UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Rafael Melo Silva

Computação evolutiva aplicada a Jogos no estilo Presa-Predador

Uberlândia, Brasil

2019

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Rafael Melo Silva

Computação evolutiva aplicada a Jogos no estilo Presa-Predador

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Ciência da Computação.

Orientadora: Profa. Dra. Christiane Regina Soares Brasil

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2019

Rafael Melo Silva

Computação evolutiva aplicada a Jogos no estilo Presa-Predador

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Ciência da Computação.

Trabalho aprovado. Uberlândia, Brasil, 18 de dezembro de 2019:

Profa. Dra. Christiane Regina Soares Brasil
Orientadora

Profa. Dra. Márcia Aparecida Fernandes

Profa. Dra. Maria Adriana Vidigal de Lima

Uberlândia, Brasil
2019

Agradecimentos

Com o fim desse trabalho se encerra também toda a jornada que vivi nesses últimos 5 anos desde que enfrentei o desafio de ingressar na graduação. Durante esse período vivi muitas batalhas mas também muitas alegrias e todas elas me marcaram muito, assim como as pessoas que participaram delas, seja ao meu lado no dia-a-dia ou de longe.

Agradeço principalmente a minha família, a minha irmã Lara (*in memoriam*) , que de onde quer que esteja saiba que a conquista desse sonho da graduação não é apenas minha mas nossa, aos meus pais Josina e José que tanto me ensinaram e que sempre batalharam pra que nunca me faltasse nada e tivesse a melhor educação possível.

Em especial, agradeço ao meu amigo Vitor que me acompanha já a tanto tempo e que me acompanhou de perto no desenvolvimento desse trabalho com todo o apoio possível, agradeço por toda a paciência em aguentar meus momentos difíceis e estar sempre ao meu lado.

Agradeço a minha companheira de casa e de vida Katherinne que me aturou todos os dias nesses últimos anos e que sempre foi uma guerreira ao meu lado, principalmente nos nossos momentos mais difíceis.

As minhas queridas amigas Aléxia, Luara e Marília, que me acompanharam em tantas aventuras e momentos inesquecíveis nessa graduação. Ao meu pequeno grande amigo Samuel, que mesmo de longe sempre esteve ali por mim. Ao Guto, Scavoni e Juliana meus companheiros da Computação, com quem compartilhei tantos momentos sofridos com as matérias mas também momentos ótimos. A Laís e Bruna, duas mulheres incríveis que admiro muito. A minha amiga *star guardian* Jéssica. A Bianca, que entrou na minha vida como uma nova companheira de apartamento mas se tornou uma grande amiga.

Agradeço também as minhas amigas araxaenses, que o destino colocou uma distância física mas que sempre estiveram em meu coração, aos meus companheiros do Juntos, com quem vivi tantas lutas, aos meu amigos da Neppo, que me acompanharam nesse último ano, aos colegas da 55 e a todos os outros amigos que passaram pela minha vida na UFU e em Uberlândia.

Por fim, sou grato a Universidade Federal de Uberlândia e a Faculdade de Computação por todo o conhecimento e experiências proporcionados. A todos os docentes com quem tive o prazer de ter aula, e principalmente à minha orientadora Christiane, que encarou não somento o desafio de me acompanhar na minha Iniciação Científica mas também nesse Trabalho de Conclusão de Curso, agradeço por todo o apoio, principalmente naqueles vários momentos em que fiquei completamente perdido e sua orientação conseguiu me mostrar um caminho.

If our existence is everyone else's chaos, so be it.
Xayah, League of Legends.

Resumo

O modelo Presa-Predador representa um problema clássico da literatura. Esse modelo é caracterizado pelos objetivos de captura e fuga, e pode ser encontrado em vários jogos eletrônicos, desde os mais antigos aos mais modernos. Atualmente existe por parte dos jogadores um desejo de que os jogos sejam mais inteligentes e adaptáveis. Neste contexto, a Inteligência Artificial tem sido amplamente aplicada e o uso da Computação Evolutiva e, principalmente, dos Algoritmos Genéticos tem crescido. Neste trabalho é realizado o desenvolvimento de um Algoritmo Genético e de um jogo com modelagem Presa-Predador, com uma relação de cadeia alimentar entre os personagens. O objetivo do trabalho é que a aplicação do Algoritmo Genético para melhorar os atributos dos *Non Player Characters* gere comportamento adaptativo nestes personagens durante o jogo.

Palavras-chave: Presa-Predador, jogos, *Non Player Characters*, Inteligência Artificial, Algoritmos Genéticos.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AE	Algoritmo Evolutivo
AG	Algoritmo Genético
CE	Computação Evolutiva
CPD	*Compressed Path Database*
EEs	Estratégias Evolutivas
IA	Inteligência Artificial
NPC	*Non-Player Character*
PG	Programação Genética
PE	Programação Evolutiva
RN	Rede Neural
RPG	*Role-Playing Game*

Sumário

1 Introdução

No atual momento no desenvolvimento de jogos eletrônicos existe um enfoque em jogos com melhores *gameplays*, de forma que estes possuam uma jogabilidade mais inteligente (CROCOMO, 2008). No entanto, o cenário não foi sempre este pois durante muito tempo o foco foi buscar realismo nas imagens e cenários gerados, de forma a obter gráficos e efeitos melhores (YANNAKAKIS, 2005). Quando foi atingido um alto nível de realismo, houve uma redução da busca por esse objetivo e uma percepção de que jogadores poderiam desejar jogos com melhor jogabilidade (CROCOMO, 2008).

A Inteligência Artificial (IA) se mostrou como uma solução para esse objetivo de melhorar a *gameplay*, sendo que suas técnicas podem ser aplicadas de várias maneiras. A aplicação da IA na melhoria da inteligência dos *Non Player Caracthers*(NPCs) [1] têm se mostrado promissora (JANG; YOON; CHO, 2009). Dentre as diversas técnicas de Inteligência Artificial, as Máquinas de Estado Finitas têm sido as mais utilizadas na inteligência dos NPCs por serem de fácil implementação e com boa performance computacional. O ponto negativo destas Máquinas de Estado Finitas é a necessidade de mapear os estados que podem ocorrer durante o jogo, podendo essa quantidade de estados ser gigantesca (SWEETSER; WILES, 2002). Outros exemplos de técnicas que têm sido utilizadas são *Scripting*, Aprendizado de máquina, Árvores de decisão, Redes neurais e Computação Evolutiva (CROCOMO, 2008).

A Computação Evolutiva é baseada na Teoria da Evolução das Espécies (DARWIN, 1859). Entre as técnicas precursoras da área, se destacam o Algoritmo Genético (AG) (HOLLAND, 1992), também conhecido como Algoritmo Evolutivo (AE) [2], e a Programação Genética (PG) (KOZA, 1992). Em ambos, uma população é um conjunto de possíveis soluções do universo do problema, que evoluem de geração em geração por meio de operadores de reprodução, tal como ocorre na natureza.

Por algum tempo, os AGs não foram muito bem aceitos na área de jogos, apesar de possuírem diversas possibilidades de aplicações. Essa dificuldade de aceitação ocorreu porque tais algoritmos geralmente necessitam de muito processamento e tempo, recursos difíceis de serem obtidos em jogos (SWEETSER; WILES, 2002). Ainda assim, com a evolução das tecnologias e técnicas, a aceitação dos AGs na área tem aumentado, sendo atualmente uma das mais utilizadas (YANNAKAKIS; TOGELIUS, 2015).

Há diversos tipos de jogos, dentre os quais existem os que se encaixam como sendo do tipo presa-predador, por exemplo, o clássico jogo Pac-man (YANNAKAKIS, 2005).

[1] Personagens não jogáveis controlados por agentes inteligentes.

[2] A rigor, Algoritmo Genético utiliza representação binária e Algoritmo Evolutivo não necessariamente. Neste material, vamos considerar Algoritmo Genético como sinônimo de Algoritmo Evolutivo.

Nesse tipo de jogo, um grupo de personagens são os predadores os quais buscam caçar, consumir ou matar um outro grupo de personagens (as presas) que busca fugir e se manter vivo (CHEROBIN, 2014). Deste modo, uma presa pode ser o personagem controlado por um jogador e, os predadores, os NPCs que necessitam de uma inteligência. No exemplo do jogo Pac-man, a presa seria o Pac que busca fugir dos fantasmas, os predadores.

Um jogo do tipo presa-predador pode ser modelado como um sistema multi-agente, onde cada predador é controlado por um agente inteligente. Neste caso, a inteligência dos NPCs do jogo pode ser controlada por um agente, sendo que um agente pode ser definido como uma entidade que percebe o ambiente e age sobre ele (RUSSEL; NORVIG, 2003).

A inteligência e a tomada de decisão dos agentes inteligentes podem ser obtidas por meio de técnicas evolutivas. Neste contexto, pode-se citar a Programação Genética (FOLETTO, 2015), que utiliza de Árvores de Decisão para a representação do indivíduo, e os Algoritmos Genéticos Co-Evolutivos (DA ROSA et al., 2013), que utilizam de uma representação dos possíveis estados da percepção do agente e tem como indivíduo as possíveis ações, além de realizarem a evolução não só dos agentes predadores, mas também dos agentes que controlam as presas.

Deste modo, este trabalho propõe o uso de Algoritmos Genéticos para gerar NPCs que sejam adaptáveis durante um jogo do tipo presa-predador, inspirado no trabalho de Cherobin (2014).

1.1 Motivação

A motivação deste trabalho vem da necessidade de jogos mais inteligentes que proporcionem desafios mais adaptáveis para os jogadores. A aplicação em jogos do tipo Presa-Predador foi escolhida porque diversos jogos da atualidade aplicam esse conhecimento, isto é, existem alvos que são perseguidos para serem capturados, ou mortos, o que acarreta a vitória do jogador, além do cenário contrário em que o jogador deve evitar a captura. Como exemplo, pode-se citar: *Spore*[1], ARK: *Survival Evolved* [2], os jogos da franquia *Resident Evil*[3] e o *multiplayer Dead By Daylight*[4], sendo este último um exemplo de um jogo Presa-Predador sem a presença de NPCs.

No contexto dos algoritmos bioinspirados, o Algoritmo Genético foi escolhido como técnica a ser aplicada devido ao seu crescimento em aplicações em jogos. Além disso, ele é um algoritmo de fácil implementação que gera não apenas uma solução ótima ou aproximada, mas um conjunto de possíveis soluções.

[1] <https://www.spore.com/>
[2] <http://playark.com/>
[3] <http://www.residentevil.com/>
[4] <https://deadbydaylight.com/>

1.2 Objetivos

O objetivo geral deste trabalho é o desenvolvimento de um Algoritmo Genético para tornar mais fortes os NPCs de um mundo virtual bidimensional com modelagem Presa-Predador. Tais NPCs devem apresentar comportamento adaptável graças a melhoria de seus atributos, seguindo como referência principal o trabalho de Cherobin (2014).

Os objetivos específicos são:

1. Realizar o desenvolvimento de um jogo em que seus NPCs possuem relações presa-predador e compõem uma cadeia alimentar.
2. Implementar um Algoritmo Genético capaz de evoluir os atributos dos NPCs do jogo.
3. Verificar o desempenho dos NPCs evoluídos pelo AG por meio da análise de métricas do jogo.

1.3 Organização do Trabalho

O presente trabalho está organizado da seguinte forma: no Capítulo 2 são apresentados os conceitos sobre a Computação Evolutiva com enfoque nos Algoritmos Genéticos. O Capítulo 3 aborda os fundamentos do modelo Presa-Predador e da aplicação da Inteligência Artificial em jogos. O Capítulo 4 traz um panorama sobre trabalhos relacionados. No Capítulo 5 são explicadas as implementações do jogo e do Algoritmo Genético desenvolvido. Por fim, os Capítulos 6 e 7 apresentam, respectivamente, os resultados obtidos, sendo feita uma comparação da execução do jogo com e sem o uso do AG, e as conclusões deste trabalho.

2 Algoritmos Genéticos

Neste capítulo são apresentados os conceitos e a teoria sobre a Computação Evolutiva (CE) com enfoque nos Algoritmos Genéticos (AGs).

A Computação Evolutiva é uma área de pesquisa pertencente ao campo de Inteligência Computacional e abrange um conjunto de técnicas de busca e otimização com inspiração na Teoria da Evolução das Espécies (DARWIN, 1859). Os métodos que pertencem a CE são comumente chamados de Algoritmos Evolutivos (AEs) e são adequados para encontrar boas soluções para variados problemas, além de possuírem uma simplicidade na sua aplicação. Esses fatores contribuem para a rápida expansão que a área vem tendo (GABRIEL; DELBEM, 2008).

Foi John Holland um dos pioneiros no desenvolvimento das primeiras pesquisas sobre simulações computacionais de sistemas genéticos, tendo este elaborado os Algoritmos Genéticos durante as décadas de 60 e 70 na Universidade de Michigan (MITCHELL, 1998). Os AGs são considerados os maiores precursores dos AEs e juntamente com as abordagens da Programação Evolutiva (PE) e das Estratégias Evolutivas (EEs), propostas, respectivamente, por Lawrence Fogel em 1966 e Ingo Rechenberg em 1965 e que foram desenvolvidas de forma independente. AGs, PEs e EEs estão entre os considerados Algoritmos Evolutivos canônicos (GABRIEL; DELBEM, 2008).

Apesar de terem sido desenvolvidas separadamente, os três AEs canônicos possuem o mesmo objetivo que é desenvolver uma maneira de utilizar os fenômenos de adaptação das espécies e da seleção natural que ocorrem na natureza em aplicações científicas através da incorporação destes aos computadores. Por possuírem o mesmo objetivo possuem também o mesmo princípio básico, onde possuindo uma população de indivíduos, que são possíveis soluções de um problema, é simulada uma seleção natural, onde os indivíduos são avaliados e os melhores selecionados possuindo mais chance de se reproduzirem e gerarem novos indivíduos, logo novas soluções, que herdam as características boas dos seus precursores. Sendo que, as principais diferenças entre as técnicas são:

- Na codificação dos indivíduos, onde os AGs utilizam vetores binários, enquanto, a PE utiliza Máquinas de Estados Finitos e as EEs vetores de números reais;
- Nos operadores genéticos, com a PE e as EEs utilizando apenas mutação, diferente dos AGs que utilizam além desta o *crossover*;
- E na ordem das etapas do algoritmo, em que nos AGs primeiro ocorre a seleção dos indivíduos e depois a aplicação dos operadores genéticos, ocorrendo uma inversão dessas etapas nas outras duas abordagens.

