Universidade do Algarve

Faculdade de ciências e tecnologias

Uma arquitetura de subsunção com capacidades adaptativas preditivas para o Pacman

Osvaldo Francisco Lopes Mendes

Maio 2012

Abstract

Computer games are a very important area of study in the area of computational intelligence. This importance stems mainly from the properties of their environments: multi-agent, competitive, dynamic, stochastic, etc; where the verification of success or failure is easy to check. In addition, games and digital entertainment in general are a growing industry that generates a considerable turnover.

The goal of this work is to develop an agent to control the famous pacman, which it is able to participate in one of the most popular competitions organized by the conference IEEE Conference on Computational Intelligence and Games. The winner of this competition is who obtains the highest average score based on three attempts per ghost team. The goal of ghost team is to minimize that score The difficulty is due to the fact that it provides a stochastic, dynamic, partially observable environment to be a game of predator / prey, with 4 predators occurring within a labyrinth, which affects movement.

The approach proposed in this thesis is to extend the architecture of reactive Brooks, called the subsumption architecture, with adaptive and predictive capabilities. The agent thus constructed will be able to predict the movement of ghosts over a time horizon in the future, based on a model that is updated with information collected in the past and to use that predictions to decide what to do in next moment.

Key-words: ARX model, Bierman algorithm, pacman, subsumption architecture

Resumo

Os jogos de computador são um domínio de estudo muito importante na área da inteligência computacional. Essa importância advém das propriedades de seus ambientes: multi-agente, competitivos, estocásticos e dinâmicos; onde a verificação de sucesso ou fracasso é de fácil verificação. Para além disso, os jogos e o entretenimento digital em geral, são uma industria em expansão que gera um volume de negócios considerável.

O objetivo deste trabalho é desenvolver um agente para controlar o famoso pacman, capaz de participar numa das mais populares competições, organizadas pela conferência IEEE em inteligência computacional e jogos. Ganha a competição quem conseguir a pontuação média mais elevada de 3 execuções por equipa de fantasmas. O objetivo das equipas de fantasmas é fazer diminuir essas pontuações A dificuldade do pacman deve-se ao facto de fornecer um ambiente estocástico, dinâmico, parcialmente observável, ser um jogo do tipo predador/presa com 4 predadores e ocorrer dentro de um labirinto, o que condiciona os movimentos.

A abordagem proposta neste trabalho é estender a arquitetura reativa de Brooks, a chamada arquitetura de subsunção com capacidades adaptativas e preditivas. O agente assim construído deverá ser capaz de prever o movimento dos fantasmas ao longo de um horizonte temporal no futuro, baseando-se num modelo que é atualizado com informação recolhida no passado e usar essas previsões para decidir o que fazer a seguir.

Palavras-chave: modelo ARX, algoritmo de Bierman, Pacman, Arquitetura de Subsunção.

Agradecimentos

Agradeço aos meus pais pelo apoio financeiro para que possa acabar os meus estudos sem sobrecarga futura. Aos meus amigos e ao meu irmão pelo apoio que deram-me nos momentos de angustias e a maneira espetacular com que lidaram comigo. Ao meu orientador de tese José Valente de Oliveira pela paciência que teve comigo.

Conteúdos

Lista de tabelas

Índice de Figuras

Acrónimos

IEEE Institute of Electrical and Electronic Engineerings

CIG Conferência em Inteligência Computacional e jogos do IEEE

SVM Support Vector Machines

RLS Recursive Least-mean Squares

IDE Integrated Development Environment

ARX AutoRegressive with eXogenous terms

UCB Upper Confidence Bound

UCT Upper Confidence bound applied to Trees

MTD Memory-enhanced Test Driver

Cerrla Cross-Entropy Relational Reinforcement Learning Agent

MSE Mean Squared Error

MedSE Median Squared Error

ES Evolutionary Strategy

SE Squared Error

MDP Markov Decision Process

1 Introdução

A conferencia CIG tem vindo a crescer desde 2005. Antes de ser uma conferencia foi um workshop e agora é uma conferencia internacional. A conferencia é um encontro de especialistas académicos e da industria dos jogos, que visam melhorar a qualidade da inteligência artificial. Ao longo dos anos tem havido um melhoramento significativo nas competições. Isso tem um impacto significativo na qualidades dos vídeo jogos, suas aplicações enquanto inteligência artificial e enquanto capacidades inerentes de cada estratégia; cujos os conteúdos podem ser usados nas mais diversas áreas.

Organizadores referem em [24] que é importante tornar a dificuldade de jogo mais atraente para os jogadores humanos. Foi constatado que os jogadores têm desviado o foco de atenção, procurando adversários online mais fortes com quem possam competir.

A iniciativa gerou várias submissões de conteúdos de competidores. As submissões resultam em apresentações na conferencia e publicações de um conteúdo muito rico. As submissões podem ser encontradas num conjunto de páginas web [25]. É também notória a presença de vídeos no youtube, que mostram exemplos de execuções de jogos com a inter-atividade do próprio computador. Um exemplo de vídeo sobre agentes na competição está presente em [26] (uma execução do ICE Pambush 3, um agente da competição para Ms. Pacman na versão de captura de ecrã) e existem muitos mais no youtube sem ser das próprias competições do CIG IEEE.

A área escolhida é o Ms. Pacman versus equipas de fantasmas, onde o pacman compete contra várias equipas de fantasmas de vários especialistas. A página desta competição encontra-se presente em [27]. Enquanto o objetivo dos fantasmas é impedir com que a Ms pacman obtenha mais pontos, o objetivo do Ms pacman é obter o máximo de pontos.

O estudo do Ms Pacman como em muitos jogos oferece muitas estratégias úteis, que podem ser usadas nas mais diversas situações. Vários exemplos possíveis podem ser áreas de otimização, aprendizagem de máquina, controlo e simulação [31]. Exemplos esses de forma mais especifica, podem ser encontrados no estado de arte: os SVM em aprendizagem máquina, algoritmos de Monte Carlo em simulação, etc.

O leitor irá encontrar muitas estratégias possíveis, em particular o meu controlador preditivo. Esse tipo de previsão e controlo (usados no meu controlador) para além de poder ser usado no Ms Pacman poderia também ser usado em outras situações como os restantes controladores presentes no estado de arte.

1.1 Objetivos de pesquisa

1.1.1 Definição do problema

A pesquisa corrente é explorar técnicas de jogo necessárias à sobrevivência e obtenção máxima de pontos, pela figura miss pacman. O jogo Ms. Pacman é um jogo do género predador/presa, difícil e que foi desde a sua criação até aos dias de hoje muito popular com muitas versões disponíveis na Internet. O ambiente de jogo com que estamos a lidar é estocástico, completo e contínuo.

O jogo contém um labirinto por nível com paredes (não dá para entrar nelas), teleporte em alguns cantos que permite acesso ao lado oposto do labirinto, pílulas, pílulas do poder, fantasmas, comida especial (a comida especial só existe na versão por captura de ecrã) e a nossa figura (Ms pacman) ao qual controlamos. As jogadas possíveis são cima, baixo, esquerda e direita. O objetivo é obter o máximo de pontos. Os fantasmas são nossos inimigos, ao qual devemos evitar caso não estejam comestíveis. Se estão comestíveis andam mais devagar e ao comermos ganhamos pontos extra, mas após um intervalo de tempo deixam de estar comestíveis e voltam a atacar de novo.

As formas de obter pontos é comer comida especial, fantasmas comestíveis, as pílulas e as pílulas do poder ao longo dos níveis. Os pontos da versão captura de ecrã são 10 por pílula, 50 por pílula do poder, 200 para o primeiro fantasma comestível após comida a mesma pílula do poder, 400 para o segundo fantasma comestível após comida a mesma pílula do poder, 800 para o terceiro fantasma comestível após comida a mesma pílula do poder, 1600 para o quarto fantasma comestível após comida a mesma pílula do poder e 100 para a comida de bónus. Os pontos de captura por parte do pacman na versão que compete contra várias equipa de fantasmas e da versão pacman por captura de ecrã são os mesmos, à exceção que não existe comida de bónus na versão que compete contra várias equipas de fantasmas.

Só se consegue avançar para outro nível, caso as pílulas e as pílulas do poder estiverem todas comidas. É suposto que o jogo fique cada mais difícil à medida que existem menos pílulas no nível.

As implementações possíveis tiveram em conta a própria natureza do jogo como o comportamento dos fantasmas com uma componente aleatória (importante para evitar a criação de um plano, tornando o jogo mais difícil), a distância aos fantasmas, métodos de previsão, reações a cada situação criada, procura de entre as situações possíveis, métodos de captura de comida, métodos de fuga, métodos de captura de fantasmas, pior caso possível, teleportes que facilitam o movimento, paredes que limitam os movimentos no labirinto e tudo isso é muito importante.

Por sua vez o melhor método de todos tem de ter em conta todos estes pormenores.

1.1.2 Objetivos

Descrever o estado corrente da pesquisa efetuada pelos elementos ao participarem na competição e criar um projeto próprio também para participar. Pretende-se como objeto de estudo uma aplicação prática de um agente a jogar Ms. Pacman. O objetivo do agente é obter o máximo de pontos, competindo com os melhores do CIG do IEEE, seguindo as regras do jogo. Mas só faz sentido participar na competição se a estratégia for nova e se o preditivo faz aumentar os pontos face ao reativo.

A qualidade da estratégia no contexto deste jogo é medida em pontos. Pretende-se assim maximizar a qualidade da estratégia.

O projeto criado tenta prever as posições futuras, a partir de um modelo linear atualizado com estados anteriores, reagir ao momento presente e aos momentos nos instantes seguintes que têm por base a previsão.

1.1.3 Ferramentas e linguagens

Esta é uma opção muito séria que pode determinar o sucesso ou fracasso do trabalho e que por causa disso deve ser escolhida com imenso cuidado. Pretende-se com ela eficiência computacional, que seja o mais prática possível para o problema em causa, que tenha recursos suficientes e que seja aceite na competição da conferencia CIG. Neste caso aproveita-se os recursos existentes criados pelo autor da competição em [28]. Não é fornecido mais nenhum código extra para além de [28]; senão as próprias bibliotecas standard do Java.

A linguagem da framework de partida é em Java e todo o código também. A linguagem Java mesmo sendo rápida, pode não ser a mais rápida por correr numa máquina virtual, mas dada a situação em que se encontra o problema em causa não é possível utilizar outra. Podia-se eventualmente traduzir o código para C++ com a ajuda de uma ferramenta, mas corria-se o risco de não ser aceite. Dado aquilo que foi fornecido e as características da estratégia não houve necessidade de usar extensões do Java como o Jess, ou ferramentas multi-agentes comuns como o Jade, ou comunicações com ferramentas noutras linguagens; poupando trabalho com a aprendizagem, instalação e tornando tudo mais simples. A própria framework já fornece um ambiente multi-agente, que possibilita todas as características típicas de um agente: independência, pro-atividade, reatividade e habilidade social.

Embora tenha-se usado o Eclipse IDE por estar mais familiarizado, o projeto contém um ficheiro de projeto do IDE IntelliJ IDEA da JetBRAINS que pode facilitar a reutilização do código.

O código da framework para trabalhar pode ser encontrado em [27].

1.2 Estrutura da tese

Este documento é organizado em 6 partes.

Capítulo 1 Introdução: este capítulo pretende criar juntamente com o resumo, uma ideia geral, introdutória e mais simples do trabalho. Começa-se com uma breve descrição da competição e um pouco de qual a importância que toda ela pode ter no quotidiano. A introdução esclarece quais as ferramentas usadas, o seu impacto e os fatores decisivos na sua escolha.

Capítulo 2 Estado de arte: Não faz sentido inovar sobre um tópico sem previamente estar a par do que sobre ele já se conhece, neste caso quais os agentes já criados para pacman da CIG. Cria-se assim um estado de arte do problema alvo. O estado de arte esclarece a situação corrente na literatura disponível, primeiro com um pequeno apanhado de cada um dos agentes com a respetiva classificação obtida, depois com uma descrição detalhada dos mesmos. Os agentes das competições contêm noções sobre o jogo, que devem ser tidos em conta, de forma a melhorar futuros agentes.

Capítulo 3 Modelo de abordagem introdutório: Segue-se vários agentes introdutórios, que são reativos e um reativo com estado. Os agentes abordados usam uma hierarquia de Brooks e são os percursores do agente preditivo. Eles servem para comparar resultados com o preditivo.

Capítulo 4 Agente composto por hierarquia de Brooks com capacidades adaptativas preditivas: Essa melhoria vai ao encontro daquilo que o título desta tese pretende como inovação. Começa com uma introdução que explica termos importantes e dá outra ideia geral, seguindo do que sucede-se em cada secção do capítulo. Inclui todos os detalhes do modelo linear, como funciona o RLS, qual a origem do RLS bem como o algoritmo de Bierman nele usado e exibe qual a arquitetura do agente. Os detalhes do modelo linear explicam como está definido este modelo de uma maneira formal, qual o motivo do seu uso, como e em que ocasiões deve ser usado no contexto deste projeto e qual a importância.

Capítulo 5 Resultados: Contém uma descrição adequada dos parâmetros, métodos utilizados e documentação que permitirá aferir do mérito da abordagem proposta.

Capítulo 6 Conclusão: É mostrada uma conclusão, a avaliação do progresso e o que com ele deve-se ter em conta, incluindo ainda uma análise do problema abordado e qual o melhor caminho a seguir de agora em diante.

Capítulo 7 Trabalhos Futuros: É indicado quais os projetos de controlador alternativos e apontada uma alternativa que não utilizaria o modelo linear, mas que poderia funcionar melhor.

2 Estado de arte

2.1 Descrição geral

O estado de arte é o capítulo da tese que descreve as várias abordagens disponíveis na literatura para o problema em estudo. Vários especialistas criaram agentes para jogar Ms pacman de forma a participarem nas competições existentes em [30] e [27].

A conferencia fornece atualmente 2 formas para jogar Pacman.

A primeira forma em [30] é a mais antiga e a mais exigente das utilizadas na competição para Ms Pacman. Ela requer que os especialistas tenham que usar um sistema de captura de ecrã e mandar sinais de teclado para interagir com a janela de jogo em [29]. As 2 frameworks disponíveis em [30] são apenas exemplos e não existe qualquer restrição nas linguagens usadas nessa competição.

A segunda das frameworks [28] contém uma totalidade de código aberto do próprio jogo. A interação do agente com o resto do jogo é feita com base num objeto e numa função. Um objeto contém dados da interface do estado de jogo e é passado como argumento da função que devolve a próxima direção. A linguagem é obrigatoriamente Java nesta competição. A maior desvantagem é não conhecer à partida como será o comportamento dos fantasmas, mas certamente irão cooperar mais uns com os outros do que na versão por captura por ecrã e têm também uma componente aleatória.

Uma vez que não estamos interessados no processamento de imagem, usamos a framework da competição de equipas de fantasmas. No entanto, a revisão da literatura inclui o trabalho desenvolvido em ambas as formas.

A documentação existente nas páginas [30] e [27] é sumariada como descrição geral dos agentes já criados. Só depois da descrição geral é fornecido os detalhes essenciais (uma descrição mais detalhada desses mesmos agentes). A finalidade desta organização de conteúdos é por um lado que seja facilitada a procura pelo assunto de interesse e por outro que seja possível perceber ao pormenor como foram feitos.

Neste estado de arte o termo “fantasma” refere-se aos nossos inimigos no jogo (Blinky a vermelho, Pinky a rosa, Inky a azul e Sue a laranja) e não é a algo omisso ou que aparece sem explicação aparente no jogo. O termo é utilizado para facilitar a linguagem, não é algo com que possa-se excluir do cenário de jogo e é reconhecido pelos próprios jogadores.

Os nomes dos agentes descritos são os apresentados na página, donde foram obtidas as respetivas pontuações da tabela de descrição breve ou de um nome sugerido pelos autores para o

agente no papel submetido para a competição. O nome sugerido pelos autores é o nome preferível.

As pontuações dos agentes que utilizam a captura de ecrã não são comparáveis aos que competem contra várias equipas de fantasmas. Caso os agentes pacman da versão contra várias equipas de fantasmas passem a ser implementados na versão por captura de ecrã, é muito provável que consigam melhores resultados com essa mudança.

Os agentes estão por ordem crescente segundo a sua pontuação média. Primeiro são apresentados os agentes da versão por captura de ecrã e depois os agentes que competem contra várias equipas de fantasmas.

Nome do Bot	Descrição breve	Pontuações
Shirakawa [17] Versão captura de ecrã	O agente funciona com 3 modos: modo de fuga, modo come e o modo SVM (support vector machines).	[34] **Máximo**: 5520 **Média**: 2493
Wirth [20] Versão captura de ecrã	As decisões são baseadas num mapa de influência, com cada objeto de jogo (fantasmas, fantasmas comestíveis, fantasmas mudando, pílulas, pílulas do poder), distâncias e número de pílulas; tendo uma influência sobre o movimento do pacman.	[34] **Máximo**: 4360 **Média**: 2510
Kwong [10] Versão captura de ecrã	Um agente que escolhe uma ação das 4 possíveis a cada momento do jogo, com base nos pesos de 9 condições. O método de tomada de decisão ao ser chamado soma os pesos das condições que verificam-se. A ação com maior peso é a tomada pelo agente.	[33] **Máximo**: 5250 **Média**: 2641
ICE Pescape [16] Versão captura de ecrã	O controlador é da mesma família que os ICE Pambush. O ICE Pescape usa 5 regras e o algoritmo A*. As regras são as mesmas que as usadas no ICE Pambush, exceto que as duas normas relativas à emboscada na ICE Pambush não estão disponíveis. O ICE Pescape é menos agressivo do que o seu controlador irmão. As vantagens do ICE Pescape são exigir menos tempo de CPU e um desempenho melhor do que o ICE Pambush numa máquina de baixo desempenho.	[34] **Máximo**: 9930 **Média**: 5058
StrathPac [19] Versão captura de ecrã	O StrathPac funciona em 2 modos: caça Fantasmas (1) e progredindo através de nível (2). No modo 1, o StrathPac dá prioridade aos fantasmas comestíveis e foge dos não comestíveis. No modo 2, o StrathPac usa um BFS modificado para encontrar uma pílula segura que seja a mais próxima do pacman.	[32] **Máximo**: 8740 **Média**: 5676
Bruce [11] Versão captura de ecrã	O algoritmo BFS e simulações de Monte Carlo que realizam simulações precisas para modelar o comportamento dos fantasmas.	[31] **Máximo**: 10820

		Média: 5702
Sojoodi [18] Versão captura de ecrã	Um agente reativo com estados que usa o algoritmo Floyd-Warshall para cálculo das distâncias.	[31] **Máximo**: 9990 **Média**: 5933
NTNU CIG 2009 [12] Versão captura de ecrã	Agente com hierarquia de regras que busca um caminho para as pílulas e fantasmas, numa área de tamanho limitado e calcula a distância em unidade de nó.	[32] **Máximo**: 9000 **Média**: 5947
SmartDijkstraPac [3] Versão captura de ecrã	O agente contém um controlador de busca por heurísticas. O conjunto de regras é baseado num conceito de caça de pílulas e em conceitos de evasão de intervalo curto e médio de fantasmas. O objetivo é ir eliminando direções possíveis até haver só uma direção possível. A eliminação é com base na segurança (por exemplo se os fantasmas estão a mais de 4 células na grelha é seguro) e caso não chegue o conceito de segurança (tem mais do que uma direção possível) ele baseia-se na distância à comida.	[34] **Máximo**: 11990 **Média**: 7371
Robles[15] Versão captura de ecrã	O agente faz primeiro uma procura em árvore para encontrar os caminhos possíveis com uma dada profundidade e anexa aos caminhos estatísticas de jogo (pílulas do poder, pílulas, fantasmas a atacar, nós vazios e fantasmas comestíveis). Com base nas informações obtidas através do algoritmo em árvore, seleciona o caminho com a ajuda de um código ad-hoc.	[32] **Máximo**: 15640 **Média**: 7664
Bruce 2 [43] Versão captura de ecrã	O algoritmo é do género Monte-Carlo. O agente introduz conceitos de teste de caminho e relatório.	[35] **Máximo**: 13,700 **Média**: 7955
Kim [9]	O agente evoluiu do algoritmo proposto por Flensbak. A	[31]

Versão captura de ecrã	única diferença face ao Flensbak é esperar perto das pílulas do poder para depois caçar os fantasmas mais facilmente. Esta técnica inovadora é denominada por 'pill-park'.	**Máximo**: 18690 **Média**: 8545
Pac-mAnt ICE [13] Versão captura de ecrã	O comportamento do agente é inspirado no comportamento de formigas recoletoras e formigas exploradoras de uma colónia de formigas. Os parâmetros são definidos com a ajuda de um algoritmo genético.	[31] **Máximo**: 21250 **Média**: 9343
Handa [4] Versão captura de ecrã	Com base em lógica difusa são empregues 4 regras de um agente reativo. A estratégia evolucionária 1+1 incide sobre os parâmetros que definem a função de pertença das variáveis difusas.	[34] **Máximo**: 12930 **Média**: 9376,667
Jave [8] Versão captura de ecrã	Consoante as regras do agente, ele pode utilizar o algoritmo de Dijkstra para distâncias e o algoritmo A* que avalia os custos em cada nó com base num mapa de influencias. O reconhecimento do ecrã inclui deteção eficiente da direção dos fantasmas e soluções para a passagem no teleporte.	[31] **Máximo**: 14660 **Média**: 10812
NTNU [38] Versão captura de ecrã 2011	O NTNU de 2011 é uma evolução do NTNU de 2009. O agente tem parâmetros afinados por um algoritmo genético e é implementado segundo um autómato. O processamento de imagem inclui outros melhoramentos.	[35] **Máximo:** 20,300 **Média:** 11305
ICE Pambush 2 [5] Versão captura de ecrã	O agente utiliza 6 regras com prioridades diferentes e 2 versões do A*.	[33] **Máximo**: 24640 **Média**: 13059
ICE Pambush 4 [7] Versão captura de ecrã	Versão 4 do ICE Pambush que resulta do melhoramento do 3. As inovações consideram a direção dos fantasmas, desconsidera o mapa interno do 3, afinou distâncias em relação ao 3 com algoritmo genético, usa 10 regras com prioridades diferentes, usa 2 versões diferentes do DFS e usa	[31] **Máximo**: 20580 **Média**: 15343

	o algoritmo de Dijkstra.	(O*s autores referem no papel um máximo de 38910 e uma média de 18561*)
ICE Pambush 3 [6] Versão captura de ecrã	A versão 3 do ICE Pambush é um melhoramento do ICE Pambush 2. O agente usa 9 regras com diferentes prioridades, 2 versões do A* e Dijkstra entre 2 nós.	[32] **Máximo**: 30010 **Média:** 17102
ICE Pambush 5 [37] Versão captura de ecrã 2011	ICE Pambush 5, evolução do ICE Pambush 4 é um agente com 13 regras, usa 2 tipos de DFS e um algoritmo de Dijkstra. Os valores do Dijkstra entre os nós já estão pré-calculados.	[35] **Máximo:** 27,240 **Média:** 18065
Nozomu Ikehata & Takeshi Ito [36] Versão captura de ecrã 2011	Algoritmo UCT (do inglês, Upper Confidence bound applied to Trees, que significa limite de confiança superior aplicado a árvores) é um algoritmo Monte-Carlo, com procura em árvore escrito em C++ e que foi inspirado num algoritmo com grande sucesso para o jogo Go.	[35] **Máximo:** 36,280 **Média:** 24341
Cerrla [54][49] Versão versus equipas de fantasmas	Cerrla ou **C**ross-**E**ntropy **R**elational **R**einforcement **A**gent é um agente que utiliza uma aprendizagem por reforço relacional com recurso ao processo de decisão de Markov, uma otimização através de um método de estudo da entropia do jogo, especialização restrita e RLGG (generalização geral mínima relativa ou **R**elational **L**east **G**eneral **G**eneralization).	[51] **Média**: 4277
BruceTong [55] Versão versus equipas de fantasmas	Agente que implementa uma versão alternativa ao minimax com corte alfa-beta. A versão melhorada do minimax chama-se MTD ou do inglês Memory-enhanced Test Driver.	[51] **Média**: 15316
Essexgp	O agente implementa uma árvore de expressão, que foi	[50]

[56] Versão versus equipa de fantasmas	criada usando Programação Genética.	**Média**: 17332
Haimat [57] Versão versus equipas de fantasmas	Um agente reativo que é resultado dos cálculos de um algoritmo genético.	[50] **Média**: 19337
ICE Pambush MCTS [58] Versão versus equipas de fantasmas	Este agente com 5 regras, é uma evolução da versão de captura de ecrã ICE Pambush 4, incluindo uma técnica de procura em árvore com recurso a técnicas de Monte Carlo inspirado no agente ICE gUCT.	[51] **Média**: 20009
PhantomMenance [42] Versão versus equipas de fantasmas	Agente que segue um conjunto de modos baseado num conjunto de critérios, mas nenhuma condição está explicita.	[40] **Média**: 32108
Emgallar [45] Versão versus equipas de fantasmas	O mesmo que a versão de captura de ecrã de 2010 descrito como Pac-mAnt ICE, mas desta vez é contra as várias equipas de fantasmas	[44] **Média**: 34139
James [46] Versão versus equipas de fantasmas	O agente utiliza um sistema baseado em regras de código ad-hoc, o conceito de densidade de fantasmas e uns parâmetros otimizados através de um algoritmo genético.	[44] **Média**: 34278
Spooks [41] Versão versus equipas de	O agente fornece um método de cálculo de bloqueio de fantasmas num ponto, usa um horizonte de eventos para avaliar os nós preferidos e prefere os nós com mais pontos.	[40] **Média**: 41447

fantasmas		

Tabela 1: Descrição geral dos agentes

2.2 Descrição detalhada

2.2.1 Shirakawa

Figura 1: Exemplo de reconhecimento do ecrã [17]

O agente funciona em 3 modos: modo de fuga, modo come e o modo SVM (Support Vector Machines).

O modo de fuga faz com que o pacman fuja dos fantasmas em ataque, caso haja um demasiado perto. A seleção da direção baseia-se em afastar-se o mais possível do inimigo mais próximo.

O modo come faz com que o pacman corra atrás de fantasmas comestíveis que estiverem mais perto. A direção tomada minimiza essa distância.

Noutros casos, o pacman usa o modo SVM. Os SVMs são um grupo de métodos de aprendizagem supervisionada que podem ser aplicados a classificação ou regressão. Os SVMs procuram por um ou vários hiperplanos de separação ótima, onde a margem é máxima. Pretende-se dizer com margem a distância mínima de separação do hiperplano em relação aos pontos de dados mais próximos. Os pontos de dados, que estão na margem são chamados vetores de suporte.

A aprendizagem SVM foi baseada em jogadas de jogadores humanos. O objetivo da rede neuronal treinada é fazer a classificação das jogadas humanas (cima, baixo, esquerda e direita), com base nas posições dos vários elementos de jogo (paredes, pílulas, pílulas do poder, o pacman e os fantasmas).

2.2.2 Wirth

As decisões são baseadas num mapa de influência, com cada objeto de jogo (fantasmas, fantasmas comestíveis, fantasmas mudando de estado, pílulas, pílulas do poder), distâncias e número de pílulas; tendo uma influência sobre o movimento do pacman.

A influência de um objeto diminui, quando a distância ao pacman aumenta. A influência das pílulas individuais aumenta, quando o conjunto de pílulas individuais diminui. Os fantasmas afetam menos, quanto mais pílulas são comidas no nível.

A soma da influencia de todos os objetos é calculada para cada posição, imediatamente adjacente ao pacman. A posição com mais influencia (que é feita a partir da soma) por parte dos objetos é a próxima posição do pacman.

A maior desvantagem reside no facto que o método para ajudar a comer as últimas pílulas degrada a segurança face aos fantasmas.

2.2.3 Kwong

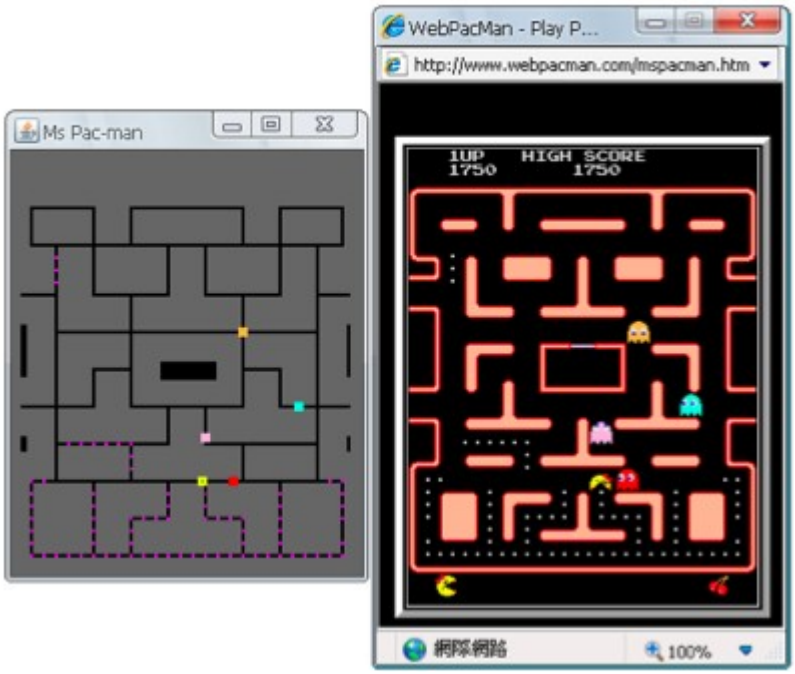

Figura 2: Janela de processamento de imagem e respetiva janela de jogo[10]

Um agente que escolhe uma ação das 4 possíveis a cada momento do jogo, com base nos pesos de 9 condições. O método de tomada de decisão ao ser chamado soma os pesos das condições que verificam-se. A ação com maior peso é a tomada pelo agente.

O ponto de referencia depende da posição do pacman e da direção possível a tomar que está à frente do pacman. A direção é resultante da ação a tomar.

As ações são cima, baixo, esquerda e direita.

As 9 condições são verificadas na tabela seguinte:

Condições	
1	Verifica se há uma pílula qualquer numa área na direcção do ponto de referencia.
2	Verifica se a pílula mais próxima está na direcção do ponto de referencia.
3	Verifica se há um fantasma qualquer numa área na direcção do ponto de referencia.
4	Verifica se está um fantasma qualquer entre o pacman e o ponto de referencia.
5	Verifica se os fantasmas estão agrupados antes de ir tomar a pílula do poder.
6	Verifica se o fantasma azul mais próxima está na direcção.
7	Verifica se a direcção é a mesma que a actual do pacman.
8	Verifica se a direcção é oposta à actual do pacman.
9	Verifica se a direcção do ponto de referencia não está disponível.

Tabela 2: Condições do agente Kwong

2.2.4 ICE Pescape

O agente após a captura de ecrã, tem o resultado da análise de imagem seguinte na janela de desenho:

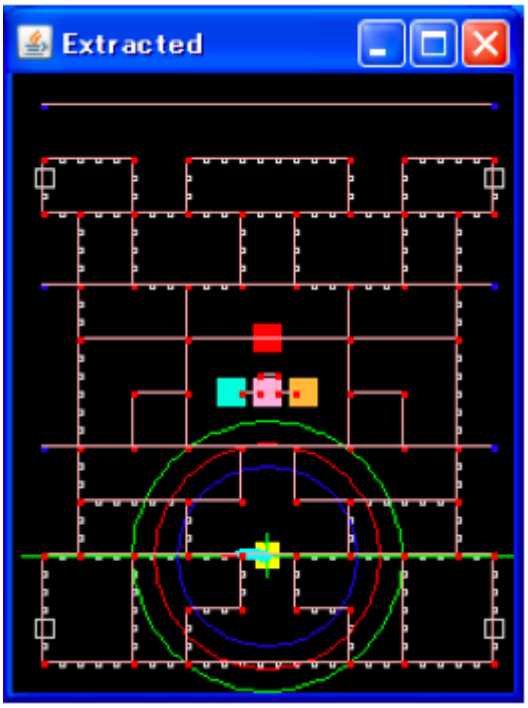

Figura 3: Extração de imagem segundo o a agente ICE Pescape

Os círculos resultam de distâncias ao pacman com a seguinte ordem do menos afastado ao mais afastado: azul, vermelho e verde.

O Controlador para jogar pacman é da mesma família que os ICE Pambush.

O agente usa 5 regras e o algoritmo A*. As regras são as mesmas que as usadas no ICE Pambush, exceto que as duas normas relativas à emboscada no ICE Pambush não estão disponíveis, tornando ICE Pescape menos agressivo do que o seu controlador irmão. No entanto, o ICE Pescape requer menos tempo de CPU, tendo um desempenho melhor do que o ICE Pambush numa máquina de baixo desempenho.

