Two Person Non-cooperative Real Time Strategy Games

Walter B. Gress V

Drexel University

Sunday, May 15 2016

This paper discusses a new way of looking at RTSes, that is, “Real Time Strategy Games.” These games are typically considered more difficult than standard cooperative games and zero-sum games. We approach the problem atomically and using a lookahead mechanism.

# introduction

Non-cooperative games under influence of time are common in the natural world, moreso than standard non-cooperative games where there is a singular situation with a common solution. Time plays a major factor and games that are static in a temporal nature cannot singly represent the many dynamic aspects of most games. In particular, the fact that the states of games over time may fluctuate or change. Also problematic is the combination that exists between multiple, changing rewards and punishments. We take an atomic stance, that the sum of the game and its look-aheads predicts the outcome of a single game.

# notation

First let us briefly discuss notation. A game g is defined as:

|  |  |  |
| --- | --- | --- |
|  | Strategy A | Strategy B |
| Player One | Reward, Punishment | Reward, Punishment |
| Player Two | Reward, Punishment | Reward, Punishment |

# Game look-aheads

As time is an element in RTSes and states change over time, we have decided to allow for look-ahead games. A look-ahead game is like a standard game, but changes over time, chaining multiple games into one game. For example, the RTS game of TRAINING UNITS:

|  |  |  |
| --- | --- | --- |
|  | A – train units | B – Attack Enemy |
| Player One | 1,1 | 1,1 |
| Player Two | 1,1 | 1,1 |
|  |  |  |

Here we score the player training units with a 1,1 because although they are producing new units, they also at the same time are vulnerable for attack.

We score Attack Enemy always with 1,1 because one is doing damage but losing men as well.

However, the question is which is the better strategy in the end? So let’s look ahead on this strategy to a new strategy.

|  |  |  |
| --- | --- | --- |
|  | A – train units | B – Attack Enemy |
| Player One | 2,1 | 1,1 |
| Player Two | 2,1 | 1,1 |

Here we score train units with a 2 instead of a 1 because while player one was defending and training more units, the enemy was attacking and losing units at the same time.

The overall summation of this series of games is

|  |  |  |
| --- | --- | --- |
|  | A- train unites | B – Attack Enemy |
| Player One | 3,2 | 2,2 |
| Player Two | 3,2 | 2,2 |

Player One scores one higher in this series of games, showing its better not to be hasty in attacking.

# GENERAL ALGORITHM

Let us consider a set of games, not turn-based but time based and dynamic. For every unique overall strategy there is a set of atomic games that when combined in the right combination yields to a master game that predicts winner or loser of the games. If this master game matches the win/lose matrix, then the game is said to be complete.

For any set of games , where G is a real time strategy game, = where is the win/lose matrix.

The win/lose matrix is:

|  |  |  |
| --- | --- | --- |
|  | Attack and Lose | Attack and  Win |
| Player One | -1,1 | 1,0 |
| Player Two | -1, 1 | 1,0 |

Therefore the ehain of games, including look-aheads, must sum to the normalized matrix above in order to be a complete game in a RTS.

# SOME SPECIFIC RTS GAMES

**Both Attack at the same time**

|  |  |  |
| --- | --- | --- |
|  | A. Attack and Win | B. Attack and Lose |
| Player One | 2,1 | 1,2 |
| Player Two | 2,1 | 1,2 |

Both players attack each other and our bound to lose men. The loser loses more.

**One collects resources, the other builds a building**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Player One | 1,0 | 1,0 |
| Player Two | 1,0 | 1,0 |

All are advantageous initially. Using a look-ahead matrix however,

|  |  |  |
| --- | --- | --- |
|  | Mine Resources | Built Barracks so better soldiers |
| Player One | 1,0 | 1,0 |
| Player Two | 1,0 | 1,0 |

They are still generally even so we will go to a maximum case.

|  |  |  |
| --- | --- | --- |
|  | Out of Resources | More and better soldiers |
| Player One | 0,1 | 1,0 |
| Player Two | 0,1 | 1,0 |

The aggregate game result is:

|  |  |  |
| --- | --- | --- |
|  | Mine Resources | Build better soldiers |
| Player One | 2,1 | 3,0 |
| Player Two | 2,1 | 3,0 |

We find that it is better to keep building better soldiers than to max out a gold mine right away.