Será dado enfoque às especificidades dos Algoritmos Genéticos por serem a técnica utilizada no presente trabalho, além de ser a técnica mais utilizada dentre os AEs canônicos. Para realizar a aplicação de um AG a um problema, segundo Pozo et al. (2005) é necessário, entre outros requisitos, que suas possíveis soluções possam ser modeladas no formato de um código genético, com cromossomos e genes (este código será melhor abordado na Seção 2.1), e também que exista um método de medir quão boa é cada solução.

Goldberg (1989) apresenta o algoritmo de um AG simples. Nele inicialmente é gerada uma população inicial de indivíduos aleatória, após isso os indivíduos são avaliados por uma função de aptidão, conhecida como *fitness*. Realizada a avaliação, ocorre a seleção dos indivíduos para a reprodução, tendo os mais aptos maiores chances de serem selecionados. Tendo sido selecionados os indivíduos acontece a reprodução havendo troca de partes entre dois indivíduos, a fim de gerar dois novos indivíduos que herdem e combinem suas características boas. Após o processo de reprodução é realizada a aplicação da mutação, que modifica características dos indivíduos buscando o aparecimento de novas características. Os novos indivíduos obtidos pela reprodução e mutação substituem os anteriores gerando uma nova população a qual o algoritmo é aplicado novamente a partir do momento da avaliação. Esses passos são repetidos até que uma condição de parada seja atingida, seja o número máximo de iterações ou a solução ótima encontrada.

O algoritmo descrito acima é ilustrado no fluxograma da Figura 1.

Figura 1 – Fluxograma do AG. Adaptado de: (POZO et al., 2005)

Para melhor entendimento dos termos e partes do AG é necessária uma descrição mais detalhada de cada, as seções a seguir trazem essa descrição.

2.1 Indivíduos

Os indivíduos do AG são uma codificação das possíveis soluções de um problema, sendo essa representação utilizada de maneira que o algoritmo possa trabalhar adequadamente. Originalmente, a representação utilizada por Holland (1992) era a codificação binária, onde o indivíduo é uma cadeia de *bits*. Essa representação é muito utilizada devido a sua fácil implementação e manipulação.

São usados termos pertencentes a genética de maneira análoga nos AGs (COLHERINHAS, 2016), assim um indivíduo é composto por cromossomos, que podem ser uma cadeia de caracteres, e cada caractere é um gene presente em uma determinada posição, ou seja, em um locus com um certo valor, o seu alelo. São também utilizados os termos genótipo para definir o indivíduo codificado e fenótipo para o decodificado, que é a solução na sua representação original.

2.2 População

A população de um Algoritmo Genético é um conjunto de indivíduos, logo um conjunto de possíveis soluções. O tamanho da população é um dos parâmetros a ser analisado na implementação do AG, já que este pode afetar o desempenho e eficiência do algoritmo. Se escolhido um tamanho muito pequeno pode haver perda da diversidade necessária pelo AG na busca pela solução ótima, uma vez que o espaço de busca seria restringido. Por outro lado, um tamanho grande geraria perda de eficiência, pois o custo computacional para percorrer a população aumentaria.

A geração da população inicial é outro fator importante do AG. Ela deve ser feita de maneira aleatória a fim de cobrir pontos diversos do espaço de soluções, podendo em algumas aplicações serem necessários ajustes de maneira a ajudar na convergência (COLHERINHAS, 2016).

2.3 Função de aptidão

A função de aptidão (*fitness*) é o ponto mais importante de um AG tendo em vista que essa juntamente com a representação dos indivíduos conecta diretamente o algoritmo ao problema. É essa função que julga quão boa uma solução é.

2.4 Seleção

Após ter sido calculado e associado a cada indivíduo da população seu *fitness* ocorre a seleção dos indivíduos que serão submetidos ao operador genético de cruzamento. A seleção deve ocorrer de maneira que indivíduos mais aptos tenham chances maiores de serem selecionados.

Entre os métodos de seleção mais populares se destacam o da roleta e do torneio, descritos abaixo:

Método da roleta: Nesse método os indivíduos são selecionados com probabilidade proporcional ao seu *fitness*. Eles são distribuídos como em uma roleta com cada um tendo uma parcela com tamanho relativo ao seu *fitness* e a soma de todos os *fitness* da população. Dessa forma, indivíduos mais aptos possuem uma porção maior e os menos aptos uma porção menor (POZO et al., 2005). Após isso, um número é sorteado e o indivíduo em que este se encontra na faixa é selecionado, de maneira análoga, seria como a bola sendo jogada na roleta de um cassino.

A Figura 2 ilustra como ocorre a distribuição dos indivíduos na roleta. Nela o indivíduo 1 possui *fitness* 20, sendo o mais apto dentre os indivíduos e possuindo a maior parcela da roleta.

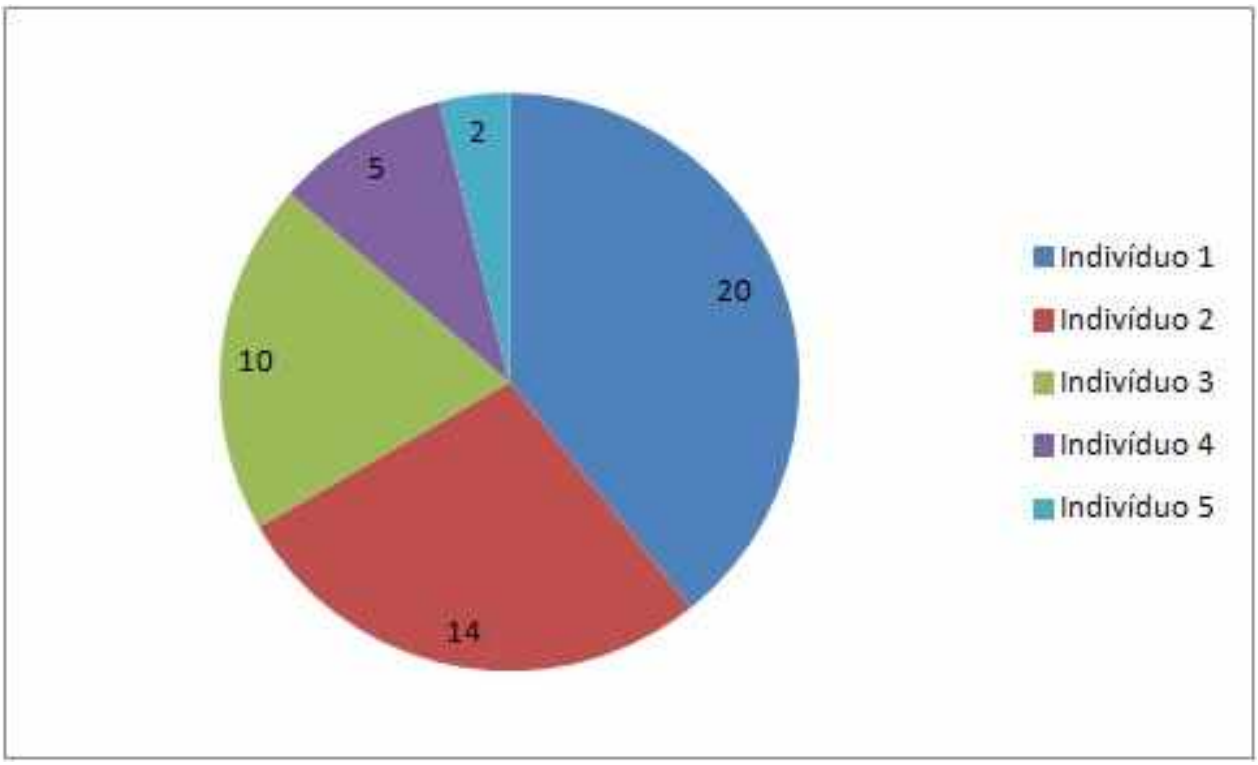

Figura 2 – Método de seleção por roleta. Adaptado de: (POZO et al., 2005)

Método do torneio: Aqui são escolhidos aleatoriamente um número k de indiví-

duos de uma população e esses indivíduos competem entre si de maneira que o mais apto é selecionado (COLHERINHAS, 2016). Observa-se então que k deve ter um valor mínimo de 2 para que haja competição.

2.5 Cruzamento

O cruzamento, ou *crossover*, tem a função de gerar novos indivíduos buscando que estes herdem as características dos indivíduos "pais"e que a partir da mistura dessas características sejam obtidas novas. É realizada a troca de trechos de um indivíduo com os trechos equivalentes do outro.

Quando utilizada a codificação binária para os indivíduos o tipo mais popular de recombinação é a de um ponto. Nela um ponto aleatório da cadeia é sorteado e o cromossomo é divido em dois, um dos novos indivíduos gerados possui a combinação da parte esquerda de um pai com a parte direita do outro, e o outro indivíduo gerado possui a outra combinação (GABRIEL; DELBEM, 2008). Esse tipo de cruzamento pode ser generalizado para n-pontos, de maneira que sejam feitos n cortes e as partes sejam trocadas de maneira análoga ao de 1-ponto. O *crossover* de um ponto é ilustrado na Figura 3, também é mostrado um *crossover* de dois pontos.

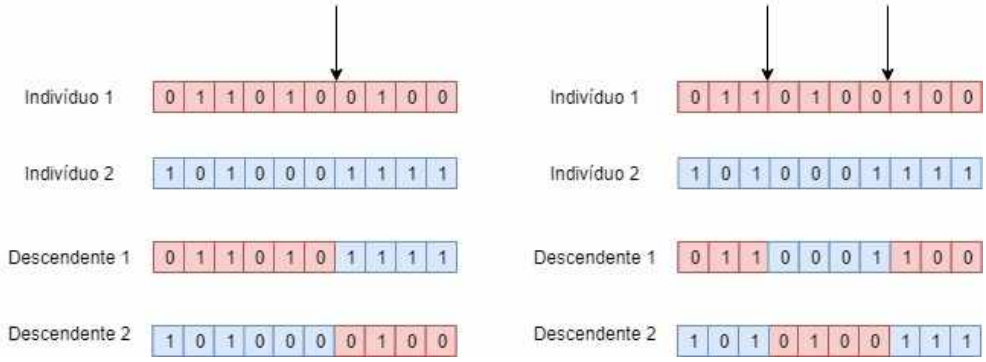

Figura 3 – Cruzamento em um ponto e cruzamento em dois pontos. Adaptado de: (GABRIEL; DELBEM, 2008)

2.6 Mutação

Realizado o cruzamento é aplicada a mutação com o intuito de serem introduzidas novas características nos indivíduos da população. Essa operação deve ser aplicada com uma taxa pequena porque ela pode ter o efeito de destruir informações importantes.

A mutação serve para manter a diversidade genética da população assim como aumentá-la, com isso permite que o AG possa alcançar qualquer ponto do espaço de busca (POZO et al., 2005).

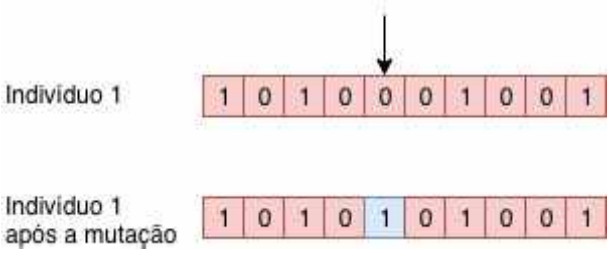

Figura 4 – Mutação de um *bit*. Adaptado de: (POZO et al., 2005)

Na maioria das codificações a mutação é aplicada trocando o valor de um gene de um indivíduo por outro valor. Na codificação binária o valor é invertido, 0 se torna 1 e 1 se torna 0, como mostrado na Figura 4, e nas codificações inteiras e decimais o valor é trocado por outro aleatório.

3 Jogos com modelo Presa-Predador

Nesta seção inicialmente são apresentados os fundamentos do modelo Presa-Predador, e após isso, é feita sua relação com jogos e apresentados conceitos sobre a aplicação de Inteligência Artificial neles.

Na natureza existem interdependências entre os seres vivos e o ambiente em si, sendo formada a partir dessa interdependência uma cadeia alimentar. Nesse esquema de cadeia os seres em níveis superiores se alimentam, ou consomem de alguma forma, os de níveis inferiores. Sendo que existem grupos que se alimentam dos grupos de níveis inferiores através da caça, possuindo com estes uma relação de caçador e caça, ou então, de predador e presa. Nesta relação, as espécies dos dois grupos estão presente num mesmo ambiente, o qual oferece recursos para a alimentação e sobrevivência das presas, as quais são posteriormente consumidas pelos predadores (CHEROBIN, 2014). É nessa situação que o problema presa-predador se baseia.

O problema presa-predador foi originalmente proposto por Benda, Jagannathan e Dodhiawalla (1986), sendo tratado como um sistema multi-agentes, com duas classes distintas desses, sendo estas a das presas e a dos predadores. Nele existem quatro agentes predadores e um agente presa que estão dispostos em um plano retangular com medidas NxN e todos possuem a mesma velocidade. O intuito do problema é então que os predadores possam capturar a presa a cercando, enquanto essa tenta fugir da captura buscando alcançar as bordas do plano. A modelagem desse sistema e os objetivos do problema podem ser vistos como uma competição ente dois grupos, podendo ser abstraído pra um conceito de jogo (YANNAKAKIS, 2005), como uma competição entre grupos de jogadores. A modelagem se aproxima ainda mais da ideia de um jogo por nela a movimentação dos agentes acontecerem por turnos, onde a cada turno cada agente escolhe como se movimentar, podendo esse movimento ser para a esquerda, direita, norte, sul ou até mesmo escolher não realizar nenhum movimento.

Um jogo que se aproxima dos conceitos do modelo apresentado pode ser considerado como pertencente ao gênero presa-predador. Num jogo desse tipo um dos agentes do sistema é controlado pelo jogador humano enquanto os outros são controlados por alguma técnica de IA, sendo utilizadas técnicas de aprendizagem por seu caráter adaptativo (YANNAKAKIS, 2005). Além disso para esse tipo de jogo existem as abordagens assíncronas e síncronas (MANDL; STOYAN, 2004), onde a síncrona é a maneira clássica em que são realizados turnos com os agentes tomando uma escolha de movimento a cada turno. Por outro lado, na assíncrona as decisões são tomadas durante o decorrer do jogo, paralelamente a aplicação das decisões já tomadas.

