Brigham Young University
BYU ScholarsArchive

All Theses and Dissertations

2003-02-27

Satisficing Applied To Simulated Soccer

Jay Packard
Brigham Young University - Provo

Follow this and additional works at: https://scholarsarchive.byu.edu/etd

 Part of the Electrical and Computer Engineering Commons

BYU ScholarsArchive Citation

Packard, Jay, "Satisficing Applied To Simulated Soccer" (2003). *All Theses and Dissertations*. 51.
https://scholarsarchive.byu.edu/etd/51

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.

SATIFICING APPLIED TO SIMULATED SOCCER

by

Jay B. Packard

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Electrical and Computer Engineering

Brigham Young University

December 2002

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Jay B. Packard

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	James K. Archibald, Associate Professor, Chair
Date	Wynn C. Stirling, Professor
Date	Richard L. Frost, Associate Professor

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Jay B. Packard in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date

James K. Archibald, Associate Professor
Chair, Graduate Committee

Accepted for the Department

A. Lee Swindlehurst, Professor
Graduate Coordinator

Accepted for the College

Douglas M. Chabries, Professor
Dean, College of Engineering and Technology

ABSTRACT

SATIFICING APPLIED TO SIMULATED SOCCER

Jay B. Packard

Department of Electrical and Computer Engineering

Master of Science

Satisficing was introduced by the economist Herbert Simon to allow for decisions that are "good enough" when there are insufficient computational resources and knowledge to obtain the optimal outcome. Autonomous multi-agent systems often require such decision making because of the complexity and unknown factors present in such an environment.

Satisficing has been extended significantly by Wynn Stirling. Through extended satisficing, he has departed from conventional approaches to autonomous multi-agent systems, based as they usually are on the assumption that each participant is motivated exclusively by its own self interest, and will therefore attempt to maximize its benefit, regardless of the benefit or cost to others. He considers an alternative view based on the assumption that, when forming its preferences, the

agent is willing to take into consideration the preferences of others.

This thesis explores the application of satisficing to simulated soccer, an autonomous multi-agent system with significant inherent complexity. The work described in this thesis shows that satisficing provides an easy way to switch between an agent's various roles, to take into consideration the likely goals and actions of other agents, and to work in conjunction with a genetic algorithm to help optimize parameters. Some principles of developing simple and concise satisficing code are suggested. Satisficing is thus shown to be an effective solution to decision making in complex multi-agent systems.

ACKNOWLEDGMENTS

I thank my committee for the time they put into reading my thesis and for offering suggestions. I especially thank James Archibald, my advisor, for helping increase my interest in AI and autonomous agents, and making it possible for me to do research in this area. He offered me wisdom from his experience in this area, guidance, and encouragement throughout the programming of my soccer agents and writing of this thesis. I also thank Wynn Stirling for his ground-breaking work in satisficing, from which I draw extensively. I also thank my dad for letting me call him often to draw from his innovative mind, and his taking the time to read my thesis. I also thank my wife Rebecca for showing interest in satisficing, for helping me phrase sentences, and for her constant support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

During the latter part of the 20th century, human equality has been emphasized: equality between men and women, equality between different races, and equal rights for people with different backgrounds in general. The more we become equal, the more we are free to act for ourselves. In many aspects of business, the practice of employers giving many instructions to their employees has given way to employees working in groups with only basic instructions from the employer. These recent trends are consistent with an increased interest in the study of autonomous (without central control) multi-agent systems.

If every agent saw things exactly the same way, it would not be much of a challenge to coordinate multi-agent systems. But many real life situations are not this way. It becomes more of a challenge to coordinate multi-agent systems when each agent has a different view or understanding than the others. It is only in this setting that an agent becomes truly autonomous, because out of different views of the world come different decisions.

There are several ways in which agents may have different views of the world.

In applications that involve the senses, the different views of the world come from limitations of the senses. For example, if agents can see only a short distance in front of them, and if they are each in a different location, then they will each see something different. In systems where agents adopt different roles, each agent pays attention to only what will help fulfill its individual goals. In systems where there is a desired common resource, each agent desires the resource for himself, leading naturally to a desire to allocate the resource differently. In epistemology, the different views of the world are due to the fact that each agent has its own set of beliefs.

Decision making in autonomous multi-agent systems is not as well developed as it is in single agent systems [12]. The majority of the work that has been done with autonomous multi-agent systems has been based on principles from single agent systems, which assume that each agent is motivated exclusively by its own self interest, and will therefore attempt to maximize its benefit, regardless of the benefit or cost of others. But this assumption does not hold in cooperative systems. A good decision-making procedure in autonomous multi-agent systems should be able to take into consideration the preferences of others when forming its own preferences.

Some believe that most everything is competitive in nature and would therefore question the value of a cooperative system. For example, Thomas Hobbes believed that individuals are selfish and that selfish individuals competing leads to a life that is "solitary, poor, nasty, brutish, and short." He concludes that a strong central government is needed [5]. Hobbes would have no confidence in autonomous agents cooperating without a central control. Video games often portray one agent out to kill everyone, suggesting very noncooperative behavior. The theory of biological

evolution often excludes cooperation as a means of survival. But evidence exists that cooperation can often replace competition and be more stable and rewarding to each agent. The breakdown of communism is evidence of the failure of a strong centralized government. In the last few decades, the Senate has shown it can cooperate without strong centralized control as it has become more decentralized, more open, with power more equally distributed [9]. Some computer games have been devised which require cooperation to perform well. The winner of Robert Axelrod's iterated prisoner's dilemma competition, a computer tournament between 2 agents, was an agent that was fundamentally cooperative [2]. Contrary to the standard theory of biological evolution, cooperation has been seen to aid survival and is common between members of the same species and even between members of different species [2]. Studies show, at least with women, that the pleasure and reward-processing centers of the brain are more active during cooperation, thus suggesting that our brains are wired for cooperation [1]. These examples do not prove that individuals are not motivated by self interest, but simply that cooperation can be beneficial to the individual.

One challenge concerning complex autonomous multi-agent systems is how to make decisions when there are insufficient computational resources and knowledge to obtain the optimal outcome. Most humans must make decisions in this sort of environment. We do not know everything that might be relevant to a decision so we cannot consistently make the best choice. Instead, we must choose options that are "good enough". For example, most people will buy a house if it looks good enough and meets some basic criteria despite some minor repairs or renovations that will need to be done. The concept of "good enough" is also important in group decisions, for if

each individual in the group is too picky and wants the very best, they are unlikely to come to an agreement. On the other hand, if every individual is willing to relax their standards somewhat, more options will be acceptable to each agent, making it more likely that they will find one option that is acceptable to everyone.

In the context of decision making, satisficing (introduced by the economist Herbert Simon) allows for choosing options that are good enough. According to Simon, as one is searching the range of options, the first option that is good enough should be chosen if it is above a certain aspiration level. No other options will be considered after that point [13]. The problem with this approach is that it requires experience to determine what this aspiration level should be. If it is too low, then it will be far from the optimal outcome and performance will be decreased. If it is too high, then it will either require too much time or too many resources to find an acceptable option, or no acceptable option will be found.

Wynn Stirling draws from the ideas of Isaac Levi to extend satisficing and the concept of "good enough". Instead of comparing attributes to externally supplied aspiration levels, he suggests the intrinsic approach of comparing the positive and negative attributes of each option. Those options which have positive attributes that outweigh the negative attributes are defined to be good enough. He also draws from John Maynard Keynes, Rudolf Carnap, and Isaac Levi in extending satisficing to draw from the laws of probability applied to preferences. This extension allows for consideration of other agent's preferences and thus permits cooperation. Stirling's version of satisficing is a promising approach to decision making in autonomous multi-agent systems for it includes the concept of "good enough" and allows for consideration of

other's preferences, both of which are needed for a good cooperative system. As will be seen, it also handles competitive situations well.

The focus of my thesis is to explore the feasibility of Stirling's extended version of satisficing in the context of decision making in autonomous multi-agent systems. Satisficing is applied to simulated soccer using the RoboCup simulator test bed since soccer is a good example of an autonomous multi-agent system that is complex enough to be interesting.

I draw from my experience in developing satisficing code to present six advantages of satisficing:

1. **Role Switching**: It is often important to be able to switch roles in a multi-agent system. Satisficing provides a straight forward way to accomplish this.

2. **Sensitivity**: Using the preference rules explained in Chapter 2, an agent using satisficing has the ability to consider all the dependencies that exist; thus, it is sensitive to both the likely goals and actions of teammates and the likely goals and actions of the opponents.

3. **Ease of Changing an Option's Importance or Priority**: When one hard codes the relative importance or priority of different options, it is difficult to alter them. With the satisficing approach, the task of altering an option's importance or priority becomes simpler.

4. **Function Reusability**: In soccer and in many other multi-agent applications, agents within a team have many of the same goals as their opponents (in opposition of course). Satisficing encourages reusing functions for agents within

a team and for predicting the actions of the opponent, thus making the code more concise and easy to maintain.

5. **Implementation Simplicity**: Satisficing tends to be simple to implement because it separates the code into five main steps: declare variables, find the utility of each option, normalize selectabilities and rejectabilities, choose the option with the highest utility, and execute the option with the highest utility. I provide a template in C++ which incorporates these five main categories. It can be copied directly and the appropriate blanks filled in. This is likely to save the programmer time in deciding how to organize his code. I suggest the use of $importancefactors$, which makes it even easier to alter the importances or priorities of the options.

6. **Genetic Algorithm Readiness**: In many competitive situations, an agent or group of agents can win by just a fraction of a second or by one point. It therefore seems that optimization is important in competitive situations, such as simulated soccer. Genetic algorithms in conjunction with satisficing are shown to be a promising approach to global optimization.

I will discuss two pitfalls to avoid when implementing satisficing:

1. **Apparent Preferential Total Law Dilemma**: When solving for an individual preference using the total law pertaining to preferences, there may be one term that is a function of an option and another term that is a function of the inverse of that option, but yet they both add to the individual preference. To

one not familiar with satisficing, the two terms appear to contradict each other. This apparent contradiction will be resolved.

2. **Apparent Equality in Maximizing Utility From All Options**: In choosing the option with the highest utility, it may appear that the results are the same whether the option is chosen from amongst all options versus from amongst only the satisficing options. An example is presented that illustrates that the results can be different. The second alternative has the advantage of allowing decision making when an agent does not have the resources to consider all possible options.

I will present the results of applying a genetic algorithm to my soccer team, showing that performance is increased. I will compare the relative program size and CPU usage of my team with the other teams to conclude that my code is concise in comparison, and that my team requires an average amount of CPU usage. I will present the competition results of my soccer team against several others, demonstrating that my team performs competitively. I will suggest some future work that is likely to improve scoring, including thoroughness, precision, and low level machine learning.

Chapter 2

Overview of Satisficing

This chapter gives an overview of satisficing with its probabilistic extensions and intrinsic approach of comparing the positive and negative attributes of each option.

2.1 Selectability and Rejectability

An option is satisficing in the context of decision making if its pros outweigh its cons. An option is said to be satisficing if the decision maker's preference to select it given the pros (selectability for short) is greater than a boldness factor, b, times his preference to reject it given the cons (rejectability for short). The term "preference" is used to refer to either selectability or rejectability. The selectability is denoted p_S and the rejectability is denoted p_R. The "p" in this notation stems from probability theory. An option, v, is defined to be satisficing if

$$p_S(v) > bp_R(v).$$

The boldness factor is the degree to which the decision maker is willing to risk rejecting an option despite its selectability so as to not bring in its rejectability. This will be

addressed in more detail shortly. Options that are not satisficing should be eliminated from the set of options under consideration. As a matter of convention, the preference of an option varies between 0 and 1, and the sum of the preferences over each option is equal to 1.

If a single option must be chosen, it makes sense to choose the option with the highest utility measure of some sort among the satisficing options. One reasonable way to define a utility measure, U, of an option, v, is

$$U(v) \equiv p_S(v) - bp_R(v).$$

After eliminating the nonsatisficing options, the utility of each option can be calculated and the option with the highest utility chosen.

Sometimes there is a notion of a group decision. In such cases, joint selectabilities and rejectabilities are also defined. For agents 1, 2,...n with options v_1,v_2,...v_n, the joint selectability is denoted

$$p_{S_1S_2,...S_n}(v_1, v_2, ...v_n)$$

and the joint rejectability is denoted

$$p_{R_1R_2,...R_n}(v_1, v_2, ...v_n)$$

and the joint utility function, $U_{1,2,...n}$, is defined as

$$U_{1,2,...n}(v_1, v_2, ...v_n) \equiv p_{S_1S_2,...S_n}(v_1, v_2, ...v_n) - bp_{R_1R_2,...R_n}(v_1, v_2, ...v_n).$$

In this case, b is a joint boldness factor which has been agreed upon by the group. If the group decides to choose the option with the highest joint utility, they will each choose their individual option which maximizes the joint utility.

Selectability and rejectability can be based on a variety of factors. For example, if one is considering buying an item, the selectability could be based on how likeable the item is, and the rejectability based on the cost of the item. In situations where an energy-conserving agent has a destination, the selectability of a move could be based on the distance closer to the destination, and the rejectability based on the energy required. Sometimes the selectability and rejectability turn out to be proportional, which creates more of a dilemma for the decision maker. This proportionality is likely to exist when one is considering whether to buy an item, for the more likeable items tend to be more expensive. It may or may not be the case for an energy-conserving agent with a destination. If there are no obstacles in the way of the destination, going straight towards the destination will result in the highest selectability and the lowest rejectability, since this move will take him closer to the goal and require less energy than any other move. In this case, the selectability and rejectability are inversely proportional. Assume, however, there is a hill between the agent and the destination and that the shortest path to the destination is straight over the hill. Also, assume the path that requires the least amount of energy is around the hill. In this case, the selectability and rejectability are proportional.

In epistemology, as Isaac Levi illustrates, a selectability and rejectability exist when deciding whether to exclude a belief from one's corpus of knowledge or not [7]. The selectability of excluding the belief in Levi's system is the incurred informational value. The rejectability of excluding the belief is the incurred likelihood of error. When the informational value and incurred error are inversely proportional, it is quite easy to decide whether to exclude a belief or not, for when the incurred informational

value of excluding the belief is high and the incurred error low, it is obvious that the belief should be excluded, and when the incurred informational value of excluding the belief is low and the incurred error high, it is obvious that the belief should be included. When the informational value and incurred error are proportional, it becomes more difficult to decide. As one such example, say one of 3 people are suspected of committing a crime but there is not sufficient proof to decide which one is guilty. Let the belief under consideration by a judge be, "I find person X (out of the 3) guilty of the crime". The informational value of blaming person X is high because this way one person can be punished. The incurred error of blaming person X is also high because the crime may have been committed by one of the others.

The boldness factor previously mentioned allows for personal bias. The term "boldness" originates from Levi, who uses the term in reference to one's boldness in excluding a belief from his corpus despite the incurred error. Different boldness factors lead to different choices. To illustrate this, say that the selectability of buying an item is .7 and the rejectability is .9. If I am thrifty with money, the cost of the item will be of great concern to me, hence, I will have a high boldness factor, say 1. Since the selectability of .7 is not greater than my boldness factor times the rejectability equal to $1 \times .9 = .9$, buying the item is not satisficing, so I will not buy it. If I am carefree with money, the cost is not so important, hence, I will have a low boldness factor, say .3. Since the selectability of .7 is greater than the boldness factor times the rejectability equal to $.3 \times .9 = .27$, buying the item is satisficing, so I will buy the item. This example illustrates how an option may or may not be satisficing depending on differences in boldness.

A boldness factor may change depending on the option. For example, say a driver's options are to drive on a road lying on flat ground at some speed, or to drive on a road near the edge of a cliff at some speed. Let the selectability be proportional to the speed, and the rejectability proportional to the chance of going off the road at that speed. If the driver values his life, he will place more of an emphasis on the rejectability (by setting the boldness factor high) if he is driving along the edge of a cliff, for going off the road in this case means certain death. This situation could be thought of differently, however, such that the boldness factor does not change depending on the option. Assume the same options for the driver as before. Let the selectability be proportional to the speed, and the rejectability proportional to the chance of getting killed at that speed. In this case, the rejectability of going off the road near the edge of the cliff is greater than on flat ground since that is more likely to lead to death. The driver values his life to a certain degree regardless of the option. Therefore, he only needs one boldness factor for all options.

2.2 Dependent selectabilities and rejectabilities

When making decisions in the context of a group, the preferences of an agent or group of agents may depend on preferences of other agents. When these dependencies exist, or when deriving joint preferences from individual preferences, or vice versa, the laws of probability provide an excellent framework for modeling preferences. From now on, for brevity's sake, when referring to satisficing, I am referring to satisficing with these probabilistic extensions. It will be shown how independent, joint, and conditional preferences can be described using the structures of marginal, joint and

conditional probability, respectively, in the context of multiple sample spaces. In this section, satisficing is presented in the most general way possible to aid anyone interested in applying satisficing to other applications. My particular application of satisficing, simulated soccer, does not incorporate all the material presented.