A regra com menos prioridade funciona em caso de o algoritmo A* não poder produzir qualquer caminho; nesse caso usa a mesma regra que no controlador da amostra; a seguir com mais prioridade vem a captura da pílula mais próxima; depois com mais prioridade ainda vêm os fantasmas comestíveis como alvos caso entrem no limite azul; a seguir ainda com mais prioridade vêm como alvo um dos 4 cantos mais afastados dos 4 fantasmas, caso um fantasma atacante entre

no perímetro vermelho; por último (com mais prioridade) caso um dos fantasmas atacantes estejam dentro do perímetro vermelho e pelo menos existe uma pílula do poder no perímetro verde, então o ponto alvo é a pílula do poder mais próxima dentro do perímetro verde (tem de conseguir chegar lá antes dos fantasmas).

A distância usada é a distância de Manhattan.

O custo da avaliação do nó feita no algoritmo A* baseia-se em 3 aspetos:

- 10000*(1000 - [a distância entre a posição do fantasma e o nó]);

- Nos 4 nós vizinhos (superior, inferior, esquerda e direita) do nó avaliado soma 1000 * (1000 - a distância entre o nó mais próximo de um fantasma e o nó vizinho).

– Nós sem pílula têm um custo adicional de 100.

2.2.5 StrathPac

Figura 5: Exemplo de execução[19]

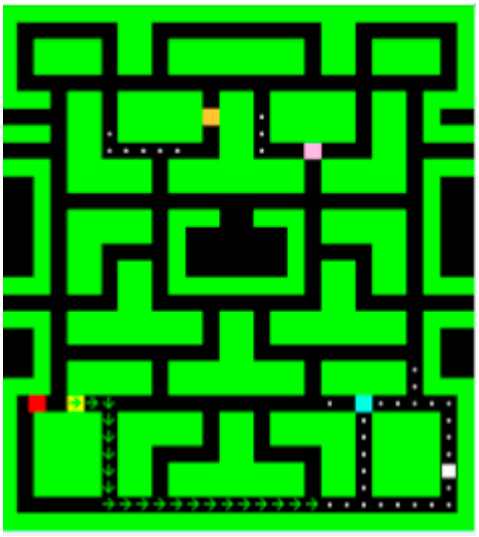

Figura 4: Respetiva janela de processamento de imagem[19]

Os autores realçam a existência de um ponto de equilíbrio entre querer mais pontos o mais breve possível ou ter que escapar melhor aos fantasmas.

O agente não é puramente determinístico, trabalha em 2 modos em função do objetivo corrente e utiliza uma procura em árvore. O objetivo corrente impõe diferentes prioridades, consoante seja caçar fantasmas (modo 1) ou progredir através do nível (modo 2).

O modo 1 é ativado quando existe fantasmas comestíveis. O pacman tenta comer os fantasmas comestíveis, sem ser apanhado pelos não comestíveis. O pacman lucra mais (ou seja ganha mais pontos e é esse o objetivo) com a regra do jogo para correr atrás dos comestíveis, do que preferisse as pílulas, dado o mesmo nível de segurança para ambas as regras. O pacman ao correr atrás dos fantasmas corre riscos, portanto ele deixa a caça quando estão demasiado longe ou a deixar de ser comestíveis.

O modo 2 usa um BFS modificado para encontrar a pílula mais próxima segura. O custo dos nós no BFS é baseado na distância aos fantasmas e alcance às pílulas do poder.

O agente utiliza o 'pill-park', ou seja, espera perto da pílula do poder até ao momento ideal (um fantasma esteja suficientemente perto).

2.2.6 Bruce

Este agente tem por base um modelo semi-probabilístico do comportamento dos fantasmas, criado previamente para todo o jogo. Esse modelo é executado várias vezes, quando o pacman está ameaçado, de forma a escolher o caminho menos provável a seguir pelos fantasmas e que maximize pontos.

Este agente não é o único a basear-se em previsões. O agente BruceTong [48] usa uma versão do algoritmo minimax. O que distingue este agente dos restantes foi a introdução de algoritmos de Monte Carlo para além da utilização de um dos algoritmo em árvore utilizados.

A modelação precisa do comportamento dos fantasmas foi realizada em [36] e está descrita com grande precisão na segunda versão do agente em [43]. O algoritmo de procura em árvore BFS serve para encontrar os elementos de jogo (fantasmas, fantasmas comestíveis, pílulas e pílulas do poder).

A maior inovação no processamento de imagem é localização automática da janela de jogo. A localização automática da janela de jogo é feita através da pesquisa na primeira linha do mapa do jogo.

Figura 6: Deteção da janela de jogo [11]

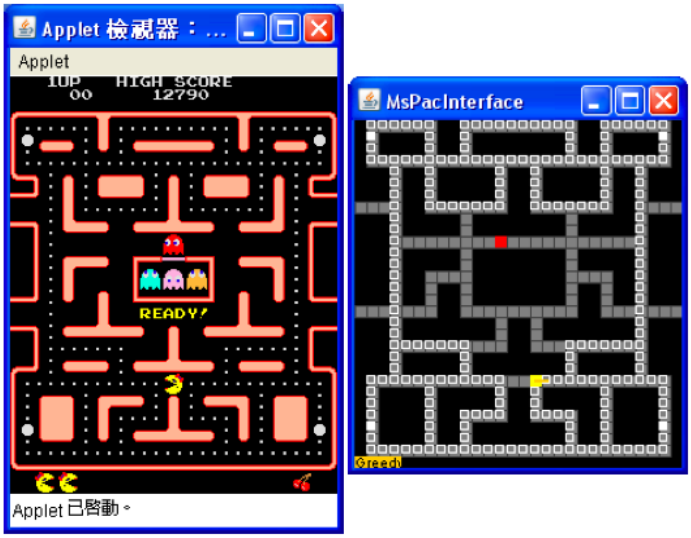

Figura 7: Exemplo de jogo e respetiva janela de reconhecimento [11]

O mapa está pré-processado com caminhos, paredes e localização das pílulas iniciais em ficheiros de texto externos. A distância e os caminhos são calculados em tempo real.

2.2.7 Sojoodi

Os autores dividiram o agente em 2 partes: processamento de imagem e escolha da ação.

A implementação usada para processamento de imagem evoluiu da classe ConnectedSet (uma classe para detetar objetos do jogo em Java e que está separada da classe responsável pela representação interna como em outros agentes, mas que interage com a representação interna). O processador pesquisa o tabuleiro da janela inteira para encontrar cada um dos objetos conhecidos e atualiza um tabuleiro (um array à parte com os resultados) depois de cada captura de ecrã.

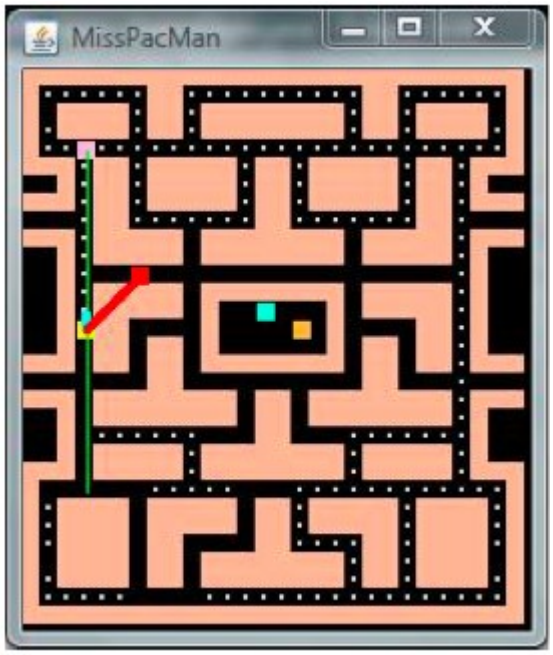

Figura 8: Janela de análise de imagem para o agente Sojoodi [18]

A implementação é um agente reativo com estados que usa o algoritmo Floyd-Warshall.

O algoritmo Floyd-Warshall calcula a distância entre cada par de nós no tabuleiro para saber onde estão os objetos de jogo em relação ao pacman. O algoritmo permite evitar a aproximação de quaisquer dos fantasmas atacantes, que estejam demasiado perto.

Os estados são 3: fuga, caça e segurança.

O estado de caça é quando acabou de comer a pílula do poder e pode perseguir os fantasmas. Nessa situação, ela tenta comer o maior número possível de fantasmas.

O estado de segurança é ativado, quando os fantasmas não estão muito perto e não está em estado de caça, deste modo, tenta comer o máximo de pílulas.

O estado de fuga faz com que o pacman tente escapar aos fantasmas. O estado de fuga é

ativado quando os fantasmas estão demasiado perto.

2.2.8 NTNU CIG 2009

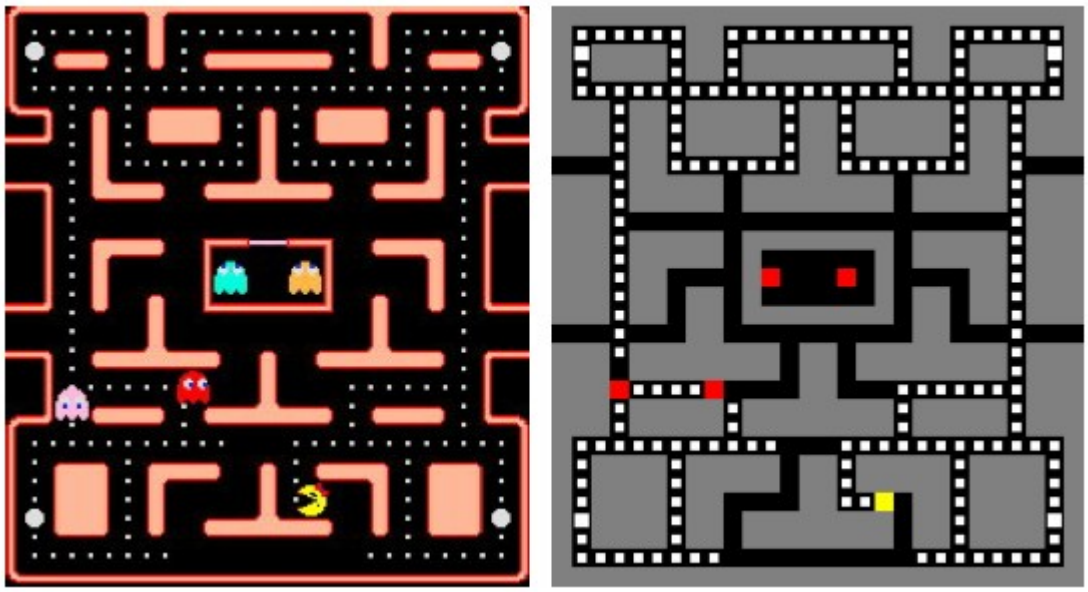

Figura 9: Jogo e respetiva janela de informação do jogo[12]

Agente que busca um caminho para as pílulas e fantasmas, numa área de tamanho limitado e calcula a distância em unidade de nó.

A direção segura é uma direção no varrimento nó a nó numa área limitada, onde não encontra-se um fantasma atacante. Caso exista pelo menos uma direção segura é seguida uma delas. Se esta condição ainda não permite distinguir qual deles é preferível (segundo lugar), escolhe o que dá mais pontuações pela seguinte ordem crescente: pílulas, pílulas do poder e fantasmas comestíveis.

Caso não exista quaisquer direções seguras, o pacman escolhe a direção que deixa o fantasma mais próximo o mais longe possível.

2.2.9 SmartDijkstraPac

O agente SmartDijkstraPac é um controlador de busca por heurísticas, com um conjunto de regras, baseado na noção de caça de pílulas e baseado nas noções de evasão de intervalo curto e médio de fantasmas.

O agente SmartDijkstraPac evoluiu do SmartPac. Ambos verificam a cada instante todas as direções possíveis com base num conjunto de regras, antes de o pacman decidir qual das direções deve ir. Quando chamados, a partir do momento que um dos 2 algoritmos ao escolherem só uma direção a partir das regras avaliadas, nenhuma outra regra é avaliada. O objetivo dos passos é ir eliminando direções possíveis até haver só uma direção possível, com base na segurança (por exemplo se os fantasmas estão a mais de 4 células na grelha, então é seguro).

O SmartPac é um agente reativo com um conjunto de 3 regras, que usa evasão de intervalo curto, que permite uma pontuação razoável.

Se dentro de 4 células discretas da grelha está um fantasma, então evita essa direção (primeira regra).

Se mais do que uma direção é segura, então o caminho que corresponde à distância mais próxima da pílula é escolhido (segunda regra).

Se nenhuma das direções são consideradas seguras (i.e. todas têm um fantasma dentro de uma distância de 4 células-grelha); então a direção, que corresponde à mais afastada dos fantasmas é considerada a menos perigosa e é escolhida (terceira regra).

As melhorias do SmartDijkstraPac face ao SmartPac são evasão do intervalo médio para além da evasão do intervalo curto do SmartPac. A evasão do intervalo médio trabalha com base na previsão de rotas possíveis dos fantasmas, numa distância de n=7 células discretas da grelha e na probabilidade de um fantasma aparecer nessas grelhas ao mesmo tempo que o pacman; usando um algoritmo de Dijkstra modificado para o efeito. O algoritmo de Dijkstra escolhe o caminho onde o fantasma tem menos probabilidades de aparecer ao mesmo tempo que o pacman. A probabilidade de o fantasma aparecer nesse ponto vai diminuindo, conforme a distância ao fantasma aumenta, mas não é referido quaisquer métodos de cálculo para essas probabilidades. Os valores mais viáveis para as probabilidades de presença dos fantasmas teriam por base uma modelação do comportamento dos fantasmas feito por Bruce [11] ou [43].

A regra 1 usa evasão de intervalo curto para eliminar direções perigosas. Se só uma direção é segura, então escolhe essa direção; ou se nenhuma é segura, então escolhe a menos insegura.

A regra 2 usa a evasão a intervalo médio para eliminar direções perigosas. Se só uma direção é segura, então escolhe essa direção. Se nenhuma direção é segura, então escolhe a menos insegura.

A regra 3 avalia a ocorrência de fantasmas comestíveis. Se qualquer fantasma comestível está relativamente perto, então escolhe uma direção segura para o fantasma comestível mais próximo.

A regra 4 escolhe a que está mais perto das pílulas mais próximas.

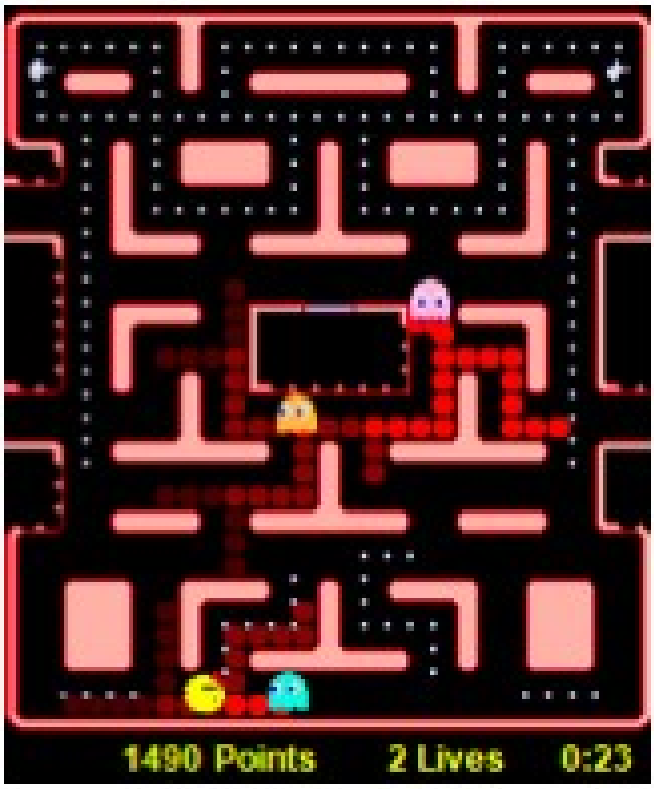

Figura 10: Esta janela mostra um cenário de jogo possível com as previsões de rotas dos fantasmas. Os pontos mais avermelhados têm maior probabilidade de terem um fantasma neles nos próximos instantes de jogo que se sucedem.

2.2.10 Robles

O agente é baseado nas vantagens dos agentes de código ad-hoc e em técnicas de inteligência computacional.

A técnica de inteligência computacional é usada primeiro com uma busca em árvore para encontrar os caminhos possíveis com profundidade 40 e anexa aos caminhos (4 a 12, dependendo da posição do pacman e do labirinto em causa) estatísticas de jogo (pílulas do poder, pílulas, fantasmas a atacar, nós vazios, fantasmas comestíveis). Cada nó da árvore só tem um tipo de conteúdo, excluindo a hipótese de ter fantasma e pílula ao mesmo tempo. Caso haja fantasma e pílula ao mesmo tempo, o fantasma terá prioridade sobre a pílula.

Uma vez tendo todos os caminhos gerados é feita a seleção do caminho. A seleção do caminho é feita com código ad-hoc, tendo como prioridade a segurança do pacman. A primeira tarefa a fazer no código ad-hoc é determinar se os caminhos são completamente seguros, suficientemente seguros ou completamente inseguros e só depois é determinado o que dá mais pontos.

Um caminho completamente inseguro é um caminho que contém pelo menos um fantasma. Um caminho completamente seguro é um caminho que não tem quaisquer fantasmas e para além disso só existe caso a árvore não tenha fantasmas.

Existe ainda a possibilidade de o caminho não ser considerado nem completamente seguro e nem completamente inseguro (o caminho não contém fantasmas, mas há fantasmas na árvore). A solução é procurar pelos caminhos com nós seguros. Um nó seguro nesse contexto é um nó do caminho, onde a distância entre nó e o pacman é menor do que a distância do fantasma mais perto a esse nó referido.

Se há vários caminhos completamente seguros ou suficientemente seguros; então ele escolhe o com mais pontuação (o caminho com pílula do poder, fantasmas comestíveis ou mais pílulas).

Caso não tenha escolhido nenhum caminho com base nos critérios anteriores, escolhe o mais favorável (o que tem maior distância em relação aos fantasmas).

2.2.11 Bruce 2

Os autores descrevem o funcionamento do agente depois existirem η pílulas no labirinto. Esse estado de jogo é denominado endGame. Quanto menos pílulas disponíveis, mais difícil é passar de nível. Essa dificuldade implica um método mais exigente.

O método proposto consiste em 2 componentes principais: geração de caminhos e teste dos caminhos.

A geração de caminhos consiste em gerar um certo número de caminhos a partir de uma procura DFS acíclica, o mais curto possível, que ligam 2 células, com o objetivo de fazer o pacman atingir as pílulas restantes. O início do caminho contém a localização corrente do pacman e a final uma das pílulas alvo. A estrutura do labirinto permite vários caminhos possíveis para 2 células, mas só um deles é considerado. Os caminhos antes de serem testados são ordenados de forma ascendente pela distância do pacman à pílula. Esta ordenação permite eliminar os caminhos mais longos encontrados, que estão no fim da estrutura de dados.

O teste dos caminhos, após eles serem gerados, começa pelos mais curtos e tem por base um algoritmo de Monte Carlo, que indicará se o caminho é seguro. As simulações $n_{simulações}$ de Monte Carlo fazem avançar alternadamente o pacman (que segue os caminhos gerados) e os fantasmas pelo labirinto (sendo que os fantasmas são baseados num conjunto de regras como se fosse o jogo) num tempo discreto. As simulações irão devolver uma taxa de sobrevivência (taxa que indica o número de vezes em que o pacman não foi apanhado) para cada caminho gerado.

Existe uma regra adicional com mais prioridade, que evita um fantasma demasiado perto ou uma situação em que todos os caminhos têm uma taxa de sobrevivência inferior a um corte, fazendo o pacman fugir para um ponto adjacente mais seguro (mais longe do que o fantasma mais próximo).

As regras de movimentos dos fantasmas têm por base um trabalho realizado em [39]. O resultado desse trabalho é apresentado de seguida.

O Blinky move-se com probabilidade 0.9, em direção à posição corrente do pacman através de um algoritmo de caminho mais curto.

O Pinky move-se com probabilidade 0.8, em direção à primeira interceção de caminhos a ser alcançada pelo pacman; mas caso a distância do pacman ao Pinky seja menor ou igual a 10 move-se com probabilidade 0.8 em direção à posição corrente do pacman.

O Inky move-se com probabilidade 0.7, em direção à interceção passada pelo pacman através do algoritmo de caminho mais curto; mas se a distância do pacman ao Inky é menor ou igual a 10, o Inky move-se com probabilidade 0.7, em direção à posição corrente do pacman.

O Sue move-se com probabilidade 0.5, em direção à posição corrente do pacman através do

algoritmo de caminho mais curto.

Os outros casos da captura de ecrã, dado que foi descrito uma probabilidade para cada um dos movimentos dos fantasmas, os outros movimentos dos fantasmas destinam-se à seleção de uma direção aleatória uniformemente distribuída.

A equipa Legacy funciona de maneira diferente.

O fantasma Blinky aproxima-se do pacman a partir da distância do caminho mais curto.

O fantama Pinky aproxima-se do pacman através da distância Euclideana.

O fantasma Sue aproxima-se do pacman a partir da distância de Manhattan.

O fantasma Inky vai atrás de um nó aleatório.

Todos os fantasmas das equipas de fantasmas que competem entre si seguem uma direção aleatória; quando o nó calculado é o mesmo que o atual; ou quando o nó calculado é o mesmo que o anterior. Em 30,85% dos casos os fantasmas invertem o movimento que tinham até ao momento. O valor 30,85% foi obtido através do calculo de um valor aleatório, distribuído de forma uniforme e compreendido entre 0 e 1; que depois de gerado determina se o fantasma inverte a marcha, caso o valor aleatório seja inferior a 0.005.

O algoritmo requer um afinamento dos parâmetros conforme sugere a tabela seguinte.

Parâmetros	Valor
η	35
$n_{simulações}$	100
β	0.95
α	0.60
d_{min}	2
t_{max}	30 ms

Tabela 3: Vários parâmetros afinados para este algoritmo de MonteCarlo

O η define o número de pílulas máximo para que este módulo seja executado.

O $n_{simulações}$ é o número de simulações a realizar.

O β é um corte que indica se a taxa de sobrevivência mínima de um caminho em particular é ou não considerado um caminho seguro.

O α é um corte que indica se a maior taxa de sobrevivência mínima é suficiente para determinar se há ou não um caminho melhor que seja seguro.

O d_{min} é um corte que indica a distância de segurança em relação aos fantasmas.

O t_{max} é o tempo máximo que o algoritmo tem para devolver uma ação ao jogo.

2.2.12 Kim

A implementação evoluiu do algoritmo incluído neste estado de arte, de outra competição. O agente de onde evoluiu foi SmartDijkstraPac [3], proposto por Flensbak e Geogios N.Yannakakis.

A única diferença do seu antecessor é uma técnica denominada 'pill-park'. O 'pill-park' consiste em esperar perto da pílula do poder até que os fantasmas atacantes estejam suficientemente perto e assim que estejam come a pílula do poder. O 'pill-park' é usado quando o pacman está próximo da pílula do poder.

Esta implementação levou a um melhoramento de 10% na pontuação média face ao seu antecessor. Os melhoramentos devem-se a uma facilidade acrescida na captura dos fantasmas comestíveis (menor distância em relação ao mesmo intervalo de tempo em que estão no estado comestível), por que o comportamento dos fantasmas assim o permite.

Ao usar-se a framework [28] tem-se de ter em conta que o pacman vai enfrentar qualquer equipa de fantasmas, pelo que não podemos usar a técnica 'pill-park'. Nesse caso, ao usar o 'pill-park', o pacman pode ficar eternamente à espera dos fantasmas, enquanto os fantasmas estão numa zona segura.

2.2.13 Pac-mAnt ICE

Este pacman foi inspirado no comportamento de uma colónia de formigas. Os tipos de formigas que existem são 2: um tipo para encontrar caminhos que fazem ganhar pontos (formigas recoletoras) e outras de outro tipo para encontrar caminhos seguros (formigas exploradoras). As formigas vão deixando feromonas por onde passam. Cada formiga tem uma distância máxima a explorar, devido a limitações no tempo de computação. Se a formiga ultrapassar essa distância morre.

O algoritmo usa várias iterações. Em cada iteração ambos os tipos de formigas são lançados para posições adjacentes em relação à posição do pacman. O pacman segue os caminhos com mais feromonas que são ou das recoletoras, ou das exploradoras. O tipo de formigas a seguir é definido por um parâmetro. Esse parâmetro é uma distância mínima a um dos fantasmas. Quando a distância ao fantasma é menor do que a distância mínima, então segue a melhor exploradora, senão segue a melhor recoletora. Uma vez escolhido o tipo de formiga segue o lugar com mais feromonas desse tipo.

O próximo nó a visitar pela formiga é semi-aleatório, gerado com a ajuda de uma função (regra de transição de estado) que depende do nível de feromona e de informações armazenadas pela formiga (informação do nó e nós visitados):

$$s^x = \begin{cases} \underset{j \in N_k(i)}{arg\ max} \{[\tau^x(i,j)] * [\epsilon^x(i,j)]^\beta, \ se\ q > q_0^x\} \\ S, \ se\ q \leq q_0^x \end{cases} \quad (1)$$

$$S = \begin{cases} \frac{([\tau^x(i,j)] * [\epsilon^x(i,j)]^\beta)}{(\sum [\tau^x(i,u)] * [\epsilon^x(i,u)]^\beta)}, \ com\ u \in N_k(i) \\ 0, \ se\ j \notin N_k(i) \end{cases}$$

As variáveis dispostas na fórmula 'q' é gerada aleatoriamente para obter um valor $q \in [0,1]$; 'x' é o tipo de formiga; τ^x é o nível de feromona da formiga de tipo x; $N_k(i)$ é um conjunto de nós, que são alcançáveis pela formiga k ('j' e 'u' têm de fazer parte dessa vizinhança); $\epsilon^x(i,j)$ é o desejo relativo ao nó (i,j), que pode ser o inverso da distância ao que se come, ou a

distância aos fantasmas atacantes; e q_0^x corresponde aos parâmetros q_0^c e q_0^a (de cada tipo de formiga, sendo formiga recoletora e formiga exploradora respetivamente), compreendidos entre 0 e 1 excluídos e determina a importância relativa do tipo de exploração. Esta formula veio da versão do algoritmo das formigas para o caixeiro viajante [21] e despreza os nós que não são da vizinhança da formiga.

A fórmula

$$\tau^x(i,j) <- -(1-\rho_0^x)*\tau^x(i,j)+\rho_0^x*\tau_0^x \quad (3)$$

regula a atualização local da feromona realizada pela formiga, onde τ_0^x é o nível mínimo de feromona e ρ_0^x é a taxa de dissipação das feromonas. A fórmula

$$\tau(i,j) <- -(1-\alpha_0^x)*\tau^x(i,j)+\alpha_0^x*\Delta\tau^x(i,j) \quad (4)$$

regula a atualização global, onde α_0^x é a taxa de deposição da feromona.

$$\Delta\tau(i,j)=\begin{cases} Q_{ib}^x, \ se(i,j)\in S_k \ é \ a \ melhor \ iteração \ da \ formiga \\ 0, \ caso \ contrário \end{cases} \quad (5),$$

onde Q_k^x é uma heurística da formiga k, do tipo x.

As formigas recoletoras definem o Q_k^x por

$$Q_k^c=\frac{\sum(\eta_{i,j}^c+\sigma*\tau^e(i,j))}{(d(P,n_{i,j}))}, (i,j)\in S_k \quad (6)$$

; onde d(P, $n_{i,j}$) é a distância do pacman ao nó $n_{i,j}$, σ é um parâmetro do sistema, $\tau^e(i,j)$ é o nível de feromona da formiga exploradora no ponto (i,j) e $\eta_{i,j}^c$ é definido de seguida. A variável $pílula_{act}$ é o número corrente de pílulas. A variável $pílula_{tot}$ é o número total de pílulas no arranque do nível. A variável χ é um parâmetro do sistema.

$$\eta_{i,j}^c=\begin{cases} \frac{1}{(pílula_{act}/pílula_{tot})}, \ se \ nó_{i,j}=pílula \\ \frac{\chi}{(pílula_{act}/pílula_{tot})}, \ se \ nó_{i,j}=fantasmas \ comestíveis \end{cases} \quad (7)$$

As formigas exploradoras definem o Q_k^x por

$$Q_k^e=\sum\eta_{i,j}^e+\omega*\tau^c(i,j), (i,j)\in S_k$$

, onde ω é um parâmetro do sistema, $\tau^c(i,j)$ é a feromona da recoletora na posição (i,j) e $\eta_{i,j}^e$ é definido como:

$$\eta_{i,j}^e=d(G,n_{i,j})/d(P,n_{i,j})^2 \ ,$$

que corresponde à divisão da distância do fantasma mais perto ao nó pelo quadrado da distância do

pacman ao nó.

Todos os parâmetros ainda por definir foram calculados por um algoritmo genético com função de avaliação correspondente à média de pontuações máximas, com população 50, com seleção baseada em torneio de sobreviventes igual a 3 e o crossover era multi-ponto e posicional.

Parâmetro	intervalo	Precisão
Número máximo de formigas	[5-30]	1
distância de segurança em relação aos fantasmas atacantes	[5-20]	1
χ	[5-50]	1
p_0^c	[0-1]	0.01
p_0^e	[0-1]	0.01
α_0^c	[0-1]	0.01
α_0^e	[0-1]	0.01
σ	[0-1]	0.01
ω	[0-1]	0.01
q_0^c	[0-1]	0.01
q_0^e	[0-1]	0.01

Tabela 4: Parâmetros a definir pelo algoritmo genético no agente Pac-mAnt ICE

2.2.14 Handa

O agente emprega a lógica difusa e uma estratégia evolucionária 1+1 com regra de sucesso 1/5 aos seus parâmetros de regras difusas. Variáveis difusas possíveis são perto, longe, rodeado, ameaçado, protegido, confiança no caminho ou dominado pelos fantasmas... Neste agente foram utilizadas como variáveis difusas a noção de perto.

As informações que os autores consideram mais relevantes a partir do processamento de imagem são distância a cada fantasma, distância a cada fantasma comestível, posição da pílula não comida mais próxima e distância à interceção fantasma/pacman mais próxima. O cálculo da distância tem em conta a estrutura do labirinto, mas não é indicado qual o método para calcular a distância. O melhor método de cálculo das distâncias é com valores pré-processados a partir do algoritmo de Dijkstra.

As regras difusas são 3. A primeira regra difusa elimina todas as direções, que vão ao encontro de cada fantasma, que esteja perto do pacman (serve para evitar os fantasmas). A segunda regra difusa vai na direção do fantasma comestível mais perto do pacman de entre os fantasmas comestíveis que estejam perto do pacman. A terceira regra difusa permite ao pacman aproximar-se da interceção do caminho do pacman com o caminho do fantasma, caso consiga chegar a essa interceção antes do fantasma e esteja rodeado por mais do que um fantasma. A terceira regra difusa permite evitar um tipo de emboscadas, que tenham uma fuga possível.

Além das 3 regras difusas há uma outra regra que não é difusa e que serve para apanhar pílulas. A regra para apanhar pílulas atua por defeito, quando nenhuma regra difusa é ativada.

O ES (1+1) com regra de sucesso um quinto implementa a ideia de que o tamanho do passo aumenta se muitos passos têm sucesso, indicando que a procura é local e diminuiria se poucos passos têm sucesso, indicando que o comprimento do passo usado para amostragem das soluções é largo.

Existe pelo menos 2 variantes do algoritmo original [53].

1: *Inicializa* X_0, σ_0 que são amostra e passo iniciais respetivamente
2: *repete*
3: $\tilde{X}_n = X_n + \sigma_n N(0, I)$ uma amostra de offspring
4: *se* $f(\tilde{X}_n) \leq f(X_n)$ *então* Se f o offspring é estritamente melhor
5: $X_{n+1} = \tilde{X}_n$ novo pai=offspring
6: $\sigma_{n+1} = 1.5\sigma_n$ tamanho do passo é aumentado
7: *senão* Se o offspring é estritamente pior
8: $X_{n+1} = X_n$ novo pai=pai antigo
9: $\sigma_{n+1} = 1.5^{(-1/4)}\sigma_n$ tamanho do passo é decrementado
10: *até critério de paragem ser reconhecido*

Figura 11: Exemplo da ES 1+1 com regra de sucesso um quinto em pseudo-código [53]

A função de fitness do algoritmo genético é a média dos passos de jogo (essa média é calculada ao jogar 5 vezes, de maneira que os parâmetros foram inicialmente código ad-hoc) para afinar 6 parâmetros.

Os 6 parâmetros são $x_1^a,\ x_2^a,\ x_1^c,\ x_2^c,\ x_1^{(gt)},\ x_2^{(gt)}$; onde cada par k (x_1^k, x_2^k) corresponde às abcissas de um intervalo de uma função de pertença em crescimento linear ou decrescimento linear: x_1^a, x_2^a referem-se a evitar fantasmas, x_1^c, x_2^c referem-se a caçar fantasmas e $x_1^{(gt)}, x_2^{(gt)}$ referem-se a passar na interceção.

2.2.15 Jave

Os vários passos deste agente pacman são executados pela ordem seguinte: deteção do estado de jogo a partir da captura de ecrã, atualização de um tabuleiro interno (mapa de influencias) e determinação da melhor regra a aplicar.