A aplicação da IA em jogos possui três abordagens diferentes: o controle da movimentação dos personagens, o controle da tomada de decisão dos personagens e o controle das táticas e estratégias do jogo (MILLINGTON; FUNGE, 2009). Sendo que a primeira trata-se da IA transformando a decisão em movimento, a segunda de realizar a decisão da próxima ação do personagem e a terceira em controlar um grupo de personagens de forma a existir uma estratégia conjunta entre as tomadas de decisões individuais.

As abordagens mais utilizadas da IA em jogos com o modelo Presa-Predador são por meio da tomada de decisão, qual ação tomar dentre atacar, andar ou fugir, e caso escolhido andar ou fugir, o controle da movimentação. Ou seja, ocorre o uso das duas primeiras abordagens em conjunto no controle individual dos personagens. Existem também trabalhos que buscam o uso da terceira abordagem através da cooperação entre os personagens do mesmo grupo a fim de realizar seu objetivo (DA ROSA et al., 2013).

Em relação a aprendizagem da técnica utilizada, esta pode ser classificada como *offline*, sendo realizada previamente ao início do jogo, ou *online*, onde a aprendizagem é realizada durante a execução. Quando utilizada esta em um jogo presa-predador ela pode ser realizada a cada turno, no caso da abordagem síncrona, ou paralelamente a execução do jogo na abordagem assíncrona (WITTKAMP et al., 2012).

Principalmente na aprendizagem *online*, quando na fase de desenvolvimento do algoritmo e de como será feito seu uso no jogo, deve ser levado em consideração o quesito de performance computacional, ou seja, o algoritmo não deve ser desenvolvido apenas com o objetivo de encontrar a solução ótima mas também de realizar este de forma eficiente sem concorrer por recursos com o jogo em si (MILLINGTON; FUNGE, 2009).

No trabalho desenvolvido foi utilizada uma abordagem de tomada de decisão, onde uma árvore de decisão simples utiliza dos atributos dos personagens para realizar a escolha da ação a ser tomada pelo personagem durante o jogo, sendo essa decisão influenciada pela evolução resultante do Algoritmo Genético, aplicado com uma aprendizagem *online* e assíncrona, a qual busca melhorar os atributos dos personagens.

4 Trabalhos Correlatos

4.1 *AI in Computer Games: Generating Interesting Interactive Opponents by the use of Evolutionary Computation*

Yannakakis (2005) desenvolveu um trabalho partindo do questionamento sobre o que torna um jogo de computador divertido e como aumentar a satisfação do jogador durante o jogo. Seus objetivos foram de estabelecer uma métrica pra medir a satisfação do jogador e utilizando desta, em conjunto com técnicas bioinspiradas, melhorar a experiência do jogo.

Foram escolhidos jogos do tipo presa-predador como base para a aplicação das técnicas desenvolvidas. Essa escolha ocorreu porque jogos desse tipo ofereciam uma análise fácil do comportamento dos personagens. Os jogos no trabalho foram tratados como sistemas multiagentes em que cada agente é controlado por uma Rede Neural (RN).

Inicialmente, foi desenvolvido um jogo sem modelagem presa-predador chamado *FlatLand* para realizar testes da aplicação de técnicas com o intuito de gerar agentes que cooperam entre si. Dentre as técnicas implementadas e avaliadas, a que se destacou foi o Algoritmo Genético, tendo este sido escolhido para ser utilizado pra evoluir os pesos da RN dos agentes tanto no aprendizado *offline* quanto no *online*, com adaptações para cada um. Nas variações do AG utilizadas a função de *fitness* é semelhante, nela ocorre uma simulação onde para cada individuo são gerados num ambiente clones do mesmo agente utilizando a RN com os pesos do individuo sendo avaliado.

Realizados os testes com o *FlatLand* o autor partiu para a aplicação das técnicas em jogos presa-predador. Primeiramente, foram aplicadas ao jogo *Dead End*, um jogo com um plano bidimensional contendo um retângulo branco, onde o personagem do tipo presa tem como objetivo alcançar esse retângulo e escapar, e os personagens predadores devem a capturar antes que ela alcance seu objetivo. Neste trabalho foi utilizado o AG na aprendizagem *offline* buscando desenvolver cooperatividade entre os agentes e na aprendizagem *online* com o intuito de melhorar os agentes, recompensando comportamentos agressivos de caça, gerando adaptatividade durante o jogo. Após os testes no *Dead End* foi realizada a aplicação em outro jogo do mesmo tipo, a fim de validar a efetividade dessa aplicação em jogos do tipo presa-predador. O jogo escolhido foi uma variação do clássico jogo *Pac-man*.

No trabalho para atingir o objetivo de identificar uma métrica de satisfação no jogo e também avaliar a aplicação das técnicas desenvolvidas, o autor realizou a aplicação de

questionários e testes com humanos, tendo obtido resultados satisfatórios na identificação da métrica.

Os resultados foram positivos para a meta de tornar os jogos mais adaptativos e de melhorar a experiência do jogador durante o jogo. Eles mostraram que o comportamento de cooperação entre os agentes enquanto buscam o objetivo da captura faz com que o nível de entretenimento do jogo seja alto e que o uso da aprendizagem *online* ajuda a manter o nível alto durante o jogo, podendo ainda aumentá-lo.

4.2 Algoritmos Evolutivos para a produção de NPCs com Comportamentos Adaptativos

Algoritmos Evolutivos (AEs) foram por muito tempo considerados inaptos para o uso em jogos por serem considerados lentos e também por poderem levar a soluções com desempenho ruim, por conta de seu caráter aleatório. Neste contexto, Crocomo, Miazaki e Simões (2007) realizaram um trabalho tendo como objetivo demonstrar a capacidade dos AEs em superar esses dois problemas.

No trabalho foi desenvolvido um jogo com modelagem presa-predador contendo personagens herbívoros, carnívoros ou onívoros. Nele as ações dos personagens são controladas por um navegador, no qual é utilizada uma tabela de controle de movimentação que dita qual o comportamento do personagem com base no que se encontra perto do mesmo. O jogador controla um personagem e tem como objetivo ser a última criatura sobrevivente.

O AE foi aplicado com aprendizagem *online* e nele os indivíduos são vetores de inteiros que codificam as saídas da tabela de movimentação. Dessa forma o objetivo do algoritmo é evoluir o navegador que controla os personagens e adaptá-los às estratégias tomadas pelo jogador. A avaliação dos indivíduos ocorre durante cada rodada do jogo e ao final desta uma nova geração é obtida, através do uso da seleção e de operadores genéticos, para ser avaliada na próxima rodada.

Os resultados obtidos mostraram a capacidade do AE de superar o desafio quanto a eficiência, tendo ele gerado adaptação satisfatória dos personagens em tempo de jogo, mas falhou em satisfazer o critério de efetividade, já que durante os testes em algumas ocasiões foram adotadas pelos personagens estratégias consideradas inadequadas. Os autores propuseram para novos trabalhos buscar maneiras de superar o problema com a efetividade.

4.3 *Real-time Evolutionary Learning of Cooperative Predator-Prey Strategies*

O objetivo de Wittkamp et al. (2012) é aplicação de um sistema de aprendizado para estratégia cooperativa em tempo real em um domínio multiagente. Um dos motivos da escolha da aplicação desse sistema é o fato de a maioria do uso de inteligência artificial em jogos ser através da aplicação da técnica de *Scripting*, que não oferece um fator de adaptabilidade aos jogos. O sistema de aprendizado utilizado é uma implementação de um Algoritmo Evolutivo.

O domínio multiagente escolhido foi um de modelagem presa-predador, onde será buscado evoluir os predadores para que estes utilizem uma estratégia cooperativa para a captura da presa.

Os autores observam que a maior dificuldade da obtenção de técnicas adaptáveis através de aprendizado *online* é a escassez de tempo e de poder computacional para execução dos algoritmos, sendo um dos objetivos do trabalho tornar possível essa execução.

No ambiente utilizado os predadores buscam capturar a presa, sendo que essa captura ocorre no momento em que um predador encosta na presa. A tomada de decisão dos predadores é gerada por meio do AE, sendo a evolução realizada a cada intervalo de tempo t, onde um personagem é selecionado para ser evoluído. A evolução acontece em paralelo aos eventos atuais do ambiente do jogo com o AE buscando evoluir quantas gerações forem possíveis durante o intervalo t e ao final deste o melhor indivíduo gerado substitui o personagem escolhido.

Os autores consideraram que o sistema desenvolvido obteve um bom desempenho no aprendizado em tempo real, conseguindo capturar a presa utilizando de uma quantidade razoável de tempo para aprendizagem. Foi observado também que a adaptabilidade foi uma característica marcante do sistema, tendo a aprendizagem sido capaz de adaptação a diferentes situações.

4.4 Co-evolução de Agentes Inteligentes Antagonistas usando Algoritmos Genéticos

da Rosa et al. (2013) buscaram dentre os objetivos de seu trabalho utilizar os algoritmos genéticos para evoluir agentes inteligentes, buscando a adaptação dos mesmos. Tendo sido a motivação para tal busca, tentar encontrar estratégias mais inteligentes e adaptáveis para controlar oponentes em jogos de computador. O trabalho objetiva a criação de agentes que controlam a tomada de decisão dos personagens de um jogo e que estes se adaptem às estratégias do jogador.

Foi criado um ambiente virtual com modelagem presa-predador possuindo uma população de carnívoros que se alimentam da população de herbívoros, que por sua vez se alimentam de plantas. O cenário desenvolvido é um campo bidimensional contendo personagens de ambas as populações e plantas espalhadas pelo mapa.

Os personagens possuem as possíveis ações de andar e de se aproximar de outro indivíduo da sua espécie. Além dessas ações comuns as duas espécies, existem as ações de fugir que pode ser tomada pelos herbívoros e a de caçar que pode ser tomada pelos carnívoros. A escolha de qual ação tomar é feita a partir de uma árvore de decisão, sendo essa árvore gerada pela aplicação de um algoritmo genético co-evolutivo.

Nos resultados foi percebido que o algoritmo que foi implementado não possui performance rápida o suficiente para ser utilizado para aprendizagem *online*. Apesar disso, na aplicação do algoritmo ao cenário foi observada a adaptabilidade das espécies, pois apesar de ocorrerem picos na quantidade de indivíduos de uma determinada espécie a situação se normalizava, com o tamanho das populações antagonistas se equiparando.

4.5 *Fast Algorithm for Catching a Prey Quickly in Known and Partially Known Game Maps*

Nesta seção é apresentado o trabalho de Baier et al. (2015), o qual ao contrário dos outros trabalhos apresentados nesse capítulo, não utiliza técnicas bioinspiradas. No trabalho em questão, os autores propõem o uso de um algoritmo de busca de alvo em movimento em um jogo com modelo presa-predador.

O algoritmo desenvolvido tem o nome de *MtsCopa* e utiliza de informação pré-computada sobre possíveis caminhos ótimos entre dois pontos no mapa, essa informação é armazenada em *compressed path databases* (CPDs), que permitem rapidamente recuperar a informação sobre qual movimento tomar. Como mapas de jogos possuem caráter dinâmico, o caminho fornecido pelo CPD pode ter se tornado inviável em tempo de execução, e quando esse cenário ocorre o algoritmo proposto utiliza de um algoritmo A* para tomar a decisão de qual movimento tomar. O A* não é utilizado para decidir o caminho completo até o alvo mas sim até o melhor ponto, a partir do qual o CPD volte a fornecer informação válida e eficiente.

O trabalho mostra que o algoritmo desenvolvido supera o estado da arte através da comparação com o algoritmo I-ARA*. Foram avaliadas as performances dos dois algoritmos em realizar a captura da presa, sendo que o *MtsCopa* se mostrou capaz de cumprir o objetivo com menos iterações, além de, ter se mostrado mais eficiente quando diminuído o limite de processamento de tempo para cada tomada de ação. Os autores concluíram que o algoritmo mostrou bons resultados, mas ressalta que este necessita de preprocessamento

e espaço de memória, ao contrário de outros algoritmos de busca de alvo em movimento.

4.6 Aplicação do modelo da cadeia alimentar juntamente com algoritmos genéticos para criação de NPCs adaptativos

No trabalho realizado por Cherobin (2014) o autor buscou utilizar a computação bioinspirada para elaborar uma abordagem para controle de NPCs com o intuito de melhorar o relacionamento destes com os jogadores. Outro objetivo buscado pelo autor foi demonstrar a possibilidade do uso de Algoritmos Genéticos em aprendizado *online*, ou seja, durante o jogo.

Foi utilizado o algoritmo de cadeia alimentar juntamente com o algoritmo genético para controlar o relacionamento de recursos entre os NPCs ao ambiente, ao outros NPCs e ao jogador.

Para aplicação dos algoritmos foi desenvolvido um protótipo de jogo do tipo *Role-playing game* (RPG), o qual foi modelado com os personagens pertencendo a uma cadeia alimentar com relações presa-predador entre os níveis desta. O jogador controla então um personagem pertencente a algum nível da cadeia, tendo uma relação de predador com os personagens de níveis inferiores, assim como, uma relação de presa com os de níveis superiores, sendo essa mesma relação presa-predador existente para os NPCs.

Os personagens do jogo possuem atributos tais como dano, defesa, vida, dentre outros, que são inicialmente definidos para cada espécie da cadeia, de forma a obedecer as relações de superioridade e inferioridade da mesma. Esse atributos são o alvo da aplicação do algoritmo genético, eles são os valores a serem evoluídos pelo AG, de forma que o NPC da espécie a qual o AG foi aplicado se torne mais apto a sobreviver no ambiente durante o jogo.

Os resultados obtidos pelo uso do AG foram de que não ocorre uma taxa continua de crescimento do tempo de vida dos NPCs durante o jogo, existindo uma variação da quantidade dos tipos de NPCs no mundo de forma a que este fique mais adaptado ao estado do jogo. Foi analisado também o desempenho do AG, tendo sido observado que foi necessário baixo custo computacional para execução do AG, tornando possível seu uso durante o jogo para o aprendizado *online*.

Considerando o que foi apresentado nesta seção sobre o trabalho de Cherobin (2014) que apresentou resultados positivos, este foi escolhido como inspiração para o presente trabalho.

5 Implementação

Neste capítulo serão apresentados o modelo e a estrutura do jogo desenvolvido, assim como, a implementação do Algoritmo Genético utilizado.

