Uma abordagem para planejamento instrucional apoiada em processos de decisão Markovianos parcialmente observáveis

Everson Freitas Giacomelli Junior

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2018

Everson Freitas Giacomelli Junior

Uma abordagem para planejamento instrucional apoiada em processos de decisão Markovianos parcialmente observáveis

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Carlos Roberto Lopes

Uberlândia
2018

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

G429a
2019

Giacomelli Junior, Everson Freitas, 1985
Uma abordagem para planejamento instrucional apoiada em processos de decisão Markovianos parcialmente observáveis [recurso eletrônico] / Everson Freitas Giacomelli Junior. - 2019.

Orientador: Carlos Roberto Lopes.
Dissertação (mestrado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://dx.doi.org/10.14393/ufu.di.2019.1285
Inclui bibliografia.
Inclui ilustrações.

1. Computação. 2. Markov, Processos de. 3. Aprendizado do computador. 4. Planejamento. I. Lopes, Carlos Roberto, 1962, (Orient.) II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU: 681.3

Angela Aparecida Vicentini Tzi Tziboy – CRB-6/947

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Computação a aceitação da dissertação intitulada "**Uma Abordagem para Planejamento Instrucional apoiado em Processos de Decisão Markovianos Parcialmente Observáveis**" por **Everson Freitas Giacomelli Junior** como parte dos requisitos exigidos para a obtenção do título de **Mestre em Ciência da Computação.**

Uberlândia, ___ de _____ de _____

Orientador: _____
Prof. Dr. Carlos Roberto Lopes
Universidade Federal de Uberlândia

Banca Examinadora:

_____
Prof. Dr. Sérgio Antônio Andrade de Freitas
Universidade de Brasília

_____
Prof. Dr. Fabiano Azevedo Dorça
Universidade Federal de Uberlândia

Agradecimentos

Agradeço aos meus familiares por todo o apoio durante o período do programa. Em especial, minha namorada Laine Moraes Souza, meus pais Everson Giacomelli e Maria Aparecida, as minhas irmãs Natasha, Talita, Luciana e Daniela e a minha tia Elaine Giacomelli. Todos foram muito importantes para que eu pudesse chegar até aqui.

Agradeço também o meu orientador Carlos Roberto Lopes, por ter me aceitado como seu orientando e pelas boas e proveitosas discussões nas reuniões de sexta-feira, que contribuíram muito para o meu crescimento como pesquisador.

Estendo os meus agradecimentos ao corpo docente do Programa de Pós-graduação em Computação da Universidade Federal de Uberlândia, de onde pude obter a formação necessária e a aquisição dos conhecimentos para a escrita deste documento.

"Sua vida pode ser dividida em dois períodos: antes de agora e a partir de agora."
(Prof. Obvious Stating)

Resumo

Uma das grandes dificuldades encontradas em uma sessão de tutoria refere-se a escolha de ações pedagógicas apropriadas, que considera os conhecimentos individuais prévios do estudante, e tem por finalidade conduzi-lo a atingir a proficiência na assimilação de um conjunto de conceitos. Estas ações devem considerar o perfil do estudante e adaptar o ensino segundo o seu estilo de aprendizagem e as suas preferências, a fim de manter o estudante engajado.

Considerando que o ensino personalizado exerce influência positiva na assimilação do conteúdo disciplinar, a tendência é que os novos ambientes virtuais de aprendizagem sejam implementados de forma a adaptar-se ao perfil do estudante. A elaboração de um sistema adaptativo deve considerar a inclusão de um módulo responsável pela geração de um plano de ações instrucionais, com capacidade de modificar o seu próprio planejamento inicial para diagnosticar falhas na compreensão de fundamentos essenciais ao conteúdo, à medida em que estas sejam percebidas ao longo da sessão de tutoria. Portanto, é de responsabilidade do sistema adaptativo conduzir um plano de ensino considerando todas as dificuldades que forem observadas para cada aluno em particular. Para atingir tal objetivo este sistema deverá ter a capacidade de inferir o conhecimento do estudante, propor ações e observar a evolução do aprendizado, supervisionando integralmente o processo de aprendizagem.

Perante as dificuldades expostas, neste trabalho apresenta-se a proposta e a implementação de um módulo instrucional para um sistema de apoio ao ensino baseado em Processos de Decisão de Markov parcialmente observáveis. A escolha do modelo justifica-se por sua capacidade de tomada de decisões diante das incertezas que caracterizam o ambiente: incerteza quanto ao conhecimento que o aprendiz apresenta frente aos conceitos a serem assimilados e os efeitos não determinísticos das ações propostas. O problema do sequenciamento de ações produzidos por este módulo pode ser representado como um problema de planejamento probabilístico e é portanto resolvido por algoritmos de planejamento.

