# Abstract

This paper presents an examination of the application of probability density functions to strategy identification in partially observable game spaces. Specifically, we focus on the design, implementation, and evaluation of the Intent Recognition Engine (IRE) for the NOVA StarCraft competition AI. We describe the challenge of determining enemy strategy with only a partial view of their total forces, of mapping strategies to a coordinate space, and of performing localization in that space. In addition, we present the results from evaluation of both win rates against enemy AI as well as prediction accuracy given post-game ground truth. Finally, we conclude with a discussion of remaining challenges and opportunities for further development.

Index Terms – Game AI, Real-Time Strategy, StarCraft, Partial Observability, Probability Density Function

# Introduction

Early prediction of enemy strategies and tactics produce a marked improvement in effectiveness against intelligent opponents (Laviers, 2009). Real-time strategy (RTS) games have become a rich platform for the development and advancement of related AI techniques due to their decision complexity, partial observability, and their variable determinism. This research has led to the production of many bots with varying strategies which compete in yearly AI competitions such as the “AIIDE StarCraft AI Competition” and the “CIG StarCraft RTS AI Competition”. NOVA is one such bot first developed in 2013 at Drexel University. The Intent Recognition Engine (IRE) builds on top of existing NOVA capabilities to improve its predictive accuracy against enemy bots.

RTS games present intractable challenges for traditional decision-making systems due to the large number of possible moves that every unit can make when combined with the large number of units that can exist at the same time. Combined with the partial observability of the environment, this results in a solution space that is far, far too large to search using traditional methods. In addition, the real-time nature of the genre means algorithms must act quickly, effectively ruling out many decision-making processes that promise optimality. Developing techniques that can rely only on observable data to make inferences about higher level enemy decision making can help ameliorate this problem by trading optimality for probabilistic predictions. Effective implementation of these techniques can produce inference systems that are effective across many types of partially observable environments.

This paper aims to detail one such technique that applies algorithms developed for geolocation given imperfect information to identification of enemy strategies by mapped them into a N-dimensional space. It is organized as follows: Section II introduces the RTS game StarCraft and its unique challenges, the concept of mapping strategies to points in space, and probability density functions. Section III reviews the algorithmic details of IRE detailing how observations in-game are converted to strategy predictions and, ultimately, counter-strategies. Section IV details the test procedure and evaluation results of IRE trials against other competitive bots. Section V touches on related work being performed in this area. Section VI outlines the conclusion made from this research and development effort. Finally, Section VII outlines future work to be performed on the algorithm to further improve results.

# Background

## RTS Games

Real-Time Strategy (RTS) games are a sub-genre of grand strategy games where players are tasked with gathering resources to construct buildings, train units, and research upgrades in order to achieve victory over one or more opponents. Although some games offer a variety of “win conditions”, the vast majority define victory as the total destruction of enemy assets. RTS games have become a hotbed of AI research and the next step in the evolution of adversarial AI beyond Chess and Go, in part because they offer the following unique challenges:

* Unlike Chess and Go, both players can perform actions simultaneously. In addition, while some moves are limited only by the speed at which the player can perform them (further limited by the rendering frame rate), other moves are *durative*, meaning the moves themselves take time to complete and in some cases can be interrupted by the player or the enemy.
* By nature of allowing simultaneous actions, RTS games are real-time, meaning game state advances regardless of player action. All other things being equal, the player which can act faster has an advantage over the other players. This means that fast, effective decision making is required for competitive systems.



Figure - In most RTS titles, you can only observe what appears in the revealed area of your units sight ranges. Objects in the fog of war will not be updated, and objects in the black mask will not be known until first revealed.

* Whereas with Chess and Go the entire board is visible at all times, many RTS games have partial observability through a “fog of war” (Figure 1) that limits view into enemy activity. All game state changes that take place outside the combined visibility ranges of all friendly assets are invisible to the player until that area is observed.
* Many RTS actions are non-deterministic, meaning they have a probabilistic chance of success or failure and thus the results cannot be predicted with perfect accuracy.
* The combined complexity of all possible unit actions across all existing units against all known, previously known, and unknown enemy states significantly exceeds that of both Chess and even Go in terms of not only state space but also decision tree size.

