UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI
Bacharelado em Sistemas de Informação
Liliane Vieira Lopes

Q-LEARNING ACELERADO POR HEURÍSTICAS (HAQL) APLICADO AO DOMÍNIO DE SISTEMAS TUTORES INTELIGENTES, COM MODELAGEM AUTÔNOMA DO APRENDIZ

Diamantina
2018

Liliane Vieira Lopes

Q-LEARNING ACELERADO POR HEURÍSTICAS (HAQL) APLICADO AO DOMÍNIO DE SISTEMAS TUTORES INTELIGENTES, COM MODELAGEM AUTÔNOMA DO APRENDIZ

Trabalho de Conclusão de Curso apresentado ao curso de Graduação em Sistemas de Informação, como parte dos requisitos exigidos para a obtenção do título de Bacharel em Sistemas de Informação

Orientador: Marcus Vinicius Carvalho Guelpeli
Coorientador: Éverton de Oliveira Paiva

Diamantina
2018

Liliane Vieira Lopes

Q-LEARNING ACELERADO POR HEURÍSTICAS (HAQL) APLICADO AO DOMÍNIO DE SISTEMAS TUTORES INTELIGENTES, COM MODELAGEM AUTÔNOMA DO APRENDIZ

Trabalho de Conclusão de Curso apresentado ao curso de Graduação em Sistemas de Informação, como parte dos requisitos exigidos para a obtenção do título de Bacharel em Sistemas de Informação

Orientador: Prof. Dr. Marcus Vinicius Carvalho Guelpeli
Coorientador: Me. Éverton de Oliveira Paiva

Data de aprovação ___/___/____.

Prof. Dr. Marcus Vinicius Carvalho Guelpeli
Faculdade de Ciências Exatas e Tecnológicas - UFVJM

Me. Éverton de Oliveira Paiva
Diretoria de Tecnologia da Informação - UFVJM

Prof. Dr. Alexandre Ramos Fonseca
Instituto de Ciência e Tecnologia - UFVJM

Diamantina
2018

AGRADECIMENTOS

À minha família, em especial aos meus pais Maria Aparecida Vieira Lopes, Odilon Lopes Pego e minha tia Maria Salete Vieira dos Santos, pela dedicação, carinho e apoio em todos os momentos.

À todos os meus amigos, em especial Arthur Almeida, Lucas Araújo, Fernanda Macedo, Natália Leal, Thalita Pinheiro, Mayra Bruce e família, Daniel Marques, Renato Gonçalves e Adriano Soares, que colaboraram para a concretização desse curso e tornaram a caminhada mais prazerosa.

Aos meus colegas e amigos do Museu do Diamante/Ibram pela compreensão e contribuições diárias.

Aos meus orientadores Marcus Vinícius Carvalho Guelpeli e Éverton de Oliveira Paiva pela oportunidade e confiança.

Ao Marcos Calebe Barcellos pelo cuidado e incentivo.

RESUMO

A incorporação de tecnologias no processo de ensino-aprendizagem tem ganhado força nas últimas décadas, em consonância com os avanços da globalização e popularização da internet. Nesse contexto, os Sistemas Tutores Inteligentes são ferramentas poderosas por promoverem o dinamismo e permitirem a personalização do processo de aquisição do conhecimento, principalmente em Ambientes Virtuais de Aprendizagem, onde inexiste a figura humana do tutor. A fim de contribuir para o aprimoramento desses sistemas, propõe-se solucionar o problema da convergência lenta do algoritmo *Q-learning* em Sistemas Tutores Inteligentes com modelagem autônoma do aprendiz. Para isso adicionou-se o *Q-learning* Acelerado por Heurísticas, na função de transição de estados do algoritmo, com vistas a sua otimização. Inicialmente, foram realizadas adaptações dos parâmetros, a fim de adequá-los ao domínio. Em seguida, foram realizados sucessivos testes, em ambiente não determinístico, utilizando uma política pedagógica menos restritiva e os modelos de aprendiz ruim, bom e excelente. Os resultados obtidos apontam para a melhoria do desempenho do algoritmo de aprendizado em Sistemas Tutores Inteligentes.

Palavras-chave: Sistemas Tutores Inteligentes. *Q-learning*. Aprendizado por reforço. HAQL. Aceleração da convergência.

ABSTRACT

The incorporation of technologies in the teaching-learning process has gained strength in recent decades, in line with the advances of globalization and popularization of the internet. In this context, the Intelligent Tutoring Systems are powerful tools to promote the dynamism and allow customization of the knowledge acquisition process, especially in Virtual Learning Environments, where the human figure of the tutor does not exist. In order to contribute to the improvement of these systems, it is proposed to solve the problem of the slow convergence of the algorithm Q-learning in Intelligent Tutors Systems with autonomous modeling of the learner. To this end, the Heuristic Accelerated Q-learning was added in the state transition function of the algorithm, with a view to its optimization. Initially, adaptations of the parameters were made, in order to adapt them to the domain. Then, successive tests were carried out, in a non-deterministic environment, using a less restrictive pedagogical policy, and poor, good and excellent learner models. The results obtained point to the improvement of the performance of the learning algorithm in Intelligent Tutoring System.

Keywords: Intelligent Tutors. Q-learning. Reinforcement learning. HAQL. Acceleration of convergence.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

AR	Aprendizado por Reforço
AVA	Ambiente Virtual de Aprendizagem
CAI	Computer-Assisted Instruction
HAL	Heuristically Accelerated Learning
HAQL	Heuristically Accelerated Q-Learning
IA	Inteligência Artificial
ICAI	Intelligence Computer-Assisted Instruction
PDM	Processo Decisório de Markov
STI	Sistemas Tutores Inteligentes

LISTA DE SÍMBOLOS

α	Taxa de aprendizagem
γ	Taxa de desconto temporal para os reforços futuros
η	Parâmetro aditivo de valor a heurística
ξ	Parâmetro que pondera a influência da função heurística
π	Um política de ações
π^*	Uma política ótima de ações
$\hat{Q}$	Valor estimado da função Valor-ação
Q	Função Valor-ação
S	Conjunto finito de estados
A	Conjunto finito de ações
R	Conjunto de reforços

SUMÁRIO

1 INTRODUÇÃO

Na década de 1950, os idealizadores da ciência da computação foram audaciosos no sentido de defenderem a possibilidade de criação de máquinas computacionais, dotadas de Inteligência Artificial (IA), capazes de agir racionalmente, simular ações humanas e executar tarefas complexas com exatidão.

Cerca de trinta anos mais tarde, a Era Digital democratizou o acesso a internet e ampliou os meios de comunicação a nível global. Na área Pedagógica, a chegada das Tecnologias da Informação e Comunicação (TIC's) disseminou as modalidades de ensino semi e não presenciais. Inicialmente, as chamadas Instruções Assistidas por Computador (CAI) substituíram os impressos e deram suporte aos métodos tradicionais de ensino. Posteriormente, esses sistemas (ICAI) agregaram técnicas de IA que permitiram aos estudantes obterem um retorno das últimas ações praticadas. A informatização dos processos de ensino provocou mudanças tanto na técnica como na cultura pedagógica. Para Belloni (2002), essas transformações desafiaram os profissionais em educação. Elas criaram um novo paradigma de ensino, no qual o contato físico entre alunos e instituições foi reduzido, oportunizando o aprendizado sem mestres.

Neste contexto, surgiram os Sistemas Tutores Inteligentes (STI's): softwares educacionais que incorporam técnicas de IA e possibilitam individualizar planos de estudo, com estratégias pedagógicas escolhidas de acordo com o perfil de cada aluno (GAVIDIA; ANDRADE, 2003; VICCARI; GIRAFFA, 1996; JESUS, 2009).

O Aprendizado por Reforço (AR) é uma subárea da IA que confere autonomia aos agentes computacionais ao promoverem o aprendizado por meio da interação com o ambiente. Ele fornece ao agente um sinal de reforço que o guia ao seu objetivo (RUSSEL; NORVIG, 2013). Esse método permite ao agente tutor dos STI's o alcance de uma política ótima que incorpore estratégias personalizadas de ensino.

A estrutura tradicional dos STI's (VICCARI; GIRAFFA, 1996) permitiu aos pesquisadores a decomposição de seus módulos para fins de estudo. Com isso, foi possível identificar problemas e propor técnicas de otimização da estrutura. Giraffa e Viccari (1999) propuseram a modelagem de STI através da estrutura de agentes e pesquisaram sobre as interações que ocorrem entre tutor e aluno. Barros e Santos (2000) estudaram técnicas de modelagem do domínio em STI, para o ensino de Geometria Descritiva. Gavidia e Andrade (2003) fizeram uma revisão histórica e discutiram a viabilidade de sua inclusão na área pedagógica. Guelpeli (2003) se dedicou a criação de uma módulo diagnóstico que permitiu, a partir de técnicas de Aprendizagem por Reforço, a captação de um modelo mais fiel do aluno durante sua interação com o tutor. Greer e McCalla (2013) contribuíram para a modelagem do aluno ao analisarem erros das atuais formas de modelagem presentes na literatura e proporem um modelo. Torres (2017) abordou um método de seleção de estratégias pedagógicas em STI.

1.1 Identificação do Problema

Sabendo-se que o processo de aprendizagem da política ótima de ações ocorre por interação do agente com o ambiente, quanto maior o número de interações necessárias entre tutor e aprendiz, maior será o tempo despendido para alcance da política ótima de ações. O que significa que a convergência dos algoritmos de aprendizado é lenta em ambientes complexos (SUTTON; BARTO, 2012; GUELPELI, 2003).

1.2 Motivação

Atualmente, os STI's se destacam como o ambiente mais utilizado na área da informática para a educação a considerar o estado afetivo do aprendiz no processo de aquisição do conhecimento (MORAIS *et al.*, 2017). Desde o seu nascimento, todos os esforços se concentram para tornar o processo de ensino-aprendizagem à distância o mais próximo possível daquele praticado em sala de aula.

Não obstante, há uma dificuldade de se representar na máquina os estados mentais humanos que englobam fatores emocionais e criativos. Como a capacidade de tutoria dos STI's está diretamente relacionada ao sucesso da técnica de modelagem do perfil do estudante e a rapidez de resposta do tutor, a aceleração da convergência do algoritmo de aprendizagem nesses ambientes se mostra como uma alternativa assertiva para aproximar o ambiente virtual do real. Essa aproximação potencializa sua função como recurso pedagógico e contribui para a otimização dos atuais Ambientes Virtuais de Aprendizagem (AVA).

1.3 Hipótese

Admite-se a possibilidade de aceleração da convergência do algoritmo de aprendizagem por reforço *Q-learning*, através da alteração de sua função de transição de estados, cuja política regulamenta a escolha das ações pelo agente tutor.

1.4 Contribuições

O presente trabalho tem por objetivo acelerar a convergência do algoritmo *Q-learning* em STI's com modelagem autônoma do aprendiz, a partir da inclusão de técnicas heurísticas na transição de estados do algoritmo *Q-learning*. As principais contribuições derivadas desse objetivo são:

- Criação de técnicas de adaptação dos parâmetros do algoritmo *Q-learning* Acelerado por Heurísticas (BIANCHI, 2004) ao ambiente de STI;
- Implementação do algoritmo HAQL em ambiente de simulação;
- Execução de testes de desempenho do algoritmo, através do método gráfico;

- Disseminação de técnicas de aperfeiçoamento de um dos mais importantes algoritmos do AR: *Q-learning* de Watkins (1989);

- Colaboração para a melhoria dos atuais recursos de Educação à Distância (EAD).

1.5 Metodologia

Após a identificação do problema de convergência do algoritmo *Q-learning*, tal como ocorre no domínio de Sistemas Tutores Inteligentes com modelagem autônoma do aprendiz (GUELPELI, 2003), procedeu-se à fase de pesquisa e estudo de técnicas com potencial de aplicação em Aprendizado por Reforço. Sendo assim, optou-se pela utilização de um método heurístico, devido a suas estratégias de abstração do problema que tornam a tomada de decisão mais rápida e precisa por parte dos agentes, levando a soluções ótimas. (GIGERENZER; GAISSMAIER, 2011).

O algoritmo escolhido para aplicação neste trabalho é o *Q-learning* Acelerado por Heurísticas (HAQL), proposto por Bianchi (2004). Ele associa uma heurística genérica à regra de transição de estados do *Q-learning*, mantendo todas as demais características e vantagens deste algoritmo no processo de aprendizagem.

Em um segundo momento, realizou-se a inclusão do HAQL no programa Protótipo Simulador de Sistema Tutor Inteligente, originalmente desenvolvido por Guelpeli (2003) e, posteriormente, adaptado por Paiva (2016). A implementação foi realizada utilizando os conceitos do paradigma de programação Orientado a Objetos e a linguagem C++, no ambiente de desenvolvimento *QT Creator*.

Em seguida, passou-se à fase de ajuste dos parâmetros heurísticos, uma vez que o trabalho original foi desenvolvido para o ambiente de navegação robótica.

O primeiro parâmetro ajustado se refere a identificação da melhor fase para extração da estrutura heurística em tempo de execução. Para isso, foram realizados sete testes com variação do tempo de captura até as iterações 5, 10, 15, 20, 30, 50 e 55. Por conseguinte, efetuou-se a análise gráfica dos resultados e eleição do período que retornou os melhores resultados.

Na sequência, buscou-se adequar o parâmetro adicional η que no trabalho original assumiu o valor 1. Diante disso, foram conduzidos cinco testes nos quais η assumiu valores entre 0 e 2. Ao final, selecionou-se o parâmetro mais eficaz.

Outros dois testes foram realizados com o parâmetro multiplicativo da influência heurística ξ, que em Bianchi (2004) adotou o valor 1. Neste caso, examinou-se o comportamento do algoritmo para os valores 0,5 e 1.

Por fim, o algoritmo adaptado foi ajustado ao STI e testado em ambiente não determinístico, utilizando a política pedagógica P1- menos restritiva, e os modelos M1 – Ruim, M2 – Bom e M3 – Excelente. Para cada um desses modelos foram realizadas 20 simulações com 500, 1000 e 3000 iterações. As médias dos valores Q(s, a) foram analisados graficamente e

comparados com os trabalhos de Guelpeli (2003) e Paiva (2016) para aferição do desempenho do algoritmo de aprendizagem.

1.6 Organização do Trabalho

No Capítulo 2 é apresentada uma visão geral acerca dos Sistemas Tutores Inteligentes e as principais técnicas de IA utilizadas para a promoção do aprendizado autônomo nos agentes computacionais. Foram apresentados os trabalhos correlatos e suas contribuições para este estudo.

No Capítulo 3 foram detalhadas as técnicas para adaptação dos parâmetros do algoritmo HAQL, bem como as análises gráficas que embasaram as alterações realizadas. Foram elencadas as etapas de implementação do algoritmo no STI e as configuração utilizadas na execução dos testes de desempenho.

Nos Capítulo 4 foram expostos e discutidos os resultados dos testes de desempenho. A análise gráfica dos valores Q(s, a) foi realizada em comparação com os métodos de explotação Greedy e de exploração Tabu e Grasp.

O cruzamento dos dados gráficos, para fins de análise comparativa, foi realizado no Capítulo 5.

Finalmente, no Capítulo 6 avaliou-se o alcance do objetivo de aceleração do aprendizado por reforço no domínio de STI, por meio da hipótese de aceleração da convergência pela introdução de métodos heurísticos. Ainda foram listadas as limitações deste trabalho e sugeridas extensões para trabalhos futuros.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta os principais conceitos e técnicas utilizadas neste trabalho. Na primeira seção fala-se sobre os Sistemas Tutores Inteligentes, sua arquitetura, vantagens e desvantagens. Posteriormente, são abordados os conceitos de Inteligência Artificial, suas subáreas e métodos de aprendizagem.

2.1 Sistemas Tutores Inteligentes

A utilização de computadores como ferramenta auxiliar de ensino na área pedagógica data de 1950. Nesse momento, os programas foram utilizados para a apresentação sistematizada, sequencial e finita de temáticas aos alunos. Nesse contexto, a tecnologia atuava como suporte para o conteúdo abordado pelos métodos tradicionais de educação e, por vezes, apenas substituíram esses recursos. Essas "apostilas digitais" ficaram conhecidas como Instrução Assistida por Computador (CAI). (VICCARI; GIRAFFA, 1996; GIRAFFA, 1999; JESUS, 2009).

Na década de 60, surgiram sistemas capazes de dar *feedback* ao estudante, através da metodologia de perguntas e respostas. Já na década de 70, a utilização dos computadores para apoio pedagógico evoluiu e passou a incluir técnicas de IA que permitiram a personalização de conteúdo. Os Sistemas de Instrução Assistida por Computador Inteligente (ICAI), como ficaram conhecidos, distinguiam-se dos sistemas CAI's na medida em que aplicavam técnicas pedagógicas com base nas aptidões individuais, a fim de proporcionar uma melhor experiência de aprendizagem e não mais promovê-la por meio de mecanismos de associação e sugestão. (VICCARI; GIRAFFA, 1996; GIRAFFA, 1999; JESUS, 2009).

Para Giraffa (1999), os STI's são uma modalidade dos ICAI's. Neles há uma preocupação de se personificar o aluno, criando um modelo que englobe seus saberes e vivências. Durante o processo de interação, o STI capta o *Know-how* do aprendiz e percebe quais os métodos o levaram aos melhores resultados. Isso o permite alterar suas bases do conhecimento e criar estratégias de ensino individualizadas (VICCARI; GIRAFFA, 1996). Essa característica o destaca de seus precursores, pois quanto mais legítimo for o protótipo do aluno, maior será a capacidade de tutoria do sistema e vice-versa.

Viccari e Giraffa (1996), Gavidia e Andrade (2003), Jesus (2009), enumeram os elementos que compõem sua arquitetura original:

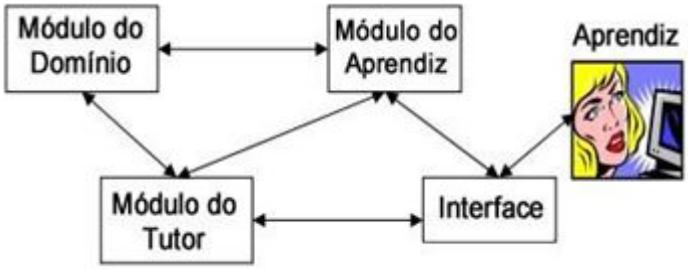
1. Módulo Domínio - é um banco de dados que contêm regras e informações sobre a ciência a ser estudada. (MCTAGGART, 2001).

2. Módulo Pedagógico - constituem as estratégias pedagógicas utilizadas para ensinar um determinado conteúdo. Elas podem se apresentar como vídeos, jogos, áudios, perguntas etc.

3. Módulo Aluno - representa o perfil do aprendiz e capta o seu conhecimento sobre determinado assunto. Conforme explicação anterior, a fidelidade na representação desse modelo é o quesito que transforma o sistema em tutor ou assistente.

