Research, Society and Development, v. 9, n.1, e129911793, 2020
(CC BY 4.0) | ISSN 2525-3409 | DOI: http://dx.doi.org/10.33448/rsd-v9i1.1793

Ensino de agentes inteligentes por meio de problemas em jogos

Teaching intelligent agents through game problems

Enseñar agentes inteligentes através de problemas de juego

Recebido: 03/10/2019 | Revisado: 09/10/2019 | Aceito: 17/10/2019 | Publicado: 29/10/2019

Gustavo Augusto Silva
ORCID: https://orcid.org/0000-0002-6856-086X
Vixteam Consultoria & Sistemas, Brasil
E-mail: gustavo.silva.ufes@gmail.com
Luís Otávio Rigo Júnior
ORCID: https://orcid.org/ 0000-0002-7119-3095
Universidade Federal do Espírito Santo, Brasil
E-mail: luis.rigo@ufes.br

Resumo

O ensino de agentes inteligentes, apresentado neste estudo, foi realizado por meio do método de Aprendizagem Baseada em Problemas. O jogo Pac-Man foi selecionado como estudo de caso. O processo de aprendizado construtivo foi executado mediante três etapas, sendo elas: (i) a implementação de agentes básicos, com objetivo apenas de completar o percurso no mapa; (ii) a implementação de agentes reais, considerando a existência de fantasmas; e, por fim, (iii) a implementação de agentes inteligentes com capacidade de aprendizado. Como parte do processo de construção do conhecimento, cada novo agente proposto pelo discente foi criado com a finalidade de solucionar problemas encontrados nas análises de desempenho de agentes anteriores. Na primeira etapa de desenvolvimento foi observada uma melhora de 33,45% no desempenho do Agente 1 para o Agente 6. Na segunda etapa, considerando o jogo real, o Agente 8 apresentou incremento de desempenho de 20,49% quando comparado ao Agente 7. Na terceira etapa, foram utilizadas redes neurais artificiais e algoritmos genéticos, o que permitiu criar um agente capaz de aprender e completar o mapa sozinho. Assim, foi possível comprovar que as técnicas selecionadas mostraram-se eficientes ao melhorar o nível de inteligência dos agentes propostos para o jogo em questão. Além disso, o emprego deste método de ensino resultou em um maior envolvimento do discente com a disciplina de Inteligência Artificial, favorecendo o domínio deste aluno em técnicas de construção de

agentes inteligentes, bem como contribuindo para maior interesse do mesmo por esta área de estudo.

Palavras-chave: Aprendizagem Baseada em Problemas; Agentes Inteligentes; Jogos; Estratégias de Busca; Algoritmos Genéticos; Redes Neurais Artificiais.

Abstract

The teaching of intelligent agents, presented in this study, was performed through the Problem Based Learning method. Pac-Man was selected as a case study. The constructive learning process was carried out through three stages, namely: (i) the implementation of basic agents, with the sole purpose of completing the route on the map; (ii) the implementation of real agents, considering the existence of ghosts; and, finally, (iii) the implementation of intelligent learning agents. As part of the knowledge building process, each new agent proposed by the student was created to solve problems found in the performance analysis of previous agents. In the first stage of development a performance improvement of 33.45% was observed from Agent 1 to Agent 6. In the second stage, considering the actual game, Agent 8 showed a performance increase of 20.49% when compared to Agent 7. In the third stage, artificial neural networks and genetic algorithms were used, which allowed us to create an agent capable of learning and completing the map alone. Thus, it was possible to prove that the selected techniques were efficient in improving the intelligence level of the agents proposed for the game in question. In addition, the use of this teaching method resulted in a greater involvement of the student with the Artificial Intelligence discipline, favoring the student's mastery in intelligent agent construction techniques, as well as contributing to his interest in this area of study.

Keywords: Problem Based Learning; Intelligent Agents; Games; Search Strategies; Genetic Algorithms; Artificial Neural Networks.

Resumen

La enseñanza de agentes inteligentes, presentada en este estudio, se realizó a través del método de Aprendizaje Basado en Problemas. Pac-Man fue seleccionado como estudio de caso. El proceso de aprendizaje constructivo se llevó a cabo a través de tres etapas, a saber: (i) la implementación de agentes básicos, con el único propósito de completar la ruta en el mapa; (ii) la implementación de agentes reales, considerando la existencia de fantasmas; y, finalmente, (iii) la implementación de agentes de aprendizaje inteligente. Como parte del proceso de construcción de conocimiento, cada nuevo agente propuesto por el estudiante fue

creado para resolver problemas encontrados en el análisis de desempeño de agentes anteriores. En la primera etapa de desarrollo, se observó una mejora del rendimiento del 33.45% del Agente 1 al Agente 6. En la segunda etapa, considerando el juego real, el Agente 8 mostró un aumento del rendimiento del 20.49% en comparación con el Agente 7 En la tercera etapa, se utilizaron redes neuronales artificiales y algoritmos genéticos, lo que nos permitió crear un agente capaz de aprender y completar el mapa solo. Por lo tanto, fue posible demostrar que las técnicas seleccionadas fueron eficientes para mejorar el nivel de inteligencia de los agentes propuestos para el juego en cuestión. Además, el uso de este método de enseñanza resultó en una mayor participación del estudiante en la disciplina de Inteligencia Artificial, favoreciendo su dominio en las técnicas de construcción inteligente de agentes, y contribuyendo a su interés en esta área de estudio.

Palabras clave: Aprendizaje Basado en Problemas; Agentes Inteligentes; Juegos; Estrategias de Búsqueda; Algoritmos Genéticos; Redes Neuronales Artificiales.

1. Introdução

A Inteligência Artificial (IA) é uma área do conhecimento cujo objetivo principal é reproduzir o comportamento dos seres vivos em ambientes artificiais. Os Agentes Inteligentes são conhecidos como sistemas computacionais que reproduzem este comportamento em diversos níveis, desde uma simples reação a um estímulo até a capacidade de aprender com os erros. O estudo destes agentes, bem como dos ambientes em que estão inseridos, tem papel fundamental no processo de descoberta do conhecimento e da reprodução do comportamento inteligente (Russell & Norvig 2009).

Nos últimos anos, a disciplina de IA tem ganhado cada vez mais espaço nas matrizes curriculares de Cursos de Ciências Exatas e de Engenharia. Apesar deste avanço, o ensino de IA tem sido implementado por meio de metodologias tradicionais, no qual o aprendizado do estudante ocorre de maneira passiva e ineficiente. Assim, a aplicação de metodologias ativas, na qual o aluno é o agente responsável pelo seu aprendizado, têm sido apontadas como técnicas de ensino alternativas. Um exemplo deste tipo de abordagem é a Metodologia de Aprendizagem Baseada em Problemas (Problem Based Learning – PBL) (Silva et al., 2018).

Em uma outra vertente, jogos de computadores também têm sido utilizados como ferramentas no ensino, sendo aplicados com o intuito de auxiliar no desenvolvimento do raciocínio lógico e das capacidades cognitivas e do aprendizado (Costa & Neto, 2009; Grando & Tarouco, 2008). Além do mais, quando se fala em IA, existe uma associação

natural aos jogos.

Ao conhecer um jogo novo, o ser humano realiza um processo construtivo de aprendizado, definindo e aprimorando estratégias para aprender a jogá-lo. Neste processo, estratégias são definidas, soluções geradas e os resultados são verificados. Em seguida, novos problemas são gerados e o processo de construção do conhecimento se repete. Ou seja, como este já é um processo natural de aprendizado dos seres humanos, por que não utilizá-lo para ensinar conceitos em uma determinada disciplina?

Diante do exposto, o presente trabalho teve como objetivo descrever a aplicação do processo de PBL no ensino e aprendizagem de agentes inteligentes, na disciplina de Inteligência Artificial, ofertada para cursos de Ciências Exatas. O intuito da aplicação do referido processo foi avaliar o interesse e o desempenho do discente no domínio das estratégias estudadas. Assim, foi proposto um método de ensino construtivo, a partir da introdução/uso dos tipos mais básicos de agentes inteligentes. Neste processo, o objetivo foi possibilitar ao discente a construção de um agente capaz de jogar o jogo sozinho, simulando o comportamento do ser humano. Para a realização deste estudo de caso, foi escolhido o jogo Pac-Man, cuja interface foi desenvolvida pelo discente em outra disciplina.