### StarCraft

StarCraft: Brood War is a RTS game released by Blizzard Entertainment in 1998. Since then it has become a cultural and competitive phenomenon, going as far as to become a televised sport in some countries. The game is set in a science-fiction universe consisting of three races: Terran, Protoss, and Zerg. Each race consists of unique buildings, units, and abilities with a heterogeneous set of capabilities between the three races. Each race specializes in different combat styles: balance for Terran, power for Protoss, and speed for Zerg making the right strategic decision a factor not only of current resources but of enemy race. That also means that enemy strategies will vary from race to race and an effective counter-strategy for one race will not apply evenly to all races. In addition, and after over a decade of regular software updates, it has also become one of the most balanced games in the genre, making it an ideal choice for the development and evaluation of AI systems against peer adversaries with heterogeneous capabilities.



Figure - StarCraft games involve heterogeneous armies battling until only one remains

## Strategy Representation

In order to identify opponent strategies, it is first necessary to produce a representation of strategy that is robust to the various tactics across all three races. In StarCraft every unit, building, and research upgrade has a fixed set of pre-requisites that must exist before the object can be produced. When all these components are combined for a given race, it produces a single “tech tree” representing the races’ complete set of capabilities (Figure 3). Stemming from this representation is the notion of “build order”. A build order is a unique set of ordered productions (building, unit, or upgrade) designed to produce certain units in certain quantities necessary to execute a specific strategy. This build order can be represented as a subtree of the races’ tech tree consisting of only the nodes and edges required to achieve that series of productions. Every strategy in StarCraft can be represented by a build order, and while there is nuance to the exact execution of that build order, such as building location placement or map exploration, the order itself cannot change due to the strict dependencies between objects.

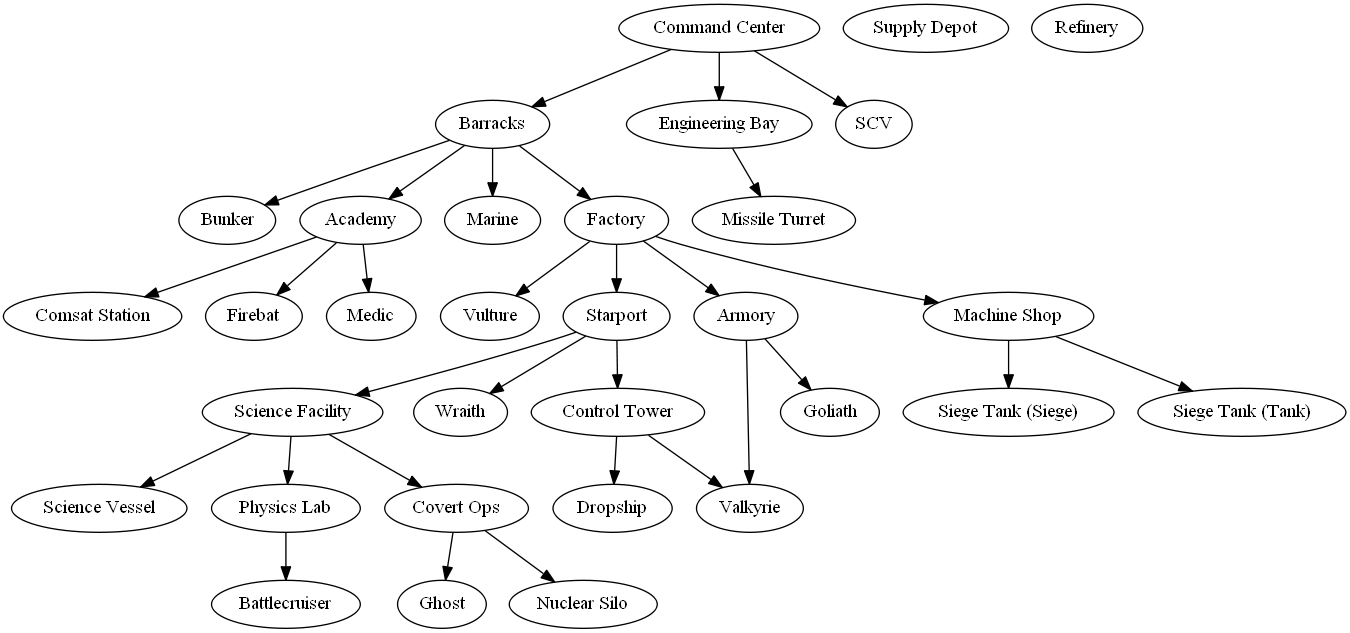


Figure - A segment of the Terran Tech Tree showing some of the dependencies between units and buildings, which are always one-directional.

