Establishing Competitive Domination Cycles for Peer-to-peer Game Combat

Owen Makin and Shaun Bangay
School of Information Technology
Deakin University

***Abstract*—Games require balance to be fair and enjoyable. In two player combat settings balance can be achieved by ensuring that both units are equally capable. The possibility of alliances changes the nature of balance when additional players are introduced. One approach to achieving balance is to use a domination loop between agents in a rock, scissors, paper style approach. This paper investigates whether such loops exist within the existing rules of game combat. Search processes within the attribute space of the game units are used to identify loops within an existing game architecture. A dominance metric is used to identify cycles where the victory is achieved by a clear threshold. Cycles with up to 5 players are demonstrated, although larger cycles require more effort to find. Use of models of game play and attribute space search are recommended as a mechanism for balancing games of this format.**

***Index Terms*—games, testing strategies, formal model**

I. Introduction

Balance in strategy games can be difficult to achieve. When only two players are present each can be given a selection of units whose attributes are chosen such that a balance during combat is possible, even though individual units may dominate others. Adding an additional player increases complexity which risks leaving one player with a distinct advantage over the others. This raises the question: is it possible to choose a third unit such that a cyclic domination relationship exists? The existence of such units prevents domination by a single player, allowing the game outcome to hinge on the alliances that form between players, further enriching the game experience.

Cyclic transitive relationships are analogous to the game 'Rock, Scissors, Paper', in which each choice has a distinct advantage in one situation and an equivalent disadvantage in another. Only identical strategies will end in a draw, but the game remains in balance due to the consistency and equal distribution of domination. Previous work [2] devises a measure of balance and dominance between pairs of single units in a simulated combat arena where each have their own sets of attributes. We use this measure to establish the existence of cycles of transitive relationships within this combat setting. Having shown the existence of such relationships for the three player environment we investigate scenarios with higher numbers of players and quantify the extent to which units can dominate others and still retain this property. We hypothesize that the number of players in such a cycle is limited by the amount of dimensions in the attribute space used to characterize the units, since each unit would need to have an advantage in at least one dimension. Our current model uses four attributes to describe each unit.

II. Literature review

Existing approaches to dynamic balance of games are primarily focused on countering human opponents. This is achieved by using a number of different techniques including reinforcement learning [7], [1] and environmental resource manipulation [4]. It has also been approached by limiting options to measure their effect on the game outcome [5]. Metrics that represent the extent of balance have been developed through the analysis of phases in a combat simulation [2]. Although the emphasis in this approach is on achieving a draw in the game, we adapt this approach to identify which one unit will dominate another in the game outcome.

Cyclic relationships have been explored in other fields. Three way cyclic relationships are used in biology to analyze cyclic predator-prey scenarios [3]. They have also been discussed in game theory in regard to whether rock, scissors, paper is a stochastic game [6]. Typically the game is displayed in normal form, but it is argued that the game is instead a stochastic game with discounting due to player's recognition of one another's previous choices.

III. Cycle Identification

We use the single combat scenario that has been previously analyzed [2] and that is described below. This analysis produces an accurate model representing the game outcomes. We use these outcomes to produce a dominance metric which allows attribute space to be searched for sets of units that satisfy the desired cyclic relationship. The search process is presented in the following sections, describing the refinements used to improve the chances of finding suitable results as the complexity of the process becomes apparent.

A. Game Simulation

The combat game scenario is a representation of combat between single units in an RTS game setting. Pairs of units can be spawned that are described by damage, health, speed and range attributes.

The environment is a featureless flat terrain. The AI uses attack and retreat strategies confined to the line between the two units. Units attempt to maintain an optimal distance from their opponent determined by their range attribute. If their opponent is out of range they move in the direction that will

attempt to restore this desired separation. Units are able to fire while they are in range of their opponent. Movement does not effect a unit's ability to fire. Units are always aware of the position of the opponent and are always facing one another. Bullets do not miss and track their opponents.