5.1 Jogo presa-predador

O jogo desenvolvido por Cherobin (2014) foi a inspiração para este Trabalho de Conclusão de Curso, onde foram baseados: os modelos de relação entre os NPCs, os atributos considerados para os NPCs e o processamento de suas ações. O jogo foi implementado utilizando o *framework* LÖVE2D[1], tendo os *sprites* do mapa, sido retirados do site <www.kenney.nl> e os *sprites* dos NPCs gerados no site <http://charas-project.net/> ou retirados do <https://opengameart.org/>.

O ambiente do jogo, mostrado na Figura 5 consiste do mapa, um plano bidimensional, e dos personagens do jogo. O objetivo de cada personagem no jogo é a sobrevivência pela maior faixa de tempo possível, sendo que o modelo presa-predador foi utilizado como modelo da relação entres os personagens no jogo. Por existirem mais que dois tipos de NPCs com a relação presa-predador, estes estão dispostos em uma cadeia alimentar, onde NPCs de níveis superiores buscam se alimentar dos de níveis inferiores. No jogo original de Cherobin (2014) existem 4 níveis na cadeia e 8 tipos de NPCs diferentes. No presente trabalho foram utilizados apenas 3 tipos de NPCs e 3 níveis na cadeia, mostrados na Figura 6.

Os NPCs possuem os atributos mostrados a seguir:

- Dano: força de seu ataque;
- Defesa: quantidade de dano mitigado ao receber um ataque;
- Visão: alcance de percepção de outros personagens;
- Velocidade: velocidade com que o NPC se movimenta;
- Fome: porcentagem de fome;
- Vida: quantidade de vida;
- Nível: nível do NPC na cadeia alimentar;
- Tipo de NPC: identificador do tipo de NPC.

[1] https://love2d.org/

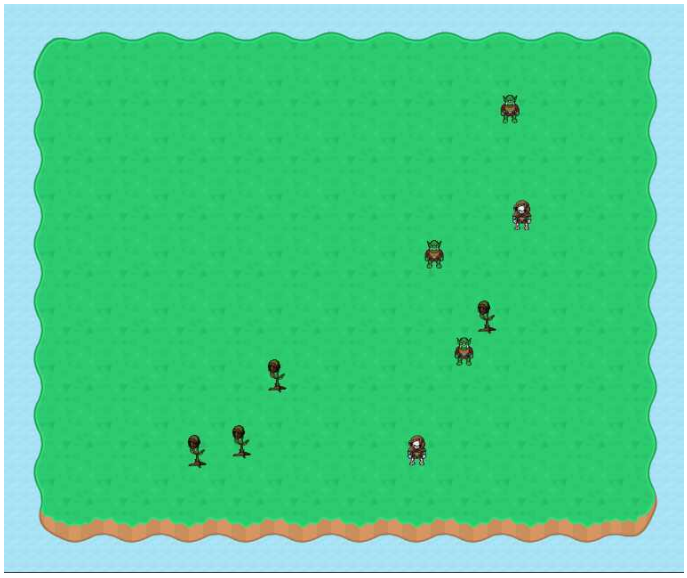

Figura 5 – Ambiente do jogo.

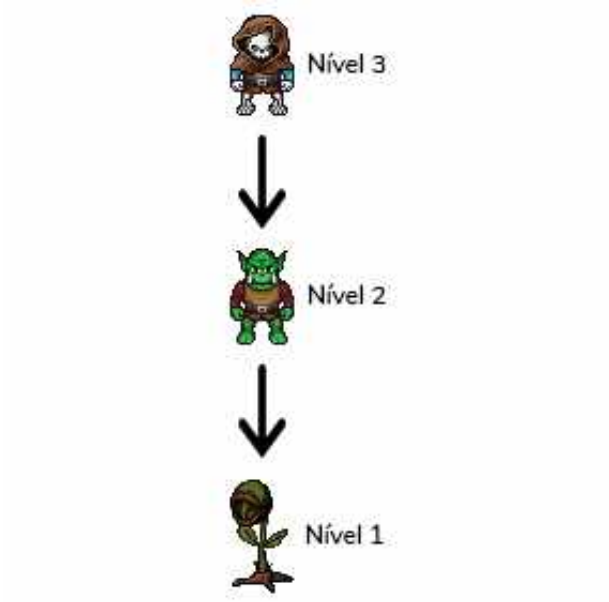

Figura 6 – Cadeia alimentar do jogo.

Dentre os atributos apresentados, os atributos de dano, defesa, visão e velocidade são inicialmente diferentes por espécie de NPC e são o foco da aplicação do AG, detalhada na Seção 5.2. Os valores iniciais para esses atributos foram retirados de Cherobin (2014), tendo sido escolhidos os valores de 3 NPCs dos 3 primeiros níveis da cadeia utilizada por este. Os valores de dano e defesa foram mantidos iguais aos de Cherobin (2014) e os de velocidade e visão foram modificados pelos autores, já que estes estão ligados a fatores da implementação do jogo, tais como tamanho do mapa e o modo como é realizada a movimentação dos personagens. Os valores desses atributos são mostrados na Tabela 1. Os atributos nível e tipo de NPC também são diferentes por espécie e não são modificados

durante o jogo. Os atributos fome e vida são inicializados, respectivamente, com 0 e 100, e são incrementados ou decrementados durante o jogo conforme as ações dos NPCs, e consequentemente, do resultado de suas ações.

Tabela 1 – Atributos dos NPCs.

Atributos	NPC1	NPC2	NPC3
Nível da Cadeia	1	2	3
Tipo	1	2	3
Dano	0,4	0,56	1.75616
Defesa	0,4	0,48	0.82944
Visão	10	30	50
Velocidade	0	40	80

No jogo, a cada passo de tempo para cada NPC é realizado o processamento de tomada de decisão para sua ação. Essa tomada de decisão é feita através de um *script* que recebe como entrada a percepção do ambiente pelo NPC sendo processado, ou seja, a informação se existe algum outro personagem no seu alcance de visão e, caso exista, a informação sobre os valores dos atributos deste. Com essa informação, a decisão é tomada pelo *script*, cujo funcionamento foi retirado de Cherobin (2014) e está representado no fluxograma da Figura 7.

Existem 3 tipos de ações possíveis, que são mostrados como possíveis saídas do fluxograma. São elas: movimentar-se em alguma direção aleatória, fugir se movimentado na direção contrária a do personagem percebido, ou atacar, onde ocorre uma espécie de batalha entre o NPC que tomou a decisão e o personagem percebido. Nessa batalha ambos os personagens tem sua vida diminuída e a quantidade que será diminuída é mostrada nos cálculos abaixo, as fórmulas utilizadas foram retiradas de Cherobin (2014). Serão utilizados no exemplo os atributos dos NPCs 1 e 2.

$$vidaPerdida_{npc1} = dano_{npc2} - (0.06 * defesa_{npc1})/(1 + 0.06 * defesa_{npc1})$$
$$vidaPerdida_{npc1} = 0.56 - (0.06 * 0.4)/(1 + 0.06 * 0.4)$$
$$vidaPerdida_{npc1} = 0.56 - 0.0234$$
$$vidaPerdida_{npc1} = 0.5365$$

$$vidaPerdida_{npc2} = dano_{npc1} - (0.06 * defesa_{npc2})/(1 + 0.06 * defesa_{npc2})$$
$$vidaPerdida_{npc2} = 0.4 - (0.06 * 0.48)/(1 + 0.06 * 0.48)$$
$$vidaPerdida_{npc2} = 0.4 - 0.02799$$
$$vidaPerdida_{npc2} = 0.3720$$

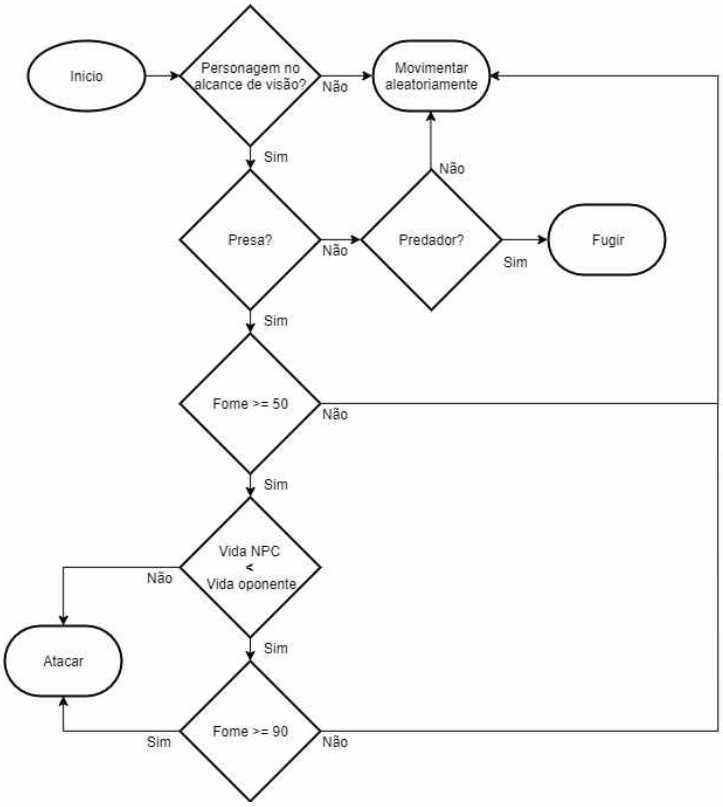

Figura 7 – Fluxograma de tomada de decisão dos NPCs. Adaptado de Cherobin (2014)

Realizada a ação do NPC, 3 cenários podem acontecer com o valor de seu atributo fome: o primeiro caso é quando forem tomadas as ações de movimentar aleatoriamente ou fugir, e deste modo, sua fome será aumentada; os outros dois cenários ocorrem para o caso de ele ter atacado, e são eles, o oponente morrer, ou seja, ter o valor de sua vida zerado após o ataque o que faz com que o valor da fome do NPC diminua, ou se o oponente não morrer, a fome tem seu valor mantido sem ser alterado. No cenário em que a fome do NPC aumenta e esta atingir o valor 120, o NPC morrerá.

Quando um NPC morre existe uma chance de 3% do AG ser executado para gerar um NPC daquela espécie mais adaptado, ou seja, com atributos melhores, buscando assim manter uma dinâmica e o aumento da dificuldade do jogo. Se o AG não for executado um novo NPC será gerado no jogo com os mesmos atributos do NPC morto. Sendo assim, a quantidade de cada tipo de NPC se mantém a mesma a cada passo do jogo.

5.2 Algoritmo Genético

Inicialmente o Algoritmo Genético foi desenvolvido inspirado no apresentado por Cherobin (2014), buscando manter a estrutura semelhante ao deste, tanto na codificação dos indivíduos quanto nos passos do algoritmo. Por conta da implementação do ambiente utilizado e de testes realizados com o AG foram realizadas modificações na estrutura,

sendo implementado um algoritmo diferente do trabalho original. As principais diferenças são na etapa de simulação, seleção dos país e funcionamento do *crossover* aplicado.

Na codificação dos indivíduos, representada na Figura 8 foi utilizado um vetor de 4 posições que representam os valores dano, defesa, visão e velocidade do NPC sendo evoluído. A geração da população inicial utiliza os valores destes mesmos 4 atributos do NPC da mesma espécie do NPC morto mais apto atualmente no jogo. Para obter os valores para os indivíduos é realizada uma multiplicação dos valores do melhor NPC por um fator aleatório, que é gerado entre 1,4 e 1,5. Essa maneira de obter os valores foi elaborada pelos autores deste trabalho já que no trabalho de Cherobin (2014) não é apresentado como é feita essa etapa.

Figura 8 – Representação do indivíduo utilizado.

Gerada a população inicial o AG entra em um laço onde são realizados os passos mostrados a seguir. Um fluxograma completo do funcionamento do AG é mostrado na Figura 10.

1. **Batalha -** Antes do cálculo do *fitness* para cada indivíduo é realizada uma espécie de batalha entre os valores dos atributos codificados no indivíduo e a média dos valores dos NPCs vivos no jogo neste momento, sendo que, vencerá quem superar os valores do oponente. Após essa etapa se algum indivíduo vencer o AG é encerrado aqui e o indivíduo vencedor é considerado apto e gerado no jogo.

2. **Cálculo do *fitness* -** Caso nenhum indivíduo vença a batalha é realizado aqui o cálculo do *fitness*, sendo este feito através da Fórmula 5.1, retirada de Cherobin (2014). O algoritmo busca minimizar o valor do *fitness*, ou seja, indivíduos em que o resultado do cálculo abaixo são mais próximos de 0 são considerados mais aptos.

$$Ap = \sum \frac{VAR_2}{VAR_1} - \frac{VAR_3}{VAR_1} \tag{5.1}$$

   Onde:

   VAR_1 = Soma dos atributos do indivíduo sendo avaliado.

   VAR_2 = Média da soma dos atributos dos predadores do NPC morto.

   VAR_3 = Média da soma dos atributos das presas do NPC morto.

3. **Seleção -** Realizado o cálculo do *fitness* é feita a seleção de quais indivíduos serão submetidos ao operador genético de *crossover*. Essa seleção é feita através da escolha

de dois indivíduos aleatórios dentre a população, no trabalho original a seleção é feita através da escolha dos dois melhores indivíduos, a mudança foi realizada pela seleção aleatória apresentar melhores resultados.

4. **Aplicação dos operadores genéticos -** Nesta etapa são aplicados os operadores genéticos de *crossover* e mutação.

   a) ***Crossover* -** Selecionados os pais a serem submetidos ao operador e utilizando a taxa de *crossover* é verificado se este será aplicado, caso ele seja aplicado é realizada a troca de partes entre os dois indivíduos, selecionadas na etapa anterior, de forma que sejam gerados dois novos indivíduos. Essa operação é mostrada na Figura 9. Se o *crossover* não for aplicado então os dois indivíduos selecionados são copiados sem modificações para o novo vetor de população.

Figura 9 – Representação do *crossover* utilizado.

O *crossover* utilizado neste trabalho se difere do apresentado por Cherobin (2014), onde o operador também realiza um ponto de corte, funcionando da seguinte forma: no momento da troca das partes podem ocorrer mudanças na posição destas, de forma que num ponto de corte no meio do indivíduo a primeira parte do Pai 1 pode se tornar a segunda parte do Filho 1, enquanto a segunda parte do Pai 2 se torna a primeira parte, assim ocorrendo não só a troca de partes entre os pais, mas também uma permutação de suas posições no filho resultante.

   b) **Mutação -** Após o *crossover* é aplicada a mutação. Para esse operador cada gene de cada indivíduo da população possui uma chance de sofrer a mutação, essa chance é verificada por meio da taxa de mutação. Caso um gene seja submetido a mutação o valor deste será modificado por um valor gerado da mesma maneira que são criados os valores na população inicial. Neste momento, o fator utilizado na multiplicação pode ser gerado numa faixa que varia durante a execução do AG, começando na primeira geração com os limites de 1,4 a 1,5 e terminando na última geração com 1,0 a 1,5.

