concept.

### 3.2.1 Decay Factor

As one progresses down the tree structure in figure 3-2, the chain from each node to the target card gets progressively longer. The initial value of each link in this chain



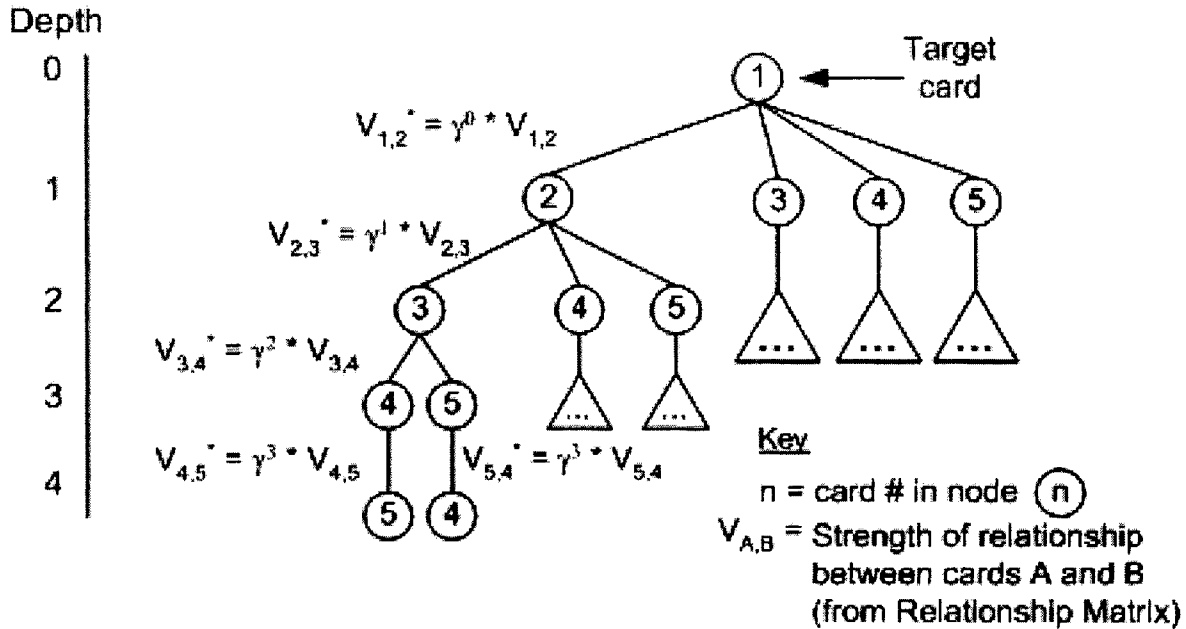


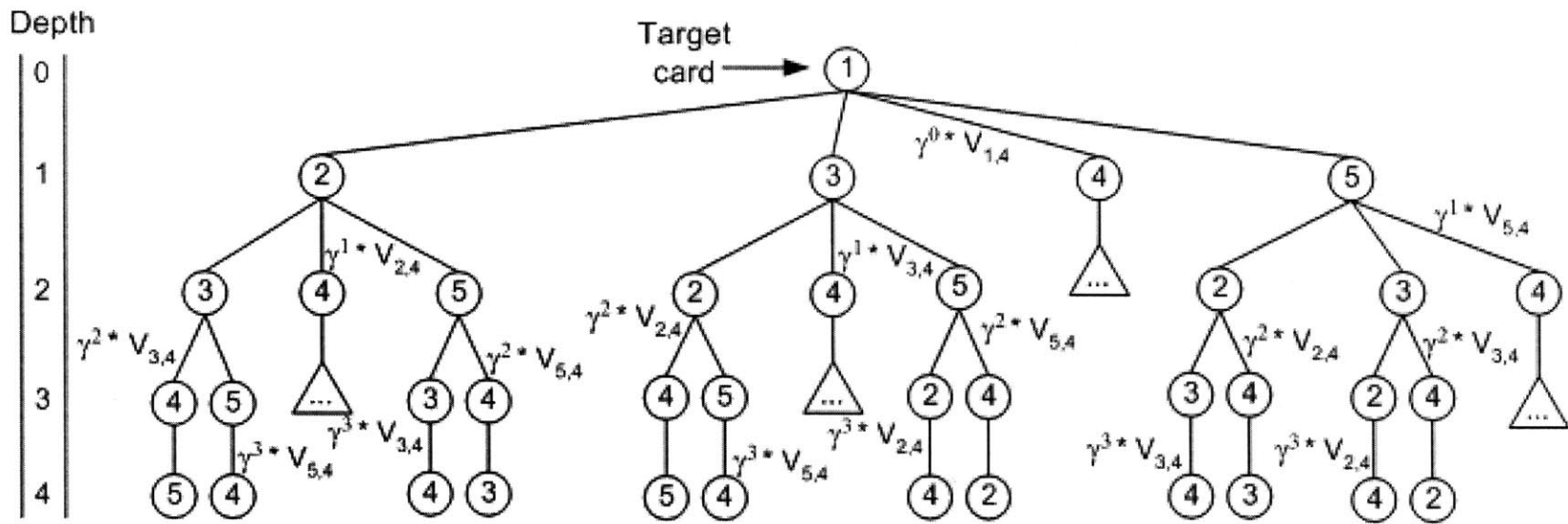
Figure 3-3: An order production tree with decayed link values.

shows all possible paths of relations between the target card and the rest of the cards. Each node in the tree represents a card that is a certain distance away from the target card in a particular chain of related cards. Since each card appears in multiple nodes on each level within the tree, we need to sum the value of all a card's nodes on one level to find the card's value for that level. Conceptually this tactic is taking into account all possible paths from that card to the target card for a path of a given depth.

The value of a card for a particular depth of the tree is the value of the card on the level above plus the discounted links connecting that card's nodes back up through the tree. For simplicity, we will set the value of a card at depth zero to be zero. Figure 3-4 illustrates the value calculation for card 4 at each level of the tree.

Ideally, we want to calculate the value of each card at the lowest level of the tree because that level encapsulates all the information of the tree (if you calculate the value of a card at level two, you will only be taking into account chains of length two, whereas if you calculate the value of a card at the lowest level, you will be taking into account chains of the longest possible length). From figure 3-4, you'll notice a pattern

Figure 3-4: An order production tree with the values for one card computed at each level of the tree.



Card 4 values at each level:

$$T_{4,0} = 0$$

$$T_{4,1} = T_{4,0} + \gamma^0 * V_{1,4}$$

$$T_{4,2} = T_{4,1} + \gamma^1 * (V_{2,4} + V_{3,4} + V_{5,4})$$

$$T_{4,3} = T_{4,2} + \gamma^2 * 2(V_{2,4} + V_{3,4} + V_{5,4})$$

$$T_{4,4} = T_{4,3} + \gamma^3 * 2(V_{2,4} + V_{3,4} + V_{5,4})$$

Key

$n$  = card # in node  $\textcircled{n}$

$T_{A,B}$  = Total value of card A at level B

$V_{A,B}$  = Strength of relationship between cards A and B (from Relationship Matrix)

developing in the formula for the total value of a card at a given level of the tree. All the Relation Matrix terms (the ones labeled  $V_{A,B}$ , where  $A$  and  $B$  are the cards we are finding the relation strength between) can be grouped together. The developing pattern in figure 3-4 can be condensed into the following formula for evaluating the value of any card at any depth of the tree:

$$T_{A,d} = V_{target,A} + \sum_{i=1}^d \left\{ \gamma^{i-1} * \frac{(n-3)!}{(n-i-1)!} \right\} * \sum_{j=1}^n \{V_{j,A} | j \neq A, target\}$$

Compared to the alternative of performing the necessary calculations on each level of the tree, this formula can be evaluated extremely quickly because it saves the work of summing the  $V_{a,b}$  terms over and over again. This fact allows us to find the value of each card and produce the ordering in a minimal time.