4. Módulo Interface - responsável pela comunicação e apresentação do conteúdo ao aluno. Vale ressaltar que quanto mais flexível for a interface, maior será a experiência proporcionada ao usuário.

Figura 1 – Arquitetura Clássica de um STI



Fonte – Gavidia e Andrade (2003).

A estrutura fragmentada da arquitetura clássica proposta para os STI's viabilizou o exame de cada elemento e suas interligações. Esse fator permitiu o aperfeiçoamento desses sistemas e impulsionou sua utilização na área de educação. Contudo, também surgiram várias limitações de natureza teórica e metodológica que retardaram a evolução desses sistemas e limitaram a exímia simulação de instrutores humanos. Para Jesus (2009), esses obstáculos estão relacionados à dificuldade de representação dos estados mentais e do próprio comportamento humano. Greer e McCalla (2013) ainda apontam a complexidade de interpretação de condutas criativas, frequentemente apresentadas pelos alunos e um ambiente ruidoso que pode levar a construção de modelos inconsistentes.

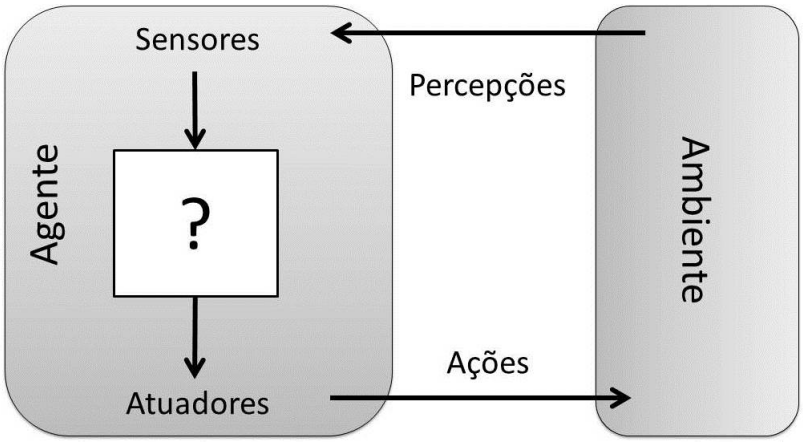
Para progredir com a concretização dos Sistemas de Tutores Inteligentes, os estudiosos buscaram respaldo na área da computação com a incorporação de técnicas de Inteligência Artificial emergentes no período.

2.2 Inteligência Artificial

O campo da IA surgiu em 1956, com o propósito de construir máquinas inteligentes. Ela se dedica a estudar e melhorar o comportamento de agentes, a fim de refinar suas tarefas e aproximá-las o tanto quanto possível daquelas desempenhadas pelos indivíduos.

Segundo Russel e Norvig (2013), um agente é uma entidade capaz de adaptar-se as mudanças do meio no qual está inserido e alcançar seus objetivos através da percepção e ação sobre este ambiente.

Figura 2 – Agentes interagem com o ambiente por meio de sensores e atuadores



Fonte – Russel e Norvig (2013).

Pela Figura 2, compreende-se que um agente percebe o ambiente por meio de sensores e reage a ele praticando ações através de atuadores. Ao praticar uma ação, este ambiente se altera e o processo continua por realimentação.

Dentro da arquitetura física dos agentes está embarcada a função agente, ou seja, o programa que irá mapear as percepções em ações. A Aprendizagem de Máquina é uma subárea da IA que propõe diversas abordagens e algoritmos para a concretização dessa estrutura dos agentes (RUSSEL; NORVIG, 2013).

2.2.1 **Aprendizagem de Máquina**

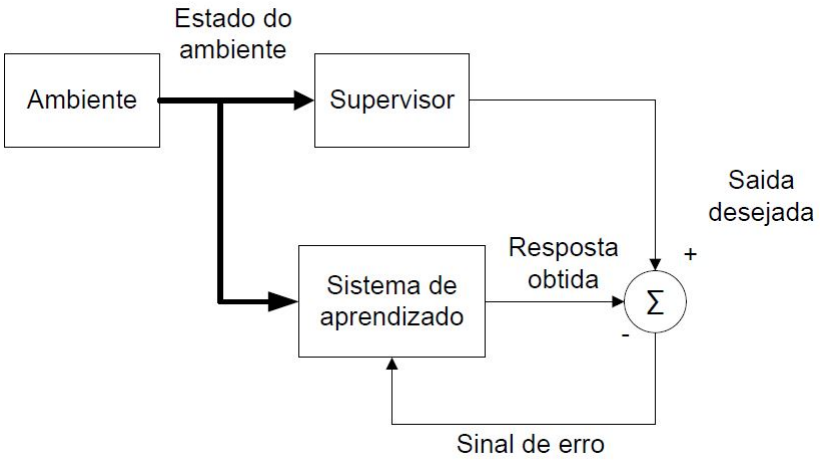
A aprendizagem de máquina é o campo da IA que se dedica a criar computadores e programas capazes de melhorar o seu desempenho a partir da própria experiência (MITCHELL, 1997). Ela estabelece três métodos de aquisição do conhecimento passíveis de implementação, que se diferem conforme o tipo de retorno emitido ao sistema de aprendizagem.

Na supervisionada, como o próprio nome sugere, existe a figura de um supervisor. Ele tem conhecimento do ambiente e diz ao agente qual é a ação desejada para aquele momento. Com base na comparação da saída desejada e a ação executada pelo agente, o sistema de aprendizagem retorna um sinal de erro.

No aprendizado não supervisionado um sistema de classificação compara as entradas com padrões pré-definidos, descartando aquelas não desejadas (RUSSEL; NORVIG, 2013).

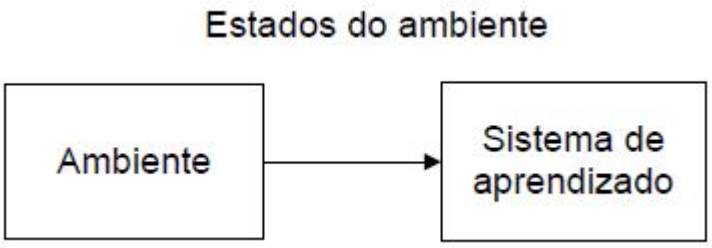
O terceiro tipo é a Aprendizagem por Reforço (AR), na qual o agente aprende a partir de interações com o ambiente. Nessa situação, um sinal de reforço é fornecido como *feedback* e pode significar, a depender do objetivo do agente, uma recompensa ou penalidade. O sinal aqui mencionado, não deve ser confundido ou comparado com o sinal de erro da aprendizagem supervisionada. Nessa o agente aprende a partir de exemplos fornecidos por um elemento externo. Essa característica não é viável em aprendizado por interação, no qual é impossível obter exemplos de comportamentos corretos para todas as situações possíveis, dentro de um

Figura 3 – Diagrama do modelo de aprendizagem supervisionada



Fonte – Lopez (2010).

Figura 4 – Diagrama do modelo de aprendizagem não supervisionada



Fonte – Lopez (2010).

ambiente inexplorado pelo agente (SUTTON; BARTO, 2012; DORÇA *et al.*, 2012). Nesse caso, o ideal é que ele aprenda a partir da experimentação.

2.2.1.1 **Aprendizagem por Reforço**

Os recursos de AR são utilizados quando se deseja conferir autonomia aos agentes. Para Sutton e Barto (2012), existem três elementos comuns nos Sistemas de Aprendizagem por Reforço. A primeira delas é a política de ações, entendida como o conjunto de regras que associam estados a ações, que orienta o comportamento do agente em um dado momento.

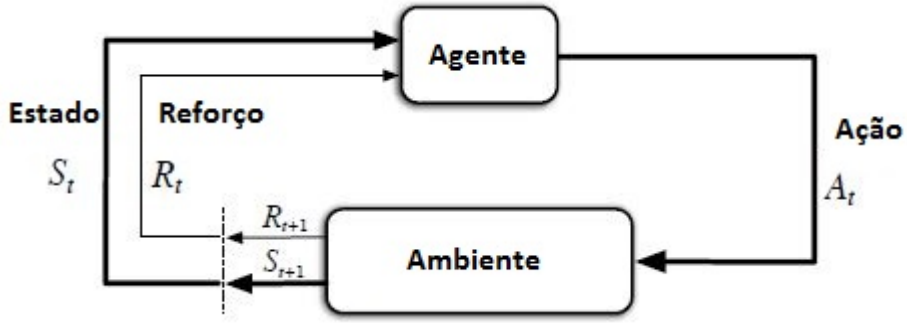
Uma função de recompensa que atribui um valor numérico para cada par estado-ação, indicando o quão útil foi a tomada de decisão do agente em um dado instante. Lembrando que, o objetivo a longo prazo de um agente de aprendizado por reforço é acumular o maior valor possível no somatório das recompensas (GUELPELI, 2003).

Ao passo que a função de recompensa indica a importância da ação no momento exato em que ela é escolhida, a função valor-ação indica a qualidade da ação a longo prazo, pois um estado pode ter um baixo valor de recompensa, mas possibilitar a chegada a estados

subsequentes com altos valores de recompensa e vice-versa (GUELPELI; OMAR; RIBEIRO, 2004).

O agente será sempre o tomador de decisão. E o ambiente será tudo o mais que interagir com ele.

Figura 5 – Interação do agente com o ambiente em AR



Fonte – Sutton e Barto (2012), tradução da autora.

Outra característica importante desses sistemas é o dilema vivido pelos agentes que devem escolher por utilizar as ações que historicamente levaram a um bom resultado (explotação) ou optar por uma ação de valor desconhecido (exploração) a fim de pesquisar por ações ainda melhores. O alcance do objetivo depende do equilíbrio dessas escolhas (GUELPELI, 2003).

Sobre os estados do ambiente, Sutton e Barto (2012) assumem que tanto as decisões como os valores são funções apenas do estado atual, ou seja, admite-se a possibilidade de obtenção de soluções partindo somente do estado atual. Essa característica indica que o ambiente dos sistemas de aprendizagem por reforço satisfaz a Propriedade Decisória de Markov (PDM). Porém, ainda que o conjunto dos estados do problema a ser modelado não obedeça estritamente a PDM, é sempre desejável que o estado atual sirva como base para a previsão dos estados subsequentes, recompensas futuras e ações.

2.2.1.2 Algoritmo *Q-learning*

O algoritmo *Q-learning* (WATKINS, 1989), representa um dos mais importantes algoritmos do AR, baseado no método de diferença temporal que dispensa política. Dado um ambiente desconhecido e sua função de valor-ação Q, que indica a importância de se escolher uma determinada ação em um dado estado, o algoritmo aprende iterativamente a política ótima. A equação que estima o valor de Q é dada a seguir:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[r_{t+1} + \gamma max_a Q(S_{t+1},a) - Q(s_t,a_t)] \tag{2.1}$$

Onde:

- $Q(s_t,a_t)$ é o valor de utilidade da ação a_t no estado s_t,

- α é a taxa de aprendizagem $(0 \leq \alpha \leq 1)$, "um fator de ponderação das atualizações dos valores Q(s, a), cuja função é satisfazer as condições de convergência para uma política ótima em um ambiente não determinístico". (MARTINS; BIANCHI, 2007);

- r_{t+1} é o valor numérico do reforço para o estado subsequente,

- γ é a taxa de desconto temporal $(0 \leq \gamma \leq 1)$ que, equilibra recompensas imediatas e futuras. "Caso o valor de γ esteja muito perto de 0 (zero) o agente tende a considerar apenas valores imediatos de recompensa, caso os valores sejam muito perto de 1 (um) o agente tem em vista as recompensas futuras com o maior peso". (JR; BIANCHI; MATSUURA, 2007), p.02;

- $max_a\ Q(S_{t+1}, a)$ representa a ação de maior valor de utilidade possível no próximo estado S_{t+1}.

Conforme visto anteriormente, em AR o agente deve decidir quando escolher uma ação aleatoriamente (explorar) ou escolher a ação de maior valor de utilidade (explotar). A função que define as regras para esta transição de estados no *Q-Learning* é chamada de ε-*greedy* e é dada pela equação abaixo:

$$\pi(s) = \begin{cases} a_{randon} & \text{q} \leq \varepsilon \\ argmax_a Q(s_t, a_t) & caso\ contrário \end{cases} \tag{2.2}$$

Onde:

- π(s) é a função de transição para o estado s;

- a_{random} representa ações escolhidas de forma aleatória;

- q representa a taxa de exploração.

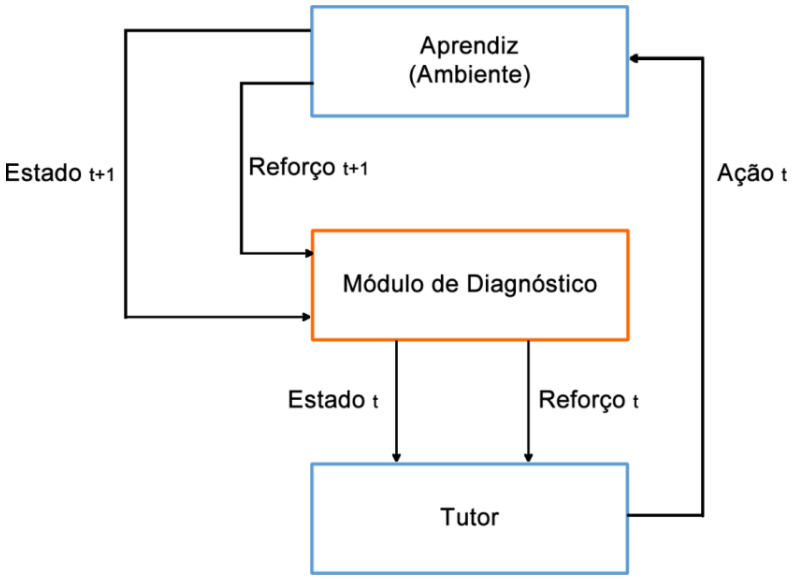
- ε é um valor numérico entre 0 e 1;

- arg $max_a\ Q(s_t a_t)$ é o argumento que representa a ação de maior valor de utilidade possível no próximo estado S_{t+1}.

2.3 Trabalhos correlatos

Motivado pela possibilidade de captar um modelo mais fidedigno do aprendiz, durante sua interação com o tutor, Guelpeli (2003) propõe a alteração da estrutura tradicional do STI. Ele introduz um módulo de diagnóstico que, utilizando o algoritmo *Q-learning*, seleciona a melhor estratégia pedagógica (maior valor numérico de utilidade) para um determinado estado de aprendizagem do aluno, levando-o a um estágio superior. O módulo proposto também é responsável pela atribuição dos reforços e atualização da matriz de utilidade que funciona como

Figura 6 – Identificação dos módulos do sistema de aprendizado por reforço para modelagem autônoma do aprendiz em tutor inteligente



Fonte – (GUELPELI; OMAR; RIBEIRO, 2004).

um sistema de "memória" do tutor, uma vez que mantém um histórico dos valores de utilidade de cada par estado-ação.

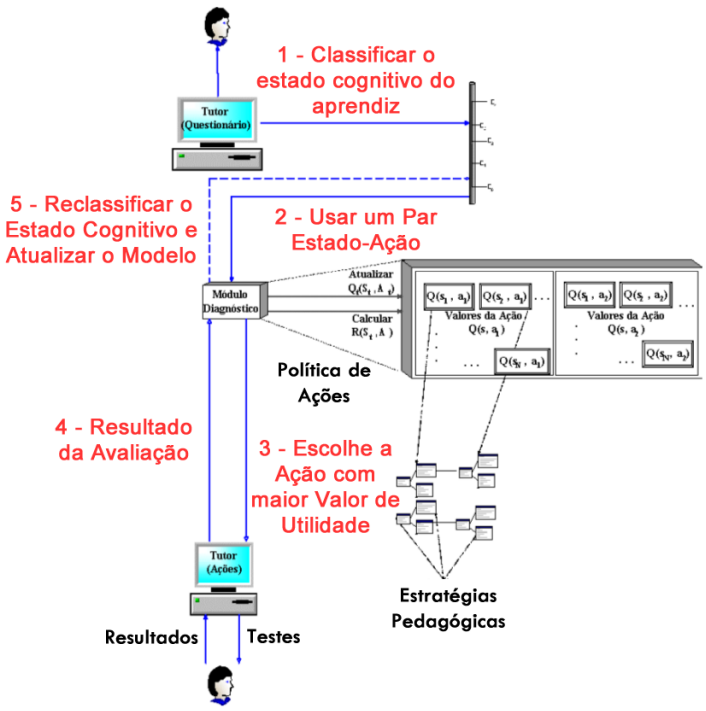
O protótipo desenvolvido por Guelpeli (2003) funciona a partir da definição de cinco estados cognitivos possíveis, que servem para classificação do aprendiz e nove ações que representam as estratégicas pedagógicas. Esses elementos formam uma matriz, localizada dentro do módulo diagnóstico proposto, que armazena os valores de utilidade de cada par estado-ação. Ela é iterativamente atualizada pela Equação 2.1 do algoritmo *Q-Learning* e possibilita a mudança de estados pelo aprendiz. Além desses dados, foram estipulados valores de reforços para cada estado de aprendizagem e os limites percentuais para explotação e exploração.

Como pode ser visto na Figura 7, o tutor classifica o aprendiz em um dos cinco estados cognitivos possíveis, com base na aplicação de um questionário. A partir da classificação obtida, escolhe a ação com maior valor de utilidade e aplica no aprendiz visando melhorar sua cognição. O tutor obtém os resultados da avaliação e reclassifica o estado cognitivo e o modelo do aluno.

Os resultados apontados por Guelpeli (2003) demonstraram que a utilização do algoritmo *Q-learning* em STI contribuiu significativamente para a modelagem autônoma do aprendiz e viabilizou a utilização do sistema em ambientes de aprendizagem online. Contudo, apontou-se a necessidade de melhoria da convergência do algoritmo que se mostrou lenta para ambientes complexos.

Bianchi (2004) propôs uma classe de algoritmos de "Aprendizado Acelerado por Heurísticas" (HAL), cujo domínio de aplicação é a navegação de robôs móveis e autônomos em ambientes com obstáculos. Segundo o autor, a classe HAL mantém todas as vantagens do AR evidenciadas na literatura e ao mesmo tempo são capazes de promover a aceleração do

Figura 7 – Sistema Tutor Inteligente com técnica de aprendizagem por reforço



Fonte – (GUELPELI; OMAR; RIBEIRO, 2004).

aprendizado.