2.2.1 Marginal, Joint, and Conditional Probability

In this section, a brief overview of probability theory in the context of multiple sample spaces will be presented [14]. Probability theory in this context defines the following concepts: marginal, joint, and conditional probability. In order to define these, we need some preliminary definitions. I have taken some liberty in naming symbols to avoid collisions with symbols later on. There exist sample spaces V_1, V_2,...V_n where v_i denotes elementary events of V_i, and $\mathcal{V}_i$ denotes events of V_i. A rectangle comprised of $\mathcal{V}_1$, $\mathcal{V}_2$,...$\mathcal{V}_n$ is denoted and defined as

$$\mathcal{V}_1 \times \mathcal{V}_2 \times ...\mathcal{V}_n \equiv \{(v_1, v_2, ...v_n) : v_1 \in \mathcal{V}_1, v_2 \in \mathcal{V}_2, ...v_n \in \mathcal{V}_n\}.$$

A rectangle may also be defined for any combination of $\mathcal{V}_i$'s such as

$$\mathcal{V}_2 \times \mathcal{V}_4 \times \mathcal{V}_5 \equiv \{(v_2, v_4, v_5) : v_2 \in \mathcal{V}_2, v_4 \in \mathcal{V}_4, v_5 \in \mathcal{V}_5\}.$$

Probability measures are functions of rectangles. The marginal probability measures of the single element rectangles, $\mathcal{V}_1$, $\mathcal{V}_2$,...$\mathcal{V}_n$, are denoted P_1, P_2,...P_n, respectively. According to standard convention, three constraints must be met to be considered a probability measure: the probability of a rectangle must be between 0 and 1, the probability of a rectangle consisting of all the sample spaces is equal to 1, and for any set of mutually exclusive rectangles, the probability of their union is equal to the

sum of the probability of each rectangle. For any $\mathcal{V}_1$, $\mathcal{V}_2$,...$\mathcal{V}_n$, the joint probability measure is denoted

$$P_{1\times 2\times ...n}(\mathcal{V}_1 \times \mathcal{V}_2 \times ...\mathcal{V}_n).$$

The joint probability may also be defined for any combination of $\mathcal{V}_i$'s such as

$$P_{1\times 3}(\mathcal{V}_1 \times \mathcal{V}_3).$$

For any $\mathcal{V}_1$, $\mathcal{V}_2$,...$\mathcal{V}_j$ given $\mathcal{V}_{j+1}$,$\mathcal{V}_{j+2}$,...$\mathcal{V}_n$, the joint conditional probability measure is denoted

$$P_{1\times ...j|(j+1)\times ...n}(\mathcal{V}_1 \times ...\mathcal{V}_j|\mathcal{V}_{j+1} \times ...\mathcal{V}_n).$$

If there is only one element to the left of the |, this equation becomes a *marginal* conditional probability measure. The joint conditional probability may also be defined for any combination of $\mathcal{V}_i$'s given any combination of different $\mathcal{V}_i$'s such as

$$P_{2\times 6|1\times 3\times 5}(\mathcal{V}_2 \times \mathcal{V}_6|\mathcal{V}_1 \times \mathcal{V}_3 \times \mathcal{V}_5).$$

In many cases, one does not have direct access to the phenomenon being modeled, but only to observations such as sensor output. The output is often calibrated to provide numerical values, but could also be calibrated to signal an event, for example, to ring a bell or turn on a light. In many cases, several elementary events in the sample space are indistinguishable from each other through observation. For example, if the sample space represents a voltage, say, in the continuous range [0,5], a voltage meter may only give discrete values, say, 0,1,2,3,4,5. Thus, it is likely that the voltage meter will output 3 for any value within the sample space in the range (2.5,3.5). Discrete random variables are functions which map from a sample space

to a discrete derived space which I will call the observation space. I will not discuss continuous observation spaces.

Let X_1, X_2,...X_n be discrete random variables that map from V_1, V_2,...V_n to the discrete observation spaces W_1, W_2,...W_n, respectively, where w_i denotes elementary events of W_i, and $\mathcal{W}_i$ denotes events of W_i. A rectangle comprised of $\mathcal{W}_1$, $\mathcal{W}_2$,...$\mathcal{W}_n$ is denoted and defined as

$$\mathcal{W}_1 \times \mathcal{W}_2 \times ...\mathcal{W}_n \equiv \{(w_1, w_2, ...w_n) : w_1 \in \mathcal{W}_1, w_2 \in \mathcal{W}_2, ...w_n \in \mathcal{W}_n\}.$$

A rectangle may also be defined for any combination of $\mathcal{W}_i$'s such as

$$\mathcal{W}_{13} \times \mathcal{W}_{14} \times \mathcal{W}_{20} \equiv \{(w_{13}, w_{14}, w_{20}) : w_{13} \in \mathcal{W}_{13}, w_{14} \in \mathcal{W}_{14}, w_{20} \in \mathcal{W}_{20}\}.$$

The derived marginal probability measure, P_{X_i}, of a single element rectangle, $\mathcal{W}_i$, is defined as

$$P_{X_i}(\mathcal{W}_i) \equiv P_i({X_i}^{-1}(\mathcal{W}_i)).$$

${X_i}^{-1}$ is called the inverse image of X_i and maps from W_i to V_i. For any $\mathcal{W}_1$, $\mathcal{W}_2$,...$\mathcal{W}_n$, the derived joint probability measure,$P_{X_1 \times X_2 \times ...X_n}$, is defined as

$$P_{X_1 X_2 ...X_n}(\mathcal{W}_1 \times \mathcal{W}_2 \times ...\mathcal{W}_n) \equiv P_{1\times 2\times ...n}({X_1}^{-1}(\mathcal{W}_1) \times {X_2}^{-1}(\mathcal{W}_2) \times ...{X_n}^{-1}(\mathcal{W}_n)).$$

The derived joint probability may also be defined for any combination of $\mathcal{W}_i$'s such as

$$P_{X_2 X_3 X_4}(\mathcal{W}_2 \times \mathcal{W}_3 \times \mathcal{W}_4) \equiv P_{2\times 3\times 4}({X_2}^{-1}(\mathcal{W}_2) \times {X_3}^{-1}(\mathcal{W}_3) \times {X_4}^{-1}(\mathcal{W}_4)).$$

For any $\mathcal{W}_1$,...$\mathcal{W}_j$ given $\mathcal{W}_{j+1}$,...$\mathcal{W}_n$, the derived joint conditional probability measure, $P_{X_1 ... X_j | X_{j+1} ... X_n}$, is defined as

$$P_{X_1 ... X_j | X_{j+1} ... X_n}(\mathcal{W}_1 \times ...\mathcal{W}_j | \mathcal{W}_{j+1} \times ...\mathcal{W}_n) \equiv$$

$$P_{1\times\ldots j|(j+1)\times\ldots n}({X_1}^{-1}(\mathcal{W}_1)\times\ldots{X_j}^{-1}(\mathcal{W}_j)|{X_{j+1}}^{-1}(\mathcal{W}_{j+1})\times\ldots{X_n}^{-1}(\mathcal{W}_n)).$$

If there is only one element to the left of the |, this equation becomes a derived *marginal* conditional probability measure. The derived joint conditional probability may also be defined for any combination of $\mathcal{W}_i$'s given any combination of different $\mathcal{W}_i$'s such as

$$P_{X_1X_5|X_3X_4X_6}(\mathcal{W}_1\times\mathcal{W}_5|\mathcal{W}_3\times\mathcal{W}_4\times\mathcal{W}_6)\equiv$$

$$P_{1\times5|3\times4\times6}({X_1}^{-1}(\mathcal{W}_1)\times{X_5}^{-1}(\mathcal{W}_5)|{X_3}^{-1}(\mathcal{W}_3)\times{X_4}^{-1}(\mathcal{W}_4)\times{X_6}^{-1}(\mathcal{W}_6)).$$

For any w_i the marginal probability mass function, p_{X_i}, is defined as

$$p_{X_i}(w_i)\equiv P_i(\{v_i : X_i(v_i)=w_i\}).$$

Probability mass functions are primarily for convenience when dealing with singleton sets. Since probability mass functions are basically probability measures of singleton sets, the same three constraints apply to probability mass functions as they do to probability measures. For any w_1, w_2,...w_n, the joint probability mass function, $p_{X_1X_2\ldots X_n}$, is defined as

$$p_{X_1X_2\ldots X_n}(w_1,w_2,\ldots w_n)\equiv$$

$$P_{1\times2\times\ldots n}(\{v_1 : X_1(v_1)=w_1\}\times\{v_2 : X_2(v_2)=w_2\}\times\ldots\{v_n : X_n(v_n)=w_n\}).$$

The joint probability mass function may also be defined for any combination of w_i's such as

$$p_{X_6X_8X_{10}}(w_6,w_8,w_{10})\equiv$$

$$P_{6\times8\times10}(\{v_6 : X_6(v_6)=w_6\}\times\{v_8 : X_8(v_8)=w_8\}\times\{v_{10} : X_{10}(v_{10})=w_{10}\}).$$

For any w_1,...w_j given w_{j+1},...w_n, the joint conditional probability mass function, $p_{X_1...X_j|X_{j+1}...X_n}$, is defined as

$$p_{X_1...X_j|X_{j+1}...X_n}(w_1, ...w_j|w_{j+1}, ...w_n) \equiv$$

$$P_{1\times...j|j+1\times...n}(\{v_1 : X_1(v_1) = w_1\} \times ...\{v_j : X_j(v_j) = w_j\} \mid$$

$$\{v_{j+1} : X_{j+1}(v_{j+1}) = w_{j+1}\} \times ...\{v_n : X_n(v_n) = w_n\}).$$

If there is only one element to the left of the |, this equation becomes a *marginal* conditional probability mass function. The joint conditional probability mass function may also be defined for any combination of w_i's given any combination of different w_i's such as

$$p_{X_1X_3X_5|X_4}(w_1, w_3, w_5|w_4) \equiv$$

$$P_{1\times3\times5|4}(\{v_1 : X_1(v_1) = w_1\} \times \{v_3 : X_3(v_3) = w_3\} \times \{v_5 : X_5(v_5) = w_5\} \mid$$

$$\{v_4 : X_4(v_4) = w_4\}).$$

According to the chain rule of probability, for any w_1, w_2,...w_n, the joint probability mass function may be obtained from the marginal and conditional probability mass functions in this way

$$p_{X_1X_2...X_n}(w_1, w_2, ...w_n) =$$

$$p_{X_n|X_{n-1}X_{n-2}...X_1}(w_n|w_{n-1}w_{n-2}, ...w_1)...p_{X_3|X_2X_1}(w_3|w_2, w_1)p_{X_2|X_1}(w_2|w_1)p_{X_1}(w_1).$$

Any X_i along with its corresponding w_i on the right hand can be switched with any other X_i along with its corresponding w_i and obtain a mathematically correct equation. Thus, there are many combinations possible. However, some combinations

are more naturally motivated by the situation. Furthermore, these structures all assume maximum interdependence. In many cases, such interdependence will not occur, and the structure is simplified to exclude dependencies that do not exist. For example, it might reduce to

$$p_{X_1X_2...X_n}(w_1, w_2, ...w_n) = p_{X_n|X_3X_1}(w_n|w_3, w_1)...p_{X_3}(w_3)p_{X_2|X_1}(w_2|w_1)p_{X_1}(w_1).$$

In the extreme case, if there is no interdependence, the joint probability mass function reduces to

$$p_{X_1X_2...X_n}(w_1, w_2, ...w_n) = p_{X_n}(w_n)...p_{X_3}(w_3)p_{X_2}(w_2)p_{X_1}(w_1).$$

A joint probability mass function may also be obtained for any combination of w_i's with their simplified interdependencies such as

$$p_{X_7X_5X_3}(w_7, w_5, w_3) = p_{X_3|X_5}(w_3|w_5)p_{X_5|X_7}(w_5|w_7)p_{X_7}(w_7).$$

According to the total law of probability, the marginal probability mass function of w_i can be obtained from the joint probability mass functions of w_1, w_2,...w_n in this way

$$p_{X_i}(w_i) = \sum_{w_1} ... \sum_{w_{i-1}} \sum_{w_{i+1}} ... \sum_{w_n} p_{X_1X_2...X_n}(w_1, w_2, ...w_n).$$

Similarly, the joint probability mass function of w_1,...w_i can be obtained from the joint probability mass function of w_1, w_2,...w_n as follows

$$p_{X_1...X_i}(w_1, ...w_i) = \sum_{w_{i+1}} \sum_{w_{i+2}} ... \sum_{w_n} p_{X_1X_2...X_n}(w_1, w_2, ...w_n).$$

A joint probability mass function may also be obtained for any combination of w_i's from a joint probability mass function that contains these and other w_i's such as

$$p_{X_2X_4}(w_2, w_4) = \sum_{w_1} \sum_{w_5} \sum_{w_8} p_{X_1X_2X_4X_5X_8}(w_1, w_2, w_4, w_5, w_8).$$

2.2.2 Individual, Joint, and Conditional Preferences

If the term "probability" is replaced with "selectability" or "rejectability", we may use the math from probability theory with an implied context to decision making. This section proceeds similarly as the previous section in order to emphasize the parallel between satisficing and probability theory. Satisficing defines the following concepts: individual, joint, and conditional selectability, rejectability, and interdependency. Interdependency simultaneously accounts for both selectability and rejectability. In order to define these concepts, we need some preliminary definitions. For agents 1, 2,...n, there exists action spaces V_1, V_2,...V_n, respectively, where v_i and $\mathtt{v}_i$ denote options of V_i, and $\mathcal{V}_i$ and $\mathtt{V}_i$ denote subsets of V_i. The term "action space" is appropriate for decision making since the act of making a decision can be considered a form of action. A rectangle comprised of $\mathcal{V}_1$, $\mathcal{V}_2$,...$\mathcal{V}_n$ is denoted and defined as

$$\mathcal{V}_1 \times \mathcal{V}_2 \times ...\mathcal{V}_n \equiv \{(v_1, v_2, ...v_n) : v_1 \in \mathcal{V}_1, v_2 \in \mathcal{V}_2, ...v_n \in \mathcal{V}_n\}.$$

A rectangle may also be defined for any combination of $\mathcal{V}_i$'s such as

$$\mathcal{V}_2 \times \mathcal{V}_4 \times \mathcal{V}_5 \equiv \{(v_2, v_4, v_5) : v_2 \in \mathcal{V}_2, v_4 \in \mathcal{V}_4, v_5 \in \mathcal{V}_5\}.$$

Selectability, rejectability, and interdependency measures are functions of rectangles. The individual selectability measures of the single element rectangles, $\mathcal{V}_1$, $\mathcal{V}_2$,...$\mathcal{V}_n$, are denoted P_{S_1}, P_{S_2},...P_{S_n}, respectively. The individual rejectability measures of the single element rectangles, $\mathtt{V}_1$, $\mathtt{V}_2$,...$\mathtt{V}_n$, are denoted P_{R_1}, P_{R_2},...P_{R_n}, respectively. The individual interdependency measures of the rectangles, $\mathcal{V}_1 \times \mathtt{V}_1$, $\mathcal{V}_2 \times \mathtt{V}_2$,...$\mathcal{V}_n \times \mathtt{V}_n$, are denoted $P_{S_1 \times R_1}$, $P_{S_2 \times R_2}$,...$P_{S_n \times R_n}$, respectively. As a matter of convention, all

preferences are subject to three constraints: the preference of a rectangle must be between 0 and 1, the preference of a rectangle consisting of all action spaces is equal to 1, and for any set of mutually exclusive rectangles, the preference of their union is equal to the sum of the preference of each rectangle. For any $\mathcal{V}_1$, $\mathcal{V}_2$,...$\mathcal{V}_n$, the joint selectability measure is denoted

$$P_{S_1 \times S_2 \times ... S_n}(\mathcal{V}_1 \times \mathcal{V}_2 \times ... \mathcal{V}_n).$$

The joint selectability measure may also be defined for any combination of $\mathcal{V}_i$'s such as

$$P_{S_1 \times S_3}(\mathcal{V}_1 \times \mathcal{V}_3).$$

For any $\mathtt{V}_1$, $\mathtt{V}_2$,...$\mathtt{V}_n$, the joint rejectability measure is denoted

$$P_{R_1 \times R_2 \times ... R_n}(\mathtt{V}_1 \times \mathtt{V}_2 \times ... \mathtt{V}_n).$$

The joint rejectability measure may also be defined for any combination of $\mathtt{V}_i$'s such as

$$P_{R_1 \times R_3}(\mathtt{V}_1 \times \mathtt{V}_3).$$

For any $\mathcal{V}_1$,...$\mathcal{V}_n$ and $\mathtt{V}_1$,...$\mathtt{V}_n$, the joint interdependency measure is denoted

$$P_{S_1 \times ... S_n \times R_1 \times ... R_n}(\mathcal{V}_1 \times ... \mathcal{V}_n \times \mathtt{V}_1 \times ... \mathtt{V}_n).$$

The joint interdependency measure may also be defined for any combination of $\mathcal{V}_i$'s and $\mathtt{V}_i$'s such as

$$P_{S_1 \times S_3 \times R_1 \times R_6}(\mathcal{V}_1 \times \mathcal{V}_3 \times \mathtt{V}_1 \times \mathtt{V}_6).$$

For any $\mathcal{V}_1$,...$\mathcal{V}_j$ given $\mathcal{V}_{j+1}$,...$\mathcal{V}_n$, the joint conditional selectability measure is denoted

$$P_{S_1 \times ... S_j | S_{j+1} \times ... S_n}(\mathcal{V}_1 \times ... \mathcal{V}_j | \mathcal{V}_{j+1} \times ... \mathcal{V}_n).$$

If there is only one element to the left of the |, this equation becomes an *individual* conditional selectability measure. The joint conditional selectability measure can also be defined for any combination of $\mathcal{V}_i$'s given any combination of different $\mathcal{V}_i$'s such as

$$P_{S_2 \times S_6 | S_1 \times S_3 \times S_5}(\mathcal{V}_2 \times \mathcal{V}_6 | \mathcal{V}_1 \times \mathcal{V}_3 \times \mathcal{V}_5).$$

For any $\mathtt{V}_1$,...$\mathtt{V}_j$ given $\mathtt{V}_{j+1}$,...$\mathtt{V}_n$, the joint conditional rejectability measure is denoted

$$P_{R_1 \times ... R_j | R_{j+1} \times ... R_n}(\mathtt{V}_1 \times ... \mathtt{V}_j | \mathtt{V}_{j+1} \times ... \mathtt{V}_n).$$

If there is only one element to the left of the |, this equation becomes an *individual* conditional rejectability measure. The joint conditional rejectability measure can also be defined for any combination of $\mathtt{V}_i$'s given any combination of different $\mathtt{V}_i$'s such as

$$P_{R_2 \times R_6 | R_1 \times R_3 \times R_5}(\mathtt{V}_2 \times \mathtt{V}_6 | \mathtt{V}_1 \times \mathtt{V}_3 \times \mathtt{V}_5).$$

For any $\mathcal{V}_1$,...$\mathcal{V}_j$,$\mathtt{V}_1$,...$\mathtt{V}_k$ given $\mathcal{V}_{j+1}$,...$\mathcal{V}_n$,$\mathtt{V}_{k+1}$,...$\mathtt{V}_n$, the joint conditional interdependency measure is denoted

$$P_{S_1 \times ... S_j \times R_1 \times ... R_k | S_{j+1} \times ... S_n \times R_{k+1} \times ... R_n}(\mathcal{V}_1 \times ... \mathcal{V}_j \times \mathtt{V}_1 \times ... \mathtt{V}_k | \mathcal{V}_{j+1} \times ... \mathcal{V}_n \times \mathtt{V}_{k+1} \times ... \mathtt{V}_n).$$

If the expression to the left of the | in the subscript is of the form $S_i \times R_i$, this equation becomes an *individual* conditional interdependency measure. The joint conditional interdependency measure can also be defined for any combination of $\mathcal{V}_i$'s given any combination of different $\mathcal{V}_i$'s, and any combination of $\mathtt{V}_i$'s given any combination of different $\mathtt{V}_i$'s such as

$$P_{S_2 \times S_5 \times R_2 \times R_4 | S_1 \times S_4 \times R_1 \times R_5}(\mathcal{V}_2 \times \mathcal{V}_5 \times \mathtt{V}_2 \times \mathtt{V}_4 | \mathcal{V}_1 \times \mathcal{V}_4 \times \mathtt{V}_1 \times \mathtt{V}_5).$$

Traditionally in satisficing, all work is done within the action space, instead of a derived space as in probability. I will follow tradition. However, one could make a case for working within a derived space in some cases. For example, sometimes one cannot control what action will result from a certain decision. For example, I may decide to take a step forward, but end up slipping and falling instead. Or perhaps I plan to say one thing but end up saying another, either as a mistake, or on purpose in order to adapt to the moment. In general, one of several options in the action space may result from a decision. In such cases, it would make sense to define mapping functions and work within a derived space that represents the decisions one can make.