2. Metodologia

Muitos dos problemas existentes no mundo dos jogos podem ser vistos como problemas de busca e, assim, resolvidos através de estratégias de busca (Cormen et al., 2009). Por exemplo, sair de um ponto do mapa até outro, ou então descobrir qual é a melhor jogada a ser realizada em um jogo de xadrez. Em todos os jogos, o objetivo de uma busca é encontrar o movimento que ofereça a melhor chance de se chegar a um objetivo final, que normalmente é a vitória. Algoritmos de busca, segundo Bastos & Jaques, (2010), são mecanismos de resolução de problemas que sistematicamente exploram alternativas e encontram uma sequência de ações para a solução do problema dado. Esta busca pode ser com informação, quando se tem alguma ideia de onde procurar a solução, ou pode ser sem informação (também chamada de busca cega), quando não se tem nenhuma informação acerca do problema (Pinheiro et al., 2009).

O Pac-Man é um jogo do tipo predador-presa que consiste de dois tipos de agentes: o Pac-Man e os fantasmas. O jogo se passa em um labirinto (mapa) de corredores com diversas bolinhas, onde o objetivo é fazer com que o Pac-Man percorra o mapa e coma todas as bolinhas sem ser capturado pelos fantasmas. Ele deve fugir dos fantasmas presentes no mapa,

exceto quando ele come as bolinhas maiores. Pois ao comê-las, o Pac-Man entra temporariamente em modo de perseguição, o que lhe dá maior velocidade e a habilidade de poder comer os fantasmas. Alguns mapas podem ter passagens que levam os agentes para o lado oposto do mapa (Gallagher & Rayan, 2003).

O simulador do jogo Pac-Man (Figura 1) e todas as estratégias que descritas neste trabalho foram desenvolvidas na linguagem C++, com a biblioteca gráfica OpenGL, que é uma API (Application Programming Interface) rápida, simples e interativa para modelagem bidimensional (Oliveira et al., 2014).

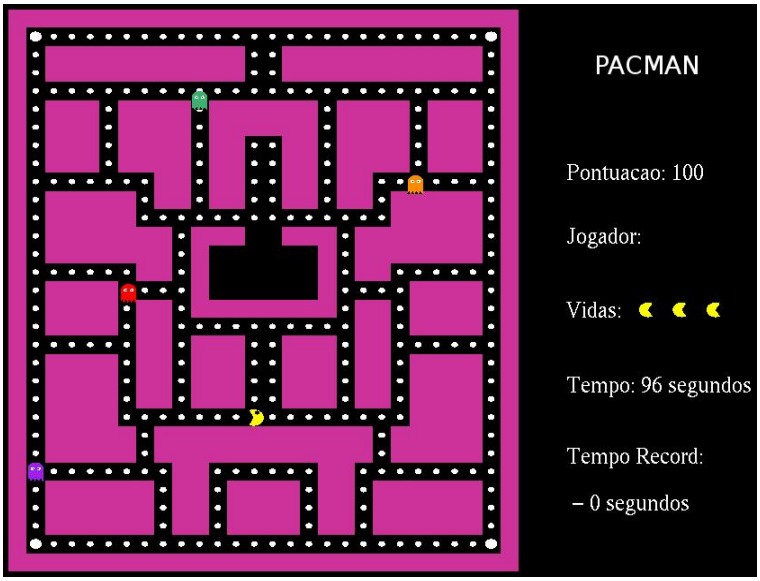

Figura 1. Simulador do jogo Pac-Man.
Fonte: Elaborado pelos autores.

A Figura 1 apresenta a interface do jogo desenvolvido pelos autores neste estudo de caso. Nesta, é apresentado um exemplo de labirinto, contendo o Pac-Man, fantasmas e bolinhas a serem comidas pelo Pac-Man. Nesta interface, também são apresentadas algumas informações ao jogador, como a sua pontuação, o número de vidas, o tempo transcorrido e o melhor tempo do jogador até o momento.

Podemos entender o agente principal do jogo, o Pac-Man, como um conjunto de estratégias de busca. Por outro lado, o Pac-Man também é um agente baseado em objetivos (Russel & Norvig, 2009), pois suas estratégias devem ser definidas tanto em função do seu objetivo principal, percorrer o ambiente para comer todas as bolinhas, bem como pelos seus objetivos secundários: realizar o menor percurso possível, não ser comido pelos fantasmas ou comer o maior número deles.

O estudo e implementação dos agentes foi dividido em três etapas. São elas:

- Em um primeiro momento, o discente foi responsável por implementar estratégias básicas de busca para o Pac-Man, que o permitia percorrer o mapa e comer as bolinhas, sem a presença de fantasmas. Nesta etapa, o único objetivo dos agentes criados era a conclusão do percurso completo do mapa;
- A segunda etapa do processo de construção do conhecimento, consistiu na ativação dos fantasmas e a implementação de estratégias mais elaboradas que permitiam ao Pac-Man fugir, caso os fantasmas se aproximassem. Com isto, o agente (Pac-Man) inclui novos objetivos secundários ao problema, tais como a decisão sobre o caminho a seguir a fim de evitar que os fantasmas o capturassem;
- O terceiro e último estágio de descoberta do conhecimento foi a implementação da capacidade do personagem Pac-Man de aprender com as suas decisões.

Em cada pequena etapa do processo de ensino e aprendizagem dos agentes, após a implementação da estratégia, ocorreu a verificação de problemas desse agente e estudo de novas estratégias que pudessem resolver os problemas identificados. Além da verificação pontual de possíveis problemas, estabeleceu-se um conjunto de métricas para análise de desempenho, que serão definidas nas subseções de análise experimental.

3. Agentes em mapas sem presença de fantasma

As três primeiras implementações que serão descritas aqui foram realizadas utilizando a estratégia busca em profundidade para percorrer o mapa. Sua característica básica é seguir em frente no mapa sempre que houver alguma célula não visitada, retrocedendo caso contrário. Os agentes implementados nesta fase, memorizam os pontos já visitados através de uma estrutura de dados chamada pilha.

Os agentes visualizam o mapa como uma matriz de células. A existência de uma conexão entre as células representa um caminho ou direção a seguir. A Figura 2 contém três exemplos de células e suas possíveis direções (conexões).

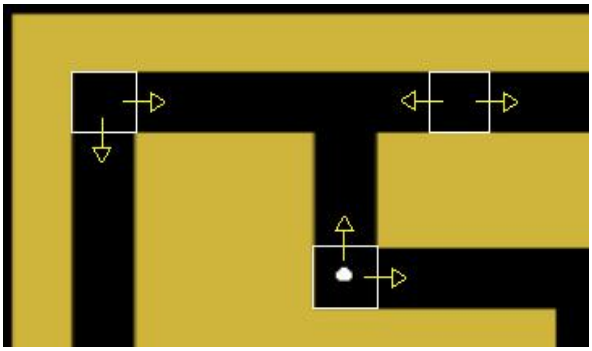

Figura 2. Células do mapa.
Fonte: Elaborado pelos autores.

Observa-se nesta figura, que, ao ocupar qualquer uma das três células sinalizadas (quadrados em branco), o agente tem as opções de caminho indicadas pelas setas. No jogo desenvolvido, além da direção possível para o agente, cada célula armazena também a informação sobre existência ou não de bolinha.

3.1. Agente 1 - Primeira estratégia

Nesta primeira implementação, a escolha pelos caminhos é predefinida, ou seja, o agente (Pac-Man) sempre dará prioridade em seguir uma determinada direção. Quando não é mais possível seguir naquela direção, uma nova direção é escolhida. Em cada movimento que o agente faz, ele empilha o movimento escolhido e a informação de direção da célula é apagada. Assim, caso o agente acesse uma célula considerada um beco sem saída (todas as direções já fora exploradas), ele deverá efetuar o caminho de retorno até que encontre uma célula que possua pelo menos uma direção inexplorada. Dessa forma, a busca irá terminar quando todos os caminhos do mapa forem explorados.