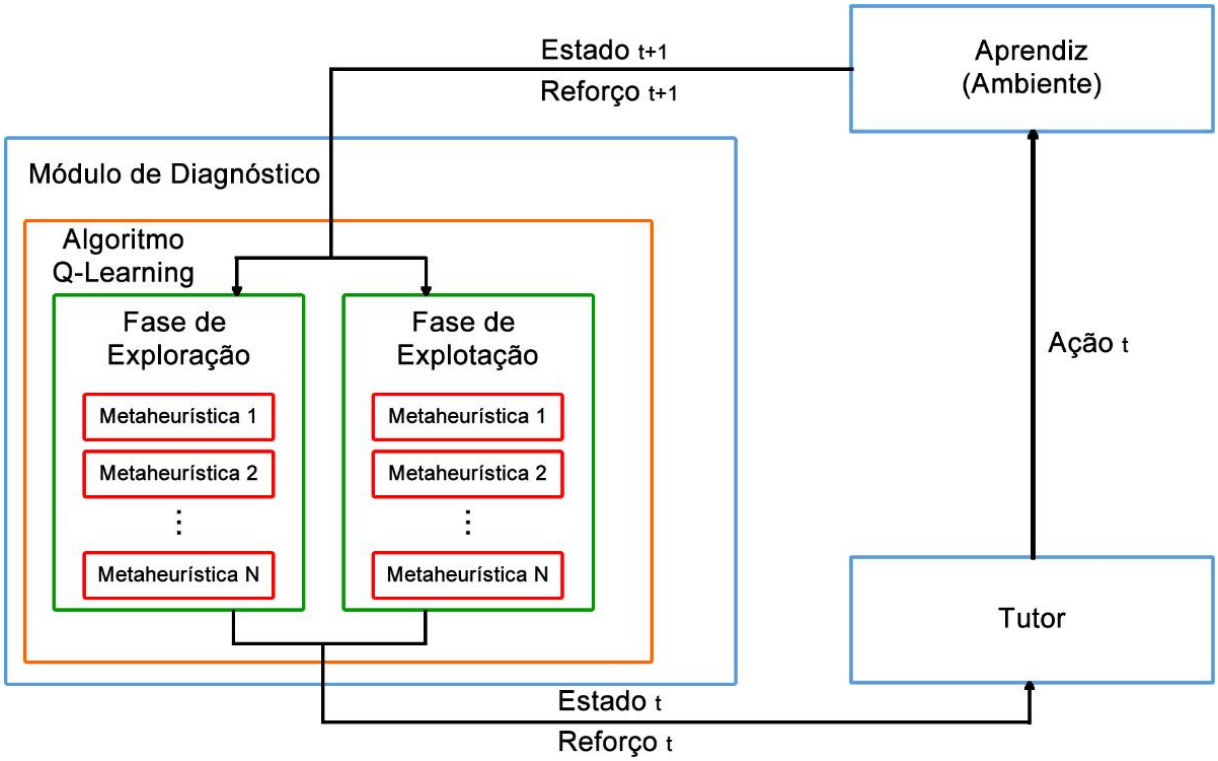
Esta classe utiliza uma função heurística que mapeia estados e ações como um número real. Este número indica a importância da escolha de uma ação a_t em um determinado estado s_t durante um intervalo de tempo, ou seja, esta função é capaz de revelar quais ações se mostraram mais relevantes para o alcance do objetivo por parte do agente.

O autor menciona a possibilidade de extração dessa "Política Heurística" logo no início do aprendizado ou a partir do instante em o agente passa a receber reforços do ambiente e utilizá-los para obter melhores resultados. Em seu trabalho, Bianchi (2004) estuda e propõe a alteração de cinco dos principais algoritmos de AR, dentre eles o *Q-Learning* que será aplicado neste trabalho.

Zhang e Liu (2008) utilizaram a metaheurística Busca Tabu para equilibrar a escolha das ações pelas fases de exploração e explotação, no problema subida de encosta.

Com a finalidade de melhorar o desempenho do algoritmo de AR e aperfeiçoar os STI's, Paiva (2016) formulou uma proposta de inclusão de metaheurísticas que pode ser aplicada tanto na fase de exploração, como na explotação, conforme demostra a Figura 8.

Figura 8 – Aceleração da convergência do algoritmo Q-Learning através de metaheurísticas



Fonte – (PAIVA, 2016)

Em seu trabalho, foram utilizadas as metaheurísticas Lista Tabu e GRASP na fase de exploração do *Q-learning*. Para fins de teste, Paiva (2016) desenvolveu um Protótipo Simulador de STI que, possibilita o acompanhamento dos resultados heurísticos em tempo real e a inclusão de novas propostas de melhorias no sistema. Para ambas as heurísticas, foram comprovadas o aumento dos valores de utilidade e aceleração de convergência do algoritmo.

3 ACELERAÇÃO DO APRENDIZADO POR REFORÇO EM STI

Neste capítulo serão apresentados os procedimentos metodológicos utilizados para a aceleração do AR em STI. Dessa foram, serão detalhados tanto o modelo como a função heurística do HAQL aplicada na função de transição de estados do *Q-learning*. Para isso serão explicitadas as técnicas de adaptação e implementação da estrutura ao Simulador de STI. Também serão descritas as configurações aplicadas na execução dos testes de desempenho.

3.1 Descrição do Modelo de Sistema Tutor Inteligente

A aceleração da convergência do algoritmo *Q-Learning* em STI's, proposta neste trabalho, foi desenvolvida com base no modelo criado por Guelpeli (2003). Dessa forma, foram mantidos todos os elementos que o compõem a fim de permitir a realização de uma análise comparativa dos resultados obtidos e aferição dos ganhos em termos de aceleração da convergência do algoritmo. A estrutura desse modelo contém os elementos descritos a seguir:

- Um conjunto composto por cinco estados cognitivos $S = \{E0, E1, E2, E3, E4\}$ para classificação do *know-how* do aprendiz. Eles são quantificados com valores que variam numa escala de 0 a 10, conforme Figura 9.
- Um conjunto de nove ações $A = \{A0, A1, A2, A3, A4, A5, A6, A7, A8, A9\}$ passíveis de serem escolhidas em cada um dos estados definidos anteriormente. Cada ação representa uma estratégia pedagógica (questionários, provas, trabalhos, dinâmicas etc.) aplicada pelo tutor no aprendiz.
- Um conjunto de cinco reforços numéricos para aplicação por parte do tutor após a transição de estados, conforme Figura 10.

Figura 9 – Quadro de modelagem de estados no protótipo STI

Estados	Valores
E0	[0, 2]
E1	]2, 4]
E2	]4, 6]
E3	]6, 8]
E4	]8, 10]

Fonte – Dados extraídos de Guelpeli (2003). Elaborado pela autora.

Figura 10 – Quadro de definição dos reforços possíveis em cada estado

Estado	Reforço	Nível Cognitivo
E0	1	Ruim
E1	3	Regular
E2	5	Bom
E3	7	Muito bom
E4	10	Excelente

Fonte – Guelpeli (2003). Elaborado pela autora.

- Duas políticas P1 e P2 que estabelecem as regras de transição entre os estados do conjunto S. Sendo que a política P2 possui regras mais rigorosas para que essa transição ocorra, enquanto que a P1 se mostra menos restritiva.
- Três modelos M1-Ruim, M2-Bom e M3-Excelente que simbolizam a capacidade de absorção de conhecimento por parte dos alunos.
- Dois tipos de ambientes que podem ser determinísticos e não determinísticos. No determinístico a política pedagógica define que a transição de estados pelo aprendiz ocorre com base em um número de acertos e erros predeterminados. No ambiente não determinístico esses valores são incertos e as transições podem ocorrer aleatoriamente segundo uma função de probabilidade.

O conjunto de estados e ações forma a matriz de utilidade Q(s, a), de tamanho 5x9, que é alimentada iterativamente e atualizada pela Equação 2.1.

Os parâmetros de Taxa de Aprendizagem α e Desconto Temporal γ encontrados na equação foram mantidos com os valores $\alpha = 0,9$ e $\gamma = 0,9$, que se mostraram mais eficazes para a obtenção dos melhores resultados, conforme comprovaram as simulações realizadas por Guelpeli (2003). É importante ressaltar que, embora esses testes tenham utilizado o algoritmo *Q-Learning* "puro", a manutenção desses valores se mostra válida para o estudo em questão. Segundo Bianchi (2004), todas as características inerentes ao AR imperam para a classe de algoritmos do Aprendizado Acelerado por Heurísticas, na qual está incluído o algoritmo HAQL.

3.2 Transição de Estados com o Algoritmo HAQL

O *Q-Learning* Acelerado por Heurísticas (HAQL) é um dos cinco algoritmos propostos por Bianchi (2004) no qual ocorre a influência de uma função heurística capaz de estimar um valor real R que representa a importância da escolha de uma ação a_t no estado s_t. Ao quantificar essas ações, o agente consegue coletar de modo autônomo um conjunto de ações que servirão

como um treinamento prévio para o início do aprendizado. Essas ações são selecionadas em detrimento das ações triviais.

As ações de treino são extraídas em tempo de execução da iteração política na qual se calcula a função valor. Ela descreve a soma dos reforços acumulados quando se persegue uma política π, a partir de um estado inicial s_t. Segundo Bianchi (2004), nesse processo, retira-se do problema a extremidade que apresenta maior variação de valores para a função valor. Esta extremidade indica no domínio de STI, aquelas ações que se mostraram mais importantes para o início do aprendizado. Isso é possível calculando a diferença entre a ação de maior valor e aquela escolhida pela política, conforme equação 3.1.

$$H(s_t, a_t) = \begin{cases} max_a \hat{Q}(s_t,\ a_t) - \hat{Q}(s_t,\ a_t) + \eta & \mathrm{a}_t = \pi^H(s_t) \\ 0 & caso\ contrário \end{cases} \tag{3.1}$$

Onde:

- $max_a \hat{Q}(s_t,\ a_t)$ representa o valor de utilidade da ação de maior valor no estado s_t,
- $\hat{Q}(s_t,\ a_t)$ o valor de utilidade da ação escolhida pela política no instante t,
- η é um parâmetro que adiciona valor a heurística.

A função heurística proposta por Bianchi (2004) é aplicada somente na função de transição de estados do *Q-Learning* que passa a incorporar uma função heurística H, conforme demostra a equação 3.2.

$$\pi(s) = \begin{cases} a_{randon} & \mathrm{q} \leq \varepsilon \\ argmax_{at}\ [\hat{Q}(s_t,\ a_t) + \xi H_t(s_t,\ a_t)] & caso\ contrário \end{cases} \tag{3.2}$$

Onde:

- $\pi(s)$ representa a política heurística,
- a_{randon} representa ações escolhidas de forma aleatória,
- $argmax_{at}\ [\hat{Q}(s_t,\ a_t) + \xi H_t(s_t,\ a_t)]$ argumento máximo da ação a no estado t, considerando a soma do valor de utilidade com a o valor heurístico,
- ξ variável que equilibra a influência da função heurística,
- q representa um valor aleatório que determina a taxa de exploração,
- ε é um valor numérico entre 0 e 1, que indica a taxa de exploração.

3.3 Adaptação dos Parâmetros Heurísticos

Considerando que o domínio de aplicação da heurística proposta por Bianchi (2004) é a navegação robótica, faz-se necessário realizar adaptações em sua estrutura visando garantir o bom desempenho do algoritmo em STI. Nesta seção serão demostrados os métodos utilizados para a seleção e adaptação dos parâmetros a serem implementados no ambiente simulador.

3.3.1 Adequação da Função Heurística

Considerando que a matriz de utilidade do *Q-Learning* é inicializada com valor 0 (zero), percebeu-se a necessidade de alteração da função de transição heurística, conforme Equação 3.3. Desse modo, a função passa a calcular o módulo dos valores de importância a fim de evitar que as ações de treino coletadas assumam números negativos que não retratem suas reais influências no contexto do problema. Assim, buscou-se garantir com esta modificação que a estrutura heurística extraída apresentasse valores válidos e representativos para o domínio da aplicação.

$$H(s_t, a_t) = \begin{cases} |max_a \hat{Q}(s_t,\ a_t) - \hat{Q}(s_t,\ a_t) + \eta| & \mathrm{a}_t = \pi^H(s_t) \\ 0 & caso\ contrário \end{cases} \tag{3.3}$$

3.3.2 Adaptação do Período de Extração da Estrutura Heurística

Por se tratar de um método que extrai informações da função valor em tempo de execução, foi imperioso determinar o período no qual ocorreria esta coleta dos valores de importância. Em Bianchi (2004) ela ocorre até o 10º episódio, porém por se tratar de um domínio diferente do trabalho originalmente proposto foram realizados testes para diferentes períodos objetivando encontrar aquele que retorna os melhores resultados.

Figura 11 – Quadro com parâmetros utilizados para os testes de extração da estrutura heurística

Parâmetro	Valor
Ambiente	Não determinístico
Política	P2
Modelo	M2 – Bom
Iterações	1500
Simulações	20
α	0,9
γ	0,9
ξ	1
η	1

Fonte – Dados extraídos de Guelpeli (2003) e Bianchi (2004).Elaborado pela autora.

A parametrização deste trabalho foi realizada com base na política P2, pois sendo ela a mais restritiva considera-se que os valores encontrados também serão admissíveis para a política menos restritiva P1.

O modelo M2 foi escolhido como base para parametrização por ser o intermediário dentre os modelos M1 e M3.

O ambiente não determinístico foi adotado em detrimento do determinístico por ser mais complexo e de difícil trato no âmbito das pesquisas.

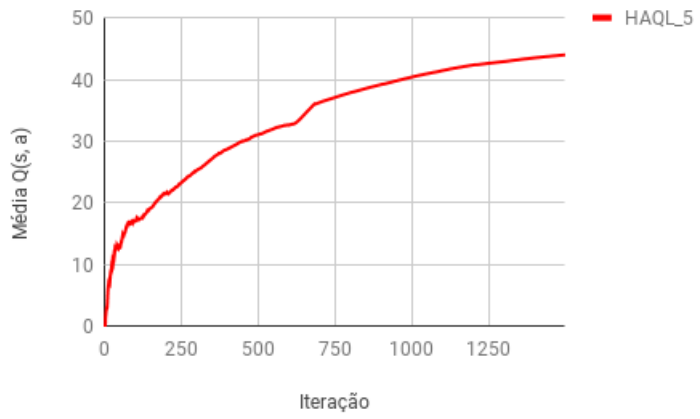
Extração heurística até a 5ª iteração

Neste experimento buscou-se avaliar o impacto da coleta dos valores de importância I(s, a) até a quinta iteração, ou seja, cinco iterações mais cedo que o trabalho original de Bianchi (2004). Na Figura 13, tem-se a parte sombreada na qual o HAQL encontra-se desativado para coleta dos valores de I(s, a) e para os quais valores de Q(s, a) são iguais a zero. O formato rugoso do gráfico após a 5ª iteração indica grande oscilação de valores que representam acertos e erros do aprendiz.

Na Figura 12 percebe-se o momento da convergência do algoritmo no eixo horizontal entre a 500ª e 1000ª iteração e no eixo vertical próximo ao valor de utilidade Q(s, a) igual a 40. Mais precisamente ao observar o arquivo com as médias dos valores tem-se que a convergência ocorre exatamente na iteração 960.

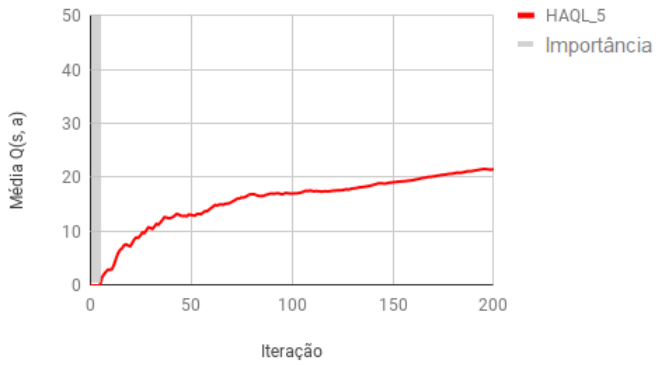
Neste teste, Q(s,a) alcançou o valor mínimo de 0 e máximo de 43,99.

Figura 12 – Gráfico do HAQL aplicado ao STI, com extração da estrutura heurística até a quinta iteração



Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

Figura 13 – Detalhe do gráfico do HAQL aplicado ao STI, com ênfase do período de extração da estrutura heurística até a quinta iteração



Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

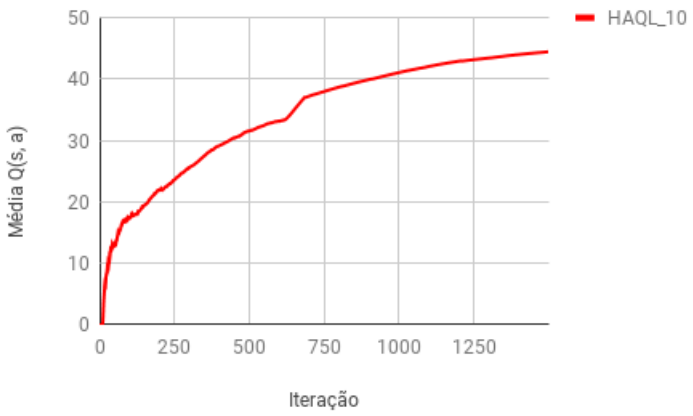
Extração heurística até a 10ª iteração

Neste experimento a coleta dos valores de I(s, a) ocorre até a décima iteração, assim como no trabalho original de Bianchi (2004). Na Figura 15, tem-se a parte sombreada na qual o HAQL encontra-se desativado para coleta dos valores de I(s, a) e para os quais valores de Q(s, a) são iguais a zero. O formato rugoso do gráfico após a 10ª iteração indica grande oscilação de valores que representam acertos e erros do aprendiz.

Na Figura 14 percebe-se o momento da convergência do algoritmo no eixo horizontal entre a 500ª e 1000ª iteração e no eixo vertical no valor de utilidade Q(s, a) igual a 40. Mais precisamente ao observar o arquivo de valores com as médias têm-se que a convergência ocorre exatamente na 906ª iteração.

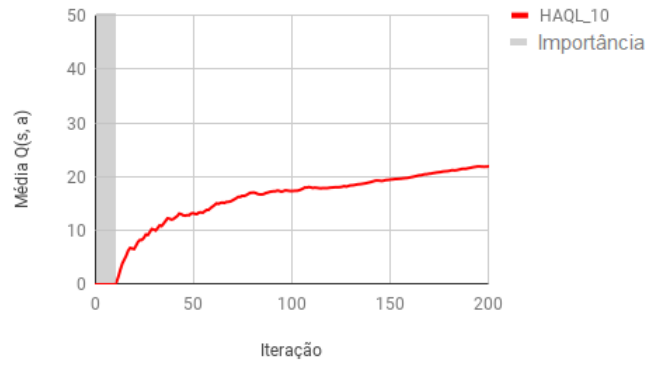
Para este teste, Q(s,a) alcançou o valor mínimo de 0 e máximo de 44,44.