Esse laço ocorre até que um limite de gerações seja atingido ou um indivíduo sobreviva a etapa da batalha. Caso ocorra até atingir o limite de gerações, o melhor indivíduo da última geração é o considerado mais apto e retornado ao jogo.

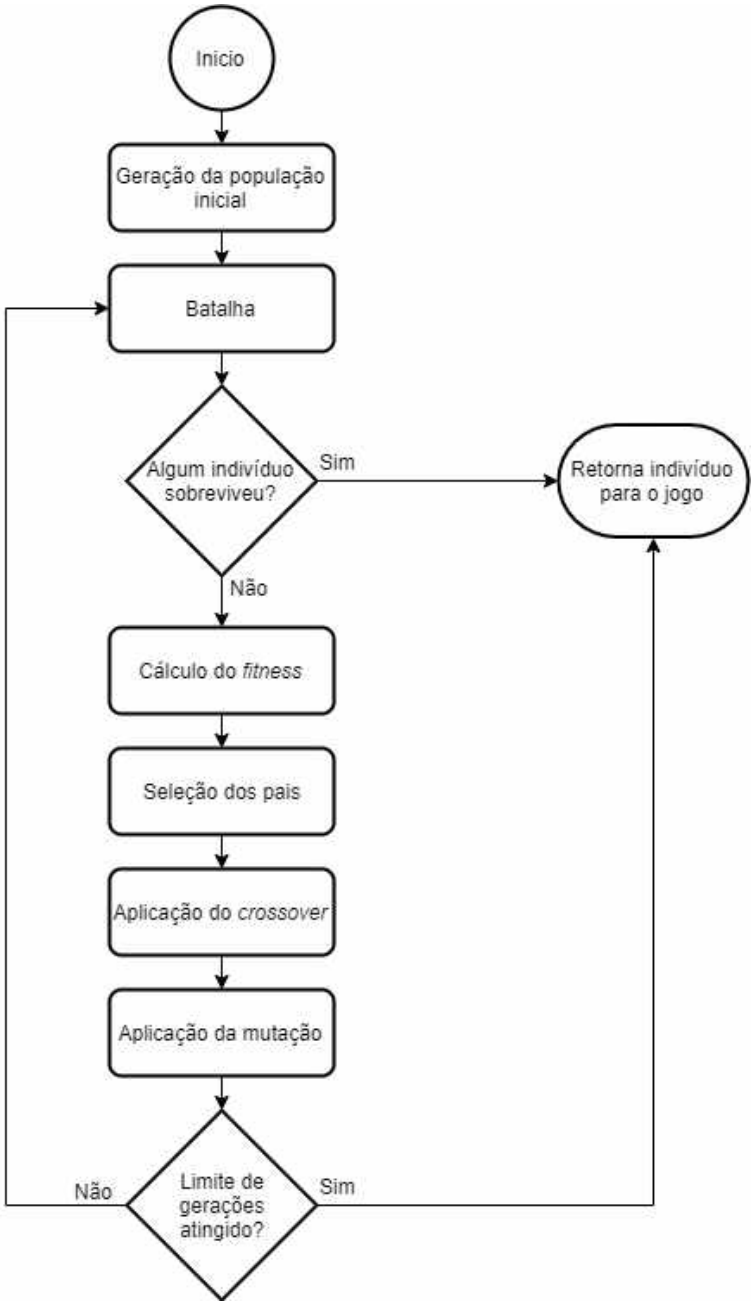

Figura 10 – Fluxograma do Algoritmo Genético utilizado.

6 Resultados

O objetivo deste Trabalho de Conclusão de Curso é apresentar uma abordagem utilizando um Algoritmo Genético para melhorar a adaptabilidade e dificuldade do jogo desenvolvido. Nesse contexto, serão apresentados dados sobre a execução do jogo sem o uso do Algoritmo Genético, e dados com o uso deste, a fim de tornar possível a comparação dos dois cenários para verificação da validade do AG implementado.

O jogo desenvolvido foi implementado na linguagem de programação Lua com o uso do *framework* LÖVE2D e o AG foi implementado na linguagem C, que possui fácil conexão com programas na linguagem Lua, na qual programas em C podem ser utilizados como uma biblioteca.

Nos testes com o AG foi usada a configuração abaixo:

- Tamanho da população: 100;
- Número máximo de gerações: 100;
- Taxa de *crossover*: 45%;
- Taxa de mutação: 3%.

Os valores para o tamanho da população, número máximo de gerações e taxa de mutação foram os mesmos utilizados por Cherobin (2014) e foi escolhido mantê-los os mesmos. Para a taxa de *crossover* foram realizados testes com os valores 18%, 45% e 80%, tendo sido averiguado melhor performance com a taxa de 45% e portanto, sido a taxa utilizada nas execuções.

Para cada cenário são retirados dois conjuntos de dados como resultados, sendo o primeiro a média dos valores dos atributos de todos os NPCs por faixa de tempo e o segundo a média do tempo de vida em segundos dos tipos de NPCs por faixa de tempo. Para cada teste foi realizado uma execução de uma hora de jogo.

6.1 Resultados preliminares do AG em 60 minutos de execução

A partir dos primeiros testes, são mostrados nas Figuras 11 e 12 os gráficos com a média por faixa de tempo dos quatro atributos utilizados na evolução. São ilustrados lado a lado os valores para a execução sem o AG e para a execução com o AG. É notável que houve a melhoria na média dos valores dos atributos quando utilizado o Algoritmo Genético e que o aumento dos valores ocorreu gradualmente durante a execução do jogo,

principalmente para os atributos de visão e velocidade, mostrados na Figura 12, o que indica um resultado coerente com o objetivo do AG. Esse aumento mostra que o AG está tornando os NPCs, de modo geral, mais fortes durante o jogo, aumentando assim a dificuldade deste.

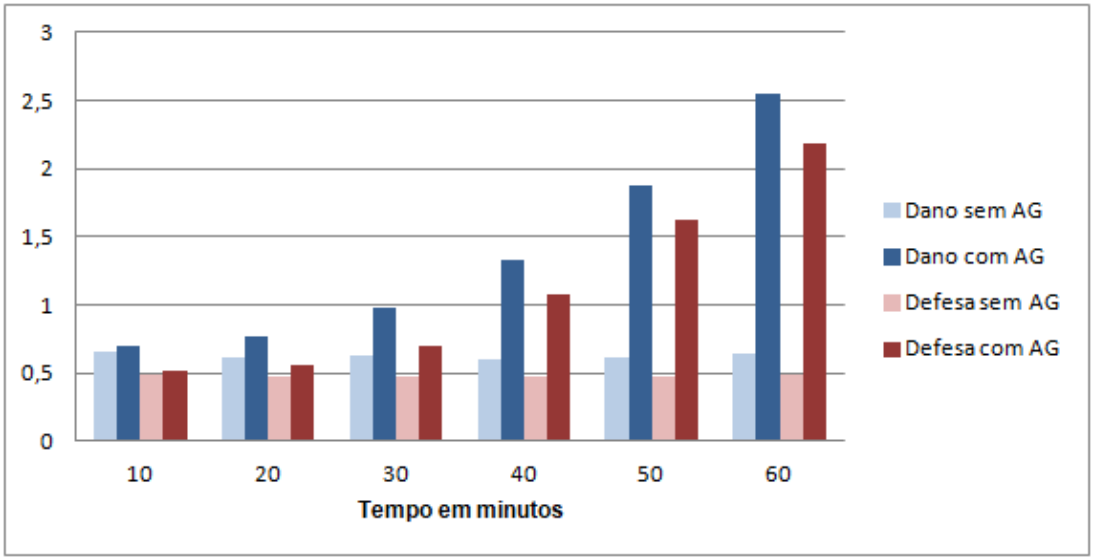

Figura 11 – Média dos atributos dano e defesa por faixa de tempo.

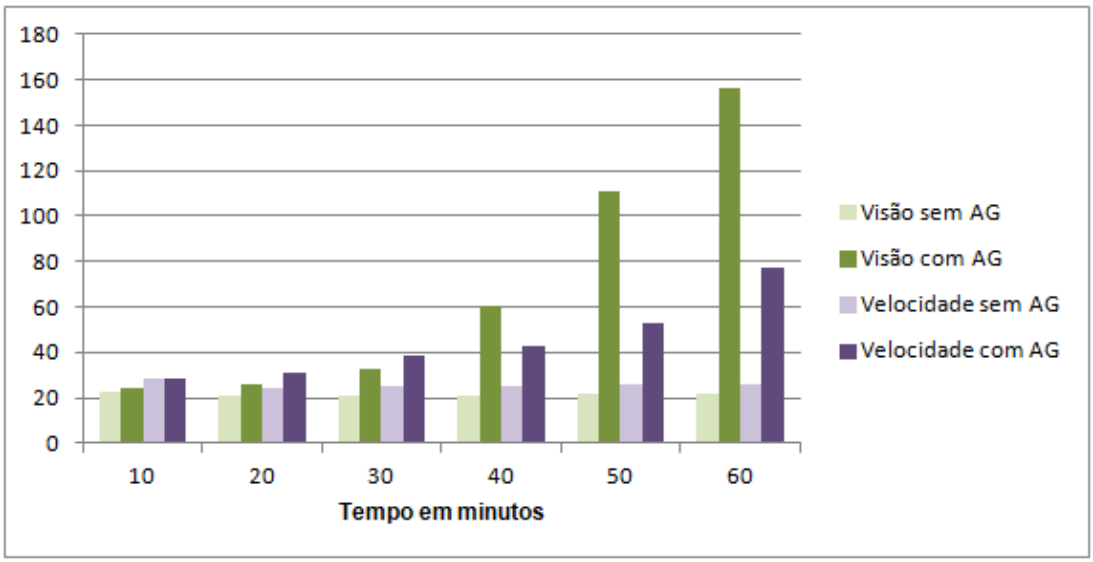

Figura 12 – Média dos atributos visão e velocidade por faixa de tempo.

As Figuras 13 e 14 apresentam, respectivamente, os gráficos para os tempos médios de vida dos tipos de NPC por faixa de tempo na execução sem AG e na execução com AG. Os dados destes dois gráficos mostram que, mesmo com o uso do AG tendo a influência de melhorar os atributos dos NPCs, no tempo médio de vida dos NPCs ocorre uma alteração de comportamento das espécies. Por exemplo, para o NPC1 o AG fez com que seu tempo

médio de vida aumentasse, que é considerado um resultado positivo, no sentido de que o AG está conseguindo tornar o NPC1 mais forte, de fato. A questão é que o NPC1 é uma presa, e o que adicionaria maior dificuldade ao jogo seria tornar um predador mais forte. Por outro lado, o NPC3 (um predador) teve queda no tempo médio de vida, o que indica que este NPC tornou-se menos forte que antes quando comparado às suas presas, como podemos observar na Tabela 2.

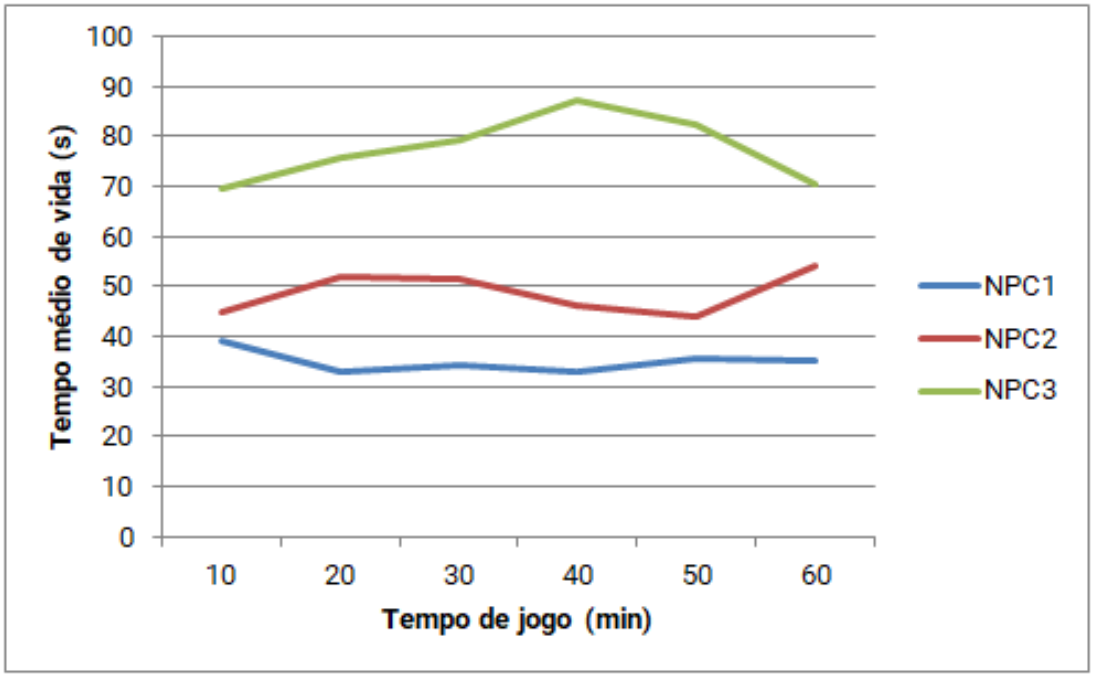

Figura 13 – Tempo médio de vida por tipo de NPC sem uso do Algoritmo Genético.