Duas inovações foram apresentadas em relação ao processamento de imagem: a direção dos fantasmas e a resolução de um problema no tele-porte.

O tele-porte é uma zona situada nos limites x ou y da janela de jogo, que permite a passagem de uma personagem do jogo para o lado oposto. Se por exemplo, a posição do fantasma ou pacman é o máximo no eixo dos xxs da janela de jogo, a personagem de jogo pode passar a zero no eixo dos xxs e ao fazer isso, essa personagem acabou por usar o tele-porte. A zona que permite fazer o tele-porte não tem paredes a impedir o acesso a essa posição. O tele-porte é bi-direcional e em termos visuais não é imediato, sendo possível visualizar uma parte da personagem num canto e a outra parte no lado oposto.

O fantasma é detetado no tele-porte, memorizando as posições dos fantasmas por um curto período de tempo. Quando um fantasma entra no tele-porte e desaparece a partir do mapa, ao sair do outro lado do tele-porte terá a sua posição atualizada novamente.

A direção dos fantasmas é feita pelo exame da posição dos seus olhos.

Figura 12: Os olhos dos fantasmas indicam a sua direção atual [8]

O reconhecimento do estado de jogo inclui: a localização dos objetos em jogo em relação ao mapa, deteta as direções dos fantasmas e cria uma grelha baseada em grafo para representar o estado do jogo.

Qual o motivo de usar 2 algoritmos como A* e Dijkstra, em vez de usar só um deles?

Os algoritmos de procura usados partem de estudos anteriores com grande sucesso como o

algoritmo A* nos ICE Pambush e o mapa de influencias como no Wirth [20]. A vantagem do A* com a posição do pacman na raiz da árvore é o cálculo do custo de um caminho especifico, que é determinado pelo custo de cada objeto ao longo do caminho. O pacman com o tipo de cálculo do A* prefere caminhos com menos fantasmas e mais pílulas, através do cálculo do custo mínimo. A limitação do A* é o peso computacional, daí uma limitação de profundidade na árvore de procura. O algoritmo de Dijkstra só dá a distância mínima entre 2 nós, mas é mais rápido.

O mapa de influencias (utilizado para cálculo do custo do A* nesta implementação) consiste numa descrição do custo do pacman ao ir para cada nó, sendo que cada nó tem uma posição no labirinto. O custo de cada posição é decidido pelo número de fantasmas, direção dos fantasmas, distância aos fantasmas e a distância à comida ou fantasma comestível a partir dessa posição. Se o fantasma está muito perto (por exemplo menos de 2 de distância), então essa posição tem custo muito alto. O custo do fantasma é ignorado, se não está muito perto e se não for na direção do pacman, ou se houver cruzamentos seguros em que o pacman pode chegar em primeiro lugar. Os custos são valorizados com os fantasmas com valor positivo alto e pílulas com valor negativo.

O comportamento do pacman baseia-se em 6 tipos de ações a tomar, que permitem-lhe adaptar-se a diversas situações: espera pelo fantasma (1), dirige-se à pílula do poder depois da espera (2), dirige-se à pílula do poder sem nenhuma espera (3), dirige-se ao fantasma comestível mais próximo para o capturar (4), vai em direção às pílulas mais próximas (5) ou evita os fantasmas (6).

Que tipo de situações e algoritmos a executar consoante esse tipo de situações devem-se coordenar de forma a cumprir cada um desses tipos de atitude de forma correta? Quando usa-se o Dijkstra e quando usa-se o A*?

Se a distância do pacman à pílula do poder mais próxima é menor ou igual a 5, o fantasma mais próximo do pacman está a uma distância superior a 2 e a distância do fantasma à pílula do poder é superior a 1; então segue ação (1).

Se no mínimo uma pílula do poder existe, não há fantasmas comestíveis, o pacman já estava à espera do fantasma (acabou de executar a ação (1)), mas desta vez a distância do pacman ao fantasma é ainda menor e a distância do fantasma à pílula do poder é ainda menor; então move-se para a pílula do poder mais próxima, usando o algoritmo de Dijkstra (2).

Se existe pelo menos uma pílula do poder, nenhum fantasma comestível, distância ao fantasma mais próximo menor ou igual a 8, distância à pílula do poder menor ou igual a 6 e a distância ao fantasma mais próximo é maior ou igual à distância da pílula do poder mais próxima; então move-se para a pílula do poder mais próxima usando o algoritmo de Dijkstra (3).

Se no mínimo um fantasma comestível existe, distância ao fantasma atacante mais próximo

superior ou igual a 9, e distância ao fantasma comestível mais próximo inferior ou igual a 13; então segue a ação (4), usando o algoritmo de Dijkstra.

Se distância ao fantasma atacante mais próximo superior ou igual a 10; então segue a ação (5), usando o algoritmo de Dijkstra.

Se nenhuma das condições acima se verificam; então segue a ação (6), usando o algoritmo A*.

A implementação só com estas regras faz com que o pacman ao restarem poucas pílulas ande constantemente a fugir dos fantasmas e a evitar assim as pílulas. Há portanto uma regra adicional para solucionar o problema.

Quando a contagem para todo o mapa é menor que uma quantidade predeterminada verifica se o valor do benefício de um percurso é o mesmo que a contagem de pílulas restantes; se este for o caso, então sabe-se que uma determinada rota contém as pílulas restantes. Quando isso acontece, adiciona-se um benefício muito grande para esse caminho a fim de obrigar o agente, tanto quanto possível, a ir ao fim e completar o nível. Nestas circunstâncias podemos ignorar os fantasmas o que poderia levar o pacman a ser comido.

2.2.16 NTNU CIG 2011

O agente é implementado segundo um autómato e tem parâmetros afinados por um algoritmo genético.

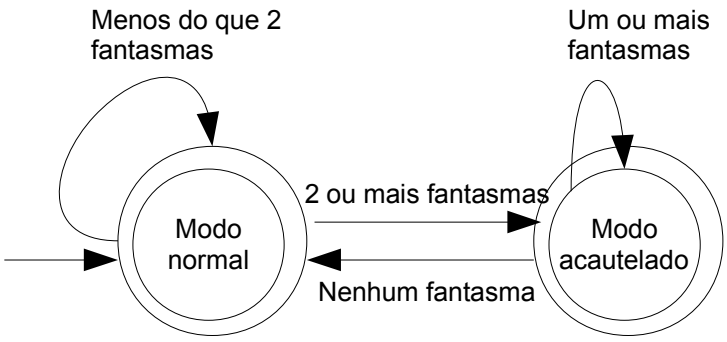

Figura 13: Autómato que representa os 2 estados do agente [38]

As setas representam transições. Tanto o modo normal como o modo acautelado podem ser estados finais do autómato. O agente começa no estado normal. Quando existem 2 ou mais fantasmas atacantes nas proximidades e o agente está no modo normal, ele muda para o modo acautelado. Quando o agente está no modo acautelado e não tem fantasmas atacantes nas proximidades, ele muda para o estado normal. Noutros casos ele mantém o estado.

O modo normal segue as regras da versão anterior do agente em 2009 [12].

A procura no modo acautelado é feita até uma distância fixa, em vez de ser até encontrar um objeto como no modo normal. Cada tipo de objeto tem uma pontuação e as pontuações são acumuladas nos caminho durante a procura no modo acautelado. As bifurcações na árvore de procura (que acontecem quando há cruzamentos de caminhos) geram um conjunto de 'n' sub-caminhos, cuja a pontuação deles têm um impacto 'n' vezes inferior ao serem somadas ao caminho principal. Este último cálculo é feito recursivamente com as bifurcações a diminuirem o impacto dos pontos dos sub-caminhos seguintes. Os cálculos evoluem assim sucessivamente até à profundidade máxima. O caminho com mais pontuação é o escolhido no modo acautelado.

O modo acautelado tem 6 parâmetros definidos por um algoritmo genético: a profundidade máxima da procura (1), a pontuação de uma pílula do poder (2), a pontuação de um fantasma atacante encontrado na sua frente (3), a pontuação de um fantasma atacante encontrado por detrás

(4), a pontuação de um fantasma comestível encontrado na sua frente (5) e a pontuação de um fantasma comestível encontrado por detrás (6).

A população inicial é composta por 6 indivíduos aleatórios. A avaliação é com base na terceira maior pontuação ao longo de 10 execuções. O acasalamento é com base em 2 pares de indivíduos de forma aleatória. Os pais são transformados temporariamente num formato binário e geram o offspring por um crossover de 2-pontos, criando 4 offsprings. Além dos 4 offsprings um outro, mas aleatório é criado, da mesma forma que na inicialização dos indivíduos. Esses 5 offsprings são avaliados e associa-se uma aptidão. A seleção é com base nos 6 indivíduos com mais aptidão. A população inicial é avaliada antes de avançar para o resto do algoritmo genético.

Os melhoramentos ao processamento de imagem são 6.

A deteção dos fantasmas só é feita na vizinhança da posição anterior. Quando um fantasma comestível está em cima do pacman é possível saber o número de fantasmas comestíveis pelo volume. Quando o pacman é capturado, o pacman não pode entrar nas zonas perigosas pré-definidas. Não há envio de mensagem de teclado até haver uma direção nova escolhida pelo agente. Existe uma extensão do mapa para conseguir visualizar a passagem no teleporte. Existe uma monitorização para detetar um novo estágio através da análise das palavras “High score” na janela de jogo.

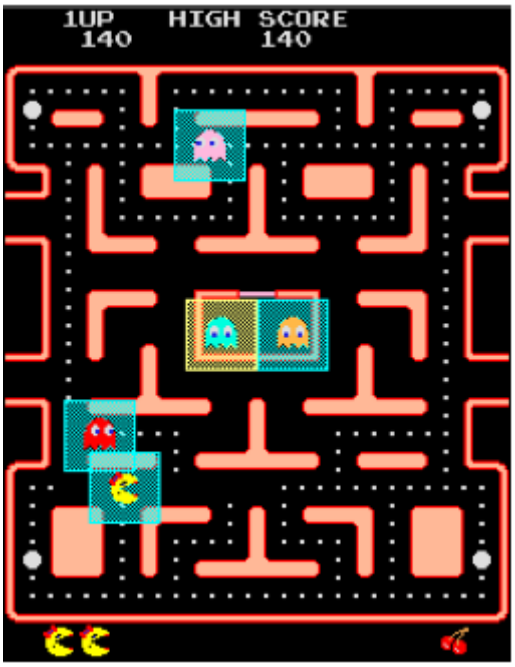

Figura 14: Vizinhança dos fantasmas e do pacman a azul transparente [38]

Figura 15: Zonas perigosas pré-definidas a azul transparente [38]

2.2.17 ICE Pambush 2

2.2.17.1Análise da imagem

A janela de reconhecimento de jogo é muito detalhada. Cada fantasma tem uma cor, assim como as pílulas, os cantos, as paredes, o vácuo e o pacman. O quadrado amarelo é o pacman, o quadrado rosa é o Pinky, o quadrado laranja a Sue, o quadrado azul o Inky e um dos quadrados vermelhos o Blinky, tendo cada fantasma uma cor. Os fantasmas que estão dentro da grelha vermelha central não aparecem.

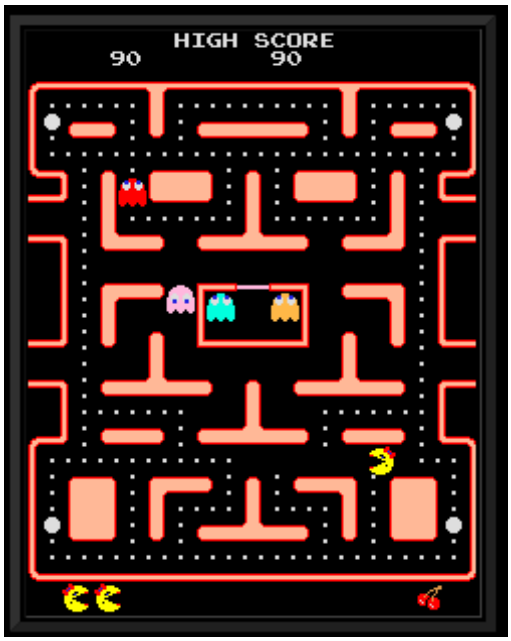

Figura 16: Janela de jogo [5]

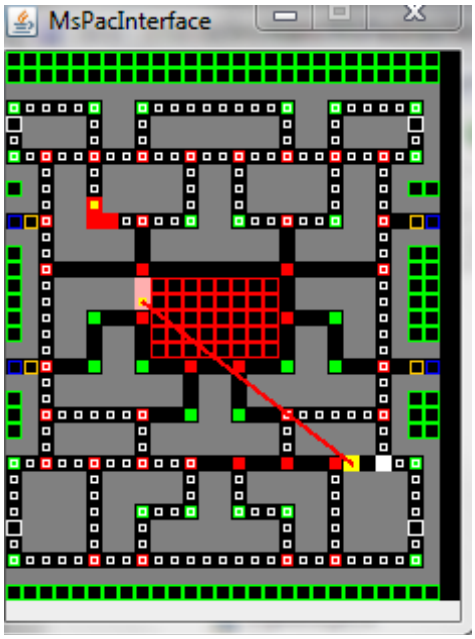

Figura 17: Respetivo processamento de imagem[5]

Desse reconhecimento constrói-se um mapa com o mapa original completo mais as metades esquerda e direita, formando um resultado com a seguinte concatenação.

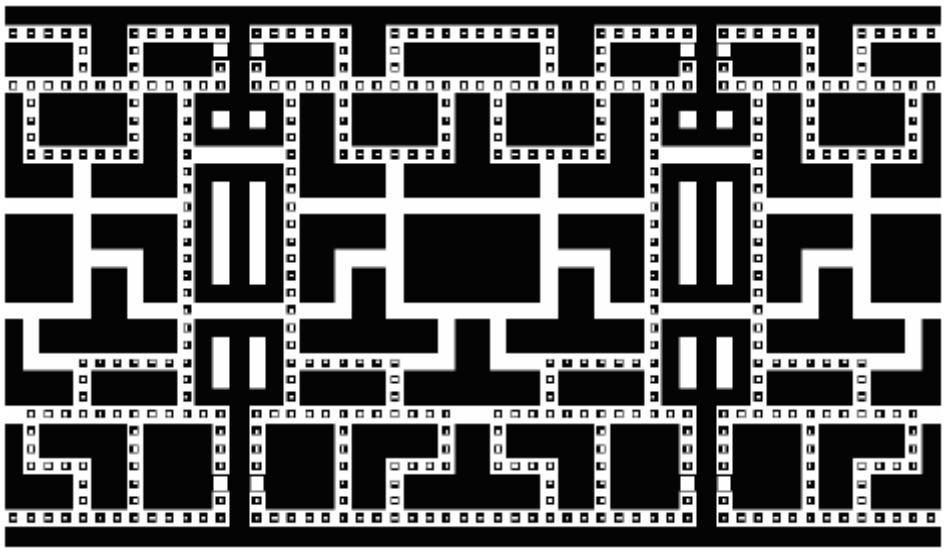

Figura 18: Expansão do mundo para simplificação da tomada de decisão [6]

Esta expansão é vantajosa por haver tele-porte, permitindo que um objeto no mapa tenha 2 posições (a mais curta indo pelo tele-porte e a mais curta sem passar pelo tele-porte) que permitem sempre o cálculo da distância mais curta de uma forma geral, facilitando o mecanismo de cálculo de distâncias entre objetos. As desvantagens são ocupar mais espaço e de aumentar o tempo de processamento com a cópia de cada uma das metades.

2.2.17.2 Tomada de decisões

O agente ICE Pambush 2 evoluiu do ICE Pescape e ICE Pambush do WCCI 2008, usa 6 regras com diferentes prioridades e 2 versões do A*. O objetivo do A* é encontrar um caminho até ao alvo que minimize o custo e a distância de Manhattan.

As regras etiquetadas com 3 e 4 usam a versão 1 do A*, enquanto as 5 e 6 usam a versão 2 do A*. Os custos associados às heurísticas das localizações no ponto X são dados segundo vários elementos em jogo: a soma da distância do pacman à posição X com distância da posição X ao alvo, subtraindo o produto da distância do pacman ao alvo por mil. O custo dos fantasmas atacantes na localização X é o rácio de 500,000 pelo quadrado da distância de Manhattan do fantasma ao ponto X. Se a localização tem um fantasma, então terá o custo do fantasma como um milhão. Se a localização situa-se num canto então terá um custo de canto igual a 5000. A pílula do poder nos cantos mais próximos à pílula do poder quando a distância do pacman ao fantasma mais próximo é 9 tem um custo de 100,000.

As regras são listadas por ordem decrescente de prioridade.

Regra 1

Se distância do pacman à pilula do poder mais próxima inferior ou igual a 5 e distância do pacman ao fantasma mais próximo superior ou igual a 4 e distância do pacman ao fantasma mais próximo da pílula do poder mais próxima do pacman superior ou igual a 6; então pára de se mover para uma emboscada perto da pílula poder mais próxima de um fantasma a se aproximar.

Regra 2

Se distância do pacman à pílula do poder mais próxima do pacman inferior à distância do pacman ao fantasma mais próximo da pílula do poder mais próxima do pacman; então move-se para a pílula do poder com a versão 1 do A*, se distância do pacman à pílula do poder mais próxima superior ou igual a 6, senão usa a versão 2 do A*.

Regra 3

Se distâncias do pacman ao fantasma mais próximo e fantasma comestível mais próximo inferior ou igual a 8; então move-se para fantasma comestível mais próximo.

Regra 4

Se distância do pacman ao fantasma mais próximo é menor ou igual a 8; então move-se para a pílula mais próxima.

Regra 5:

Se distância do pacman ao fantasma mais próximo é maior ou igual a 9 e que a distância do fantasma comestível mais próximo é menor ou igual a 8; então move-se para o fantasma comestível mais próximo.

Regra 6:

Se a distância do pacman ao fantasma mais próximo é maior ou igual a 9; então move-se para a pílula mais próxima.

2.2.18 ICE Pambush 4

O ICE Pambush 4 é um melhoramento do ICE Pambush 3. Várias mudanças foram feitas como considerar a direção dos fantasmas, desconsidera o mapa interno do 3, afinou distâncias (estão nas regras de forma manual) em relação à versão 3 com algoritmos genéticos, usa 10 regras com prioridades diferentes, usa 2 versões diferentes do DFS (DFS-A e DFS-B) e Dijkstra.

As regras são referidas por ordem decrescente de prioridade, sendo que a 1 e a 2 fazem parte só do mapa 1.

Regra 1

Se distância do pacman à pílula do poder mais próxima <=5 e distância do pacman ao fantasma mais próximo <=10 e distância do pacman ao fantasma mais próximo da pílula do poder mais próxima >=7; então pare de mover-se e entra no estado de emboscada, à espera para depois apanhar a pílula do poder, quando o fantasma estiver perto.

Regra 2

Se existe pelo menos uma pílula do poder e nenhum fantasma comestível existe e pacman está no estado de emboscada e (distância do pacman ao fantasma mais próximo <=3 ou distância do pacman ao fantasma mais próximo à pílula do poder mais próxima) <=7); então move-se para a pílula do poder mais próxima do pacman com o DFS-B.

Regra 3

Se existe pelo menos uma pílula do poder e nenhum fantasma comestível e (distância do pacman ao fantasma mais próximo<=10 e distância do pacman à pílula do poder mais próxima <=4); então move-se para a pílula do poder mais próxima do pacman com DFS-B.

Regra 4

Se existe pelo menos uma pílula do poder e nenhum fantasma comestível existe e distância do pacman ao fantasma mais próximo <=10; então move-se para a pílula do poder com o DFS-A.

Regra 5

Se existe pelo menos um fantasma comestível e a distância do pacman ao fantasma atacante mais próximo <=10 e a distância do pacman ao fantasma comestível mais próximo <=8; então move-se para o fantasma comestível mais próximo do pacman com o DFS-A.

Regra 6

Se pelo menos uma pílula existe e distância do pacman ao fantasma mais próximo <=10 e não existe pílula não situada entre um par de dois pontos de travessia, cujo percurso ligado contém uma pílula de energia; então move-se para a pílula mais próxima do pacman com o DFS-A.

Regra 7

Se existe pelo menos uma pílula e distância do pacman ao fantasma mais próximo <=10; então move-se para a pílula# (pílula# significa não situada entre um par de dois pontos de travessia, cujo percurso ligado contém uma pílula de energia) mais próxima com DFS-A.

Regra 8

Se existe pelo menos um fantasma comestível e distância do pacman ao fantasma mais próximo >=11 e distância do pacman ao fantasma comestível mais próximo <=12; então move-se para o fantasma comestível mais próximo com DFS-B.

Regra 9

Se existe pelo menos uma pílula e (nenhuma pílula# existe ou distância do pacman ao fantasma mais próximo>=11 e distância do pacman à pílula do poder mais próxima>=20 e distância do pacman à pílula mais próxima<=8); então move-se para a pílula mais próxima do pacman com o DFS-B.

Regra 10

Se existe pelo menos uma pílula e distância do pacman ao fantasma mais próximo >= 11; então move-se para a pílula# mais próxima do pacman com DFS-B.

O DFS-A com profundidade 5 considera o custo da distância, o custo do fantasma e o custo do canto.

O DFS-B com profundidade 10 considera o custo da distância. O algoritmo de Dijkstra continua a calcular apenas a distância entre nós.

2.2.19 ICE Pambush 3

Um agente com 9 regras que recorre a algoritmos como o A* e Dijkstra.

As regras são referidas por ordem decrescente de prioridade.

Regra 1

Se a distância do pacman à pílula do poder mais próxima <=5 e a distância do pacman ao fantasma atacante mais próximo <=8 e a distância do pacman ao fantasma atacante mais próximo à pílula do poder mais próxima>=6; então páre de mover-se, entra no estado de emboscada e fique à espera do fantasma atacante até estar mais próximo, perto da pílula do poder mais próxima.

Regra 2

Se existe pelo menos uma pílula do poder e nenhum fantasma comestível, o pacman está no estado de emboscada e (distância ao fantasma atacante mais próximo <=4 ou distância ao fantasma atacante mais próximo e mais perto da pílula do poder mais próxima <=6); então move-se para a pílula do poder mais próxima.

Regra 3

Se existe pelo menos uma pílula do poder, nenhum fantasma comestível e distância do pacman ao fantasma atacante mais próximo <=8 e distância do pacman à pílula do poder mais próxima <=4; então move-se para a pílula do poder mais próxima.

Regra 4

Se existe pelo menos uma pílula do poder e nenhum fantasma comestível e distância do pacman ao fantasma atacante mais próximo <=8; então move-se para a pílula do poder.

Regra 5

Se existe pelo menos um fantasma comestível e distância do pacman ao fantasma atacante mais próximo <=8 e distância do pacman ao fantasma comestível mais próximo <= 10; move-se para fantasma comestível mais próximo.

Regra 6

Se existe pelo menos uma pílula e distância do pacman ao fantasma atacante mais próximo <=8; então move-se para a pílula mais próxima.

Regra 7

Se pelo menos um fantasma comestível existe e distância do pacman ao fantasma atacante mais próximo>=9 e distância do pacman ao fantasma comestível mais próximo <=10; então move-se para o fantasma comestível mais próximo.

Regra 8

Se pelo menos uma pílula existe e distância do pacman ao fantasma mais próximo >=9 e

distância do pacman à pílula do poder mais próxima >=20 e <=10; então move-se para a pílula mais próxima.

Regra 9

Se existe pelo menos uma pílula e distância do pacman ao fantasma mais próximo >=9; então move-se para a pílula mais próxima.

As regras 9, 8, 7, 3, 2 usam a versão 2 do A*. As regras 4, 5 e 6 usam a versão 1 do A*. O Dijkstra apenas dá uma ideia mais precisa da distância entre nós. A versão 1 do A * ao encontrar um caminho, considera o custo de distância, o custo de fantasma, e os custos de canto, enquanto a versão 2 do A* considera apenas o custo da distância.

2.2.20 ICE Pambush 5

O agente é composto por um conjunto de 13 regras disparadas pela seguinte ordem de prioridade: regra x tem menos prioridade que a regra y se e só se x>y.

Regra 1

Se (distância do pacman à pílula do poder mais próxima menor ou igual a 5) E (distância do pacman ao fantasma mais próximo entre 3 e limite superior Border) E (distância do pacman ao fantasma mais próximo da pílula do poder maior ou igual a 7); então o pacman pára de mover-se e entra no estado de emboscada ao canto ou ponto de travessia próximo da pílula do poder esperando que o fantasma aproxime-se.

Regra 2

Se (existe pelo menos uma pílula do poder E não há fantasmas comestíveis E o pacman está no estado de emboscada E distância ao fantasma mais próximo inferior ou igual a 3) OU distância do pacman ao fantasma mais próximo da pílula do poder inferior ou igual a 7: então move-se para a pílula do poder com DFS-B

Regra 3

Se existe pelo menos uma pílula do poder E nenhum fantasma fantasma comestível existe E (distância do pacman ao fantasma mais próximo menor ou igual a Border) E (distância do pacman à pílula do poder inferior ou igual a 4); então o pacman move-se em direção à pílula do poder com DFS-B.

Regra 4

Se (existe pelo menos uma pílula do poder) E (nenhum fantasma comestível existe) E (distância do pacman ao fantasma mais próximo inferior ou igual a Border); então move-se para a pílula do poder mais próxima com DFS-A.

Regra 5

Se existe pelo menos uma pílula do E distância do pacman ao fantasma mais próximo inferior ou igual a Border E distância do pacman ao fantasma comestível mais próximo inferior ou igual a 8; então o pacman move-se para o fantasma comestível mais próximo com DFS-A.

Regra 6

Se o número de pílulas abaixo de 8 E todas as pílulas ao longo de uma linha reta E distância à pílula mais próxima mais o número de pílulas menos um inferior ao produto da distância da pílula

mais longe do pacman à pílula mais próxima do fantasma por 0.8; então o pacman move-se à pílula mais próxima com DFS-B.

Regra 7

Se pelo menos uma pílula existe E distância do pacman ao fantasma mais próximo inferior ou igual a Border E nenhuma pílula#(pílula situada entre um par de pontos de cruzamento cujo o caminho ligado contém uma pílula do poder e não situa-se numa zona perigosa); então o pacman move-se para a pílula mais próxima com o DFS-A.

Regra 8

Se pelo menos uma pílula existe E distância do pacman ao fantasma mais próximo inferior ou igual a Border; então o pacman move-se para a pílula# mais próxima com DFS-A.

Regra 9

Se pelo menos um fantasma comestível existe E distância do pacman ao fantasma atacante superior a Border E distância do pacman ao fantasma comestível mais próximo inferior ou igual a 12; então o pacman move-se para o fantasma comestível mais próximo com DFS-B.

Regra 10

Se um item existe E distância do pacman ao item inferior ou igual a 9 E distância do pacman ao item inferior ou igual à distância do item ao fantasma mais próximo do item: então o pacman move-se para o item com DFS-B.

Regra 11

Se pelo menos uma pílula existe E pelo menos uma pílula existe numa zona perigosa E distância à zona perigosa inferior ou igual a 19; então o pacman move-se para a pílula mais próxima na zona perigosa.

Regra 12

Se existe pelo menos uma pílula E (nenhuma pílula# existe OU (distância do pacman ao fantasma mais próximo superior a Border E distância do pacman à pílula do poder mais próxima superior ou igual a 20 E distância do pacman à pílula mais próxima inferior ou igual a 8)); então o pacman move-se para a pílula mais próxima com DFS-B.

Regra 13

Se existe pelo menos uma pílula E distância do pacman ao fantasma mais próximo superior a Border; então o pacman move-se para a pílula# com DFS-B

O DFS-A considera o custo da distância, o custo dos fantasmas e o custo dos cantos com profundidade 5.

O DFS-B considera só o custo da distância com profundidade 10.

O custo da distância para as 2 versões do DFS num ponto X é soma da distância do pacman ao ponto X, da distância do ponto X à localização alvo e do produto da distância do pacman ao alvo com 1000.

Os fantasmas têm diferentes custos consoante a sua situação no labirinto.

O custo de um fantasma num canto ou ponto de cruzamento é 500,000.

O custo de um fantasma num canto ou ponto de cruzamento atrás de um fantasma capturando um pacman no mesmo corredor é 2,000,000.

Noutra situação o custo do fantasma num ponto X afastado do fantasma Y é o quociente de 500,000 pela distância do fantasma Y ao ponto X.

O custo de cada canto é 5,000.

2.2.21 Nozomu Ikehata & Takeshi Ito

Algoritmo UCT (do inglês, Upper Confidence bound applied to Trees, que significa limite de confiança superior aplicado a árvores), escrito em C++ e que foi inspirado num algoritmo com grande sucesso para o jogo Go. O objetivo é evitar situações de encurralamento, que os fantasmas fazem ao pacman e ao mesmo tempo limpar o mapa.

O algoritmo UCT é um dos algoritmos de procura em árvore baseado em simulações de Monte Carlo, realizadas antes do jogo real e que prioriza movimentos vantajosos, medindo um ganho médio. Os movimentos vantajosos têm um grande número de simulações associadas e a árvore cresce quando o número de simulações excede um corte; permitindo uma profundidade cada vez maior até uma condição de paragem. O resultado do algoritmo UCT é fornecer táticas e informação vital às condições das táticas que determinam a estratégia a adotar.

O UCB (limite de confiança superior e que faz parte do UCT) serve para alocar um número de simulações para cada movimento disponível e é dado pela fórmula:

$$UCB(i)=\begin{cases} X_i + C\sqrt{(\frac{(\ln T)}{T_i})} & para\ (T_i>0) \\ n & para\ (T_i=0) \end{cases}$$

que substitui o papel de uma função de avaliação. As variáveis i, X_i , T_i , T e C significam respetivamente o nó corrente, ganho médio de um nó filho i, o número de procuras realizadas para o mesmo nó filho i, o número de procuras realizado para o nó pai e o parâmetro de balanço. n é um valor suficientemente largo para começar. A simulação fornece ganhos baseado nos seus resultados. Os ganhos são: sobrevivência (um valor binário que indica se o pacman sobreviveu, por que a árvore tem tamanho limitado), a percentagem de pílulas comidas em relação ao total que o nível oferece no inicio e a percentagem de fantasmas comidos.

A árvore tem como nó raiz a posição atual do pacman e cada ligação pai/filho corresponde a um caminho em liga reta que liga 2 pontos de cruzamentos com outros caminhos em linha reta. Esse caminho foi denominado por caminho C. A árvore de jogo não restringe os movimentos dos fantasmas para não a fazer crescer abruptamente, mas são escolhidos de uma maneira não-determinística.

As simulações recomeçam, quando o pacman atingiu um nó terminal sem encontrar um fantasma numa procura em árvore. O objetivo de uma simulação é obter um ganho para uma experiência. Durante uma simulação (não é em tempo real), o pacman e os fantasmas farão os seus respetivos movimentos alternadamente, sob um conjunto único de regras que são diferentes das regras de jogo original do Ms. Pacman. Estas regras são alteradas para o propósito de simplificação

do jogo, tal que o custo computacional para uma atualização será suficientemente pequena na execução do algoritmo heurístico; enquanto dando considerações aos movimentos das personagens no jogo Ms Pacman original para uma extensão que não indetermina as vantagens da discretização do espaço.
P é o pacman. G é o fantasma. C é uma passagem de nó pai para nó filho.

número	regra
1	P e G moverão em frente num caminho C (não volta atrás até chegar a um cruzamento)
2	P e G determinarão o curso seguinte do movimento só nos pontos de cruzamento. Os pontos de cruzamento são nós, que têm mais de 2 nós adjacentes.
3	P saltará a atualização do ciclo seguinte sempre que ele come um certo número de pílulas
4	P atualizará o seu estado sucessivamente sempre que ele vira um certo número de cantos
5	Quando P come uma pílula do poder, G torna-se comestível e inverte a sua rota.
6	G no estado comestível não fará a sua atualização do ciclo seguinte depois de cada 2 movimentos
7	Uma vez G ter-se tornado comestível, o seu estado não será redefinido
8	G não voltará à vida uma vez que seja comido
9	G adicional não sairá da cabana dos fantasmas
10	Não existirá item de bónus

Tabela 5: Regras aplicadas no espaço discretizado. Serve para simplificar a criação da árvore de jogo. As regras discretizadas são diferentes dos movimentos do jogo Ms Pacman.

Durante uma simulação, cada personagem move-se de uma maneira não-determinística, baseado num certo conjunto de políticas comportamentais. A política comportamental do pacman é eliminar um caminho, obviamente perigoso. A exceção é o pacman estar preso entre 2 ou mais fantasmas. As regras seguintes são estabelecidas para atingir este objetivo.

número	regra
1	O pacman não voltará atrás no caminho C, se o pacman tinha escolhido um caminho no ponto de cruzamento
2	Ao escolher um caminho no ponto de cruzamento, se existe um fantasma atacante no caminho C e o pacman pode mover-se e o fantasma está dirigindo-se ao pacman; então o pacman elimina esse caminho C da sua lista de caminhos disponíveis
3	Se um único caminho C não é determinado pela aplicação das regras (1) e (2), o pacman escolherá aleatoriamente um caminho C dos caminhos C restantes.
4	Se existe um fantasma atacante dentro de um certo número de grelhas à frente do pacman num caminho C, o pacman inverte o seu andamento no local

Tabela 6: Regras para o movimento não determinístico do pacman [36]

Uma vez definido o movimento não-determinístico do pacman é definido os movimentos não-determinísticos dos fantasmas.