For any v_i, the individual selectability mass function, or just selectability, p_{S_i}, is defined as

$$p_{S_i}(v_i) \equiv P_{S_i}(\{v_i\}).$$

Since preference mass functions are basically preference measures of singleton sets, the same three constraints apply to preference mass functions as they do to preference measures. For any $\mathtt{v}_i$, the individual rejectability mass function, or simply rejectability, p_{R_i}, is defined as

$$p_{R_i}(\mathtt{v}_i) \equiv P_{R_i}(\{\mathtt{v}_i\}).$$

For any v_i and $\mathtt{v}_i$, the individual interdependency mass function, or simply interdependency, $p_{S_iR_i}$, is defined as

$$p_{S_iR_i}(v_i, \mathtt{v}_i) \equiv P_{S_i \times R_i}(\{v_i\} \times \{\mathtt{v}_i\}).$$

Selectability, rejectability and interdependency mass functions also range between 0 and 1. They are primarily for convenience when dealing with singleton sets. I

will use the mass function notation for the rest of this thesis since I only consider singleton sets. For any v_1, v_2,...v_n, the joint selectability mass function, or simply joint selectability, $p_{S_1S_2...S_n}$, is defined as

$$p_{S_1S_2...S_n}(v_1, v_2, ...v_n) \equiv P_{S_1 \times S_2 \times ... S_n}(\{v_1\} \times \{v_2\} \times ...\{v_n\}).$$

The joint selectability may also be defined for any combination of v_i's such as

$$p_{S_6S_8S_{10}}(v_6, v_8, v_{10}) \equiv P_{S_6 \times S_8 \times S_{10}}(\{v_6\} \times \{v_8\} \times \{v_{10}\}).$$

For any $\mathtt{v}_1$, $\mathtt{v}_2$,...$\mathtt{v}_n$, the joint rejectability mass function, or simply joint rejectability, $p_{R_1R_2...R_n}$, is defined as

$$p_{R_1R_2...R_n}(\mathtt{v}_1, \mathtt{v}_2, ...\mathtt{v}_n) \equiv P_{R_1 \times R_2 \times ... R_n}(\{\mathtt{v}_1\} \times \{\mathtt{v}_2\} \times ...\{\mathtt{v}_n\}).$$

The joint rejectability may also be defined for any combination of $\mathtt{v}_i$'s such as

$$p_{R_6R_8R_{10}}(\mathtt{v}_6, \mathtt{v}_8, \mathtt{v}_{10}) \equiv P_{R_6 \times R_8 \times R_{10}}(\{\mathtt{v}_6\} \times \{\mathtt{v}_8\} \times \{\mathtt{v}_{10}\}).$$

For any v_1, v_2,...v_n and $\mathtt{v}_1$, $\mathtt{v}_2$,...$\mathtt{v}_n$, the joint interdependency mass function, or simply joint interdependency, $p_{S_1...S_nR_1...R_n}$, is defined as

$$p_{S_1...S_nR_1...R_n}(v_1, ...v_n, \mathtt{v}_1, ...\mathtt{v}_n) \equiv P_{S_1 \times ... S_n \times R_1 \times ... R_n}(\{v_1\} \times ...\{v_n\} \times \{\mathtt{v}_1\} \times ...\{\mathtt{v}_n\}).$$

The joint interdependency may also be defined for any combination of v_i's and $\mathtt{v}_i$'s such as

$$p_{S_2S_4R_1R_4}(v_2, v_4, \mathtt{v}_1, \mathtt{v}_4) \equiv P_{S_2 \times S_4 \times R_1 \times R_4}(\{v_2\} \times \{v_4\} \times \{\mathtt{v}_1\} \times \{\mathtt{v}_4\}).$$

For any $v_1, ...v_j$ given $v_{j+1}, ...v_n$, the joint conditional selectability mass function, or simply conditional selectability, $p_{S_1...S_j|S_{j+1}...S_n}$, is defined as

$$p_{S_1...S_j|S_{j+1}...S_n}(v_1, ...v_j|v_{j+1}, ...v_n) \equiv P_{S_1 \times ... S_j|S_{j+1} \times ... S_n}(\{v_1\} \times ...\{v_j\}\,|\{v_{j+1}\} \times ...\{v_n\}).$$

If there is only one element to the left of the |, this equation becomes an *individual* conditional selectability. The joint conditional selectability may also be defined for any combination of v_i's given any combination of different v_i's such as

$$p_{S_1S_3S_5|S_4}(v_1, v_3, v_5|v_4) \equiv P_{S_1\times S_3\times S_5|S_4}(\{v_1\} \times \{v_3\} \times \{v_5\} \,|\{v_4\}).$$

For any $\mathtt{v}_1$,...$\mathtt{v}_j$ given $\mathtt{v}_{j+1}$,...$\mathtt{v}_n$, the joint conditional rejectability mass function, or simply joint conditional rejectability, $p_{R_1\ldots R_j|R_{j+1}\ldots R_n}$, is defined as

$$p_{R_1\ldots R_j|R_{j+1}\ldots R_n}(\mathtt{v}_1, \ldots \mathtt{v}_j|\mathtt{v}_{j+1}, \ldots \mathtt{v}_n) \equiv P_{R_1\times\ldots R_j|R_{j+1}\times\ldots R_n}(\{\mathtt{v}_1\}\times\ldots\{\mathtt{v}_j\}|\{\mathtt{v}_{j+1}\}\times\ldots\{\mathtt{v}_n\}).$$

If there is only one element to the left of the |, this equation becomes an *individual* conditional rejectability. The joint conditional rejectability may also be defined for any combination of $\mathtt{v}_i$'s given any combination of different $\mathtt{v}_i$'s such as

$$p_{R_1R_3R_5|R_4}(\mathtt{v}_1, \mathtt{v}_3, \mathtt{v}_5|\mathtt{v}_4) \equiv \; P_{R_1\times R_3\times R_5|R_4}(\{\mathtt{v}_1\} \times \{\mathtt{v}_3\} \times \{\mathtt{v}_5\} \,|\{\mathtt{v}_4\}).$$

For any $v_1 \ldots v_j$,$\mathtt{v}_1$,...$\mathtt{v}_k$ given v_{j+1},...v_n,$\mathtt{v}_{k+1}$,...$\mathtt{v}_n$, the joint conditional interdependency mass function, or simply joint conditional interdependency, $p_{S_1\ldots S_jR_1\ldots R_k|S_{j+1}\ldots S_nR_{k+1}\ldots R_n}$, is defined as

$$p_{S_1\ldots S_jR_1\ldots R_k|S_{j+1}\ldots S_nR_{k+1}\ldots R_n}(v_1, \ldots v_j, \mathtt{v}_1, \ldots \mathtt{v}_k|v_{j+1}, \ldots v_n, \mathtt{v}_{k+1}, \ldots \mathtt{v}_n) \equiv$$

$$P_{S_1\times\ldots S_j\times R_1\times\ldots R_k|S_{j+1}\times\ldots S_n\times R_{k+1}\times\ldots R_n}(\{v_1\} \times \ldots\{v_j\} \times \{\mathtt{v}_1\} \times \ldots\{\mathtt{v}_k\} \,|$$

$$\{v_{j+1}\} \times \ldots\{v_n\} \times \{\mathtt{v}_{k+1}\} \times \ldots\{\mathtt{v}_n\}).$$

If there is only one element to the left of the |, this equation becomes an *individual* conditional interdependency. The joint conditional interdependency may also be

defined for any combination of v_i's given any combination of different v_i's, and any combination of $\mathtt{v}_i$'s given any combination of different $\mathtt{v}_i$'s such as

$$p_{S_1S_3R_1|S_2R_2R_4}(v_1, v_3, \mathtt{v}_1|v_2, \mathtt{v}_2, \mathtt{v}_4) \equiv$$

$$P_{S_1\times S_3\times R_1|S_2\times R_2\times R_4}(\{v_1\} \times \{v_3\}) \times \{\mathtt{v}_1\} \,|\{v_2\} \times \{\mathtt{v}_2\} \times \{\mathtt{v}_4\}).$$

According to the chain rule of probability, for any v_1, v_2,...v_n, the joint selectabilities may be obtained from the marginal and conditional selectabilities in this way

$$p_{S_1S_2...S_n}(v_1, v_2, ...v_n) =$$

$$p_{S_n|S_{n-1}S_{n-2}...S_1}(v_n|v_{n-1}v_{n-2}, ...v_1)...p_{S_3|S_2S_1}(v_3|v_2, v_1)p_{S_2|S_1}(v_2|v_1)p_{S_1}(v_1).$$

Any S_i along with its corresponding v_i on the right hand can be switched with any other S_i along with its corresponding v_i and obtain a mathematically correct equation. Thus, there are many combinations possible. However, some combinations are more naturally motivated by the situation. Furthermore, these structures all assume maximum selectability dependence. In many cases, such dependence will not occur, and the structure is simplified to exclude dependencies that do not exist. For example, it might reduce to

$$p_{S_1S_2...S_n}(v_1, v_2, ...v_n) = p_{S_n|S_3S_1}(v_n|v_3, v_1)...p_{S_3}(v_3)p_{S_2|S_1}(v_2|v_1)p_{S_1}(v_1).$$

In the extreme case, if there is no interdependence, the joint selectability reduces to

$$p_{S_1S_2...S_n}(v_1, v_2, ...v_n) = p_{S_n}(v_n)...p_{S_3}(v_3)p_{S_2}(v_2)p_{S_1}(v_1).$$

A joint selectability may also be obtained for any combination of v_i's with their simplified interdependencies such as

$$p_{S_7S_5S_3}(v_7, v_5, v_3) = p_{S_7|S_5}(v_7|v_5)p_{S_5|S_3}(v_5|v_3)p_{S_3}(v_3).$$

For any $\mathtt{v}_1$, $\mathtt{v}_2$,...$\mathtt{v}_n$, the joint rejectabilities may be obtained from the individual and conditional rejectabilities in the same way as were the joint selectabilities above (simply replace S with R and v with $\mathtt{v}$). For any v_1, v_2,...v_n and $\mathtt{v}_1$, $\mathtt{v}_2$,...$\mathtt{v}_n$, the joint interdependencies may be obtained from the individual and conditional interdependencies in this way

$$p_{S_1,S_2,...S_nR_1,R_2,...R_n}(v_1, v_2, ...v_n, \mathtt{v}_1, \mathtt{v}_2, ...\mathtt{v}_n) =$$

$$p_{S_n|\_\_\_}(v_n|\_\_\_)...p_{S_3|\_\_\_}(v_3|\_\_\_)p_{S_2|\_\_\_}(v_2|\_\_\_)p_{S_1|\_\_\_}(v_1|\_\_\_)\cdot$$

$$p_{R_n|\_\_\_}(\mathtt{v}_n|\_\_\_)...p_{R_3|\_\_\_}(\mathtt{v}_3|\_\_\_)p_{R_2|\_\_\_}(\mathtt{v}_2|\_\_\_)p_{R_1|\_\_\_}(\mathtt{v}_1|\_\_\_)$$

where the appropriate dependencies (either rejectabilities or selectabilities) are inserted in place of ___. To make use of this equation, the dependencies must be limited as follows: if a selectability or rejectability depends on another selectability or rejectability, the reverse cannot be true. For example, if $p_{S_1}(v_1)$ depends on $p_{R_6}(\mathtt{v}_6)$, then $p_{R_6}(\mathtt{v}_6)$ cannot also depend on $p_{S_1}(v_1)$. As another example, if $p_{S_3}(v_3)$ depends on $p_{S_2}(v_2)$ which depends on $p_{R_1}(\mathtt{v}_1)$, then $p_{R_1}(\mathtt{v}_1)$ cannot also depend on $p_{S_3}(v_3)$. A joint interdependency may also be obtained for any combination of v_i's and $\mathtt{v}_i$'s with their simplified dependencies such as

$$p_{S_4S_8R_2R_1R_8}(v_4, v_8, \mathtt{v}_2, \mathtt{v}_1, \mathtt{v}_8) =$$

$$p_{S_8|R_2R_1}(v_8|\mathtt{v}_2\mathtt{v}_1)p_{R_2|S_4}(\mathtt{v}_2|v_4)p_{S_4}(v_4)p_{R_1|R_8}(\mathtt{v}_1|\mathtt{v}_8)p_{R_8}(\mathtt{v}_8).$$

According to the total law of probability, the individual selectability of v_i can be obtained from the joint interdependency of v_1, v_2,...v_n and $\mathtt{v}_1$, $\mathtt{v}_2$,...$\mathtt{v}_n$ in this way

$$p_{S_i}(v_i) = \sum_{v_1} ... \sum_{v_{i-1}} \sum_{v_{i+1}} ... \sum_{v_n} \sum_{\mathtt{v}_1} ... \sum_{\mathtt{v}_n} p_{S_1...S_nR_1...R_n}(v_1, ...v_n, \mathtt{v}_1, ...\mathtt{v}_n).$$

The individual rejectability of $\mathtt{v}_i$ can be obtained from the joint interdependency of v_1, v_2,...v_n and $\mathtt{v}_1$, $\mathtt{v}_2$,...$\mathtt{v}_n$ as follows

$$p_{R_i}(\mathtt{v}_i) = \sum_{v_1} ... \sum_{v_n} \sum_{\mathtt{v}_1} ... \sum_{\mathtt{v}_{i-1}} \sum_{\mathtt{v}_{i+1}} ... \sum_{\mathtt{v}_n} p_{S_1...S_nR_1...R_n}(v_1, ...v_n, \mathtt{v}_1, ...\mathtt{v}_n).$$

The individual interdependency of v_i and $\mathtt{v}_i$ can be obtained from the joint interdependency of v_1, v_2,...v_n and $\mathtt{v}_1$, $\mathtt{v}_2$,...$\mathtt{v}_n$ in this way

$$p_{S_iR_i}(v_i, \mathtt{v}_i) = \sum_{v_1} ... \sum_{v_{i-1}} \sum_{v_{i+1}} ... \sum_{v_n} \sum_{\mathtt{v}_1} ... \sum_{\mathtt{v}_{i-1}} \sum_{\mathtt{v}_{i+1}} ... \sum_{\mathtt{v}_n} p_{S_1...S_nR_1...R_n}(v_1, ...v_n, \mathtt{v}_1, ...\mathtt{v}_n).$$

Similarly, the joint selectability of v_1,...v_j can be obtained from the joint interdependency of v_1, v_2,...v_n and $\mathtt{v}_1$, $\mathtt{v}_2$,...$\mathtt{v}_n$ as follows

$$p_{S_1...S_j}(v_1, ...v_j) = \sum_{v_{j+1}} ... \sum_{v_n} \sum_{\mathtt{v}_1} ... \sum_{\mathtt{v}_n} p_{S_1...S_nR_1...R_n}(v_1, ...v_n, \mathtt{v}_1, ...\mathtt{v}_n).$$

A joint selectability may also be obtained for any combination of v_i's from a joint interdependency that contains these and other v_i's and $\mathtt{v}_i$'s such as

$$p_{S_2S_4}(v_2, v_4) = \sum_{v_1} \sum_{\mathtt{v}_3} p_{S_1S_2S_4R_3}(v_1, v_2, v_4, \mathtt{v}_3).$$

The joint rejectability of $\mathtt{v}_1$,...$\mathtt{v}_j$ can be obtained from the joint interdependency of v_1, v_2,...v_n and $\mathtt{v}_1$, $\mathtt{v}_2$,...$\mathtt{v}_n$ as such

$$p_{R_1...R_j}(\mathtt{v}_1, ...\mathtt{v}_j) = \sum_{v_1} ... \sum_{v_n} \sum_{\mathtt{v}_{j+1}} ... \sum_{\mathtt{v}_n} p_{S_1...S_nR_1...R_n}(v_1, ...v_n, \mathtt{v}_1, ...\mathtt{v}_n).$$

A joint rejectability may also be obtained for any combination of $\mathtt{v}_i$'s from a joint interdependency that contains these and other v_i's and $\mathtt{v}_i$'s such as

$$p_{R_2R_4}(\mathtt{v}_2, \mathtt{v}_4) = \sum_{v_1} \sum_{\mathtt{v}_3} p_{S_1R_2R_3R_4}(v_1, \mathtt{v}_2, \mathtt{v}_3, \mathtt{v}_4).$$

The joint interdependency of v_1,...v_j and $\mathtt{v}_1$,...$\mathtt{v}_k$ can be obtained from the joint interdependency of v_1, v_2,...v_n and $\mathtt{v}_1$, $\mathtt{v}_2$,...$\mathtt{v}_n$ as such

$$p_{S_1...S_jR_1...R_k}(v_1, ...v_j, \mathtt{v}_1, ...\mathtt{v}_k) = \sum_{v_{j+1}} ... \sum_{v_n} \sum_{\mathtt{v}_{k+1}} ... \sum_{\mathtt{v}_n} p_{S_1...S_nR_1...R_n}(v_1, ...v_n, \mathtt{v}_1, ...\mathtt{v}_n).$$

A joint interdependency may also be obtained for any combination of v_i's and $\mathtt{v}_i$'s from a joint interdependency that contains these and other v_i's or $\mathtt{v}_i$'s such as

$$p_{S_2R_4}(v_2, \mathtt{v}_4) = \sum_{v_1} \sum_{v_4} \sum_{\mathtt{v}_5} p_{S_1S_2S_4R_4R_5}(v_1, v_2, v_4, \mathtt{v}_4, \mathtt{v}_5).$$

2.3 Satisficing Examples

Three satisficing examples are presented to help solidify satisficing in the reader's mind. Example 1 illustrates use of the chain law of probability *without* dependencies. Example 2 illustrates use of the chain law of probability *with* dependencies. Example 3 illustrates use of the total law of probability.