Figura 14 – Gráfico do HAQL aplicado ao STI, com extração da estrutura heurística até a décima iteração



Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

Figura 15 – Detalhe do gráfico do HAQL aplicado ao STI, com ênfase do período de extração da estrutura heurística até a décima iteração



Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

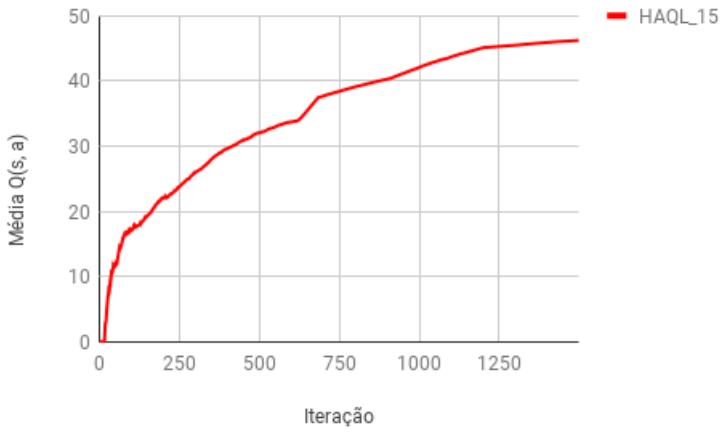
Extração heurística até a 15ª iteração

Neste experimento a coleta dos valores de I(s, a) ocorre até a 15ª iteração, cinco passos mais tarde que o trabalho original de Bianchi (2004). Na Figura 17, tem-se a parte sombreada na qual o HAQL encontra-se desativado para coletar os valores de I(s, a), nessa fase os valores de Q(s, a) são iguais a zero. O formato rugoso do gráfico após a 15ª iteração indica grande oscilação de valores que representam acertos e erros do aprendiz.

Na Figura 16 percebe-se o momento da convergência do algoritmo no eixo horizontal entre a 500ª e 1000ª iteração e no eixo vertical no valor de utilidade Q(s, a) próximo a 40. Mais precisamente ao observar o arquivo de valores com as médias têm-se que a convergência ocorre exatamente na 874ª iteração.

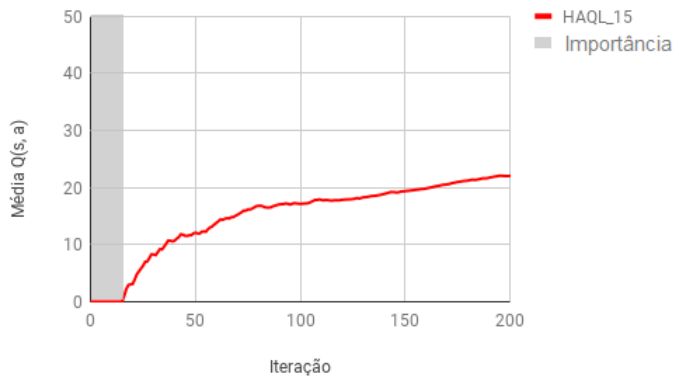
Para este teste, Q(s, a) alcançou o valor mínimo de 0 e máximo de 46,26.

Figura 16 – Gráfico do HAQL aplicado ao STI, com extração da estrutura heurística até a décima quinta iteração



Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

Figura 17 – Detalhe do gráfico do HAQL aplicado ao STI, com ênfase do período de extração da estrutura heurística até a décima quinta iteração



Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

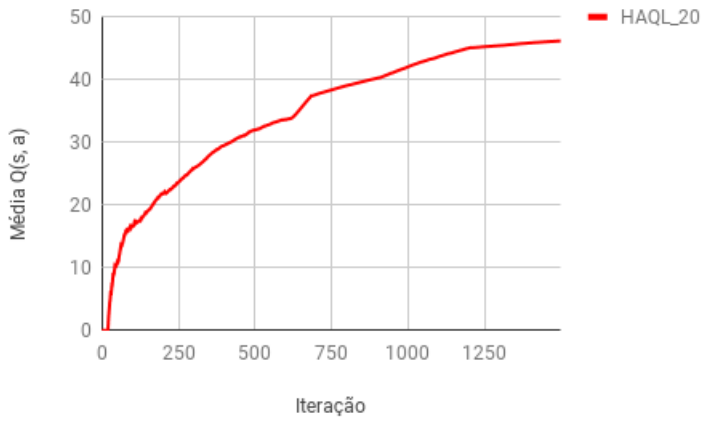
Extração heurística até a 20ª iteração

Neste experimento a extração dos valores heurísticos ocorre até a 20ª iteração, ou seja, dez passos mais tarde que o trabalho original de Bianchi (2004). Na Figura 19, tem-se a parte sombreada na qual o HAQL encontra-se desativado para coletar os valores de I(s, a), nessa fase os valores de Q(s, a) são iguais a zero. O formato rugoso do gráfico após a 20ª iteração indica grande oscilação de valores que representam acertos e erros do aprendiz.

Na Figura 18 percebe-se o momento da convergência do algoritmo no eixo horizontal entre a 500ª e 1000ª iteração e no eixo vertical no valor de utilidade Q(s, a) igual a 40. Mais precisamente ao observar o arquivo de valores com as médias têm-se que a convergência ocorre exatamente na 878ª iteração.

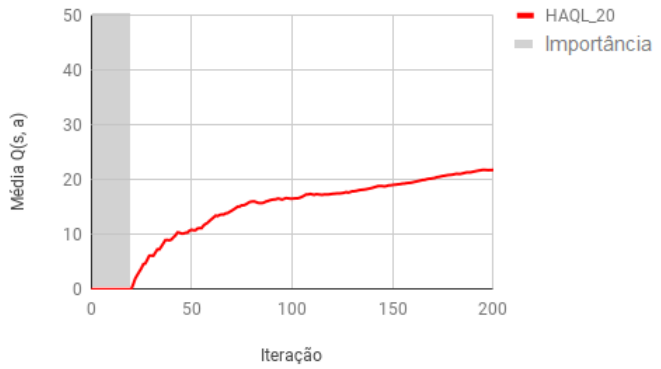
Para este teste, Q(s, a) alcançou o valor mínimo de 0 e máximo de 46,19.

Figura 18 – Gráfico do HAQL aplicado ao STI, com extração da estrutura heurística até a vigésima iteração



Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

Figura 19 – Detalhe do gráfico do HAQL aplicado ao STI, com ênfase do período de extração da estrutura heurística até a vigésima iteração



Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

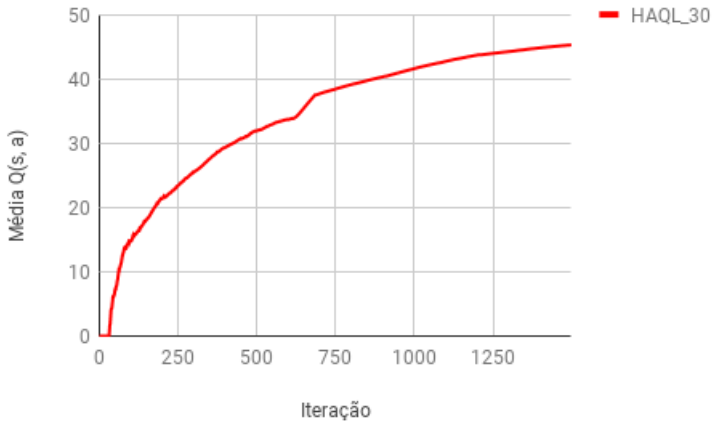
Extração heurística até a 30ª iteração

Neste experimento a extração dos valores I(s, a) ocorre até a 30ª iteração, ou seja, vinte passos mais tarde que o trabalho original de Bianchi (2004). Na Figura 21, tem-se a parte sombreada na qual o HAQL encontra-se desativado para coletar os valores de I(s, a), nessa fase os valores de Q(s, a) são iguais a zero. O formato rugoso do gráfico após a 30ª iteração indica grande oscilação de valores que representam acertos e erros do aprendiz.

Na Figura 20 percebe-se o momento da convergência do algoritmo no eixo horizontal entre a 500ª e 1000ª iteração e no eixo vertical no valor de utilidade Q(s, a) igual a 40. Mais precisamente ao observar o arquivo de valores com as médias têm-se que a convergência ocorre exatamente na 874ª iteração.

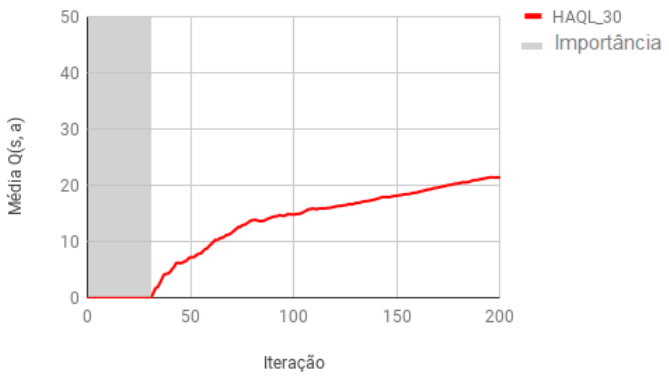
Neste teste, Q(s, a) alcançou o valor mínimo de 0 e máximo de 45,27.

Figura 20 – Gráfico do HAQL aplicado ao STI, com extração da estrutura heurística até a trigésima iteração



Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

Figura 21 – Detalhe do gráfico do HAQL aplicado ao STI, com ênfase do período de extração da estrutura heurística até a trigésima iteração



Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

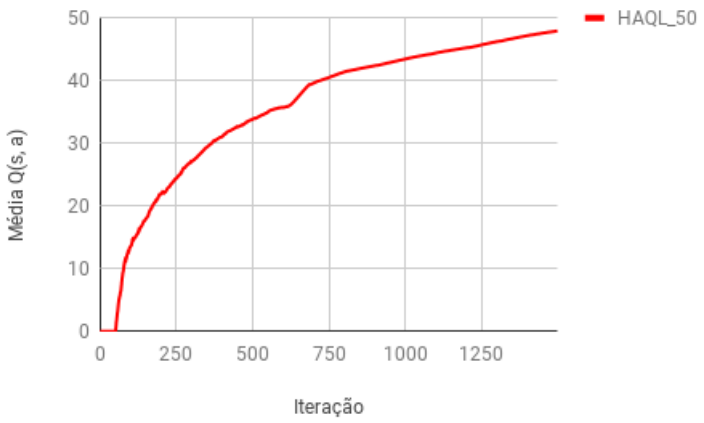
Extração heurística até a 50ª iteração

Neste experimento a extração dos valores heurísticos ocorre até a 50ª iteração, ou seja, quarenta passos mais tarde que o trabalho original de Bianchi (2004). Na Figura 23, tem-se a parte sombreada na qual o HAQL encontra-se desativado para extrair os valores de I(s, a), nessa fase os valores de Q(s, a) são iguais a zero. O formato rugoso do gráfico após a 50ª iteração indica que ainda há grande oscilação de valores que representam acertos e erros do aprendiz.

Na Figura 22 percebe-se o momento da convergência do algoritmo no eixo horizontal entre a 500ª e 1000ª iteração e no eixo vertical no valor de utilidade Q(s, a) igual a 40. Mais precisamente ao observar o arquivo de valores com as médias têm-se que a convergência ocorre exatamente na 720ª iteração.

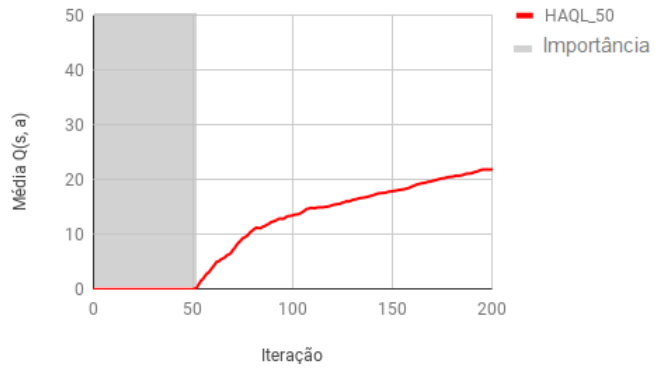
Para este teste, Q(s, a) alcançou o valor mínimo de 0 e máximo de 47,95

Figura 22 – Gráfico do HAQL aplicado ao STI, com extração da estrutura heurística até a quinquagésima iteração



Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

Figura 23 – Detalhe do gráfico do HAQL aplicado ao STI, com ênfase do período de extração da estrutura heurística até a quinquagésima iteração



Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

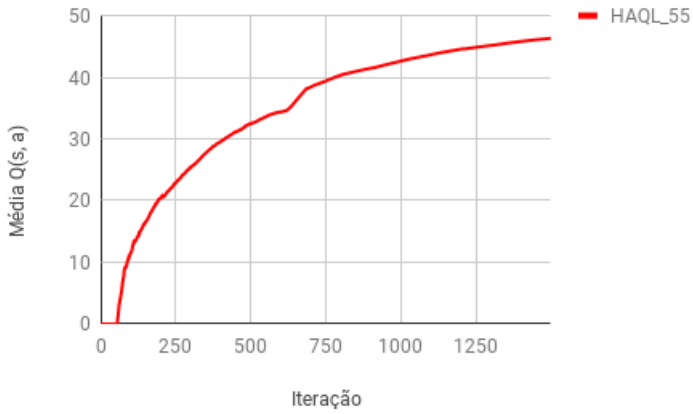
Extração heurística até a 55ª iteração

Neste experimento a extração dos valores heurísticos ocorre até a 55ª iteração, ou seja, quarenta e cinco passos mais tarde que o trabalho original de Bianchi (2004). Na Figura 25, tem-se a parte sombreada na qual o HAQL encontra-se desativado para extrair os valores de I(s, a), nessa fase os valores de Q(s, a) são iguais a zero. Percebe-se que o formato do gráfico após a 55ª iteração não é tão rugoso indicando que não há grande oscilação de valores.

Na Figura 24 percebe-se o momento da convergência do algoritmo no eixo horizontal entre a 500ª e 1000ª iteração e no eixo vertical no valor de utilidade Q(s, a) igual a 40. Mais precisamente ao observar o arquivo de valores com as médias têm-se que a convergência ocorre exatamente na 779ª iteração.

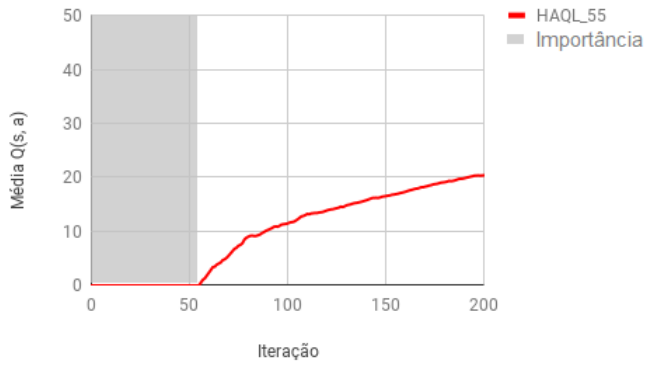
Para este teste, Q(s, a) alcançou o valor mínimo de 0 e máximo de 46,33.

Figura 24 – Gráfico do HAQL aplicado ao STI, com extração da estrutura heurística até a quinquagésima quinta iteração



Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

Figura 25 – Detalhe do gráfico do HAQL aplicado ao STI, com ênfase do período de extração da estrutura heurística até a quinquagésima quinta iteração

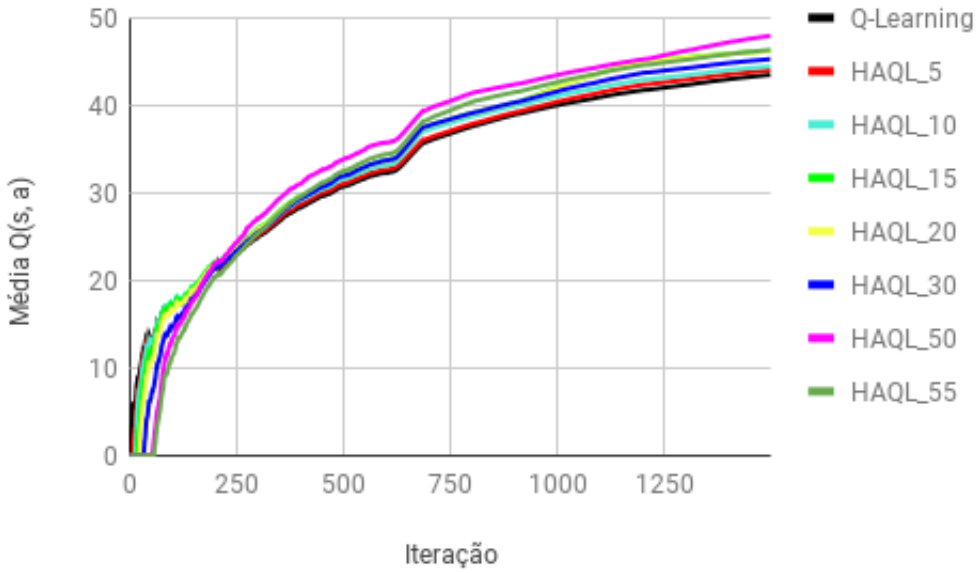


Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

Análise de Extração da Estrutura Heurística

Com base nos testes realizados e análise da Figura 26 conclui-se que o HAQL com extração da estrutura heurística até a 50ª iteração trouxe os melhores resultados para o domínio em estudo.

Figura 26 – Gráfico comparativo do HAQL, com extração da estrutura heurística em diferentes períodos



Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

3.3.3 Adaptação do parâmetro η

Analisando a estrutura geral da heurística proposta 3.1, observa-se a existência do parâmetro adicional η que pode assumir valores no intervalo de [0, +[. No domínio de navegação

robótica, Bianchi (2004) utiliza $\eta = 1$. Mais uma vez, faz-se necessário a realização de testes adicionais com valores variáveis para mais e para menos, a fim de verificar o que melhor se adéqua ao propósito. Lembrando que quanto maior for o valor de η, maior será a prevalência da heurística em relação a estratégia de explotação do *Q-Learning* e vice-versa.

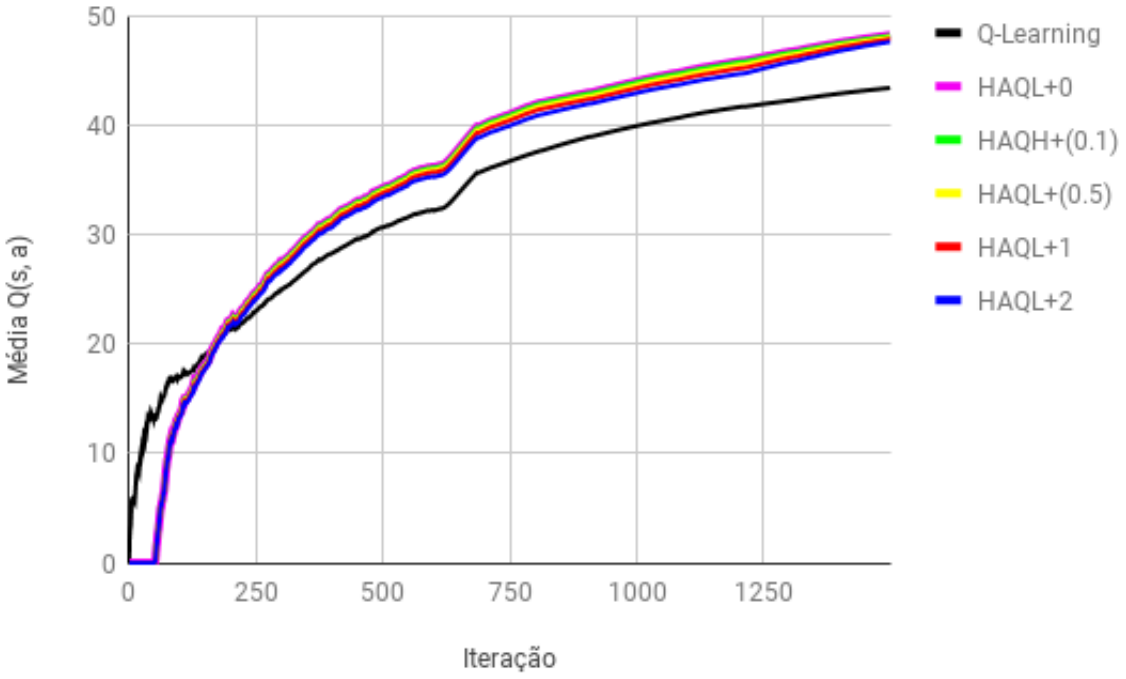
Para este experimento, variou-se o parâmetro η com os valores 0; 0,5; 1 e 2. Observou-se que quanto maior o valor de η, menor o desempenho. Percebe-se pela Figura 28 que os melhores valores de Q, foram obtidos para $\eta = 0$.

