Target Selection for AI Companions in FPS Games

Jonathan Tremblay
School of Computer Science
McGill University, Montréal
Québec, Canada
jtremblay@cs.mcgill.ca

Christopher Dragert
School of Computer Science
McGill University, Montréal
Québec, Canada
chris.dragert@mail.mcgill.ca

Clark Verbrugge
School of Computer Science
McGill University, Montréal
Québec, Canada
clump@cs.mcgill.ca

ABSTRACT

Non-player Characters (NPCs) that accompany the player enable a single player to participate in team-based experiences, improving immersion and allowing for more complex gameplay. In this context, an Artificial Intelligence (AI) teammate should make good combat decisions, supporting the player and optimizing combat resolution. Here we investigate the *target selection problem,* which consists of picking the optimal enemy as a target in a modern war game. We look at how the companion's different strategies can influence the outcome of combat, and by analyzing a variety of non-trivial First Person Shooter (FPS) scenarios show that an intuitively simple approach has good mathematical justification, improves over other common strategies typically found in games, and can even achieve results similar to much more expensive look-up tree approaches. This work has applications in practical game design, verifying that simple, computationally efficient target selection can make an excellent target selection heuristic.

Keywords

Artificial Intelligence, Video Games

1. INTRODUCTION

NPC companions in modern games are intended to improve player experience by augmenting player capability and acting as a surrogate teammate or co-adventurer. Simple companion AIs, however, often result in the companion demonstrating a poor degree of cooperation [1], failing to recognize and comport with player intention. This can induce frustration and break immersion, especially during time-intensive and stressful situations like combat, where the degree of cooperation can result in significant differences in the outcome of the battle.

In this work we explore the challenges of cooperation in terms of *target selection* for FPS games. Optimal combat in FPS games relies heavily on AI team members selecting appropriate targets, matching or complementing player choices. Although this is a complex problem in its full generality, for simple situations the problem can be feasibly expressed and analyzed formally. Using a mathematical model, we first show that an intuitive, "threat-based" selection strategy has theoretical justification for being optimal, at least within a simplified context. We then compare the results of various common heuristics for target selection in a number of non-trivial FPS test environments, showing the relative performance of using a closest, strongest, weakest, or highest threat enemy, as well as the effect of mimicking the player's strategy. This analysis demonstrates relative performance of these heuristics, and in conjunction with a further comparison of threat to the optimal results of a full look-up tree search, shows that the theoretical basis for a threat heuristic remains valid in more complex environments.

An efficient means of optimizing AI combat strategy is important in FPS games, as it enables companions to more closely model the behaviour of clever teammates, eliminates the need for extra-narrative control of companions during combat, and gives another means of controlling game difficulty. More specific contributions of our work include:

- We derive a simple but justified target selection strategy through mathematical analysis of a reduced problem space. Our approach is similar to prior analytical work by Churchill *et al.* and Furtak *et al.* [3, 2], but considers scenarios more common to FPS and Role-Playing Games (RPG) contexts. This analysis verifies that a "threat ordering," prioritizing enemies based on two dimensions (health and attack), is theoretically optimal.
- Through experimental analysis of several representative scenarios, we show that threat ordering outperforms other commonly used target selection strategies, even when considering the additional complexity of real-time gameplay, probabilistic hit and damage, character movement, and physical occlusion.

2. BACKGROUND & RELATED WORK

Our work is motivated by trying to improve the integration of NPCs within the player game experience, in which context we specifically concentrate on the combat target selection problem. Below we introduce and discuss related work in both areas.

Human Player Game Experience - Considerable previous work exists in trying to understand the gaming experience [11]. For the purpose of this work, human experience is modelled in terms of maximizing an economic utility function, defined as the total team health. This maximizing function is translatable as finding the highest rewarding Nash equilibrium [8] in terms of selecting strategies. With respect to our game set-up, the player has to work with an NPC as a companion and trust her to make the right choice. When human players do not trust each other they never reach an optimal equilibrium [4], and thus to maximize the value of a companion a player needs to be able to trust her to make appropriate decisions.

This problem of trust is found in many modern FPS and RPG games. In *Skyrim*, for instance, the NPC companion often exhibits sub-optimal strategy—targeting inappropriate enemies, entering battle too quickly, and generally interfering with the player's combat intentions. This results in less effective combat, or even the death of the companion. After losing trust in the companion, the player instead adopts a sub-optimal strategy, such as keeping the companion out of the combat [5].