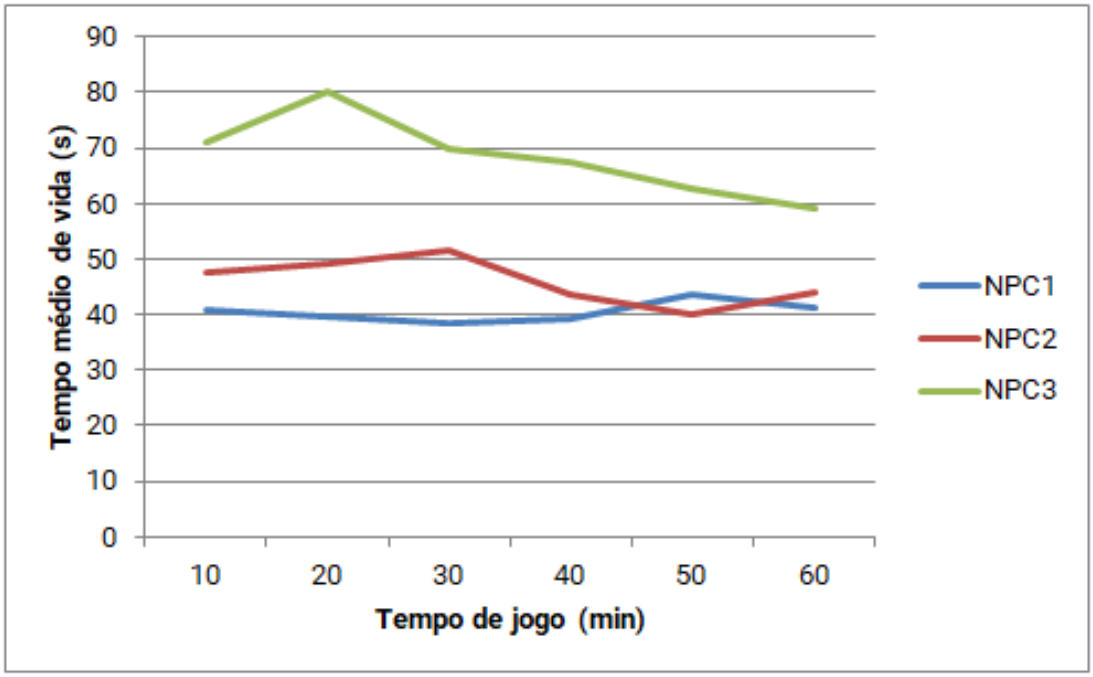

Figura 14 – Tempo médio de vida por tipo de NPC com uso do Algoritmo Genético.

Tabela 2 – Valores dos atributos dos NPCs ao final da execução com AG.

Atributos	NPC1	NPC2	NPC3
Dano	3,004	1,849	2,699
Defesa	3,391	0,942	1,244
Visão	238,678	78,677	74,911
Velocidade	0	164,938	122,829

6.2 Resultados com variações no AG em 60 minutos de execução

Na implementação inicial do *fitness* utilizado no Algoritmo Genético, a qual gerou os resultados já apresentados, foi utilizado o módulo do valor obtido da Equação 5.1. Isto foi feito a fim de realizar uma análise inicial sobre a função de *fitness*, supondo, à princípio, que pudesse ter resultados melhores. Após os testes com o uso do módulo foram realizados testes sem o uso deste, ou seja, permitindo valores de *fitness* negativos, que é como Cherobin (2014) efetuou seu trabalho. Os resultados da execução sem o uso do módulo são apresentados nas Figuras 15, 16, 17 e 18. Na execução com a modificação os valores dos atributos apresentaram crescimento menor do que comparado com a execução anterior.

A modificação realizada tem mudança mais significativa para o NPC 3, uma vez que nele o primeiro fator da subtração da fórmula do *fitness* (Equação 5.1) é sempre zero por conta deste não possuir nenhum predador e é onde se desejava alterar o comportamento do mesmo, a fim de aumentar a média de tempo de vida deste. Analisando o gráfico da Figura 18 e comparando com a Figura 14 é visível uma estabilização na curva do NPC 3, tendo essa chegado a crescer entre os momentos 20 e 30, e ao final, pode-se observar um resultado melhor comparado à Figura 14. Para o NPC 1 sua curva apresentou valores menores, mas ainda teve crescimento durante a execução e o NPC2 em ambas as execuções teve uma queda na sua curva.

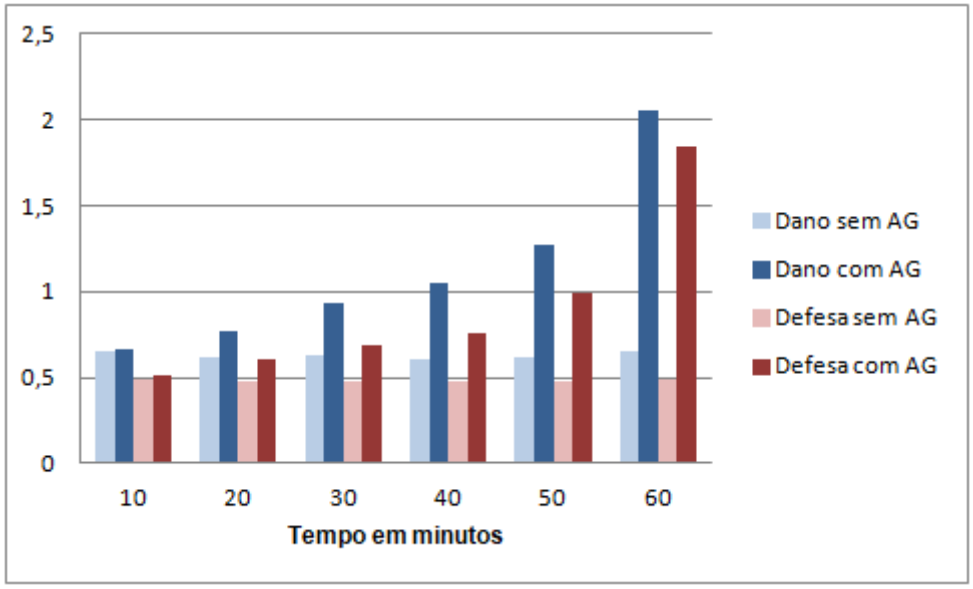

Figura 15 – Média dos atributos dano e defesa por faixa de tempo sem modularizar o *fitness*.

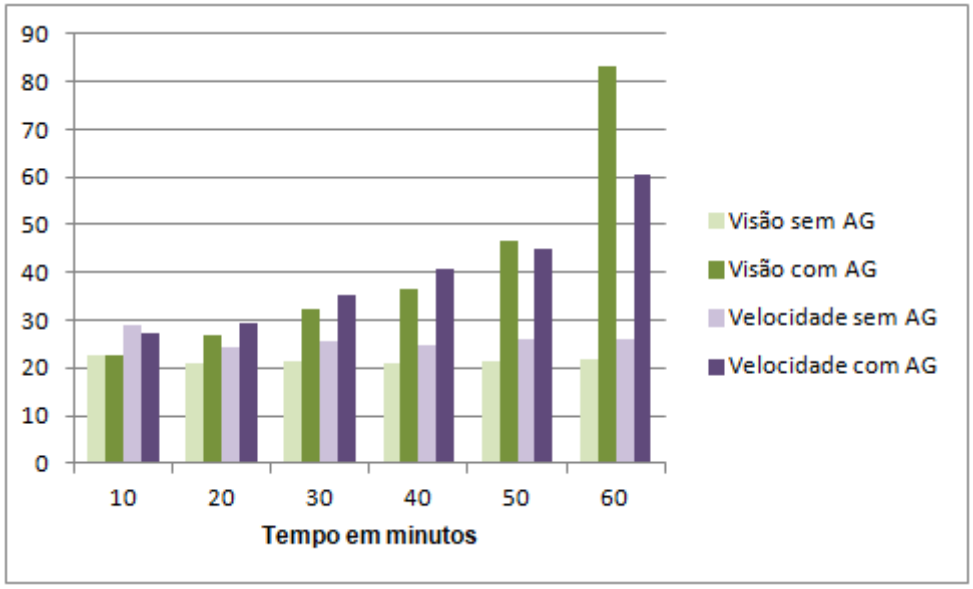

Figura 16 – Média dos atributos visão e velocidade por faixa de tempo sem modularizar o *fitness*.

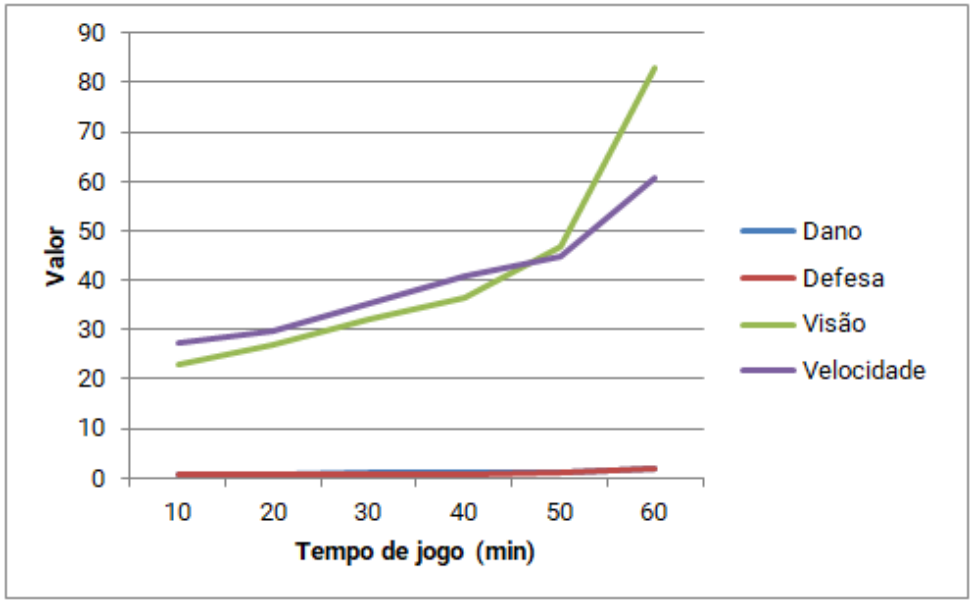

Figura 17 – Média de atributos por faixa de tempo sem modularizar o *fitness*.

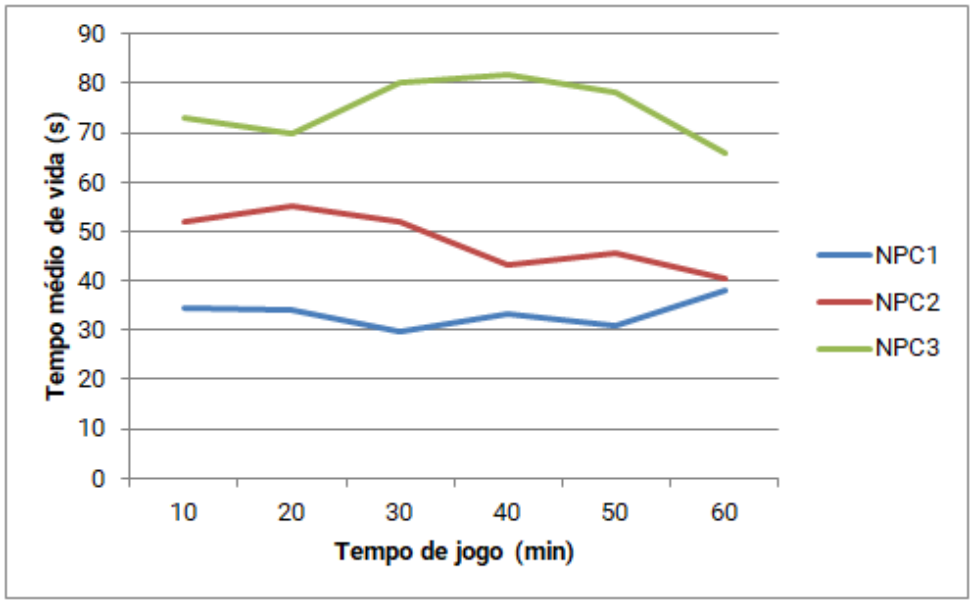

Figura 18 – Tempo médio de vida por tipo de NPC sem modularizar o *fitness*.

Após serem obtidos os resultados apresentados foi realizada uma modificação no AG para limitar o atributo de visão até 80, pois observando os experimentos foi possível notar um crescimento muito grande neste atributo. O objetivo dessa limitação foi equilibrar os NPCs. Os resultados obtidos da execução com essa modificação são apresentados nas Figuras 19, 20, 21 e 22. Analisando os gráficos e comparando com os resultados já obtidos pode-se verificar que limitar um valor causou modificações na tendência das curvas de tempo médio de vida dos NPCs. Pode-se observar nessa execução que o NPC 1 teve uma queda em seu valor e o NPC 2 um aumento, mostrando neste caso que o NPC1

tornou-se mais fraco que antes e o NPC2, mais forte, que é um resultado coerente e bem mais interessante para a jogabilidade do usuário. Considerando o NPC1 sendo uma presa, será bom ao jogador que ele seja fraco, enquanto que o NPC2 sendo um predador, será mais desafiador ao jogador que ele seja mais forte. Nesta execução o NPC3 ainda teve queda no tempo médio de vida, mas com valores melhores que no experimento anterior, observando uma melhoria no resultado comparado à Figura 18. Além disso, nessa execução a média dos atributos cresceram ainda menos do que na execução sem o limite na visão, tendendo a se equilibrarem, deste modo, os NPCs.

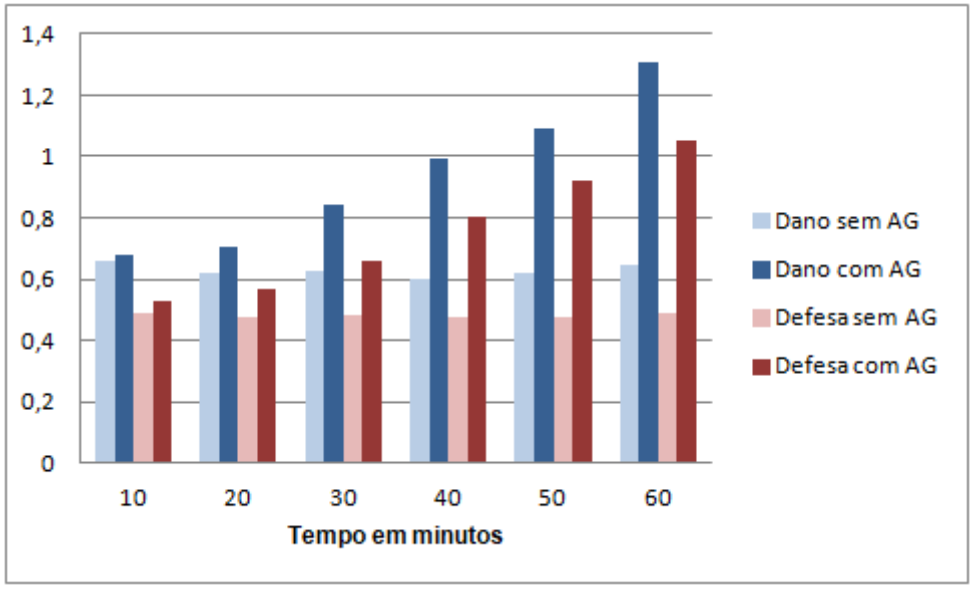

Figura 19 – Média dos atributos dano e defesa por faixa de tempo sem modularizar o *fitness* e com limitante no atributo visão.