Figura 27 – Quadro com parâmetros utilizados para os testes de definição de η

Parâmetro	Valor
Ambiente	Não determinístico
Política	P2
Modelo	M2 – Bom
Iterações	1500
Simulações	20
α	0,9
γ	0,9
ξ	1
Fase de coleta heurística	Até 50ª iteração

Fonte – Dados extraídos de Guelpeli (2003). Elaborado pela autora.

Figura 28 – Gráfico comparativo do HAQL, com variação do parâmetro η



Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

3.3.4 **Adequação do parâmetro ξ**

O parâmetro ξ encontrado na Equação 3.2 multiplica a influência da heurística na função de transição e pode assumir valores no intervalo [0, 1]. Quanto maior o valor de ξ, maior a influência da função heurística e vice-versa.

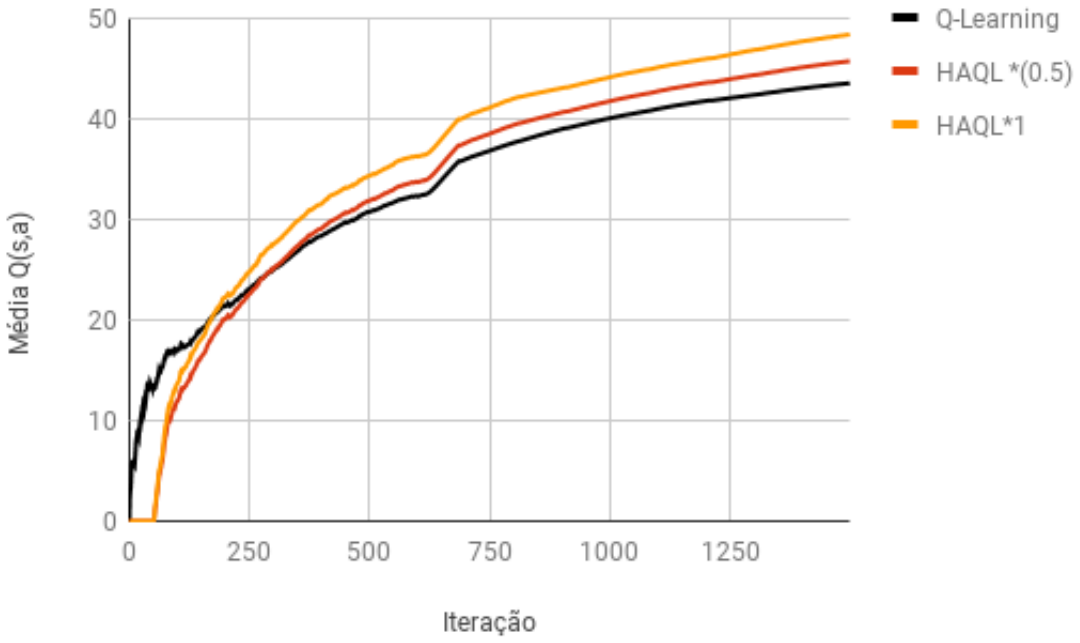
Figura 29 – Quadro com parâmetros utilizados para os testes de definição de ξ

Parâmetro	Valor
Ambiente	Não determinístico
Política	P2
Modelo	M2 – Bom
Iterações	1500
Simulações	20
α	0,9
γ	0,9
Fase de coleta heurística	Até 50ª iteração
η	0

Fonte – Dados extraídos de Guelpeli (2003) e Bianchi (2004). Elaborado pela autora.

Com este experimento observou-se que a diminuição do valor do parâmetro ξ

Figura 30 – Gráfico comparativo do HAQL, com variação do parâmetro ξ



Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

produziu uma perda de desempenho do algoritmo HAQL, conforme demonstrado na Figura 30.

Para $\xi = 1$, Q(s, a) alcançou o valor mínimo de 0 e máximo de 48,31. Enquanto que para $\xi = 0.5$, Q(s, a) alcançou o valor mínimo de 0 e máximo de 45,67. Dessa forma, foi mantido o parâmetro $\xi = 1$, que levou aos melhores resultados.

3.4 Simulador

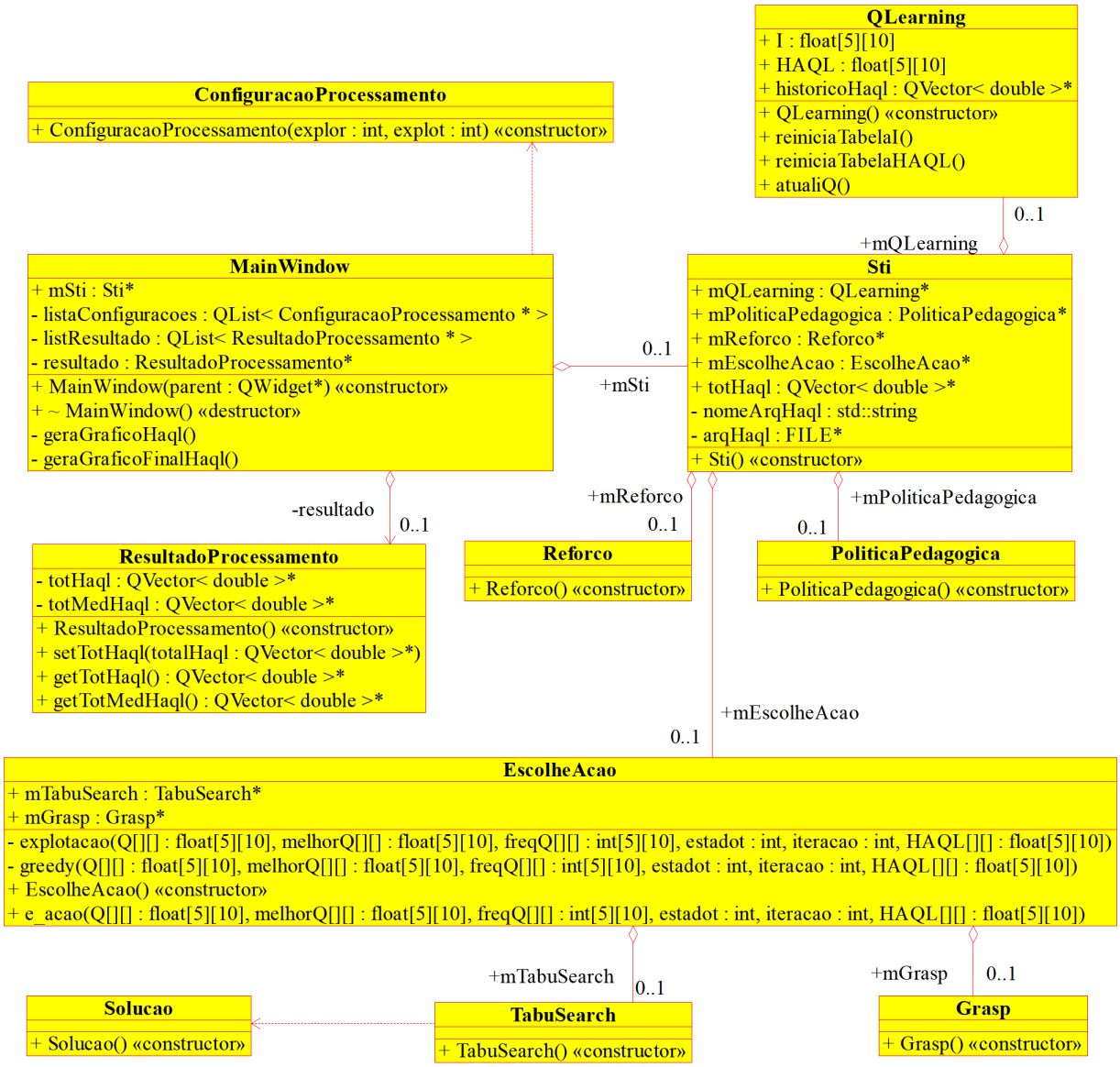
Esta seção tem por objetivo descrever as etapas de implementação do algoritmo HAQL no ambiente simulador de STI, proposto por Paiva (2016), com os parâmetros ajustados na seção anterior. Pretende ainda descrever as configurações necessárias para a execução dos testes de desempenho.

3.4.1 Implementação do Algoritmo HAQL

A implementação foi realizada utilizando os conceitos do paradigma de programação Orientado a Objetos e a linguagem C++, no ambiente de desenvolvimento *QT Creator*.

As principais alterações foram realizadas nos arquivos *Headers− >qlearnig.h* e *Sources− >qlearning.cpp*. Na classe *QLearning* foram criadas as funções públicas responsáveis por reiniciar as matrizes de importância I(s, a) *void reiniciaTabelaI();* e de utilidade do HAQL *void reiniciaTabelaHAQL();*. A cada simulação, estas funções inicializam as matrizes com o valor zero para que não haja sobreposição de valores.

Figura 31 – Diagrama de Classes das modificações realizadas no simulador de STI.



Fonte – Dados extraídos do simulador de STI. Elaborado pela autora.

No cabeçalho foram declaradas as tabelas HAQL *float HAQL [5][10];* e de importância *float I [5][10];*. Elas são responsáveis pelo armazenamento dos valores de utilidade atualizados iterativamente pela Equação 2.1.

Na Figura 32 pode-se observar o desenvolvimento das estruturas necessárias para o cálculo dos valores de importância, na função *AtualizaQ*, conforme Função Heurística dada pela Equação 3.3. Na sequência, também se definiu o cálculo dos valores Q(s, a) de acordo com o estabelecido pela Equação 3.2.

A Figura 33 exibe as funções responsáveis por reiniciar as tabelas I(s, a) e HAQL.

Figura 32 – Código - C++

```
void QLearning::atualiQ(int &estado, int &estadot, int &acao,
    int &acao_Max, float &ref, float &refmed, int iteracao)
{
    //Calcula o reforco descontado
    //Calculo do algoritmo Q-Learning
    Q[estado][acao] += this->alfa * (ref + this->gama * Q[
        estadot][acao_Max] - Q[estado][acao]);

    //Calcula o valor da Tabela de Importancia

    if (iteracao<=50){

            I[estado][acao] = fabs(Q[estado][acao_Max]- Q[estado
                ][acao]+ 0);

    }

    //Caclular o valor da Tabela HAQL
    if (iteracao>50){
        HAQL[estado][acao] = Q[estado][acao] + I[estado][acao];
    }

    //Calcula o refor o m dio descontado
    refmed+=(ref * this->gama);

    //Atualiza a tabela com os maiores valores encontrados para
        o estado-a   o
    if(Q[estado][acao] > melhorQ[estado][acao]){
        melhorQ[estado][acao] = Q[estado][acao];
    }
}
```

Fonte – Dados extraídos do simulador de STI.

Figura 33 – Código - C++

```
/**
 * @brief QLearning::reiniciaTabelaI reinicia todas as c lulas
    da tabela para valor inicial 0
 */
void QLearning::reiniciaTabelaI()
{
    for(int l = 0; l < 5; l++)
    {
        for(int c = 0; c < 10; c++)
        {
            this->I[l][c] = 0.0;
        }
    }
}

/**
 * @brief QLearning::reiniciaTabelaHAQL reinicia todas as
    c lulas da tabela para valor inicial 0
 */
void QLearning::reiniciaTabelaHAQL()
{
    for(int l = 0; l < 5; l++)
    {
        for(int c = 0; c < 10; c++)
        {
            this->HAQL[l][c] = 0.0;
        }
    }
}
```

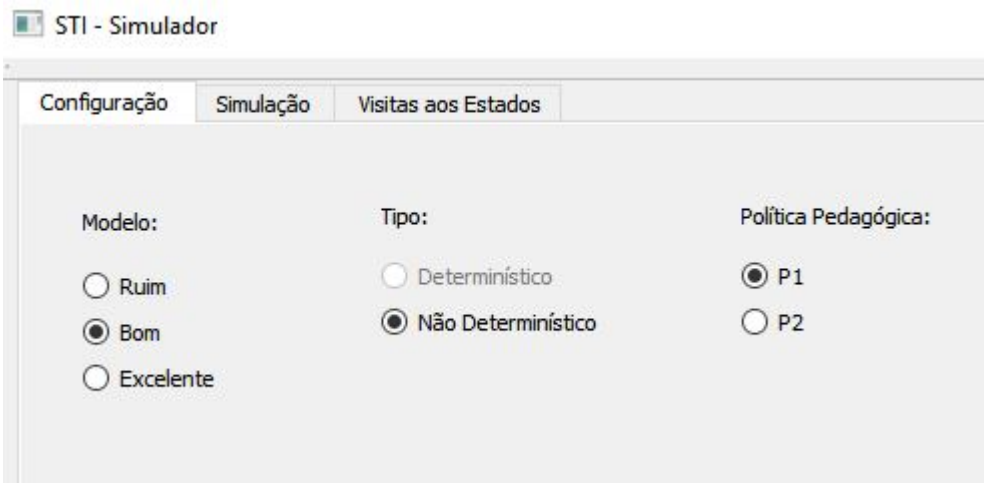
Fonte – Dados extraídos do simulador de STI.

3.4.2 **Configurações do Simulador**

A interface do software simulador de STI é composto por três telas. Na **tela de Configuração** é possível definir os atributos **Modelo do Aprendiz**, **Tipo de Ambiente**, **Política Pedagógica**, **Número de Iterações**, **Número de Simulações**, **Tipo de Simulação**, **Estratégias de Exploração** e **Estratégias de Explotação**.

O **Modelo do Aprendiz** agrupa os perfis de alunos conforme a capacidade de absorção de conteúdo. Assim tem-se do menor para o maior os modelos **M1-Ruim**, **M2-Bom** e **M3-Excelente**.

Figura 34 – Tela do Simulador - Configurações do Modelo do Aprendiz, Tipo de Ambiente e Política Pedagógica

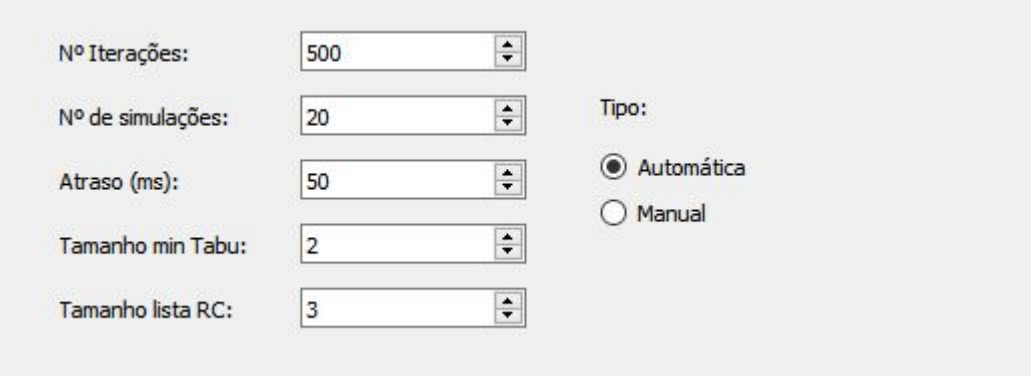


Fonte – Simulador de STI.

O **Tipo de Ambiente** pode ser **Determinístico** ou **Não Determinístico**. No determinístico a transição de estados pelo aprendiz se dá conforme um número de acertos e erros preestabelecidos pela Política Pedagógica. No ambiente não determinístico não é possível predizer o comportamento do aprendiz, pois ele ocorre conforme uma função probabilística definida pela Política Pedagógica.

A **Política Pedagógica** estabelece regras para que o aprendiz mude de um estado para o outro. No simulador estão disponíveis as políticas **P1** e **P2**. Na política P1 estas regras se mostram menos rigorosas e o aprendiz pode mudar de estados com um número de acertos menor que o estabelecido na política P2 que é mais restritiva.

Figura 35 – Tela do Simulador - Configurações do Número de Iterações e Simulações



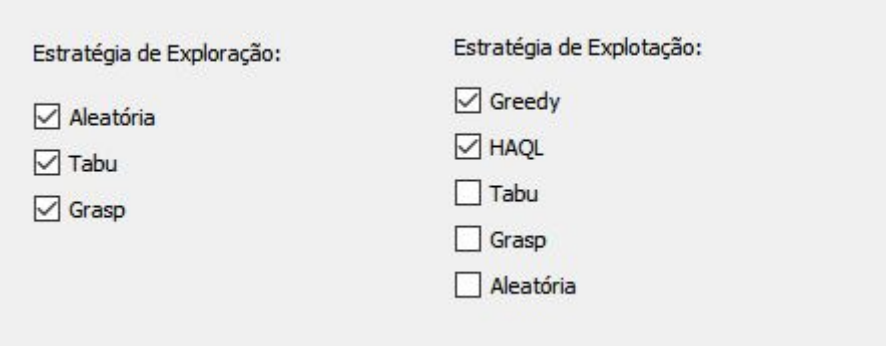
Fonte – Simulador de STI.

O **Número de Iterações** está relacionado à quantidade de troca de mensagens realizadas entre tutor e aprendiz. Como a comunicação entre eles ocorre a partir da aplicação de questionários, o número de iterações representa a quantidade de questionários aplicados pelo tutor no aprendiz.

O **Número de Simulações** estipula a quantidade de vezes que cada teste será repetido. Ao final calcula-se a média dos valores Q(s,a) para cada iteração.

No **Tipo de Simulação** pode-se escolher entre a manual, onde é possível visualizar os dados de cada iteração ou a automática.

Figura 36 – Tela do Simulador - Configurações das Estratégias de Exploração e Explotação



Fonte – Simulador de STI.

Em **Estratégias de Exploração** é possível selecionar os métodos pelos quais as ações serão escolhidas quando o Tutor estiver na fase de Exploração. Na estratégia **Aleatória** todas as ações tem a mesma probabilidade de serem escolhidas. Nas estratégias **Tabu** e **Grasp** a escolha da ação em cada estado ocorre conforme o comportamento de cada metaheurística.