Ao estudar o resultado desta estratégia para alguns mapas, detectou-se que o agente faz muitas visitas desnecessárias a algumas células. O problema ocorre, pois as células podem ser visitadas por diferentes caminhos e o agente, nesta primeira implementação, só descobre se existe uma bolinha ao acessar a célula. Desta maneira, é possível que uma célula seja visitada várias vezes em um percurso.

3.2. Agente 2 - Segunda estratégia

O Agente 2 foi criado para corrigir o problema existente no anterior. Nesta implementação, o agente é capaz de olhar para as células ao seu redor, antes de realizar o movimento. Se a célula não possuir bolinha, o agente não irá visitá-la a menos que esteja realizando o caminho de retorno.

Outro problema detectado no Agente 1, que permanece no Agente 2, ocorre no momento em que o agente decide fazer o caminho de retorno. Em alguns casos, o agente executa um caminho de retorno maior que o necessário. Um exemplo desse problema é mostrado na Figura 3.

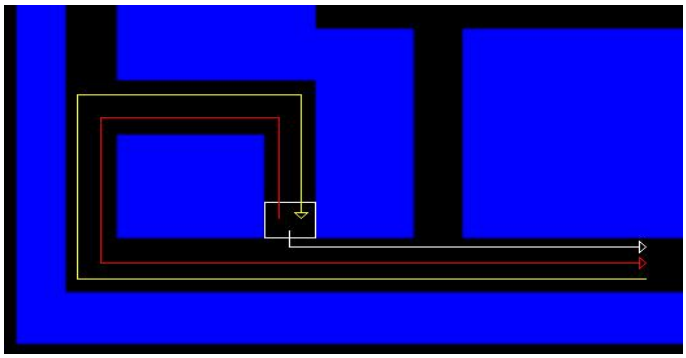

Figura 3. Problema com os ciclos.
Fonte: Elaborado pelos autores.

Nesta figura, após o agente fazer o caminho marcado em amarelo, ele verifica que a próxima célula está vazia e decide que deve retornar. Ao invés de fazer o caminho de volta marcado em vermelho, o ideal seria que o agente fizesse o caminho marcado em branco, uma vez que parte do caminho vermelho já foi percorrido e não há mais nada nele.

3.3. Agente 3 - Terceira estratégia

Com o objetivo de corrigir o problema remanescente das estratégias anteriores e, consequentemente, melhorar o desempenho do agente, foram definidas características adicionais ao Agente 3. Neste, antes de realizar o retorno, o agente (Pac-Man) faz uma simulação desse retorno (desempilhando temporariamente sua pilha de memória). A simulação de retorno acaba em duas situações: (i) quando a célula à sua frente é acessada (fechamento de um ciclo) e (ii) quando uma célula com caminhos não explorados é acessada. Em (i), o agente elimina o ciclo da memória e realiza o movimento para a célula seguinte. Em (ii), o retorno é efetivamente realizado pelo agente até a célula que contém os caminhos inexplorados.

3.4. Agente 4 - Quarta estratégia

O próximo agente implementa a estratégia de busca com informação. Para este agente, foi implementada a estratégia de busca conhecida como A*, uma das estratégias de busca mais utilizadas em jogos (Pinheiro et al., 2009). Ela combina uma função de custo g(n) com uma função heurística h(n). Onde, g(n) corresponde ao custo para alcançar cada célula adjacente n e h(n) corresponde o custo de ir da célula adjacente n até a célula objetivo que contém a bolinha mais próxima (Russell & Norvig, 2009). Na implementação, foi adotado

valor de g(n)=1 para todas as direções. A heurística utilizada no cálculo do h(n) foi a Distância Euclidiana, que calcula a distância em linha reta da célula alvo a partir da célula adjacente n (Deza & Deza, 2016). Assim, o agente escolhe o movimento para a célula que apresenta o menor valor de custo g(n)+h(n). Destaca-se que esta heurística é uma estimativa otimista que nem sempre representa o custo do caminho real até uma bolinha.

Apesar de ser uma busca informada e conduzir o agente em direção a bolinha mais próxima, em algumas situações o agente fica "preso". Isto ocorre quando qualquer movimento do agente o deixará mais distante da bolinha mais próxima. A Figura 4 demonstra um exemplo desta situação.

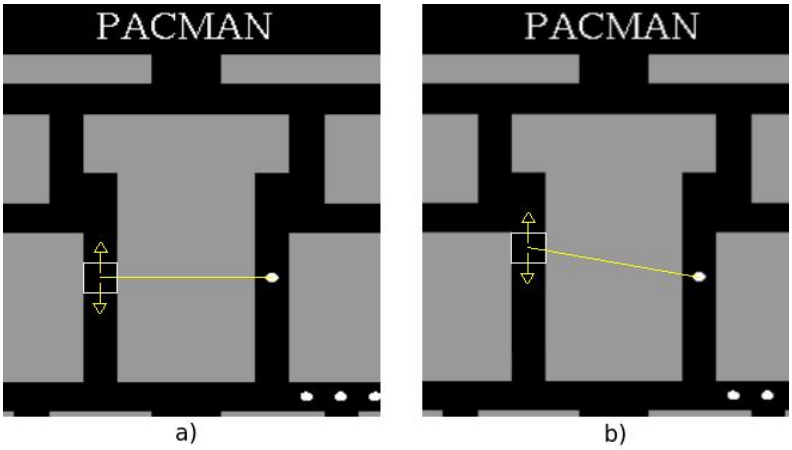

Figura 4. Problema do ciclo infinito.
Fonte: Elaborado pelos autores.

Nesta figura, ao realizar este movimento do cenário 'a' para cima, resultando no cenário 'b', o agente decidirá retornar à posição anterior (cenário 'a'), que está mais próxima da bolinha. Desta maneira, o agente fica preso em um ciclo infinito, sem conseguir evoluir em direção ao objetivo.

3.5. Agente 5 - Quinta estratégia

Para corrigir o problema anterior, após escolher o movimento e a boinha de destino, o Agente 5 memoriza as células pelas quais passa e não retorna nelas enquanto a bolinha de destino não for comida ou enquanto não estiver em um beco sem saída. Este agente também introduziu uma nova função heurística h(n), a Distância de Manhattan (Deza & Deza, 2016), como parte da estratégia A*.

Um novo problema com esta estratégia ocorre em alguns mapas, onde o caminho traçado pelo agente não corresponde ao menor caminho entre a sua posição atual e a bolinha

mais próxima. Um exemplo deste problema é apresentado na Figura 5.

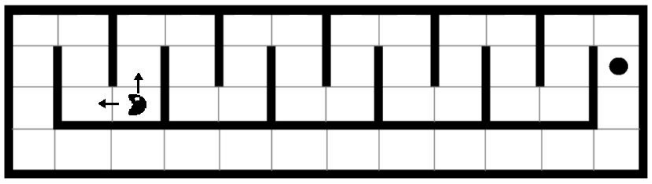

Figura 5. Problema na escolha do percurso.
Fonte: Elaborado pelos autores.

Como pode ser observado nesta figura, o agente escolherá o caminho mais longo (célula acima) ao invés do caminho mais curto (célula à esquerda). Este problema ocorre porque, no momento de escolher entre 2 caminhos (sinalizado no exemplo pela seta) a célula de cima está mais próxima da bolinha do que a célula à esquerda.

3.6. Agente 6 - Sexta estratégia

Como um próximo passo do estudo, optou-se por implementar uma estratégia de busca pelo menor caminho (ou "pathfinding", em Inglês). Esta é um tipo de estratégia aplicada em jogos do tipo Tower Defense (Patel, 2018). Neste tipo de jogo, os agentes seguem para seu objetivo utilizando um menor caminho real a partir do seu ponto inicial.

Para o Agente 6, portanto, realiza-se uma implementação da estratégia de busca pelo menor caminho por meio da simulação de um "espalhamento" no mapa, a partir do ponto atual do agente. O espalhamento nada mais é do que uma busca em largura off-line pela primeira bolinha. A Figura 6 apresenta um exemplo deste espalhamento.

Figura 6. Exemplo de estratégia de espalhamento para escolha do menor caminho.
Fonte: Elaborado pelos autores.