### 3.3 Score

An integral part of evaluating how well the Order Producing algorithm is working is the ability to score the order produced against a target order. The Order Producing algorithm creates an order with respect to the target card it is given. We can analyze how accurate this order is if we are also given a target order to compare it against. At first glance, being given a target order ahead of time seems like it defeats the purpose of the Order Producing algorithm (if you have the answer to a problem, why do you need to bother solving it in the first place?). However, once the Order Producing algorithm has been trained against known situations, it can then be used on unknown situations with confidence that the results it produces are viable.

Given the Order Producing algorithm's output and a target ordering, we can compute the score of the algorithm's output in the following manner. We want the score to penalize the produced ordering for any cards that are out of place, assessing more of a penalty for cards further out of place than cards only displaced by a small amount. We can accomplish this goal simply by measuring the displacement of each card from its position in the target ordering and summing these displacements. To

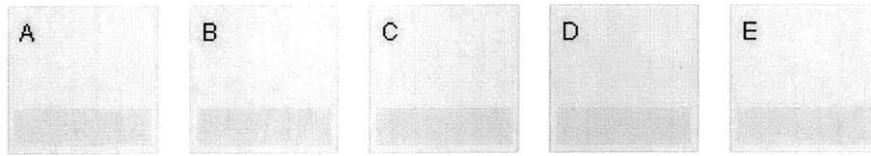


Figure 3-5: A target ordering of cards to be compared against an ordering created by the Order Producing Algorithm.

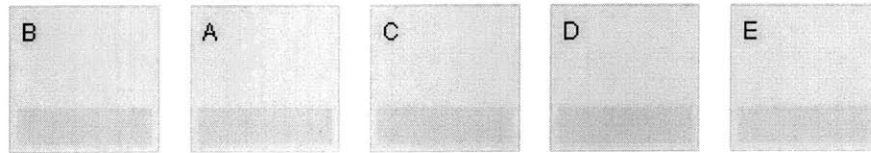


Figure 3-6: A sample ordering of cards to be compared against the target ordering in figure 3-5. Cards A and B are both displaced from their spots in the target ordering by one place, thus the score of this ordering is  $1 + 1 = 2$ .

illustrate this scoring method, figure 3-5 represents a sample target ordering and figures 3-6 and 3-7 represent orderings produced by the Order Producing algorithm. In this document, all linear orderings are presented in a left-to-right arrangement (i.e. the card most relevant to the target card will appear first, on the left, while the card least relevant will appear last, on the right).

In figure 3-6, cards A and B are switched while the rest of the ordering is correct. Card A is displaced from its spot in the target ordering by 1 place and card B is similarly displaced by 1 place, so the score for the order in figure 3-6 is  $(1 + 1 = 2)$ . In figure 3-7, card A is displaced by 3 places, card B is displaced by 1 place and card D is displaced by 2 places, thus the score for this order is  $(3 + 1 + 2 = 6)$ . The ordering in figure 3-8 is the same as the target ordering, thus its score is 0, the best possible score. The ordering in figure 3-9 is exactly the reverse of the target ordering and consequently, its score is the maximum possible. As a quick proof that such an ordering produces the maximum score possible, note that switching any two (or more) cards will produce a score that is lower or equal to this score, thus this arrangement must have produced the maximum score.

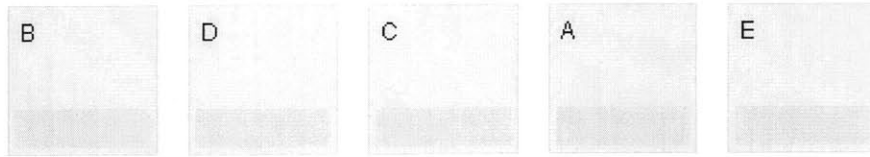


Figure 3-7: A sample ordering of cards to be compared against the target ordering in figure 3-5. Card A is displaced from its spot in the target ordering by 3 places, card B is displaced by 1 place and card D is displaced by 2 places, thus the score of this ordering is  $3 + 1 + 2 = 6$ .

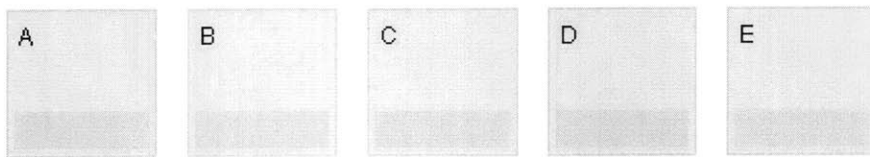


Figure 3-8: A sample ordering of cards to be compared against the target ordering in figure 3-5. All cards are in their correct places, thus the score of this ordering is 0, the best score possible.

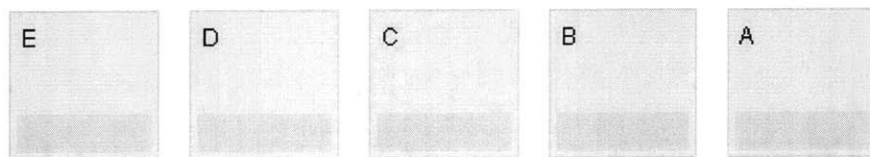


Figure 3-9: A sample ordering of cards to be compared against the target ordering in figure 3-5. This ordering is exactly the reverse of the target ordering and produces the maximum score possible for an ordering of five cards:  $4+2+0+2+4 = 4*2+2*2+0 = 12$ . As a quick proof that such an ordering produces the maximum score possible, note that switching any two (or more) cards will produce a score that is lower or equal to this score, thus this arrangement must have produced the maximum score.



# Chapter 4

## Simulated Annealing Algorithm

Chapter 3 discussed how we produce an ordering of cards relative to a target card given the relationships between the all cards. Section 3.1 discussed how to produce the Relationship Matrix used in the Order Production algorithm and how we assigned each different type of relationship a weight. Chapter 3 also demonstrated how we can score an ordering of cards against the target ordering. This chapter will combine these concepts to show how we can adjust the relationship weights to produce better orderings (i.e. ones with scores as close to 0 as possible). Section 4.1 overviews the Simulated Annealing algorithm I use to produce progressively better orderings. In section 4.2, I detail how the algorithm changes the weights of the different types of relations. In sections 4.3 and 4.4, I discuss the cooling schedule and probability of accepting a change in the relation weights that leads to a worse ordering.

### 4.1 Algorithm Overview

Our goal is to get the Order Production algorithm to produce the best ordering possible relative to a target card. When we create our ordering of cards using the Order Production algorithm, there are several parameters we can tweak to affect the output of the algorithm: the relationship weights, the decay factor and the depth of the evaluation tree. Each time we change one of these parameters, we need to recalculate the relationship matrix and re-run the Order Production algorithm to

produce the new ordering. The new ordering may or may not be better than the previous ordering.

We could run a simple hill-climbing algorithm to try to produce the best possible ordering, as described in Russell and Norvig’s book, *Artificial Intelligence: A Modern Approach* [6]. Such an algorithm would only accept those parameter changes that produce a better ordering from one iteration to the next. Visually, this algorithm would “climb the hill” of the parameter feature space until making any parameter change only produces a worse ordering. For example, if you had two parameters,  $x$  and  $y$ , and you graphed the results of changes to those  $x$  and  $y$  parameters on the  $z$  axis, this algorithm would look for values of  $x$  and  $y$  that produced the best possible  $z$ .<sup>1</sup> The downfall of such a hill-climbing algorithm is that it tends to get stuck in local maxima. Unfortunately, we have a non-trivial number of parameters to adjust on each iteration of the Order Production algorithm, thus there exists a fairly good potential for many local maxima in our results.

Instead of always accepting only those changes that produce better outcomes of the Order Production algorithm, we will use a modified version of the Simulated Annealing algorithm in Russell and Norvig’s book [6]. The point of using a Simulated Annealing algorithm is that it occasionally accepts changes that produce worse outcomes in hopes of eventually finding the globally optimal solution. The Simulated Annealing algorithm works like a glass maker slowly cooling (annealing) glass to reduce its brittleness. Initially, atoms of glass are very hot and have a lot of kinetic energy. As the glass cools and loses energy, it forms a more uniform, crystalline structure, eventually creating a stronger product than glass that has been cooled quickly.