Em **Estratégias de Explotação** é possível selecionar por quais métodos as ações serão escolhidas quando o Tutor estiver na fase de explotação. Na estratégia **Greedy**, a ação de maior valor em cada estado será sempre a escolhida pelo tutor. Nas estratégias **Tabu**, **Grasp** e **HAQL** a escolha da ação em cada estado ocorre conforme o comportamento de cada metaheurística.

A **Tela de Simulação** permite o acompanhamento das emulações em tempo de execução por meio de gráficos que exibem os resultados parciais dos Reforços, Transições de Estados e Valores de Utilidade Q(s,a) e tabelas que apresentam aos valores numéricos de Q(s, a) do *Q-learning* e HAQL e ainda os valores heurísticos I(s, a) extraídos em tempo de execução.

Na **Tela de Visita aos Estados** são exibidos ao final das simulações, gráficos de pizza com os valores percentuais de visita para cada estado. Esses dados não fazem parte do escopo deste trabalho e, portanto, não foram utilizados.

3.4.3 **Arquivos de Saída**

Ao final de cada teste são gerados quatro arquivos no formato CSV, com os resultados das médias dos valores Q(s, a) para cada par de estratégias possíveis. Esses arquivos fornecem os dados necessários para a confecção de gráficos e tabelas para a comparação de desempenho de cada estratégia, conforme os pares a seguir:

1. (Aleatória, Greedy),

2. (Aleatória, HAQL),
3. (Tabu, Greedy),
4. (Grasp, Greedy).

A Tabela 37 resume as configurações adotadas para cada um dos nove testes realizados.

Figura 37 – Quadro de configurações do Simulador

Nº	Modelo Aprendiz	Tipo Ambiente	Política Pedagógica	Nº Iterações	Nº Simulações	Estratégia Exploração	Estratégia Explotação
1ª	Ruim	Não determinístico	P1	500	20	Aleatória Tabu Grasp	Greedy HAQL
2ª	Ruim	Não determinístico	P1	1000	20	Aleatória Tabu Grasp	Greedy HAQL
3ª	Ruim	Não determinístico	P1	3000	20	Aleatória Tabu Grasp	Greedy HAQL
4ª	Bom	Não determinístico	P1	500	20	Aleatória Tabu Grasp	Greedy HAQL
5ª	Bom	Não determinístico	P1	1000	20	Aleatória Tabu Grasp	Greedy HAQL
6ª	Bom	Não determinístico	P1	3000	20	Aleatória Tabu Grasp	Greedy HAQL
7ª	Excelente	Não determinístico	P1	500	20	Aleatória Tabu Grasp	Greedy HAQL
8ª	Excelente	Não determinístico	P1	1000	20	Aleatória Tabu Grasp	Greedy HAQL
9ª	Excelente	Não determinístico	P1	30000	20	Aleatória Tabu Grasp	Greedy HAQL

Fonte – Elaborado pela autora.

4 RESULTADOS

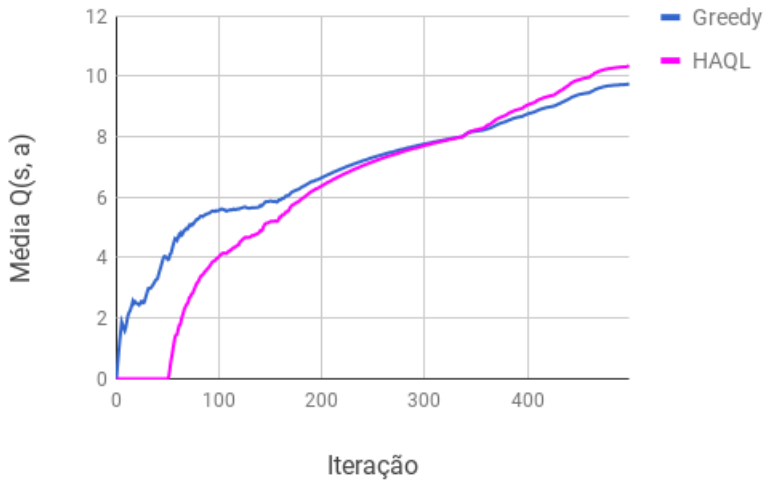
Este Capítulo apresenta os resultados dos testes realizados em ambiente não determinístico, utilizando a política pedagógica P1- menos restritiva e os modelos M1 – Ruim, M2 – Bom e M3 – Excelente. Para cada um desses modelos foram realizadas 20 simulações com 500, 1000 e 3000 iterações. As médias dos valores Q(s, a) foram analisadas graficamente e comparadas com os trabalhos de Guelpeli (2003) e Paiva (2016) para aferição de desempenho.

4.1 Modelo de Aprendiz M1 - Ruim

Teste com 500 iterações

A Figura 38 mostra a comparação de desempenho das heurísticas **Greedy e HAQL** aplicadas como estratégia de explotação do algoritmo *Q-Learning* em STI. O resultado foi obtido com base na média dos valores de utilidade Q(s, a). Para esse teste foram realizadas **20 simulações**, com **500 iterações** cada, utilizando o modelo de aprendiz **M1 – Ruim**. Observa-se pelo experimento que a heurística HAQL obteve os melhores resultados de Q(s,a) a partir da **345ª iteração**.

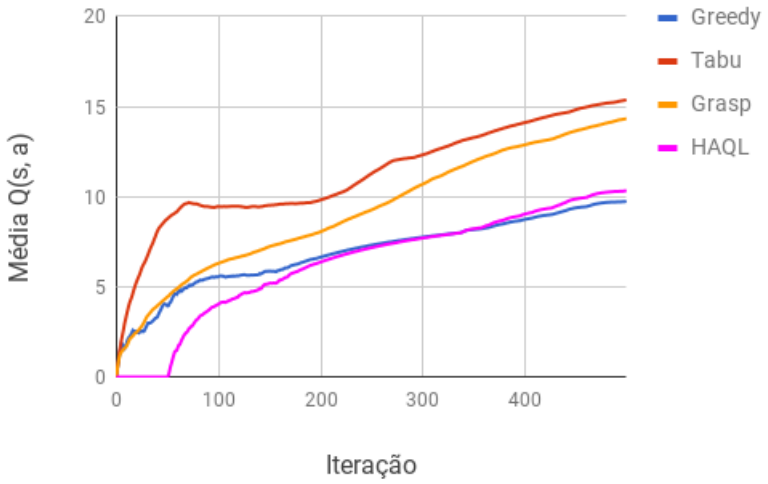
Figura 38 – Comparação das estratégias Greedy e HAQL - Modelo M1 - 500 iterações



Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

A Figura 39 mostra a comparação de desempenho das heurísticas **Greedy e HAQL** aplicadas como estratégia de explotação, com as metaheurísticas **Tabu e Grasp** aplicadas como estratégia de exploração do algoritmo *Q-Learning* em STI. O resultado foi calculado com base na média dos valores de utilidade Q(s, a). Para este teste foram realizadas **20 simulações**, com **500 iterações** cada, utilizando o modelo de aprendiz **M1 – Ruim**. Observa-se pelo experimento que a heurística HAQL obteve o terceiro melhor resultado a partir da **345ª iteração**.

Figura 39 – Comparação das estratégias Greedy, Tabu, Grasp e HAQL - Modelo M1 - 500 iterações

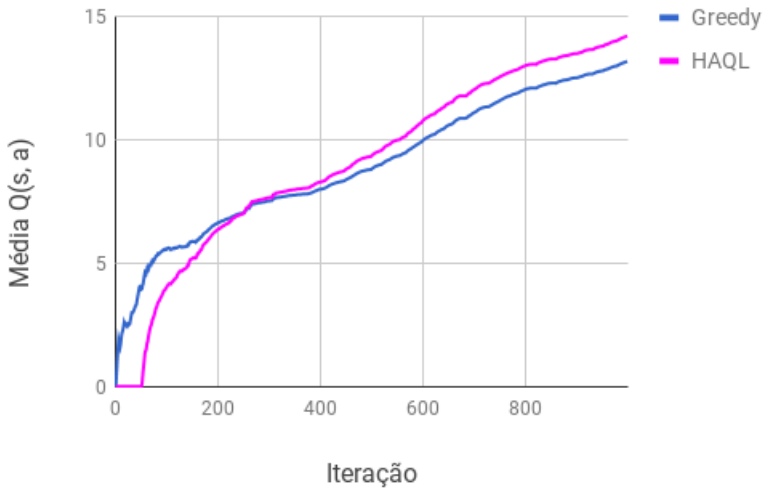


Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

Teste com 1000 iterações

A Figura 40 mostra a comparação de desempenho das heurísticas **Greedy e HAQL** aplicadas como estratégia de explotação do algoritmo *Q-Learning* em STI. O resultado foi calculado com base na média dos valores de utilidade Q(s, a). Para este teste foram realizadas **20 simulações**, com **1000 iterações** cada, utilizando o modelo de aprendiz **M1 – Ruim**. Observa-se pelo experimento que a heurística HAQL obteve os melhores resultados de Q(s,a) a partir da **252ª iteração**.

Figura 40 – Comparação das estratégias Greedy e HAQL - Modelo M1 - 1000 iterações

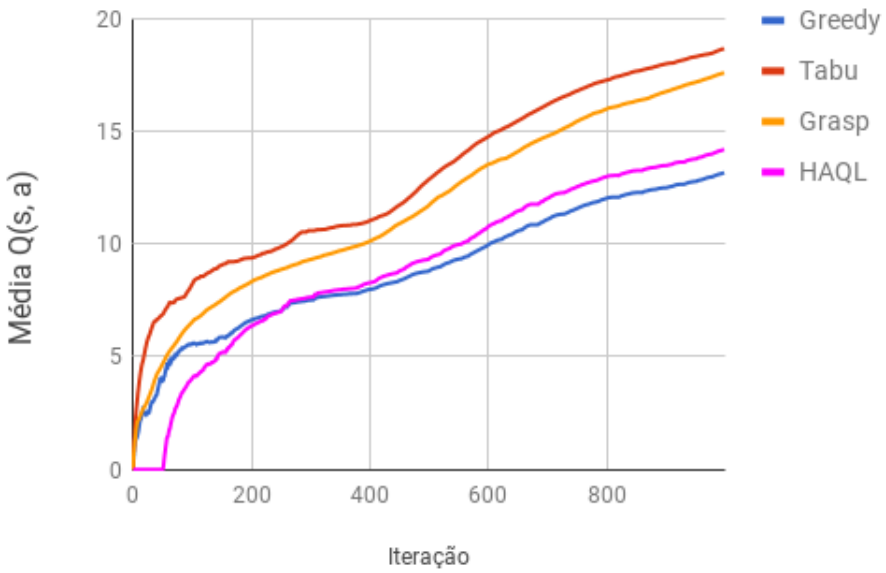


Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

A Figura 41 mostra a comparação de desempenho das heurísticas **Greedy e Tabu**

aplicadas como estratégia de explotação, com as metaheurísticas **Grasp e HAQL** aplicadas como estratégia de exploração do algoritmo *Q-Learning* em STI. O resultado foi calculado com base na média dos valores de utilidade Q(s, a). Para este teste foram realizadas **20 simulações**, com **1000 iterações** cada, utilizando o modelo de aprendiz **M1 – Ruim**. Observa-se pelo experimento que a heurística HAQL obteve o terceiro melhor resultado a partir da **252ª iteração**.

Figura 41 – Comparação das estratégias Greedy, Tabu, Grasp e HAQL - Modelo M1 - 1000 iterações



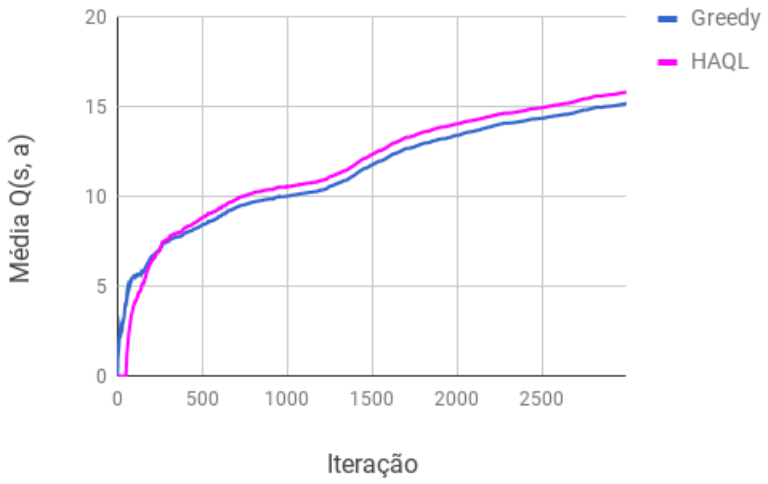
Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

Teste com 3000 iterações

A Figura 42 mostra a comparação de desempenho das heurísticas **Greedy e HAQL** aplicadas como estratégia de explotação do algoritmo *Q-Learning* em STI. O resultado foi calculado com base na média dos valores de utilidade Q(s, a). Para este teste foram realizadas **20 simulações**, com **3000 iterações** cada, utilizando o modelo de aprendiz **M1 – Ruim**. Observa-se pelo experimento que a heurística HAQL obteve os melhores resultados de Q(s,a) a partir da **252ª iteração**.

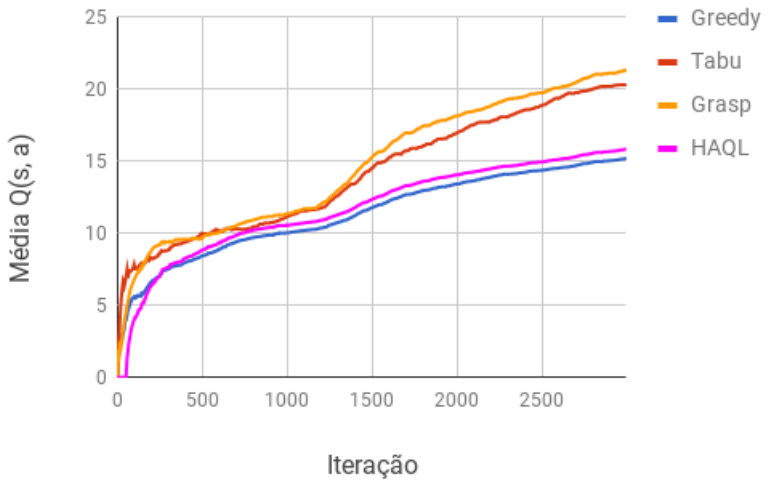
A Figura 43 mostra a comparação de desempenho das heurísticas **Greedy e HAQL** aplicadas como estratégia de explotação, com as metaheurísticas **Tabu e Grasp** aplicadas como estratégia de exploração do algoritmo *Q-learning* em STI. O resultado foi calculado com base na média dos valores de utilidade Q(s, a). Para este teste foram realizadas **20 simulações**, com **3000 iterações** cada, utilizando o modelo de aprendiz **M1 – Ruim**. Observa-se pelo experimento que a heurística HAQL obteve o terceiro melhor resultado a partir da **252ª iteração**.

Figura 42 – Comparação das estratégias Greedy e HAQL - Modelo M1 - 3000 iterações



Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

Figura 43 – Comparação das estratégias Greedy, Tabu, Grasp e HAQL - Modelo M1 - 3000 iterações



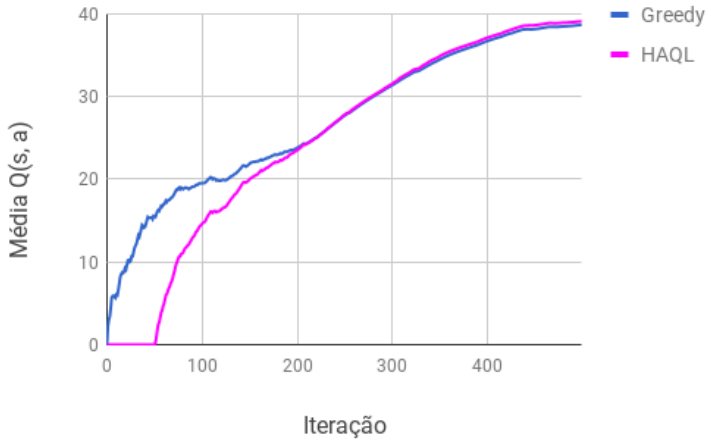
Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

4.2 Modelo de Aprendiz M2 - Bom

Teste com 500 iterações

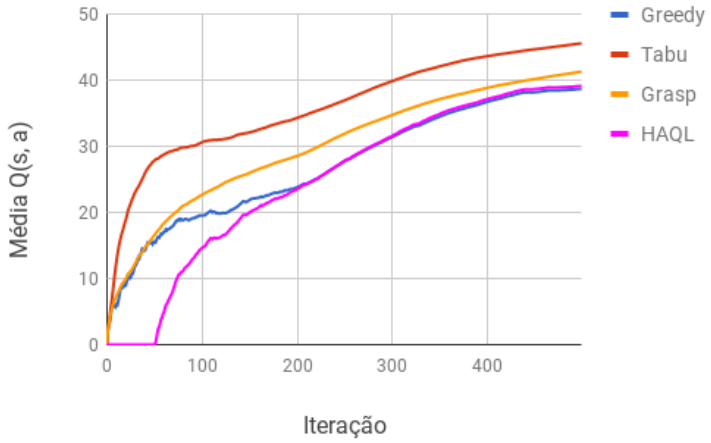
A Figura 44 mostra a comparação de desempenho das heurísticas **Greedy e HAQL**, aplicadas como estratégia de explotação do algoritmo *Q-Learning* em STI. O resultado foi calculado com base na média dos valores de utilidade Q(s, a). Para este teste foram realizadas **20 simulações**, com **3000 iterações** cada, utilizando o modelo de aprendiz **M2 – Bom**. Observa-se pelo experimento que a heurística HAQL obteve os melhores resultados de Q(s,a) a partir da **232ª iteração**.