Some strategies rely on aggressive assaults while others are designed to defend until a large enough army can be produces. Some strategies are more effective against ground units, while others are more effective against air units. By identifying a global set of parameters, every strategy can be given a unique fingerprint consisting of an intensity measurement for that strategy against each parameter. This complete set of features can then be combined to create an N-dimensional space representing all possible strategies, henceforth called the “strategy space”. This strategy space consists of virtual axes that represent each feature, where each axis is normalized from -1 to 1. A point in strategy space, then, represents a possible strategy whose parameter intensities match the location provided by the point.

The first axis measures a strategy’s focus on ground units versus anti-ground units. The second axis measures a strategy’s focus on air units versus anti-air units, and the third axis measure’s a strategy’s overall aggressiveness versus defensiveness. The closer to zero a point is along any given axis, the more “balanced” that strategy is in producing units that can achieve, or fails to achieve, both extremes. For example, a strategy that emphasizes aggressively building ground units that cannot attack air units to attack the enemy without concern for self-defense would measure as a 1 on the ground axis, a 0 on the air axis, and a 1 on the aggression axis. Conversely, a strategy that involves building heavy base defenses to produce a late-game aerial armada would measure as a -1 on the ground axis, a 1 on the air axis, and a -1 on the aggression axis.

These axes were selected based on the fixed nature of unit capabilities in RTS games. All units in StarCraft are either air or ground units, and each can either attack ground units, air units, or both. In addition, successful strategies in StarCraft either emphasize attacking as quickly as possible or defending against opponent “rushes” in the hopes of producing more powerful units. In addition, these axes were evaluated against the subset of strategies modeled by IRE to ensure that no two strategies had the same fingerprint. Although these particular axes were chosen due to their relevance to the most common build orders, any arbitrary axis could be selected and added to the code without consequence to IRE. For testing, the 18 most common build orders were codified in this manner and scored [TABLE X Y Z].

|  |  |  |  |
| --- | --- | --- | --- |
| **Terran Strategy Name** | **A vs AA** | **G vs AG** | **Atk vs Def** |
| Bio (marines/medics) | -0.25 | 0.75 | 1 |
| Rax\_fe (second base) | 0 | 0 | -1 |
| Two\_facto(tanks) | 0 | 1 | 0.5 |
| Vultures | 0 | 1 | 0.75 |
| Air(wraiths) | 0.75 | -0.5 | 0.25 |
| drop | 0.25 | 0.25 | -0.25 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Protoss Strategy Name** | **A vs AA** | **G vs AG** | **Atk vs Def** |
| Two\_gates (zealots) | 0 | 1 | 1 |
| fast\_dt (dark templars) | -0.25 | 0 | 0 |
| Templar | -0.5 | 0 | 0.5 |
| Speedzeal (zealots+upgrades) | 0 | 1 | 0.75 |
| Corsair | -1 | 0 | -0.75 |
| Nony (dragoons) | 0 | 0 | 0.5 |
| Reaver\_drop | 0.25 | 0 | -0.25 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Zerg Strategy Name** | **A vs AA** | **G vs AG** | **Atk vs Def** |
| Speedlings (zerlings) | 0 | 1 | 1 |
| Fast\_mutas (mutalisks) | 1 | -1 | 0.25 |
| Mutas (expand+mutas) | 0.75 | -0.75 | -0.25 |
| Lurkers | 0 | 0 | -0.25 |
| Hydras | -0.25 | 1 | 0.75 |

When a strategy is loaded into strategy space, each node in the subtree is given a 3-dimensional point representing its intensity along those axes. Strategies are normalized to their unique depth, where the lowermost nodes have intensities matching the strategy’s overall values and all other nodes scale evenly between those values and zero. So, a node halfway down a strategy with an aggressiveness of 1 would have an aggressiveness value of 0.5.