A probabilidade P de um fantasma aproximar-se do pacman é dado por:

$$P(G_{type}) = A(G_{type}) \frac{(AP - RP)}{AP}$$

, onde AP representa o número de todas as pílulas iniciais no labirinto e $A(G_{type})$ é a agressividade dos fantasmas que são valorizados conforme os valores da janela seguinte.

Cor do fantasma	agressividade	Pontos alvo
vermelho	0.9	À frente do pacman
laranja	0.8	Atrás do pacman
rosa	0.7	À frente do pacman
Azul claro	0.6	Por detrás do pacman

Tabela 7: Agressividade dos fantasmas nos seus pontos alvo [36]. O agente Bruce 2 [39] concluiu qual o comportamento dos fantasma.

As condições de terminação da simulação são 3: o pacman teve contacto com um fantasma que não está comestível, todas as pílulas do nível foram comidas ou um determinado número de ciclos foram realizados depois da simulação ter começado. O número de simulações foi definido a 300 e o número máximo de ciclos de simulações foi definido a 30.

Há uma otimização que sugere que o ganho baseado nas pílulas consumidas deve ser substituído pela seguinte fórmula:

$$X = \sum_{k=1}^{n} W_{i,j} \; (se \; n_{pílulas \; do \; poder} = 0)$$
$$n \; (se \; n_{pílulas \; do \; poder} > 0)$$

X contabiliza o perigo existente. $W_{i,j}$ é o nível de perigo, no ponto (i,j), distribuído da seguinte forma pelo nível na figura seguinte.

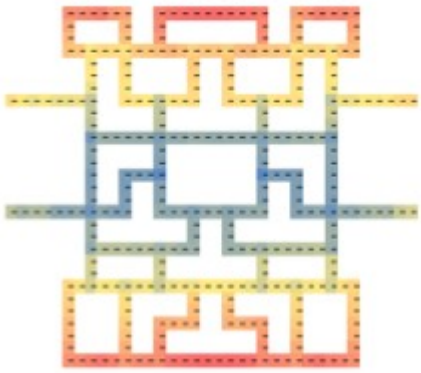

Figura 19: Perigo no mapa. As zonas mais avermelhadas são as mais perigosas [36]

A variável 'n' é um valor suficientemente pequeno para os caminhos onde as pílulas do poder possam ser obtidas durante a simulação, por causa do risco ser mínimo e não haver necessidade de tê-las como risco ao consumir.

Quando a simulação termina, a informação de cada nó da árvore é atualizada conforme os ganhos obtidos. A informação é propagada dos nós terminais da árvore até à raiz. O nó pai fica com o valor máximo de cada um dos ganhos dos seus filhos.

Existe um conjunto de 5 condições que sugerem objetivos do pacman:

- se a taxa de sobrevivência do frame anterior estava acima de um corte, então as táticas “alimentar-se” são adotadas;
- se a taxa de sobrevivência do frame anterior estava abaixo de um corte, então as táticas “sobrevive” são adotadas;
- se os fantasmas estão comestíveis e o fantasma mais próximo está a menos de uma certa distância, então as táticas “persegue-os” são adotadas;
- se os fantasmas estão comestíveis e o fantasma mais próximo está acima de uma certa distância, então as táticas “alimentar-se” são adotadas;
- se todos os fantasmas estão no estado atacante, então as táticas de “sobrevive” são adotadas.

Os objetivos das táticas conforme indicadas nas condições são 3: “alimentar-se”, “sobrevive” e “persegue-os”. Os métodos de decisão associados a cada um dos objetivos são descritos na tabela seguinte:

Objetivos	Método de seleção
sobrevive	Seleciona um caminho no qual a taxa de sobrevivência é a mais alta entre os caminhos ligados ao pacman
alimentar-se	Seleciona o caminho o qual contém mais pílulas que possam ser comidas de um dos caminhos ligados ao pacman
persegui-os	Seleciona o caminho no qual mais fantasmas possam ser comidos de entre os caminhos ligados ao pacman

Tabela 8: Objetivos+método de seleção das táticas escolhidas, consoante os resultados do UCT [36]

A maior vantagem do algoritmo é evitar muitas situações em que o pacman está encurralado pelos fantasmas, aumentando as hipóteses de sobrevivência. Os autores referem que ao prever os movimentos dos fantasmas (o que é muito difícil) é possível evitar movimentos que levam ao estado em que o pacman está encurralado.

2.2.22 Cerrla

Cerrla ou **C**ross-**E**ntropy **R**elational **R**einforcement **A**gent é um agente que utiliza uma aprendizagem por reforço relacional com recurso ao processo de decisão de Markov, uma otimização através de um método de estudo da entropia do jogo, especialização restrita e RLGG (generalização geral mínima relativa ou **R**elational **L**east **G**eneral **G**eneralization). O agente em vez de ser programado com uma hierarquia de regras, que definem a sua estratégia; cria a sua própria estratégia para jogar o pacman, através de uma combinação de especialização de regras incremental e um método de estudo de entropia de jogo, o qual ajusta dinamicamente a taxa de aprendizagem conforme o progresso do agente.

O RRL (relational reinforcement learning ou em português aprendizagem por reforço relacional) é uma extensão da aprendizagem por reforço com uma representação do ambiente para o agente trabalhar. A representação é um formalismo de relações entre objetos e ações descritos como relações lógicas de primeira ordem, numa framework para processos de tomada de decisão de Markov. O objetivo da framework é minimizar o esforço do programador.

Será possível representar o ambiente do pacman nesta framework?

Um ambiente de vídeo jogo é visto como uma coleção de objetos, relações, ações e ganhos. A representação do ambiente do agente é definida como grupos de objetos com relações (distância entre 2 objetos, estados dos fantasmas, cruzamentos existentes, a ação que faz mover-se para um nó, a ação que o impede de mover-se para um nó, a ação que o faz mover-se em direção a um encontramento e uma hierarquia entre objetos de jogo). Portanto um ambiente de jogo como o pacman pode ser representado numa framework para processos de decisão de Markov, que utiliza uma aprendizagem por reforço relacional.

Observações
distância(coisa,#D) coisa está a #D unidades do pacman pelo caminho mais curto **comestível**(Fantasma) fantasma comestível **Piscando**(Fantasma) fantasma piscando **CruzamentoSeguro**(cruzamento, #J) cruzamento tem valor seguro #J. O predicado indica se o pacman pode chegar 4 passos antes do que o fantasma mais próximo do pacman ao cruzamento através da diferença entre a distância do caminho mais curto entre o cruzamento e o fantasma mais próximo pela distância do caminha mais curto entre o cruzamento e o pacman.
Ações (onde #D (distância) e #J (Cruzamento seguro) são meta-informação para resolução de ações)
MoverPara(Coisa, #D) move-se em direção a Coisa **MoverDe**(Coisa,#D) não mover-se para Coisa **ParaCruzamento**(Junção,#J) move-se para cruzamento
Tipo de hierarquia
Todos os objetos são coisas. Os fantasmas têm mais prioridade; depois fruta; depois pílula; depois pílula do poder e finalmente centro dos fantasmas.
Cruzamento uma interceção de caminhos

Tabela 9: Relações lógicas de primeira ordem [49]. O objeto que está mais próximo ou o cruzamento com a segurança mais alta determina a direção. Se existe mais do que uma direção melhor, o pacman segue uma direção que não o faz voltar atrás.

Uma vez definida a representação do ambiente, qual a estratégia implementada que trabalha com essa representação?

O modelo de processos de decisão de Markov é um standard em inteligência artificial, aprendizagem de máquina, investigação operacional e de campos relacionados. Um MDP é definido por um tuplo (Z,A,P,r) onde: Z={1,...,n} é um conjunto finito de estados; A={1,...,m} é o conjunto de ações possíveis por parte do gerador de decisões; P é a matriz de probabilidade de transição entre estados; r(z,a) é o ganho ao realizar a ação a a partir do estado z (r pode ser uma variável aleatória). As ações mudam os estados através de funções de transição. Sem ações não há transições entre estados. Uma política π é uma regra que determina, para cada história ou episódio $H_k = z_1, a_1, ..., z_{k-1}, a_{k-1}, z_k$ de estados e ações, a distribuição da probabilidade de decisão do gerador de ações no instante de tempo discreto k. A política é chamada de Markov, caso cada ação seja determinística e dependa só do estado corrente z_k .

O gerador de decisões observa o estado corrente z_k a cada instante de tempo k e determina a ação a ser tomada (ou a_k), controlado por política π . O resultado é um ganho dado por $r(z_k, a_k)$, denotado por r_k e um novo estado z' é escolhido de acordo com a probabilidade de

transição do estado corrente para o estado novo. O objetivo do gerador de decisão é maximizar os ganhos em função da política.

Como é que o gerador de decisões maximiza os ganhos?

O método de estudo da entropia (ou cross-entropy) do jogo é um algoritmo de otimização e a maneira que ele tem para aprender políticas, o qual mantém uma distribuição de soluções possíveis para o problema. O método pode ser usado para uma resolução eficiente de problemas de otimização estocásticos e determinísticos difíceis e para encontrar estimadores eficientes de forma adaptativa (o que implica uma probabilidade de mudança suficientemente larga, que neste caso é dinâmica) para probabilidades de eventos raros.

Este método é usado ao mesmo tempo que é mantida distribuições múltiplas de regras relacionais para usar na geração de políticas relacionais.

O cross entropy consiste em 2 fases com 2 condições de paragem que produzem dados melhores e que possibilitam uma maior convergência na atualização das amostras, diminuindo o espaço de procura. As 2 fases são geração de amostras e atualização da distribuição. As 2 condições de paragem para a geração de amostras são um limite para o número de iterações ou caso tenha encontrado a solução ótima.

Cada distribuição de amostras está contida num slot com mais 2 parâmetros (desvio e média de uso), para a mesma ação. Cada amostra tem uma probabilidade associada.

As regras geradas ($X=\{x_1,...,x_n\}$) são armazenadas numa distribuição de probabilidades $p_0(X):=\{p_1,...,p_n\}$. A distribuição é então amostrada N vezes ($\{x^{(1)},...,x^{(N)}\}$), selecionando dados baseado na sua (inicialmente igual) probabilidade onde $x^{(i)}=x_j$ com probabilidade p_j . As amostras são então testadas com funções f(x) e ordenadas de forma descendente. N_E 'elite' sub-amostras $E=\{e^{(1)},...,e^{(N_E)}\}$ são então extraídas das amostras (onde $e^{(i)}=x^{(j)}$). Amostras elite são amostras com $f(x^{(j)})\geq f(x^{(\rho \cdot N)})$ onde tipicamente $\rho:=0.05$. A distribuição observada $p'(X):=\{p'_1,...,p'_n\}$ é então calculada usando a frequência de dados vistos na amostra elite, dado por:

$$p'_j:=\frac{(\sum_{(e^{(i)}\in E)} igual\ a\ 1(se\ e^{(i)}=x_j), igual\ a\ 0(caso\ contrário))}{N_E}$$

p'_j é a contagem de todas as amostras elitistas que representam os dados x_j e N_E é o número total de amostras elite.

A distribuição é atualizada numa moda passo a passo, utilizando α (tipicamente igual a 0.6) para modificar a distribuição de probabilidades e para permitir a variância ao longo das iterações do algoritmo:

$$p_j := \alpha * p'_j + (1-\alpha) * p_j$$

, que serve para qualquer slot.

As políticas são geradas de uma distribuição D_S das distribuições da regra (uma coleção de slots S, onde cada slot tem probabilidade p_S e tamanho |S|). O slot tem a notação $a : S^a = \{r_1^n, \ldots r_n^n\}$. A média varia entre zero e infinito, sendo inicialmente 0.5. O desvio standard varia entre 0 e 0.5 e é inicialmente 0.5.

A política pode eventualmente ser re-amostrada se o agente não está progredindo. O agente regista os estados, que tem encontrado, para todos os episódios. Se o agente encontra um estado anterior visitado e não está recebendo mais do que o ganho médio (repetindo o mesmo estado pode ser lucrativo), uma variável re-amostragem $X \in [0,1]$ é incrementada por $(0.1 * avSteps)^{-1}$. avSteps é a média observada de número de passos por episódio e 0.1 representa a porção máxima do episódio que o agente pode visitar os mesmos estados. A variável X é decrementado pelo mesmo conjunto se um estado novo é encontrado. X representa a probabilidade com que o agente re-amostrará a política nesse passo (o qual redefine X:=0). O valor de 0.1 foi selecionado arbitrariamente e mais experimentos são necessários para encontrar o valor ideal.

A política que completa a maior parte dos passos dentro de um episódio é uma para o qual o ganho do episódio está associado. As políticas poderiam ser associadas com o ganho obtido enquanto elas forem ativas; mas nos ambientes onde o ganho é só recebido no fim do episódio, só a política final receberia ganho.

O número de elementos da elite é dado por:

$$N_E = max\left[\frac{\left(\sum_{(S \in D_S)} (\mu(S) * |S|)\right)}{\left(\sum_{(S \in D_S)} (\mu(S))\right)}, \sum_{(S \in D_S)} \mu(S)\right]$$

e $N = N_E / \rho * N_E$ é igual ao número médio de regras para um número de slots $\mu(S)$ relativamente alto, ou é igual à soma de $\mu(S)$ sobre todos os slots.

Depois de $2 * N_E$ amostras terem sido avaliadas, o agente começa a atualização da distribuição usando um parâmetro α' modificado passo a passo. O agente continua a amostrar

políticas e revendo as amostras elitistas, fazendo atualizações a α' todas as iterações. Quaisquer amostras em E (o qual tem sido apresentado para mais do que N passos) são removidos. Depois de N atualizações, α' coincide com α no método cross-entropy online de Lorincz e Szita.

A probabilidade observada p'_{Sj} para cada Sj em Ds é calculado pela primeira determinação da posição média de Sj dentro E:

$$q_{Sj} = 1 - \frac{1}{|(E_{Sj})|} * \sum_{(\pi \in E(Sj))} \left(\frac{(index(Sj, \pi))}{|(\pi)|}\right)$$

onde $|(\pi)|$ é o tamanho de π , E(Sj) são as políticas em E que utilizam o slot Sj, e $index(Sj, \pi) \in [0, |(\pi)|]$ retorna o indice de Sj na política, onde 0 é o primeiro.

A fórmula $p'_{Sj} = \frac{q_{Sj}}{(\sum_{(S \in D_s)} q_S)}$ é a normalização da distribuição de probabilidades. Devido a todo o slot ser amostrado quando uma nova política é gerada, p_S representa a probabilidade do slot sendo amostrado primeiro.

No processo de atualização, só as regras utilizadas e slots na política são atualizados, porém regras não usadas e slots são atualizados negativamente implicitamente, resultando num tamanho mínimo de políticas contendo só regras úteis.

A divergência de Kullback-Leibler é usada para determinar, quando a otimização convergiu: quando a divergência entre todas as atualizações de regras de distribuição e as atualizações de valor de slot é menos do que $\beta * \alpha$ onde $\beta = 0.01$ em experimentos.

Depois do processo de atualização, o agente pode criar regras novas, se a regra está pronta para especializar. Como é feita a especialização de regras?

As regras RLGG definem as condições gerais mínimas para cobertura de todas as ações possíveis para todo o estado. Sempre que o agente encontra um estado onde as regras RLGG não cobrem todas as ações possíveis, o RLGG é generalizado para cobrir ações não cobertas. O RLGG de toda a regra é computado pela substituição de todas as instâncias não numéricas dos objetos em ϕ em cada regra para variáveis parametrizáveis se os objetos não coincidem. Todos os outros objetos nas regras que não combinam são substituídos por uma variável anónima '?'. As regras são encaixadas pela interceção para tornarem-se uma regra RLGG. Qualquer condição contendo só variáveis anónimas são removidas da regra RLGG resultante. Os valores numéricos no RLGG são trabalhados de maneira diferente dos não numéricos. Os valores numéricos no RLGG depois de gerados, sofrem uma simplificação para serem armazenados como um intervalo numérico variável,

o qual define um valor mínimo e um valor máximo.

O agente precisa depois de regras mais especializadas com uma probabilidade de ser selecionada associada a cada regra. Existem 2 tipos de especialização: especialização guiado e divisão de intervalo. A probabilidade média inicial é de $(|S|)^{-1}$ para cada especialização.

Especialização guiada cria regras novas pelas condições especificas acrescentadas incrementalmente à regra, que têm sido apresentadas anteriormente quando a ação da regra tem estado disponível.

O agente aprende um modelo especifico constituído por associações entre observações de estado para inferir um conjunto de regras que definem quais observações de estado são sempre verdade, nunca verdade, e ocasionalmente verdade quando uma observação particular é verdade. O agente pode também usar um background do conhecimento do ambiente.

A divisão do intervalo cria regras especializadas pela divisão de um intervalo de tamanho r existente num número arbitrário de sub-intervalos mais pequenos.

Slot splitting é então alcançada pela utilização de uma especialização guiada imediata de cada regra RLGG como semente para cada slot split. As regras RLGG são criadas/modificadas quando elas não cobrem todas as ações, mas criando todas as especializações de uma vez pode encharcar o agente e resultados numa procura de força bruta sobre todas as soluções possíveis. Sempre que uma regra RLGG é criada/modificada, ou o conjunto de especializações guiadas possíveis muda, o procedimento de slot splitting é chamado para gerar/modificar slots para cada especialização de passo único da regra RLGG. Cada slot é então preenchido com mais um conjunto de especializações, usando a regra semente do slot (ou RLGG) como a regra para especializar. Regras pré-existentes dentro de slots modificados, que não são por mais tempo válidos (de acordo com as crenças do agente) são removidos. Similar à procura beam, regras úteis são então mais especializadas até as especializações falharem para melhorar a performance. Especializações candidatas são selecionadas pela amostragem de uma dada regra de cada slot. depois de todo o slot atualizar. Especialização de uma regra é restringida por 4 condições:

-não tem sido usada para especialização antes.

-tem sido amostrada no mínimo N_E vezes.

-é mais provável ser amostrada do que seu 'pai' (a regra que criou esta regra), isso é $p_r > p_{pai(r)}$.

-o slot da regra não está cheio (|S| é menos que o número máximo de especializações de regra possível para a ação a: o número de especializações guiadas possíveis + o número de divisões do

intervalo).

Todas as especializações aumentam o número de regras, desse modo aumentando N_E (e N) e abrandando a taxa de aprendizagem. Se a regra é atualizada positivamente, raramente ela pode ser abatida da distribuição e restringida de sendo criada outra vez. Depois de toda a atualização; se a regra r_i dentro de um slot tem uma probabilidade de seleção de $p_i \leq (1-\alpha)^{\theta} * (|S|)^{-1}$ é excluída, onde α é valor atualizado passo a passo pelo método cross-entropy, |S| é o tamanho do slot S e $\theta * N$ é o número mínimo de atualizações necessárias para uma regra inútil ser excluída (isto é a regra nunca estar presente em E). $\theta = 2$ são os experimentos.

```
1: iniciaAmbiente(jogo)  --Inicializa definições de predicados
--Os predicados indicam qual a representação do ambiente
2: D_S:={}                    --guarda o slot de distribuição
3: amostras:={}  --guarda a coleção de amostras
4: n:=0          --Número de politicas avaliadas
   --início do processo de decisão de Markov
5: repete
6:   π_n:=geraPolitica(D_s)  --Geração de politica do slot
7:   repete
8:     s:=observaEstado()        --observações do estado relacional
9:     π_n:=podeSerReamostrado()  --reamostragem se não progredindo
10:    a:=selecionandoAção(π_n, s)       --política determina ação
11:    tomarAção(a)                  --avalia ação relacional
12:  até episódioCompleto           --até estado terminar
13:  r_n:=Ambiente.ganho()    --notar ganho
--inicio do método de estudo da entropia
14:  amostras.acrescenta(π_n, r_n)              --acrescenta a amostra à coleção
15:  N_E:=determinaMinTamanhoElite(D_S)  --determina número de elementos da elite
16:  if n≥2*N_E  --atualiza depois de 2N_E amostras
17:      amostras.remove(π_i: r_i<rN_E)  --remove amostras que não fazem parte da elite
18:      α':=α/max(N-n, N_E)        --α' inicialmente baixo
19:      atualiza(D_S, amostras, α')        --Atualiza passo a passo
         --Utilização do RLGG
20:    podeSerEspecializadaNovaRegra(D_S)  --Especializa regras de interesse
--acabou por usar o método de estudo da entropia
21: Até convergida(D_S)              --Determinado pelas atualizações da soma
```

Figura 20: Pseudo-código do agente CERRLA [49].

2.2.23 BruceTong

O agente tem características de agentes anteriores como mapa pré-processado, pílulas iniciais armazenadas, paredes armazenadas, caminhos armazenados, mapa de jogo em forma de grafo e utilização do algoritmo Floyd-Warshall para caminhos entre um nó origem e um nó destino.

A inovação introduzida é a utilização do algoritmo minimax MTD (Memory-enhanced Test Driver) que serve para a tomada de decisões do pacman. A função de avaliação deste minimax foi afinada manualmente e tem por base as distâncias entre o pacman e os fantasmas, a mobilidade do pacman, o número de pílulas e quais devem ser comidas e as hipóteses de comer fantasmas comestíveis. A profundidade máxima da árvore de procura é 14 e é nos nós folha que a função de avaliação é chamada.

Existe ainda outra alternativa para este agente chamado o algoritmo minimax Negascout (muito comum nos jogos de xadrez atuais), mas é de pior qualidade segundo uma fonte do MIT [47]. As vantagens do minimax Negascout face ao minimax MTD é menos código e uma profundidade de procura fixa. O MTD pode ter variações de tempo tão grandes, que implica cortes na procura através de um temporizador, quando a profundidade máxima não foi atingida.

```
1:  função MTDF(raíz:tipo nó; f:inteiro; d:inteiro):inteiro;
2:    g=f;
3:    limiteSuperior=+∞;
4:    limiteInferior=−∞;
5:    repete
6:      se g=limiteInferior então beta=g+1 senão beta=g;
7:       g=AlfaBetaComMemoria(root, beta−1, beta, d);
8:      se g<beta então limiteSuperior=g else limiteInferior=g;
9:    until limiteInferior≥limiteSuperior;
10:   retorna g;
11: função AlfaBetaComMemoria(n:tipo nó; alfa, beta, d:inteiro):inteiro;

12:   se reaver(n)=OK então       table lookup de transposição
13:     se n.limiteInferior≥beta então retorna n.limiteInferior;
14:     se n.limiteSuperior≤alfa então retorna n.limiteSuperior;
15:     alfa=max(alfa, n.limiteInferior);
16:     beta=min(beta, n.limiteSuperior);
17:   se d=0 então g=avalia(n);       −− nó folha
18:   senão se n=MAXNODE então
19:     g=−∞; a=alfa;       −− guarda valor alfa original
20:     c=primeiroFilho(n);
21:     Enquanto(g<beta)e(c≠SEMFILHO) faz
22:       g=max(g, AlfaBetaComMemoria(c, a, beta, d−1));
23:       a=max(a, g);
24:       c=irmãoSeguinte(c);
25:     else       n é um MINNODE
26:       g=+∞; b=beta;       −− guarda o valor beta original
27:       c=primeiroFilho(n);
28:       Enquanto(g>alpha)e(c≠SEMFILHO) faz
29:         g=min(g, AlfaBetaComMemoria(c, alfa, b, d−1));
30:         b=min(b, g);
31:         c=irmãoSeguinte(c);
                        −− Tabela de transposição tradicional armazenando os limites
                              −− Falha baixo resultado implica um limite superior
32:   se g≤alfa então n.limiteSuperior=g; armazena n.limiteSuperior;
33:           −− Encontrado um valor minimax apurado − não ocorrerá se chamado com janela zero
34:   se g>alfa e g<beta então
35:     n.limiteInferior=g; n.limiteSuperior:=g; armazena n.limiteInferior, n.limiteSuperior;
36:           −− Falha de alto resultado implica limite inferior
37:   se g≥beta então n.limiteInferior:=g; armazena n.limiteInferior;
38:   retorna g;
39:
40: aprofundando iterativamente(raíz:tipo nó):inteiro −− função principal
41:   primeiro convidado:=0;
42:   para p=1 até máxima profundidade de procura faz
43:     primeiro convidado=MTDF(raíz, primeiro convidado, p);
44:     se ultrapassou limite de tempo então
45:       sai do ciclo;
46:   retorna primeiro convidado;
```

Figura 21: Pseudo-código do MTD[47]. Este é o algoritmo mais provável.

As previsões do modelo linear do meu agente deveriam ter vantagens face ao minimax MTD. Dado um fantasma (o inimigo) e o pacman (a figura que controlamos), em vez de criar todos os nós para o jogador inimigo (filhos na árvore de procura que são o resultado das jogadas possíveis do fantasma), supõe-se que o movimento do fantasma já é conhecido pelo modelo linear. A vantagem do modelo linear face ao minimax é o fator de ramificação causado pelo inimigo no algoritmo minimax afetar em muito o tempo de computação para as tomadas de decisão do agente. A maior desvantagem do modelo linear é a perda de precisão à medida que o horizonte aumenta.

2.2.24 Essexgp

O agente implementa uma árvore de expressão que foi criada usando Programação Genética.

Existem 2 fases: a criação dos cenários e a utilização de um algoritmo genético.

Os cenários que são 3 fazem parte da divisão e conquista do problema: limpar o mapa (ou comer todas pílulas e pílulas do poder), escapar dos fantasmas atacantes e capturar fantasmas comestíveis. O algoritmo genético é executado durante 50 gerações para cada um dos cenários em particular.

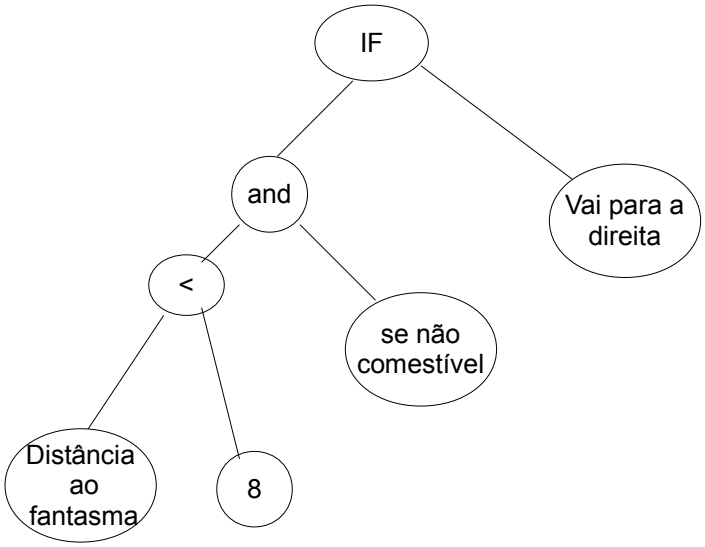

Figura 22: Exemplo de uma árvore de expressão possível para o problema

A representação dos indivíduos é uma árvore de expressão. O conjunto de funções é dividido em 3 categorias e cada uma tem 2 tipos de nós. Os tipos de nós são nós intermédios e nós folha. As categorias são aritmética, lógica e ação. A categoria aritmética consiste numa função, com os 4 operadores básicos (+,-,*,/), que retorna um valor numérico do nó folha, com base nas distâncias aos objetos e o tempo que resta aos fantasma para deixarem de ser comestíveis. A categoria lógica consiste em 3 tipos de funções booleana. A primeira (com base em 2 nós filho do terminal lógico) compara os seus valores usando ands e ors booleanos. A segunda que compara operadores aritméticos (¿ , ≥ , < , ≤ , = , ≠ ¿). A terceira que responde sim/não a um terminal (se o fantasma é comestível, se comeu tudo, se a pílula é segura e se o comestível é seguro). A ação é uma função só com uma condição e uma ação associada com o papel de pegar num valor booleano do filho e caso

seja verdade aciona uma direção para o pacman, para um nó adjacente mais próximo do nó alvo.

A população inicial de 100 indivíduos foi criada, usando a técnica meia-a-meia de Rumbeld.

O fitness é a pontuação média de 4 execuções (3 para irem para a lista externa e uma que testa as dessa lista externa). A lista externa é uma lista que contém os melhores indivíduos.

O método de seleção é o torneio.

Os indivíduos novos surgem em 30% através da mutação e 70% surgem através de crossover de 2 indivíduos. Os autores não referem qual o método de mutação ou crossover utilizado.

2.2.25 Haimat

Um agente reativo que é resultado dos cálculos de um algoritmo genético ao longo de 50 gerações (até não haver evolução dos indivíduos), que utiliza as bibliotecas standard do Java JGAP para computação evolutiva.

O tamanho da população é 100.

Os indivíduos são blocos de construções básicas, com conjuntos de funções de baixo-nível e conjuntos de terminais pré-definidos.

Os conjuntos de funções representam as 4 operações aritméticas básicas (+,-,*,/) e os cálculos mínimo, máximo e média de 2 números. A divisão segura (que é usada neste algoritmo) é um valor igual a um, atribuído ao resultado da divisão por zero.

Os conjuntos terminais consistem em 2 grupos principais: terminais que retornam um valor de pontuação único para uma dada direção, baseado numa propriedade de estado de jogo especifica e constantes que nunca mudam seus valores. Os terminais que retornam um valor de pontuação único são todos baseados numa direção específica da vista da posição corrente do pacman. Seus valores de retorno são normalizados para um intervalo [0,1] e são: distância do pacman à pílula seguinte, distância do pacman à pílula do poder seguinte, distância do pacman ao fantasma comestível seguinte, distância do pacman ao fantasma atacante seguinte, distância do pacman ao canto seguinte, distância do pacman ao próximo cruzamento, um valor booleano a indicar se o fantasma é comestível, um valor booleano a indicar se a pílula do poder está mais próxima do que um fantasma atacante, um conjunto de pílulas, um conjunto de pílulas do poder, um conjunto de fantasmas comestíveis, um conjunto de fantasmas atacantes e conjunto de cruzamentos. O valor retornado por estes terminais são calculados por um controlador todas as vezes que é chamado.

O grupo de terminais constantes consiste dos valores seguintes: -1.0, 0.0, 0.1, 0.5, 2.0, 5.0 e 10.0. Estes terminais constantes podem ser usados pelo algoritmo genético para modificar as pontuações dos terminais do tipo anterior a este.

O fitness é a média de pontuações ao longo de 10 simulações.

Os autores não referem como foi usado os crossover, seleções e mutações.

2.2.26 ICE Pambush MCTS

Este agente com 5 regras, é uma evolução da versão de captura de ecrã ICE Pambush 4, incluindo uma técnica de procura em árvore com recurso a técnicas de Monte Carlo inspirado no agente ICE gUCT. As melhorias face ao gUCT são 5.

O primeiro melhoramento indica que o melhor caminho é o que forneceu mais pontos e durou mais tempo. O segundo melhoramento indica que todos os fantasmas na simulação só se movem em relação ao pacman pelo caminho mais curto. O terceiro indica que o número máximo de vezes que o pacman pode voltar ao nó raíz é de 4000. O quarto indica que o pacman na simulação move-se sempre em frente e não volta atrás. O quinto indica que o MCTS pára e retorna a sua direção de saída 10 ms antes do tempo de jogo devido ao retorno da direção do pacman para o servidor.

O DFS encontra caminhos com custo mínimo com profundidade máxima de 5. Os custo são relativos a fantasmas, cantos e um cálculo relativo a 3 distâncias. O custo relativo aos fantasmas é 500,000 dividido pelo quadrado da distância entre o nó e o fantasma. O custo relativo ao canto é dado pelo produto de x ao cubo por Pi; onde x é maior que 14, Pi é definido pela evolução diferencial e x é o comprimento mínimo do corredor para incluir o nó com este custo. O último custo é dado por A+B-C. A é a distância do pacman ao nó alvo. B é a distância do nó corrente ao nó alvo. C é a distância do nó corrente ao pacman.

A evolução diferencial define os parâmetros Dm e Pi sem a execução do algoritmo de Monte Carlo contra a equipa Legacy. O número de individuos e sua gerações foram 10 e 181 respetivamente. O fitness de cada individuo é a média de pontos de 51 execuções. A probabilidade para adotar uma criança é de 0.005 para evitar uma derrapagem genética.

As regras listadas são disparadas por ordem decrescente de prioridade.

Regra 1:

Se a distância do pacman ao fantasma mais próximo está entre 8 e Dm excluídos e nenhum fantasma comestível existe; então faz emboscada. Dm é um parâmetro de distância derivado por evolução diferencial para o mapa i.

Regra 2:

Se a distância do pacman ao fantasma mais próximo é inferior a Dm, nenhum fantasma comestível existe e a distância do pacman à pílula do poder mais 2 é inferior à distância entre o fantasma atacante mais próximo e essa pílula do poder; então move-se para a pílula do poder mais

próxima.

Regra 3:

Se a distância do pacman ao fantasma é superior ou igual a Dm e pelo menos um fantasma comestível existe; então move-se o fantasma comestível mais próximo usando o algoritmo DFS para encontrar o caminho para a junção de nós em frente dos fantasmas.