General settings and details of the starting conditions include:

- Each attribute is limited to a value between 0 and 50 each.
- Bullets travel at a constant speed of 60 squares per second.
- Units can fire 1 bullet per second.
- Units start 30 squares apart.
- Units occupy 2 by 2 squares.
- When one unit is defeated, all bullets on screen are immediately deleted.

B. Prediction Method and Balance Metric

As simulating the game can be time consuming, we use a more efficient predictive model of the game outcomes [2]. This model has been validated against the simulation, but for completeness we further verify any final results against the original game simulation as well. The prediction method breaks battles into three distinct phases:

- Phase 1: Both units are out of range and move in to engage one another.
- Phase 2: One unit has come into range and begins firing, the other may also be able to fire if it is in range. The two units move relative to one another to each best achieve its desired separation.
- Phase 3: The faster unit is within range and ensures that separation between units remain constant. This phase continues until the match concludes

We use a prediction based on a model of the process to analyze units during each phase, to predict which unit wins based on the initial attributes and other game parameters. Within the 3 possible phases there are distinct circumstances that will lead to a victory for one unit. In many of these the victor will win without losing any health. In all other cases the final health value of the winning unit is calculated.

The degree of dominance of the winning unit over the other is calculated by adding their final health scores (which is 0 for the loser) and normalized by dividing by their combined initial health. Under this metric a value of 0 indicates a draw, and the further the value is above 0 indicates higher proportion of health retained by the winner. We further set the sign of the result to indicate a positive or negative value for unit 1 or unit 2 victories respectively. This is necessary to ensure that the domination metric describes the order in which units are related. Thus we define the dominance metric:

$$d(u_1, u_2) = \begin{cases} \frac{h_{1f}+h_{2f}}{h_{1i}+h_{2i}} & \text{if } h_{1f} \geq h_{2f} \\ -\frac{h_{1f}+h_{2f}}{h_{1i}+h_{2i}} & \text{otherwise} \end{cases}$$

Here h_{if} and $h_{ii}$are the final and initial health values for unit u_i. The final health values are provided by the predictive model. We can now define a dominance relationship $>_t$ such that $u_1 >_t u_2$ iff $d(u_1, u_2) > t$. The parameter t represents a threshold factor by which the first unit must clearly dominate the second.

Our goal is to find, for some threshold level t, a set of n units $\{u_1, u_2, \ldots u_n\}$ such that $u_i >_t u_{(i\%n)+1}$ for $i \in \{1, 2, \ldots, n\}$. We search our space of attributes, comprising of damage, health, range, and speed to determine if such a set exists. A gradient descent algorithm using the predictive model of game outcomes to calculate values for the domination metric is used for this search.

Algorithm 1 Sequential Searching Algorithm for a Size 3 loop.

```
def sequential_search (loopSize, threshold)
  for search in range(searchLimit):
      for i in range(loopSize):
          balance (Unit i, Unit (i+1))
      if Unit i <threshold Unit i+1 for all i
          return
```

Where 'balance' is the function that performs a gradient descent search for dominating relationships and arithmetic is modulo loopSize.

C. Cyclic relationship searching

In this case we want each unit of the loop to dominate another distinctly. This requires searching for a dominance value above 0 to indicate a winning outcome. Two methods are employed to search for a cyclic relationship. The first searches sequentially, using a gradient descent approach to go through each combination of units in the loop. The second searches for the attributes of all units simultaneously. This approach selects attributes at random and then compares them to the balance metric to see if they successfully form a loop. Any successful loops will be verified in the game simulation to ensure accuracy.

The sequential method is detailed in algorithm 1. The gradient descent system is based on defining two single units at a time. Unit 1 searches the attribute space to find a complementary match for unit 2. This is repeated for every pair of the units in a loop. It will keep checking pairwise combinations one at a time until it finds a setup where each dominates one another consistently in the desired order.

The advantage of the sequential searching method is that it uses gradient descent which should ensure efficiency. However as discussed in section IV this turns out to be more inefficient than expected. The disadvantage of this sequential approach is that connecting the tail to the head of the cycle cannot be searched by gradient descent without interfering with balance values previously searched. This effectively leaves the last step to chance. We would expect that this process would require a number of repetitions in order to be successful, as a single change to one unit's attributes will effect every other unit.