Target Selection Problem - The general target selection problem consists of two teams attacking each other, with each entity selecting an opponent to fight. The goal is to maximize the total team health at the end of the combat. We find this problem in FPS, real-time strategy, adventure, and other games. Work by others has closely examined the case of 1 player against n enemies, showing that the problem of minimizing health loss for even a single player is NP-Hard [3]. In our case we are interested in the case of a player and her companion in a FPS or RPG scenario, a (small) team vs. team approach, which has mainly been previously addressed through look-up tree search.

Look-up tree search consists of exploring the reachable state space of the game. The naïve way to do it would be to explore every possible strategy at every state, reaching all possible end-game states [7]. From there the optimal choice is the one propagating back from the leaf with the best solution. Even for small scenarios, however, exponential growth in the size of the state space makes such an exhaustive search unrealistic in practice, at least within the context of real-time commercial games.

Look-up tree search typically assumes that players play in discrete turns. In a real-time environment this does not hold, as entities take time to perform actions and may do multiple actions between opponent moves, magnifying the branching factor in a tree search. Churchill *et al.* explored ways to reduce the space of exploration in real time strategy games by using an alpha-beta search variant [2]. They were able to solve an 8 vs. 8 combat using a game abstraction model that reduces the search space by focusing on important game features. Although this was done in real-time (50ms), the relative cost of this approach is still expensive for the rapidly changing context of FPS games, where high frame-rates are paramount, and CPU is limited.

Heuristic valued approaches offer a more efficient solution by attempting to approximate enemy value through a weighted function of observed data. The enemy with the highest aggregate score is then selected as a target. In the game *Uncharted 2,* for instance, they used a target selecting system that computed, for each enemy, a weighted function of distance, cover, whether the enemy shot the player last time, who the enemy is targeting, close-range radius, and so on [9]. In general, their NPC would try to target different enemies by adding a negative weight when multiple entities target the same enemies, while staying on that enemy using a sticking factor. This approach can be effective, but as a complex and highly heuristic measure it must be closely tuned to a given context, and does not necessarily result in overall better combat results.

Finally, we note that many games offer some level of *manual* control over target selection. In *Drakensang*, a player may override a companion's target choice, and direct her toward a specific enemy. Such extra-narrative control avoids the need for optimal target selection, but requires invoking an out-of-game interface, and if frequently necessary reduces companions from teammates to tools. Middle ground is found in games such as *Dragon Age 2,* which lets the player choose very high-level strategies for her companion(s), toggling between *aggressive* or *defensive* mode. This reduces the interaction complexity by hiding detail, but also makes it less obvious to the player what the companion will do, without giving confidence that the best results will be achieved.

3. MOTIVATING ANALYSIS

The target selection problem exists in general within *(Basic) Attrition Games*, games wherein two sides, players and enemies, seek to eliminate the other. It has been previously shown that solving Basic Attrition Games is exponential (i.e., BAGWIN $\in$ EXPTIME) [3], while the decision problem is PSPACE-hard, and is therefore not feasible in real-time. Rather than solve the general form of the problem, we aim instead to explore the faster heuristics that can easily be computed in polynomial time and which are typically applied in a FPS setting.

The goal of such heuristics is to find an enemy attack order that maximizes total remaining player health without evaluating the entire combat tree (state space). A naïve heuristic might be to have all players attack the enemy with the lowest health, or target the enemy with highest attack, or even attack the enemy with the highest health. However, any of these obvious heuristics will fare poorly under different scenarios. For instance, attacking the enemy with the lowest health is a poor choice when there is an enemy with only slightly greater health but much greater attack power. Intuitively, we should target enemies that are easy to kill and which may cause lots of damage first, and enemies which are hard to kill but induce low player damage last. The former represent high *threat* enemies, while the latter have less priority. Below we demonstrate that this simple model actually has a well justified mathematical basis, describing first a discrete time context, and then extending the result to a more realistic real-time environment. Note that this formulation builds on the mathematical analyses found in work by others, but deviates in order to accommodate our context and goals.