Figura 44 – Comparação das estratégias Greedy e HAQL - Modelo M2 - 500 iterações



Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

Figura 45 – Comparação das estratégias Greedy, Tabu, Grasp e HAQL - Modelo M2 - 500 iterações



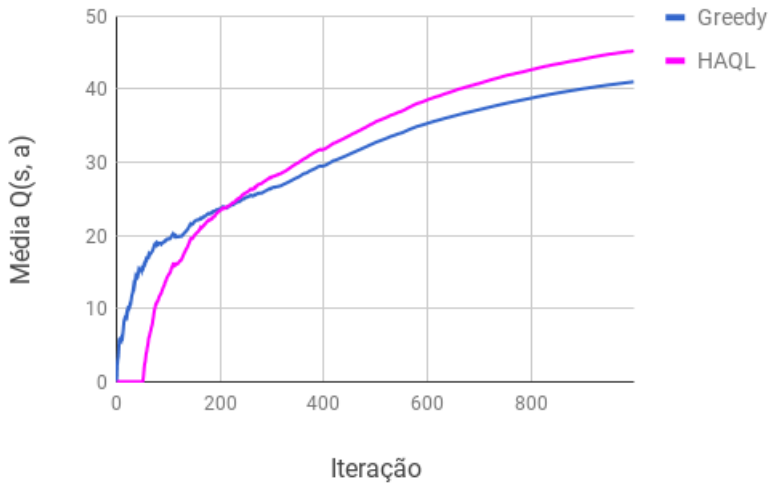
Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

A Figura 45 mostra a comparação de desempenho das heurísticas **Aleatória e HAQL**, aplicadas como estratégia de explotação, com as metaheurísticas **Tabu e Grasp** aplicadas como estratégia de exploração do algoritmo *Q-learning* em STI. O resultado foi calculado com base na média dos valores de utilidade Q(s, a). Para este teste foram realizadas **20 simulações**, com **3000 iterações** cada, utilizando o modelo de aprendiz **M2 – Bom**. Observa-se pelo experimento que a heurística HAQL obteve o terceiro melhor resultado a partir da **232ª iteração**.

Teste com 1000 iterações

A Figura 46 mostra a comparação de desempenho das heurísticas **Aleatória e HAQL**, aplicadas como estratégia de explotação do algoritmo *Q-Learning* em STI. O resultado foi calculado com base na média dos valores de utilidade Q(s, a). Para este teste foram realizadas **20 simulações**, com **1000 iterações** cada, utilizando o modelo de aprendiz **M2 – Bom**. Observa-se pelo experimento que a heurística HAQL obteve os melhores resultados de Q(s, a), a partir da **211ª iteração**.

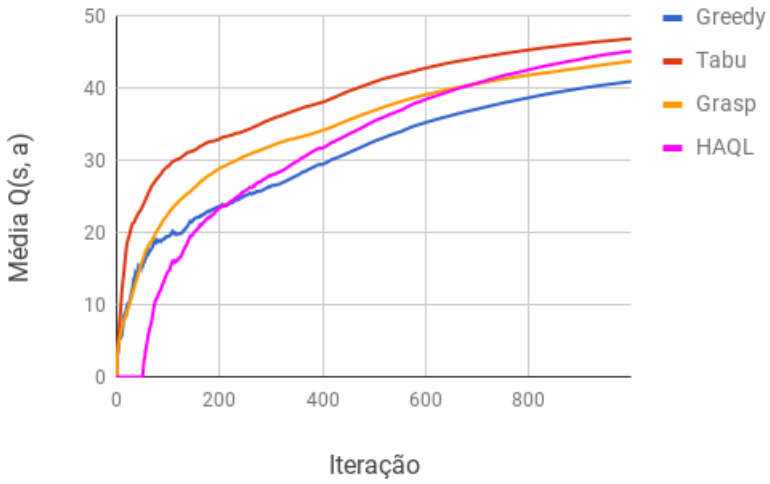
Figura 46 – Comparação das estratégias Greedy e HAQL - Modelo M2 - 1000 iterações



Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

A Figura 47 mostra a comparação de desempenho das heurísticas **Greedy e HAQL**, aplicadas como estratégia de explotação, com as metaheurísticas **Tabu e Grasp** aplicadas como estratégia de exploração do algoritmo *Q-Learning* em STI. O resultado foi calculado com base na média dos valores de utilidade Q(s, a). Para este teste foram realizadas **20 simulações**, com **1000 iterações** cada, realizadas com o modelo de aprendiz **M2 – Bom**. Observa-se pelo experimento que a heurística HAQL obteve o terceiro melhor resultado entre as **iterações 211 e 688** e o segundo melhor resultado entre as **iterações 689 e 1000**.

Figura 47 – Comparação das estratégias Greedy, Tabu, Grasp e HAQL - Modelo M2 - 1000 iterações

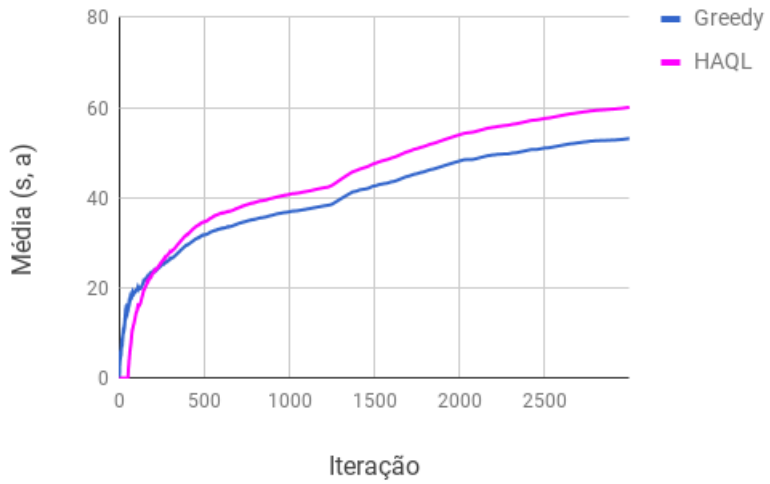


Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

Teste com 3000 iterações

A Figura 48 mostra a comparação de desempenho das heurísticas **Greedy e HAQL**, aplicadas como estratégia de explotação do algoritmo *Q-Learning* em STI. O resultado foi calculado com base na média dos valores de utilidade Q(s, a). Para este teste foram realizadas **20 simulações**, com **3000 iterações** cada, utilizando o modelo de aprendiz M2 – Bom. Observa-se pelo experimento que a heurística HAQL obteve os melhores resultados de Q(s, a) a partir da **211ª iteração**.

Figura 48 – Comparação das estratégias Greedy e HAQL - Modelo M2 - 3000 iterações

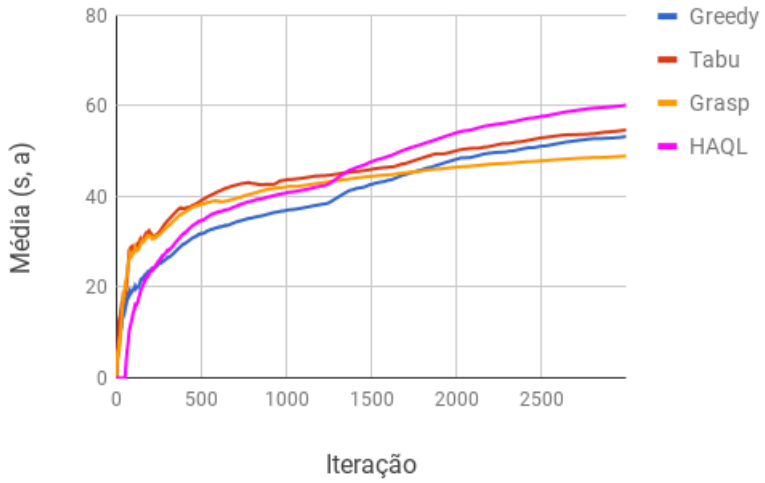


Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

A Figura 49 mostra a comparação de desempenho das heurísticas **Greedy e HAQL**,

aplicadas como estratégia de explotação, com as metaheurísticas **Tabu e Grasp** aplicadas como estratégia de exploração do algoritmo *Q-Learning* em STI. O resultado foi calculado com base na média dos valores de utilidade Q(s, a). Para este teste foram realizadas **20 simulações**, com **3000 iterações** cada, Utilizando o modelo de aprendiz **M2 – Bom**. Observa-se pelo experimento que a heurística HAQL obteve o melhor resultado a partir da **1347ª iteração**.

Figura 49 – Comparação das estratégias Greedy, Tabu, Grasp e HAQL - Modelo M2 - 3000 iterações



Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

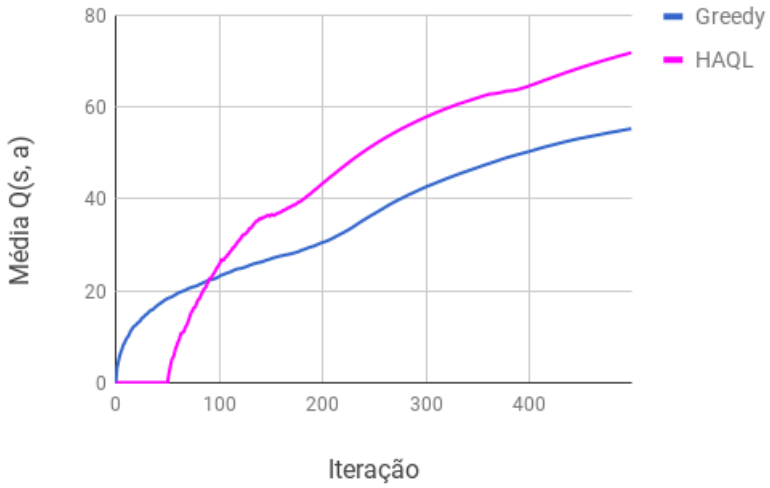
4.3 Modelo de Aprendiz M3 - Excelente

Teste com 500 iterações

A Figura 50 mostra a comparação de desempenho das heurísticas **Greedy e HAQL**, aplicadas como estratégia de explotação do algoritmo *Q-Learning* em STI. O resultado foi calculado com base na média dos valores de utilidade Q(s, a). Para este teste foram realizadas **20 simulações**, com **3000 iterações** cada, Utilizando o modelo de aprendiz **M3 – Excelente**. Observa-se pelo experimento que a heurística HAQL obteve os melhores resultados de Q(s, a) a partir da **90ª iteração**.

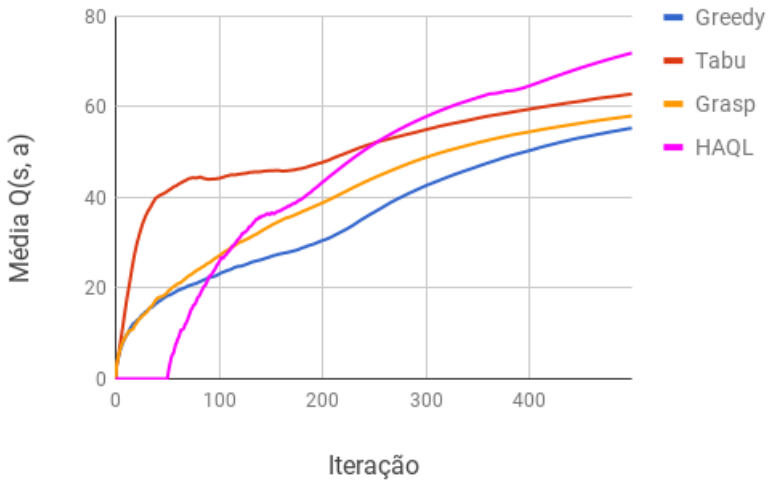
A Figura 51 mostra a comparação de desempenho das heurísticas **Greedy e HAQL**, aplicadas como estratégia de explotação, com as metaheurísticas **Tabu e Grasp** aplicadas como estratégia de exploração do algoritmo *Q-Learning* em STI. O resultado foi calculado com base na média dos valores de utilidade Q(s,a). Para este teste foram realizadas **20 simulações**, com *3000 iterações* cada, utilizando o modelo de aprendiz **M3 – Excelente**. Observa-se pelo experimento que a heurística HAQL obteve o melhor resultado a partir da **253ª iteração**.

Figura 50 – Comparação das estratégias Greedy e HAQL - Modelo M3 - 500 iterações



Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

Figura 51 – Comparação das estratégias Greedy, Tabu, Grasp e HAQL - Modelo M3 - 500 iterações

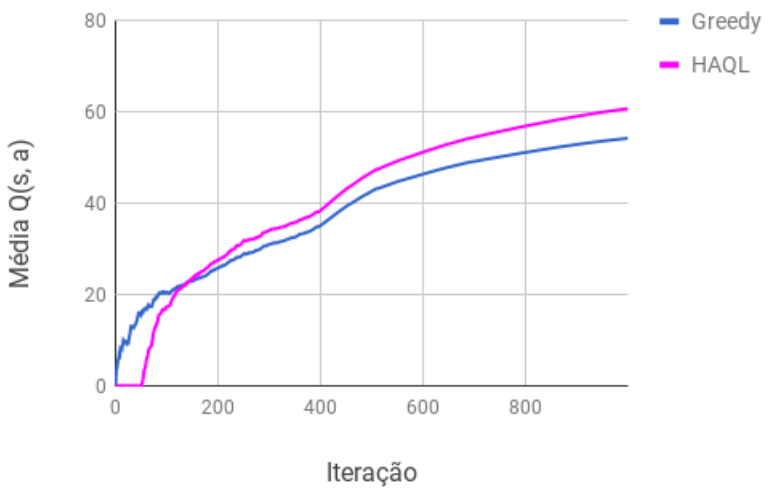


Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

Teste com 1000 iterações

A Figura 52 mostra a comparação de desempenho das heurísticas **Greedy e HAQL**, aplicadas como estratégia de explotação do algoritmo *Q-Learning* em STI. O resultado foi calculado com base na média dos valores de utilidade Q(s, a). Para este teste foram realizadas **20 simulações**, com **1000 iterações** cada, utilizando o modelo de aprendiz **M3 – Excelente**. Observa-se pelo experimento que a heurística HAQL obteve os melhores resultados de Q(s,a) a partir da **139ª iteração**.

Figura 52 – Comparação das estratégias Greedy e HAQL - Modelo M3 - 1000 iterações



Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

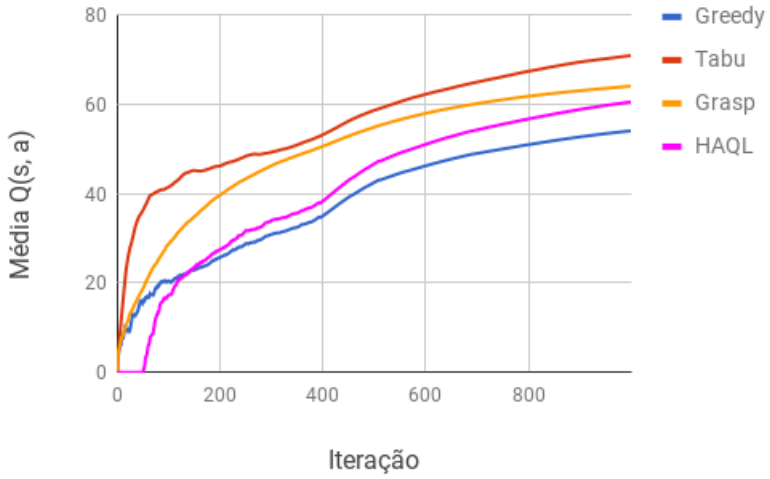
A Figura 53 mostra a comparação de desempenho das heurísticas **Greedy e HAQL**, aplicadas como estratégia de explotação, com as metaheurísticas **Tabu e Grasp** do algoritmo *Q-Learning* em STI. O resultado foi calculado com base na média dos valores de utilidade Q(s, a). Para esse teste foram realizadas **20 simulações**, com **1000 iterações** cada, utilizando o modelo de aprendiz **M3 – Excelente**. Observa-se pelo experimento que a heurística HAQL obteve o terceiro melhor resultado a partir da **139ª iteração**.

Teste com 3000 iterações

A Figura 54 mostra a comparação de desempenho das heurísticas **Greedy e HAQL**, aplicadas como estratégia de explotação do algoritmo *Q-Learning* em STI. O resultado foi calculado com base na média dos valores de utilidade Q(s, a). Para esse teste foram realizadas **20 simulações**, com **3000 iterações** cada, utilizando o modelo de aprendiz **M3 – Excelente**. Observa-se pelo experimento que a heurística HAQL obteve os melhores resultados de Q(s, a), a partir da **139ª iteração**.

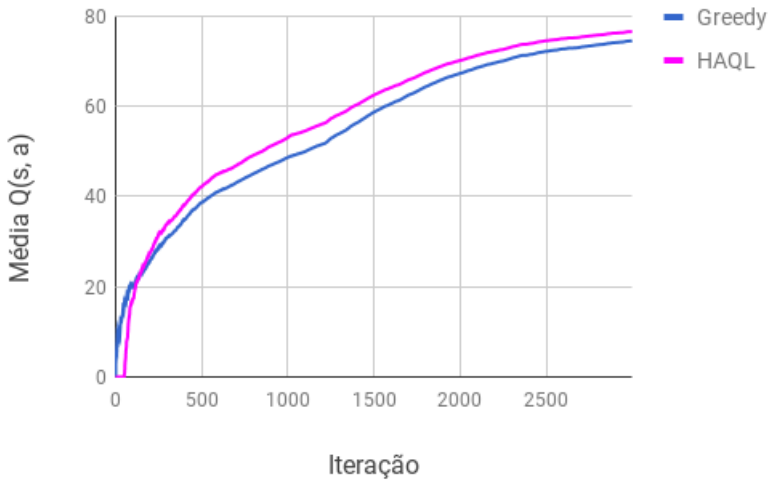
A Figura 55 mostra a comparação de desempenho das heurísticas **Greedy e HAQL**, aplicadas como estratégia de explotação, com as metaheurísticas **Tabu e Grasp** aplicadas com

Figura 53 – Comparação das estratégias Greedy, Tabu, Grasp e HAQL - Modelo M3 - 1000 iterações



Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

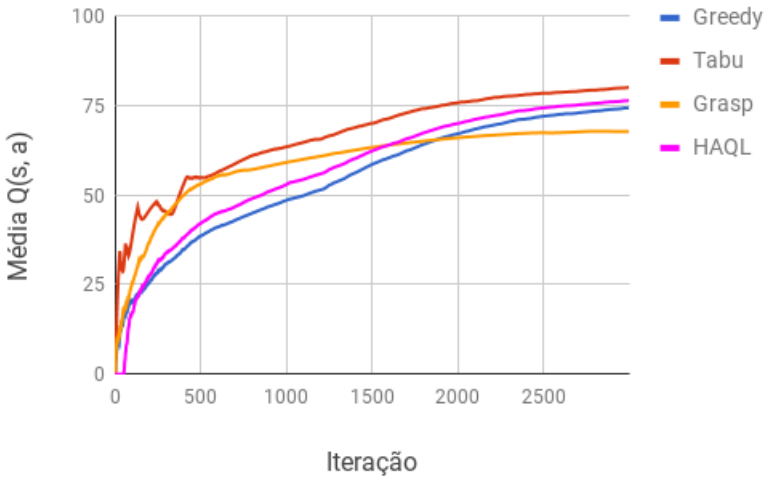
Figura 54 – Comparação das estratégias Greedy e HAQL - Modelo M3 - 3000 iterações



Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

estratégia de exploração do algoritmo *Q-Learning* em STI. O resultado foi calculado com base na média dos valores de utilidade Q(s, a). Para este teste foram realizadas **20 simulações**, de **3000 iterações** cada, realizadas com o modelo de aprendiz **M3 – Excelente**. Observa-se pelo experimento que a heurística HAQL obteve o terceiro melhor resultado a até a **iteração 1592** e o segundo melhor resultado entre as **iterações 1593 e 3000**.