Nesta figura, cada número representa a distância da célula até a posição do agente. Números em verde representam células sem bolinha e os números em amarelo representam células com bolinha. Ao final dessa simulação, o agente percorre efetivamente apenas o menor caminho até a primeira célula com bolinha. Neste exemplo, o agente terá 4 opções de caminhos cuja primeira bolinha encontra-se à uma distância de 9 células.

Um ponto positivo desta estratégia é possibilidade de utilizar o espalhamento para encontrar qualquer elemento do mapa (bolinhas, bolinhas maiores, fantasmas ou passagens).

3.7. Análise experimental

Após o estudo e implementação de 6 agentes diferentes, realizou-se uma análise experimental comparativa por meio da execução das estratégias em 17 mapas distintos. Cada estratégia foi executada uma única vez em cada mapa, pois elas não envolvem componentes aleatórios.

O resumo dos resultados encontram-se descritos na Tabela 1, que apresenta a média de passos realizados e o tempo médio gasto (em segundos) pelo agente até comer a última bolinha dos mapas. Cada linha descreve o critério analisado e os valores que estão em negrito representam os melhores resultados obtidos.

Tabela 1 – Média de passos e de tempo gasto (em segundos) pelos agentes nos 17 mapas.

Critério	Age. 1	Age. 2	Age. 3	Age. 4	Age. 5	Age. 6
Passos	620,6	560,1	528,6	427,0	421,2	**413,0**
Tempo (s)	282,5	252,6	237,8	188,0	185,8	**180,8**
Melhor	0	0	0	2	4	**11**
Qtd. Mapas	17	17	17	3	17	17

Fonte: Elaborado pelos autores.

Como pode ser observado na Tabela 1, mesmo com as otimizações que foram realizadas nas três primeiras técnicas, estas ainda assim apresentaram resultados piores de tempo e passos, se comparadas com os demais agentes. Este desempenho ruim ocorre em virtude da aplicação de busca cega pelos primeiros agentes. Porém, apesar do Agente 4 apresentar bom resultado médio, ele não foi muito eficaz pois não conseguiu finalizar 14 dos 17 mapas.

A análise do tempo gasto pelos agentes é necessária como parâmetro complementar,

pois algumas estratégias envolvem maior tempo de processamento para simular os possíveis caminhos e realizar a sua escolha de direção. Apesar desse tempo extra para a simulação dos passos, os Agentes 5 e 6 também são melhores com relação ao critério de tempo médio de execução.

Por fim, é possível verificar que o Agente 6 apresentou melhor resultado na maioria dos mapas, com a menor quantidade de passos em 11 mapas. Isto ocorre pois o agente sempre segue pelo menor caminho até chegar à bolinha mais próxima. Desta maneira, o Agente 6 representa o maior grau de inteligência quando comparado aos demais.

Até o presente momento o objetivo das estratégias desenvolvidas era o de completar o mapa do Pac-Man comendo todas as bolinhas. Objetivo este que fora cumprido por 5 das 6 estratégias implementadas.

4. Agentes em mapas com presença de fantasma

O próximo estágio no processo de aprendizagem baseada em problemas (PBL) trata-se da inclusão dos fantasmas nos mapas. Dessa forma, o problema não se restringe mais a percorrer o mapa em busca de todas as bolinhas. Mas agora, em determinados momentos, o agente (Pac-Man) deverá decidir se deve continuar seguindo por um caminho, retornar ou optar por outro caminho, a fim de fugir dos fantasmas. Isso faz com que o grau de dificuldade e o tempo de execução aumentem.

4.1. Estratégia dos fantasmas

Os fantasmas foram implementados com o seguinte conjunto de estratégias: (i) realizam movimentos aleatórios na maior parte do tempo; (ii) quando o Pac-Man estiver próximo, utilizam a estratégia do caminho mais curto para persegui-lo; e, (iii) quando o Pac-Man estiver próximo mas estiver em modo de perseguição, o fantasma faz o movimento de fuga.

4.2. Agente 7 - Sétima estratégia

Este agente implementa uma variação da sexta estratégia. Antes de realizar o espalhamento (busca em largura off-line), o agente bloqueia em sua memória todas as células próximas aos fantasmas. A Figura 7 apresenta a simulação do espalhamento realizado pelo

agente.

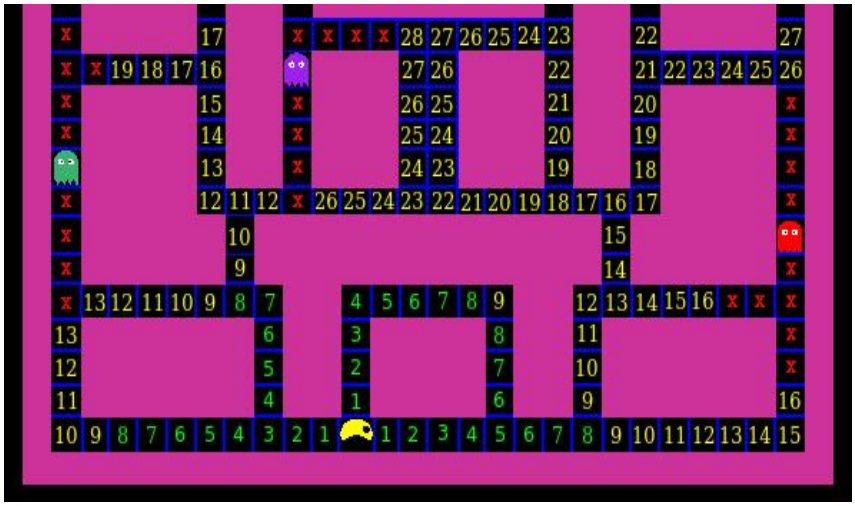

Figura 7. Exemplo de espalhamento com "bloqueio" de caminhos próximos aos fantasmas.
Fonte: Elaborado pelos autores.

No exemplo apresentado, as células que o agente não deve acessar (4 células mais próximas de cada fantasma) estão sinalizadas por um X em vermelho. No momento de realizar a busca em largura pela bolinha mais próxima, o agente não incluirá caminhos que passem perto dos fantasmas. Feita a simulação do espalhamento, o agente realiza o melhor percurso disponível.

Uma desvantagem desta implementação é que, após traçado o caminho entre o agente e a bolinha mais próxima, o agente realiza o percurso mesmo que no meio do percurso surja um fantasma ou ele passe próximo de outra bolinha, que agora está liberada por afastamento do fantasma. Ou seja, o agente não faz nenhum ajuste em tempo real (online) de percurso percorrido e, em algumas vezes, acaba morrendo.

4.3. Agente 8 - Oitava estratégia

Este novo agente realiza o ajuste de caminhos a cada passo do jogo. Por causa desta verificação, ele não se aproximará dos fantasmas e incluirá novas possibilidades toda vez que um fantasma se afastar.

4.4. Análise experimental

Com o objetivo de realizar a análise dos dois últimos agentes implementados, foi

preciso estabelecer um novo conjunto de experimentos e métricas de análise de desempenho. Como estas novas estratégias operam em mapas com fantasmas, que possuem movimentos aleatórios, foi necessário repetir a execução do agente várias vezes para o mesmo mapa e extrair a média dos resultados. Além disso, foram incluídos dois novos critérios na análise do agente: número de vezes que o Pac-Man foi morto por algum fantasma e número de vezes que ele comeu um fantasma.

O resumo dos resultados obtidos encontram-se na Tabela 2, que apresenta os valores mínimos, máximos e médios dos dois agentes sobre os critérios de passos realizados, de tempo gasto (em segundos) pelo agente, do número de vezes em que ele foi morto e do número de vezes que ele capturou o fantasma. As colunas indicam o menor, o maior e a média dos resultados obtidos pelo agente considerando os 170 execuções (10 vezes por mapa). Além disso, os valores em negrito correspondem aos melhores resultados médios.

Tabela 2 – Média de passos e de tempo gasto (em segundos) pelos agentes nos 17 mapas.

	Agente 7			Agente 8		
Critério	**menor**	**maior**	**média**	**menor**	**maior**	**média**
Passos	451	2429	728,9	409	1107	**579,5**
Tempo (s)	202	1350	328,6	184	482	**247,6**
Mortes	0	113	7,4	0	5	**0,9**
Captura	0	5	1,0	0	7	**1,4**

Fonte: Elaborado pelos autores.