2.3.1 Example 1

The following is an example of calculating the joint utility for the case when the options are selectively and rejectively independent: Say a husband and wife are trying to decide whether to put their tax refund towards some new furniture or a vacation. Let the selectability be the amount of enjoyment received from each option. Say for

the husband, the selectability of the vacation is .7 and .3 for the furniture. Say for the wife, the selectability of the vacation is .25 and .75 for the furniture. Assuming for now that the boldness factor is 0, which makes any rejectabilities irrelevant, the joint utility of the vacation, V, is shown below.

$$U_{wife,husband}(V,V) = p_{S_{wife}S_{husband}}(V,V) - bp_{R_{wife}R_{husband}}(V,V)$$

$$= p_{S_{wife}}(V)p_{S_{husband}}(V) - 0$$

$$= .25 \times .7 = .175$$

Similarly, the joint utility of the furniture, F, is shown below.

$$U_{wife,husband}(F,F) = p_{S_{wife}S_{husband}}(F,F) - bp_{R_{wife}R_{husband}}(F,F)$$

$$= p_{S_{wife}}(F)p_{S_{husband}}(F) - 0$$

$$= .75 \times .3 = .225$$

If the husband and wife agree to make a joint decision, they should choose the option with the highest joint utility; thus, they will choose to buy the furniture. Because the wife feels stronger about the furniture than the husband does about the vacation, she will influence the group decision more.

The husband and wife could also have rejectabilities for the options based on a variety of factors that are completely independent from the selectabilities, such as worry about getting hijacked on the vacation or having to maintain the new furniture. Say for the wife, the rejectability of the vacation is .2 and .8 for the furniture. Say for the husband, the rejectability of the vacation is .65 and .35 for the furniture.

Assuming their joint boldness factor is 1, that is, they value the selectabilities and rejectabilities equally, the joint utility of the vacation is shown below.

$$U_{wife,husband}(V,V) = p_{S_{wife}S_{husband}}(V,V) - bp_{R_{wife}R_{husband}}(V,V)$$

$$= p_{S_{wife}}(V)p_{S_{husband}}(V) - bp_{R_{wife}}(V)p_{R_{husband}}(V)$$

$$= .25 \times .7 - 1 \times .2 \times .65 = .045$$

and the joint utility of the furniture is shown below.

$$U_{wife,husband}(F,F) = p_{S_{wife}S_{husband}}(F,F) - bp_{R_{wife}R_{husband}}(F,F)$$

$$= p_{S_{wife}}(F)p_{S_{husband}}(F) - bp_{R_{wife}}(F)p_{R_{husband}}(F)$$

$$= .75 \times .3 - 1 \times .8 \times .35 = -.055$$

If they choose the option with the highest joint utility (the only one that is satisficing in this case) they will choose the vacation.

2.3.2 Example 2

The following is an example of calculating the joint utility when the options are selectively dependent: Consider a wife who is considering whether to buy a dress or some jeans. The husband is planning to take his wife out this weekend and is deciding between the opera and the movies. The wife will be more inclined to buy the dress if they go to the opera because she could wear it there. She is more inclined to buy the jeans if they go to the movies because she could wear them there. Assume that for the wife, the selectability of the dress is .8 and .2 for the jeans given her husband ascribes all his selectability to the opera. Also, assume that for the wife,

the selectability of the jeans is 1 and 0 for the dress given her husband ascribes all his selectability to the movies. Assume that for the husband, the selectability of the opera is .6 and .4 for the movies. Assuming the boldness factor is 0, which makes any rejectabilities irrelevant, the joint utility for the wife choosing the dress, D, and the husband choosing the opera, O, is shown below.

$$U_{wife,husband}(D,O) = p_{S_{wife}S_{husband}}(D,O) - bp_{R_{wife}R_{husband}}(D,O)$$

$$= p_{S_{wife}|S_{husband}}(D|O)p_{S_{husband}}(O) - 0$$

$$= .8 \times .6 = .48$$

The joint utility for the wife choosing the dress and the husband choosing the movies, M, is shown below.

$$U_{wife,husband}(D,M) = p_{S_{wife}S_{husband}}(D,M) - bp_{R_{wife}R_{husband}}(D,M)$$

$$= p_{S_{wife}|S_{husband}}(D|M)p_{S_{husband}}(M) - 0$$

$$= .2 \times .6 = .12$$

The joint utility for the wife choosing the jeans, J, and the husband choosing the movies is shown below.

$$U_{wife,husband}(J,M) = p_{S_{wife}S_{husband}}(J,M) - bp_{R_{wife}R_{husband}}(J,M)$$

$$= p_{S_{wife}|S_{husband}}(J|M)p_{S_{husband}}(M) - 0$$

$$= 1 \times .4 = .4$$

The joint utility for the wife choosing the jeans and the husband choosing the opera is shown below.

$$U_{wife,husband}(J,O) = p_{S_{wife}S_{husband}}(J,O) - bp_{R_{wife}R_{husband}}(J,O)$$

$$= p_{S_{wife}|S_{husband}}(J|O)p_{S_{husband}}(O) - 0$$

$$= 0 \times .6 = 0$$

If they each agree to choose the individual option that produces the highest joint utility, the wife should buy the dress and the husband should take his wife to the opera.

Note that in this example, the wife took into consideration her husband's selectability in forming her selectability but not vice-versa. It would be problematic if the wife's selectability depended on her husband's and her husband's selectability depended on his wife's, for there would be a dependency loop that would continue forever. As an illustration of this, imagine two people, 1 and 2, who are trying to decide what to do. Their conversation might go something like

1: What do you want to do?

2: That depends on what you want to do?

1: That depends on what you want to do dependent upon what I want to do...

It would be an interesting research topic, however, to explore possible convergence theories when there are dependency loops.

The husband and wife could also have a nonzero boldness factor and rejectabilities for each of the options. These rejectabilities could be included in the joint utility equation in a similar way as they were in example 1.

2.3.3 Example 3

The following is an example of calculating the individual utility from the joint utilities: Suppose robot 1 is traveling along and is about to collide with robot 2. Each robot will consider the following options: go straight, S, veer to the left, L, and veer to the right, R. For both robots, let the selectability of each option be proportional to the likelihood of avoiding a collision and the rejectability proportional to the amount of energy a particular option takes. Assume, to avoid having to choose randomly between equal utilities, that each robot is slightly to the left of the other robot as shown in Figure 2.1. Notice that left, right, and straight for a particular robot is from that robot's perspective. The selectabilities of each robot are dependent on the other robot's selectabilities since they both influence the final avoidance or collision. Because there are several joint selectability values, it is better to show the data in a table. Assume the joint selectabilities are as shown in Table 2.1. Assume for robot 1, the rejectabilities are independent with the values shown below.

$$p_{R_1}(L) = .4$$

$$p_{R_1}(R) = .4$$

$$p_{R_1}(S) = .2$$

Given these values and the situation, we wish to determine which direction robot 1 should go.

The joint option with the highest joint selectability at .4 is both robots veering to the left. This makes sense because they are already somewhat to the left of each

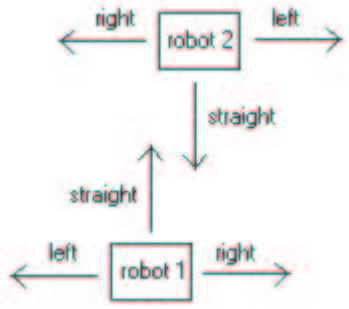

Figure 2.1: Two robots avoiding a collision

Table 2.1: Robot's joint selectabilities

	Robot 2 left	Robot 2 right	Robot 2 straight
Robot 1 left	.4	0	.1
Robot 1 right	0	.3	.05
Robot 1 straight	.1	.05	0

other. Both robots veering to the right is second highest at .3 because they will probably still avoid each other. The lowest joint option at 0 is for both robots to go straight or veer in opposite directions because they would surely collide. One can verify that the other joint selectability values are reasonable given the situation.

Using the total law of probability, for robot 1, the individual selectability of each option is computed as shown below.

$$p_{S_1}(L) = p_{S_1S_2}(L,L) + p_{S_1S_2}(L,R) + p_{S_1S_2}(L,S) = .4 + 0 + .1 = .5$$

$$p_{S_1}(R) = p_{S_1S_2}(R,L) + p_{S_1S_2}(R,R) + p_{S_1S_2}(R,S) = 0 + .3 + .05 = .35$$

$$p_{S_1}(S) = p_{S_1S_2}(S,L) + p_{S_1S_2}(S,R) + p_{S_1S_2}(S,S) = .1 + .05 + 0 = .15$$

For robot 1, assuming b=.5, the individual utilities are computed as shown below.

$$U_1(L) = p_{S_1}(L) - bp_{R_1}(L) = .4 - .5 \times .4 = .2$$

$$U_1(R) = p_{S_1}(R) - bp_{R_1}(R) = .35 - .5 \times .4 = .15$$

$$U_1(S) = p_{S_1}(S) - bp_{R_1}(S) = .15 - .5 \times .2 = .05$$

If robot 1 chooses the option with the highest individual utility, it will veer to the left.

2.4 Summary

An option is satisficing if its selectability is greater than a boldness factor times its rejectability. Options that are not satisficing should be eliminated from the set of options to choose from. If one must choose a single option, it makes sense to choose the option with the highest utility from the satisficing set. When making joint decisions, joint options that are not satisficing should be eliminated from the set of joint options to choose from. If the group must choose a single option, it makes sense that each individual choose the option that produces the highest joint utility from the satisficing set.

When making decisions in the context of a group, the selectabilities and rejectabilities of an agent or group of agents may depend on selectabilities and rejectabilities of other agents. When these dependencies exist, or when deriving joint preferences from individual preferences, or vice versa, the laws of probability provide an excellent framework for modeling preferences.

Chapter 3

Satisficing Applied to Soccer Agents

The test bed I chose to implement satisficing is simulated soccer. Its initial appeal for me was that it allows for the study of both cooperative and competitive agents. Using the RoboCup platform (described shortly), each agent is autonomous, meaning there is no central control. This platform embraces many real world complexities such as limited senses and stamina as well as noisy sensors and actuators, meaning that they do not sense the world exactly as it is nor can they affect the world exactly as intended. Because of this, there are lessons to be learned from simulated soccer that can be applied to other autonomous multi-agent systems.

3.1 Background

The RoboCup soccer server [8] has been used as the basis for successful international competitions [10] and research challenges [6]. Alen Mackworth is credited with recognizing the challenges of this domain. Hiroaki Kitano and Minoru Asada have been the driving force behind the RoboCup competitions. Itsuki Noda wrote

the soccer server and has continued to develop it. With the help of these individuals, both the robotic and simulated soccer community as a whole continues to grow. Kitano's long term goal is to enable the creation of a humanoid robotic soccer team that is capable of playing on a full size field and beating the best human soccer team by the year 2050 [15].

My soccer team was programmed in C++ using some basic lower level building blocks that were released by Peter Stone with the help of Patrick Riley and Manuela Velosso. Their team, named CMUnited99, was the champion at the 1999 Stockholm simulator competition, scoring 13.8 to 0 on average against the 8 teams it played.

The lower-level functions used from Stone's code (presented abstractly) are kick, travel to a point, move to a point, turn body, turn neck, some orientation functions, some functions to determine if an object's position is valid, some sorting functions to determine things such as the closest teammate or opponent to a player, a point, or a line, and some variables to keep track of which play mode the game is in. Some functions were removed in this release such as *SmartDribbleTo* (it was declared in the header file but did not exist in the source file), and a function (which I assume used to exist) that chooses between all the various types of kicks. The dribbling function provided worked so poorly that I assume certain optimizations were removed (Stone discusses the learning algorithms he used to obtain various optimizations within CMUnited99 [15]). I ended up using the kick function with low power instead of his dribbling function. I do not know if the other parameters in this release were optimized or not.

Stone developed a fairly sophisticated world state model using confidence levels

Figure 3.1: RoboCup soccer server

and various prediction methods. This model allows for remembering an object's position or predicted position even when it is not currently within view. Through the orientation, validity, and position functions, I had access to this world state model.

Each soccer player is run as a separate process that uses these lower-level functions to send to and receive from the RoboCup server through a sockets protocol. The graphical interface for the RoboCup soccer server is shown in Figure 3.1. The server receives basic commands from the player: *say*, *hear*, *turn*, *turn_neck*, *dash*, *kick*, *catch*, *sense_body*, and *change_view*. It sends back information concerning what can be seen, heard, or sensed. Since each player is run as a separate process, they cannot communicate directly. Instead, they must use *say* and *hear* commands to communicate with each other through the server.

There are also some limitations enforced by the server which reflect real soccer

and other real life applications. These limitations are

1. Speeds over the maximum speed for players and the ball are not allowed.

2. Limited sight up to a certain angle and distance.

3. Limited communication range and capacity.

4. Limited stamina. The lower the stamina, the slower the player goes. If the player's stamina level drops below a certain point, it cannot recover full stamina again.

5. Noise in the actuator parameters, object motion, and visual perception.

6. Only one *turn*, *dash*, *kick*, *catch*, or *change_view* command, two *hear* commands, and three *sense_body* commands can be executed each simulation time step.

3.2 Agent's Options

Now that I've introduced the basic setup, I will describe my team. After getting in initial positions, each player during each simulator step decides whether to

1. **Handle Ball**: This consists of the following suboptions:

   Kick: Kick towards a point.

   Dribble: Dribble towards a point.

2. **Retrieve Ball**: Go to a point along the projected path of the ball.

3. **Intercept Ball**: Go to a point along the projected path of the ball. The difference between this and the previous option will be described later.

4. **Receive Ball**: Go towards a point specified by the kicker to receive the ball.

5. **Defend Goal/Opponent**: Stay in between the ball and own goal, or between the ball and an opponent.

6. **Move to Position**: Move to a position based upon the player number, which contributes towards an overall formation.

7. **Turn Body**: Turn body around until ball is in sight.

8. **Do Nothing**: This recovers stamina.

Out of the satisficing options, the high-level option with the highest utility is chosen first. If the first option is chosen, the suboption with the highest utility is then chosen. Each of these will be explained in detail shortly.

Along with these options, the player turns its neck each simulator step to face the ball if possible (the player is allowed to turn its neck only so far). It is important to know where the ball is at all times.

The rejectability of each option in my implementation is always based on the player's current stamina, except for turning and doing nothing which require no energy, in which case the rejectability is 0. Besides the options with 0 rejectability, the rejectability is initially set to

$$1 - (curStam - recStam)/(maxStam - recStam)$$

If the player's stamina drops below $recStam$, the player will be unable to recover its full stamina. The variable $maxStam$ is the maximum amount of stamina a player can possess. The variable $curStam$ is the player's current level of stamina. The important thing to notice is that the rejectability is inversely proportional to the current stamina. After calculating all the selectabilities and rejectabilities and discarding nonsatisficing options, if an option is satisficing, its rejectability is multiplied by this option's importance factor (explained shortly). This rejectability is not yet normalized, meaning the sum of the rejectabilities over all remaining options do not add to 1. The normalized rejectability is obtained by dividing by the sum of the unnormalized rejectabilities over the remaining options. For the special case where all the unnormalized rejectabilities are 0, dividing by the sum (also equal to 0) is undefined. Therefore, instead of dividing by the sum, the normalized rejectability is set to 1 over the number of remaining options, thus making them all equal.

The way in which the selectabilities are initially set will be described in the upcoming subsections. After calculating all the rejectabilities and selectabilities and discarding nonsatisficing options, if an option is satisficing, its selectability is multiplied by this option's importance factor (explained shortly). This selectability is not yet normalized, meaning the sum of the selectabilities over all remaining options do not add to 1. The normalized selectability is obtained by dividing by the sum of the unnormalized selectabilities over the remaining options. For the special case where all the unnormalized selectabilities are 0, the normalized selectability is set to 1 over the number of remaining options, thus making them all equal.

The reason for using the importance factors mentioned earlier will be described in Section 4.1.5 and 4.1.6. For now, it suffices to say that it is for programming convenience and to allow for the use of a genetic algorithm.

Each option has its own boldness factor such that the rejectability becomes more or less important in the utility function. This is done because certain options are more critical than others, especially options where the player is in a position to take control of the ball. The boldness factor is set to 0 during these critical options, which causes the agent to disregard its rejectability, and hence disregard its stamina level. For the less critical options, however, such as getting into position, it is more important for an agent to recover stamina. The boldness factor is set higher for these noncritical options, which decreases the utility of the option as the stamina level decreases (since this increases the rejectability). The way in which the boldness factors are set is described in the following subsections.

3.2.1 Handle Ball (Kick or Dribble)

If the ball is close enough to be kicked or caught, determined by Stone's functions $BallKickable$ and $BallCatchable$, the selectability is set to 1. The boldness factor of this option is set to 0 since being within kicking distance of the ball is a critical position. Thus, the player will not take into consideration its stamina level for the moment. Players seem to get rid of the ball fast enough that this is not a problem.