Simultaneous searching places each unit's attributes in distinct locations within attribute space, and then checks to see if they dominate each other in the right order and by the specified amount. The search method is not based on the gradient descent algorithm, and instead places attributes randomly at each search step. On each step it then checks if the predicted balance metric meets the required threshold value for a successful loop.

	Lp size	Threshold	Lps Found	Time	Verified
Seq*	3	0.1	0	4691 secs	-
		0.3	0	4875 secs	-
Sim*	3	0.1	100	28 secs	89
		0.3	100	317 secs	74
		0.4	100	1689 secs	50
	4	0.4	100	1164 secs	47
	5	0.4	100	19596 secs	45

Seq*: Sequential searching
Sim*: Simultaneous searching

Table I
TABLE OF RESULTS

The strength of this approach is that by searching each point simultaneously, it avoids the issue of connecting the head to the tail and deals with connecting all units from the beginning. This allows for faster searching. The weakness of this approach is that it is largely unguided, meaning that it could be inefficient. However the time spent in the search provides an indication of how frequently such cycles occur.

The simulation is built to support two opposing units in adversarial combat. We search for cycles involving 3 to 5 units. Pairwise combinations of every unit in the cycle are then played through the original game simulation sequentially to verify any successful loops found by either searching method.

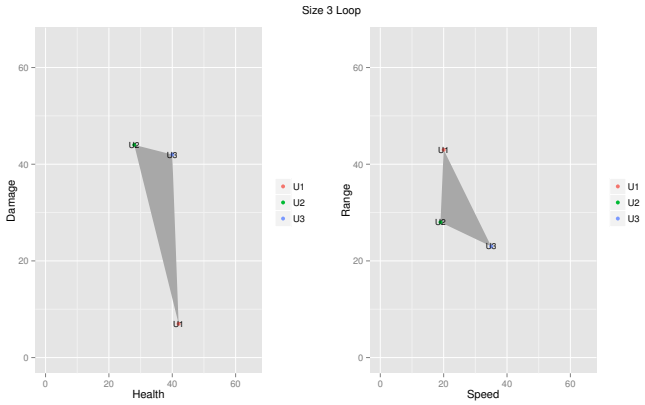

Figure 1. Size 3 Loop

	Damage	Health	Range	Speed
Unit 1	7	42	43	20
Unit 2	44	28	28	19
Unit 3	42	40	23	35

	Advantage Over Opponent
Unit 1 vs Unit 2	Unit 1 has higher range and speed.
Unit 2 vs Unit 3	Unit 2 has slightly higher damage.
Unit 3 vs Unit 1	Unit 3 has higher speed and higher damage.

Table II
LOOP BREAKDOWN TABLE OF FIGURE 1

IV. RESULTS

Each test searches for 100 successfully predicted loops. Each test varies the threshold value or the loop size (the number of units in the cycle). The time for the entire search process to complete is recorded, and the successful loops are saved. These are then verified in the game simulation, which describes outcome information including final health, time of match, bullets fired, bullets hit, and the starting distance. After verification the results are checked and successful loops are recorded. The test results described in table I demonstrate that sequential searching has no success in finding loops at a low threshold, despite sufficient amount of time. Simultaneous searching however is consistently successful in discovering loops. At high thresholds searches take longer, and when verified against the simulation are shown to have increased errors.

An example of a successful loop can be seen in figure 1, which describes a loop of threshold 0.4 discovered by the simultaneous searching method. The domination in this case is based on the differences described in table II.

Sequential searching is time consuming due to the gradient descent search through attribute space for each unit in the cycle. The gradient descent searches neighbouring locations in attribute space carefully to find a balance slope toward a higher value. This becomes problematic for cyclic relationships due to its slow searching for each combination of units.