Como pode-se observar na Tabela 2, o Agente 8 obteve melhores resultados se comparado com o Agente 7, considerando todos os critérios desempenho adotados. Entretanto, em virtude da necessidade de fugir dos fantasmas, a média de passos e o tempo gasto pelo Agente 8 foi maior que para o Agente 6.

5. Agentes com capacidade de aprender

Nas etapas anteriores, foram desenvolvidos agentes cuja inteligência era definida pelo programador (discente). Ou seja, o agente “nascia” com toda inteligência e capacidade de executar as tarefas do ambiente em que estava inserido. Assim, se o programador não tivesse previsto uma determinada situação, o agente não poderia se adaptar a ela.

Entretanto, a capacidade de aprender com experiências, exemplos e com os erros é a principal característica que define o conceito de inteligência (Mitchell, 1997). Um agente

inteligente que possui a capacidade se adaptar, aprender e melhorar o seu desempenho representa o tipo de agente mais desejado para a maioria dos problemas reais (Russell & Norvig, 2009).

A subárea da IA que estuda algoritmos capazes de habilitar o aprendizado em agentes é chamada de Aprendizado de Máquina (Machine Learning – ML) (Mitchell, 1997).

Nesta última etapa do processo, foi realizada uma extensa revisão bibliográfica sobre a aplicação de ML para jogos. Em seguida, foi selecionado um método capaz de aprender e outro de ensinar. Por fim, foram implementados 4 agentes inteligentes com capacidade de aprendizado, efetuando melhorias nos parâmetros dos algoritmos.

5.1. Rede neural artificial multi-layer perceptron

Uma Rede Neural Artificial (RNA) é um sistema computacional constituído por unidades conhecidas como neurônios interligados, trabalhando em paralelo para realizar uma determinada tarefa (Haykin, 2001). As RNA's são bastante utilizadas em jogos para aprendizado dos parâmetros de controle dos personagens, tanto durante a etapa de desenvolvimento do jogo, quanto durante a execução dos jogos pelos usuários (Crocomo, 2008). Um exemplo desse tipo de aplicação pode ser visto em Crocomo (2008), onde as RNA's adaptam as estratégias dos agentes virtuais (não controlados pelos seres humanos) à estratégia utilizada pelos seres humanos (usuários do jogo).

A Multi-Layer Perceptron (MLP) é uma rede neural do tipo feedforward (Rumelhart et al., 1986), com uma camada de entrada, uma camada de saída e uma ou mais camadas escondidas. Nela, os neurônios de uma camada estão conectados com todos os neurônios das camadas adjacentes. A Figura 8 apresenta a estrutura de uma MLP padrão.

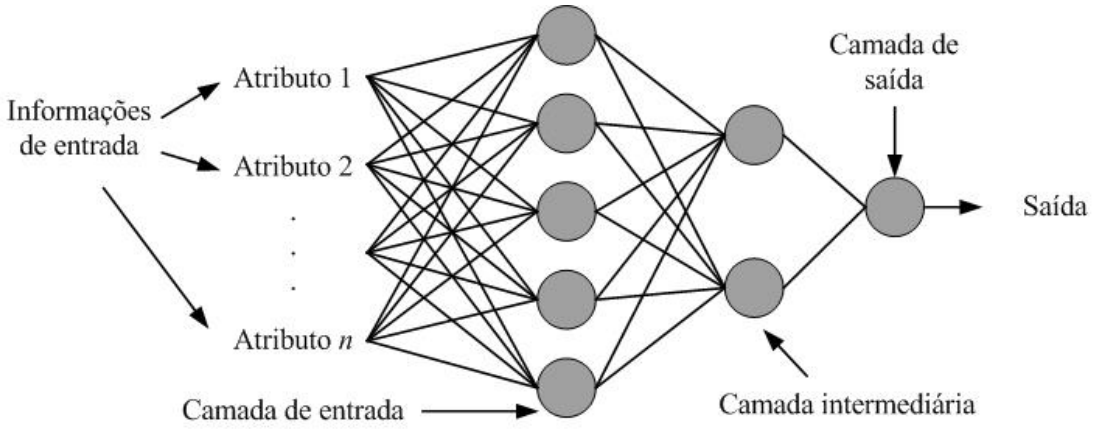

Figura 8. Ilustração da rede neural MLP.
Fonte: Elaborado pelos autores.

Nesta figura, os círculos em cinza representam os neurônios e as retas em preto representam as conexões entre eles. Os atributos são as informações de entrada de um determinado exemplo e são apresentados aos neurônios da primeira camada. Por fim, a saída corresponde à resposta da rede sobre aquele exemplo e encontra-se armazenada no neurônio da camada de saída.

A rede neural MLP implementada neste trabalho recebe 13 informações dos sensores do ambiente do Pac-Man. Como o agente pode conter até 4 células vizinhas (leste, oeste, norte e sul), calcula-se a distância (por espalhamento) da célula vizinha até a primeira célula que contém uma bolinha menor, maior e um fantasma. Quando a célula vizinha é uma parede, o valor da informação é sinalizada com 0. Por fim, é realizado um processo de normalização inversa dos resultados das distâncias. Por exemplo, a direção que contém a bolinha mais próxima recebe 1 e a direção com a bolinha mais distante recebe 0. A 13a informação é um flag (0 ou 1) que informa se o agente está em modo de perseguição. A Tabela 3 apresenta um resumo dessas informações.

Tabela 3 – Conjunto de entradas para a rede MLP.

Sensores	Descrição dos sensores
01 até 04	Distância até a bolinha mais próxima.
05 até 08	Distância até a bolinha maior mais próxima.
09 até 12	Distância até o fantasma mais próximo.
13	Sinalizador de modo de perseguição.

Fonte: Elaborado pelos autores.

Ao receber um conjunto de informações (sensores), a rede MLP realiza a propagação do sinal até os neurônios da camada de saída. A propagação do sinal em um neurônio ocorre através da aplicação da função de ativação sigmóide sobre a soma das entradas do neurônio ponderadas pelo pesos das conexões. A MLP implementada possui 4 neurônios na camada de saída (uma para cada direção). E, a direção escolhida pelo agente corresponderá ao neurônio da camada de saída que possuir o maior nível de ativação. Caso a rede escolha uma direção com parede, o agente seguirá na direção representada pelo segundo neurônio com maior ativação e assim sucessivamente.

O treinamento da RNA para reproduzir o comportamento desejado consiste no ajuste dos pesos (conexões entre os neurônios) através de exemplos e pode ser realizado por diversos algoritmos. Tradicionalmente, os dois algoritmos mais utilizados para o treinamento

de MLPs são o Backpropagation (Rumelhart et al., 1986) e o Levenberg–Marquardt (Marquardt, 1963). Entretanto, estes algoritmos realizam um aprendizado supervisionado que necessita de um conjunto grande de exemplos para que a rede possa aprender a reproduzir o comportamento desejado.

5.2. Algoritmo genético como método de aprendizado

Desenvolvido por Goldberg (1989), o Algoritmo Genético (AG) é um algoritmo de otimização inspirado na Teoria de Seleção Natural de Darwin (Darwin, 1859). O AG simula o comportamento evolucionário das espécies na sua adaptação ao meio, através dos operadores de reprodução, seleção natural, mutações genéticas e cruzamentos (Mitchell, 1996).

Assim como as RNAs, os AGs também são bastante utilizados dentro da área de jogos. Em Ficheman et al. (2006), é descrito um jogo educacional voltado para a educação no trânsito, onde o AG é utilizado para realizar a simulação da evolução de uma população de pedestres autônomos, que apresentam comportamento de cordo suas características genéticas. Em Barone & Silveira (1998), o AG é utilizado para definir os movimentos de dois agentes (abelha e pinguim), que competem entre si. A abelha tem o objetivo de encurralar o pinguim e o pinguim possui o objetivo de coletar os diamantes espalhados no mapa, sem ser encurralado.