Regra 4:

Se a direção de saída do MCTS conduz a um destino obviamente fatal com um fantasma; então escolhe o DFS para capturar a pílula mais próxima.

Regra 5:

Se a distância à pílula do poder mais próxima é igual a um; então faz emboscada.

2.2.27 Phantom Menance

O agente consiste num conjunto de modos e critérios para seguir determinado modo.

Os modos são escapar, comer pílulas, comer pílulas do poder, comer fantasmas e evitar caça.

Os critérios são número de rotas para escapar, tempo decorrido, duração de um fantasma comestível, distância ao fantasma comestível mais próximo, pontuação ao comer o fantasma, distância ao fantasma atacante e rácio do número de pílulas restantes pelo número de pílulas do poder restantes.

O pacman afasta-se para a zona segura mais longe no modo escapar.

O pacman escolhe o caminho com base na sua pontuação e distância no modo comer pílulas. Ele prefere caminhos com menos pílulas de forma a não deixar muitas isoladas.

O pacman espera perto da pílula do poder para depois comer a pílula do poder no modo comer pílula do poder.

O pacman dirige-se ao fantasma comestível mais próximo no modo comer fantasma comestível.

O pacman dirige-se a um ponto equidistante de 2 fantasmas quando encurralado por eles no modo evitar caça.

2.2.28 James

O agente utiliza um sistema baseado em regras de código ad-hoc e uns parâmetros otimizados através de um algoritmo genético. As regras não têm ordem de prioridade. As regras são construídas conforme uma exclusão de partes a cada momento que acaba por passar por uma condição, onde é possível concluir se vai passar por outra condição ou se vai tomar uma decisão final. O diagrama presente nesta secção permite ilustrar esse modo de funcionamento.

A primeira condição verifica se a densidade dos fantasmas é maior ou igual a um corte. Caso seja verdade verifica se a distância à pílula do poder mais próxima é inferior ou igual a 2. Caso seja falsa verifica se existe fantasmas comestíveis.

Caso a condição que verifica se há fantasmas comestíveis seja verdade, então vai atrás do fantasma comestível mais próximo, senão vai em direção à pílula mais próxima. Caso a condição que verifica se a distância à pílula do poder mais próxima é inferior ou igual a 2 seja verdade, então verifica se a distância ao fantasma mais próximo é inferior ou igual a 6, senão vai para um nó de fuga. Caso a condição que verifica se o fantasma mais próximo está a uma distância inferior ou igual a 6, então vai em direção à pílula do poder mais próxima, senão mantém posição corrente.

A densidade dos fantasmas é dado por:

$$D_n = \sum_{i=1}^{4} \left(\frac{T_{gi}}{(d(n, n_{gi}))} \right)$$

A função d(x,y) devolve a distância do ponto x ao ponto y. A variável n_{gi} é o ponto do fantasma gi. A variável n representa o ponto do pacman. A variável T_g é o valor de nuvem de densidade base do fantasma g e é definido por uma estratégia evolucionária 10+10, usando a equipa Legacy.

O nó de fuga da figura é determinado por um protocolo prioritário que devolve um nó seguro. O nó seguro tem como alvo maximizar a distância ao fantasma mais próximo e rotas sem fantasmas atacantes. Caso haja mais do que um nó seguro, escolhe o nó que dá mais pontos.

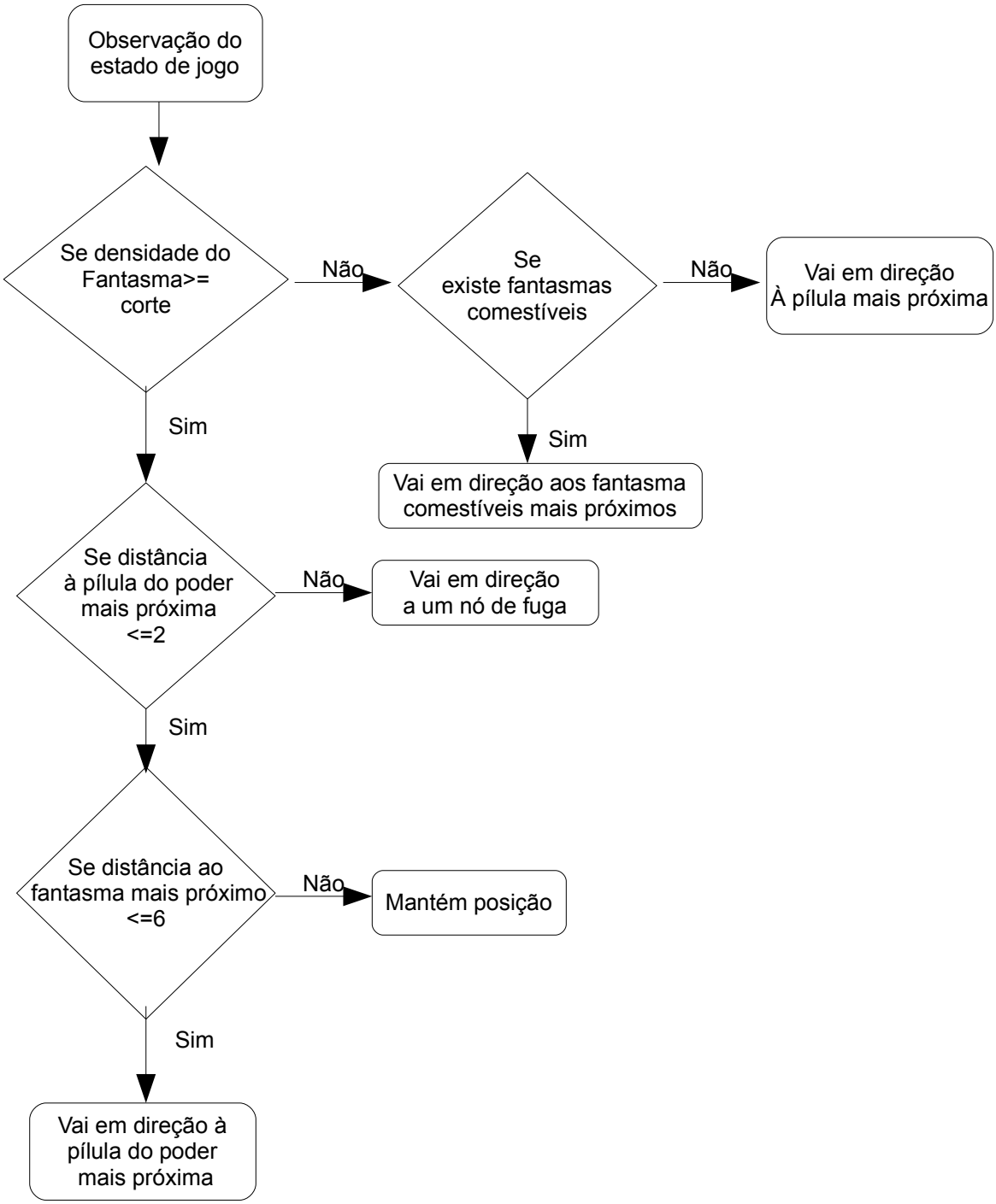

Figura 23: Diagrama de fluxo do processo de tomada de decisão do agente [46]

Os losângulos representam condições. As setas representam resultados das condições. Os retângulos com pontas arredondadas representam ações a tomar.

2.2.29 Spooks

Os autores partem do princípio que cada fantasma tem uma probabilidade de bloquear um caminho e que a segurança é uma prioridade. Cada junção de caminhos encontrada reduz a probabilidade de bloqueio. Essa probabilidade reduz-se tantas vezes como o número de direções possíveis para o pacman poder escapar. O agente é obrigado a usar uma procura em árvore, embora não referido. A fórmula de cálculo da probabilidade de bloqueio é dada por:

$Prob = 1-(1-g_1)(1-g_2)(1-g_3)(1-g_4)$, onde g_i é a probabilidade de bloqueio do fantasma i. A definição de probabilidade de acontecimentos independentes e a definição da probabilidade de um acontecimento complementar explicam o motivo desta fórmula ser assim, mas não é explicado como g_i foi calculado.

Os cálculos apresentados como a pontuação final só incidem sobre nós seguros. Um nó seguro tem uma distância de segurança em relação a um dos fantasmas. O horizonte define os nós seguros, mas pelo menos um nó vizinho do horizonte tem de ser inseguro.

Vários objetos têm um valor em pontos associado: uma pílula tem 10 pontos, uma pílula do poder tem 50 pontos e um fantasma comestível tem 100 pontos.

Os caminhos para escapar são pontuados a 50 mais a distância ao fantasma mais próximo multiplicada pela probabilidade do ponto não ser bloqueado e dividido pelo número de caminhos onde o pacman pode-se escapar (quanto menor o número de caminhos com que o pacman pode escapar, maior a prioridade desses caminhos.). O caminho com mais pontos é o escolhido.

2.3 Outros

A maioria dos agentes tiveram uma descrição detalhada neste estado de arte. Todos esses agentes eram de uma competição a nível internacional; mas houve outros agentes implementados, ao qual não conseguiu-se acesso à informação suficiente, para compreensão do seu funcionamento. Entre eles estão [14] e [2].

Um algoritmo genético como [14] para ser programado precisa de ter acesso a todos os pormenores típicos dos algoritmos genéticos como seleção, mutação, avaliação, inicialização da população, tamanho da população, tipo de cruzamento, representação dos indivíduos e caso trate-se de uma estratégia que usa vários núcleos de CPU qual o algoritmo paralelo utilizado. Não consegui ter acesso a todos esses pormenores importantes, mas tive acesso ao resultado.

Entre as implementações existentes houve uma implementação em redes neuronais [2] (e existem muitos tipos de redes neuronais). Eu só encontrei para além do artigo publicado na conferencia, as referencias de um livro ao qual estive sem acesso às páginas que considerava vitais para um entendimento apropriado do assunto em causa.

2.4 Conclusões

O estado de arte exibe muitos pormenores, que permitem a escolha de um método, para obter resultados significativos na competição e revela muitas dificuldades existentes no jogo.

Ao longo dos anos tem havido um aumento significativo das pontuações médias dos agentes propostos pelos participantes.

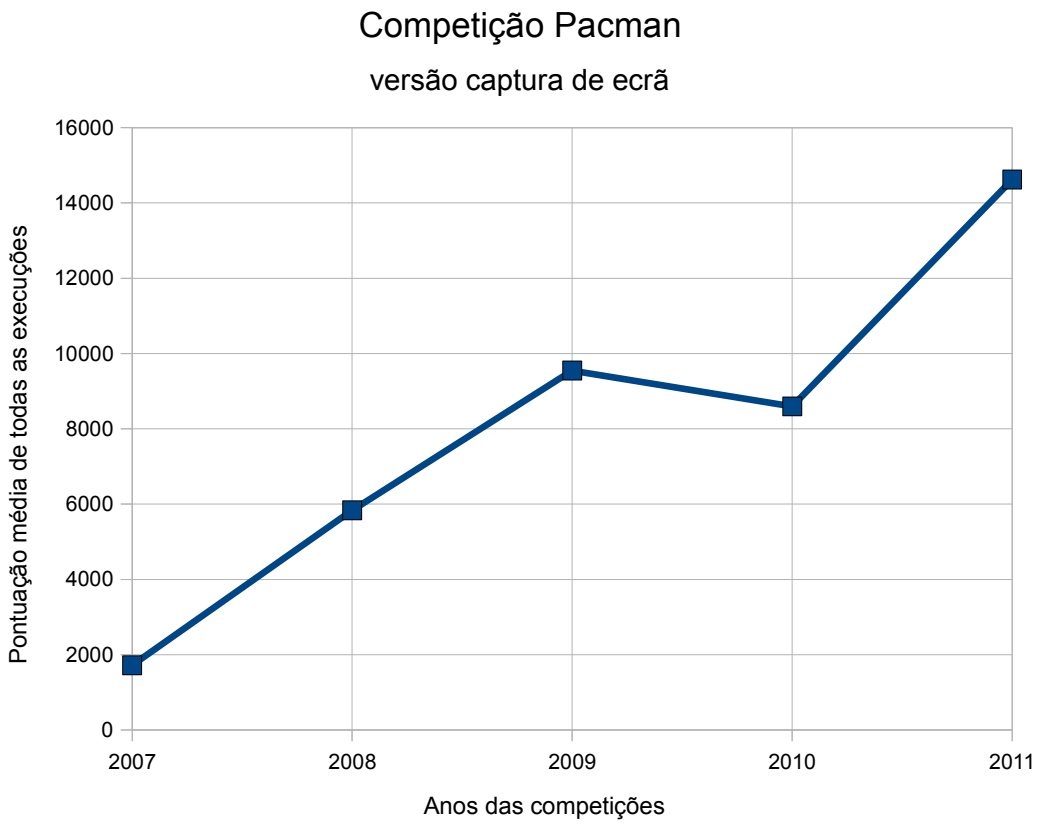

Figura 24: Evolução dos agentes propostos no site da competição

Os agentes que fizeram parte desta evolução podem ser classificados da seguinte forma.

	Agentes reativos/reativos com estado	**Agentes deliberativos**
Métodos clássicos	Kwong, Wirth, Sojoodi	Flensbak, Kim, NTNU CIG 2009, StrathPac
Métodos de inteligência computacional	Handa, Shirakawa	Bruce, Bruce 2, Robles, Pac-mAnt ICE, Jave, a generalidade dos ICE Pambush, ICE Pescape, NTNU CIG 2011, Nozomu Ikehata & Takeshi Ito

Tabela 10: Os agentes presentes na competição, que utilizam captura de ecrã, consoante o tipo de implementação (se foi reativo, deliberativo, com métodos clássicos ou com métodos de inteligência computacional)

	Agentes reativos/reativos com estado	Agentes deliberativos
Métodos clássicos	6533.33	9910
Métodos de inteligência computacional	9225	20420.83

Tabela 11: Média das pontuações máximas obtidas pelos agentes que utilizam captura de ecrã

	Agentes reativos/reativos com estado	Agentes deliberativos
Métodos clássicos		Spooks, PhantomMenance
Métodos de inteligência computacional	James, Haimat, essexgp	Emgallar, ICE Pambush MCTS, CERRLA, BruceTong

Table 12: Os agentes presentes na competição, que não utilizam captura de ecrã, consoante o tipo de implementação (se foi reativo, deliberativo, com métodos clássicos ou com métodos de inteligência computacional)

	Agentes reativos/reativos com estado	Agentes deliberativos
Métodos clássicos		36777.5
Métodos de inteligência computacional	23649	18435.25

Tabela 13: Média das pontuações máximas obtidas pelos agentes que não utilizam captura de ecrã. Os resultados têm conclusões diferentes dos da versão por captura de ecrã.

Os agentes reativos apenas reagem ao momento presente (processo estímulo-resposta), não têm memória, não fazem planos e nem sequer criam uma noção do futuro. Existem outros exemplos de agentes reativos como os veículos de Braitenberg ou os robots com arquitetura de subsunção, propostos por Brooks.

Os agentes reativos com estado têm para além dos agentes reativos variáveis que guardam informação obtida no passado e podem-se basear nessa informação para tomar decisões, mas não conseguem prever o futuro.

Os agentes deliberativos têm a capacidade de procurar no espaço de soluções por uma solução que seja o melhor possível dentro dos limites computacionais, podendo lembrar-se de dados

anteriores do seu ambiente de forma a condicionar eventos futuros, ou usar quaisquer mecanismos preditivos de forma a antever a melhor ação a ser tomada.

Os agentes deliberativos foram os que obtiveram uma pontuação melhor.

Vários pormenores estiveram presentes ao longo do estado de arte e que convêm realçar, mesmo para quem não esteja a par do assunto.

Quando a situação é inevitavelmente má, mesmo assim escolhe a menos má, trazendo vantagens associadas a esse comportamento. A natureza com um pequeno fator aleatório dos inimigos do pacman faz com que muitas das vezes a escolha numa situação condicionalmente má acabe às vezes numa fuga com sucesso do agente desenvolvido.

Um agente para o pacman pode ter estratégias melhores para programadores menos experientes em inteligência artificial como [18]; enquanto outros que apesar de fornecerem estratégias com tempos de computação elevados, tiveram muitos pontos devido a código fonte otimizado como [6]. Ficou demonstrado que agentes com tipos de estratégias novas (algo incentivado pelos organizadores) na competição como [13] (primeiro agente a inspirar-se em insetos na competição, que neste caso são as formigas), conseguiram melhores resultados do que os restantes nesse ano.

A situação de jogo complica à medida que há cada vez menos pílulas e pílulas do poder. Há várias situações possíveis. As pílulas podem estar longe pelo que há fantasmas pelo meio. As poucas pílulas que faltam podem estar muito dispersas pelo labirinto, mas isso pode ser combatido comendo as mais próximas. Os fantasmas podem estar no lugar onde situam-se as pílulas e como são poucas somos forçados a aproximarmos-nos dos fantasmas para completar o nível.

Ao considerar-se os agentes existentes, muitos têm dificuldade com os cantos.

Abordagens como grafos e procuras em árvore estão sendo muito exploradas.

Muitos autores (como [43]) referiram a importância de fazer previsões e como é difícil fazer previsões. A previsão de posições futuras suficientemente grande permite evitar situações de bloqueio. Qual a maneira mais eficaz da sua implementação?

3 Agentes reativos percursores da versão preditiva e um agente mais complexo e reativo com estados que maximize os pontos

3.1 Introdução

Vários agentes foram testados. Todos eles são reativos à exceção de um deles que é um reativo com estado. Todos estes agentes usam uma hierarquia de subsunção.

Um agente reativo é um agente que só tem noção do presente, que possuí um mapeamento de situações e respostas associadas, que permite que quando um estado ambiental ocorre, o agente executa a ação correspondente (processo estímulo-resposta).

Brooks criou a arquitetura de subsunção nos anos 80, com base em 5 princípios. A inteligência é uma propriedade emergente de certos sistemas complexos (primeiro principio). Comportamento inteligente surge como resultado da interação de um agente com seu ambiente (segundo principio). A inteligência está "no olho do observador", não é uma propriedade inata isolada (terceiro principio). A inteligência está situado no mundo, não em sistemas não embebidos, como provadores de teoremas ou sistemas especialistas (quarto principio). Comportamento inteligente pode ser gerado sem representação de conhecimento explícito, e sem raciocínio abstrato explícito (quinto principio).

Baseando-se nestas 5 hipóteses, Brooks propõe uma hiearquia de subsunção. Quando o agente quer decidir qual a ação a tomar, percorre a hierarquia de baixo para cima até que haja uma condição de uma das regras que satisfaça o estado observado do ambiente, sendo depois disparada uma ação associada à condição dessa regra. As regras com mais prioridade estão na parte debaixo do diagrama e à medida que sobe-se, elas vão tendo menos prioridade. As regras mais abaixo são responsáveis por comportamentos mais primitivos como evitar obstáculos ou garantir a sobrevivência. As regras do topo são responsáveis pelo objetivo final.

A arquitetura é computacionalmente eficiente, o que significa que comparado com outras arquiteturas de agentes consome poucos recursos de CPU.

Propomos o desenvolvimento de vários agentes baseados na hierarquia de Brooks como passo intermédio à construção do agente final.

Os resultados obtidos com estes agentes servem de comparação com um agente final. O agente final tem de conseguir pontuações o mais acima possível do seu antecessor reativo.

3.2 Implementação

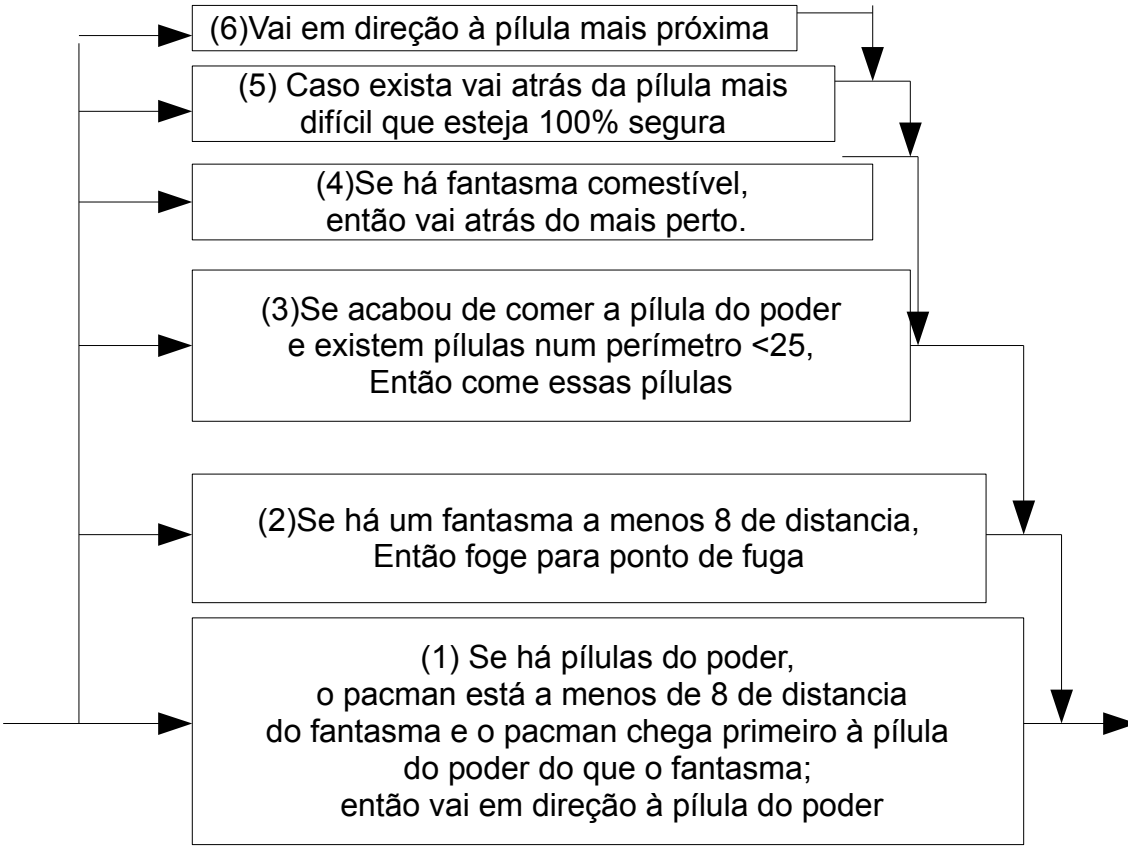

Figura 25: Agente reativo com 2 estados

A hierarquia usada consiste em 5 regras. As distâncias utilizadas nas regras são todas respetivas ao caminho mais curto entre 2 nós do grafo.

A regra 1 ao fazer o pacman correr para a pílula do poder controla um tipo de emboscadas, onde o pacman pode ficar cercado e a única hipótese é comer a pílula do poder para depois ser possível perseguir os fantasmas. Muitas vezes o pacman ao seguir o seu ponto adjacente mais longe do fantasma mais próximo, corre o risco de não ir em direção à pílula do poder, caso a pílula do poder esteja por perto.

A regra 2 evita o fantasma mais próximo, mas a distância máxima dos pontos adjacentes do pacman em relação ao fantasma mais próximo pode ser partilhado por vários pontos . Uma vez que o fantasma mais próximo está a menos de 8 de distância do pacman, o pacman foge para um nó adjacente que tem como prioridade evitar o primeiro fantasma a ameaçar mais próximo, depois o segundo fantasma a ameaçar mais próximo, depois o terceiro fantasma a ameaçar mais próximo, depois o quarto fantasma a ameaçar mais próximo e só depois o nó mais perto das pílulas. Neste

caso um fantasma mais próximo é uma ameaça caso esteja a menos de 62 de distância do pacman; exceto o fantasma mais longe, cujo o perímetro é 25.

A regra 3 é ativada depois de comer uma pílula do poder. A regra tem como objetivo comer as pílulas que estão dentro de um certo perímetro. O lugar nas proximidades da pílula do poder é mais perigoso do que os restantes e nessa altura os fantasmas não são perigosos. Quando comendo um fantasma comestível podemos desperdiçar esta oportunidade, por causa que os fantasmas comidos passam a atacantes e o pacman pode ir parar a um sítio muito longe e com muitos fantasmas atacantes. Se o pacman come pílulas à volta da pílula do poder, ele não precisa de voltar atrás nesse lugar perigoso. Quando uma pílula do poder é comida no nível corrente, ela não voltará a aparecer de novo. Desta maneira é pretendido uma maneira mais fácil de completar o nível em vez de mais pontos instantâneos com fantasmas comestíveis. Mesmo que coma um fantasma nesse perímetro não significa desvantagem, por que ele vai ter de limpar essa zona e pode ou pôde estar cercado. Tem a desvantagem de evitar às vezes fantasmas comestíveis fora do perímetro; ou de comer às vezes uma quantidade de pílulas escassas, com outras não comidas ao pé para depois ir comer pílulas, que estão do outro lado dentro do perímetro. As desvantagens são minimizadas por uma distância suficientemente pequena. Esta regra permitiu aumentar a pontuação média em relação a um reativo anterior.

A regra 4 é responsável por comer os fantasmas comestíveis. A vantagem reside no facto de que comer um fantasma comestível permite ganhar mais pontos do que se o pacman comesse só pílulas e pílulas do poder.

A regra 5 vai atrás das pílulas mais difíceis que estejam 100% seguras. A pílula é segura caso a distância do pacman à pílula e da pílula à pílula do poder mais próxima dessa pílula, mais a distância de segurança entre o pacman e o fantasma seja inferior à distância do fantasma mais próximo da pílula do poder referida à pílula do poder referida. As pílulas mais difíceis são as que estão mais longe da pílula do poder à exceção das que estão nas zonas mais perigosas do labirinto. As pílulas que estão nas zonas mais perigosas do labirinto têm mais prioridade. As zonas perigosas foram definidas em 3 dos 4 labirintos. Não foi possível determinar quais zonas no quarto labirinto são as mais perigosas.

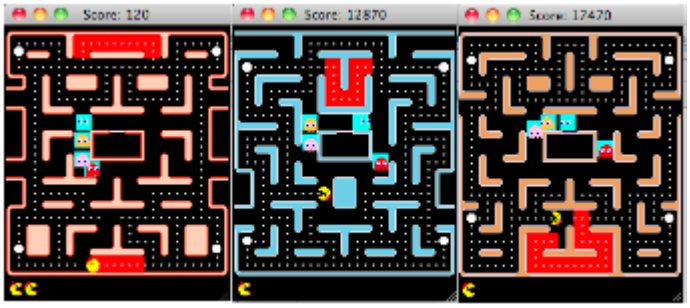

Figura 26: Zonas perigosas a vermelho dos vários níveis considerados

A regra 6 come pílulas mais próximas do pacman sejam elas pílulas do poder ou não. Num ambiente sem fantasmas permite que todas as pílulas e pílulas do poder sejam comidas para passar de nível.

Um reativo que saiba lidar com os cantos, quando os fantasmas estão por perto evitaria outros tipos de emboscadas; mas um agente preditivo poderia evitar muito mais do que o reativo.

Esta abordagem com agente reativo foi a que conseguiu-se implementar com mais sucesso (mais pontos).

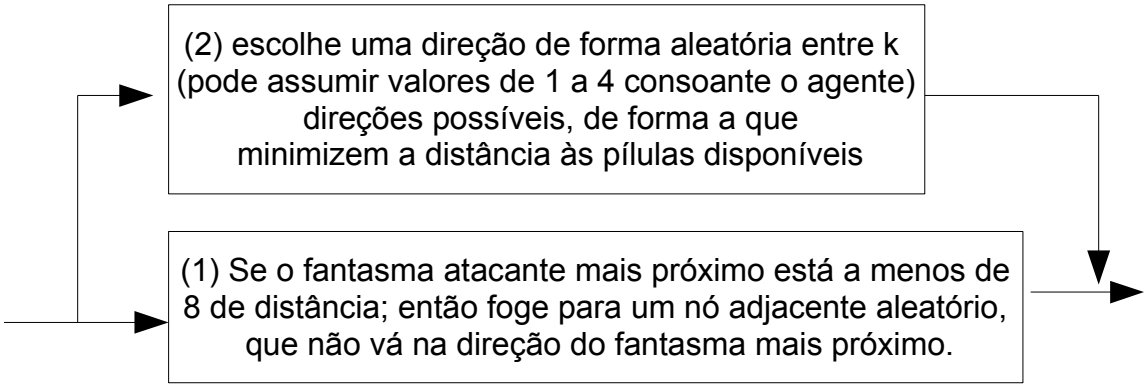

Figura 27: Agentes reativos semi-probabilísticos

Os agentes reativos puros usados têm apenas 2 regras.

A regra com mais prioridade evita um fantasma perigoso que esteja o mais próximo possível, fugindo nas outras direções de forma aleatória. A regra com menos prioridade é responsável pela captura das pílulas.

O valor de k permite a implementação de 4 agentes. Cada agente tem um valor de k especifico. Os valores de k variam de 1 a 4 incluídos. As direções com que o agente pode seguir são k direções possíveis.

3.3 Resultados

Estes resultados correspondem aos pontos feitos pelo agentes propostos. Os resultados foram obtidos ao longo de várias execuções, com as 3 vidas restantes no inicio do jogo, contra a equipa de fantasmas Legacy.

pontos obtidos
14350
17270
10920
12730
12800
17420
11880
21000
11950
11180
23760
23560
22550
20100
19480
19930
18090
16120
23980
7700

média	mediana	máximo	mínimo
16838,5	17345	23980	7700

Tabela 14: Resultados do agente reativo com 2 estados

1 alternativa	2 alternativas	3 alternativas	4 alternativas
pontuações	pontuações	pontuações	pontuações
6600	8910	4260	5260
6170	6270	6950	2840
5140	3850	7190	2860
9060	5500	1720	4800
6400	6440	5120	2810
11940	2310	4130	7950
3400	6320	3940	2990
12690	8110	7580	1740
8910	7580	4450	6620
10120	9030	3160	3780
8840	4600	2070	3860
11800	5080	6150	4100
9080	2350	3460	4280
10290	3750	570	4060
10470	6870	5230	3190
4940	5090	4320	3040
8150	5150	3340	5460
12570	4840	3030	2550
5400	8110	5870	2070
11000	5660	1980	5670

Alternativas (k)	média	desvio padrão	mínimo	máximo
1	8648,5	2776,32	3400	12690
2	5791	1946,57	2310	9030
3	4226	1906,99	570	7580
4	3996,5	1578,95	1740	7950

Figura 28: 4 Reativos simples semi-probabilísticos

3.4 Conclusões

O reativo simples (com k=1) é melhor do que as suas variantes semi-probabilísticas. O reativo com 1 estado é superior ao reativo simples e aos reativos semi-probabilísticos.

Esta abordagem não é a suficiente para ganhar a competição, porque existem implementações com muito melhores resultados. Esse facto demonstra as limitações deste agente reativo. O estado de arte demonstra que agentes deliberativos conseguem melhores resultados do que reativos. As suas vantagens são eficiência computacional e simplicidade comparado com outros algoritmos. As desvantagens são ter só noção do presente e reagir a ele, independentemente do número de regras ou da sua organização. Mesmo com um estado o agente tem limitações que podem ser ultrapassadas com capacidades preditivas e assim evitar situações que conduzem inevitavelmente ao encurralamento do pacman por parte dos fantasmas.

Um reativo melhor poderia ser com regras diferentes conforme o labirinto que o agente está a jogar (há versões do ICE Pambush que consideram essa hipótese na prática, mesmo não sendo um agente reativo). Cada nível tem zonas com mais possibilidades de os fantasmas apanharem o pacman do que outras conforme [36] demonstra. Assim poderia-se considerar distâncias diferentes de segurança em relação aos fantasmas conforme a posição do pacman no labirinto e com isso diferentes pontos de fuga caso o pacman esteja ameaçado.

Se não está cercado, o nó adjacente pertence ao nó do beco e estão os 2 fantasmas mais próximos a uma distância menor do que a necessária à entrada e fuga desse beco; então não vai para o nó adjacente que faça parte desse beco. Esta regra é um exemplo que variaria de labirinto para labirinto, após uma análise a cada um dos problemas que as paredes dos labirintos fornecem. Infelizmente essa abordagem daria muito trabalho, de forma a obter a melhor solução possível, tendo em conta cada beco.

Apesar das hipóteses 4º e 5º (que estiveram na origem da hierarquia de Brooks) é possível incorporar estratégias preditivas, tendo por base uma arquitetura que seja eficiente.

O agente sucessor deste irá procurar reduzir estas desvantagens.

4 Agente composto por uma hierarquia de Brooks com capacidades preditivas adaptativas

4.1 Introdução

O melhoramento face ao reativo é a inserção de previsões nos movimentos e uma habilidade para usar essas previsões. Nesta introdução é feita a definição dos conceitos principais, o objetivo e o tema que cada subsecção aborda.

O controlador adaptativo baseia-se numa identificação da dinâmica, com o método dos mínimos quadrados recursivos, refazendo-se o cálculo dos ganhos repetidamente dos fantasmas, em tempo real.