## Probability Density Functions

In order to predict enemy strategies in partial observability, a probabilistic approach must be taken since perfect data will almost always be unavailable. This also benefits early prediction as multiple similar strategies may look nearly identical during early phases of gameplay when the same types of assets are being produced but diverge later when advanced assets become available. As such, IRE extends the concept of mapping strategies into a coordinate space to geolocate the actual enemy strategy in this space using probabilistic methods, in particular Discrete Probability Distributions (DPD) applied to Probability Density Functions (PDFs).

*In probability theory, a probability density function (PDF), or density of a continuous random variable, is a function, whose value at any given sample (or point) in the sample space (the set of possible values taken by the random variable) can be interpreted as providing a relative likelihood that the value of the random variable would equal that sample.[citation needed] In other words, while the absolute likelihood for a continuous random variable to take on any particular value is 0 (since there are an infinite set of possible values to begin with), the value of the PDF at two different samples can be used to infer, in any particular draw of the random variable, how much more likely it is that the random variable would equal one sample compared to the other sample. (Wikipedia)*

PDFs have many uses, but one such use is in the localization of objects in space. Using only a signal intensity and bearing to a transmitter, a receiver can use repeated measurements over time in order to reduce the number of possible places the object could be until eventually it converges with very high probability on the exact location. [CITE] Each individual measurement produces a discrete probability density (DPD) map which represents a PDF for the target from a given source of measurement. These DPD maps are layered on top of each other, and overlapping areas represent higher probability locations for the target. Over time and with additional observations, the actual location of the target will emerge with very little error as the most overlapped location additively rises above all other possibilities.

IRE applies this concept to locate an enemy strategy in an N-dimensional strategy space. Enemy unit or building observations act as “emissions” in strategy space. A strategy must inherently contain units and buildings, and so each one observed provides an error-bound estimation of the enemy’s strategy. Ground units will appear in ground strategies, air units will appear in air strategies, late-game units will appear in defensive strategies, etc. A unit or building type may appear in multiple strategies, and as such no single observation is sufficient to confidently predict the enemy strategy. However, even a single observation can provide guidance: a unit that appears in multiple ground strategies will necessarily imply that, if however slight, the enemy is more likely to be attempting a ground strategy versus an air strategy if that is the only unit observed.

# Converting Observations into Predictions

MISSING ALGORITHM/DETAILS TO ALLOW OTHERS TO RE-IMPLEMENT

As with geolocating, each observation produces a range of possible positions in the space, but as observations are laid on top of each other, the number of potential overlapping strategies decreases. With enough observations, eventually one strategy will emerge as the most likely. This gradually increasing predictive accuracy means that IRE and NOVA can begin producing a mix of relevant counter-units which converges on the most effective units as then number of possible strategies decreases. If the enemy changes strategies early in production, those new observations will converge on a different part of strategy space.