Na internet, existem diversas aplicações com o uso de RNAs e AGs em conjunto, com a RNA responsável pelo comportamento do agente e o AG responsável pelo seu aprendizado. Em Seidel (2015), estas técnicas são aplicadas para que o computador aprenda a jogar o jogo do Dinossauro do Google, presente no navegador Google Chrome, sem a interferência de um usuário. Para tal, são utilizados sensores de distância, tamanho e velocidade do obstáculo para que a RNA determine o momento de pular o abaixar. O aprendizado é realizado pelo AG, aplicando o processo evolucionário sobre uma população de redes candidatas.

Outro exemplo é apresentado em Arzt (2016), onde os agentes são carros que devem percorrer uma pista. Cada agente tem o seu comportamento definido por uma rede neural que analisa as informações de sensores e define a direção a seguir. O aprendizado é realizado efetuando várias tentativas de percurso da pista. Em cada percurso, um novo conjunto de carros é gerado pelo AG.

Inspirado nestes exemplos, optou-se por aplicar o AG na busca pelo melhor conjunto de pesos para a rede neural. Para tanto, foi escolhido um AG do tipo geracional com elitismo (Mitchell, 1996). Em um AG geracional os indivíduos de uma geração são substituídos total

ou parcialmente por novos indivíduos na geração seguinte. O elitismo consiste na manutenção dos melhores indivíduos nas próximas gerações, sem sofrerem alteração nos seus genes. A seguir, serão descritos os demais componentes do AG implementados no presente trabalho.

A. *Indivíduos:* Um indivíduo é formado por um código genético (conjunto de genes). Um gene representará o futuro valor do peso das conexões entre os neurônios da MLP. Após alguns experimentos iniciais, optou-se por fixar a arquitetura da MLP com 8 neurônios na camada escondida e 4 neurônios na camada de saída, totalizando 136 conexões (13 x 8 + 8 x 4).

B. *População:* A primeira população de indivíduos é criada com a escolha aleatória de valores entre -1 e 1 para os seus genes. As demais populações são geradas através da simulação do processo de seleção natural. Primeiramente, os melhores indivíduos (elitismo) são automaticamente copiados para a próxima geração. Em sequência, 2 indivíduos são selecionados para realização do cruzamento genético e, possivelmente, da mutação, gerando 2 novos indivíduos que são adicionados a nova geração. O processo se repete até que a nova geração esteja completa.

C. *Aptidão do indivíduo:* O processo de avaliação dos indivíduos como solução para o problema é realizado pela execução do agente (Pac-Man) em um mapa. Nesta, os movimentos do agente são definidos através das respostas da MLP, calibrada com os pesos do indivíduo. Ao final da execução, verifica-se o número de bolinhas que o agente conseguiu comer e este valor é utilizado como aptidão do indivíduo. O processo é realizado uma única vez para cada indivíduo de uma geração.

D. *Processo de seleção:* A seleção dos indivíduos ocorre de acordo com o método de seleção por ranking linear (Rigo, 2005). Neste método, os indivíduos com uma maior aptidão têm maior chance de serem selecionados. Esta chance é dada por uma função linear onde o melhor indivíduo tem chance L de ser escolhido, o pior indivíduo tem chance 1 e todos os outros indivíduos tem probabilidade de serem escolhidos de acordo com sua posição no ranking.

E. *Cruzamento:* O cruzamento dos dois indivíduos selecionados é realizado, neste trabalho, pelo método conhecido como cruzamento uniforme, onde usa-se uma máscara binária aleatória para realizar a troca do material genético entre os indivíduos atuais (pais), gerando dois novos indivíduos (filhos) (Rigo, 2005).

F. *Mutação:* Após a geração dos filhos por cruzamento, é realizado um sorteio a fim de verificar se o operador de mutação será ou não aplicado. Caso decida-se aplicar o operador, escolhe-se aleatoriamente um subconjunto de genes para efetuar a

mutação, atribuindo um valor aleatório entre -1 e 1 para cada gene selecionado. A mutação possibilita o surgimento de novas características e de corrigir a perda de material genético potencialmente útil (Ribeiro, 2008).

G. *Critérios de parada:* O AG é finalizado quando atingir um número pré-determinado de gerações. Ao final do processo evolutivo, a população estará melhor adaptada ao problema (Darwin, 1859). E, como consequência, o indivíduo que detêm a melhor aptidão da última geração corresponde ao conjunto de pesos do melhor agente para o mapa escolhido.

A Figura 9 apresenta o mapa contendo 314 bolinhas, escolhido para a realização do último conjunto de experimentos.

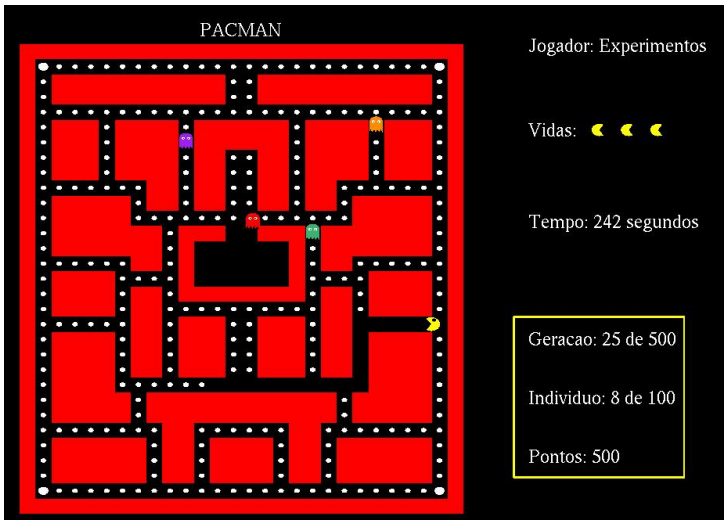

Figura 9. Nova interface do jogo Pac-Man com um exemplo de mapa.
Fonte: Elaborado pelos autores.

Esta figura também apresenta a nova interface gráfica, adaptada para mostrar algumas informações sobre a evolução do agente, onde são apresentados os pontos do agente, a geração e o indivíduo atual do AG (informando o número total de gerações e de indivíduos).

5.3. Agente 9 - Nona estratégia

O Algoritmo Genético responsável pela geração do Agente 9, foi configurado com o seguinte conjunto de parâmetros:

- Número de gerações: 500;
- Tamanho da população: 50;
- Genes por indivíduo: 136;
- Tamanho da elite: 10% da população;

- Fator de normalização (L) p/ seleção: 200;
- Taxa de mutação: 10% de chance de ocorrer;
- Genes modificados por mutação: 10% do indivíduo.

Com o objetivo de avaliar o processo de aprendizado dos novos agentes foi realizado o monitoramento da aptidão do melhor indivíduo de cada geração. Esse critério tem como objetivo monitorar a melhor solução encontrada a cada passo pelo AG. A Figura 10 mostra a aptidão do melhor indivíduo de cada geração do agentes implementados. Como pode ser observado, o Agente 9 foi capaz de evoluir, mas ao final do processo ele foi capaz de comer apenas 88 bolinhas, ou seja, 28,02% das bolinhas do mapa.

Também é possível observar que após 50 gerações a aptidão do melhor indivíduo evoluiu pouco. Após vários experimentos, com resultados parecidos, surgiu uma hipótese de que pudesse estar acontecendo uma convergência prematura do algoritmo.

Com o objetivo de analisar melhor o motivo do desempenho ruim do Agente 9, foi realizada uma análise mais detalhada dos indivíduos da população. Nesta, foi verificada a média e a variância da aptidão dos indivíduos em cada geração. As Figuras 11 e 12 apresentam, respectivamente, estes valores.

Como pode ser observado, em menos de 100 gerações, a aptidão média dos indivíduos convergiu para um valor entre 40 e 60 e a variância da população ficou em torno de 0.05. Ou seja, os indivíduos se tornaram rapidamente muito parecidos, o que caracteriza a situação de convergência prematura. Segundo Catarina & Bach (2003), uma maneira de evitar essa convergência prematura consiste em aumentar a taxa de mutação.

5.4. Agente 10 - Décima estratégia

Com o aumento da taxa de mutação, para 30%, na construção do Agente 10, foi possível observar uma melhora gradativa na aptidão do melhor indivíduo de cada geração. Entretanto, os valores de média e variância continuaram baixos após 100 gerações.