In our algorithm, a variable representing temperature will indirectly control the likelihood of accepting a change in parameter settings that produces a worse outcome from one iteration to the next. In the beginning, when the temperature is high, an unfavorable change will likely be accepted, but as the temperature cools, the likelihood

---

<sup>1</sup>Since score is our measure of success of the Order Production algorithm and the better scores are low (not high), the  $z$  axis would have to plot something like  $\frac{1}{score}$  for the term “hill-climbing” to make sense in a strictly literal interpretation.

of accepting an unfavorable change will grow smaller and smaller. In this way, we can hopefully bounce out of local maxima early on and land on the peak containing the global maxima.

Another feedback system will control the maximum amount that we can change the parameters with each iteration. As the temperature cools, the allowable changes decrease in size, thus the algorithm attempts to “hone-in” on a solution and tolerates less drastic jumps around the solution space. In this way, the Simulated Annealing algorithm gives us the combination of decreasing change acceptance and shrinking parameter size movement that produces an efficient way of searching the solution space for the optimal parameter settings.

## 4.2 Changing Relation Weights

In my implementation of the Simulated Annealing algorithm, I chose to focus on only optimizing the relation weights and not the depth and decay factor. This led to a simpler and faster implementation of the algorithm and kept the focus on the relation weights, which are the most relevant part of this thesis from a cognitive science standpoint. I chose to limit the weights to a range of  $[-10, 10]$ , inclusive, and allowed them to take the value of any real number in this range. As explained in section 3.1.1, the weights on each type of relation represent the relative importance of that type of relation in comparison to the other types of relations. Changes in the relation weights affect the Relationship Matrix and make some cards more highly related to the target card than others.

Each iteration of the Simulated Annealing algorithm begins with adjusting the relation weights by a random amount between the allowable range of new values. As the temperature cools, this allowable range of new values shrinks. In practice, I implemented this by multiplying a random real number between  $[-1, 1]$ , inclusive, by the maximum change for each weight. Initially, the maximum change for each weight was 10. I will discuss how I adjusted the maximum change for each weight in section 4.5.

## 4.3 Cooling Schedule

The cooling schedule for the temperature parameter is a crucial part of the Simulated Annealing algorithm. Many different cooling schedule exist, but two of the more popular ones are linear and exponential decay. In the linear schedule, the temperature is reduced by a set amount on each iteration (e.g.  $Temp = Temp - 20$ ). In the exponential decay schedule, the cooling is weighted toward the early iterations. In such a schedule, the temperature drops quickly in the beginning, then much more slowly as time progresses (e.g.  $Temp = Temp * 0.9$ ). After using both schedules for several tests and comparing the results, I selected the exponential decay schedule because it produced the effect of honing-in on the solution faster by accepting a comparatively smaller percentage of undesirable parameter changes in the early stages of the algorithm. Typically, I would start the algorithm at a temperature of 1000 degrees and run it to a final temperature of 1 degree with an exponential cooling decay factor of 0.95.

## 4.4 Probability of Success

A change in the parameters yielding a new ordering will automatically be accepted if that new ordering is better than the previous best ordering (conceptually, moving up a hill is always good). Conversely, a change in parameters that produces a worse ordering will be accepted with a probability that diminishes over time. I calculated this probability of successfully accepting a change in the parameters for the second case in the following manner. First, the probability should be comparatively larger when the change produced only a slightly unfavorable outcome than when it produced a much worse outcome. Second, the probability should decrease as time went on (i.e. as the temperature decreased). To accomplish these goals, I calculated the probability using the following formula.

$$P(Success) = e^{\{-c * \frac{Score - BestScoreSoFar}{AverageScore} * \frac{CurrentStep}{TotalSteps}\}}$$

$(Score - BestScoreSoFar)$  represents the difference between the score that the current parameter settings produced and the best score achieved to that point.  $AverageScore$  represents the average score of all accepted changes to that point.  $CurrentStep$  represents the current iteration of the algorithm and  $TotalSteps$  represents the total number of steps the algorithm will run (which can be computed using the initial temperature, final temperature and the cooling schedule). The constant  $c$  simply represents a tweak to compensate for the  $CurrentStep$  being exceedingly small in the first few iterations of the algorithm. A value of  $c = 4$  was usually sufficient to produce desirable probabilities of success over the lifetime of the algorithm. Notice how this function achieves the two desired goals stated previously. When the change in score is small, the probability of success is larger than when the change in score is large. In addition, as time goes on and  $CurrentStep$  grows larger, the probability of success drops.

## 4.5 Updating the Maximum Weight Change

As discussed in section 4.2, the amount of change in a relationship weight from one iteration of the algorithm to the next is computed by multiplying a random real number between  $[-1, 1]$  by the maximum change for each weight. In calculating the maximum change for each weight, we want a function that will decrease over time, yet decrease slowly when the parameter changes are not producing better orderings so as to bounce out of those local maxima. To accomplish these tasks, I chose to update the maximum change as a sum of two parts. The first part represents a percentage of the previous maximum change amount. The second part implements the idea that the maximum change should shrink slowly when the parameter changes do not produce better orderings. To this extent, I calculated the maximum change in the following manner:

$$MaxChange_{n+1} = \alpha * MaxChange_n + (1 - \alpha) * \beta * (AverageSuccessfulChange) \quad (4.1)$$

This function is a modification of one found in a paper by Lester Ingber [1]. I will now discuss each parameter in the following sections.

### 4.5.1 Alpha

In formula 4.1 for recalculating the maximum change allowed in each parameter weight,  $\alpha$  represents the percentage of the maximum change to be determined by the previous maximum change value. Typically, a rate of  $\alpha = 0.9$  provided a good balance between a static value ( $\alpha = 1$ ) and a value that would respond too much when parameter changes did not produce better orderings.

### 4.5.2 Beta and Average Successful Change

In the second half of the formula 4.1 for recalculating the maximum change allowed in each parameter weight, *AverageSuccessfulChange* represents the absolute value of the average change in that parameter. Change in a parameter is measured as the difference in that parameter between iterations of the algorithm, but in this case, we only consider those changes where a better ordering of cards was produced.

In formula 4.1,  $\beta$  represents how much of the average successful change to incorporate into the new maximum change value. When  $\beta$  is low (i.e. in the range  $[0, 0.25]$ ), the average successful change will not be all that important when compared to the previous maximum change value. In contrast, when  $\beta$  is high (i.e. in the range  $[0.75, 1.0]$ ), the average successful change will play a bigger role in determining the new maximum change value. In this way, the formula implements the second goal of shrinking the maximum change value slowly when the parameter changes do not produce better orderings. In such a case, the average successful change will be large because large changes are needed to “bounce” the parameter from a range that is producing poor card orderings.

# Chapter 5

## Results

This chapter details the results of my experiments in testing the Simulated Annealing and Order Production algorithms on various arrangements of information. In section 5.1, I explain the benchmarks against which I evaluated my results. Section 5.2 introduces the scenarios I used in my evaluations and section 5.3 walks through the how I chose the final parameter values for the algorithms. Finally, section 5.4 highlights and analyzes the results the Simulated Annealing algorithm produced for each scenario.

### 5.1 Expected Score of a Random Ordering

As a benchmark for my experiments' results, I needed something to compare to the best score achieved by the Simulated Annealing algorithm. I chose to compare the Simulated Annealing algorithm's score to the expected score of a random ordering of cards. If the Simulated Annealing algorithm is producing meaningful results, the score of the ordering it produces should be very much lower than that of a random ordering of those cards. The computation of the expected score of a random ordering of cards is expressed by the following algorithm:

**Algorithm 5.1.1:** EXPECTEDSCORE( $n$ )

```
eScore ← 0
for  $i \leftarrow 1$  to  $\lfloor n/2 \rfloor$ 
  for  $j \leftarrow n$  downto 1
    eScore ← eScore +  $|j - i| * 2$ 

comment: for odd  $n$ , count the middle card's contributions as well

if  $n \bmod 2 \neq 0$ 
  midway ←  $\lceil n/2 \rceil$ 
  for  $k \leftarrow n$  downto midway
    eScore ← eScore +  $|k - \textit{midway}| * 2$ 

eScore ← eScore/ $n$ 
return (eScore)
```

Table A.1 lists the number of cards in an ordering, compared with the expected score of a random ordering and the maximum score for an ordering of that size.