If the utility of handling the ball is the highest of all the options, the player will then consider whether to dribble or kick with the following exception: if the ball

is catchable, the goalie will first attempt to catch the ball by calling Stone's function, *goalie_catch*, before choosing to dribble or kick. The boldness factor of kicking and dribbling is set to 0 for the same reason as before.

In determining the kicking selectability, the player considers kicking towards the opponents goal and to each visible teammate. For each visible teammate and the goal, it calls my function, $KickSel$, which returns the selectability of this kick and the best kick point near the receiver or the best kick point along the goal to which it should kick so as to avoid an opponent interception (the way in which this function determines the selectability and best kick point will be explained in detail later in this chapter). After considering each visible teammate and the goal, the kicking selectability is set to the selectability of the kick with the highest selectability. The best kick point for this kick is stored for later use.

The dribbling selectability is set to 1. It can be thought of as a default action when a player has the ball but can't find a good teammate to pass to and does not have a good shot at the opponents goal. My function, $BestKick$, is called to determine the best dribble point (this function will be described in more detail later in this chapter). The best dribble point is the point which best avoids running into opponents within a large angle towards the goal (140 degrees in my implementation), and a small distance away (5 units in my implementation). From now on, I will exclude the comment, "in my implementation", and simply specify the value used in my implementation; for example, instead of "(140 degrees in my implementation)", I will simply write "(140 degrees)".

If the utility of kicking is higher than the utility of dribbling, it will kick the

ball towards the best kick point using Stone's function, *smart_kick_hard_abs*, with the option of being either a hard or a moderate kick. When the kick is hard, the player first pulls the ball back before kicking such that the kick is harder. A hard kick is appropriate when the teammate is far away or for shooting towards the goal. If it is right next to an opponent, however, it will not choose this option because the ball might get intercepted as it pulls the ball back. The function *smart_kick_hard_abs* takes as a parameter the target velocity of this kick, which is the velocity of the ball immediately after being kicked. For shooting, this parameter is set to the maximum ball speed. For passing, this parameter is set to the value returned by my function *KickSpeed*. This function calculates how fast the player should kick the ball such that it is traveling at a certain speed when it reaches the best kick point. I will call this the destination speed. For destination speed, s_d, ball acceleration, a_b (negative), and distance to the best kick point, d, *KickSpeed* returns the value

$$\sqrt{{s_d}^2 - 2a_b d}.$$

This is a rearrangement of a standard physics equation. The destination speed should not be too fast or too slow. If it is too fast, the receiver will not be able to stop the ball. If it is too slow, the ball will probably get intercepted.

If the kick is towards a teammate, it will "say" who it is passing to, at what point it is passing, and the square root of the current time (explained shortly) using Stone's function, *send_message*, with a "say" command. This way, the chosen receiver can begin to go towards the best point well before the ball reaches that point. The square root of the current time is a security measure (any function of the current

time could be used). There is always a possibility that an opponent could hear what the passer said and repeat the same message later on just to confuse my team. With the square root of the current time spoken, the receiver can check to make sure that the heard square root of the current time equals its calculated square root of the current time. If they match, it will know it is from a teammate. If the opponent simply repeats the message later on, the square root of the current time will not match and all players on my team will ignore the message. Of course, if the opponent could figure out that the square root of the current time in included in my messages, it could include it as well and say things that would confuse my team. I assumed, however, that the other teams would not be able to crack my simple code. The teams I played against certainly did not have this capability.

If the utility of dribbling is higher than the utility of kicking, the player dribbles towards the best dribble point using Stone's function, *smart_kick_hard_abs* with the target velocity set to a relatively low value (0.65 units/step). This velocity value is increased a bit the further away the closest opponent is to the best dribble point, for the player should dribble as fast as it is safe.

3.2.2 Retrieve Ball

For the goalie, the selectability of retrieving the ball is set to 1 if the ball is in the goalie box. Otherwise, it is set to 0. For other players, the selectability of retrieving the ball is set to 1 if this player is the closest teammate to the ball, unless it is further away from its goal than is the opponent with the ball; otherwise, it is set to 0. If this exception is met, the next closest player is considered, and so forth.

This exception helps eliminate the situation where a player follows behind a dribbling opponent and is unable to steal the ball away. My function, *InterceptSel* is called to determine the best retrieve point along the projected path of the ball (this function is described in detail later in this chapter). The boldness factor of this option is set to 0, the same as for kicking the ball, since being in a position to retrieve the ball is critical; for if no player is retrieving the ball, the opponent will have no problem dribbling towards the player's goal and scoring.

If the utility of retrieving the ball is the highest, the player will go towards the best retrieve point using Stone's function, *test_go_to_point*, at the maximum possible speed (1 unit/step assuming it has maximum stamina).

3.2.3 Intercept Ball

InterceptSel is called again to determine the best intercept point along the projected path of the ball and the likelihood of being able to intercept the ball. The selectability of intercepting the ball is set to the likelihood of interception. The boldness factor of this option is set to 0 since being in a position to intercept is critical.

If the utility of intercepting the ball is the highest, the player will go towards the best intercept point using Stone's function, *test_go_to_point*, at the maximum possible speed.

3.2.4 Receive Ball

If the player receives a message from the teammate with the ball saying that the ball will be passed to it, the selectability of going towards the receive point

specified by the teammate with the ball is set to 1. Otherwise, it is set to 0. The selectability will remain at this value for the predicted amount of time it takes for the player to go to the point. This is a critical position to be in, so the boldness factor of this option is set to 0. The boldness factor has been set to 0 for all the options that have been mentioned so far. Some of the options to follow will have nonzero boldness factors since these options are not as critical. I did try nonzero boldness factors for this and previous options, but performance tended to decrease.

If the utility of receiving the ball is the highest, the player will go towards the receive point specified by the passer using Stone's function, *test_go_to_point*, at the maximum possible speed (2.7 units/step).

Note that since there is actuator noise, the passer may not pass the ball exactly to the point it specified. If the receiver goes to the receive point and the ball goes somewhere else, the player will miss the ball. It therefore needs the ability to decide when it will drop the original plan to receive at the point specified by the passer, and instead, go to the best intercept point calculated previously. The interception point will be a more accurate point at which to go for the ball than the receive point because it is determined from the actual position, speed, and deceleration of the ball, whereas the receive point was only an estimate by the passer of the point to which it would try to kick the ball. Nevertheless, if the receiver initially moves towards the receive point specified by the passer as soon as it receives the message, the receiver will have an advantage over the opponent since it is able to move nearer to the correct point even before it knows it is in a good position to intercept. My players have the ability to drop the plan to receive and take up the plan to intercept assuming the

importance factor of intercepting is higher than that of receiving.

3.2.5 Defend Goal/Opponent

The goalie will defend its goal by positioning itself near the goal between the ball and the center of the goal. The selectability for defending the goal for the goalie is always set to 1. The boldness factor of this option is proportional to the distance from the ball, such that the closer the ball is, the more important it is to ignore its stamina for the moment in defending the goal, and the further away the ball is, the less important it is to defend the goal and the more important it is to preserve stamina.

All other players, except for the two most forward players, consider defending an opponent if they are between the ball and their goal along the length axis, and there is an opponent reasonably close to the ball to make defending worthwhile. If so, the player considers guarding each visible opponent, starting with the closest. The closest opponent that does not meet any of the following conditions becomes the tentative opponent to guard:

1. The opponent is the closest opponent to the ball (the retrieving player covers this opponent).

2. The opponent is already being guarded by another teammate.

3. The opponent is over a certain distance away such that the player is not likely to get there in time to make a difference.

The selectability of guarding the chosen opponent, if any, is set to 1. If no opponent is chosen, the selectability is set to 0. The boldness factor of this option is proportional to the distance from the ball such that the closer the ball is, the more important it is to ignore its stamina for the moment in defending the opponent, and the further away the ball is, the less important it is it defend the opponent and the more important it is to preserve stamina. The best defend point is calculated as a point between the ball and the opponent slightly in front of the opponent.

If a defending player number 2, 3 or 4 has not yet been assigned an opponent to guard, it will help the goalie defend the goal, in which case, the selectability is set to 1. My function *BestKick* is then called from the point of view of the opponent with the ball to determine where the opponent is likely to shoot the ball along the goal. The best defend point is calculated as a point in between the distance from the ball to the point at which the opponent is likely to shoot. This point is calculated as a fraction of this distance (.65) along the path that the ball is predicted to travel. The boldness factor of this option is determined in the same way as for the goalie.

If the utility of defending is the highest, the player will move to the best defend point using Stone's function, *test_go_to_point*.

3.2.6 Move to Position

Each player is given an ideal position based on its uniform number, creating an overall formation for the team which can be seen in Figure 3.2. If this option is chosen, a player may move within a rectangular area around this ideal position; it may move approximately 1/4 the length of the field left and right and 1/4 the width of the

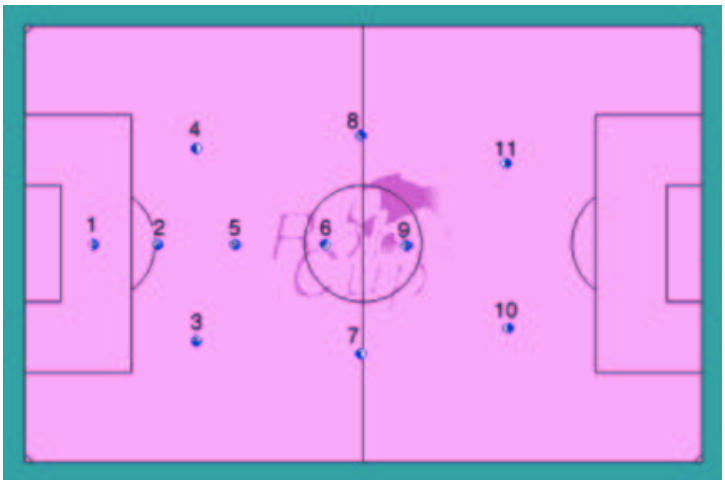

Figure 3.2: Ideal positions

field up or down. Players calculate the best position to be in such that they maintain their relative positions to each other and the team as the whole hovers around the ball as can be seen in Figure 3.3, where my players have a darker shade. They do so as follows: each player is assigned an ideal position, which is some horizontal x and vertical y offset from the origin. The best position point is calculated as the position of the ball plus the ideal offset. This dynamic type of positioning can be very beneficial; if the players have fixed positions throughout the game, they have to be spread out more to cover the field. Because the distance between players is increased, the probability of an interception by the other team is increased.

If the best position point is within the player's allowed rectangle, the selectability of moving to position is set to a distance factor; otherwise, it is set to 0. The distance factor is proportional to the distance from the player's current position to the player's best position point such that the selectability of moving to position increases

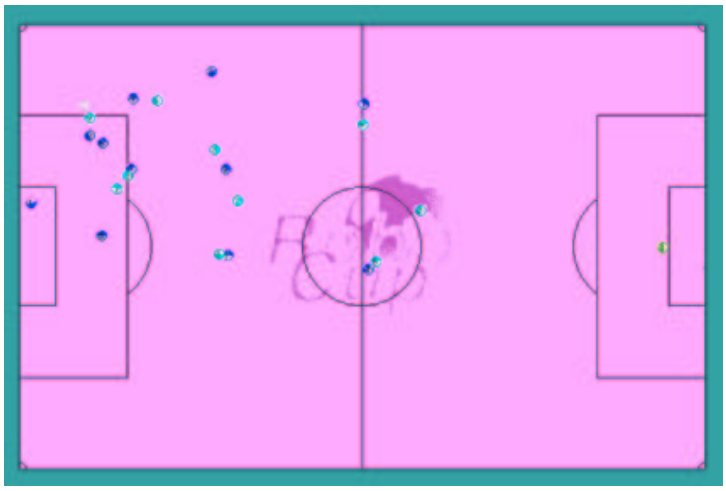

Figure 3.3: Formation hovering around the ball

the further away the player is from the best position point.

The boldness factor of this option is set to a medium value (0.5), thus decreasing the utility of positioning with increasing rejectability. Without this fairly high boldness factor, this option is called too frequently; it is called even when a player's stamina is low, thus decreasing its stamina to the extent that the player becomes slow and ineffective. Of course, if the boldness factor is set too high, the rejectability will have too much of an influence and this option will hardly ever be called.

The goalie is an exception to this positioning scheme. The only positioning the goalie does is to move to a point along the outer goal box (which it can do instantaneously) once it has caught the ball, thus moving the ball away from the goal to a safer position. The goalie considers several points along the outer goal box and chooses the point which is furthest from an opponent for the best position point. If the goalie has caught the ball, the selectability of moving is set to 1. Otherwise, it is

set to 0. The boldness factor in this case is set to 0, since moving the ball away from the goal is a critical move.

If the utility of moving to position is the highest, the player will move towards the best position point using Stone's *move* function if the player is the goalie, and his *test_go_to_point* function for all other players. For the second case, they travel at a medium speed (0.75 units/step) to conserve energy.

3.2.7 Turn Body

If Stone's function, $BallPositionValid$, returns false, the selectability of turning its body is set to 1. Otherwise, it is set to 0. The value of the boldness factor is irrelevant since the rejectability of this option is 0 (since this option requires no energy).

If the utility of this option is the highest, the player will take one turn counterclockwise using Stone's function, *turn*. It turns by an amount equal to the angle of visibility imposed by the server (60 degrees).

3.2.8 Do Nothing

The selectability of doing nothing is equal to

$$1 - (curStam - recStam)/(maxStam - recStam).$$

The main benefit of doing nothing is to recover stamina. The value of the boldness factor is irrelevant since the rejectability of this option is 0 (since this option requires no energy).

If the utility of this option is the highest, the player will not send any messages to the server (besides turning its neck to face the ball, which it does regardless of the chosen option).

3.3 My Functions

The functions $KickSel$, $InterceptSel$, and $BestKick$ were mentioned previously. Each one will be described in detail.

3.3.1 KickSel

$KickSel$ function returns the kicker's individual selectability for kicking to a receiver or the goal, and the best point at which to kick near the receiver or along the goal line. This frequently called function is the only part of the program that uses the probabilistic extensions of satisficing that take the likely actions of teammates and opponents into consideration. The best kick point is obtained from calling my function, $BestKick$. It is used in calculations within the function and returned at the end of the function. The nearest opponent, o, to the line between the ball and the best kick point is assumed to have two options: intercept, I, and not intercept, I^c. The receiver, r, which is passed into this function, also has two options: receive, R, and not receive, R^c. The goal may be thought of as a receiver where the selectability of receiving is 1 and the selectability of not receiving is 0. We wish to determine the selectability of kicking, K, for the kicker, k. This is dependent upon the opponent's and receiver's selectability values. The selectability of kicking will be calculated in this way: First, derive the joint selectabilities from the kicker's conditional selectability

values and the receiver's and opponent's individual selectability values using the chain law of probability. Second, derive the kicker's individual selectability of kicking from the joint selectability values using the total law of probability. The following values must first be found

1. $p_{S_o}(I)$: The selectability of intercepting the ball from the opponent's view.
2. $p_{S_o}(I^c)$: The selectability of not intercepting the ball from the opponent's view.
3. $p_{S_r}(R)$: The selectability of receiving the ball from the receiver's view.
4. $p_{S_r}(R^c)$: The selectability of not receiving the ball from the receiver's view.
5. $p_{S_k|S_oS_r}(K|I,R)$: The selectability of the kick (from the kicker's view) given the opponent intercepts and the receiver receives.
6. $p_{S_k|S_oS_r}(K|I^c,R)$: The selectability of the kick (from the kicker's view) given the opponent does not intercept and the receiver receives.
7. $p_{S_k|S_oS_r}(K|I,R^c)$: The selectability of the kick (from the kicker's view) given the opponent intercepts and the receiver does not receive.
8. $p_{S_k|S_oS_r}(K|I^c,R^c)$: The selectability of the kick (from the kicker's view) given the opponent does not intercept and the receiver does not receive.

The term $p_{S_o}(I)$ is set to the likelihood of intercepting returned by my function, $InterceptSel$, from the opponent's point of view. The term $p_{S_o}(I^c)$ is set to $1-p_{S_o}(I)$. The term $p_{S_r}(R)$ is set to the likelihood of intercepting returned by $InterceptSel$ from the receiver's point of view. The term $p_{S_r}(R^c)$ is set to $1-p_{S_r}(R)$.

The term $p_{S_k|S_oS_r}(K|I,R)$ is set to 0 since this is an impossible situation. For passing, the three remaining conditional values depend on the distance closer that the best kick point is to the opponents goal and the distance to the best kick point. If the distance closer to the goal is too short, it is not helping the team's cause, which is to get the ball towards the opponents goal. If the distance closer to the goal is too large, either it is too far for the kicker to kick, or, even if the kicker is able to kick that far, it is risky because a little bit of actuator noise can make a big difference at that distance. Therefore, the optimal distance closer to the goal is somewhere in between. For the distance to the best kick point, d, the distance closer to goal, Δd, constants, c_1, c_2, and c_3, one reasonable way to determine $p_{S_k|S_oS_r}(K|I,R^c)$ is as

$$p_{S_k|S_oS_r}(K|I^c,R) = \sin(\pi(\Delta d + c_1)/(c_1+c_2))sin(\pi d/c_3)$$

for $-c_1 < \Delta d < c_2$ and $0 < d < c_3$; otherwise, it is set to 0. This can be seen graphically in Figure 3.4. For most cases, c_1 is set to about 1/20 the length of the field, and c_2 and c_3 to about 1/3 the length of the field. The nearer the player is to the opponent's goal, the more c_1 is increased and c_2 is decreased since the main intent is no longer to get the ball closer to the goal, but rather, to pass around to other players near the goal until a good opportunity to shoot is found.