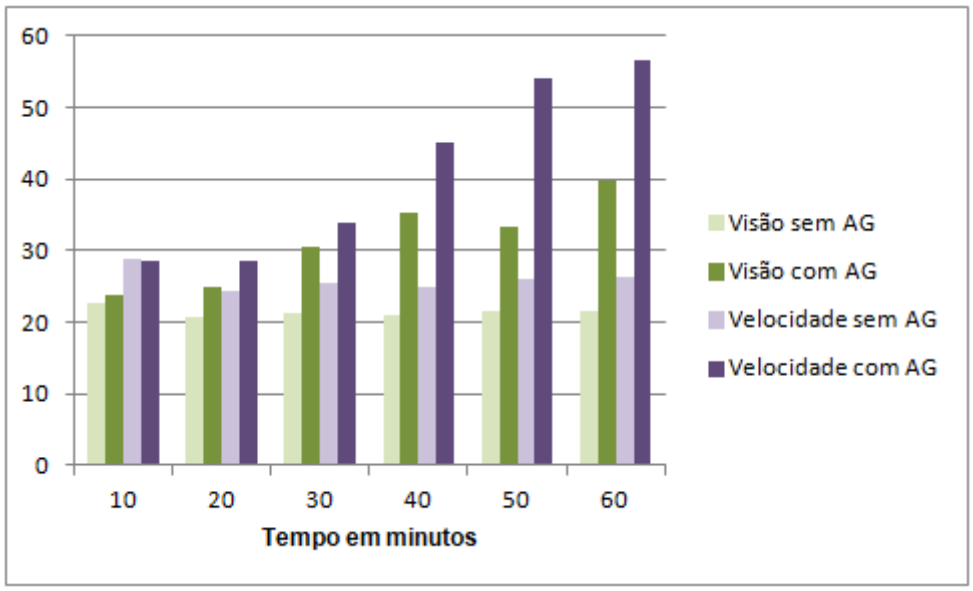

Figura 20 – Média dos atributos visão e velocidade por faixa de tempo sem modularizar o *fitness* e com limitante no atributo visão.

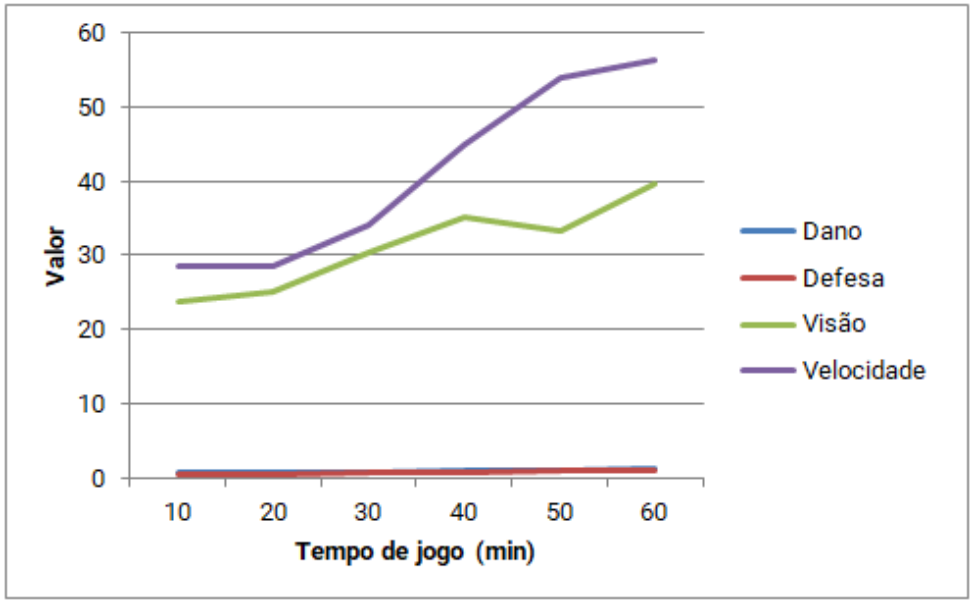

Figura 21 – Média de atributos por faixa de tempo sem modularizar o *fitness* e com limitante no atributo visão.

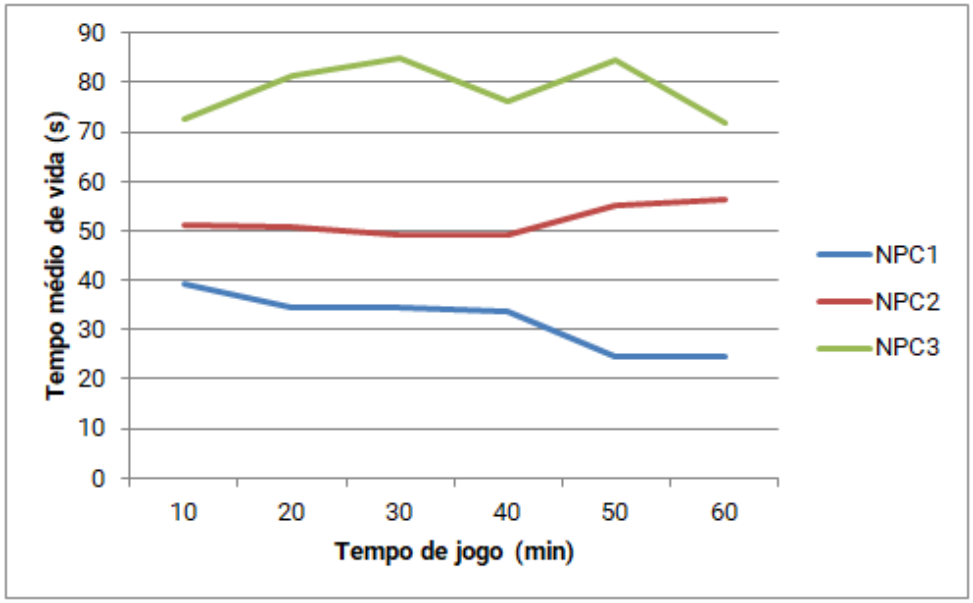

Figura 22 – Tempo médio de vida por tipo de NPC sem modularizar o *fitness* e com uso de limitante no atributo visão.

6.3 Resultados com variações do AG em 360 minutos de execução.

Para uma análise mais profunda dos efeitos da aplicação do AG em relação ao tempo decorrido foram realizadas quatro execuções com 6 horas de duração: uma com AG com *fitness* sem ser modularizado e sem limitante da visão, outra com AG com *fitness* sem ser modularizado e com limitante na visão, outra com AG com *fitness* modularizado, e a última sem AG. A finalidade destes experimentos foi verificar o comportamento dos NPCs

ao longo de um tempo mais prolongado, bem como Cherobin (2014) também realizou. Para as duas primeiras execuções (Figuras 23 e 24) a queda do tempo médio de vida do NPC 3 aconteceu de maneira mais intensa durante os primeiros 100 minutos, tendo após isso continuado a cair mas em um ritmo menor, chegando a se estabilizar após a faixa de tempo entre 250 e 280 minutos na execução sem o limite no atributo visão. Além disso em ambas o tempo médio de vida do NPC 1 cresceu, sendo que com limitante de visão o NPC1 tende a se estabilizar (Figura 24) ao longo do tempo.

Na execução com o limite de visão, na Figura 24 os NPCs 1 e 2 tiveram a intensidade da variação de sua curva semelhantes, ou seja, tendo a variação sido mais intensa nos primeiros 100 minutos e após isso tendo estabilizado. A limitação no atributo visão é a provável causa da estabilização das curvas dos 3 tipos de NPCs, já que na faixa de tempo entre 70 e 130 os valores do atributo visão para os NPCs 1 e 3 atingiram o limite, como mostrado na Figura 25.

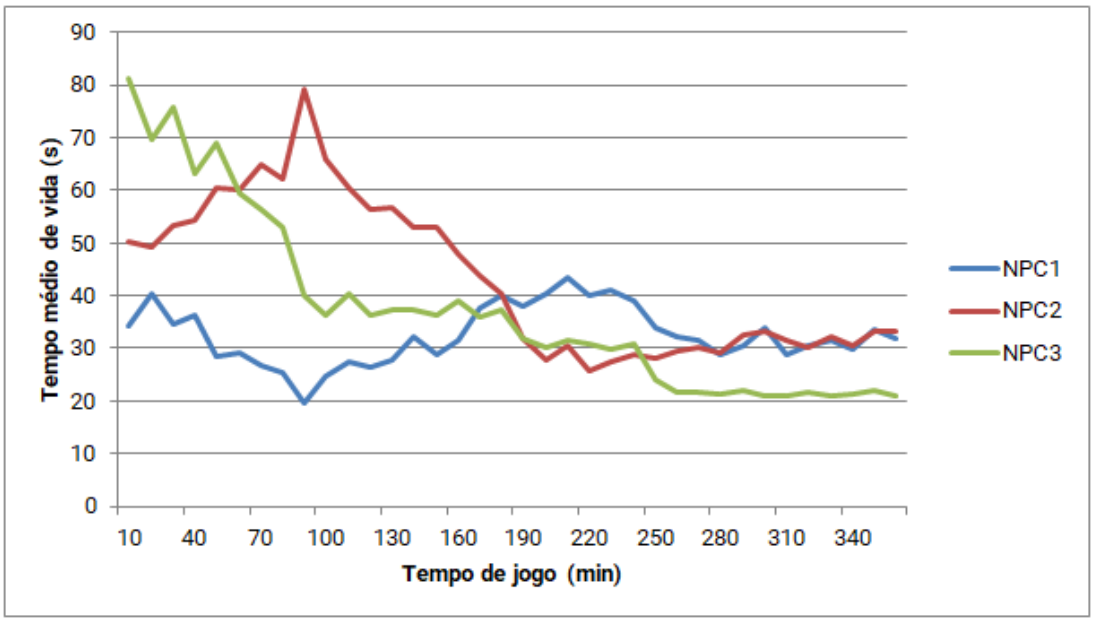

Figura 23 – Tempo médio de vida por tipo de NPC sem modularizar o *fitness*.

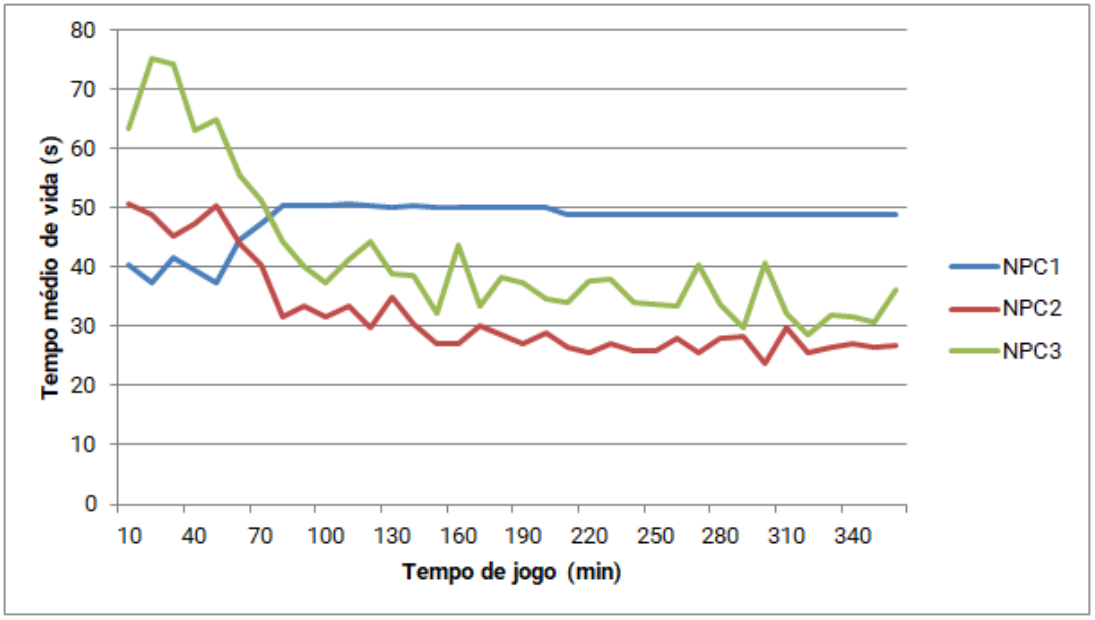

Figura 24 – Tempo médio de vida por tipo de NPC sem modularizar o *fitness* e com limitante no atributo visão.

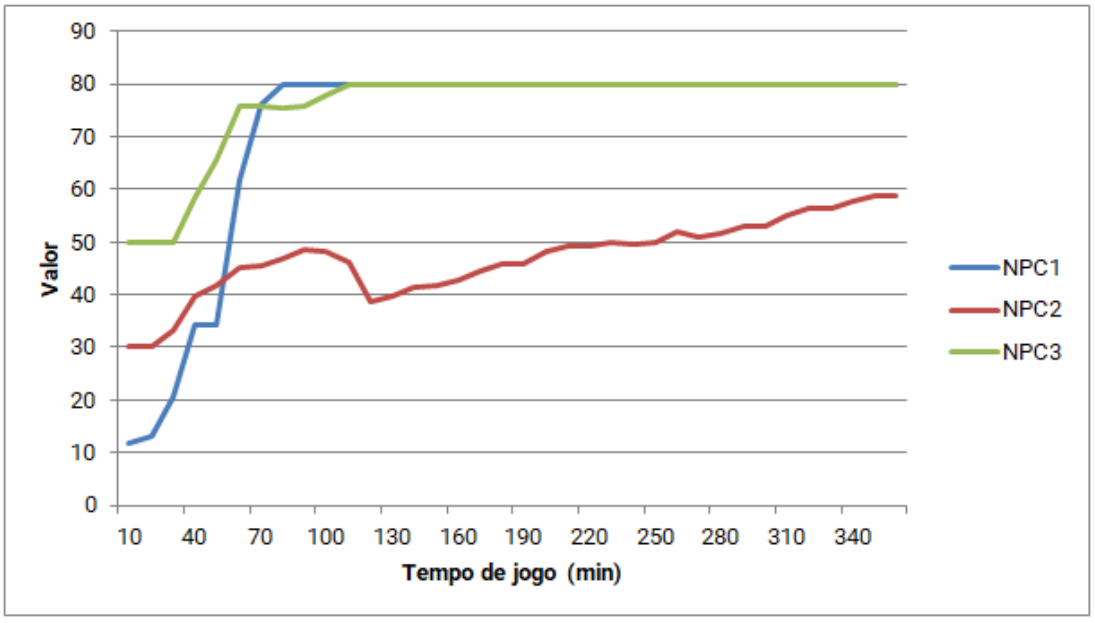

Figura 25 – Média do atributo visão por tipo de NPC para a execução da Figura 24.