## 5.2 Experiment Scenarios

To test the Order Producing and Simulated Annealing algorithms, I first needed to ascertain which parameter settings would produce results in the most efficient manner. To do so, I designed a series of experiments for a few different scenarios. I gained inspiration for some of these scenarios from Jintae Lee's Ph.D thesis [3], which describes a system for capturing and analyzing decision rationales. I will now describe the scenarios I worked with and present the arrangement of cards I created for each.

### **5.2.1 Experiment Scenario 1: Where to Eat Lunch?**

The first scenario involves the questions one might have to answer when deciding where to eat lunch, such as “how hungry am I?” and “how much will it cost?” Figure 5-1 shows the cards I created for this scenario and the arrangement of those cards. The numbers on the cards represent the order they were created, which is used to determine the historical relations between cards.

### **5.2.2 Experiment Scenario 2: Paul’s Ordering**

Scenario 2 appears a bit plain at first glance, but has a good story associated with it. While presenting my thesis work to Paul Keel, the EWall Project [4] coordinator, Paul wanted to try out the software for himself. He created the arrangement of cards in figure 5-2 and dictated the target ordering relative to Card A, then ran the Simulated Annealing algorithm. On the very first run, the algorithm produced a near perfect ordering of the cards!

### **5.2.3 Experiment Scenario 3: Thesis Organization**

This scenario came about out of necessity. The arrangement of cards in figure 5-3 represents a first draft of my thesis organization I created long before actually writing my thesis.

### **5.2.4 Experiment Scenario 4: Baseball Standings**

This scenario represents the current standings for the American League of Major League Baseball. The arrangement of cards in figure 5-4 represent the three divisions within the American League and the overall standings.

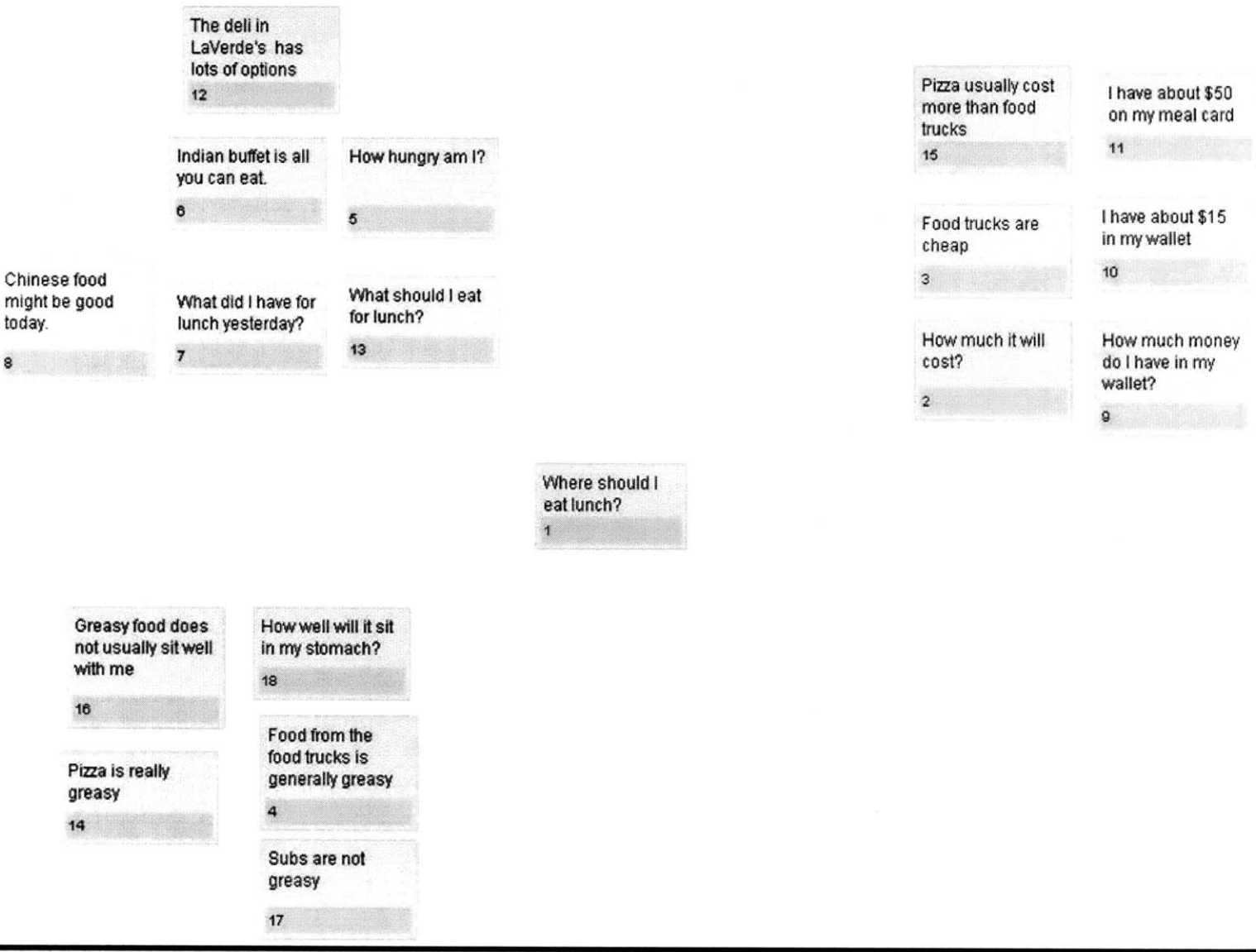


Figure 5-1: This figure includes cards representing the thoughts you might have when you are trying to decide what to eat for lunch. The cards are grouped somewhat by topic. The numbers on the cards represent the order in which they were created. This order is used to determine what historical relations exist between the cards.

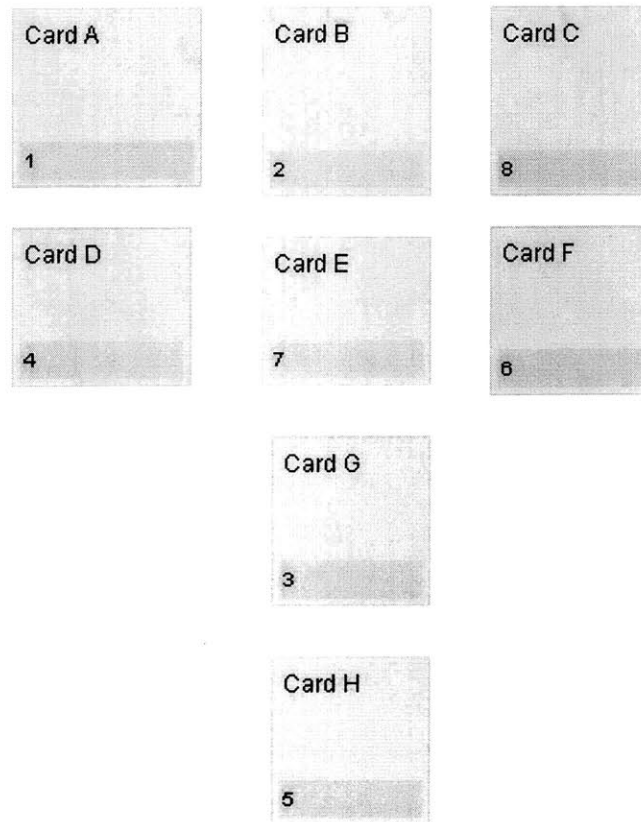


Figure 5-2: The arrangement of cards created by Paul Keel, the EWall project coordinator, when first attempting to use the software I created for this thesis. The numbers on the cards represent the order in which they were created. This order is used to determine what historical relations exist between the cards.