For shooting, the shorter the distance from the goal, the better, since a shot from a short distance away will arrive at the goal faster, making it harder to intercept, and is more accurate. For the distance to the best kick point (along the goal) d, and constant, c_4, a reasonable way to determine $p_{S_k|S_oS_r}(K|I^c,R)$ is as

$$p_{S_k|S_oS_r}(K|I^c,R) = \cos((\pi/2)(d/c_4)).$$

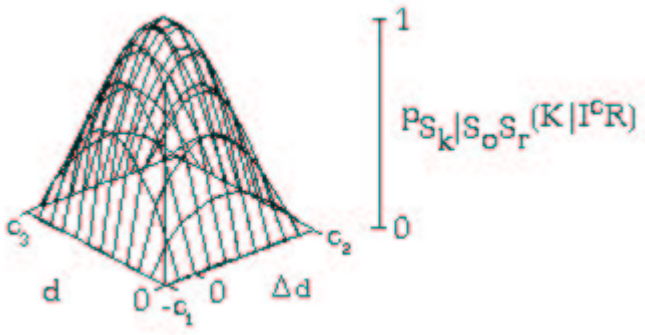

Figure 3.4: Passing selectability

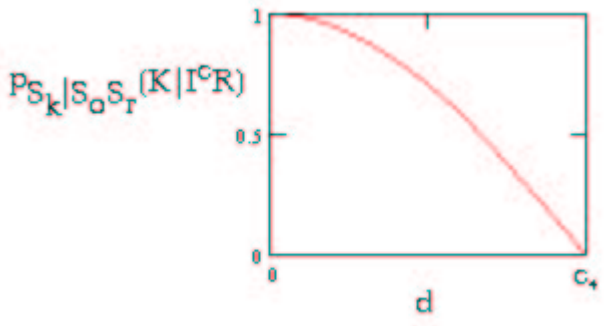

Figure 3.5: Shooting selectability

This can be seen graphically in Figure 3.5. The variable c_4 is set to about 1/3 the length of the field.

Even if the ball is intercepted, as long as it moves closer to the opponent's goal, it is still worth a small amount. Therefore $p_{S_k|S_oS_r}(K|I, R^c)$ is set to $0.1p_{S_k|S_oS_r}(K|I^c, R)$, and $p_{S_k|S_oS_r}(K|I^c, R^c)$ is set to $0.2p_{S_k|S_oS_r}(K|I^c, R)$.

Using the chain law of probability, the joint selectability of all combinations of options are shown below.

$$p_{S_k S_o S_r}(K, I, R) = p_{S_k|S_o S_r}(K|I, R) p_{S_o}(I) p_{S_r}(R) = 0$$

$$p_{S_k S_o S_r}(K, I^c, R) = p_{S_k|S_o S_r}(K|I^c, R) p_{S_o}(I^c) p_{S_r}(R)$$

$$p_{S_k S_o S_r}(K, I, R^c) = p_{S_k|S_o S_r}(K|I, R^c) p_{S_o}(I) p_{S_r}(R^c)$$

$$p_{S_k S_o S_r}(K, I^c, R^c) = p_{S_k|S_o S_r}(K|I^c, R^c) p_{S_o}(I^c) p_{S_r}(R^c)$$

Using the total law of probability, the individual selectability of kicking is

$$p_{S_k}(K) = p_{S_k S_o S_r}(K, I, R) + p_{S_k S_o S_r}(K, I^c, R) + p_{S_k S_o S_r}(K, I, R^c) + p_{S_k S_o S_r}(K, I^c, R^c)$$

$$= p_{S_k S_o S_r}(K, I^c, R) + p_{S_k S_o S_r}(K, I, R^c) + p_{S_k S_o S_r}(K, I^c, R^c).$$

This value is then returned as the selectability of this kick.

3.3.2 BestKick

BestKick is called both from my team's point of view and the opponent's point of view (to predict the action of an opponent when it possesses the ball). It is passed, among other values, a destination point, which can either be the opposing team's goal or a teammate. It returns a kick point near the destination which best avoids the opponents. For passing, the kicker should only consider kicking within a certain angle towards the receiver that allows the receiver to receive the ball. We wish to find this limiting angle, θ_{limit}. If the ball is kicked at a certain angle, it will follow a path which I call the "ball path". The fastest way a receiver can travel to the ball path is to travel perpendicular to it at its maximum speed, $s_{r_{max}}$, for this is the

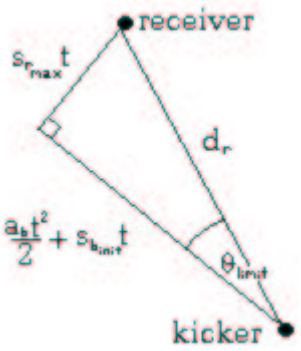

Figure 3.6: Limiting angle

shortest distance. For an amount of time, t_r, the receiver will travel $s_{r_{max}}t_r$ distance units. For an amount of time, t_b, ball acceleration, a_b, and initial ball speed, $s_{b_{init}}$, the ball will travel $a_b(t_b)^2/2 + s_{b_{init}}t_b$ distance units. When the receiver barely gets to the path of the ball in time to receive it, the ball has been kicked at the limiting angle. This occurs when $t_b = t_r$. For time t such that $t = t_b = t_r$, and distance to the receiver, d_r, the situation can be modeled with a right triangle as seen in Figure 3.6. Using the following two geometric equations derived from this right triangle

$$\tan(\theta_{limit}) = s_{r_{max}}t/(a_b t^2/2 + s_{b_{init}}t)$$

$$\sin(\theta_{limit}) = s_{r_{max}}t/d_r$$

we can solve for θ_{limit}.

For passing, if the ball is kicked at θ_{limit} from the receiver's angle, an opponent at an angle as far away as $2\theta_{limit}$ and within a distance of d_r plus a bit, which is denoted d_{limit}, is in a position to intercept the ball. To limit within an angle $2\theta_{limit}$ and distance d_{limit} defines a cone with a curved top (like a piece of pie). All opponents

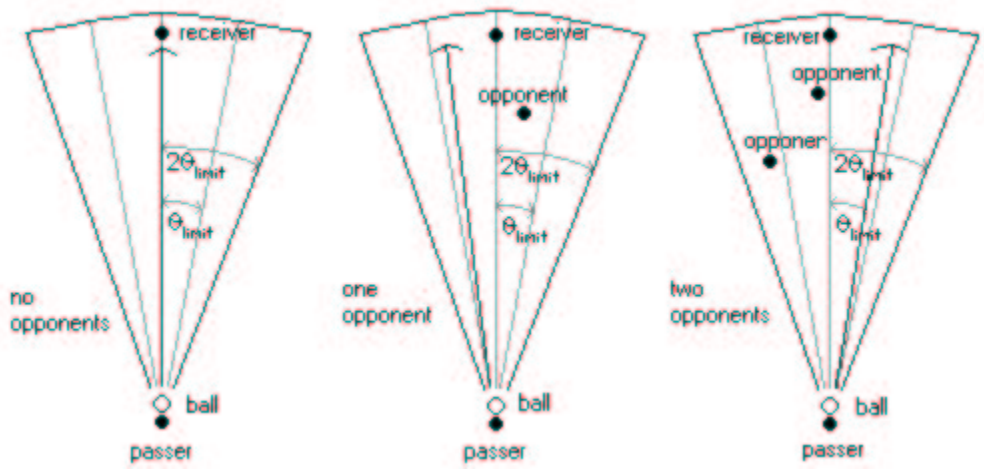

Figure 3.7: Clearest kicking angle examples

within this cone are considered a threat. If there are no opponents within the cone, $BestKick$ returns the receiver's position as the best kick point. If there is one or more opponents within the cone, the kicking angle which best avoids an opponent is determined. Some examples of this angle are shown in Figure 3.7. This angle is determined by first sorting the opponents by angle and determining the largest angle difference between consecutive opponents (as well as between an opponent nearest to a cone edge and that cone edge), and second, calculating the angle in the middle of the largest angle difference. If, in the first step, the middle of the largest angle falls outside the limiting angle, it is decreased such that the middle is at the limiting angle.

For passing, the kick point returned by this function is calculated as the point $.75d_r$ distance units away on this ball path. The reason for the .75 factor is to increase the chances that the receiver will get to the ball before an opponent.

For shooting, the kicker should consider kicking only within an angle that will allow the ball to enter the goal. Therefore, θ_{limit} is the angle from the center of the goal to the edge of the goal and d_{limit} is the distance to the goal. All opponents within an angle of $2\theta_{limit}$ and distance d_{limit} are again considered a threat. The best kicking angle is determined in the same way as for passing, and the returned kick point is the point where the ball path intersects the goal line.

$BestKick$ is also used to determine which way a player should go as it dribbles to best avoid an opponent. In this case, $BestKick$ is passed the center of the goal as the destination point, θ_{limit} is set to some large angle (140 degrees), and d_{limit} is set to a moderate value (10 units).

3.3.3 InterceptSel

$InterceptSel$ calculates the likelihood of intercepting and the best point at which to intercept along the predicted path of the ball. The term "intercept" is used here in the broadest sense such that it applies to any player trying to take possession of the ball, and not just to players stealing the ball from a pass that was meant for an opponent. The best intercept point is the point along the predicted path of the ball such that if the intercepting player goes towards this point, it will arrive there as the ball reaches that point. If the player arrives at a point along the predicted ball path other than the best intercept point, it will either miss the ball or arrive at the point and wait for the ball to come to it, which will take longer and increase the chances of the ball being intercepted by an opponent. The best intercept point thus occurs when the player time to the point, t_p, and the ball time to the point, t_b, are

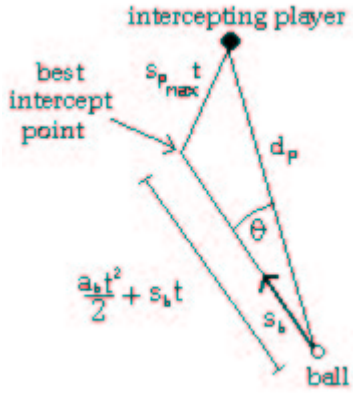

Figure 3.8: Intercepting situation

the same. Given the ball speed, s_b, the ball acceleration, a_b, the player's maximum speed, $s_{p_{max}}$, the distance from the ball to the player, d_p, and the angle difference between the direction of the ball and the angle from the ball to the receiver, θ, we wish to determine the time, t, such that $t = t_b = t_p$. This situation can be modeled with a triangle as shown in Figure 3.8. Using the law of cosines, we have the following

$$a(t)^2 = b(t)^2 + c^2 - 2b(t)c\cos(\theta)$$

where

$$a(t) = s_{p_{max}}t$$

$$b(t) = a_b t^2/2 + s_b t$$

$$c = d_p.$$

If s_b is very low or θ is greater than 90 deg, this function is not equipped to deal with the situation, and thus returns a likelihood of intercepting of 0.

The equation above, when expanded, has the terms t^4, t^2, and t, which makes it difficult to solve for t. Therefore, Newton's method is used to numerically solve for t. For an approximate solution to

$$f(t) \equiv -a(t)^2 + b(t)^2 + c^2 - 2b(t)c\cos(\theta) = 0,$$

Newton's method can be stated as

$$t^{[n+1]} = t^{[n]} - f(t^{[n]})/f'(t^{[n]})$$

where n is an iteration value. Substituting for $a(t)$, $b(t)$, and c, and taking the derivative of $f(t)$, we have

$$f'(t) = {a_b}^2 t^3 + t^2(-2{s_{p_{max}}}^2 + 3a_b s_b) + t(2{s_b}^2 - 2d_p a_b\cos(\theta)) - 2d_p s_b\cos(\theta).$$

$t^{[0]}$ is set to $((-s_b + \sqrt{{s_b}^2 + 4(a_b/2)(d_r/2)}\,)/2)(a_b/2)$, which is the time for the ball to travel distance $d_r/2$. This turns out to be a good starting value for iterating. Starting with $t^{[0]}$, Newton's method goes through a maximum of 10 iterations. If, for some n, $t^{[n+1]} - t^{[n]}$ is below a certain threshold, an approximate t has been found and the iterations cease. If, for some n, $t^{[n+1]} - t^{[n]} > t^{[n]} - t^{[n-1]}$, then $t^{[n]}$ is diverging, which means this method cannot find a solution to t, and the function returns an likelihood of intercepting of 0.

If Newton's method converged, $t \approx t_b \approx t_p$. If it did not, there will be a significant discrepancy between t_p and t_b. This suggests there is something nonideal about this interception. To take this into account, the likelihood should be decreased proportional to this discrepancy. Assume $t_b = t$. The best intercept point is calculated as the predicted position where the ball will be in time t_b. We can solve for t_p as the

distance from the player to the best intercept point divided by $s_{p_{max}}$. For some factor f, the likelihood of intercepting is set to $1 - |f(t_b - t_p)|$, where f is a small number (0.125).

3.4 Summary

My soccer agents make decisions based upon the principles of satisficing. During each simulator step, an agent calculates the utility of each option and chooses the option with the highest utility, thus providing a good mechanism for changing roles. The function, $KickSel$, makes use of the probabilistic extensions of satisficing, thus allowing a player to take into consideration the preferences of its opponents and teammates in forming its own preferences.

I have also described some techniques unrelated to satisficing which lead to more intelligent actions in soccer. These techniques include deciding when to retrieve the ball, using geometrical and convergence equations to determine the likelihood of an interception and at what location to intercept the ball, determining the kick angle to best avoid opponents, coordinating passing and receiving, positioning dynamically, and deciding which opponent to defend.

Chapter 4

Observations from an Implementation of Satisficing

I draw from my experience in developing satisficing code to present six advantages of satisficing. I also present two pitfalls to avoid when using satisficing.

4.1 Advantages of Satisficing

As I programmed my soccer agents, I discovered six advantages of using satisficing over other multi-agent programming approaches. These advantages are:

1. Role Switching
2. Sensitivity
3. Ease of Changing Option's Importance
4. Function Reusability
5. Implementation Simplicity
6. Genetic Algorithm Readiness

These can be applied to autonomous multi-agent systems in general. Each of these advantages will be described in the following subsections.

4.1.1 Role Switching

Often there is a need to switch between different behaviors or roles. In real life, we must switch between different roles such as a worker, student, husband or wife, father or mother, son or daughter, friend, driver, pedestrian, etc. Satisficing provides a straight forward way to switch roles. There has been significant interest in systems that allow for role switching. One such system is fuzzy logic, in which each member of a set has a certain degree of truth, and one member is chosen through a defuzzification process [3]. Another such system is potential fields, where each role is represented as a vector which corresponds to the speed and orientation of the agent, after which the vectors of each role are combined to produce the agent's movement [4]. While satisficing does not necessarily support role switching better than these other systems, one of its inherent advantages is a structure that makes role switching simple.

Some multi-agent systems are thought to be better suited to specialized agents than to agents which switch roles. For example, it is often assumed that workers in a factory line should specialize. It is also commonly accepted that we should specialize in our careers. Many also believe that one can earn a higher ranking in society and thus concentrate on more desirable tasks, leaving the less desirable tasks to others. While I will not deny that specialization is important, the importance of being able to play many roles is often overlooked. Might it prevent the factory worker from getting worn out from doing the same thing repeatedly and help her know how to make her part of the product fit in better with the whole if she switched roles from

time to time? Might clashing be avoided between two individuals from different careers working together if they try to understand each other somewhat? Might higher ranking people be more compassionate if they put themselves in the lower rank positions from time to time?

Initially, my soccer agents were more specialized than they are now. Current players can take upon themselves the role of kicker, retriever, receiver, intercepter, defender, positioner, and turner. For example, originally only the backmost players would defend. I realized after a while, however, that it is advantageous for most of the players, wherever they are on the field, to defend an opponent when it appears that the opponent with the ball might pass to it (assuming the player is in a good position to defend). As a second example, only the player closest to the ball used to go for the ball. I realized after a while, however, that is advantageous for any player to try to intercept if they are in a good position to do so. Thus I came to realize that soccer agents should not be as specialized as I had initially thought they should be.

It would be wise to take a second look at other autonomous multi-agent systems thought to be best handled with specialization, and look for ways that role switching might be advantageous. Satisficing provides a straight forward way of switching roles.

4.1.2 Sensitivity

Using the probabilistic extensions of satisficing, an agent has the ability to consider all the dependencies that exist; thus, it is sensitive to both the likely goals and actions of teammates and the likely goals and actions of the opponents.

Those programming autonomous multi-agent systems without satisficing are probably aware of certain dependencies and implement them in an ad hoc way using various mathematical formulas, but they lack a formal system for implementing them which satisficing provides. If one is using a formal system that describes how to reflect the dependencies, one is more likely to include all the dependencies.

Satisficing is powerful in its ability to allow for sensitivity. But, used incorrectly, this ability can lead to problems. Some of the ways it could be used incorrectly are:

1. Including irrelevant factors mistaken for real dependencies.

2. Weighting a dependency too much or too little.

3. Making poor predictions of other agent's preferences.

4. Considering dependencies which depend too much on future uncertainties.

It is hard to know what the real dependencies are, what the correct weighting of dependencies should be, how much to predict another agent's preferences, and how far to extend the dependencies into an uncertain future. Trial and error may be the only way to discover the right balance. But note that satisficing does not cause these problems. It is simply powerful enough that these inherent problems arise and must be addressed.

Satisficing allows for many or few dependencies. Being flexible in this way is very useful because if one does not have the processing power to calculate or the time

1. if I can't see the ball, turn body
2. else if the ball is kickable, then
3. if the probability of a shoot not being intercepted is greater than ___, shoot.
4. else if the probability of a pass not being intercepted is greater than ___, pass.
5. else dribble.
6. else if the probability of intercepting is greater than ___, go for ball.
7. else if closest to ball, go for ball.
8. else if passer says it will pass to me, go to best pass point.
9. else if the probability that an opponent will be passed to is greater than ___ and my stamina level is above ___, defend that opponent
10. else if stamina level is above ___, move to position
11. else do nothing

Figure 4.1: Hard-coded soccer agent implementation

to program all these dependencies, she can make simplifying assumptions, keeping the more important dependencies and leaving out the less important ones.

4.1.3 Ease of Changing Option's Importance or Priority

When one hard-codes the relative importance or priority of different options, it is difficult to alter them. With the satisficing approach, the task of altering an option's importance or priority becomes simpler. A common programming approach is to hard code the priorities and importances of the options. For example, one might code a soccer player (in pseudocode) as shown in Figure 4.1.