Discrete Time - The following combat scenario will be used to define our basic attrition game. We begin with a set P of players (1 human and some companions) that are fighting a set E of enemies, where $|P| = n$, and $|E| = m$. Each entity $p \in P$ and $e \in E$ has attack a and health h where $a, h \in \mathbb{N}^+$. Fighting occurs in rounds, where the players and enemies each select an opposing entity to attack. A player's attack is resolved by deducting p_a from an enemy's health e_h. If this leaves $e_h \leq 0$, the enemy is killed. Players hit first, meaning that a defeated enemy will not attack during the round in which it is killed. Any attack exceeding the health of the target is wasted. The game ends when either all players or enemies have been killed. Enemies will choose their targets randomly, and for convenience, $p_h \gg e_a$, simulating role playing games where players typically have an advantage in order to ensure continued gameplay.

An enemy will deal damage each round until it is dead, so health savings for the player are maximized when the enemy is killed as quickly as possible. We express the maximum health savings $S(e)$ for an enemy e as

$$S(e) = e_a \cdot (T_{actual} - T^{\alpha}(e)) \tag{1}$$

where T is the length of combat, and $T^{\alpha}(e)$ is the minimum length of time needed to kill e. Unfortunately, T_{actual} and $T^{\alpha}(e)$ are variable based on target assignment and the degree of *overkill* (when $p_a > e_h$). We can lower-bound T_{actual} as

$$T_{actual} \geq \left\lceil \frac{E_h}{P_a} \right\rceil \tag{2}$$

where P_a is the players total attack, and E_h is the enemies total attack. However, the possibility of overkill means the actual combat length may exceed T. For instance, consider the sitation where $n = 1$. It will take at least m turns to defeat m enemies, regardless of E_h and P_a. Instead, we approximate T_{actual} using

$$T \simeq \sum_{e \in E} \left\lceil \frac{e_h}{P_a} \right\rceil \tag{3}$$

This provides a reasonable estimate since it accounts for the number of enemies. It does allow for overestimation of T_{actual} (e.g., in the case where every player can kill any enemy in a single attack and $n \geq m$), but this overestimation turns out to be necessary. Consider the situation where $T = T^{\alpha}(e) = 1$. This means that $S(e) = 0$ for all $e \in E$. If there is overkill, T_{actual} could be greater than T, yet our savings estimates are all zero, providing no guidance. Overestimating guarantees that we maintain information about enemy attacks and thus can still differentiate targets even in the presence of overkill.

In Eq. (1), we also need $T^{\alpha}(e)$, which is given by

$$T^{\alpha}(e) \geq \left\lceil \frac{e_h}{P_{e,a}} \right\rceil \tag{4}$$

where $P_{e,a}$ is the total attack of the subset of players targeting e. We use this subset of attack values to reduce overkill. If $p_a > e_h$, then it would not make sense to consider all P_a, so using this reduced attack value allows us to take into account the effects of spreading out attacks among enemies. With values for T and $T^{\alpha}(e)$, we now expand Eq. (1) to get our final equation for savings

$$S(e) = e_a \cdot \left(\sum_{e \in E} \left\lceil \frac{e_h}{P_a} \right\rceil - \left\lceil \frac{e_h}{P_{e,a}} \right\rceil \right) \tag{5}$$

Target selection proceeds by summing $S(e)$ over all enemies for every possible pairing, C, of P on E, which has m^n possibilities since an enemy can be targeted by more than one player:

$$\max_{c \in C} \left[\sum_{e \in E} S(e) \right] \tag{6}$$

The pairing c that gives us the maximum savings is our target selection. Evaluating Eq. (6) takes $O(m^n)$, and requires no manipulation or transformation from the basic parameters of the problem. As combat proceeds, we reevaluate each round to determine the optimal savings given that enemies have had their health reduced and may have died.

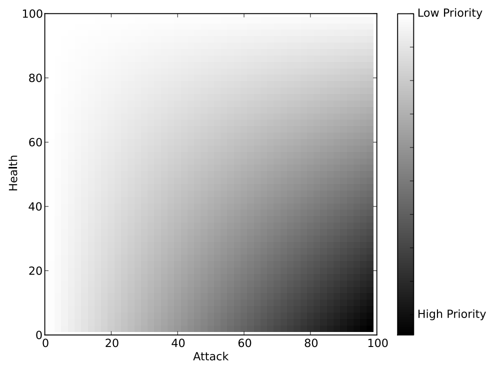

Figure 1: Plot of equation (7), showing threat order for different combinations of enemy health and attack

Real-Time Problem - The real-time formulation allows for entities to evaluate the best target at every moment. This means that players can react to changes in game state, such as an enemy dying, and change their attack instead of wasting it. By eliminating the possibility of overkill, Eq. (2) becomes exact. Thus, we can evaluate exactly which enemy offers the highest savings. The priority of all targets decreases in linear proportion to time, and so relative priority ranking remains constant over time. Eliminating targets in priority order thus guarantees an optimal outcome in real-time scenarios. We reach the same conclusion as [6], and find that changing targets is suboptimal as it guarantees that the optimal savings will not be reached. In general, all players should always be attacking the same enemy.