O controlo adaptativo é composto por um identificador e pelo projeto do controlador. O **Identificador** estima continuamente os parâmetros de um modelo, a partir dos dados de entrada e saída medidos. O **Projeto do Controlador**, que recalcula continuamente o movimento do pacman, tendo em conta as novas estimativas do modelo.

O algoritmo RLS obtém estimativas de mínimos quadrados combinando uma estimativa anterior com novos dados, aproveitando cálculos já efetuados anteriormente. Neste trabalho, o RLS faz parte do identificador. O projeto do controlador faz uso das previsões com base no modelo.

O modelo linear é apresentado primeiro, depois o do respetivo RLS que ajusta os pesos das variáveis, depois os modelos lineares que são possíveis para os fantasmas e onde incidem e finalmente como as previsões acabam por ser usados numa hierarquia de Brooks.

4.2 Modelo linear

Esta secção começa com a definição de modelo, seguido de uma definição de modelo linear e depois é apresentada a justificação da sua utilização.

Um modelo matemático é uma representação ou interpretação simplificada da realidade, ou uma interpretação de um fragmento de um sistema, segundo uma estrutura de conceitos mentais ou experimentais.

O método de cálculo do modelo linear é representado como sendo:

$y = a_0 + a_1 x_1 + ... + a_{m-1} x_{m-1} + e$ (1), em que: 'y' é a saída; $x_1, ..., x_{m-1}$ são variáveis que irão fazer parte de um vetor de dados observáveis $\phi = [1, x_1, .., x_{m-1}]$; $\Omega = [a_0, .., a_{m-1}]$ de comprimento 'm' são os parâmetros constantes do modelo, que neste caso irão ser ajustados pelo algoritmo RLS; 'm-1' é a ordem do modelo e 'e' é um erro.

O modelo linear é apenas um dos muitos modelos possíveis que podem ser usadas para explicar a resposta em termos das variáveis explicativas (ver [22]). As vantagens são várias em relação aos outros modelos [22]: é mais simples, melhor entendido, mais fácil de interpretar do que a maioria dos outros modelos e a formulação do modelo linear é útil em certos modelos não-lineares.

4.3 RLS

Nesta secção começa-se com o motivo que leva à utilização do RLS e onde deve ser usado no Ms pacman; depois quais as variáveis do modelo linear usadas pelo RLS; seguido de um método que cálculo que é melhorado e finalmente as velocidades dos algoritmos, quer convergência, quer número de passos.

A utilização do modelo linear levanta um problema. O problema existe devido a situações em que a dinâmica do sistema a controlar pode-se alterar ao longo do tempo e não é possível conhecer à priori qual o controlador a utilizar numa dada situação (como é o caso do movimento dos fantasmas). Esse comportamento dinâmico implica a necessidade de uma adaptação constante. O papel do RLS é a adaptação à dinâmica de comportamento dos fantasmas.

Quanto às pílulas não são abrangidas por esta situação, por serem estáticas e desaparecem com a passagem do pacman.

O RLS requer um modelo linear nos parâmetros. Um modelo linear nos parâmetros pode ser escrito da seguinte forma: $\hat{y}[k]=\phi'[k]\Omega$ (2); onde $\hat{y}[k]$ é um escalar com a saída estimada do modelo no instante k, $\phi[k]$ é um vetor de dados observáveis (tais como funções das saídas e controlos anteriores), e Ω é o vetor de parâmetros a estimar.

$$\Omega[k]=\Omega[k-1]+K[k](y[k]-\Omega[k-1]\phi[k-1]) \quad (3)$$

$$K[k]=P[k]\phi[k-1] \quad (4)$$

$$P[k]=(P[k-1]-\frac{(P[k-1]\phi[k-1]\phi'[k-1]P[k-1])}{(\lambda+\phi'[k-1]P[k-1]\phi[k-1])}) \quad (5)$$

As fórmulas (3), (4) e (5) formam ao todo o método de cálculo, no instante k, do algoritmo RLS, antecessor à implementação com base no algoritmo de Bierman que tem mais robustez numérica, mas é computacionalmente mais lento.

A variável K[k] é um vetor de comprimento 'm', com o ganho de Kalman no instante k, que multiplica pelo erro ($y[k]-\Omega'[k-1]\phi[k-1]$ que é a diferença entre o valor esperado/estimado com o obtido pela leitura do sistema a controlar) presente na fórmula (3).

A variável λ é um valor escalar, que representa o fator de esquecimento e pertence ao intervalo]0,1[. O fator de esquecimento serve para evitar o adormecimento dos mínimos quadrados recursivos. O adormecimento dos mínimos quadrados recursivos ($\lambda=1$) é um fenómeno em que as estimativas tendem a tornar-se constantes e caso o sistema a identificar sofra alguma alteração,

será necessário muito tempo para que as estimativas aproximem-se do valor novo. Ao invés, um valor λ demasiado pequeno cria falta de precisão. A memória assimptótica relaciona-se com o lambda através da formula: $N = \frac{1}{(1-\lambda)}$.

A variável P[k] é a matriz de co-variância, com dimensão 'm' por 'm', no instante k, que devido a questões de robustez numérica foi fatorizada. Essa fatorização relaciona P com os fatores U e D da seguinte forma: $P = UDU^T$. A matriz U é triangular superior com 1's na diagonal e a matriz D é diagonal, que são usados no algoritmo de Bierman em vez de P.

O algoritmo de Bierman (que é teoricamente equivalente ao de Peterka[23]) é responsável pelo cálculo do ganho de Kalman (K[k+1]), da matriz U[k+1] e da matriz D[k+1], num dado instante k; a partir das entradas $\phi[k]$, U[k], D[k] e λ num dado instante k. O tempo computacional requer $O(n^2)$ multiplicações e 2n+1 divisões [23].

A sequencia de cálculos começa com o algoritmo de Bierman, depois é feito os cálculos em (3), depois atualiza-se o Ω e por fim o ϕ é atualizado.

4.4 Modelo linear no contexto pacman

O meu agente utiliza um modelo linear para fazer previsões, em vez de ser uma regra semi-probabilística. O meu modelo linear é adaptado em tempo real, em vez de ser totalmente descoberto antes do jogo ser jogado pelo agente final como o modelo utilizado por Bruce. O meu modelo linear só devolve um valor possível para um determinado número de passos e para a mesma situações de jogo; enquanto este agente proposto por Bruce pode devolver vários valores possíveis, consoante as probabilidades indicadas pelo modelo.

A formula (1) do capítulo 4.1 deve ser utilizada de forma a adaptar-se ao problema em causa. O problema em causa é a previsão de movimentos dos fantasmas atacantes. À partida o problema em causa possibilita 2 abordagens possíveis para os modelos.

A primeira abordagem descreve-se da seguinte forma:

$$A\ influência\ do\ fantasma\ a\ prever = [\sum_{(l=0)}^{(L_1-1)} b_{i,2l} G_x(i,k-l) + b_{i,2l+1} G_y(i,k-1)] \quad (8)$$

$$A\ influência\ do\ pacman = [\sum_{l=0}^{(L_2-1)} c_{2l} P_x(k-l) + c_{2l+1} P_y(k-l)] \quad (9)$$

$$A\ influência\ dos\ outros\ fantasmas = [\sum_{(j\neq i)} (\sum_{(l=0)}^{(L_3-1)} d_{i,2l} G_x(j,k-l) + d_{i,2l+1} G_y(j,k-l))] \quad (10)$$

$$G_x(i,k+1) = a_{i,0,x} \quad (11)$$
$$+influência\ do\ fantasma\ a\ prever$$
$$+influência\ do\ pacman$$
$$+influência\ dos\ outros\ fantasmas$$

$$G_y(i,k+1) = a_{i,0,y} \quad (12)$$
$$+influência\ do\ fantasma\ a\ prever$$
$$+influência\ do\ pacman$$
$$+influência\ dos\ outros\ fantasmas$$

A segunda abordagem descreve-se da seguinte forma:

$$A\ influência\ do\ fantasma\ a\ prever = [\sum_{(l=0)}^{(L_1-1)} b_{i,2l} G_x(i,k-l) + b_{i,2l+1} G_y(i,k-1)] \quad (13)$$

$$A\ influência\ do\ pacman = [\sum_{l=0}^{(L_2-1)} c_{2l} P_x(k-l) + c_{2l+1} P_y(k-l)] \quad (14)$$

$$\textit{A influência dos outros fantasmas} = [\sum_{(j \neq i)} (\sum_{(l=0)}^{(L_3-1)} d_{i,2l} G_x(j, k-l) + d_{i,2l+1} G_y(j, k-l))] \quad (15)$$

$$D(i, k+1) = a_{i,0} + \textit{influência do fantasma a prever} + \textit{influência do pacman} + \textit{influência dos outros fantasmas} \quad (16)$$

Ambas as formulas das 2 abordagens são influenciadas pelas coordenadas x e y do fantasma a prever, do pacman e dos outros fantasmas em jogo. A variável 'k' é o instante de tempo discreto no momento atual. As variáveis $a_{i,0,x}, a_{i,0,y}, a_{i,0}, b_{i,l}, c_{i,l}, d_{i,l}$ são as constantes do modelo, anteriormente descritas como Ω . A variável $G_x(i, k)$ é a posição no eixo dos xx, do fantasma 'i', no instante 'k'. A variável $G_y(i, k)$ é a posição no eixo dos yy, do fantasma i, no instante k. A variável $P_x(k)$ é a posição do pacman nos eixos dos xx, no instante k. A variável $P_y(k)$ é a posição do pacman no eixo dos yy, no instante k.

A primeira abordagem contém 2 modelos distintos: um para o eixo dos xxs e outro para o eixo dos yys. O resultado dos 2 modelos é um ponto bidimensional $(G_x(i, k+1), G_y(i, k+1))$ que indica a próxima posição do fantasma 'i'.

A segunda abordagem contém só um modelo linear. O resultado do modelo linear é a próxima direção $D(i, k+1) \in \{0,1,2,3\}$ a seguir pelo fantasma 'i'. Após este cálculo com o modelo obtêm-se $(G_x(i, k+1), G_y(i, k+1))$ em função de $D(i, k+1)$ e $(G_x(i, k), G_y(i, k))$.

Pretende-se com estes modelos definir com a maior exatidão possível as posições $(G_x(i, k+1), G_y(i, k+1)), (G_x(i, k+2), G_y(i, k+2)), \ldots, (G_x(i, k_{max}), G_y(i, k_{max}))$, em que k_{max} é o instante de tempo discreto máximo, com que o modelo será capaz de predizer e que maximize os pontos.

As paredes do labirinto afetam as previsões de 2 formas: posições anteriores do pacman e dos fantasmas e previsões inválidas. As previsões inválidas são previsões que indicam que as próximas posições do fantasma atacante irão penetrar as paredes do labirinto ou uma direção que não existe. As previsões inválidas são substituídas por outras na direção do pacman ou arredondadas, que não coincidem com a última prevista anteriormente e que não entram pelas paredes dentro. Caso a previsão inválida seja respetiva a um passo à frente, ela nunca vai coincidir com a posição atual do fantasma. Todas as correções passam a fazer parte do modelo.

Pretende-se após as simulações com diferentes abordagens dos modelos descritos, concluir qual deles é o mais apropriado para prever os movimentos dos fantasmas.

4.5 Utilização e gestão das previsões com base no modelo

O objetivo é substituir o algoritmo minimax, através de uma hierarquia de Brooks com capacidades preditivas adaptativas. Se é sabido qual o movimento dos fantasmas é escusado gerar todos os movimentos possíveis para os fantasmas. O movimento dos fantasmas é dado pelo modelo linear. Além do modelo linear adaptar-se ao movimento dos fantasmas através do RLS, o número de regras necessárias adapta-se ao labirinto e à posição do pacman no labirinto.

A hierarquia começa por testar os caminhos que aproximam-se mais rapidamente das pílulas e depois conforme sobe-se na hierarquia os caminhos vão aproximando-se menos das pílulas. Uma vez encontrado o caminho melhor, ou seja, o caminho que minimize a distância às pílulas e que permita segurança face aos fantasmas através de uma distância de segurança, o pacman seguirá esse caminho. A distância de segurança serve para o pacman não ser comido pelo fantasma mais próximo nesse instante de tempo e é igual a 8.

As previsões com horizontes mais pequenos têm mais prioridade do que as previsões com horizontes maiores. Caso haja um problema num horizonte mais pequeno, a hierarquia segue uma regra que indique outro caminho em vez de continuar a avaliar o mesmo caminho mas com um horizonte maior. Uma vez avaliada uma parte do caminho não é preciso avaliar essa parte de novo.

Uma comparação é uma operação muito rápida em computação, a hierarquia de Brooks é computacionalmente eficiente face a outras hierarquias e a hierarquia usada dá prioridade aos caminhos que mais provavelmente atingem as pílulas mais depressa. O facto de a procura ser DFS com profundidade máxima igual ao horizonte máximo H, torna o programa eficiente com a quantidade de memória usada.

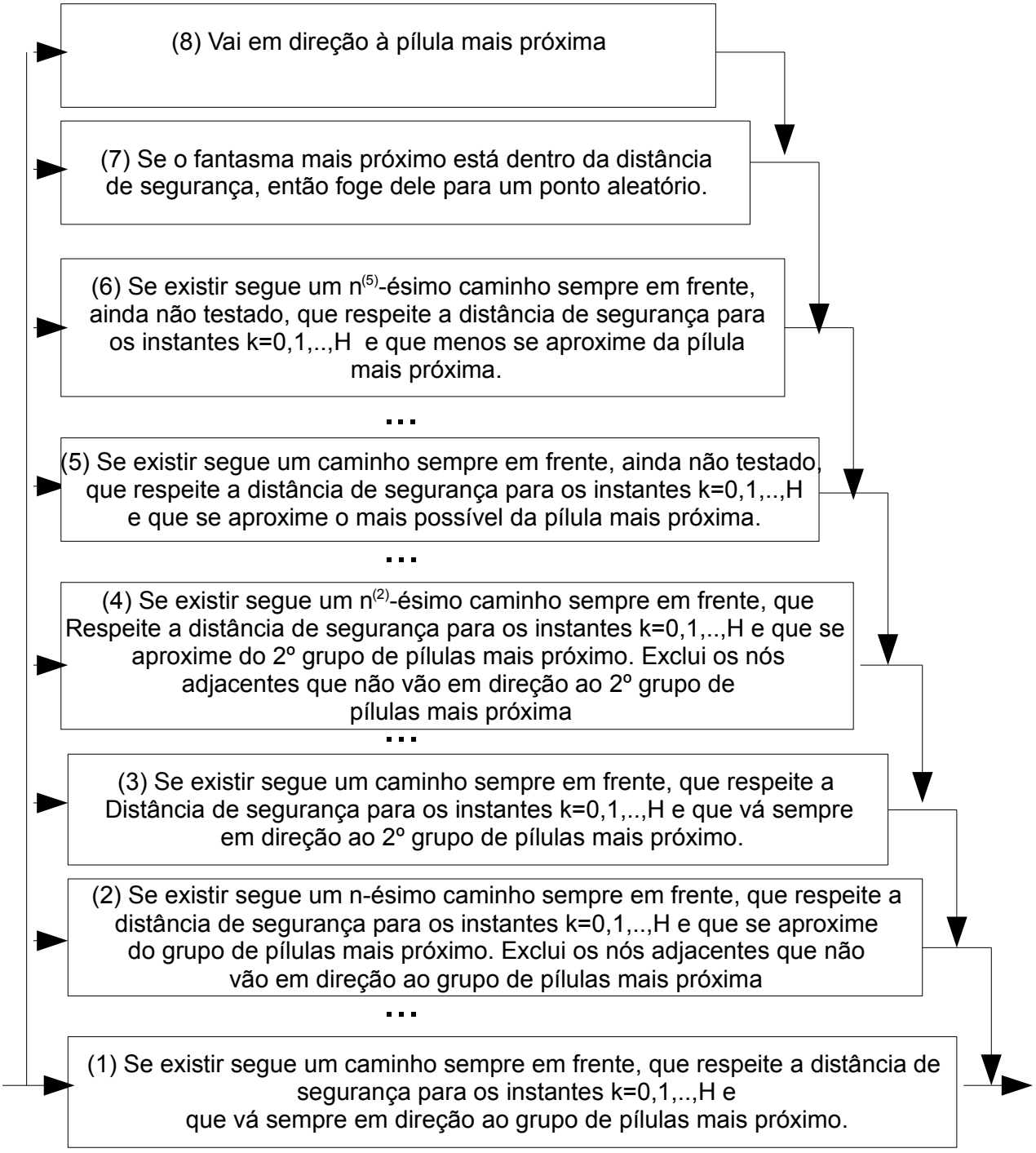

Figura 29: Preditivo evoluído do reativo semi-probabilístico

Um grupo de pílulas (que está representado na hierarquia) é uma zona do labirinto com pílulas para um mesmo nó adjacente ao pacman que mais minimiza a distância a essas pílulas. Os caminhos são conjuntos de nós desde a posição atual do pacman até a um horizonte H. Nenhum caminho na hierarquia faz inversão de marcha, ou seja, o nó anterior nunca é o mesmo que o nó seguinte. O pacman não tem necessariamente de seguir o caminho escolhido num instante de tempo do principio ao fim, em vez disso move-se para o segundo nó do caminho escolhido a cada instante de tempo.

As regras que vão desde (1) a (6) utilizam as previsões dos modelos, testam caminhos e não existem na versão reativa do respetivo agente proposto. As regras (7) e (8) são as regras de uma versão reativa.

A regra (1) verifica o que acontece se o pacman for em direção à pílula mais próxima pelo caminho mais curto. As regras que vão de (1) a (2) testam vários caminhos alternativos à regra (1) e tentam aproximar-se do grupo de pílulas mais próximo. As regras que vão de (3) a (4) testam vários caminhos alternativos às anteriores, mas para o segundo grupo de pílulas mais próximo. O mesmo raciocínio repete-se com ordem decrescente de prioridade para o terceiro grupo de pílulas mais próximo e para o quarto grupo de pílulas mais próximo, até acabar de verificar todos os nós adjacentes ao pacman ou deixar de haver mais grupos de pílulas. As regras de (5) a (6) testam os outros caminhos possíveis, por ordem crescente em relação à distância à pílula mais próxima. A regra (6) testa o caminho que mais se afasta da pílula mais próxima e tem menos prioridade que as regras mais abaixo.

Este agente tem semelhanças e diferenças com o agente SmartDijkstraPac [3]. As semelhanças em ambos são: o presente é uma prioridade, têm uma versão reativa, projetam previsões sobre o labirinto e a versão preditiva reage a fantasmas que estejam mais longe do que a distância de segurança usada na versão reativa. O meu agente é diferente: na hierarquia usada, no modelo linear que faz as previsões que só indicam uma e só uma direção a cada instante de tempo futuro e as regras são estendidas como se fosse um minimax (em vez de serem só 2 regras: uma para o presente e outra para o futuro).

5 Resultados

Os resultados têm a notação seguinte: model_x_y_z acompanhados de lambda que é o fator de esquecimento. O x é o número de posições anteriores do fantasma a prever. O y é o número de posições anteriores do pacman. O z é o número que representa z posições de cada um dos fantasmas. Se z=3 e o modelo prevê o fantasma Blinky; então 3 posições do fantasma Pinky são consideradas, 3 posições do Inky serão consideradas e 3 posições do Sue são consideradas. O número de vidas utilizado foi 3. As execuções dos fantasma quando estão comestíveis não fazem parte dos resultados. O método action é o método que devolve a próxima direção do pacman. A equipa de fantasmas é a equipa Legacy (à exceção no ambiente determinístico, em que o Inky passa a seguir o nó que minimiza a distância do caminho mais curto até ao pacman) e o agente reativo utilizado tem por base o reativo mais simples possível.

Os resultados apresentados estão divididos em 4 sub-capítulos.

Será apresento em primeiro lugar os erros quadráticos do modelo linear, cujo o significado pode fazer sentido para o algoritmo de Bierman, mas não para um caso mais especifico como o jogo pacman. Será dada na mesma secção, informação, que permitiu escolher o tipo de previsão dos modelos.

O próximo passo é usar modelos adequados ao jogo pacman. Nesse caso é preciso fazer correções que afetam os erros apresentados. Esses resultados permitem decidir os parâmetros x, y, z e lambda dos modelos lineares usados para modelar os comportamentos dos fantasmas.

Como o modelo mais preciso possível é dado pelo próprio código dos fantasmas acessível ao programador, esse código é usado para comparar com os modelos lineares usados.

Serão consideradas nas várias secções ambientes determinísticos sempre que necessário.

Finalmente é exibida as pontuações obtidas de vários modelos estudados.

5.1 Erros do modelo linear puro

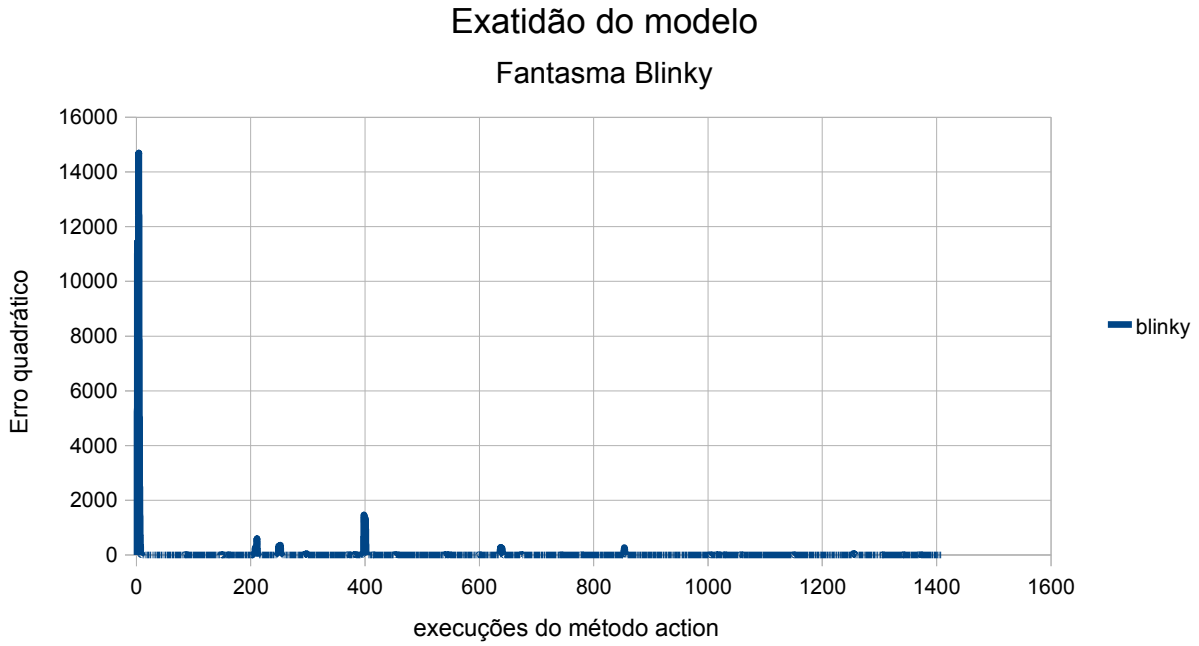

Figura 30: Exemplo dos erros quadráticos de um modelo linear puro. Neste modelo excluem-se as correções. As correções são respetivas às previsões das direções que os fantasmas não conseguem fazer no jogo. O modelo usado é o model_5_6_0 com lambda=0.975.

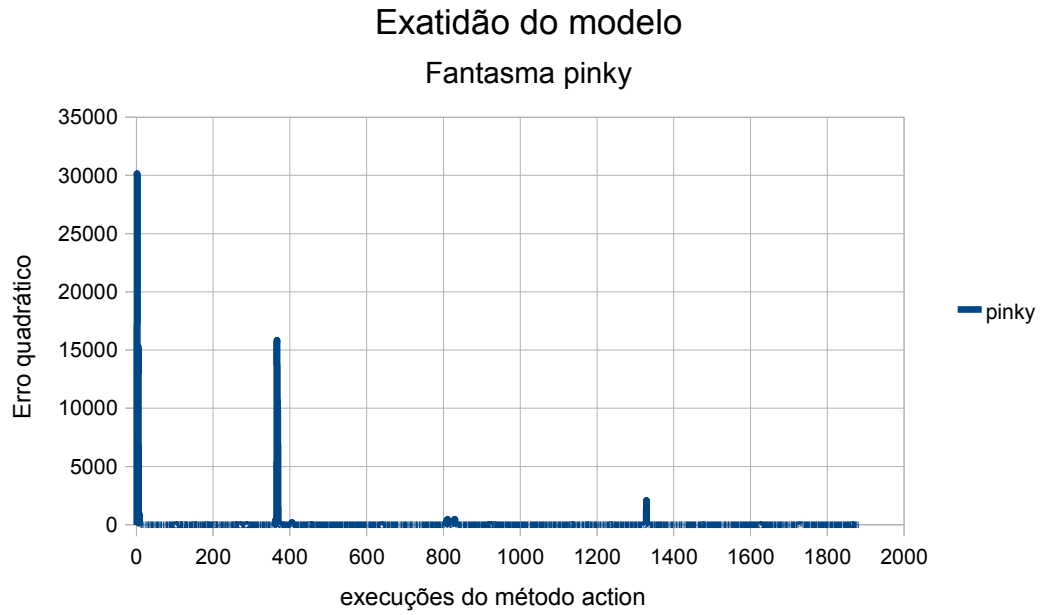

Figura 31: Exemplo dos erros quadráticos de um modelo linear puro. Neste modelo excluem-se as correções. As correções são respetivas às previsões das direções que os fantasmas não conseguem fazer no jogo. Previsão da próxima direção para um passo à frente com model_5_6_0 e lambda=0.975

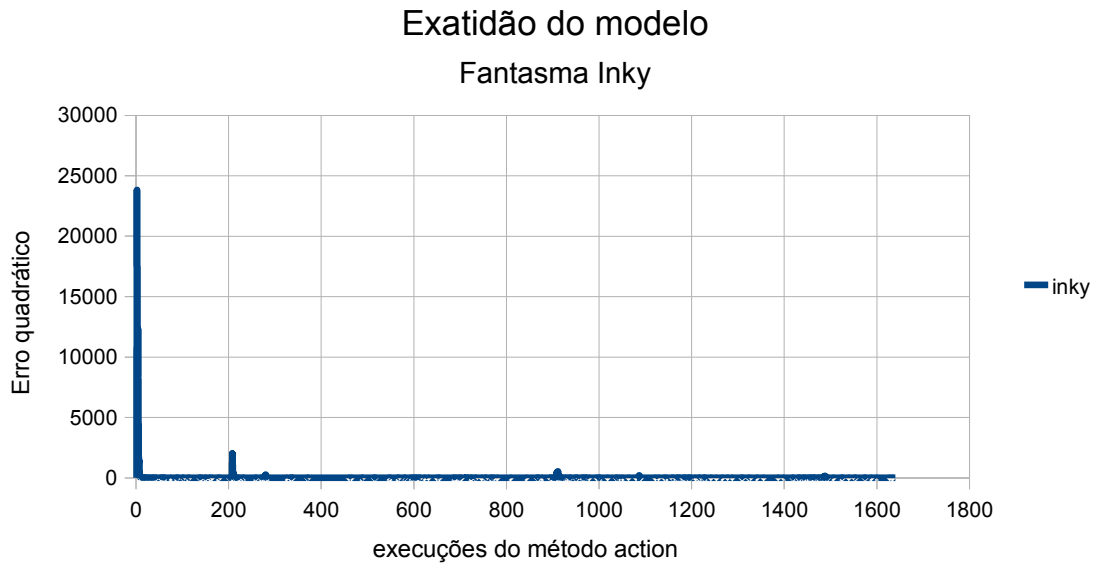

Figura 32: Exemplo dos erros quadráticos de um modelo linear puro. Neste modelo excluem-se as correções. As correções são respetivas às previsões das direções que os fantasmas não conseguem fazer no jogo. Previsão da próxima direção para um passo à frente com model_5_6_0 e lambda=0.975

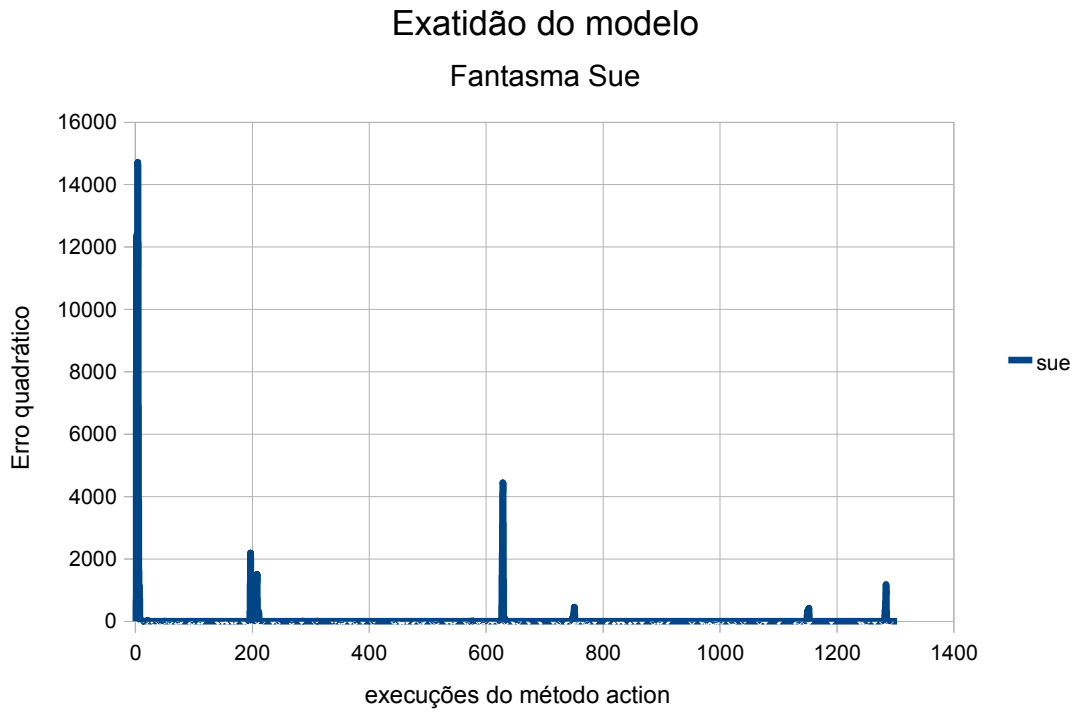

Figura 33: Exemplo dos erros quadráticos de um modelo linear puro. Neste modelo excluem-se as correções. As correções são respetivas às previsões das direções que os fantasmas não conseguem fazer no jogo. Previsão da próxima direção para um passo à frente com model_5_6_0 e lambda=0.975

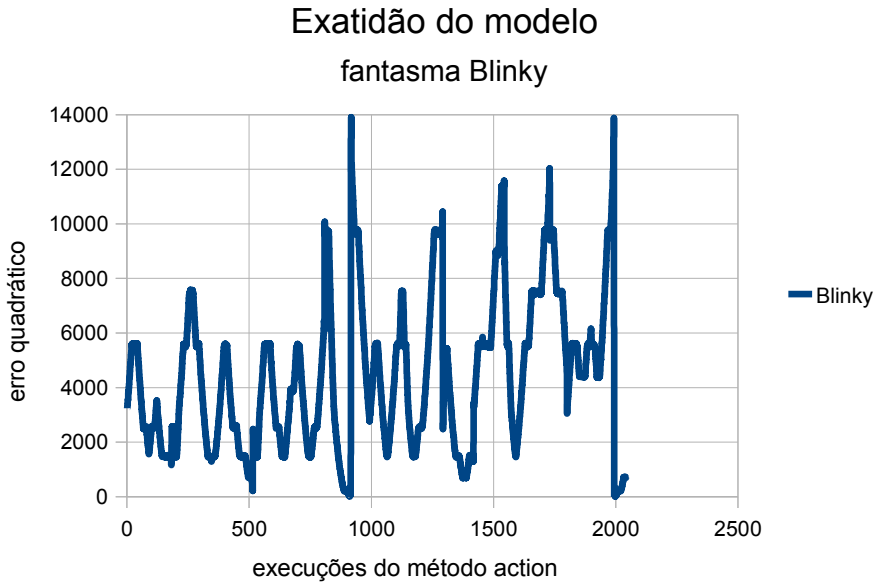

Figura 34: Previsão da próxima abcissa para um passo à frente com model_7_8_1 e lambda=0.995. O modelo tem uma grande dificuldade em adaptar-se, o mesmo acontece com os outros fantasmas.