Simultaneous searching produced better results due to its rapid searching. The simultaneous method checks if a loop is successful once, and then randomly moves to a different point. Compared to the sequential method, which may search multiple neighbouring attributes for each unit for every search, the simultaneous method checks loops rapidly. This speed of searching allows it to consider options faster causing it to find more successful loops, despite being randomly placed. However searching at higher thresholds increases the search time and errors in verification.

The higher rate of error occurs at higher thresholds is due to the level of domination. Higher thresholds during searching will find units that will win while retaining as much of their original health as possible. In order to do this the attributes chosen are often damage to health combinations that allow units to win with a single bullet before the other can fire. Alternatively units raise their speed and range to such a combination that they take no damage. This succeeds in allowing units to dominate clearly, however the margin of error is very low. Because all units in the cycle are chosen to achieve this level of victory small perturbations in these situations cause the prediction to swing the victory to the opposing unit. This creates more errors at higher threshold values when tested against the simulation. This also explains the high error rating when searching for loops containing four and five units. The high threshold that previously caused errors combined with the requirement to search more unit combinations in a single loop increases the amount of errors.

The loops displayed in figure 2 are successfully generated from a threshold of 0.4. These indicate where successful searches have occurred. The triangles representing the loops are stretched on the damage axis, but with little variations between points on the health axis. Changes in the damage attribute need to be more pronounced to have an effect. The same points are evenly spaced on the range and speed dimensions, and the loops tend toward a similar shape and angle. This implies that there may be a consistent spacing ratio between range and speed to achieve cyclic balance.

The size five loop displayed in figure 3 shows a progressive

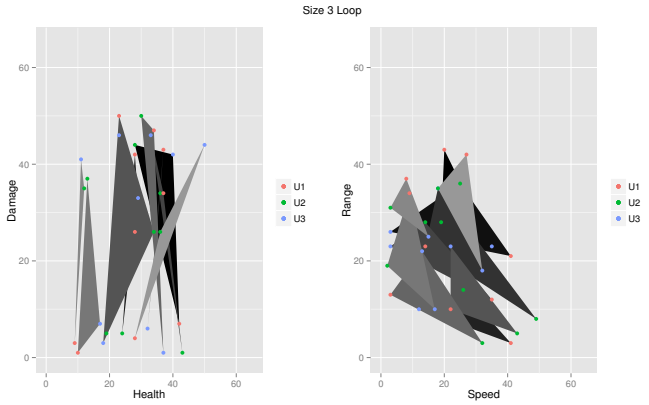

Figure 2. Multiple Size 3 Loops

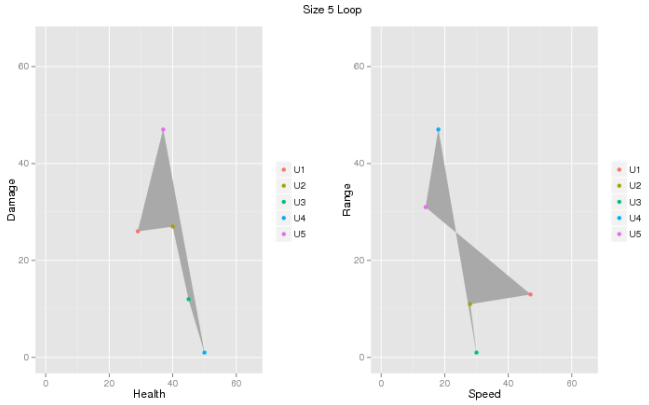

Figure 3. Size 5 Loop

drop in damage between units 3 and 5. Overall many of the points align on a similar angle.

V. Analysis

A. Search methods

The tests show that sequential searching has difficulty discovering cylic relationships, and in our tests had no success. Simultaneous searching produces successful results, despite the error rate for increased thresholds.

B. Constraints on cyclic relationships

We have shown the existence of cyclic dominance loops in a combat game running a simulation with generic combat rules. The existence of even the 3 way cycle suggests a game setting where no single player can have a completely dominant unit - but also one where an alliance of any two players will result in defeat of the third. Such combinations of alliances become more complex as the length of the cycle is increased.