Figura 55 – Comparação das estratégias Greedy, Tabu, Grasp e HAQL - Modelo M3 - 3000 iterações

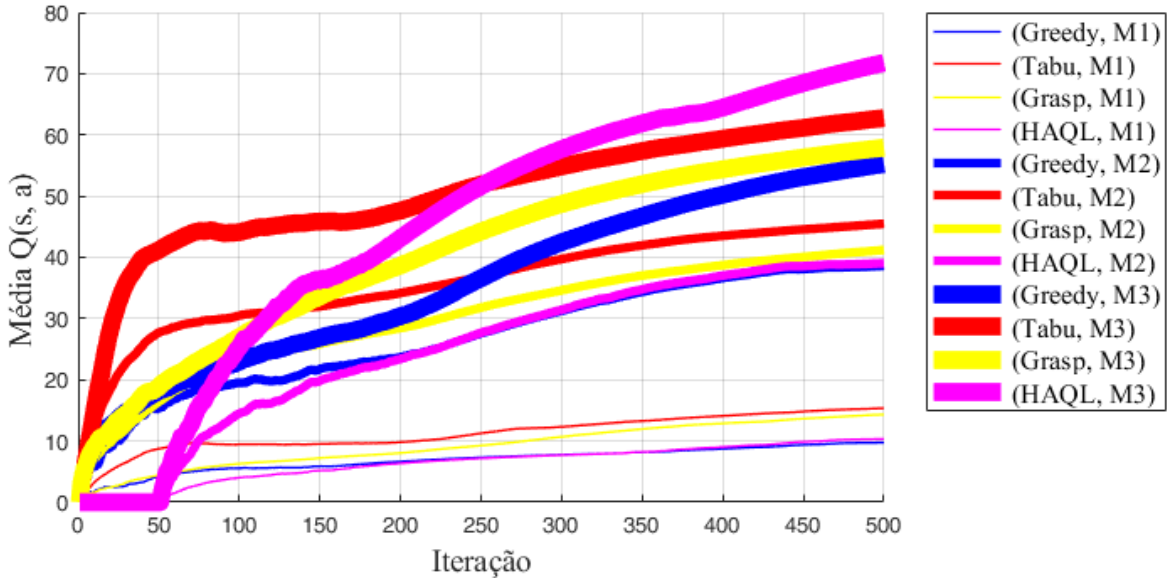


Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

5 ANÁLISE DE RESULTADOS

Para fins de aferição do desempenho das estratégias implementadas realizou-se o cruzamentos das médias dos valores Q(s, a) para os métodos Greedy, Tabu, GRASP e HAQL. As Figuras 56, 57, 58 se referem as simulações de 500, 1000 e 3000 iterações, respectivamente.

Figura 56 – Comparação das estratégias Greedy, Tabu, GRASP e HAQL – Modelos M1, M2 e M3 – 500 iterações



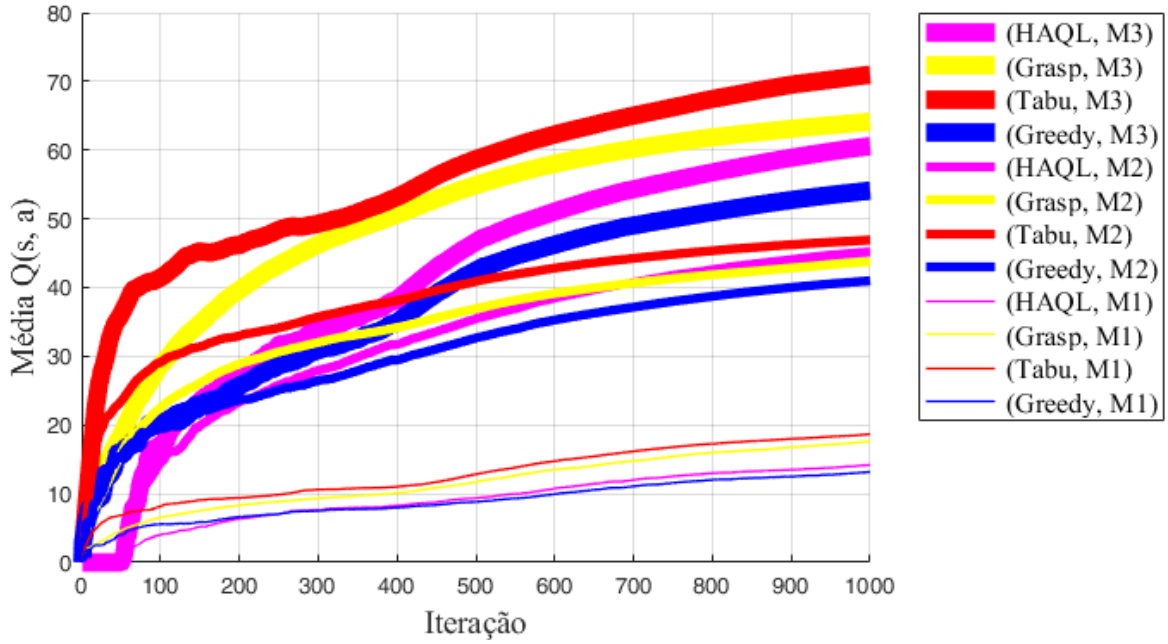
Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

Pela análise gráfica realizada pode-se perceber que a seleção de ações de treino em fase de execução, implementada pelo HAQL, acelera a convergência do algoritmo e oferece ganhos em termos de tempo de execução para todos os modelos M1, M2 e M3, se comparado com a estratégia de explotação Greedy do *Q-learning*, adotata por Guelpeli (2003).

Por outro lado, ao contrapor o desempenho da estratégia de exploração do HAQL com as metaheurísticas Tabu e Grasp implementados por Paiva (2016), percebe-se que estas demonstraram melhor performance de convergência para os M1 (500, 1000 e 3000 iterações), M2 (500 e 1000 iterações) e M3 (1000 e 3000 iterações).

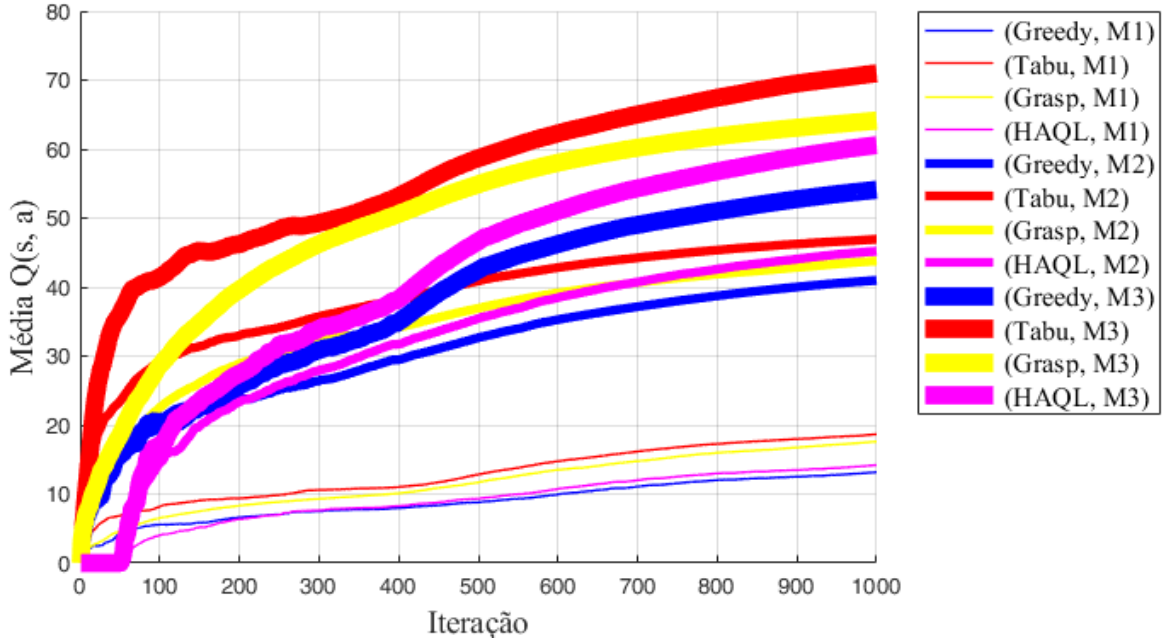
É interessante observar que para os modelos M2 (3000 iterações) e M3 (500 iterações) a estratégia de explotação com o algoritmo HAQL demonstrou maior eficácia que a metaheurística Tabu. E para os modelos M2 (1000 iterações) e M3 (3000 iterações) atingiu o segundo lugar em termos de desempenho, superando os resultados obtidos com a metaheurística Grasp. Esta constatação revela sua aplicabilidade em ambientes que requerem um grande número de iterações entre tutor e aprendiz.

Figura 57 – Comparação das estratégias Greedy, Tabu, GRASP, HAQL – Modelos M1, M2 e M3 – 1000 iterações



Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

Figura 58 – Comparação das estratégias Greedy, Tabu, GRASP e HAQL – Modelos M1, M2 e M3 – 3000 iterações



Fonte – Dados extraídos do simulador de STI. Gráfico elaborado pela autora.

6 CONSIDERAÇÕES FINAIS

Este trabalho propôs a implementação do algoritmo HAQL na transição de estados do *Q-learning*, aplicado ao domínio de Sistemas Tutores Inteligentes com Modelagem Autônoma do Aprendiz.

As simulações aqui apresentadas apontam graficamente a eficiência da técnica de inclusão de métodos heurísticos nos algoritmos de Aprendizado por Reforço.

Os resultados expostos denotam aceleração da convergência do algoritmo HAQL para todos os modelos, em comparação com o método de explotação Greedy do *Q-learning*, proposto como *baseline*. Dessa forma, considera-se comprovada a hipótese inicial pelo demonstrado nas Figuras 56, 57, 58.

Uma característica importante desta técnica é a utilização de informações extraídas em tempo de execução, que permite o aproveitamento de dados do problema que não estão explícitos ao pesquisador. A vantagem de utilização dessa técnica é que ela abstrai parte do problema, ou seja, reduz o espaço de busca do algoritmo e, consequentemente, o tempo de convergência.

Este ganho em termos de tempo de execução viabiliza sua utilização em Ambientes Virtuais de Aprendizagem e aproxima a interação virtual que ocorre nesses sistemas, daquela realizada em sala de aula.

6.1 Limitações

Nas simulações realizadas não foram analisadas as métricas Reforço Médio, Total de Visitas aos Estados e Transição de Estados.

A política P2 - mais restritiva também não foi considerada para fins de experimentação e testes.

6.2 Trabalhos Futuros

Como trabalhos futuros, sugere-se a realização de simulações com a política P2 - mais restritiva, e análise das métricas Reforço Médio, Total de Visitas aos Estados e Transição de Estados.

Aponta-se a necessidade de comparação e análise de resultados por métodos estatísticos.

Para o domínio de STI, indica-se a possibilidade de utilização de outras técnicas de aceleração do algoritmo *Q-learning* e aplicação de conceito de hiper-heurísticas apresentado por Sucupira (2004).

REFERÊNCIAS

BARROS, L. N. d.; SANTOS, E. T. Um estudo sobre a modelagem do domínio de geometria descritiva para a construção de um sistema tutor inteligente. In: **Anais do Simpósio Brasileiro de Informática na Educação**. [S.l.: s.n.], 2000. p. 259–268.

BELLONI, M. L. Ensaio sobre a educação a distância no brasil. **Educação & sociedade**, SciELO Brasil, v. 23, n. 78, p. 117–142, 2002.

BIANCHI, R. A. d. C. **Uso de heurísticas para a aceleração do aprendizado por reforço.** Tese (Doutorado) — Universidade de São Paulo, 2004.

DORÇA, F. A. *et al.* **Uma abordagem estocástica baseada em aprendizagem por reforço para modelagem automática e dinâmica de estilos de aprendizagem de estudantes em sistemas adaptativos e inteligentes para educação a distância**. Tese (Doutorado) — Universidade Federal de Uberlândia, 2012.

GAVIDIA, J. J. Z.; ANDRADE, L. C. V. d. Sistemas tutores inteligentes. **Trabalho de Conclusão da Disciplina de IA, Universidade Federal do Rio de Janeiro. Rio de Janeiro–RJ: UFRJ**, 2003.

GIGERENZER, G.; GAISSMAIER, W. Heuristic decision making. **Annual review of psychology**, Annual Reviews, v. 62, p. 451–482, 2011.

GIRAFFA, L. M. M. Uma arquitetura de tutor utilizando estados mentais. 1999.

GIRAFFA, L. M. M.; VICCARI, R. M. Estratégias de ensino em sistemas tutores inteligentes modelados através da tecnologia de agentes. **Revista Brasileira de Informática na Educação. Florianópolis**, 1999.

GREER, J. E.; MCCALLA, G. I. **Student modelling: the key to individualized knowledge-based instruction**. [S.l.]: Springer Science & Business Media, 2013. v. 125.

GUELPELI, M. V. C. **Utilização de aprendizado por reforço para modelagem autônoma do aprendiz em sistemas tutores inteligentes**. Tese (Doutorado) — Instituto Tecnológico de Aeronáutica, 2003.

GUELPELI, M. V. C.; OMAR, N.; RIBEIRO, C. H. C. Aprendizado por reforço para um sistema tutor inteligente sem modelo explícito do aprendiz. **Brazilian Journal of Computers in Education**, v. 12, n. 2, p. 69–77, 2004.

JESUS, A. de. Sistemas tutores inteligentes uma visao geral. **Revista Eletrônica de Sistemas de Informação ISSN 1677-3071 doi: 10.21529/RESI**, v. 2, n. 2, 2009.

JR, L. A. C.; BIANCHI, R. A.; MATSUURA, J. P. Aprendizado por reforço acelerado por heurísticas no domínio do futebol de robôs simulado. **Anais do VIII Simpósio Brasileiro de Automação Inteligente**, Natal: Sociedade Brasileira de Automática, 2007.

LOPEZ, A. G. T. **Controle Preditivo com Aprendizado por Reforço para Produção de Óleo em Poços Inteligentes**. Tese (Doutorado) — PUC-Rio, 2010.

MARTINS, M. F.; BIANCHI, R. A. Comparação de desempenho de algoritmos de aprendizado por reforço no domínio do futebol de robôs. **Anais do VIII Simpósio Brasileiro de Automação Inteligente**, Natal: Sociedade Brasileira de Automática, 2007.

MCTAGGART, J. Intelligent tutoring systems and education for the future. **512X Literature Review April**, v. 30, n. 2, 2001.

MITCHELL, T. M. **Machine Learning**. [S.l.]: McGraw-Hill Science/Engineering/Math, 1997. ISBN 0070428077.

MORAIS, F.; SILVA, J. da; REIS, H.; ISOTANI, S.; JAQUES, P. Computação afetiva aplicada à educação: uma revisão sistemática das pesquisas publicadas no brasil. In: **Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)**. [S.l.: s.n.], 2017. v. 28, n. 1, p. 163.

PAIVA, É. d. O. **Melhoria na convergência do algoritmo Q-Learning na aplicação de sistemas tutores inteligentes**. Dissertação (Mestrado) — UFVJM, 2016.

RUSSEL, S.; NORVIG, P. **Inteligência artificial**. 3ª. ed. Rio de Janeiro: Elsevier, 2013. ISBN 978-85-352-3701-6.

SUCUPIRA, I. R. Métodos heurísticos genéricos: metaheurísticas e hiper-heurísticas. **USP: São Paulo**, 2004.

SUTTON, R. S.; BARTO, A. G. **Reinforcement learning: An introduction**. London: MIT press Cambridge, 2012.

TORRES, I. T. M. Sistema para la selección del método pedagógico en sistemas tutores inteligentes/system for selection of pedagogical method in intelligent tutorial systems. **International Journal of Innovation and Applied Studies**, International Journal of Innovation and Applied Studies, v. 20, n. 2, p. 643, 2017.

VICCARI, R. M.; GIRAFFA, L. M. Sistemas tutores inteligentes: abordagem tradicional x abordagem de agentes. **XIII Simpósio Brasileiro de Inteligência Artificial, Curitiba**, 1996.

WATKINS, C. J. C. H. **Learning from delayed rewards**. Tese (Doutorado) — King's College, Cambridge, 1989.

ZHANG, X.; LIU, Z. An optimized q-learning algorithm based on the thinking of tabu search. In: IEEE. **Computational Intelligence and Design, 2008. ISCID'08. International Symposium on**. [S.l.], 2008. v. 1, p. 533–536.

APÊNDICE A - VALOR DE UTILIDADE DA AÇÃO (Q(S,A))

O Apêndice A mostra os resultados de desempenho do valor de utilidade Q(s,a) de um par (estado(s), ação(a)) das heurísticas Greedy, Tabu, GRASP e HAQL aplicadas ao Modelo de Aprendiz M1 – Ruim, M2 – Bom e M3 – Excelente, política pedagógica P1. Para cada um desses modelos de aprendiz foram realizadas simulações de 500, 1000 e 3000 iterações. Os resultados apresentados foram obtidos através da média de 20 execuções de cada simulação.

Disponível eletronicamente em:

- M1 - 500 iterações
- M1 - 1000 iterações
- M1 - 3000 iterações
- M2 - 500 iterações
- M2 - 1000 iterações
- M2 - 3000 iterações
- M3 - 500 iterações
- M3 - 1000 iterações
- M3 - 3000 iterações

APÊNDICE B - TESTES PARA ADAPTAÇÃO DO PARÂMETRO η

O Apêndice B mostra os resultados de desempenho do valor de utilidade Q(s, a) de um par (estado(s), ação(a)) da heurística HAQL, com variação do parâmetro η, aplicada ao Modelo de Aprendiz M2 – Bom, política pedagógica P2. Para esse teste foram realizadas simulações de 1500 iterações. Os resultados apresentados foram obtidos através da média de 20 execuções de cada simulação.

Disponível eletronicamente em: Variação do Parâmetro η

APÊNDICE C - TESTES PARA ADAPTAÇÃO DO PARÂMETRO ξ

O Apêndice C mostra os resultados de desempenho do valor de utilidade Q(s,a) de um par (estado(s), ação(a)) da heurística HAQL, com variação do parâmetro ξ, aplicada ao Modelo de Aprendiz M2 – Bom, política pedagógica P2. Para esse teste foram realizadas simulações de 1500 iterações. Os resultados apresentados foram obtidos através da média de 20 execuções de cada simulação.

Disponível eletronicamente em: Variação do Parâmetro ξ

APÊNDICE D - TESTES PARA EXTRAÇÃO DA ESTRUTURA HEURÍSTICA

O Apêndice D mostra os resultados de desempenho do valor de utilidade Q(s,a) de um par (estado(s), ação(a)) da heurística HAQL, com variação do período de extração da estrutura heurística, aplicada ao Modelo de Aprendiz M2 – Bom, política pedagógica P2. Para esse teste foram realizadas simulações de 1500 iterações. Os resultados apresentados foram obtidos através da média de 20 execuções de cada simulação.

Disponível eletronicamente em: Teste de extração da estrutura heurística