Coding this way leads to several inconveniences when changing the priority or importance of an option. Say the programmer wants to make receiving higher priority than going for the ball. She would have to cut line 8 (or the several lines of code associated with this option) and paste it in front of line 7. Now suppose she wants to make dribbling more important than it is currently is. This time, she cannot simply cut line 5 (or the several lines of code associated with this option) and

paste it above line 4, because then passing would never be called. Instead, to make dribbling more important, she has to make shooting and passing less important by decreasing the values that fill in the blanks in lines 3 and 4. If, however, she wants to make shooting more important than it currently is, she will simply decrease the value that fills in the blank in line 3. Thus there are three ways in which an option is made more or less important in the hard-coded style. It takes time to figure out which way is appropriate and to make the changes without introducing errors.

With the satisficing approach, however, an option is made more or less important or set to a different priority the same way each time: simply increase or decrease the utility of an option. There remains one inconvenience though. Assume one has established a good balance between the selectability and rejectability by choosing the boldness factor appropriately. She may increase the utility of an option by either increasing the selectability or decreasing the rejectability. She may decrease the utility of an option by either decreasing the selectability or increasing the rejectability. To change either the selectability or rejectability alone, however, disrupts the balance that existed between the selectability and rejectability. In Section 4.1.5, I present a convenient method using importance factors which makes it possible to increase or decrease the utility of an option without disrupting the balance between the selectability and rejectability.

4.1.4 Function Reusability

In soccer and in many other multi-agent applications, agents within a team have many of the same goals as their opponents (in opposition of course). If so, it

makes sense that agents use the same functions to guess what their opponents will do since they have many of the same goals. For example, in my code, my functions *InterceptSel* and *KickSel* are called from the point of view of both my players and my opponents.

Satisficing encourages reusing functions for agents within a team and for predicting the actions of the opponent. For, in determining how to assign the preferences of an agent for a given situation, she is likely to go through the following thought process:

1. The preferences of the agent under consideration depend on other agent's preferences.

2. I must determine the way in which other agents form their preferences.

3. I observe that some agents form their preferences identically.

4. I will write functions that are general enough to be used by multiple agents.

A programmer not using satisficing does not need to know the preference values of other agents to write the code for a particular agent, making it less likely she will be thinking about the way other agents act, making it less likely she will implement functions that apply to more than one agent.

The way a function is written will determine if it may be used by one or multiple agents. For example, Stone provides two functions, *GetTeammatePosition*, which determines a teammate's position from its number, and *GetOpponentPosition*, which determines an opponent's position from its number. If a programmer needs to

write a function which requires the position of a player, she can do it one of two ways. The first way is to write the function such that it takes in the number of the player, and then within this function, call *GetTeammatePosition* with the player's number. Doing this, however, makes it impossible to reuse this function for an opponent, which would require the function *GetOpponentPosition* instead. The second option is to write the function such that it takes in the position of the player. This makes it possible to pass in the position of both the teammate and the opponent.

Some may question that we should use the same functions for both teammates and opponents. If one assumes that two different teams that are good tend to make similar choices because they have both grasped some underlying principles of success, then it makes sense to have the same functions for both. This is what I assumed in writing my agents. Assume, however, team A is quite intelligent and plays against team B which does not play as intelligently. Should team A use the same functions for both its teammates and team B's players? Probably not. For example, children playing soccer have been known to swarm around the ball instead of staying open for a pass. Not only that, children cannot kick as far. Therefore, if a professional team were to play against children, (assuming they have no pity on the children and play as competitively as possible) they should not predict the children's behavior by the way they would themselves behave. They would want to predict fewer and shorter passes. Therefore, if playing a less intelligent or less capable team, one should either use completely different predicting functions or adjust certain values (which are somehow learned by observation) within the reusable functions, such as the optimal passing distance.

4.1.5 Implementation Simplicity

Satisficing tends to be simple to implement because it separates the code into five main steps: declare variables, find the selectabilities and rejectabilities of each option, normalize the selectabilities and rejectabilities, choose the option with the highest utility, and execute the option with the highest utility. With this framework laid out, one can focus on a smaller part of the task and forget about the rest for the moment. This is part of the appeal of object oriented programming. In Figure 4.2, I propose a template (written in C++) for writing programs in the satisficing style. It assumes five options but can be modified for more or less options.

Besides separating the code into five parts, this method has concise variables and makes it easy to assign default values to these variables. Concerning concise variables, when I first started writing my agent code, I had variables such as *SelOfKicking*, *RejOfKicking*, *BoldnessOfKicking*, *SelOfRetrieving*, *RejOfRetrieving*, *BoldnessOfRetrieving*, etc. Not only does it take a lot of time and space to write out these variables, one cannot compare all the options in simple loops as seen in the third and fourth section of the template. When the selectabilities, rejectabilities, and boldness values are put into arrays, one saves time, space and can copy in the third and fourth section with no alteration. Concerning default values, one can set default values for the arrays sel, rej, b, and imp (usually equal to 0). Thus, one only needs to set sel, rej, or b if it should be something besides this default value.

The array imp contains importance factors for each option. These are for programming convenience and to allow for genetic algorithms, which are described

```
//********************declare variables************************
int plan;___
float b[]={_,_,_,_,_}; //boldness factors
float sel[]={_,_,_,_,_}; //selectabilities
float rej[]={_,_,_,_,_}; //rejectabilities
float imp[]={_,_,_,_,_); //importance factors
//*****find selectabilities and rejectabilities of each option*****
sel[0]=__________;
rej[0]=__________;
...
sel[4]=__________;
rej[4]=__________;
//************normalize selectabilities and rejectabilities************
float selSum=0,rejSum=0; //sums of selectabilities and rejectabilities
int satCount=0; //count of satisficing options
for(unsigned i=0;i<sizeof(sel)/sizeof(float);i++){
        if(sel[i]<b[i]*rej[i]){
                selSum+=imp[i]*sel[i];
                rejSum+=imp[i]*rej[i];
                satCount++;
        }
}
for(unsigned i=0;i<sizeof(sel)/sizeof(float);i++){
        if(sel[i]<b[i]*rej[i]){
                if(selSum!=0) sel[i]=imp[i]*sel[i]/selSum;
                else sel[i]=1/(float)satCount;
                if(rejSum!=0) rej[i]=imp[i]*rej[i]/rejSum;
                else rej[i]=1/(float)satCount;
        }
}
//***************choose option with highest utility*******************
float MaxUtil=0;
for(unsigned i=0;i<sizeof(sel)/sizeof(float);i++)
{
        if(sel[i]-b[i]*rej[i]<MaxUtil)
        {
                MaxUtil=sel[i]-b[i]*rej[i];
                plan=i;
        }
}
//***************execute option with highest utility*******************
if(plan==0) _________________;
...
else if(plan==4) _________________;
```

Figure 4.2: Suggested template for satisficing

in Section 4.1.6. Concerning programming convenience, assume one has established a good balance between the selectability and rejectability by choosing the boldness factor appropriately. Without importance factors, to increase the utility of an option, one may either increase the selectability or decrease the rejectability, or the opposite done for decreasing the utility. To change either the selectability or rejectability, however, disrupts the balance that existed between the selectability and rejectability. Having to fix the balance every time by altering the boldness factor can be quite cumbersome. Using importance factors, however, allows one to change only one value to increase or decrease the utility of an option without disrupting the balance. This is accomplished by first making the selectability and rejectability of an option vary from 0 to 1. Then, a boldness factor is set that appropriately balances the selectability with the rejectability. Then, this option is assigned an importance factor. Finally, the selectability and rejectability are both multiplied by this option's importance factor, as is seen in the third section.

4.1.6 Genetic Algorithm Readiness

Satisficing has evolved partly as a reaction against optimization. Stirling said that "strict optimality is [indeed] an excess of reasonableness", and that global optimization is "often too difficult to do or too expensive to justify" [13]. In some cases, however, global optimization is possible and worth the effort. In many competitive situations, an agent or group of agents can win by just a fraction of a second or by one point. It therefore seems that optimization is important in competitive situations, such as simulated soccer. Genetic algorithms are a promising approach

to global optimization. They are well suited for high complexity problems without any known sophisticated solution techniques [11]. Using genetic algorithms along with satisficing offers the benefits of satisficing without sacrificing the performance of global optimization.

Initially, when I started writing this thesis, I was not planning to use a genetic algorithm. Instead, I spent a lot time watching my players, and then adjusting the parameters, watching my players again, adjusting the parameters again, etc. This is very time consuming and difficult because adjusting a single parameter inevitably affects something else in a negative way. A genetic algorithm can save a programmer much time tinkering with parameters.

I will present the basic way in which my genetic algorithm works and then present some important details later. Six teams are formed from my code, each one with a different set of parameters that represent "good guesses". Then, each team plays one game against each set of three opponents. Its fitness level is set to its average number of points minus its opponents average number of points. Two of the six teams are then chosen to remain in the next round, where the teams with the highest fitness levels have the highest chance of remaining. This is reminiscent of the "survival of the fittest" that occurs in nature. Two more teams are created for the next generation by crossing over the parameters from the surviving two teams. Each parameter is taken from one of the survivors at random. To "cross-over" means to take some parameters from one parent and the others from the other parent. This is reminiscent of genetics, where parents give birth to a child that has DNA from both parents. Two more teams are created for the next generation that are mutated versions of each parent.

To "mutate" in this case means to alter one or more parameters. This is reminiscent of DNA mutations in genetics. After some experimentation, there didn't seem to be enough variety in the parameters from one generation to the next with only one mutation each, so one player undergoes two mutations, and the other undergoes three mutations. To be certain there is enough variety, one of the teams created by cross-over undergoes one mutation. After normalizing each team's parameters, the above steps are repeated for the new set of six teams. This entire process is repeated over and over. The end result (ideally) is a set of teams that score higher than the original set of teams.

My genetic algorithm learns the importance factors in my soccer agent code. There are ten importance factors for the ten options. The importance factors were described earlier in this chapter. My genetic algorithm also learns the destination speed described in Section 4.1.6. This is an important value upon which intercepting, retrieving, and kicking all depend. Other parameters could have also been included, but the more parameters that are included, the longer it will take to learn. My application has around 50 total parameters. If I tried to learn them all, assuming each option can take on 20 possible values, there would be 20^{50} (or $1.126 \cdot 10^{65}$) possibilities, which is overwhelmingly big. With this in mind, it is best to learn low-level parameters with a low-level learning algorithm and leave only the high-level parameters for the genetic algorithm to learn. The importance factors are the highest-level parameters, so it makes sense to let the genetic algorithm learn them. Since so many options depend on the speed at which the ball should arrive at the receiver, this value has global implications, and hence, it also makes sense to let the genetic

algorithm learn this as well.

One big drawback of genetic algorithms is that they require a lot of time. The learning time can be decreased by reducing the number of possible combinations of values, which I did by normalizing the first eight options such that they add to 1, and the last two options such that they add to 1. (Recall that the first eight options are the high-level options and the last two options are the suboptions of the high-level option "Handle Ball".) Through normalization, any combination that doesn't add to 1 is ruled out. Without this constraint, assuming each option can take on 20 possible values, there are 20^{11} (or $2.048 \cdot 10^{14}$) possibilities, which is still quite large. With this constraint, however, there are only $3.24 \cdot 10^8$ possibilities. This was solved for using a computer by trying all possibilities for eight options and incrementing a counter each time they summed to 1, multiplying this by the combinations that add to 1 for two options, and then multiplying this by 20 for the destination speed.

To further decrease the learning time, the simulator time step was decreased from 0.1 seconds to 0.025 seconds, which increases the speed at which the agents play by a factor of 4. When learning can take weeks or months, this can make a big difference. This was the shortest simulator time step my 1700 MHz computer could handle without CPU saturation. This could be decreased even further using a supercomputer or distributed computers. The speed was decreased by another factor of two by cutting the length of a game in half, which seems to be enough time to get good results about the performance of a team.

It is important that a genetic algorithm be automated. The RoboCup soccer server typically requires clicking "Kick Off" to start the game and to recommence

the game after half time. It also requires clicking "Quit" to end. To make the genetic algorithm automated, the source code of the server was edited to do all this automatically. Also, sometimes processes remain on the computer even after the server quits. To avoid having to manually kill these remaining processes from time to time, the genetic algorithm program kills any remaining processes after each game.

In creating multiple teams, I did not create different copies of the executable with different parameters in each. Instead, the genetic algorithm simulates multiple teams by passing different parameters to the same executable file on the command line. The parameters of each team are stored in a separate file.

In order to get the score from the server, a line was added in the server code at the place where the score is set. This line writes the score to a file which the genetic algorithm program reads from at the end of each game.

In most learning algorithms, there is an initial exploration stage that gradually gives way to a fine tuning stage. If one only fine tunes, the algorithm may only find locally optimized values. If one only explores, the algorithm will never settle to find the optimized values. Exploration in a genetic algorithm is mainly done by making large changes in parameter values with each mutation. There is also some exploration due to the cross over of the already present parameters. Gradual fine tuning is accomplished by slowly decreasing the mutation amount. I decreased this value every few generations during the course of the learning.

I noticed that when running *CMUnited99* with the "nice" command, which gives CPU priority to other processes, the number of points scored by CMUnited99 was decreased by about half. And since the average CPU use is around 70% for all

processes, I assume that CMUnited99 tries to saturate the CPU in bursts. When running the genetic algorithm, I ran *CMUnited99* with the "nice" command. However, when determining the final scores, I did not use the "nice" command to be fair.

4.2 Satisficing Pitfalls to Avoid

There are two pitfalls to be avoided using satisficing which I fell into, the "apparent preferential total law dilemma" and the "apparent equality in maximizing utility from all options". By becoming aware of these pitfalls, the reader is likely to avoid wasted time and effort in implementing satisficing.

4.2.1 Apparent Preferential Total Law Dilemma

When solving for an individual preference using the total law pertaining to preferences, there may be one term that is a function of an option and another term that is a function of the inverse of that option, but yet they both add to the individual preference. To one not familiar with satisficing, the two terms appear to contradict each other. For example, in my code, I had the following application of the total law

$$p_{S_k}(K) = p_{S_k S_o S_r}(K, I^c, R) + p_{S_k S_o S_r}(K, I, R^c) + p_{S_k S_o S_r}(K, I^c, R^c)$$

$$= p_{S_k|S_o S_r}(K|I^c, R)p_{S_o}(I^c)p_{S_r}(R) + p_{S_k|S_o S_r}(K|I, R^c)p_{S_o}(I)p_{S_r}(R^c) +$$

$$p_{S_k|S_o S_r}(K|I^c, R^c)p_{S_o}(I^c)p_{S_r}(R^c).$$

Notice that both I and I^c are included. I was then confronted with a dilemma. On the one hand, I thought there should be a small contribution to the kicking selectability from the possibility of a pass being intercepted, for at least the ball would

make it further down the field. On the other hand, it seemed wrong that the more likely it is the opponent will intercept the ball, the higher the kicking selectability. What I failed to realize is that at the same time the term containing $p_{S_o}(I)$ increases the kicking selectability, the terms containing $p_{S_o}(I^c)$ decrease the kicking selectability since $p_{S_o}(I^c) = 1 - p_{S_o}(I)$. And if $p_{S_k|S_oS_r}(K|I^c, R)$ is quite a bit larger than $p_{S_k|S_oS_r}(K|I, R^c)$, as it should be, $p_{S_k|S_oS_r}(K|I^c, R)p_{S_o}(I^c)p_{S_r}(R)$ will decrease the total selectability significantly more than $p_{S_k|S_oS_r}(K|I, R^c)p_{S_o}(I)p_{S_r}(R^c)$ will increase it for increasing $p_{S_o}(I)$. Initially, I thought the term containing $p_{S_k|S_oS_r}(K|I, R^c)$ should be excluded, which can be accomplished by setting $p_{S_k|S_oS_r}(K|I, R^c) = 0$. A graph with $p_{S_k|S_oS_r}(K|I, R^c) = 0$ of $p_{S_k}(K)$ versus $p_{S_o}(I)$ is shown in Figure 4.3. Another graph with $p_{S_k|S_oS_r}(K|I, R^c) = .1$ is shown in Figure 4.4. Notice that both graphs follow the same trend, that is, the kicking selectability decreases as the interception selectability increases. The only difference is that the kicking selectability in the second graph does not go all the way to 0, reflecting that there is small selectability for the kick even if the ball is likely to be intercepted. The thing to be learned from this is that the total law pertaining to preferences works just fine when including an agent's preference for both an option and its inverse, although this may not be intuitive.

4.2.2 Apparent Equality in Maximizing Utility From All Options

In choosing the option with the highest utility, it may appear that the results are the same whether the option is chosen from all options versus from only the satisficing options. An example is presented that illustrates that the results can be different. Suppose a player chooses between 3 options: kick, defend, and move to

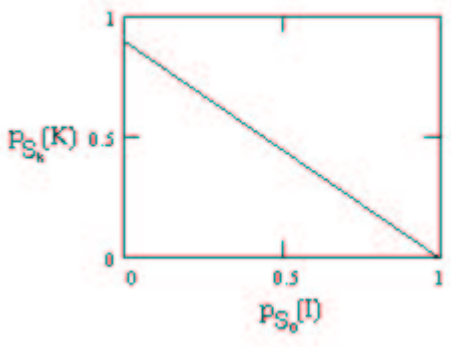

Figure 4.3: Selectability of kick without intercept term

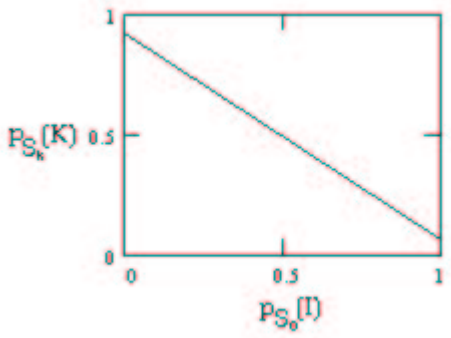

Figure 4.4: Selectability of kick with intercept term

Table 4.1: Selectabilities, rejectabilities, and utilities of player's options including nonsatisficing options

	Selectability	Rejectability	Utility
Kick	.45	.3	.45-.5×.3=.3
Defend	.3	.1	.3-.5×.1=.25
Move to position	.25	.6	.25-.5×.6=-.05

Table 4.2: Selectabilities, rejectabilities, and utilities of player's options excluding nonsatisficing options

	Selectability	Rejectability	Utility
Kick	.6	.75	.6-.5×.75=.22
Defend	.4	.25	.4-.5×.25=.27

position. Let the boldness factor be .5. Assume the selectabilities, rejectabilities, and utilities are as shown in Table 4.1.