A successful loop of 5 units implies that our original hypothesis was flawed, as it is possible to achieve this despite the limit on dimensions to explore. Analysis of the 5 way loop shows that units found new ways of using attributes by utilizing advantages through the starting conditions. It was expected that advantages would be permanent, and would last the entire match (such as higher damage). However using the starting conditions is a temporary advantage, and therefore is an unexpected result. It also implies that the environment is always used, even in simulated conditions where it is simplified considerably. Other units simply used attributes in advantageous combinations, including the combination of range and speed.

VI. Conclusions

We have shown the domination loops exist in the context of single unit combat scenarios. The predictive model supports our domination metric and straight forward search over attribute space is capable of finding frequent examples of units in a cyclic dominance relationship. The size of the loop is not constrained by the dimensions in attribute space as it is apparent that attribute combinations can produce unexpected advantages. These are enough to allow for larger loop sizes.

Finding loops takes longer with a higher threshold and the error rate is increased with higher thresholds and larger loops. This shows that there may be a maximum threshold limit as the higher the threshold the more difficult a loop is to discover. Increased errors and search time with a loop of 5 show that larger loops are more difficult to discover, implying there may be a potentially maximum loop size for a fixed dimensional attribute space.

The opportunities for exploring unit advantages within alliances becomes an area for further investigation, particularly as the number of units in the sets increase.

VII. Future Work

In future we would like to search for patterns in successful loops and investigate the tendencies observed of centering around particular regions. We would also like to continue expanding the size of loops to confirm the existence of a definitive limit. Automatic generation of dominance loops would support procedural game play generation and can serve as a tool to decrease development time.

VIII. Discussion

Cyclic transitive relationships are common in games for the purpose of balance. It is also common for game designers to carefully construct agent relationships, cyclic or otherwise, to inform the balance and choices available to the player. In either circumstance there is a benefit to analysing cyclic relationships, and measuring balance as a tool to inform game designers.

Researching methods of measuring and manipulating relationships of any hierarchy also has benefits for dynamic games. For example this could be used to generate agents with individual attributes, which could then be automatically adjusted to maintain balance within the world. Furthermore this same approach could be used for other purposes, including virtual wildlife generation. While not directly linked to gameplay, the balance mechanisms could be used for the purpose of biodiversity to create relationships that form a sustainable food chain. This would help create richer virtual worlds.

Another area that could gain from the research is a virtual economy. An economy benefits from continuous cyclic competition. In this regard it is similar to gameplay, and as such the same balance mechanisms could be employed to help regulate and balance the in game market. This is especially valuable when items effect dominance and can be readily purchased. As such agents, items, and pricing could be balanced using the same mechanism.

REFERENCES

[1] G. Andrade, G. Ramalho, H. Santana, and V. Corruble. Extending reinforcement learning to provide dynamic game balancing. In *Nineteenth International Joint Conference on Artificial Intelligence*, pages 7–12, Edinburgh, Scotland, 2005.

[2] S. Bangay and O. Makin. Modelling attribute dependencies in single unit game combat settings. In *5th International IEEE Consumer Electronics Society Games Innovation Conference*, 2013.

[3] M. Frean and ER Abraham. Rock-scissors-paper and the survival of the weakest. *Proceedings of the Royal Society London B 2001*, 268(1474):1323–1327, 2001.

[4] R. Hunicke and V. Chapman. Ai for dynamic difficulty adjustment in games. In *Challenges in Game Artificial Intelligence AAAI Workshop*, pages 91–96, 2004.

[5] A. Jaffe, A. Miller, E. Anderson, Y-E Liu, A. Karlin, and Z. Popovic. Evaluating competitive game balance with restricted play. In *The Eigth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 26–31, 2012.

[6] S. Loertscher. Rock-scissors-paper and evolutionarily stable strategies. *Economic Letters*, 118(3):473–474, 2013.

[7] J. Ludwig and A. Farley. A learning infrastructure for improving agent performance and game balance. In *Annual Artificial Intelligence and Interactive Digital Entertainment Conference*, Stanford University, California, 2007. AAAI.