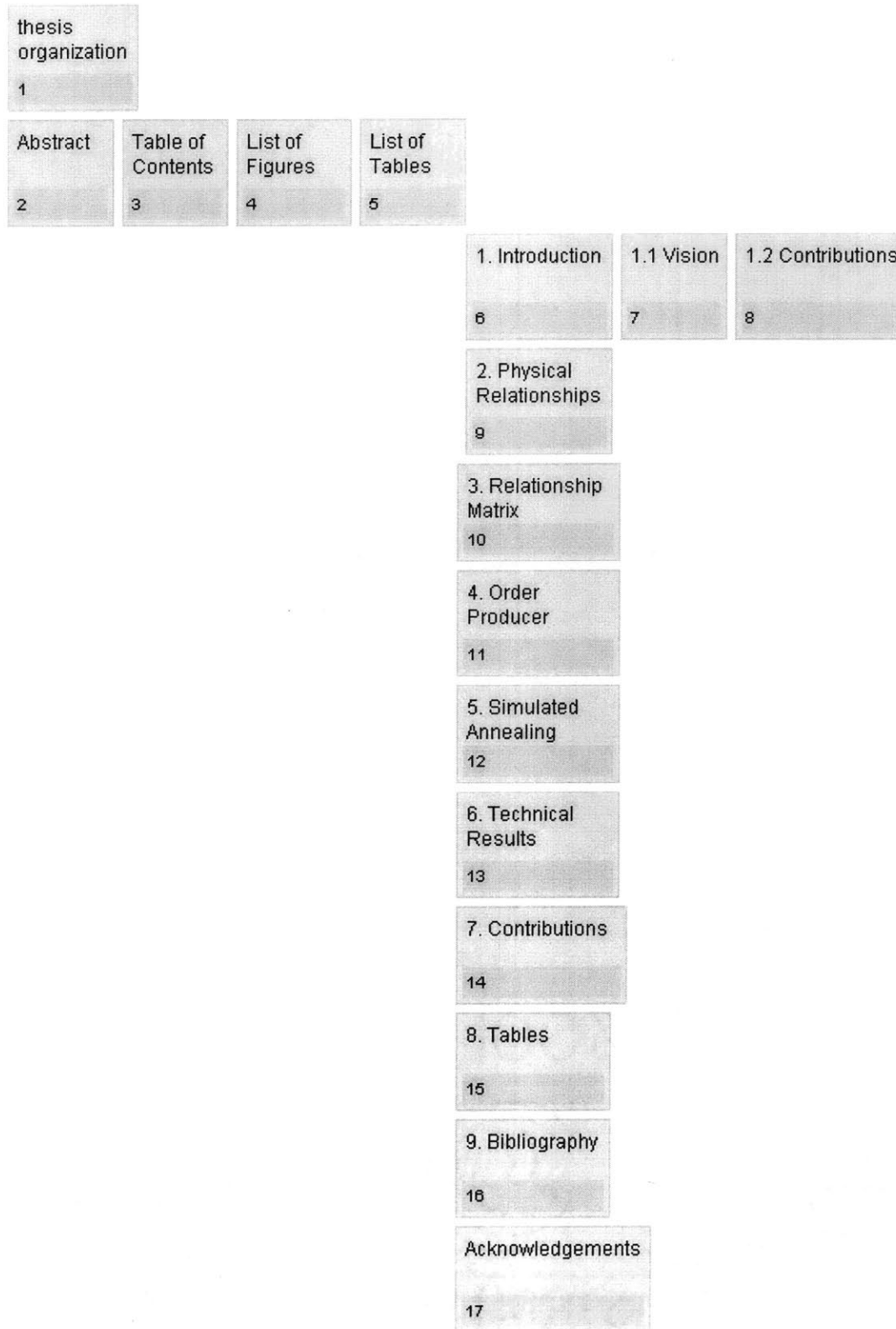


Figure 5-3: This arrangement of cards represents a first draft of my thesis organization, created long before I actually wrote my thesis. The numbers on the bottom part of the cards are the order in which they were created. This order is used to determine what historical relations exist between the cards.

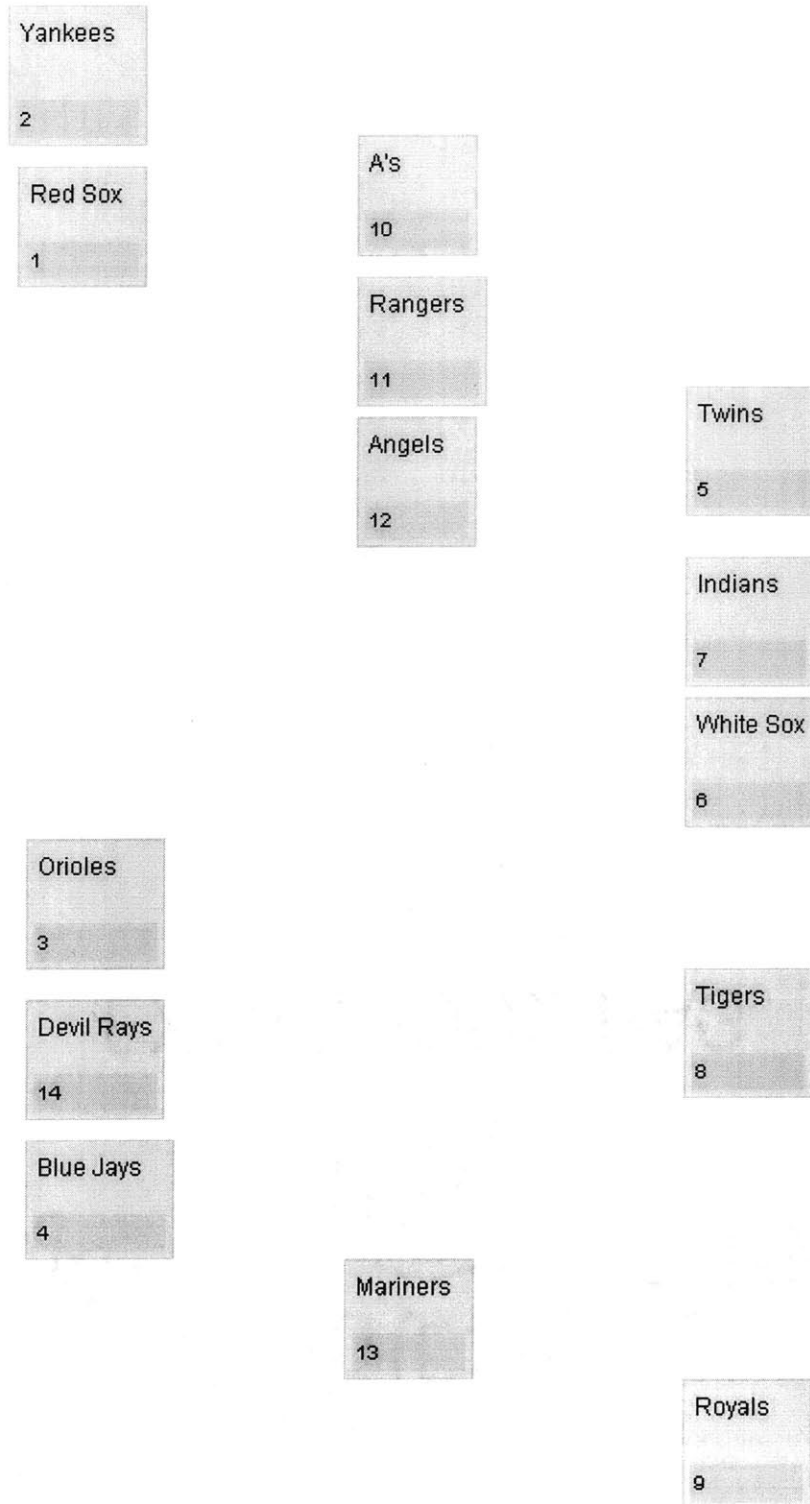


Figure 5-4: This arrangement of cards represents the teams in the American League of Major League Baseball and the overall standings of that league. The numbers on the bottom part of the cards are the order in which they were created. This order is used to determine what historical relations exist between the cards.