Using this knowledge we can rewrite Eq. (5). Here, $P_{e,a}$ is equal to P_a as the players will all pick the same enemy. Using that knowledge we can drop P_a and get

$$\max_{e \in E} \left[e.a \cdot (\lceil E_h - e_h \rceil) \right] \tag{7}$$

What this means is that targets combining low health and high attack are preferred. We call this strategy *threat ordering.* Figure 1 plots Eq. (7) for varying combinations of e_a and e_h while E_h is kept constant. The scale on the right shows relative threat order for different enemy statistics.

4. SIMULATION

A theoretical explication necessarily abstracts over many details, such as variant firepower, entity movement, physical occlusion (cover), and so forth. It is possible that in more complex contexts the threat-based heuristic we justified and found mathematically optimal is not in fact much better than other common and even more trivial heuristics that greedily focus on enemy health, proximity, or other one-dimensional factors. We thus here explore the relative value of these heuristics in practice by applying them to a game context representative of typical FPS games, built using the Unity 3D game development framework. In this we consider 4 common targeting strategies within 4 varied scenarios (sets of enemies), as well as the impact of imperfect information due to the existence of cover, and the result when companion and player strategies are perfectly aligned.

Simulation Set Up - The simulation consists of a basic Third-Person Shooter game where the player has to explore a level by reaching goal locations while eliminating encountered enemies. The game is over when the player has eliminated all enemies and reached all goals. The player is accompanied by a single companion; the companion follows the player around and will engage combat when she sees an enemy. Every NPC in the game is given a health and attack value.

The human player in the game is played by an NPC in order to simplify testing. We are interested in the influence of the companion's behaviour over the outcome of the game, and by simulating the human player with artificial intelligence we assure that her behaviour will not be evolving over time and she will act in the same manner for all simulations. In the simulation the human player's behaviour is described using a behaviour tree. She will explore the space to reach all goals and engage combat with every enemy she sees. Enemy detection is also facilitated by sound—when she hears gun fire, she will investigate the situation by walking towards the sound. The level is designed in such way that all enemies are near goals to ensure the occurrence of combat.

The companion's behaviour is supportive; she will closely follow the player during exploration. She will engage in combat with every enemy she sees and, like the player, is aware of sounds. Enemy behaviour reflects game industry standards; if NPCs do not see any enemies, they will patrol around using pre-determined waypoints. When they hear fire they will move towards that position. If they see an enemy they will engage in combat by firing upon the closest target. Any agent in the simulation will shoot for 2-3 seconds and will then move to the left or the right; this behaviour simulates dodging.

Target Selection Strategies - In order to fully test NPC targeting action, we implemented 4 different strategies inspired by modern FPS games. In each case, the selection is constrained to *visible* enemies, and the human player uses the *threat ordering* strategy.

- *Closest* strategy will pick the closest enemy using Euclidean distance.
- *Highest Attack* strategy will pick the enemy with the highest attack.
- *Lowest health* strategy will pick the enemy with the lowest health .
- *Threat ordering* strategy will pick the enemy that has the highest priority according to equation 7.

Note that the closest strategy is strongly affected by the level design. Through careful placement of enemies, a designer could set up the level in a way that the companion choice will match other strategies, at least initially. We thus randomize enemy starting positions, and so closest acts more like a random selection.

Scenarios and Levels - Four scenarios inspired by actual game situations were developed in order to compare the different strategies.

- *Uniform* scenario is composed of six enemies with the same attack and health value.
- *Boss* scenario is composed of five enemies with the same attack and health value, and a boss with high attack and health.
- *Medley* scenario is composed of two enemies with low health and high attack value, two enemies with high health and low attack, and one enemy with medium-high attack and high health.
- *Tank* scenario is composed of five enemies with the same attack and health value, and an enemy with very high health value but only slightly higher attack.

Each combination of scenario and strategy was tested in two environments. *Simple level:* an obstacle-free, open field with no geometry blocking NPC vision. This is an optimal situation for our threat ordering strategy, as it was designed with access to perfect information in mind. *Pillar level:* a high-occlusion environment, with pillars blocking vision. Vision constraints increase the problem complexity in ways not accounted for in Eq. 7, limiting target choices (in our experiments entities just pick a new target when the lose sight of their initial target), and making movement time a significant cost.