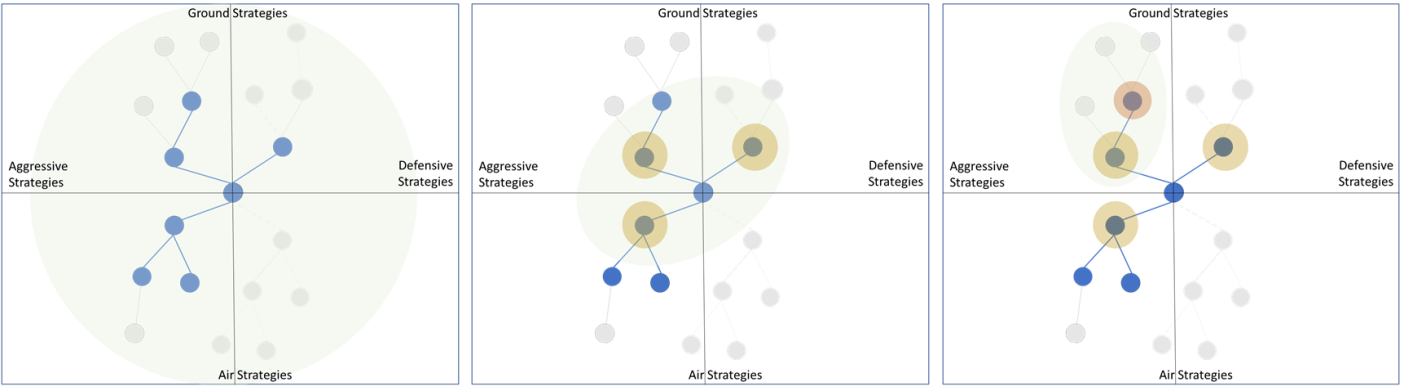


Figure - Initially all strategies are possible. As observations are made, the error bounds around the actual strategy being used converges, eventually settling on the actual strategy being used

The combined set of all strategies in a single strategy space create a virtual N-dimensional space representing all known enemy strategies. The goal of IRE is to use observed enemy units to attempt to “geolocate” the enemy’s actual strategy within this space. Initial observations will begin to bias NOVA’s decisions in what units to produce towards those that are more effective against certain strategies. As more and more enemy units are observed, likely strategy “locations” will begin to layer on top of each other, creating a multiplicative effect until the single most likely strategy is observed. Because producing units takes time and resources, the probability that an enemy attempting an anti-air strategy as they produce units incapable of achieving these strategies approaches zero as the number of observed units increases.

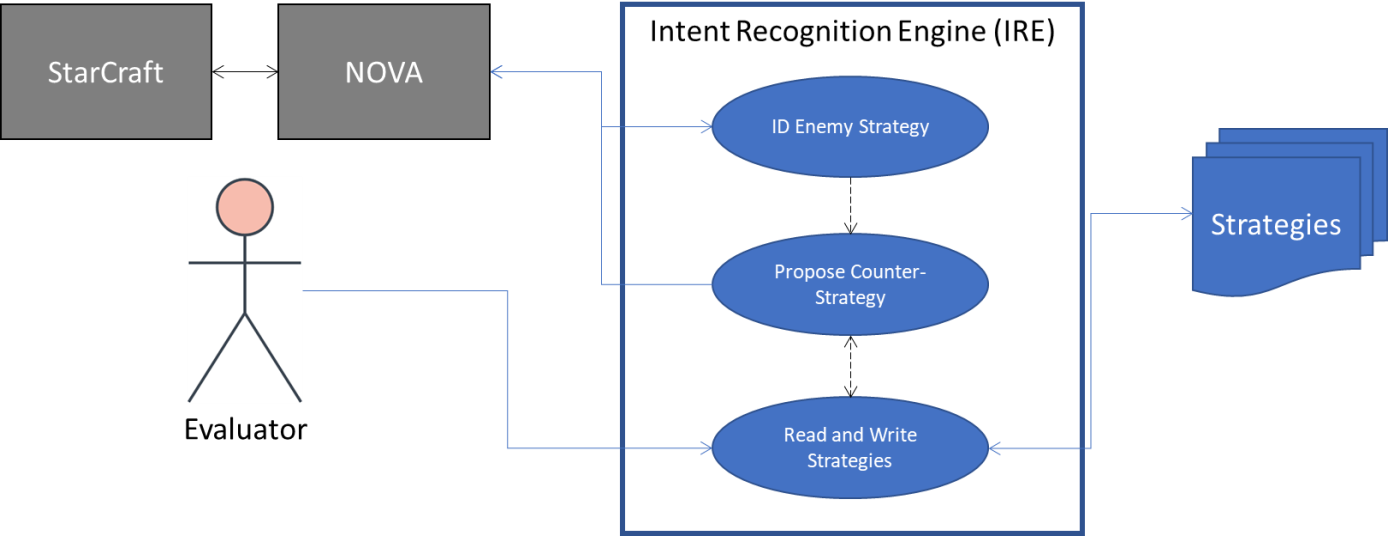


Figure - IRE uses observed enemy units to infer the opponent's possible strategies and use the most likely candidates to bias unit production towards effective counter-units

onAddStrategy(NEW strategy)

FORALL nodes in strategy.tree

Set node depth

FOREACH axis in strategyAxes

Node.axis = node\_depth / max\_depth \* axis.value

onObserveEnemyAsset(NEW Observation)

FORALL Strategies

FORALL nodes

If(node == observation)

node.intensity++

onDecideStrategy()