## 5.3 Choice of Parameter Values

As discussed in section 4.5, the  $\alpha$  and  $\beta$  parameters affect the Simulated Annealing algorithm by adjusting the maximum amount the relation weights can change on each iteration. Section 3.2.1 detailed how the Order Production algorithm is affected by the decay factor  $\gamma$  chosen to discount longer chains of related cards. I will now discuss the experiments I performed to choose appropriate values for  $\alpha$ ,  $\beta$  and the decay factor  $\gamma$ .

### 5.3.1 Alpha

Section 4.5.1 explained that  $\alpha$  controls how much of the previous maximum change amount a relation weight can shift from one iteration of the Simulated Annealing algorithm to the next. To choose an appropriate value for  $\alpha$ , I ran the Simulated Annealing algorithm multiple times for each experiment scenario while changing the  $\alpha$  parameter and keeping the rest of the parameters constant. Tables A.2 through A.5 outline my results. From the data, I chose to use a value of 0.9 for  $\alpha$  because it produced better results (lower average scores) and less variance of scores. I felt uncomfortable in setting  $\alpha > 0.9$  because doing so would make the maximum weight change too big, causing the Simulated Annealing algorithm to choose seemingly random weights on each iteration of the algorithm.

### 5.3.2 Beta

As discussed in section 4.5.2, the  $\beta$  parameter controls how much of the average successful change in a weight to incorporate into the maximum amount that weight can change between iterations of the Simulated Annealing algorithm. To choose an appropriate value for  $\beta$ , I ran the Simulated Annealing algorithm multiple times for each experiment scenario while changing the  $\beta$  parameter and keeping the rest constant. Tables A.6 through A.9 outline my results. From the data, I chose to use a value of 0.9 for  $\beta$  because it produced good results (low average scores) with somewhat lower variance of scores. In the first two experiment scenarios, scores

generally decreased as  $\beta$  increased, but in the last two experiment scenarios, no clear trend of scores existed. Thus, choosing a higher beta of 0.9 seemed a logical choice for minimizing scores across most scenarios.

### 5.3.3 Decay Factor

Section 3.2.1 explained that the decay factor  $\gamma$  controls how much to discount the strength of relationships between cards in a chain of related cards. The farther away the link you are examining is from the target card, the more discounted that link should be. A high  $\gamma$  near 1.0 means links get discounted less as you move away from the target card, whereas a low  $\gamma$  near 0.0 means links get discounted more.

I chose an appropriate value for  $\gamma$  in the same manner as for  $\alpha$  and  $\beta$ . Tables A.10 through A.13 outline the results of the experiments I ran to choose  $\gamma$ . From the data in these experiments, I chose a  $\gamma$  value of 0.9. The data shows that no value of  $\gamma$  produces good results across all scenarios. For example, in table A.10,  $\gamma = 0.1$  produces a high mean score, but in table A.13,  $\gamma = 0.1$  produces the lowest mean score. This finding suggests that the value of  $\gamma$  one chooses should not drastically effect the overall best ordering produced by the Simulated Annealing algorithm. In particular, I chose a value of  $\gamma = 0.9$  because I did not want to drastically discount links farther away in a chain of related cards.

## 5.4 Relation Weights

Now that I have explained how I chose values for the  $\alpha$ ,  $\beta$  and  $\gamma$  variables, I will present the results I obtained for the four experiment scenarios outlined earlier in section 5.2. To re-cap, I chose the following values for each parameter:  $\alpha = 0.9$ ,  $\beta = 0.9$ ,  $\gamma = 0.9$ , *depth* = maximum possible for each scenario, *IntitalTemperature* = 1000, *FinalTemperature* = 1, *TemperatureDecayFactor* = 0.95. For each scenario, I ran the Simulated Annealing algorithm a total of 100 times and recorded the results.

- In scenario 1 (Where to Eat Lunch), the target card was card 1 (“Where should

I eat lunch”). Since there are 17 cards in the target ordering ( $n - 1$  cards), we see from table A.1 that the maximum score is 144 and the expected score of a random ordering is 96. Table A.14 details my results for this scenario. The average score achieved by the Simulated Annealing algorithm was 72.58, which is 24.4% better than the expected score for a random arrangement of 17 cards. The best score produced during these trials was 62, which was created by the following weights for the relations: Horizontal = 7.6772, Vertical = 9.3596, Close Proximity = -7.5697, Historical = 0.6037.

Comparing these relation weights to the arrangement of the cards in figure 5-1, cards close to each other in the target ordering more likely appeared in horizontal lines or vertical lines, rather than close in historical ordering. For example, the target ordering starts out as [2, 13, 5, 6, ...]. While there would be a historical relationship between cards 5 and 6, there would be none between cards 2, 13 and 5. At the same time, cards 2 and 13 are aligned horizontally (as are cards 5 and 6), and cards 13 and 5 are aligned vertically. Interestingly, most cards in the target ordering are close together physically, thus it is somewhat surprising that the close proximity weighting received such a negative value in comparison to the horizontal and vertical weights.

- For scenario 2 (Paul’s Ordering), the target card was card 1 (“Card A”). Table A.14 details my results, including an average score of 6.1 which represents a score 61.9% better than the expected score of a random arrangement of 7 cards (16). The best ordering produced a score of 2 (a near-perfect ordering!) with the following weights for the relations: Horizontal = 7.9922, Vertical = 3.7449, Close Proximity = -6.0749, Historical = 4.9399.

Comparing these relation weights to the arrangement of the cards in figure 5-2, cards close to each other in the target ordering more likely appeared in horizontal lines than in vertical lines. For example, cards 2 and 8, 6 and 4 are in horizontal lines and are adjacent in the target ordering, while cards 2 and 3 are in a vertical line, yet are not close together in the target ordering. In addition, the weight

on historical relationships makes sense because the target ordering has several pairs of cards close together that have historical relationships between them (i.e. cards 7 and 8, 4 and 5).

- For scenario 3 (Thesis Organization), the target card was card 1 (“Thesis Organization”). Table A.14 details my results, including an average score of 24.56 which represents a score 71.1% better than the expected score of a random arrangement of 16 cards (85). The best ordering produced a score of 14 with the following weights for the relations: Horizontal = -5.3909, Vertical = -0.0874, Close Proximity = -0.3703, Historical = 0.3835.

Comparing these relation weights to the arrangement of the cards in figure 5-3, it makes sense that the largest weights are on vertical relationships and historical relationships. Most of the cards are arranged in a vertical line and the target ordering matches the creation order of the cards.

- For scenario 4 (Baseball Standings), the target card was card 2 (“Yankees”). This scenario was designed to produce the ordering of teams by record, with the Yankees having the best record and the Royals having the worst. Table A.14 details my results, including an average score of 30.9 which represents a score 44.8% better than the expected score of a random arrangement of 13 cards (56). The best ordering produced a score of 20 with the following weights for the relations: Horizontal = -3.9408, Vertical = -3.5083, Close Proximity = -7.2873, Historical = 4.1586.

Comparing these relation weights to the arrangement of the cards in figure 5-4, again it makes sense that the largest weights are on historical relationships. Several adjacent cards in the target ordering were created close together in time (i.e. cards 11 and 12, 7 and 6). Additionally, the cards are grouped into three vertical lines, leading to lots of vertical relationships between the cards, yet the target ordering jumps between the lines and most of the cards in direct alignment with the target card are towards the end of the target ordering.



# Chapter 6

## Discussion

This chapter focuses on the implications of the results reported in chapter 5. I discuss why physical positioning of pieces information can be used to determine which pieces are most related to each other. I then explore why the Simulated Annealing and Order Producing algorithms work for this problem domain while others, such as genetic algorithms, do not. Finally, I explain how we can apply the best relationship weights found for one arrangement of cards to produce relevance orderings in other, similar arrangements.