Figure 2 shows a top-down view and in-game, playtime screen-shots of the pillar level (simple level is the same with no pillars). Red circles represent the enemies, the blue circle is the human player and the green circle the companion.

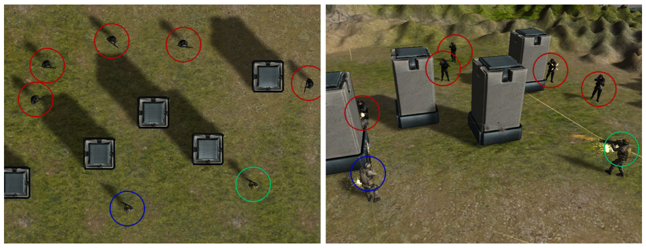
Figure 2: Top-down view and in-action playtime screenshot of the pillar level

A final set of experiments was also done having the companion make the same target choices as the player (**Mimic Behaviour**), as the player uses different target selection strategies in the *simple level.* This experiment gives insights into how the player's behaviour could be destructive when the companion does not make her own decisions.

5. RESULTS

For each combination of strategy, scenario, and level, and again for the mimicking situation, we ran 31 simulations. This was sufficient to show trends in the data, while still resulting in a feasible experimental approach. From the data we plotted average final team health and standard deviation. Results can be seen in figures 3 to 14, where we plot player+companion health over the duration of combat (in seconds) for the different combinations.

Simple Level - Figure 3 shows results for all the strategies given the uniform enemy scenario. Since the enemies are identical in terms of attack and health, any target is a good target. Therefore any strategy is good as long as the players do not deviate between the enemies, giving us a baseline for variance and sanity check for our simulation. It is interesting to point out that with respect to our theoretical justification, all enemies would sit at one point in figure 1, emphasizing the lack of need for specialized strategies.