SORT(strategies.nodes, node.intensity, greaterThan)

FOREACH axis in strategyAxes

FOR top 5 nodes in sortedNodes

totalAxisValue += Node.axis.value

axisAverage = totalAxisValue / 5

suggestedCounter = axisAverage \* -1

setBuildPriority(axis, suggestedCounter)

# Experimental Evaluation

## Overview

The semi-random nature of enemy AI in videogames makes normal evaluations difficult. A single trial, no matter how thorough, isn’t likely to exercise all paths through the system and even a carefully selected series of trials won’t account for the randomness inherent in enemy decision making. As such, IRE evaluation is a natural fit for Monte Carlo simulations. To properly determine IRE’s effectiveness, many hundreds of trials were performed across different enemy types. These results were collected, evaluated, and are summarized below. Access to source code, documentation, and examples can be found at <https://github.com/mikewkozak/IRE> for anyone interested in reproducing these results.

(NEED enough details to allow others to replicate)

## Metrics

IRE evaluation focused on three primary metrics: overall win rate, enemy-specific win rate, and prediction accuracy. The first two metrics are primarily concerned with ensuring that IRE improves or at least matches the current effectiveness of NOVA without decreasing existing capabilities. The final metric does not consider existing NOVA performance. All trials were performed with 2 configurations of NOVA: one with and one without IRE. The evaluative procedures for each metric are as follows:

### Metric 1: Win Ratio

Of the three, this metric is the most straightforward. The total number of wins across all trials was divided by the total number of trials performed for both configurations of the system, with the goal being an improvement in win rate when using IRE.

### Metric 2: Race-Specific Win Ratio

NOVA does not perform equally well against all enemy races. To further examine the nuances in benefit that IRE provides, both configurations were examined on a per-race basis. The trials performed for each configuration were first sorted by opposing race. Each sorted list is then evaluated using the same win ratio process as above, with the goal being an improvement in the win rate for at least once race when using IRE.

### Metric 3: Prediction Accuracy

* Measure **Jaccard index** of the system:
  + Where A = number of accurate predictions
  + And B = number of total predictions

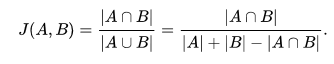


Figure - The Jaccard Index measures the overall predictive accuracy of IRE against enemy strategy selection

## Evaluation Procedure/Setup

* Detail test harness
* Detail automated trial runs
* Detail log data (CSV?) for post-processing of results

## Results

Base NOVA averaged a win ratio of X% across all trials. NOVA with IRE providing additional support by comparison had an average win ratio of Y%.

<Charts and graphs>

# Related work

The work presented in this document is related to work in the field of planning in partial observability, specifically Opponent Modeling, Plan Recognition, and Markov Decision Processes. Each of these techniques takes a unique approach to modeling and predicting adversarial intent and comes with their own strengths and weaknesses relative to IRE.

(request permission from anyone specifically for graphics)

## Opponent modeling

## Plan recognition (expert systems)

## Markov Decision Processes (MDTs and POMDPs)

# Conclusions

# Future Work

* Notes about what didn’t quite work out
* Notes about adding probabilistic edges in strategies to link units that are often but not always together
* Notes about reinforcing strategy edges using post-game ground truth and saving to file
* Notes about individualizing models to learn specific AI habits to bias results
* Notes about autonomously learning new strategies based on post-game ground truth

# References

J. T. Isaacs, e. a. (2014). "GPS-optimal micro air vehicle navigation in degraded environments". *American Control Conference* (pp. pp. 1864-1871). Portland, OR: IEEE.

K. Wendlandt, M. B. (2005). "Indoor localization with probability density functions based on Bluetooth". *IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications, Berlin*, pp. 2040-2044 Vol. 3.

S. Venkateswaran, e. a. (2013). "RF source-seeking by a micro aerial vehicle using rotation-based angle of arrival estimates". *American Control Conference* (pp. pp. 2581-2587). Washington, DC: IEEE.

Santiago Ontañon, G. S. (2013, 5 (4)). A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft. *IEEE Transactions on Computational Intelligence and AI in games, IEEE Computational Intelligence Society*, pp.1-19.