### 6.1 Why Physical Positioning Analysis Works

Kirsh's work on complementary strategies [2] shows that humans physically arrange information to make sense of it and to ease their cognitive load when performing tasks with this information. It is a natural tendency to group related pieces of information when physically arranging that information (like grouping all your quarters together when counting your change). Everyone arranges information differently, but built into in each person's arrangements are relations between the pieces of information. It is these relations that I attempt to exploit when when analyzing the arrangements for physical and temporal relationships.

The key to the order production process for a particular arrangement is weighting the relationships to determine which is most important for that arrangement. These weights are necessary because everyone arranges information differently, thus you would expect that different arrangements of the same information would have different weights for each type of relation. Even across the four scenarios I used in this thesis, the relation weights which produced the best ordering were different for each scenario.

The Order Producing algorithm takes the relationship weights and produces an overall strength of relationship matrix. This matrix represents the strength of relationship between any two pieces of information in an arrangement. The Relationship Matrix has a direct analogue to grouping related items together. In the Relationship Matrix, the entries for a group of pieces of information that are all strongly related will all be large. Essentially, we are distilling the relationships displayed in the physical positioning of information into a matrix form our algorithms can operate on.

From there, we simply create chains of related pieces of information and discount the links farther away from our target piece. This operation performs much the same as humans consider supporting evidence for an argument. Evidence that directly supports an argument is given a higher weight than evidence that indirectly supports it by reinforcing the pillars of the argument. Thus, the entire order production process works because it uses ideas that have direct analogues to how humans order information physically and sort it mentally.

## **6.2 Why Simulated Annealing is Appropriate**

The Simulated Annealing algorithm works particularly well for this problem domain because the solution space for each scenario is somewhat continuous. During the course of producing my results, I often observed each solution space having multiple global maxima (best solutions) and many local maxima. The Simulated Annealing algorithm that I use hones in on changes in relation weights that produce better orderings from one iteration of the algorithm to the next. When the algorithm gets stuck in a local maxima, the temperature allows a change that produces a worse result

with a certain probability. This design allows the algorithm to “jump out” of local maxima, especially early on when the temperature is high.

Other algorithms, such as genetic or gradient ascent algorithms, would not work as well in this particular problem domain. Genetic algorithms [9] would suffer from the following two problems. First, genetic algorithms would tend to get caught on the many local maxima that appear to be part of each solution space. Second, both genetic and gradient ascent algorithms would tend to change one or two weights at a time between iterations, but in this domain, all the weights probably have to change simultaneously. Changes in the Order Producing algorithm’s results mostly depend on the ratio between the weights, thus if only one weight changes at a time, most of the ratios between the weights will remain the same and the Order Producing algorithm will tend to produce similar results from one iteration to the next. Thus, for any algorithms making use of the Order Producing algorithm, it is important that several relation weights be changed at a time.

The Order Producing algorithm is particularly suited for this domain because it features a way to discount the relation strengths in chains of cards the farther you move from the target card. Conceptually, this is akin to the “supporting evidence for an argument” example I gave earlier in this chapter. In addition, the Order Producing algorithm provides a fast way to evaluate its formulas. This attribute is extremely useful when the Order Producing algorithm is used by other algorithms that perform many iterations, such as the Simulated Annealing algorithm or any genetic algorithm. If an algorithm had to traverse the entire tree in figure 3-4 to evaluate the total value of a card in relation to the target card, it would be much slower than the Order Production algorithm. This fact is due to such an algorithm having an exponential growth in the number of nodes to evaluate with respect to the amount of cards being considered in the ordering. In comparison, the Order Producing algorithm only has to scan once through the relationship matrix and perform a closed form calculation for the decay factor, leading to a much faster algorithm runtime.

## 6.3 Application of Weights to Similar Arrangements

Once we have run the Simulated Annealing algorithm to find the best set of relationship weights for one arrangement of cards, we can then apply these weights to other, similar arrangements. While my thesis did not attempt to do this, I will explain how it could be done. To begin with, one would need a measure of similarity between two arrangements. One possible metric is to compare the relative percentage of relationship types found in each arrangement. Another is to compare the textual content of the cards for similarity, perhaps using lexical attraction [10]. When two arrangements are judged to be sufficiently similar, one could then apply the relationship weights from the known arrangement to the unknown arrangement. With these weights, one could use the Order Production algorithm on the unknown arrangement to produce an ordering of cards relative to one target card that should be close to what a human would produce for that arrangement. A logical extension of this thesis that is particularly suited to the EWall project [4] would be to test such similarity metrics and the scores produced on unknown arrangements.

# Chapter 7

## Contributions

In this thesis, my primary contributions are:

- demonstrating that relationships identified between physically arranged pieces of information are enough to generate orderings of relevance between those pieces of information. The generated orderings of relevance are close to what a human would list for the same pieces of information and are up to 70 percent more accurate than randomly generated orderings.
- demonstrating how a Simulated Annealing algorithm can be used to learn which types of relationships are most important in each arrangement of information. I also explain how the relative weights placed on each type of relationship can then be applied to similar arrangements of information in the future to produce accurate orderings of that information.
- creating formulas used to generate orderings of pieces of information in relation to one target piece. These formulas only need to operate on lists of relationships between those pieces of information and are structured in such a way that allows them to be evaluated extremely quickly. Fast evaluation of such formulas is a necessity for algorithms that complete many iterations before reaching a solution, like Simulated Annealing algorithms or Genetic algorithms.
- creating formulas used to score generated orderings of information. These for-

mulas are simple to evaluate and can be applied to different orderings, regardless of the type of algorithm used to generate those orderings.

Additionally, in this thesis, I have:

- empirically tested several of the parameters of the algorithms to determine how they impact the speed and quality of order production and final solution generation.
- produced working code modules for inclusion and use with the EWall project. These code modules can serve both as a base for further experimentation with various types of relationships, and as a platform on which to build other experiments dealing with user collaboration software.

# Appendix A

## Tables

Table A.1: Number of cards in an ordering compared to the expected score of a random ordering of that size and the maximum score of an ordering of that size.

$n$	$E(\text{Score})$	Max Score
1	0	0
2	1	2
3	$2 \frac{2}{3}$	4
4	5	8
5	8	12
6	$11 \frac{2}{3}$	18
7	16	24
8	21	32
9	$26 \frac{2}{3}$	40
10	33	50
11	40	60
12	$47 \frac{2}{3}$	72
13	56	84
14	65	98
15	$74 \frac{2}{3}$	112
16	85	128
17	96	144
18	$107 \frac{2}{3}$	162

Table A.2: Experiment Scenario 1: Where to Eat Lunch? This table records the scenario results for the  $\alpha$  parameter. All results were produced with the following settings for the other parameters:  $InitialTemperature = 1000$ ,  $FinalTemperature = 1$ ,  $TemperatureDecayFactor = 0.95$ ,  $\beta = 0.9$ ,  $DecayFactor = 0.9$ ,  $depth = \text{maximum}$  for each scenario. For each row of the table, the I ran the Simulated Annealing algorithm 20 times and recorded the minimum score for each run. The data in each row represents the mean and standard deviation of those minimum scores. I list the maximum score and expected score of a random arrangement (as calculated in section 5.1) for reference.