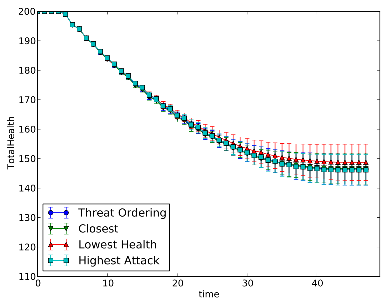

Figure 3: Simple Level Uniform

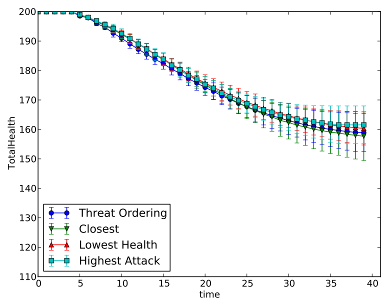

Figure 4: Pillar Level Uniform

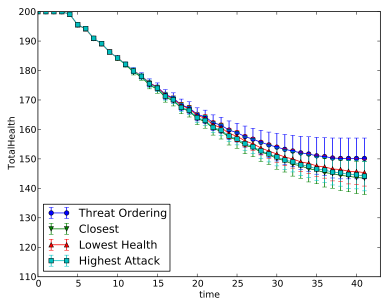

Figure 5: Mimic Level Uniform

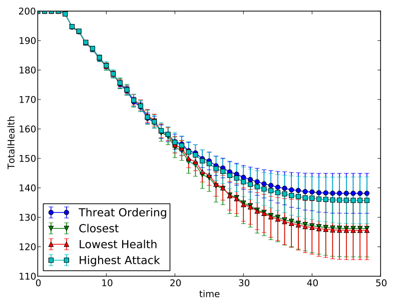

Figure 6: Simple Level Boss

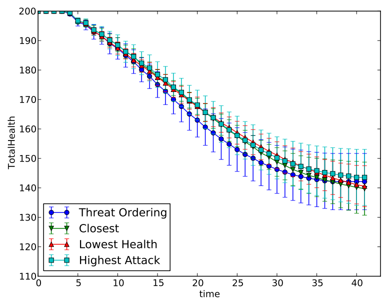

Figure 7: Pillar Level Boss

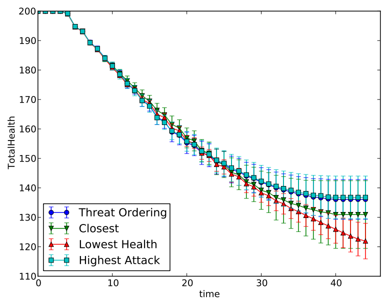

Figure 8: Mimic Level Boss

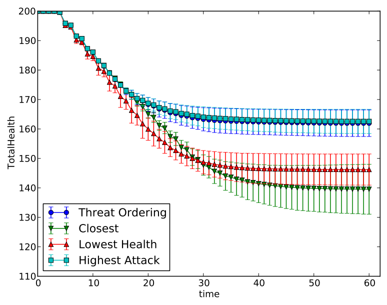

Figure 9: Simple Level Medley

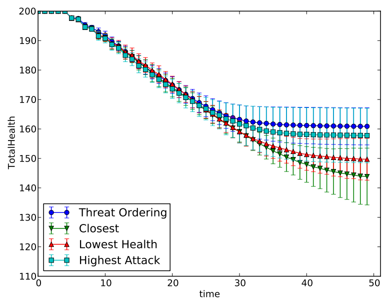

Figure 10: Pillar Level Medley

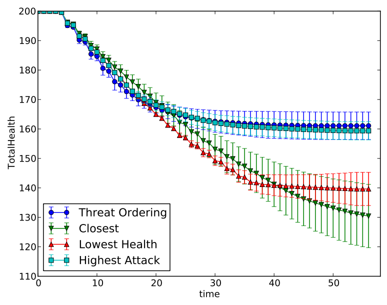

Figure 11: Mimic Level Medley

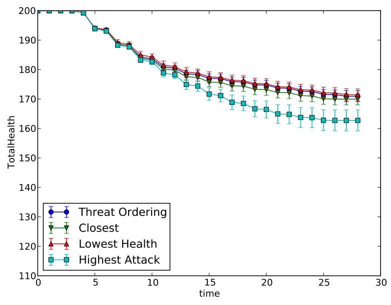

Figure 12: Simple level Tank

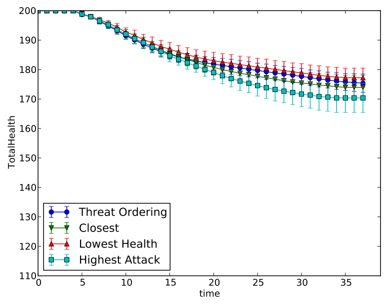

Figure 13: Pillar Level Tank

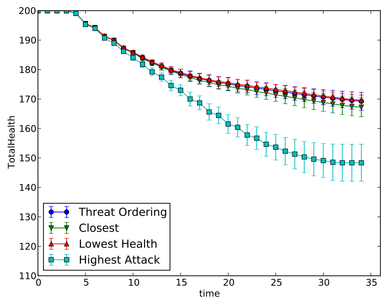

Figure 14: Mimic Level Tank

In the boss and medley scenarios, figures 6 and 9, highest attack outperforms lowest health as a strategy, and matches our threat ordering approach. This is not always an ideal choice, however, as shown in the tank scenario, figure 12. In this case highest attack is actually the worst strategy; it picks the tank (high health enemy) first because it also has slightly higher attack, and thus spends a long time receiving non-negligible damage from the other enemies. Threat ordering, as well as the lowest health strategy, do not fall into the same trap, prioritizing instead enemies that are more quickly eliminated and thus reducing total health loss.

Pillars - The addition of pillars to the level design reduces visibility, preventing entities from seeing all targets in general, and dynamically changing the set of available targets as entities move.

In general having subsets of enemies adds more noise to the simulation, and results are largely similar to the simple level, but with larger variance. This is evident in the uniform scenario, figure 4, and especially in the medley experiment, figure 10. Most evident in the uniform scenario, however, is that total health tends to be higher in the pillar versions. This is due to the enemies queueing behind each other because of the limited room between pillars, thus reducing their visibility and allowing only a subset to shoot at the players. With less shots fired at them, overall player health ends up greater, and this argument holds for every pillar scenario.

For the tank level, figure 13, we see a small but interesting change in the relative difference between strategies. The highest attack strategy is still the worst, but the gap between it and other strategies is not as big as in the simple level scenario. Repeated occlusion in the pillar level reduces the ability of the companion to stay focused on their sub-optimal choice, ameliorating the otherwise negative impact of this strategy. This is further verified by measuring and comparing the number of times the companion targets an enemy with respect to the number of times she would target the ideal target for her strategy; in this case the companion is able to choose her intended but sub-optimal target only around 1/3 of the time.