Another is a surprise attack.

**Surprise Attack**

|  |  |  |
| --- | --- | --- |
|  | Attack | Defend |
| Player One | 2,1 | 1,1 |
| Player Two | 2,1 | 1,1 |

The player doing the surprise attack obviously scores higher, but because some of his soldiers are going to be wounded, his score is 2,1. The defender, for defending gets a score of 1, but is punished by a score of 1 for damages.

**Building vs Attacking**

|  |  |  |
| --- | --- | --- |
|  | Build: Better Troops | Attack Immediately |
| Player One | 1,0 | 2,1 |
| Player Two | 1,0 | 2,1 |

Here the scores are equal at this stage. Building better troops has no disadvantage, not counting time or usage of resources. Attack immediately has a score of 2 because you are catching the enemy by surprise and a punishment of 1 since you lose some troops.

However, what is the look-ahead?

|  |  |  |
| --- | --- | --- |
|  | Build better soldiers over time | Attack |
| Player One | 2,0 | 1,0 |
| Player Two | 2,0 | 1,0 |

The attack has balanced out and now attack only has one point for reward. But let’s say you built a barracks and are now building better soldiers. The improvement in soldiers without punishment is a reward of 2.

This makes the overall summation matrix:

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Player One | 3,0 | 3,1 |
| Player Two | 3,0 | 3,1 |

At first this doesn’t seem right. But if you think about it, early in the game if you just start building structures, the opposing player can quickly build a few soldiers and knock you off the map. It’s a close balance, but in this model attacking immediately wins.

|  |  |  |
| --- | --- | --- |
|  | Surprise Attack | Defend |
| Player One | 2,1 | 1,1 |
| Player Two | 2,1 | 1,1 |

The surprise attacking player gets a bonus of 1 while the defending player has a reward of one for defending and a punishment of one for damages. But the battle continues, so the look-ahead is:

|  |  |  |
| --- | --- | --- |
|  | Attack | Defend |
| Player One | 1,1 | 1,1 |
| Player Two | 1,1 | 1,1 |

Now the attacks and defends are balanced, but it is easy to see the surprise attack gives the player and overall edge of 1. This kind of behaviour we would expect.

A more interesting game is when both players clash and attack each other at the same time. Who wins?

|  |  |  |
| --- | --- | --- |
|  | Attack | Attack |
| Player One | 1,1 | 1,1 |
| Player Two | 1,1 | 1,1 |

Without a trick or an edge, the scores zero out and nobody wins. That is, everyone loses.

What happens if both defend?

|  |  |  |
| --- | --- | --- |
|  | Defend | Defend |
| Player One | 1,1 | 1,1 |
| Player Two | 1,1 | 1,1 |

Again, nobody wins, everyone loses, it’s a stalemate.

How about a simple attack and defend matrix?

|  |  |  |
| --- | --- | --- |
|  | Attack | Defend |
| Player One | 2,1 | 1,2 |
| Player Two | 2,1 | 1,2 |

Here the attack strategy dominates. Defense must use 2 to prevent the attack and attack uses one for collateral damage. Here the best defence is a good offense.

# An example of a total strategy for a player

* Build upgrade versus Attack
  + 1,0 vs -1,0
* Build Structure versus Attack
  + 1,0 vs 2,0
* Look-ahead
  + 3,1 vs. 3,0
* Train soldiers
  + 1,1 vs. 1,0
* Look-ahead
  + 1,1 versus 1,0
* Attack
  + 1,1 versus 1,0

Here, the patient player wins.

# CONCLUSION

In this paper we discussed methods to apply to games of real time strategy, a dynamic set of games that are difficult to model. We used the notion of an atomic structure to individual games and the concept that they can be chained together to form an overall strategy which must meet a winning condition. We introduced the idea of look-ahead games, where instead of a single time-slice or general game, the look-ahead games allow us to see some temporal continuity in the nature of a single game.