Realizadas as duas execuções de 6 horas sem uso do módulo no *fitness* foram realizadas mais duas execuções com esse tempo, mas agora com a configuração das execuções apresentadas nas Figuras 13 e 14, ou seja, uma execução com o uso do AG com *fitness* modularizado e uma execução sem o uso do AG. Comparando a execução sem AG com as execuções com AG é visível que em todas houve variação do cenário e da média de tempo de vida dos NPCs.

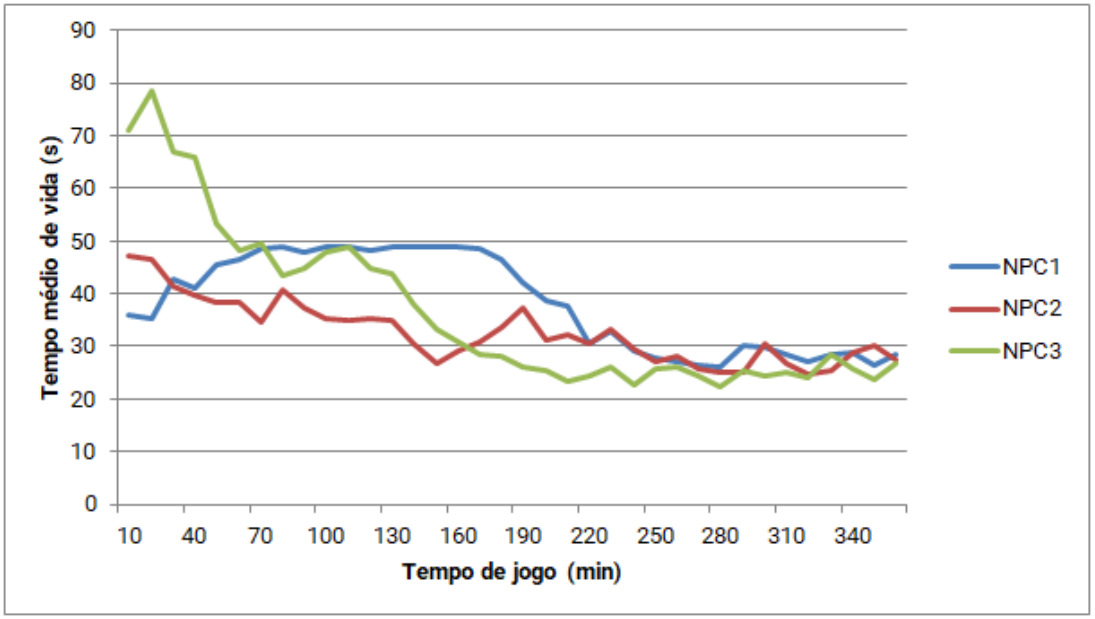

Figura 26 – Tempo médio de vida por tipo de NPC com uso do AG modularizando o *fitness*.

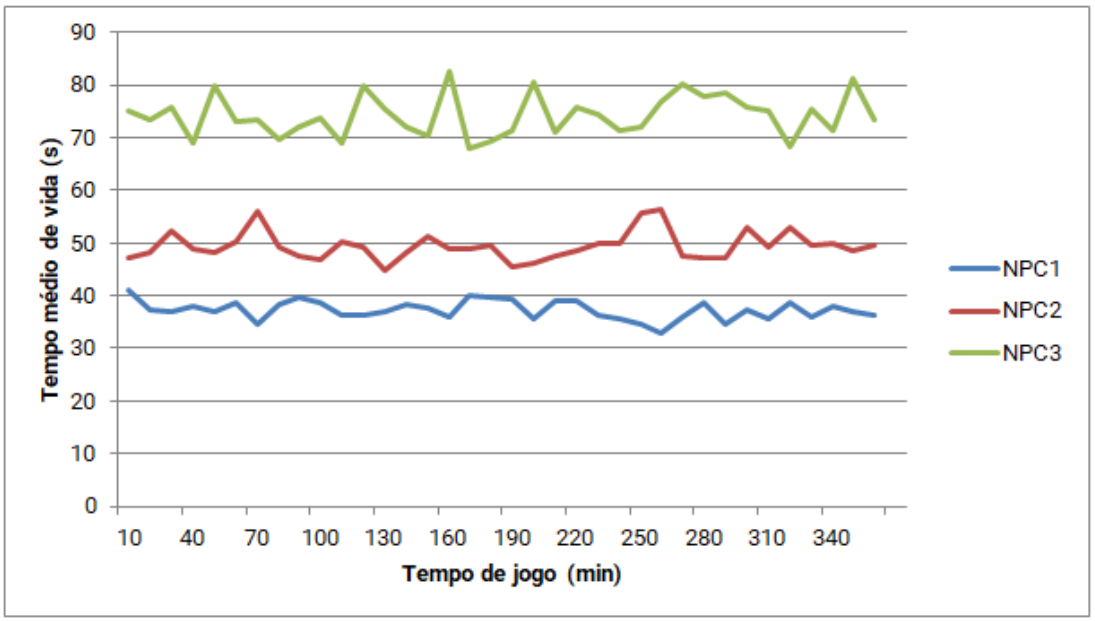

Figura 27 – Tempo médio de vida por tipo de NPC sem o uso do AG.

Com essas execuções mostradas nas Figuras 26 e 27 pode-se verificar que quando foi aplicado o AG com modularização os NPCs tenderam a viver menos, mas com a vantagem de serem mais competitivos entre si, ou seja, não prevalecendo um mais forte que outro. Por outro lado, sem o AG os NPCs viveram mais, porém os personagens permaneceram com comportamentos estáveis durante o jogo, o que tornou o jogo mais previsível.

Deste modo, analisando os quatro casos, o AG sem ser modularizado e sem o limitante de visão torna o jogo mais competitivo no sentido de que o comportamento

dos NPCs não é mais tão previsível quanto antes, principalmente devido aos NPCs 1 e 2 que se mostraram mais fortes. A desvantagem deste é que eles tenderam a viver menos, especialmente o NPC3, justamente por serem mais equiparados entre si.

7 Conclusão

Neste Trabalho de Conclusão de Curso foram desenvolvidos um jogo, com relações Presa-Predador entre seus NPCs, e um Algoritmo Genético, que busca evoluir esses NPCs. Tal desenvolvimento foi feito com inspiração no trabalho de Cherobin (2014) sem haver a replicação deste. Este trabalho representa a simplificação do jogo, com o uso de apenas 3 tipos de NPC, e as modificações nas etapas do AG, especificamente na seleção e operadores de reprodução. Portanto, a análise do funcionamento do algoritmo desenvolvido foi realizada comparando as métricas obtidas de execuções do jogo com e sem sua aplicação.

Na comparação entre os resultados foi observado que o AG e suas variações têm forte influência em aumentar os valores dos atributos dos NPCs, de modo geral. No entanto, apareceram efeitos inesperados para cada NPC com essas variações, onde foi possível notar a queda no tempo de vida dos mesmos. Logo, os NPCs tornaram-se mais fortes em relação aos atributos, mas com inversão de comportamento ao longo do tempo.

Com o desenvolvimento deste trabalho juntamente com os resultados obtidos, foi concluído que a aplicação do AG para melhorar os atributos dos NPCs e gerar comportamento adaptável é possível, mas que o efeito da adaptabilidade no panorama geral da execução do jogo foi de tornar por vezes as presas mais fortes que seus predadores. Isso é refletido nas métricas de tempo médio de vida, onde mesmo com o aumento dos atributos, os NPCs, de modo geral, viveram menos. Esse cenário, de certa forma, é surpreendente e pode ser interessante para a jogabilidade do usuário, uma vez que enfatiza a competitividade entre os NPCs, tornando o jogo menos previsível. Para resolver o problema de os NPCs viverem menos, a ideia é adaptar a função de *fitness*, de modo que permita o crescimento dos atributos sem que interfira negativamente no tempo médio de vida dos NPCs.

Referências

BAIER, J. A. et al. Fast algorithm for catching a prey quickly in known and partially known game maps. *IEEE Transactions on Computational Intelligence and AI in Games*, v. 7, n. 2, p. 193–199, 2015. Citado na página 26.

BENDA, M.; JAGANNATHAN, V.; DODHIAWALLA, R. *On optimal cooperation of knowledge sources (Tech. Rep. BCS-G2010-28)*. [S.l.: s.n.], 1986. Citado na página 21.

CHEROBIN, R. *Aplicação do modelo da cadeia alimentar juntamente com algoritmos genéticos para criação de NPCS adaptativos*. Dissertação (Mestrado) — Universidade do Vale do Itajaí, 2014. Disponível em: <https://siaiap39.univali.br/repositorio/handle/repositorio/1025>. Citado 15 vezes nas páginas 6, 13, 14, 21, 27, 28, 29, 30, 31, 32, 33, 35, 38, 43 e 47.

COLHERINHAS, G. B. *Ferramenta de otimização via algoritmos genéticos com aplicações em engenharia*. Dissertação (Mestrado) — Universidade de Brasília, Faculdade de Tecnologia, Departamento de Engenharia Mecânicas., 2016. Disponível em: <https://repositorio.unb.br/bitstream/10482/21832/1/2016_GinoBertolluccColherinhas.pdf>. Citado 2 vezes nas páginas 17 e 19.

CROCOMO, M. K. *Um algoritmo evolutivo para aprendizado on-line em jogos eletrônicos*. Dissertação (Mestrado) — Universidade de São Paulo, 2008. Citado na página 12.

CROCOMO, M. K.; MIAZAKI, M.; SIMõES, E. d. V. Algoritmos evolutivos para a produção de npcs com comportamentos adaptativos. In: *Simpósio Brasileiro de Jogos para Computador e Entretenimento Digital*. [S.l.]: UNISINOS, 2007. Citado na página 24.

DA ROSA, J. et al. Co-evolution of antagonistic intelligent agents using genetic algorithms. *Procedia Computer Science*, v. 18, p. 692 – 701, 2013. 2013 International Conference on Computational Science. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877050913003761>. Citado 3 vezes nas páginas 13, 22 e 25.

DARWIN, C. *On The Origin of Species by Means of Natural Selection*. [S.l.]: London: John Murray, 1859. Citado 2 vezes nas páginas 12 e 15.

FOLETTO, J. C. *Programação genética recorrente aplicada ao problema presa-predador*. Monografia (Trabalho de Conclusão de Curso) — Universidade Regional do Noroeste do Estado do Rio Grande do Sul, 2015. Citado na página 13.

GABRIEL, P. H. R.; DELBEM, A. C. B. *Fundamentos de algoritmos evolutivos*. 2008. ICMC-USP. Disponível em: <http://conteudo.icmc.usp.br/CMS/Arquivos/arquivos_enviados/BIBLIOTECA_113_ND_75.pdf>. Citado 3 vezes nas páginas 6, 15 e 19.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. [S.l.]: Addison-Wesley Longman Publishing Co., Inc., 1989. Citado na página 16.

HOLLAND, J. H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. [S.l.]: MIT Press, 1992. Citado 2 vezes nas páginas 12 e 17.

JANG, S.-H.; YOON, J.-W.; CHO, S.-B. Optimal strategy selection of non-player character on real time strategy game using a speciated evolutionary algorithm. In: *2009 IEEE Symposium on Computational Intelligence and Games*. [S.l.: s.n.], 2009. p. 75–79. Citado na página 12.

KOZA, J. R. *Genetic programming: on the programming of computers by means of natural selection*. [S.l.]: MIT Press, 1992. Citado na página 12.

MANDL, S.; STOYAN, H. Evolution of agent coordination in an asynchronous version of the predator-prey pursuit game. In: LINDEMANN, G. et al. (Ed.). *Multiagent System Technologies*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 47–57. Citado na página 21.

MILLINGTON, I.; FUNGE, J. *Artificial Intelligence for Games, Second Edition*. 2nd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009. ISBN 0123747317, 9780123747310. Citado na página 22.

MITCHELL, M. *An Introduction to Genetic Algorithms*. [S.l.]: MIT Press, 1998. Citado na página 15.

POZO, A. et al. *Computação Evolutiva*. 2005. Departamento de Informática da Universidade Federal do Paraná. Disponível em: <http://www.inf.ufpr.br/aurora/tutoriais/Ceapostila.pdf>. Citado 5 vezes nas páginas 6, 16, 18, 19 e 20.

RUSSEL, S. J.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. [S.l.]: Prentice Hall, 2003. Citado na página 13.

SWEETSER, P.; WILES, J. Current ai in games: a review. *Australian Journal of Intelligent Information Processing Systems*, v. 8, n. 1, p. 24–42, 2002. Citado na página 12.

WITTKAMP, M. et al. Real-time evolutionary learning of cooperative predator-prey strategies. In: *Proceedings of the Thirty-fifth Australasian Computer Science Conference - Volume 122*. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2012. (ACSC '12), p. 81–90. ISBN 978-1-921770-03-6. Disponível em: <http://dl.acm.org/citation.cfm?id=2483654.2483664>. Citado 2 vezes nas páginas 22 e 25.

YANNAKAKIS, G. N. *AI in Computer Games: Generating Interesting Interactive Opponents by the use of Evolutionary Computation*. Tese (Doutorado) — University of Edinburgh. College of Science and Engineering. School of Informatics., 2005. Disponível em: <https://era.ed.ac.uk/handle/1842/879>. Citado 3 vezes nas páginas 12, 21 e 23.

YANNAKAKIS, G. N.; TOGELIUS, J. A panorama of artificial and computational intelligence in games. *IEEE Transactions on Computational Intelligence and AI in Games*, v. 7, n. 4, p. 317–335, dec 2015. Disponível em: <https://ieeexplore.ieee.org/document/6855367>. Citado na página 12.