Companion Mimicking the Player - Since optimal theoretical solutions suggest that concentrating attacks on one enemy is optimal, a trivial strategy for companions is to simply mimick whatever the human player does. The success of this approach, however, depends very much on the strategy the player chooses. The medley scenario, figure 11 betters our understanding of the context. With a more independent companion we had at least one player selecting an optimal choice, decreasing the impact of any wrong choices on part of the other player. With mimicking, however, a wrong choice ends up multiplying the negative impact, and sub-optimal strategies such as lowest health and closest end up with dramatically lower team health values. This is also evident in the highest attack strategy of the tank scenario, figure 14.

Of course when the player is an expert at picking the right target, mimicking performs well, as both players cooperate and use good strategies. However, given that this will also occur if the companion makes an independent choice to use threat order, and that will still imply generally better outcomes if the human player is not an expert, mimicking seems like an overall poor approach. We note that this is not necessarily the case for every game or game context, and mimicking has been explored and shown to an effective approach in more complex contexts where learning from the player is worthwhile, such as in fighting combat games [10].

Look-up Tree Search - The heuristics we examine offer simplicity and efficiency advantages over look-up tree searches, but even the overall best, threat ordering, due to its inherent abstraction is not always necessarily optimal. We thus also compare performance with search based targeting to see how far from optimal threat ordering ends up being. For this we used a non-graphical, discrete-time simulation, allowing us to explore a large number of scenarios, and avoiding any concerns with perturbing the real-time simulation. Note that even this reduced problem is still NP-hard as shown by Furtak *et al.* [3].

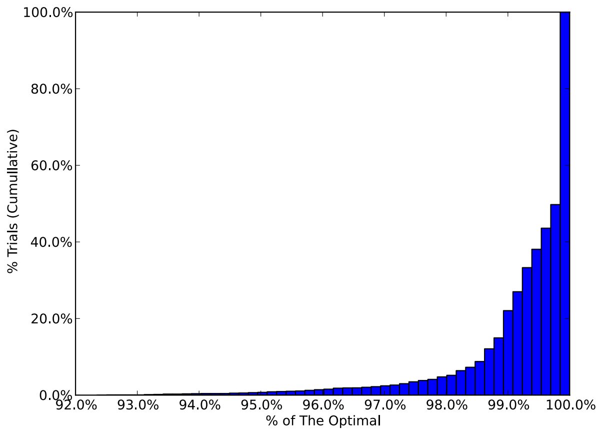

Figure 15: Cumulative histogram, showing how close to optimal threat ordering performs (discrete context).

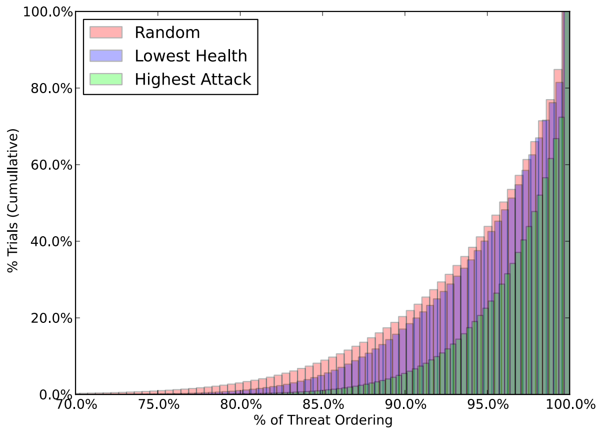

Figure 16: Cumulative histogram, showing how highest attack, lowest health, and random targeting fare in comparison to threat ordering. Note that the x-axis scale differs from figure 15.

We ran 500000 simulations in the discrete world with 2 players against 2 to 5 enemies. We compared a full look-up tree search with the discrete form of the threat ordering heuristic, given as equation 6. The players in this simulation have attack 3 and health 500 each, with enemy attack varying from 1 to 10 and health from 1 to 9. Although these are fairly arbitrary values, there exists 6 billion ways to arrange the enemies with a large variety of difficulty. Results are shown in figure 15. We can see that threat ordering finds the optimal solution 50% of the time, is usually within around 1% of optimal, and never results in a total team health less than 8% of the optimal.