Maximum Score: 144

Expected Score: 96

$\alpha$	Mean Score	Std. Dev.
0.1	87.0	12.490
0.3	83.9	11.706
0.5	81.5	13.040
0.7	77.7	11.828
0.9	72.6	10.179

Table A.3: Experiment Scenario 2: Paul's Ordering. This table records the scenario results for the  $\alpha$  parameter. The settings for the other parameters in this experiment are the same as those described in Table A.2

Maximum Score: 24

Expected Score: 16

$\alpha$	Mean Score	Std. Dev.
0.1	11.7	4.824
0.3	12.2	3.995
0.5	13.0	4.424
0.7	9.2	4.324
0.9	6.5	2.585

Table A.4: Experiment Scenario 3: Thesis Organization. This table records the scenario results for the  $\alpha$  parameter. The settings for the other parameters in this experiment are the same as those described in Table A.2

Maximum Score: 128

Expected Score: 85

$\alpha$	Mean Score	Std. Dev.
0.1	44.7	23.272
0.3	38.2	16.324
0.5	34.4	18.543
0.7	29.8	14.066
0.9	27.7	10.141

Table A.5: Experiment Scenario 4: Baseball Standings. This table records the scenario results for the  $\alpha$  parameter. The settings for the other parameters in this experiment are the same as those described in Table A.2

Maximum Score: 84

Expected Score: 56

$\alpha$	Mean Score	Std. Dev.
0.1	39.2	11.560
0.3	36.8	10.451
0.5	35.7	9.409
0.7	34.0	8.105
0.9	33.6	7.989

Table A.6: Experiment Scenario 1: Where to Eat Lunch? This table records the scenario results for the  $\beta$  parameter. All results were produced with the following settings for the other parameters: *InitialTemperature* = 1000, *FinalTemperature* = 1, *TemperatureDecayFactor* = 0.95,  $\alpha$  = 0.9, *DecayFactor* = 0.9, *depth* = maximum for each scenario. For each row of the table, the I ran the Simulated Annealing algorithm 20 times and recorded the minimum score for each run. The data in each row represents the mean and standard deviation of those minimum scores. I list the maximum score and expected score of a random arrangement (as calculated in section 5.1) for reference.

Maximum Score: 144

Expected Score: 96

$\beta$	Mean Score	Std. Dev.
0.1	78.0	10.219
0.3	76.6	11.185
0.5	76.7	9.044
0.7	74.2	10.660
0.9	72.6	10.179

Table A.7: Experiment Scenario 2: Paul's Ordering. This table records the scenario results for the  $\beta$  parameter. The settings for the other parameters in this experiment are the same as those described in Table A.6

Maximum Score: 24

Expected Score: 16

$\beta$	Mean Score	Std. Dev.
0.1	11.6	3.283
0.3	11.2	4.274
0.5	8.5	5.424
0.7	8.3	2.849
0.9	6.5	2.585

Table A.8: Experiment Scenario 3: Thesis Organization. This table records the scenario results for the  $\beta$  parameter. The settings for the other parameters in this experiment are the same as those described in Table A.6

Maximum Score: 128  
Expected Score: 85

$\beta$	Mean Score	Std. Dev.
0.1	29.0	13.338
0.3	30.5	15.693
0.5	32.2	13.779
0.7	21.4	6.361
0.9	27.7	10.142

Table A.9: Experiment Scenario 4: Baseball Standings. This table records the scenario results for the  $\beta$  parameter. The settings for the other parameters in this experiment are the same as those described in Table A.6

Maximum Score: 84  
Expected Score: 56

$\beta$	Mean Score	Std. Dev.
0.1	35.4	10.802
0.3	33.4	6.492
0.5	33.1	11.742
0.7	29.9	8.447
0.9	33.6	7.989

Table A.10: Experiment Scenario 1: Where to Eat Lunch? This table records the scenario results for the  $\gamma$  decay factor parameter. All results were produced with the following settings for the other parameters: *InitialTemp* = 1000, *FinalTemp* = 1, *TempDecayFactor* = 0.95,  $\alpha = 0.9$ ,  $\beta = 0.9$ , *depth* = maximum for each scenario. For each row of the table, the I ran the Simulated Annealing algorithm 20 times and recorded the minimum score for each run. The data in each row represents the mean and standard deviation of those minimum scores. I list the maximum score and expected score of a random arrangement (as calculated in section 5.1) for reference.

Maximum Score: 144

Expected Score: 96

$\gamma$	Mean Score	Std. Dev.
0.1	73.8	7.016
0.3	74.4	8.249
0.5	73.0	8.473
0.7	73.4	7.486
0.9	72.6	10.179

Table A.11: Experiment Scenario 2: Paul's Ordering. This table records the scenario results for the  $\gamma$  decay factor parameter. The settings for the other parameters in this experiment are the same as those described in Table A.10

Maximum Score: 24

Expected Score: 16

$\gamma$	Mean Score	Std. Dev.
0.1	6.7	1.625
0.3	6.9	3.698
0.5	8.1	4.376
0.7	6.8	3.334
0.9	6.5	2.585

Table A.12: Experiment Scenario 3: Thesis Organization. This table records the scenario results for the  $\gamma$  decay factor parameter. The settings for the other parameters in this experiment are the same as those described in Table A.10

Maximum Score: 128

Expected Score: 85

$\gamma$	Mean Score	Std. Dev.
0.1	25.9	11.814
0.3	30.3	13.830
0.5	27.0	11.360
0.7	22.5	9.644
0.9	27.7	10.142

Table A.13: Experiment Scenario 4: Baseball Standings. This table records the scenario results for the  $\gamma$  decay factor parameter. The settings for the other parameters in this experiment are the same as those described in Table A.10

Maximum Score: 84

Expected Score: 56

$\gamma$	Mean Score	Std. Dev.
0.1	25.5	7.366
0.3	29.1	5.599
0.5	32.7	8.138
0.7	28.7	5.886
0.9	33.6	7.989

Table A.14: This table presents an overall summary of the experiment results for all four scenarios. Average score and standard deviation reported for a sample size of 100 trials with the parameter values detailed in section 5.4.

Scenario:	1: Where to Eat Lunch?	2: Paul's Ordering	3: Thesis Organization	4: Baseball Standings
Target Card:	1 (Where should I eat for lunch?)	1 (Card A)	1 (Thesis Organization)	2 (Yankees)
Target Ordering:	[2,13,5,6,7,12,8,18,4,14,17,16,3,15,9,10,11]	[2,8,7,3,6,4,5]	[2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17]	[10,1,11,12,5,7,6,3,8,14,4,13,9]
Maximum Depth:	17	7	16	13
Maximum Score:	144	24	128	84
Expected Score:	96	16	85	56
Average Score:	72.58	6.1	24.56	30.9
Standard Deviation:	9.102	3.246	11.495	9.182
Best Score:	62	2	14	20
Horizontal Weight:	7.6772	7.9922	-5.3909	-3.9408
Vertical Weight:	9.3596	3.7449	-0.0874	-3.5083
Close Proximity Weight:	-7.5697	-6.0749	-0.3703	-7.2873
Historical Weight:	0.6037	4.9399	0.3835	4.1586

# Bibliography

- [1] Lester Ingber. Simulated annealing: Practice versus theory. *Mathematical and Computer Modelling*, 18(11):29–57, 1993.
- [2] David Kirsh. Complementary strategies: Why we use our hands when we think. In *Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society*, Hillsdale, NJ, 1995. Lawrence Erlbaum. On the web: <http://cogsci.ucsd.edu/~kirsh/Cogsci95/cogsci95.html>.
- [3] Jintae Lee. *Capturing, Reusing, and Managing the Reasons for Decisions*. PhD dissertation, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, February 1992.
- [4] 1996-present. The Electronic Card Wall (EWall) project. Computer Science and Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology. On the web: <http://ewall.mit.edu>.
- [5] William M. Peña and Steven A. Parshall. *Problem Seeking: an Architectural Programming Primer*. John Wiley & Sons, Inc., New York, fourth edition, 2001.
- [6] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*, chapter 4.4. Prentice Hall, Saddle River, New Jersey, first edition, 1995.
- [7] Max Wertheimer. *Productive Thinking*. Greenwood Publishing Group, Westport, Connecticut, reprint edition, 1978.

- [8] Max Wertheimer. Laws of organization in perceptual forms. In S. Yantis, editor, *Visual Perception: Essential Readings (Key Readings in Cognition)*, pages 216–224. Psychology Press, Philadelphia, 2000.
- [9] Patrick H. Winston. *Artificial Intelligence*, chapter 25. Addison-Wesley, Reading, Massachusetts, third edition, 1992.
- [10] Deniz Yuret. *Discovery of Linguistic Relations Using Lexical Attraction*. PhD dissertation, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, May 1998.