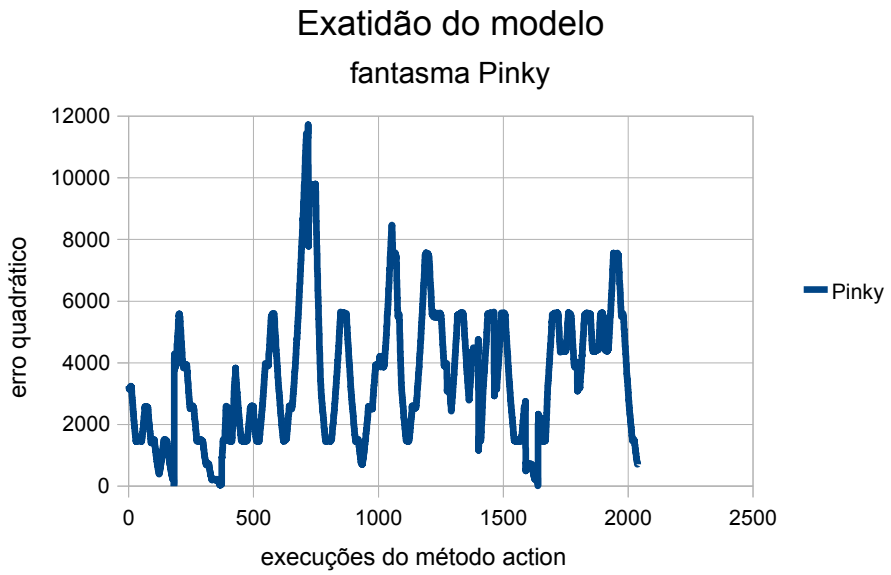

Figura 35: Previsão da próxima abcissa para um passo à frente com model_7_8_1 e lambda=0.995

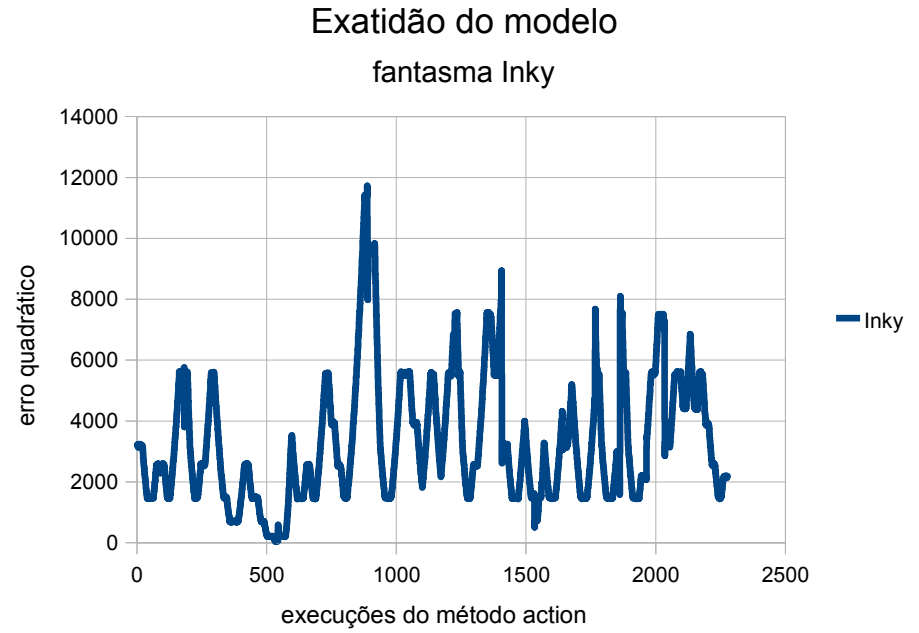

Figura 36: Previsão da próxima abcissa para um passo à frente com model_7_8_1 e lambda=0.995.

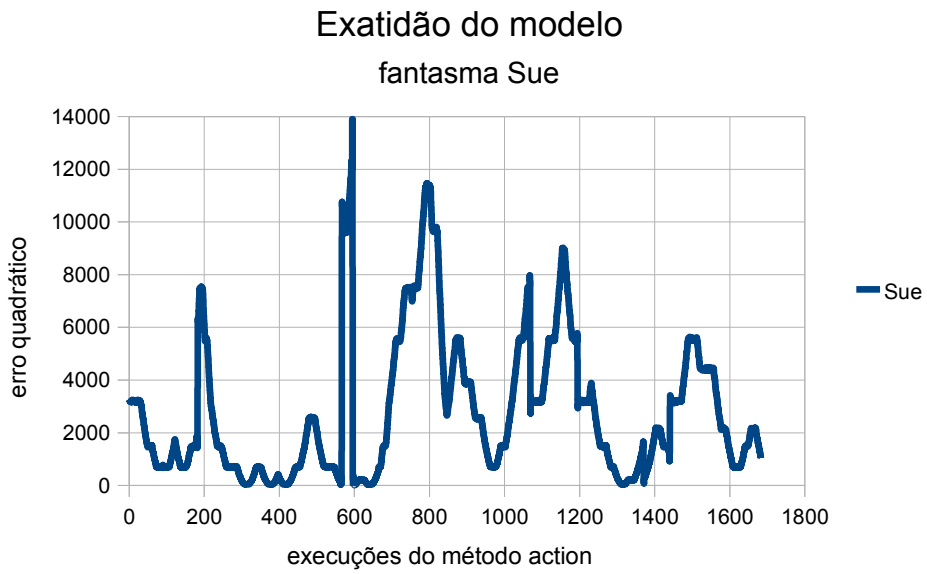

Figura 37: Previsão da próxima abcissa para um passo à frente com model_7_8_1 e lambda=0.995

5.2 Erros do modelo linear com correções

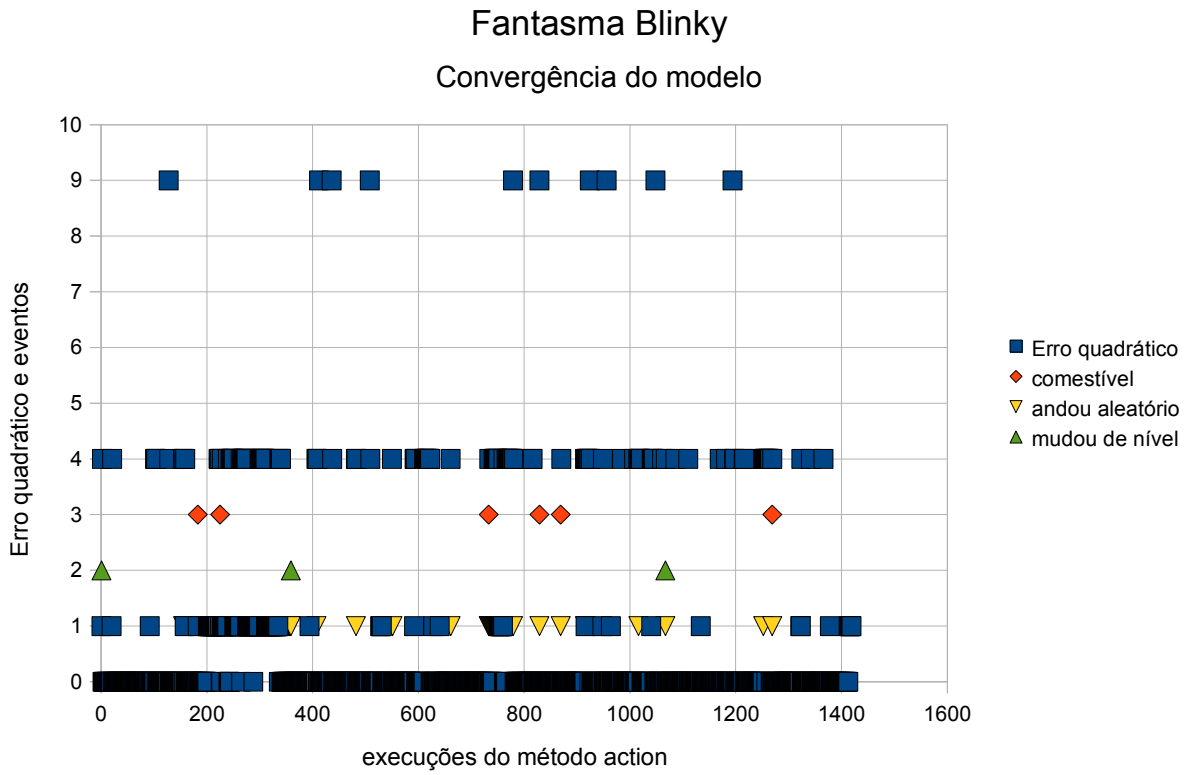

Figura 38: Previsão da próxima direção para um passo à frente com model_1_1_0 e lambda=0.9. O ambiente é não determinístico.

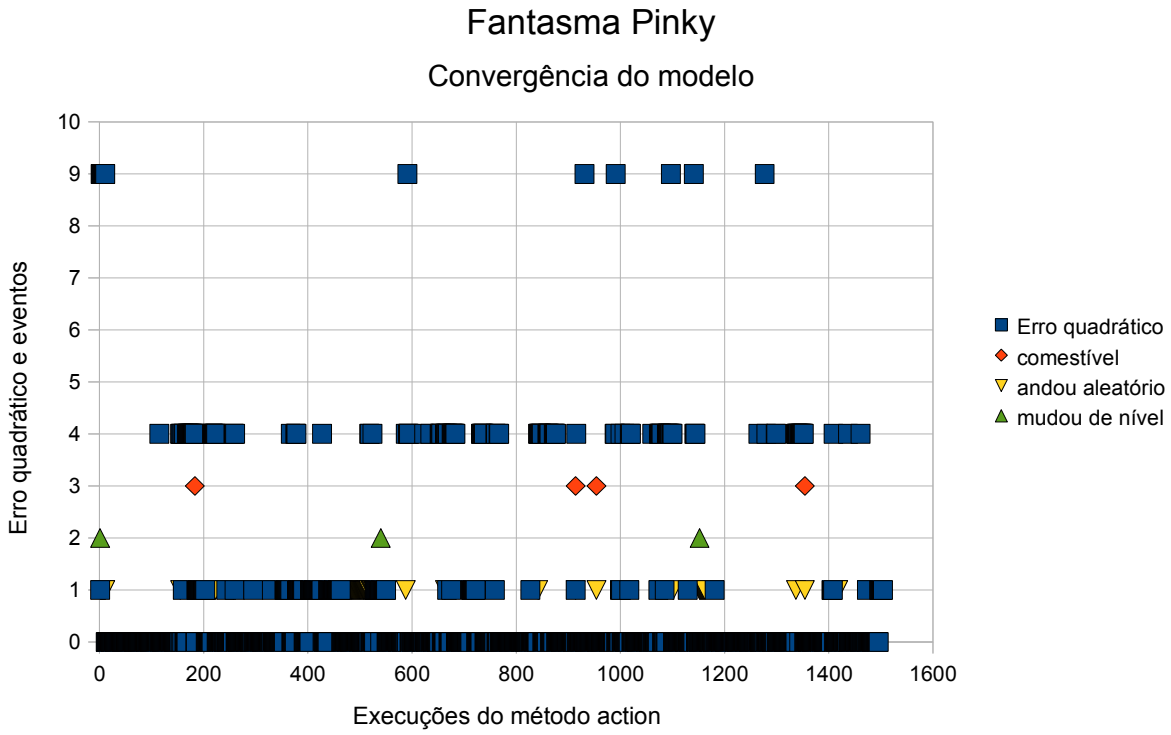

Figura 39: Previsão da próxima direção para um passo à frente com model_1_1_0 e lambda=0.9.O ambiente é não determinístico.

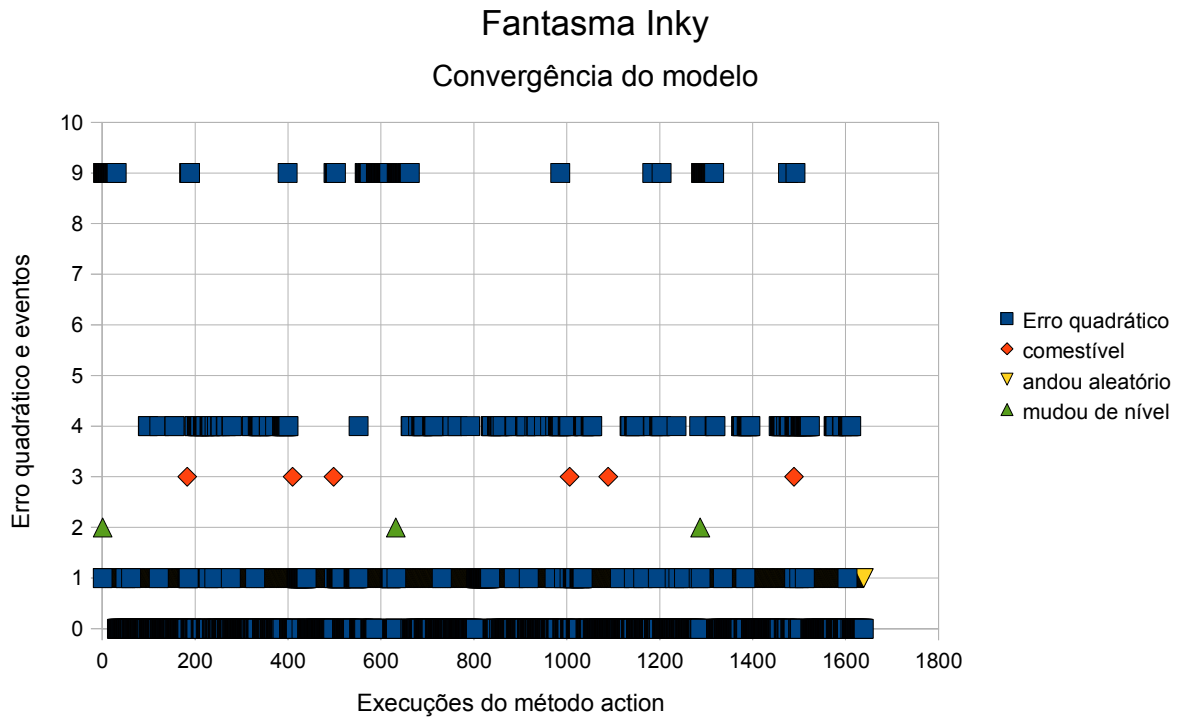

Figura 40: Previsão da próxima direção para um passo à frente com model_1_1_0 e lambda=0.9. O ambiente é não determinístico.

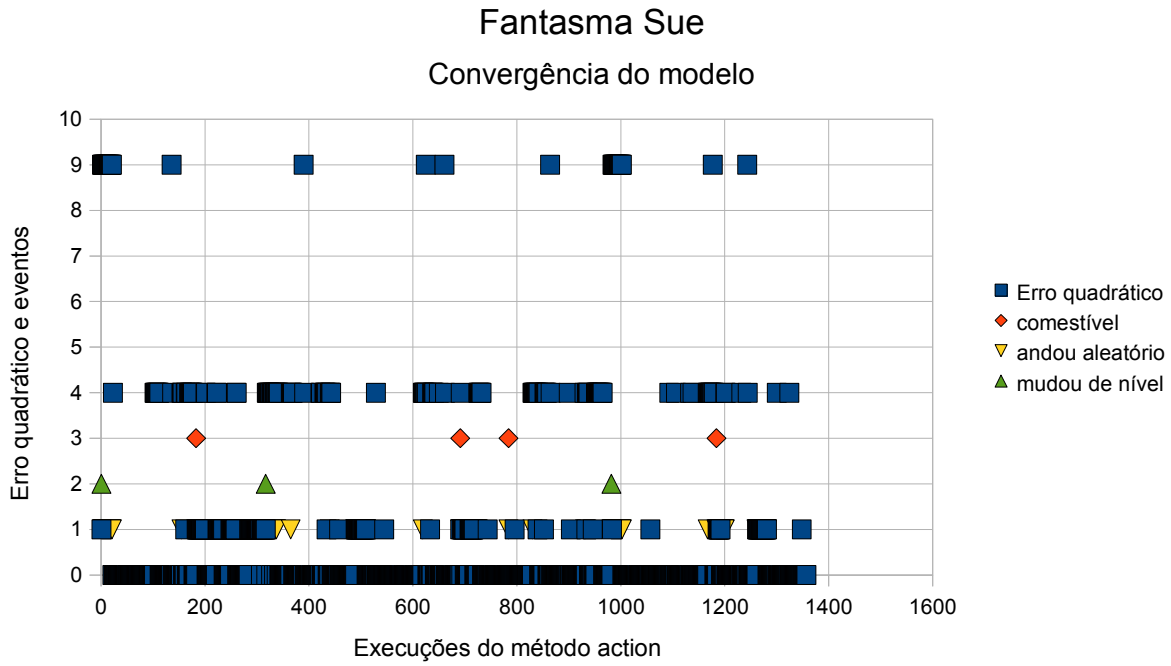

Figura 41: Previsão da próxima direção para um passo à frente com model_1_1_0 e lambda=0.9. O ambiente é não determinístico.

y	MSE (todos)	MSE (sem Inky)
2	1,12874	0,97714
3	0,89651	0,71283
4	**0,76307**	**0,54480**
5	1,09698	0,90405
6	0,90375	0,77876

Tabela 16: Média dos erros quadráticos (5 casas decimais), do model_5_y_0, para prever direções, com lambda=0,94, durante 3 simulações. O ambiente é não determinístico.

Lambda	N	MSE (todos)	MSE (sem Inky)
0,920000	12,5	1,22783	1,10432
0,933320	15	1,03333	0,81650
0,940000	**16,67**	**0,76307**	**0,54480**
0,941200	17,01	0,86458	0,77171
0,950000	20	0,79620	0,60666
0,960000	25	0,81759	0,67347

Tabela 15: Média dos erros quadráticos (5 casas decimais), do model_5_4_0, para prever direções, com vários fatores de esquecimento, durante 3 simulações.O ambiente é não determinístico.

x	MSE (todos)	MSE (sem Inky)
3	0,91660	0,73118
4	0,66047	0,65691
5	**0,76307**	**0,54480**
6	0,94604	0,82331
7	0,90625	0,69406

Tabela 17: Média dos erros quadráticos (5 casas decimais), do model_x_4_0, para prever direções, com lambda=0.94 durante 3 simulações.O ambiente é não determinístico.

z	MSE (todos)	MSE (exceto Inky)
0	**0,763073**	**0,544795**
1	0,737908	0,592433
2	0,967006	0,805021
3	0,891489	0,708566

Tabela 18: Média dos erros quadráticos (5 casas decimais), do model_5_4_z, para prever direções, com lambda=0.94, durante 3 simulações. O ambiente é não determinístico.

lambda	MSE (todos)	MSE (exceto Inky)
0,8750	0,822324	0,79881
0,8887	0,764510	0,69347
0,9000	**0,635242**	**0,52508**
0,9091	0,915546	0,85359
0,9250	0,840385	0,69174

Tabela 19: Média dos erros quadráticos (5 casas decimais) do model_1_1_0, para prever direções com lambda variável, durante 3 simulações. O ambiente é não determinístico.

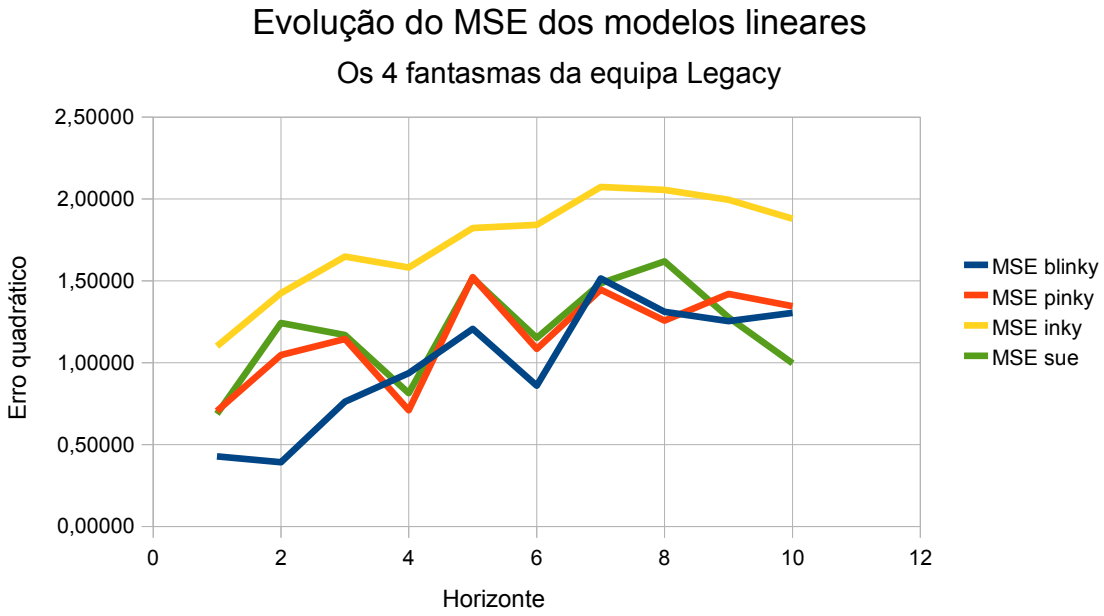

Figura 42: Gráfico que demonstra a capacidade de previsão do model_5_4_0 com lambda=0.94 de 1 até 10 passos à frente. O ambiente é não determinístico.

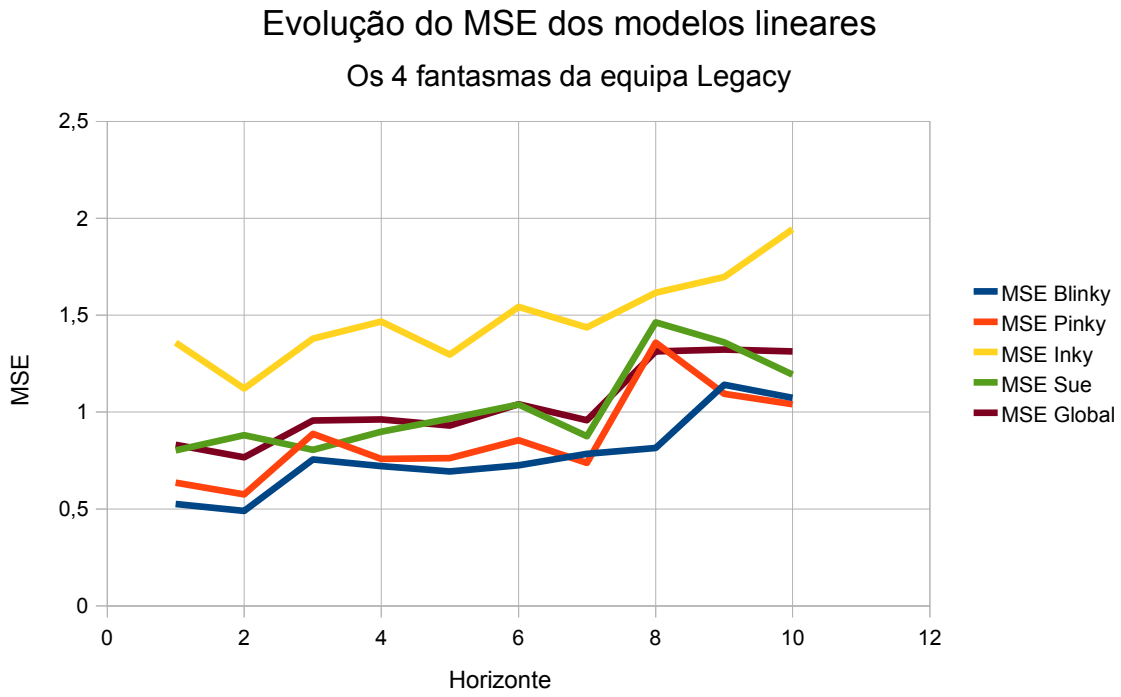

Figura 43: Gráfico que demonstra a capacidade de previsão do model_1_1_0 com lambda=0.9 de 1 até 10 passos à frente. O ambiente é não determinístico.

	MSE
Blinky	0,24780
Pinky	0,39220
Inky	0,52515
Sue	0,73986
Todos	0,47625

Tabela 20: Os erros quadráticos do model_1_1_0 com lambda=0.9, para prever direções, durante 3 simulações, num ambiente determinístico. A diferença do Inky face ao Blinky pode ser a mudança de posições entre os estados ataque e comestível e a mudança de labirinto. Os erros de um modo geral são afetados pelo número de passos necessários para que o algoritmo RLS possa adaptar o modelo linear.

5.3 Erros do modelo com base no código dos fantasmas

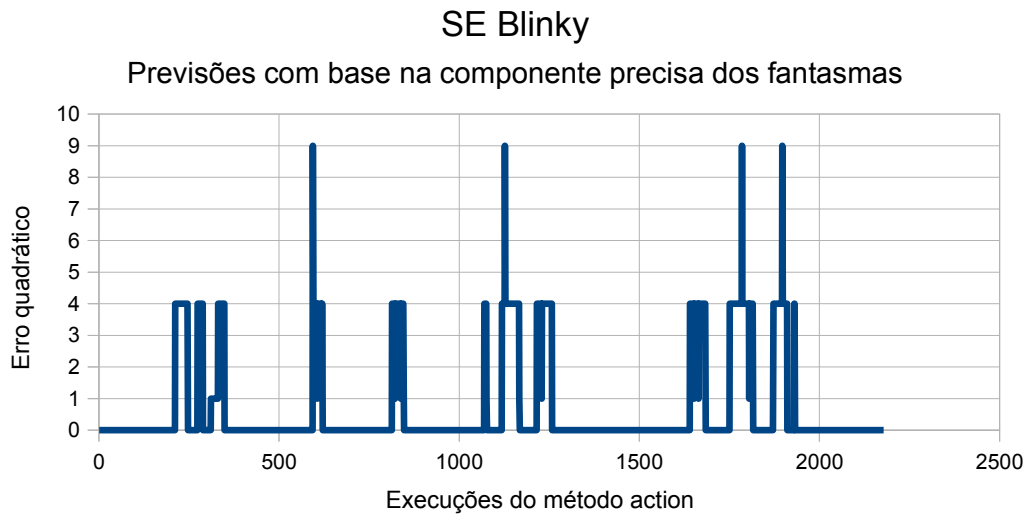

Figura 44: O fantasma Blinky aproxima-se do pacman, através da distância do caminho mais curto, na maior parte dos casos. Noutras situações anda aleatoriamente. O ambiente é não determinístico.

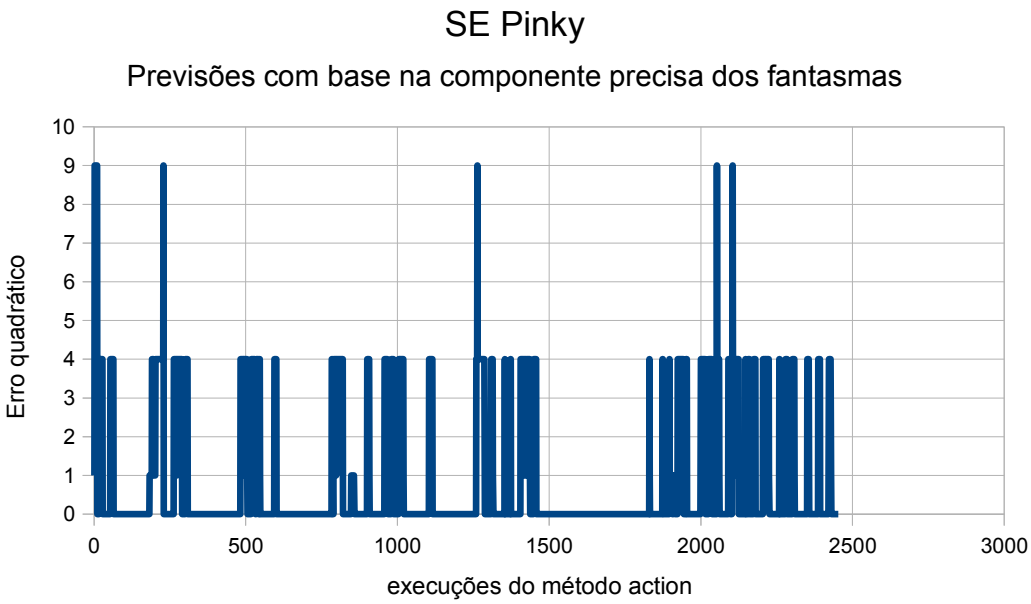

Figura 45: O fantasma Pinky aproxima-se do pacman, através da distância euclidiana, na maior parte dos casos. Noutras situações anda aleatoriamente. O mabiente é não determinístico.

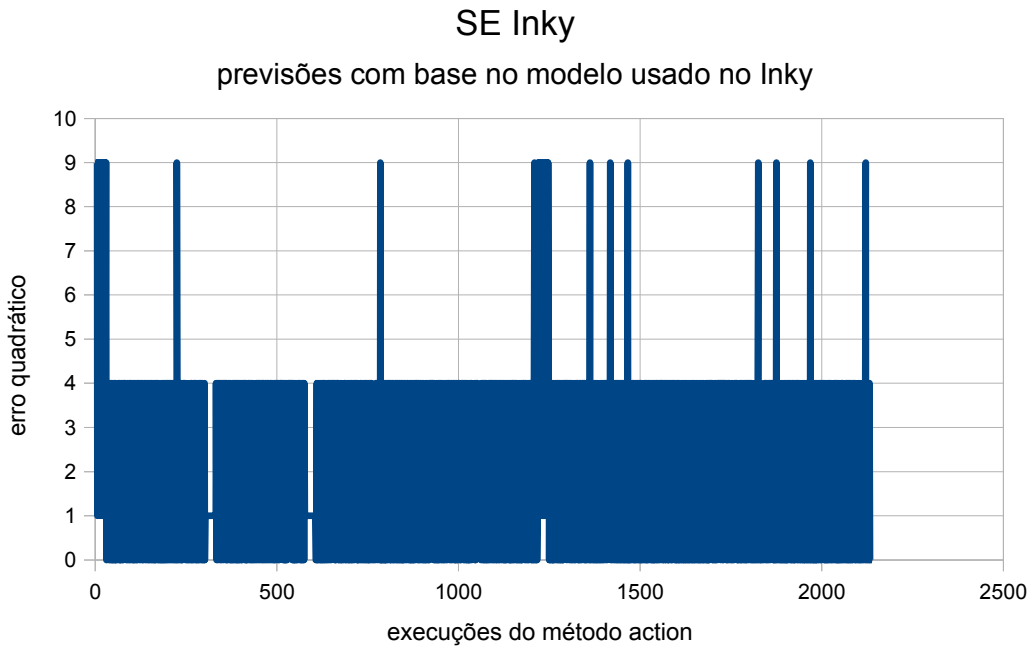

Figura 46: O fantasma Inky escolhe um ponto aleatório para seguir, o que faz que seja muito difícil de prever o seu movimento.

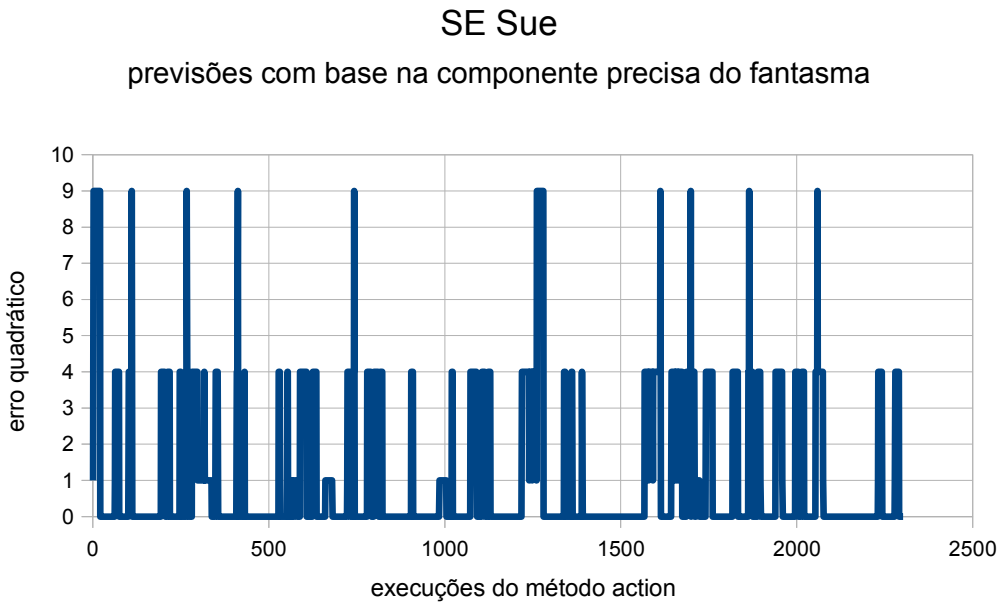

Figura 47: O fantasma Sue aproxima-se do pacman, através da distância de Manhattan, na maior parte dos casos. Noutras situações anda aleatoriamente. O ambiente é não determinístico.

fantasmas	MSE
blinky	0,69192
pinky	0,80962
inky	2,08138
sue	1,03519
todos	1,13835
sem inky	0,84558

fantasma	número de falhas	rácio de falhas
blinky	395	0,1813
pinky	529	0,2156
inky	1165	0,5446
sue	632	0,2744

Tabelas 21: Resultado dos modelos de comportamento preciso dos fantasmas para um passo à frente ao longo de vários execuções do método action numa simulação. O ambiente é não determinístico

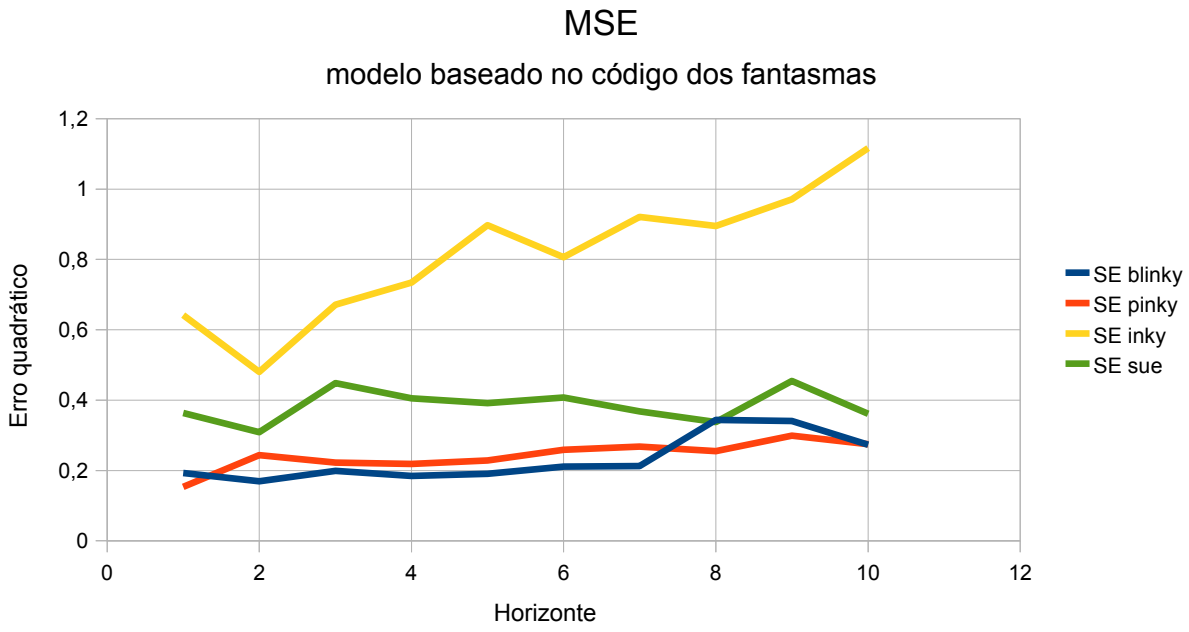

Figura 48:Os valores do gráfico foram obtidos cada um com 3 execuções do programa principal. Os resultados demonstram que existe um aumento ligeiro do erro conforme o horizonte aumenta. O fantasma Inky é o fantasma, cujo o comportamento é mais difícil de prever e é o que faz aumentar mais o erro para horizontes maiores do que os restantes. O ambiente é não determinístico.