Figure 16 compares the behaviour of highest attack, lowest health, and random targeting strategies to threat ordering at each trial; this experiment used 50000 simulations, and 2–10 enemies (other parameters are the same). Note that in this discrete simulation we do not represent geometric constraints, and so random replaces closest. In no cases did these two strategies exceed threat ordering, but highest attack is clearly the better of the two, matching threat ordering about 25% of the time, and being within 50% of threat ordering over 97% of the time. Lowest health, however, only barely improves over random.

These discrete, analytical results largely mirror the results shown in the more complex, real-time data given in figures 3 to 14. In general, focusing on high attack enemies is most important, eliminating weaker individuals is next, and a weighted combination of these results is close to optimal prioritization. Physical proximity has relatively little relevance, although this is likely also an artifact of our combat simulation; as future work it would be interesting to see how close-combat versus distance weapons alter the weighting of proximity.

6. CONCLUSION

Optimal solutions to enemy target selection in combat games are complex, and ideally solved through expensive state-space searches that are not practical in game contexts. Designers thus frequently resort to simple, but fast and easy to compute heuristics such as choosing the closest enemy, strongest enemy, mimicking the player, and so forth. In this work we explored and compared several such common heuristics, showing that a slight variant (threat ordering) can be mathematically justified, and also performs notably better than other simple heuristics in realistic simulation. We also compared this result to a simplified, but exhaustive state space analysis to verify that this approach is not only relatively better, but also demonstrably close to the theoretical optimum. Understanding and validating these kinds of targeting heuristics is important in terms of building interesting and immersive gameplay where companions behave sanely and can perform effectively.

There are a number of interesting extensions to this work. Combat complexity, for instance, is often increased in RPGs by giving characters a variety of special attacks, greatly limited attack resources (as with magic), or by introducing specific enemy weaknesses or strengths. We are interested in seeing if threat ordering or other simple, perhaps higher dimension heuristics would still perform well in such complex environments. In more long-lasting or difficult combats, the availability of defensive cover and resource restoration adds even more factors that should be considered in optimizing combat behaviour [12]. Our main interest, however, is in exploring how to improve the value of companions to the player by maximizing their utility and ensuring appropriate, human-like combat behaviours, as well as in using the flexibility of companion choices to help games adapt to different player skills.

7. ACKNOWLEDGEMENTS

This research was supported by the Fonds de recherche du Québec - Nature et technologies, and the Natural Sciences and Engineering Research Council of Canada.

8. REFERENCES

[1] S. Bakkes, P. Spronck, and E. O. Postma. Best-response Learning of Team Behaviour in Quake III. In *Workshop on Reasoning, Representation, and Learning in Computer Games*, pages 13–18, 2005.

[2] D. Churchill, A. Saffidine, and M. Buro. Fast heuristic search for RTS game combat scenarios. In *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2012.

[3] T. Furtak and M. Buro. On the complexity of two-player attrition games played on graphs. In *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2010.

[4] J. K. Goeree and C. A. Holt. Ten little treasures of game theory and ten intuitive contradictions. *American Economic Review*, pages 1402–1422, 2001.

[5] F. W. P. Heckel and G. M. Youngblood. Multi-agent coordination using dynamic behavior-based subsumption. In *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2010.

[6] A. Kovarsky and M. Buro. Heuristic search applied to abstract combat games. In *Advances in Artificial Intelligence*, Lecture Notes in Computer Science, pages 66–78. Springer Berlin Heidelberg, 2005.

[7] I. Millington and J. Funge. *Artificial Intelligence for Games.* Morgan Kaufmann, second edition, 2009.

[8] N. Nisan, T. Roughgarden, É. Tardos, and V. V. Vazirani, editors. *Algorithmic Game Theory.* Cambridge University Press, 2007.

[9] B. Russell. The Secrets Of Enemy AI In *Uncharted 2*. `http://www.gamasutra.com/view/feature/134566/the_secrets_of_enemy_ai_in_.php`, 2010.

[10] S. Saini, P. Chung, and C. Dawson. Mimicking human strategies in fighting games using a data driven finite state machine. In *Information Technology and Artificial Intelligence Conference (ITAIC), 2011 6th IEEE Joint International*, volume 2, pages 389–393, 2011.

[11] K. Salen and E. Zimmerman. *Rules of Play: Game Design Fundamentals.* The MIT Press, 2003.

[12] Y. Shi and R. Crawfis. Optimal cover placement against static enemy positions. In *Proceedings of the 8th International Conference on Foundations of Digital Games*, FDG 2013, pages 109–116, 2013.