Ainda segundo Catarina & Bach (2003), uma outra forma de corrigir o problema da convergência prematura é através do aumento do tamanho da população, isso porque ela fornece uma cobertura mais representativa do espaço de busca. No entanto, trabalhar com uma população muito grande pode ser inviável, já que necessitará de maiores recursos computacionais. Segundo Rigo (2005), outra maneira de evitar a convergência prematura consiste em diminuir a pressão de seleção, dando mais chances aos indivíduos menos aptos.

5.5. Agente 11 - Décima primeira estratégia

Dessa forma, para a criação do Agente 11 foi utilizada uma população com o dobro de indivíduos (100) e o fator de normalização L foi reduzido para 150. Os resultados obtidos pelo Agente 11, sobre o mesmo mapa, demonstram que o problema da convergência prematura foi resolvido. Através da Figura 10, é possível observar que o AG demorou mais tempo para encontrar uma solução razoável, entretanto atingiu uma solução melhor que as anteriores.

Observa-se, também, que houve uma evolução gradativa na variância da população, comprovando a solução do problema anterior.

Porém, ainda não foi possível criar uma RNA capaz de concluir o mapa, comendo todas as 314 bolinhas e, ao mesmo tempo, fugindo dos fantasmas quando necessário.

5.6. Agente 12 - Décima segunda estratégia

Representando uma mudança mais radical, para realização deste último experimento foi utilizada uma pressão de seleção ainda mais baixa que os demais (L = 10).

Para este último agente, verificou-se melhoras consideráveis na sua aptidão próximo das 500 gerações. Assim, optou-se por continuar o aprendizado, aumentando o número máximo de gerações (1000).

5.7. Análise experimental

Um resumo dos resultados experimentais dos agentes com capacidade de aprendizado é apresentado a seguir. O primeiro fator analisado foi à evolução da aptidão do melhor indivíduo de cada geração do AG. Este critério é apresentado no gráfico da Figura 10.

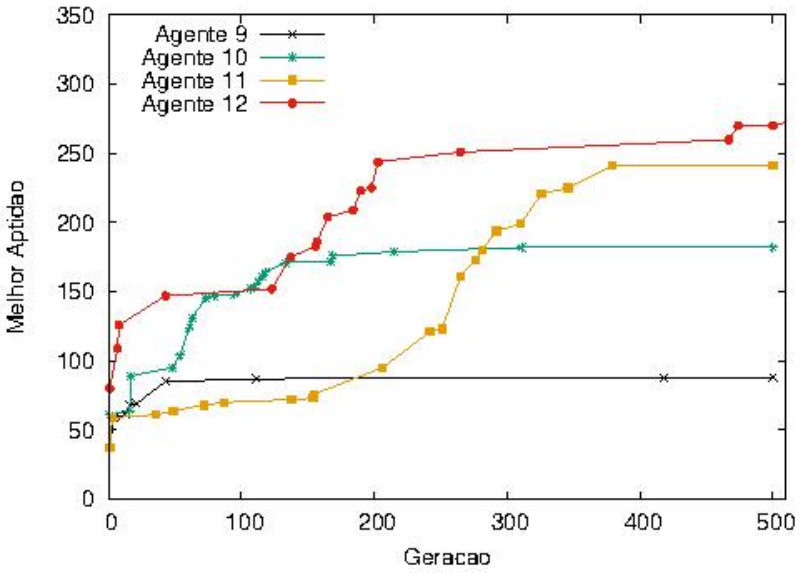

Figura 10. Evolução da aptidão dos agentes em cada geração.
Fonte: Elaborado pelos autores.

Na Figura 10, são apresentadas 4 curvas, correspondendo à melhor aptidão de cada um dos agentes 9, 10, 11 e 12. O ponto de interseção sinalizado pelas figuras geométricas representa os pontos em que houve evolução da aptidão do Agente. É possível observar que o houve uma melhoria progressiva na aptidão a cada agente criado, onde o Agente 12 obteve a melhor aptidão ao final do processo de aprendizado.

O segundo fator avaliado representa a evolução da aptidão média dos indivíduos da população do AG em cada passo do aprendizado. Este critério é apresentado no gráfico da Figura 11.

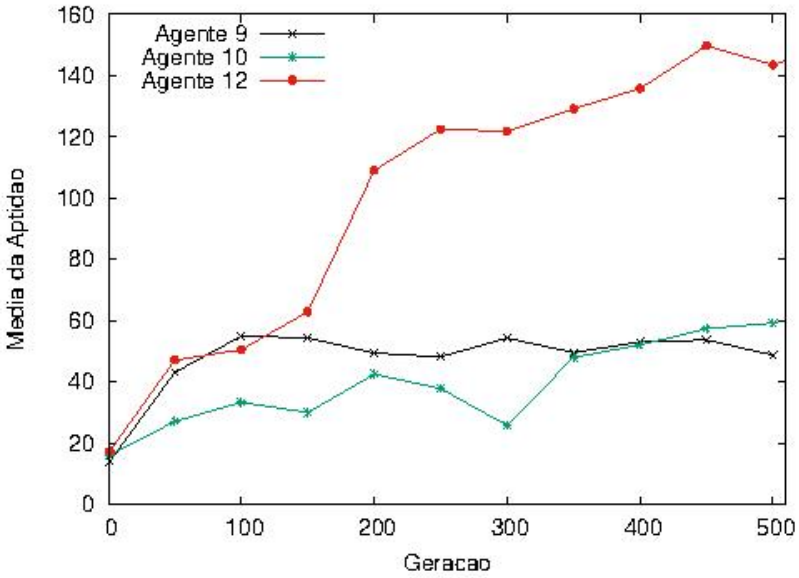

Figura 11. Média da aptidão da população em cada geração.
Fonte: Elaborado pelos autores.

Esta figura apresenta às curvas de evolução da aptidão média da população dos Agentes 9, 10 e 12. Nesta, as figuras geométricas representam os valores de a cada 50 gerações. Isto foi necessário para que os gráficos não ficassem tão ruidosos e a informação pudesse ser avaliada adequadamente. Observa-se que, na média, 50% do mapa concluído pelos indivíduos criados no processo de aprendizado do Agente 12. Uma melhora considerável, quando comparado aos primeiros experimentos, apenas 16% do mapa para os Agentes 10 e 12.

Por fim, o terceiro fator analisado nos experimentos foi a variância na aptidão da população em cada geração do AG. Este critério é apresentado no gráfico da Figura 11.

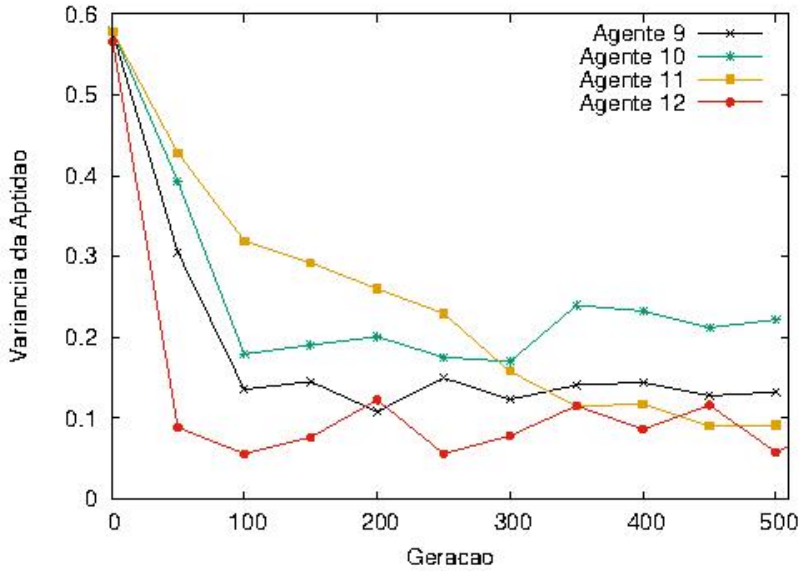

Figura 12. Variância da aptidão da população em cada geração.
Fonte: Elaborado pelos autores.