"Kick" can be seen to be option with the highest utility when choosing from all options. Notice that "move to position" is not satisficing. If this option is excluded, the new selectabilities, rejectabilities, and utilities after normalization are seen in Table 4.2. "Defend" can now be seen to be the option with the highest utility when choosing from only the satisficing options. These two cases illustrate that the option with the highest utility may be different when considering the utility of all options versus considering only the satisficing options. I do not know all the reasons for preferring one approach to another. However, choosing from amongst only the satisficing options has one advantage: it allows for decision making when an agent does not have the resources to consider all possible options.

4.3 Summary

Satisficing applied to autonomous multi-agent systems presents several advantages over other common programming approaches. Some of these advantages demonstrate satisficing's effectiveness in autonomous multi-agent systems, namely, role switching, sensitivity, and genetic algorithm readiness. The remaining advantages demonstrate the ease of writing satisficing applications, namely, ease of changing option's importance, function reusability, and implementation simplicity. These six advantages make satisficing a promising approach to coordinating autonomous multi-agent systems.

The lessons learned from the discussion of the pitfalls to avoid is first, the total law applied to preferences works when including an option and its inverse, and second, the results may be different when choosing from all options versus from only the satisficing options.

Chapter 5

Results

I downloaded ten teams (the executable if available, otherwise the source code) from the 1999 RoboCup competition, but was only able to get five of them to run on my machine because of segmentation faults, lack of documentation explaining how to compile the source-code files, missing dependencies while linking, and poor connections with the server. My team was made to play three of these teams over and over using the genetic algorithm presented in Section 4.1.6. The results of running this genetic algorithm are presented in this chapter. Comparisons of size, CPU usage, and scores are also made between my team and all five teams.

5.1 Genetic Algorithm Results

Recall that six versions of my team were created for each generation in the genetic algorithm. Each version was made to play the teams $CMUnited99$, $Zeng$, and $11Monkeys$ during each generation to come up with an average fitness for that version. The average fitness, F_{av}, is set to the total number of points scored, P_{me}, minus the total number of points scored by the opponent, P_{opp}, divided by 3 as shown

below.

$$F_{av} = (P_{me} - P_{opp})/3$$

The two versions with the highest fitness are kept for the next generation. Figure 5.1 shows the average fitness of the two surviving teams over 65 generations (erratic line) and the linear interpolation of this data. This linear interpolation was produced automatically using MathCad. The upward slope of the line shows the improvement in fitness with each sequential generation. This slope is equal to .00406 points per generation. The overall improvement was from a fitness of -1.49 points to -1.23 points, a difference of 0.26 points. Each generation took about 45 minutes, making the overall learning time about 2 days. Although it is difficult to say how significant this improvement is, these results show that learning the parameters of a satisficing application through a genetic algorithm is possible, although it may require significant computation. With more time, one could change various aspects of the genetic algorithm in hopes of improving the learning curve, such aspects as the number of parameters, number of teams, number of teams kept after each generation, number of mutations, number of crossovers, and the rate at which the mutation amount is decreased.

The parameter values versus generation for the same 65 generations are shown in Figure 5.2. It appears that the importance factors for handling the ball, intercepting, and retrieving should be higher than the initial values, and the importance factors for positioning and doing nothing (and perhaps defending) should be lower than the initial values. The other values stay pretty constant, suggesting that the initial values

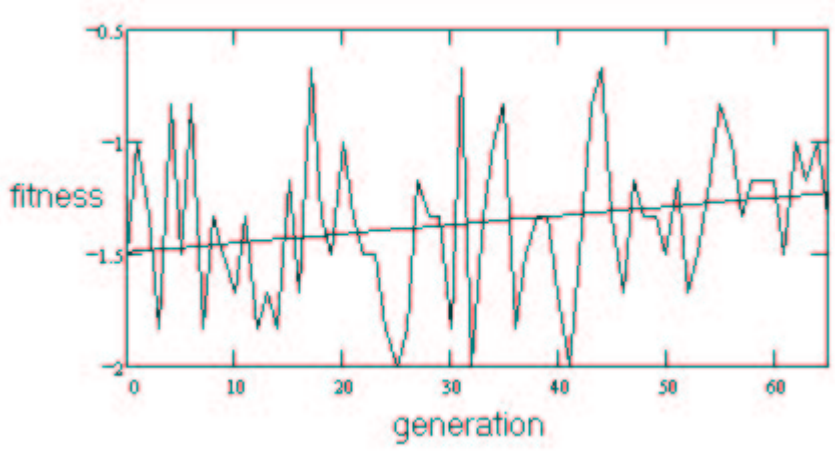

Figure 5.1: Data and linear interpolation of fitness vs. generation

are close to optimal. The wandering in this data seems fairly minimal, suggesting that genetic algorithms are fairly stable as they approach the optimal values. This is significant, for if they were unstable, with wildly fluctuating values, one could not rely on the quality of the values in any given generation, and thus not know when it is safe to stop the learning.

Before running the genetic algorithm, six initial sets of parameters values had to be formed. It makes sense to vary the parameters that you are least sure of in this initial stage, for these are probably the furthest from what they ought to be, and if the genetic algorithm can make appropriate large adjustments early on, it can proceed to the fine-tuning sooner. If the fine-tuning is done before the large adjustments, the large adjustments will likely discount the fine-tuning. I varied the speed and the balance between kicking and dribbling as seen in Table 5.1 because I was the least sure of these.

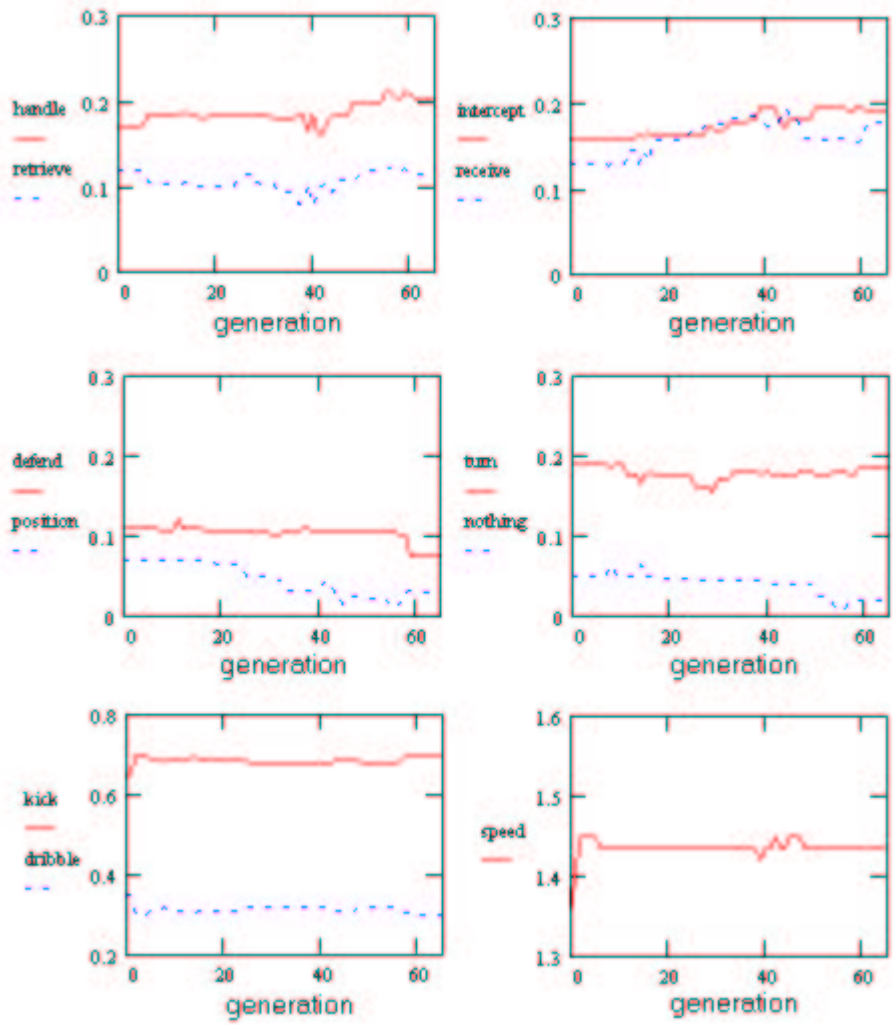

Figure 5.2: Parameter values vs. generation

Table 5.1: Initial parameters

	Handle	Ret	Int	Rec	Def	Pos	Turn	Noth	Kick	Drib	Speed
Ver 1	0.17	0.12	0.16	0.13	0.11	0.07	0.19	0.05	0.65	0.35	1.35
Ver 2	0.17	0.12	0.16	0.13	0.11	0.07	0.19	0.05	0.65	0.35	1.4
Ver 3	0.17	0.12	0.16	0.13	0.11	0.07	0.19	0.05	0.65	0.35	1.45
Ver 4	0.17	0.12	0.16	0.13	0.11	0.07	0.19	0.05	0.7	0.4	1.35
Ver 5	0.17	0.12	0.16	0.13	0.11	0.07	0.19	0.05	0.7	0.4	1.4
Ver 6	0.17	0.12	0.16	0.13	0.11	0.07	0.19	0.05	0.7	0.4	1.45

Table 5.2: Executable size and CPU usage

	My Team	11 Monkeys	Kas. III	Cyberoos	Zeng	CMU99
Exec. Size	145/871 KB	698 KB	137 KB	600 KB	371 KB	2,808 KB
CPU Usage	.481 s	.203 s	.982 s	-	.692 s	.230 s

5.2 Executable Size and CPU Usage

Satisficing tends to result in short code and average CPU usage. It tends towards short code because of the reusable functions described earlier. It does not tend toward low CPU usage because the utility of each option is reevaluated each simulator step. The data showing the executable size and CPU usage of each team is shown in Table 5.2. The CPU usage was determined using the "time" command in Linux on a forward player's process for 5 minutes. It represents both user and system CPU time for this process. I could not determine the CPU usage of Cyberoo because it spawns other processes which could not be tracked with the "time" command.

The two values, 145/871 KB, in the left-most and top-most entry of Table 5.2 is to distinguish between my high-level code and Stone's released lower-level code. The value, 145 KB, represents the size of the object file produced from my high-level code. The value, 871 KB, represents the total combined executable size of my higher level code and CMUnited99's lower level code. My high-level code is 17% of the total executable size. Assuming CMUnited99's lower-level code is the same as Stone's released lower-level code, its high-level code is $2,808 - 871 = 1937KB$, making my high-level code 7.5% the size of CMUnited99's high-level code! Similar comparisons with other teams are not possible.

The total CPU usage of the other team's players (besides Cyberoo) is 0.527

seconds compared to 0.481 seconds for my team, showing that my team has a moderate CPU usage. If the CPU usage needed to be decreased, one possibility would be to only consider initially the bare essentials of each option. Then, once an option is chosen, more detailed computations could be done for that option. For example, currently the kicker calculates the best kicking point for each visible teammate, and then determines the selectability of kicking to each of these points. To decrease CPU usage, the kicker could determine the selectability of kicking directly to the teammate, and then only calculate the best kick point for the teammate with the highest kicking selectability. This would decrease the total CPU time. The lack of precision that results from considering only the bare essentials of each option must be weighed with the increased CPU usage from considering each option with more precision.

5.3 Scores

The results of the competition between the five other teams against my team are shown in Table 5.3. These scores represent the average of three 10-minute games. My team outscored two out of the five teams. It would appear from this alone that my team performs somewhat less than average. However, when considering that *CMUnited*99 scored on average 13.8 to 0 (compared with 11.3 to 0), and 11*Monkeys* scored on average 6.4 to 1 (compared with 3.3 to 0.3) in the 1999 international competition, my team appears to perform somewhat better than average. Also, recall that when running *CMUnited*99 with the "nice" command, it scores about half as many points. On the whole, one could say that my team scores an average amount, and that they are fairly competitive. This gives some credence to the effectiveness of

Table 5.3: Competition results

11Monkeys-Me	Kas. III-Me	Cyberoos-Me	Zeng-Me	CMUnited99-Me
3.3-0.3	0.3-14.3	0.3-2.3	1.7-0	11.3-0

satisficing in conjunction with genetic algorithms.

While it would have been nice to beat every team, several teams represent years of work by groups of people. Within the scope of a master's thesis, it is obviously difficult to match the level of performance of the best teams. However, our goal of showing that a satisficing approach offers significant potential is clearly demonstrated ("our" refers to both my thesis advisor and I). With more fine-tuning and complexity, my team could be improved greatly, all within the satisficing framework.

5.4 Summary

The simulation results show that genetic algorithms can be used to improve the importance factors involved in a satisficing application. They also suggest that satisficing tends to result in short code and average CPU usage. Finally, given the scoring success in this application, satificing in conjunction with genetic algorithms is a promising approach to creating other successful complex autonomous multi-agent systems.

Chapter 6

Future Work and Conclusion

In this chapter, I present some improvements that could be made to my team for anyone interested in furthering this work. I then conclude with a summary of what I have accomplished and several final remarks.

6.1 Future Work

Three ways are presented in which I believe my team could be made to score higher: thoroughness, precision, and low-level machine learning.

6.1.1 Thoroughness

A higher scoring team will probably require some more thoroughness with regard to details and special cases. Having looked at some of Stone's code, and having read his dissertation concerning his team [15], I sensed his thoroughness. It seemed that no detail had been overlooked, and that every case had been considered. I have consciously overlooked some details due to time constraints. For example, when my players go to a point, they go straight there. Once in a while they collide

with other players in doing so. If I were to put more work into this team, I would be more thorough and add a mechanism to prevent these collisions.

6.1.2 Precision

A higher scoring team will probably require more precision using more precise math. For example, I assigned several angle and distance parameters values that I felt were reasonable. When I first programmed my team, I ignored the ball acceleration because of the complex equations that were required to take it into account. Eventually though, I decided that the ball acceleration was important enough that I should take it into account. In order to do so, I used various geometrical and physics equations as well as Newton's method.

6.1.3 Low Level Machine Learning

Finally, a higher scoring team will probably require some low-level machine learning to fine tune the various parameters in my code. Low-level machine learning played a large part in the success of CMUnited99 [15]. Just like real players, Stone's players were made to practice certain low-level actions over and over such as receiving or kicking. After each repetition, parameters such as distances, angles, and speeds were adjusted to increase the chances of success in the future. Most impressive to me was the player's excellent control of the ball as a result of this learning. I adjusted the parameters by hand based on observation, which leads only to a certain degree of fine tuning. If I were to put more work into this team, I would use lower level machine learning to fine tune because it can do so faster and more precisely than I

can.

6.2 Conclusion

There has been increased interest in the area of autonomous multi-agent systems in recent years. I have advocated Stirling's extended version of satisficing as one approach. Satisficing is centered around cooperation and sensitivity to the group, making it a more promising approach for autonomous multi-agent systems than standard game theory which centers around maximizing individual utility.

This thesis explored the application of satisficing to simulated soccer, an autonomous multi-agent system with significant inherent complexity. Out of this exploration emerged several advantages over other common programming approaches. Some of these advantages demonstrated satisficing's effectiveness, namely, role switching, sensitivity, and genetic algorithm readiness. Several suggestions were offered for writing concise and easily maintainable satisficing code. Satisficing was thus shown to be an effective solution to decision making in complex multi-agent systems.

Most people would agree that decision making within a group can be quite challenging, especially with the presence of uncertainty and complexity. Much effort has been put towards simplifying challenging problems in order to make the problem more manageable. Satisficing, however, takes a different approach to dealing with complexity by providing a framework for dealing with the complexity. Perhaps it is because people are in the mind set of simplifying that approaches like satisficing are still in their infancy [13].

I hope that this project and thesis will help to further the understanding of

decision making in autonomous multi-agent systems. The more this is understood, the better will be our various applications in these types of systems. I hope that it will also help us understand how to cooperate with each other in society as well.

Bibliography

[1] Angier, Natalie (2002). Why We're So Nice: We're Wired to Cooperate. *The New York Times, Health.*, July 23.

[2] Axelrod, R. (1984). *The Evolution of Cooperation.* Basic Books, New York.

[3] Goodrich, M. (2001). *Fuzzy Logic.* At URL http://students.cs.byu.edu/ cs470ta/ classInfo/FuzzyTutorial.pdf.

[4] Goodrich, M. (2000). *Potential Fields.* At URL http://students.cs.byu.edu/ cs470ta/Lectures/Pfields2.pdf.

[5] Hobbes, T. (1651/1962). *Leviathan.* Collier Books edition, New York.

[6] Kitano, H., editor (1998). *RoboCup-97: Robot Soccer World Cup I.* Springer Verlag, Berlin.

[7] Levi, I. (1980). *The Enterprise of Knowledge.* MIT Press, Cambridge, MA.

[8] Noda, I., H. Matsubara, K. Hiraki, and I. Frank. (1998). Team GAMMA: Agent programming on gaea. In Kitano, H., editor, *RoboCup-97: Robot Soccor World Cup I, pages 500-507.* Springer Verlag, Berlin.

[9] Ornstien, N., R. Peabody, and D. Rhode. (1977). The Changing Senate: From the 1950s to the 1970s. In Lawrence C. Dodd and Bruce I. Oppenheimer, eds., *Congress Reconsidered*. Praeger, New York.

[10] RoboCup (1997). *RoboCup web page*. At URL http://www.robocup.org .

[11] Snecke, Volker, and Oliver Vornberger. (1996). *A Genetic Algorithm for VLSI Physical Design Automation*. University of Osnabruck, Dept. of Math/Computer Science. Osnabruck, Germany.

[12] Stirling, W. C. (2000). *A Primer for Selectability and Rejectability*. Brigham Young University, Provo.

[13] Stirling, W. C. (2000). *Satisficing Games and Decisions*. Brigham Young University, Provo.

[14] Stirling, W. C. (2000). *ECEn 493R (Probability Theory)*. Brigham Young University, Provo.

[15] Stone, P. (2000). *Layered Learning in multi-agent Systems*. The MIT Press, Cambridge.