5.4 Pontuações de vários modelos

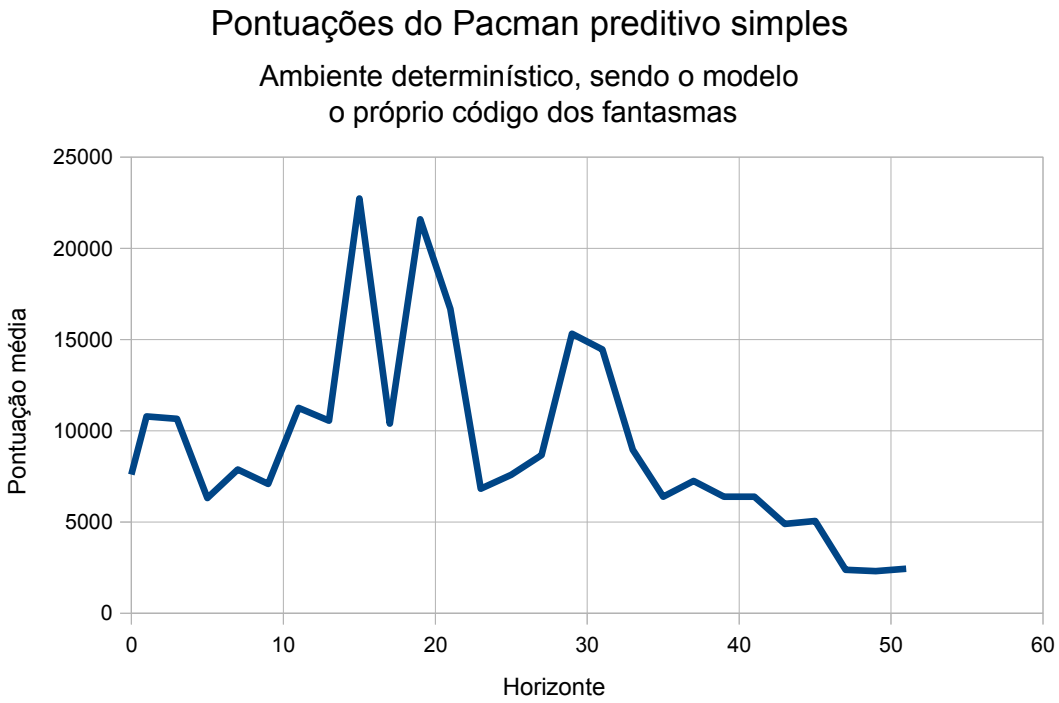

Figura 49: Evolução da pontuação média conforme aumenta-se o horizonte. O Inky passou a seguir o pacman com base no caminho mais curto para que o pacman pudesse trabalhar num ambiente determinístico com os fantasmas atacantes.

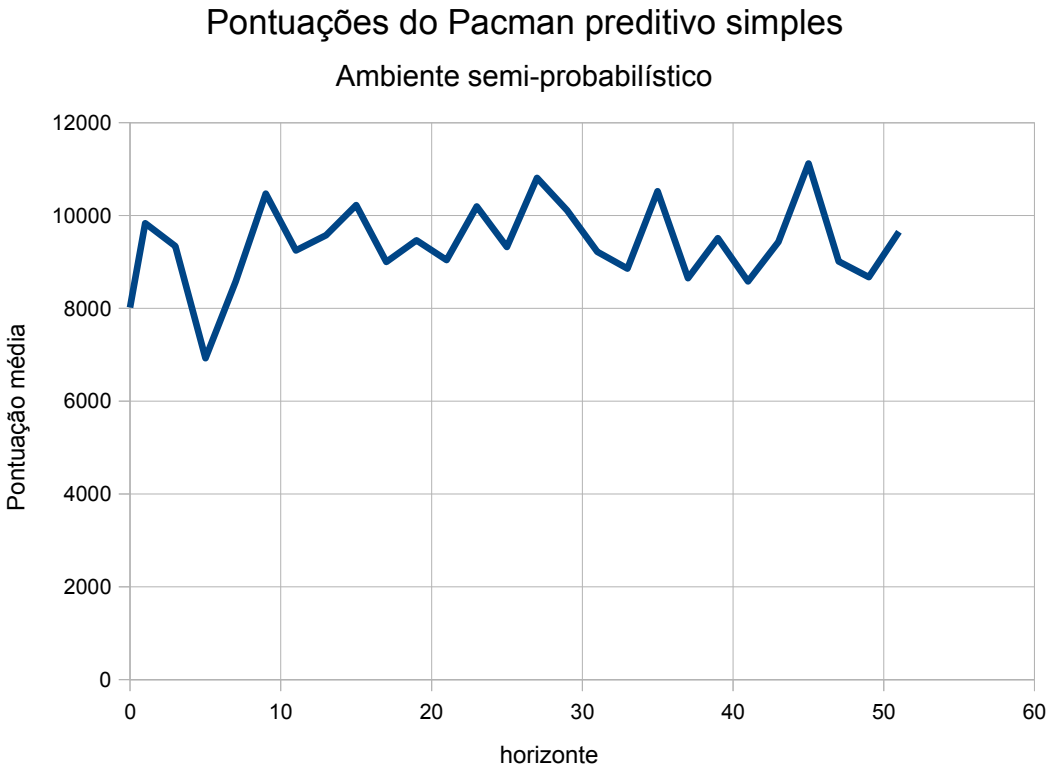

Figura 50: Evolução da pontuação média conforme aumenta-se o horizonte. O modelo utilizado é o model_1_1_0 com lambda=0.9. A vantagem face ao reativo não é muito grande, mas é existente como se pode constatar.

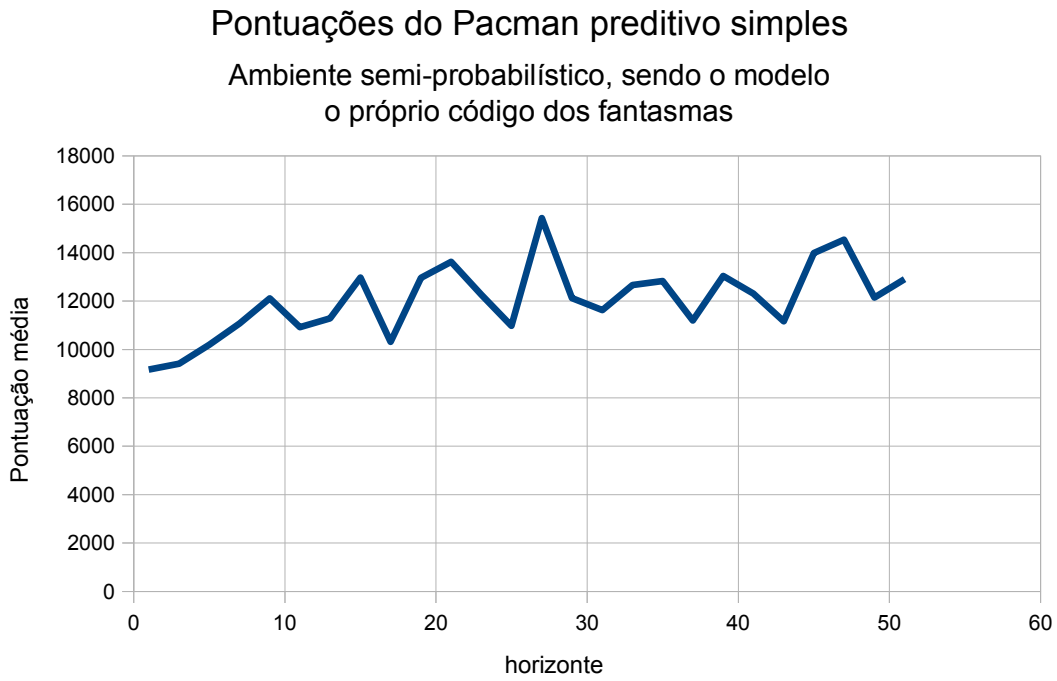

Figura 51: Evolução da pontuação média conforme aumenta-se o horizonte. O modelo utilizado é com base no próprio código dos fantasmas à exceção de Inky, cujo o modelo é o que minimiza a distância do caminho mais curto até ao pacman.

6 Conclusões

O melhor modelo é o que prevê as direções.

Os erros dos modelos demonstram que neste caso a sucessão (posições anteriores do pacman e posições anteriores dos fantasmas) é efeito na maioria das situações para horizontes pequenos como os gráficos demonstram, por causa do erro quadrático, embora o agente tenha dado mais pontos para horizontes maiores. O facto de para horizontes maiores dar mais pontos deve-se à tendência de aproximação que os fantasmas têm para com o pacman, mas não quer dizer que o façam sempre.

Os picos de erros nos gráficos demonstram a presença da componente aleatória nos fantasmas, quando deixam de ser comestíveis, da mudança de comportamento dos fantasmas e da mudança de labirinto; havendo depois uma convergência das posições previstas em relações às reais. Se a equipa Legacy fosse puramente aleatória não seria possível aplicar um modelo linear com sucesso, porque não existiria qualquer comportamento definido.

Os modelos precisos permitem melhores resultados do que o modelo linear à exceção do fantasma Inky. O fantasma Inky escolhe um ponto aleatório para seguir, o que dá tempo ao algoritmo de Bierman de adaptar o modelo linear ao seu comportamento. A cópia do código do Inky piorou os resultados, por causa que acabava por gerar também pontos aleatórios, mas que raramente coincidiam com os pontos reais.

O model_1_1_0 com lambda=0.9 é o modelo linear mais adequado para a equipa Legacy. Isso deve-se ao facto de a próxima direção dos fantasmas Blinky, Pinky e Sue depender só da posição do pacman e do fantasma atuais; através da minimização de distâncias (do caminho mais curto, euclidiana e manhanttan respetivamente) entre 2 pontos.

A arquitetura preditiva adaptativa utilizada ainda não produziu vantagens face à reativa com estados. No entanto o reativo com estados produziu resultados muito superiores em relação ao reativo mais simples. Ainda existem problemas no controlador preditivo devido à oscilação de pontos, quando utiliza o modelo linear ou o modelo do próprio código dos fantasmas num ambiente não determinístico. O algoritmo mais adequado seria o minimax em vez da arquitetura de subsunção com capacidades preditivas adaptativas ou outra arquitetura que em vez de considerar n grupos de pílulas tem as regras ordenadas por distância à pílula mais próxima em relação à posição corrente do pacman.

As equipas que o meu agente teria de defrontar iriam ter comportamentos diferentes. Apesar de o model_1_1_0 com lambda=0.9 é o melhor para a equipa Legacy, as equipas de fantasma adversárias irão ter comportamentos muito mais complexos. A diferença dos erros quadráticos

obtidos entre o model_5_4_0 com lambda=0.94 e o model_1_1_0 com lambda=0.9 é muito pequena. Uma boa cooperação entre fantasmas permite uma estratégia melhor da equipa, o que nesse caso faz com que os fantasmas influenciem-se entre si. Não se sabe qual dos fantasmas é o mais independente em relação aos outros. O movimento do pacman é o mais importante a ter em conta por parte dos fantasmas, mas o modelo escolhido não pode ter um erro muito diferente do que o usado na equipa Legacy. O melhor que posso fazer é escolher um modelo linear o mais adequado possível para uma equipa de fantasmas mais realística e não focar-me num ou noutro fantasma (model_5_4_3 e lambda=0.94).

O modelo linear tem o problema dos parâmetros que não são ajustados pelo RLS.

O trabalho pode ser obtido em [59].

7 Trabalhos futuros

A hierarquia de Brooks usada poderia ser simplificada. A única prioridade imposta às regras e refletida na hierarquia deveria ser a distância dos caminhos em relação à pílula mais próxima. A condição de uma distância de segurança para todos os avanços do pacman caso seguisse um desses caminhos seria uma boa aposta, mas enfrentar só um fantasma nunca foi problema, os pontos comidos pelo caminho são muito importantes e o pacman pode voltar atrás se for possível. Esta não é a melhor opção.

Outra abordagem seria o minimax ou uma alternativa ao minimax, mas desta vez em vez de gerar todos os movimentos possíveis para cada fantasma o modelo linear daria o movimento conhecido dos fantasmas até um dado horizonte. Só nos horizontes não cobertos pelo modelo linear é que a variante imitaria o algoritmo original. A função de avaliação teria em conta os pontos obtidos, situações de morte do pacman e de passagem de um nível do jogo para o outro nível do jogo. O problema desta estratégia pode ser a eficiência versus perda de precisão do modelo linear para horizontes grandes.

Uma outra abordagem utilizaria algumas regras do reativo mais complexo (apresentado neste documento) modificadas e uma procura por DFS com deteção de bloqueio dos fantasmas aos movimentos possíveis do pacman, profundidade limitada e com capacidades de reconhecer os pontos obtidos ou proximidade às pílulas.

O DFS avançaria de cruzamento em cruzamento sem nunca voltar ao cruzamento imediatamente anterior, contabilizando os pontos. Em cada cruzamento faria o cálculo do bloqueio dos fantasmas. O cálculo do bloqueio de fantasmas é baseado nas distâncias entre cruzamentos, que servem para preencher uma matriz de bloqueio. A matriz de bloqueio indica se o fantasma pode bloquear a fuga por aquele caminho ou não. Caso os fantasmas sejam capazes de bloquear todas as fugas, o cruzamento torna-se inviável. Quando o cruzamento é inviável o caminho em causa não gera mais sub-caminhos e é preferível saber até que ponto a contar do cruzamento anterior o pacman pode ir, que é onde esse caminho acaba.

Um fantasma bloqueia uma direção que não volta atrás num cruzamento, caso a distância percorrida pelo pacman desde o seu ponto atual, indo pelo caminho até ao próximo cruzamento depois do avaliado + a distância de segurança é superior à distância do ponto atual do fantasma até ao mesmo cruzamento para onde iria o pacman. O fantasma bloqueia a direção que volta atrás caso duas vezes a distância até ao cruzamento anterior + a distância de segurança + distância do cruzamento anterior, indo pelo caminho até à posição do pacman é superior à distância do fantasma

até ao cruzamento anterior, indo por detrás do pacman (como se estivesse a persegui-lo, enquanto os outros vão tentar cercar pelos outros pontos).

Se o fantasma está comestível o pacman tem a vantagem em distância em função do tempo em que o fantasma está comestível, caso não o coma.

g\r	0	1	2	3
Blinky	x	x		x
Pinky		x		x
Inky				
Sue				

Tabela 22: Exemplo de uma matriz de bloqueio. Os quadrados marcados são as direções que os fantasmas são capazes de bloquear. A variável g é o fantasma. A variável r é a direção.

Uma vez tendo a matriz de bloqueio é preciso saber se o cruzamento é inviável. Caso haja direções sem marcas de fantasmas, então o cruzamento é viável, senão terá de fazer mais avaliações para ver se todas as direções podem ser cobertas. Para cada 1ª repetição tem de haver alternativa, exceto num dos r, porque não pode estar 2 em simultâneo mas terá de ocupar um deles. Para a 2ª repetição tem de haver alternativa sem ser a alternativa já usada. Repete o raciocínio até à n-ésima repetição.

O último algoritmo tem como alvo a pior situação possível, que deve ser tida em conta.

A escolha do caminho prioriza por ordem decrescente de importância: o comprimento do caminho, os pontos obtidos pelo caminho e a proximidade à comida.

8 Referências

[1] Martin V. Butz & Christoph Häberlein, “Ms PacMan Controller”, publicado em 2010, acedido a 19 Maio 2012, em http://dces.essex.ac.uk/staff/sml/pacman/cig2010/CoboPac01.pdf

[2] Nico Piatkowski, Georg Neugebauer, and Boris Naujoks, “Neural Net based Controller for Ms Pac-Man”, publicado em 2008, acedido a 19 Maio 2012, em http://dces.essex.ac.uk/staff/sml/pacman/wcci2008/entries/dortmund.pdf

[3] Jonas Flensbak, Georgios N. Yannakakis, “Ms. Pacman AI controller”, 2008, acedido a 19 Maio 2012, em http://dces.essex.ac.uk/staff/sml/pacman/wcci2008/entries/flensbank.pdf

[4] Hisashi Handa, “Evolutionary Fuzzy Rule Acquisition for Playing Ms.PacMan”, publicado em 2008, acedido a 19 Maio 2012, em http://dces.essex.ac.uk/staff/sml/pacman/wcci2008/entries/handa.pdf

[5] Hiroshi Matsumoto, Chota Tokuyama, and Ruck Thawonmas, “ICE Pambush 2”, publicado em 2009, acedido a 19 Maio 2012, em http://dces.essex.ac.uk/staff/sml/pacman/cec2009/ICEPambush2.pdf

[6]Hiroshi Matsumoto, Takashi Ashida, Yuta Ozasa, Takashi Maruyama, and Ruck Thawonmas, ”ICE Pambush 3”, acedido a 19 Maio 2012, publicado em 2009, em http://dces.essex.ac.uk/staff/sml/pacman/cig2009/ICEPambush3/ICE%20Pambush%203.pdf

[7]Takashi Ashida, Takeru Miyama, Hiroshi Matsumoto, and Ruck Thawonmas, ”ICE Pambush 4”, publicado em 2010, acedido a 19 Maio 2012, em http://dces.essex.ac.uk/staff/sml/pacman/cig2010/ICE%20Pambush%204.pdf

[8] Nathaniel Bell, Xinghong Fang, Rory Hughes, Graham Kendall, Senior Member, IEEE, Edward O’Reilly, Shenghui Qiu, “Ghost Direction Detection and other Innovations for Ms. Pac-Man” , 2010, publicado em 2010, acedido a 19 Maio 2012, em http://dces.essex.ac.uk/staff/sml/pacman/cig2010/jave.pdf

[9]SungUk Jeong and Kyung-Joong Kim, “Improving SmartDijkstraPac Algorithm with Waiting near Power Pills Strategy”,2010, publicado em 2010, acedido a 19 Maio 2012, em http://dces.essex.ac.uk/staff/sml/pacman/cig2010/KimEntry.pdf

[10]Ho Ho Kwong,“Ms. Pac Man Software Agent” 14 May, 2009, publicado em 2009, acedido a 19 Maio de 2012, em http://dces.essex.ac.uk/staff/sml/pacman/cec2009/kwong.pdf

[11] Mr. Bruce K. B. Tong, “Ms. Pac-Man Competition in CIG2010”, publicado em 2010, acedido a 19 Maio 2012, em http://dces.essex.ac.uk/staff/sml/pacman/cig2010/Ms.%20Pac-Man%20Competition%20in%20CIG2010%20-%20Bruce.pdf

[12] Ting-Chi Li, Yu-Chieh Lin, and Tsung-Che Chiang, “A Software Agent for Ms. Pac-Man”, publicado em 2009, acedido a 19 Maio 2012, em http://dces.essex.ac.uk/staff/sml/pacman/cig2009/NTNU/MsPacMan-NTNU.pdf

[13] Martin Emilio, Martinez Moises, Recio Gustavo, Saez Yago Member IEEE, “Pac-mAnt: Optimization Based on Ant Colonies Applied to Developing an Agent for Ms. Pac-Man”, publicado em 2010, acedido a 19 Maio 2012, em http://dces.essex.ac.uk/staff/sml/pacman/cig2010/emilio.pdf

[14] Alan Fitzgerald, Peter Kemeraitis, and Clare Bates Congdon, “RAMP: A Rule-Based Agent for Ms. Pac-Man”, publicado em 2008, acedido a 19 Maio 2012, em http://dces.essex.ac.uk/staff/sml/pacman/wcci2008/entries/usm.pdf

[15] David Robles and Simon M. Lucas, Senior Member, IEEE “A Simple Tree Search Method for Playing Ms. Pac-Man”, publicado em 2009, acedido a 19 Maio 2012, em http://dces.essex.ac.uk/staff/sml/pacman/cig2009/Robles/CigPacTree2009.pdf

[16] Yoshita Kashifuji, Junichi Oda, Hiroshi Matsumoto, Daichi Hirono, and Ruck Thawonmas, “ICE Pescape”,publicado em 2008, acedido a 19 Maio 2012, em http://dces.essex.ac.uk/staff/sml/pacman/wcci2008/entries/RuckPescape.pdf

[17] Jun Ando, Shinichi Shirakawa, Junji Otsuka, Rina Takahashi, Yasuhiro Totsuka and Tomoharu Nagao,“Active Control of Ms.PacMan using SVM learning with human playing data”, publicado em 2008, acedido a 19 Maio 2012, em http://cswww.essex.ac.uk/staff/sml/pacman/wcci2008/entries/shirikawa.pdf

[18] AmirHossein Sojoodi, Mahsa Asadi, “Miss PacMan AI Controller For IEEE CIG 2010 – Ms.PacMan Competition”, publicado em 2010, 19 Maio 2012, acedido a 19 Maio 2012, em http://dces.essex.ac.uk/staff/sml/pacman/cig2010/Sojoodi.pdf

[19] L. Kelly, L. Dicken, J. Levine,“StrathPac - An Automated Player for Ms. Pac-Man”, publicado em 2009, acedido a 19 Maio 2012, em http://dces.essex.ac.uk/staff/sml/pacman/cig2009/StrathPac/StrathPac-final.pdf

[20] Nathan Wirth & Marcus Gallagher, “Influence Maps Ms Pacman Controller CEC 2008 Ms Pacman Competition Entry”, publicado em 2008, acedido a 19 Maio de 2012, em http://dces.essex.ac.uk/staff/sml/pacman/wcci2008/entries/wirth.pdf

[21] Martin Emilio, Martinez Moises, Recio Gustavo, Saez Yago Membros do IEEE“Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem”, Université Libre de Bruxelles, Belgium, publicado em Maio de 2006, acedido a 19 Maio de 2012, em http://www.idsia.ch/~luca/acs-ec97.pdf

[22] Debasis Sengupta & Sreenivasa Rao Jammalamadaka, “Series on Multivariate Analysis”, publicado em Março de 2003, acedido a 19 Maio de 2012, em

http://www.worldscibooks.com/mathematics/4674.html, página 4

[23] Anders Sjo, "Updating Techniques in Recursive Least-Squares Estimation", publicado em 10 de Outubro de 1992, acedido a 19 Maio de 2012, em http://www.maths.lth.se/na/Publications/Sjo92b.pdf

[24] Philipp Rohlfshagen & Simon M. Lucas, "Ms Pac-Man VERSUS Ghost Team CEC 2011 Competition", publicado em 2011, acedido a 19 Maio 2012, em http://algoval.essex.ac.uk/rep/games/PacmanVersusGhostTeam2011.pdf

[25] IEEE, Página principal da conferência, atualizado em 2012, acedido a 19 Maio 2012, em http://www.ieee-cig.org/

[26] Video que mostra uma execução de um dos agentes da competição, publicado em 17 de Agosto de 2009, acedido a 19 Maio de 2012, em http://www.youtube.com/watch?v=Zo0YujjX1PI

[27] Philipp Rohlfshagen & David Robles & Simon Lucas, Página oficial da competição Pacman contra equipas de fantasmas, acedido a 19 Maio 2012, em http://www.pacman-vs-ghosts.net/

[28] Repositório de código da minha framework, acedido a 28 Fevereiro de 2010, em http://code.google.com/p/ms-pacman/

[29] Janela do ms. Pacman, utilizada para processamento de imagem, acedido em Setembro de 2011, em http://www.webpacman.com/mspacman.htm

[30]Simon M. Lucas, Página da competição pacman com captura de ecrã, acedido a 19 Maio de 2012, em http://dces.essex.ac.uk/staff/sml/pacman/PacManContest.html

[31] Simon M. Lucas, Competição pacman captura de ecrã CIG 2010, publicado em 2010, acedido a 19 Maio de 2012, em http://dces.essex.ac.uk/staff/sml/pacman/CIG2010Results.html

[32] Simon M. Lucas, Competição pacman captura de ecrã GIG 2009, publicado em 2009, acedido a 19 Maio de 2012, em http://dces.essex.ac.uk/staff/sml/pacman/CIG2009Results.html

[33] Simon M. Lucas, Competição pacman captura de ecrã CEC 2009, publicado em 2009, acedido a 19 Maio de 2012, em http://dces.essex.ac.uk/staff/sml/pacman/CEC2009Results.html

[34] Simon M. Lucas, Congresso mundial em inteligência computacional, publicado em 2008, acedido a 19 Maio de 2012, em http://dces.essex.ac.uk/staff/sml/pacman/WCCI2008Results.html

[35] Simon M. Lucas, Conferencia CIG 2011, publicado em 2011, acedido a 19 Maio 2012, em http://dces.essex.ac.uk/staff/sml/pacman/CIG2011Results.html#[1]

[36] Nozomu Ikehata and Takeshi Ito, “Monte-Carlo Tree Search In Ms. Pac-Man, Proceedings of IEEE Conference on Computational Intelligence and Games 2011”, Seoul, Korea. Publicado em

2011, acedido a 19 Maio 2012, em
http://cilab.sejong.ac.kr/cig2011/proceedings/CIG2011/papers/paper7.pdf
[37] Takeru Miyama, Asami Yamada, Yuki Okunishi, Takashi Ashida, and Ruck Thawonmas, "ICE Pambush 5", publicado em 2011, acedido a 19 Maio 2012, em
http://dces.essex.ac.uk/staff/sml/pacman/ICEPambush5.pdf
[38]Kuan-Wei Chen, Chu-Ming Wang, and Tsung-Che Chiang, "NTNU CIG 2011", publicado em 2011, acedido a 19 Maio de 2012, em
http://dces.essex.ac.uk/staff/sml/pacman/NTNUCIG2011.pdf
[39] B. K. B. Tong and C. W. Sung, "A Monte-Carlo Approach for Ghost Avoidance in the Ms. Pac-Man Game," International IEEE Consumer Electronics Society's Games Innovations Conference (ICE-GIC), pp. 1-8, 2010.
[40] CIG 2011do pacman contra equipas de fantasmas, publicado em 2011, acedido a 23 de Maio de 2012, em http://www.old.pacman-vs-ghosts.net/cig11results/
[41]Daryl Tose, Spooks, publicado em 2011, acedido a 23 de Maio de 2012, em
http://www.old.pacman-vs-ghosts.net/cig11results/report.php?id=33
[42] Daryl Tose, University of Essex, "PhantomMenance", publicado em 2011, acedido a 23 de Maio de 2012, em http://www.old.pacman-vs-ghosts.net/cig11results/report.php?id=35
[43]B. K. B. Tong, C. M. Ma, and C. W. Sung, "A Monte-Carlo Approach for the Endgame of Ms. Pac-Man, Proceedings of IEEE Conference on Computational Intelligence and Games 2011", publicado em 2011, acedido a 19 Maio de 2012, em
http://cilab.sejong.ac.kr/cig2011/proceedings/CIG2011/papers/paper3.pdf
[44]Philipp Rohlfshagen, CEC 2011 equipa de fantasmas contra pacman, publicado em 2011, acedido a 1 Janeiro de 2012, em http://www.old.pacman-vs-ghosts.net/cec11results/rankings.php
[45]Philipp Rohlfshagen, CEC 2011 pacman contra equipas de fantasmas, publicado em 2011, acedido a 1 Janeiro de 2012, em www.old.pacman-vs-ghosts.net/cec11results/viewReport.php?id=5
[46] James, H. B. Low School of Computer Science Faculty of Science University Of Nottingham Malaysia Campus, "CEC 2011 Ms Pac-Man vs Ghost Team Competition", publicado em 2011, acedido a 1 Janeiro de 2012, em http://www.old.pacman-vs-ghosts.net/cec11results/viewReport.php?id=7
[47] Aske Plaat, "Minimax MTD(f) for Chess", acedido a 19 de Maio de 2012, em
http://people.csail.mit.edu/plaat/mtdf.html#abmem
[48] Philipp Rohlfshagen, Repositório de relatórios do CIG 2011, publicado em 2011, acedido a 23 Maio de 2012, em http://www.old.pacman-vs-ghosts.net/cig11results/view_reports.php
[49] Samuel Sarjant, CERRLA, publicado em 2011, acedido a 19 Maio de 2012, em

http://cilab.sejong.ac.kr/cig2011/proceedings/CIG2011/papers/paper47.pdf

[50] Ranking dos agentes apresentados no CEC 2011, publicado em 2011, acedido a 23 de Maio de 2012, em http://old.pacman-vs-ghosts.net/cec11results/rankings.php

[51] Ranking do CIG 2011, publicado em 2011, acedido a 23 de Maio de 2012, em http://old.pacman-vs-ghosts.net/cig11results/

[52] Relatórios dos agentes apresentados no CEC 2011, publicado em 2011, acedido a 23 de Maio de 2012, em http://old.pacman-vs-ghosts.net/cig11results/view_reports.php

[53] Anne Auger, "Benchmarking the (1+1) Evolution Strategy with One-Fifth Success Rule on the BBOB", publicado em 2009, acedido a 19 Maio de 2012, em http://dl.acm.org/citation.cfm?id=1570342

[54] Samuel Sarjant, "Cerrla: A Ms. Pacman Agent", publicado em Setembro de 2011, acedido a 24 de Maio de 2012, em http://old.pacman-vs-ghosts.net/cig11results/report.php?id=38

[55] Bruce Kwong-Bun Tong & Chi Wan Sung, Departamento de engenharia electrónica, universidade da cidade de Hong Kong, "CIG 2011 Ms Pac-Man Competition (Ms Pac-Man vs Ghost Team)", publicado em Setembro de 2011, acedido a 24 de Maio de 2012, em http://old.pacman-vs-ghosts.net/cig11results/report.php?id=40

[56] Atif M. Alhejali & Simon M. Lucas, School of Computer Science and Electronic Engineering, University of Essex, UK, "Developing a Ms. Pac-Man Controller for The CEC 2011 Competition Using Training Camp and Genetic Programming", publicado em Setembro de 2011, acedido a 24 de Maio de 2012, em http://old.pacman-vs-ghosts.net/cec11results/viewReport.php?id=8

[57] Matthias F. Brandstetter, De Montfort University, "An Evolved Controller for the Ms. Pac Man Video Game based on Genetic Programming", publicado em Setembro de 2011, acedido a 24 de Maio de 2012, em http://old.pacman-vs-ghosts.net/cec11results/viewReport.php?id=6

[58] Masahiro NAKAMURA & Kien Quang NGUYEN & Ruck THAWONMAS, universidade Ritsumeikan, "ICE pAmbush CIG11", publicado em Setembro de 2011, acedido a 24 de Maio 2012, em http://old.pacman-vs-ghosts.net/cig11results/report.php?id=50

[59] Osvaldo F. L. Mendes, universidade do Algarve, "Arquitetura de subsunção com capacidades preditivas adaptativas para o pacman", publicado em 1 de Junho de 2012, acedido a 1 de Junho de 2012, em http://intranet.deei.fct.ualg.pt/~a27961

Nota: alguns ficheiros online podem ser visualizados mudando a extensão de php para pdf