Nesta figura são apresentadas 4 curvas, correspondendo à melhor aptidão de cada um dos agentes 9, 10, 11 e 12. E, as figuras geométricas representam os valores de variância a cada 50 gerações. Como pode ser observado, a variância dos indivíduos da população foi aumentando à medida que os agentes 9, 10 e 11 foram criados. Entretanto, ao contrário do esperado, o Agente 12 apresentou a menor variância na aptidão dos seus indivíduos ao longo do processo de aprendizado.

Assim, dentre os experimentos realizados com aprendizado, conclui-se que o Agente 12 foi o que obteve o melhor resultado. Neste, por volta da geração de número 750, o Agente finalmente conseguiu completar o mapa. Destaca-se, também, que na geração 1000 desta execução, os membros da elite comeram em torno de 284 bolinhas (90,44% do mapa). Caso houvesse mais gerações, acredita-se que mais indivíduos conseguiriam completar o objetivo.

Considerações finais

O presente trabalho descreveu a aplicação de um estudo de caso (jogo Pac-Man) sobre o ensino de agentes inteligentes por meio da aplicação do método de Aprendizagem Baseada em Problemas.

Como pôde ser observado, cada agente foi proposto com base nos problemas descobertos na análise dos agentes anteriores. Como era de se esperar, em cada passo, houve um incremento no nível de inteligência do agente, necessário para resolver um problema verificado experimentalmente.

No primeiro estágio de desenvolvimento de agentes inteligentes (com o objetivo simples de completar o mapa sem fantasmas), foi possível notar melhora de 33,45% e 36% no desempenho do Agente 1 para o Agente 6, com relação ao número de passos e ao tempo gasto, respectivamente. Já, no segundo estágio de desenvolvimento de agentes (considerando o jogo real com fantasmas), foi possível notar a evolução do Agente 7 para o Agente 8, representando uma melhora de 20,49% no número de passos, de 24,65% no tempo médio e de 87,84% no número de mortes.

E por fim no terceiro estágio de desenvolvimento, foi utilizada duas técnicas de aprendizado de máquina (redes neurais artificiais e algoritmos genéticos) de modo que, ao invés de os agentes utilizarem estratégias pré-definidas para alcançar o objetivo, estes seriam capazes de criar sua própria estratégia. Depois de diversos experimentos realizados, um conjunto de parâmetros utilizado fez com que o agente conseguisse concluir o mapa sem morrer nenhuma vez, ou seja, este conseguiu comer todas as bolinhas do mapa enquanto fugia dos fantasmas.

Ao final do processo de ensino e aprendizagem dos agentes propostos, foi possível concluir que o discente apresentou um domínio mais fundamentado das estratégias estudadas, bem como, um maior interesse em continuar os estudos na área.

Outro benefício direto do trabalho realizado pelo discente foi a disponibilização de uma ferramenta de ensino prático de Inteligência Artificial. Além disso, a ferramenta foi implementada com suporte para fácil inclusão de novas estratégias, tanto para o Pac-Man quanto para os fantasmas. Desta maneira, ela servirá para a ampliação e experimentos com outros métodos de IA.

Referências

Arzt, S. (2016). Deep learning cars [Site]. Acesso em 20 de maio, em https://arztsamuel.github.io/ en/projects/unity/deepCars/deepCars.html

Barone, D. A. C. & Silveira, S. R. (1998). Jogos educativos computadorizados utilizando a abordagem de algoritmos genéticos. Congresso da Rede Iberoamericana de Informática na Educação, Brasília, DF, Brasil.

Bastos, R. & Jaques, P. (2010). Antares, um sistema web de consulta de rotas de ônibus como serviço público. Revista Brasileira de Computação Aplicada, 2(1), 41–56.

Catarina, A. S. & Bach, S. L. (2003). Estudo do efeito dos parâmetros genéticos sobre a solução otimizada e sobre o tempo de convergência em algoritmos genéticos com codificações binária e real. Acta Scientiarum. Technology, 25(2), 147–152.

Costa, N. M. S. & Netto, J. F. M. (2009). Desenvolvimento de um jogo educacional multiusuário usando bluetooth. Relatório de Iniciação Científica do Programa PIBIC 2008–2009, Universidade Federal do Amazonas, UFAM, Brasil.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L. & Stein, C. (2009). Introduction to Algorithms (3rd ed). Cambridge, Massachusetts: The MIT Press.

Crocomo, M. K (2008). Um algoritmo evolutivo para aprendizado on-line em jogos eletrônicos. (Dissertação de mestrado). Instituto de Ciências Matemáticas e de Computação – ICMC-USP, São Carlos, SP, Brasil.

Darwin, C. (1859). On the Origin of Species by Means of Natural Selection. Murray, London.

Deza, M. M. & Deza, E. (2016). Encyclopedia of Distances (4th ed.). Springer-Verlag.

Ficheman, I. K.; Assis, G. A.; Corrêa, A. G. D.; Netto, M. L. & Lopes, R. D. (2006). Educatrans: um jogo para educação no trânsito. Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação – SBIE), 19–21, UNB/UCB, Brasília, DF, Brasil, XVII SBIE.

Gallagher, M. & Rayan, A. (2003). Learning to play pac-man: An evolutionary, ruse-based approach. The 2003 Congress on Evolutionary Computation, CEC, Camberra, ACT, Australia. doi:10.1109/CEC.2003.1299397

Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning. MA: Addison-Wesley Reading.

Grando, A. & Tarouco, L. M. R. (2008). O uso de jogos educacionais do tipo rpg na educação. CINTED-UFRGS, Revista Novas Tecnologias na Educação, vol. 6, n. 1. doi: https://doi.org/10.22456/1679-1916.14403

Haykin, S. S. (2001). Redes Neurais - Princípios e Prática (2nd ed.). Bookman.

Marquardt, D. W. (1963). An algorithm for least-squares estimation of non-linear parameters. Journal of the Society for Industrial and Applied Mathematics, 11(2), 431–441.

doi:10.1137/0111030

Mitchell, M. (1996). An Introduction to Genetic Algorithms. Cambridge. MA: The MIT Press.

Mitchell, T. (1997). Machine Learning. McGraw Hill Higher Education.

Oliveira, F. F.; Piteri, M. A. & Menequette, M. (2014). Desenvolvimento de uma plataforma de software para a modelagem digital de terrenos baseada em tin. Boletim de Ciências Geodésicas, Revista UFPR, Curitiba, PR, Brasil, 20(1), 117-131. doi: 10.1590/S1982-21702014000100008

Patel, A. (2018). Pathfinding for tower defense cars [Site]. Acesso em 01 de novembro, em www.redblobgames.com/pathfinding/tower-defense/.

Pinheiro, E.; Kubo, C. C. ; Rangel, M. S. ; Arcari, T. A. & Dias, C. G. (2009). Navegação autônoma de um agente inteligente: Um estudo comparativo usando lógica fuzzy e algoritmo de busca a*. Exacta, São Paulo, SP, Brasil, 7(1), 87–98. doi: 10.5585/exacta.v7i1.1531

Rigo Jr., L. O. (2005). Evolução de autômatos celulares para a previsão de séries temporais correlacionadas. (Dissertação de mestrado). Programa de Engenharia de Sistemas e Computação / COPPE - UFRJ, Rio de Janeiro, RJ, Brasil.

Ribeiro, L. M. P. (2008). Otimização e dimensionamento de treliças panas de madeira empregando o método dos algoritmos genéticos. (Dissertação de mestrado). Programa de Engenharia Civil, Universidade Federal de Uberlândia, Uberlândia, MG, Brasil.

Rumelhart, D. E.; Hinton, G. E. & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533–536.

Russell, S. J. & Norvig, P. (2009). Artificial intelligence: a modern approach (3rd ed.). Pearson.

Seidel, I. (2015). Inteligência artificial com dinossauro da google [Video]. Acesso em 20 de maio, em https://www.youtube.com/watch?v=P7XHzqZjXQs&t=71s

Silva, A. B. D.; Bispo, A. C. K. A.; Rodriguez, D. G. & Vasquez, F. I. F. (2018). Problem-based learning: A proposal for structuring pbl and its implications for learning among students in an undergraduate management degree program. REGE Revista de Gestão, 25(2), 160–177. doi: 10.1108/REGE-03-2018-030

Porcentagem de contribuição de cada autor no manuscrito

Gustavo Augusto Silva – 70%

Luís Otávio Rigo Júnior – 30%