Existem diversos algoritmos de planejamento probabilísticos que poderiam ser utiliza-

dos no desenvolvimento de um módulo instrucional para um sistema adaptativo de ensino, sendo que os resultados obtidos por cada um deles podem divergir. Um planejador ou sistema de planejamento é um *software* que implementa um algoritmo de planejamento. Neste trabalho aplica-se uma metodologia baseada em aprendizagem de máquina para a seleção de um planejador probabilístico que proporcione a escolha das melhores ações instrucionais. A escolha do planejador é realizada considerando as características de um domínio de especificação que ensina conceitos e os avalia conforme a proficiência do estudante.

Ao término do processo de seleção, validou-se o resultado executando-se o planejador apontado pela metodologia ao domínio que descreve o módulo instrucional. O desempenho obtido pela sua execução foi comparada com a de outros planejadores candidatos, comprovando dessa forma a eficiência na aplicação do método proposto.

Palavras-chave: Planejamento Probabilístico, Sistemas Tutores Inteligentes, Aprendizagem de Máquina, Processos de Decisão Markovianos Parcialmente Observáveis.

Abstract

One of the major difficulties encountered in a tutoring session is the choice of appropriate pedagogical actions that take into account the student's previous individual knowledge and aims to lead the student to achieve a proficiency in the assimilation of a set of concepts. These actions should consider the student's profile and adapt the teaching according to his or her learning style and preferences in order to keep the student engaged.

Considering that personalized education has a positive influence on the assimilation of the disciplinary content, the tendency is that the new virtual learning environments are to be implemented in a way that adapts to the student's profile. The elaboration of an adaptive system should consider the inclusion of a module responsible for generating an instructional action plan, capable of modifying its own initial planning to diagnose failures in the comprehension of essential fundamentals of content as they are perceived throughout the tutoring session. Therefore, it is the responsibility of the adaptive system to conduct a teaching plan considering all the difficulties that are observed for each student in particular. To achieve this goal, this system should have the capacity to infer the student's knowledge, propose actions and observe the evolution of learning, fully supervising the learning process.

Given the difficulties presented, this paper presents the proposal and the implementation of an instructional module for a support system for education based on Partially Observable Markov Decision Processes. The choice of the model is justified by its capacity to make decisions in the face of the uncertainties that characterize the environment: uncertainty regarding the knowledge the learner presents about the concepts to be assimilated and the non-deterministic effects of the proposed actions. The problem of the sequencing of actions produced by this module can be represented as a problem of probabilistic planning and are solved by planning algorithms.

There are several probabilistic planning algorithms that could be used in the development of an instructional module for an adaptive teaching system, and the results obtained by each of them may differ. A planner or planning system is software that implements

a planning algorithm. In this work, a methodology based on machine learning is applied to select a probabilistic planner that provides the choice of the best instructional actions. The choice of the planner is accomplished by considering the characteristics of a specification domain that teaches concepts and evaluates them according to the student's proficiency.

At the end of the selection process, the result was validated by executing the planner indicated by the methodology to the domain that describes the instructional module. The performance obtained by its execution was compared with that of other candidate planners, thus proving the efficiency in the application of the proposed method.

Keywords: Probabilistic Planning, Intelligent Tutoring Systems, Machine Learning, Partially Observable Markov decision process.

Lista de ilustrações

Lista de tabelas

Lista de siglas

ADL Action Description Language

AIT Autonomous Intelligent Tutor

IPC International Planning Competition

MDP Markov Decision Process

PDDL Planning Domain Definition Language

POMDP Processos de Decisão Markovianos Parcialmente Observáveis

RDDL Relational Dynamic Influence Diagram Language

STI Sistemas Tutores Inteligentes

Sumário

Capítulo 1

Introdução

1.1 Motivação

Até pouco tempo, as relações de ensino se restringiam a figura de um professor lecionando um conjunto de conteúdos disciplinares, previamente definidos, a um grupo de estudantes limitados em um espaço físico. As discussões se restringiam a didática e a metodologia de ensino e os estudantes eram tratados de maneira uniforme, sem considerar quaisquer aspectos individuais relativos ao seu estilo de aprendizagem, comportamento ou domínio no assunto. Falhas na compreensão dos conceitos muitas vezes eram percebidas de modo tardio. Os recentes avanços da tecnologia e, sobretudo, da inteligência artificial, que por meio de agentes inteligentes conseguem a cada dia simular de forma mais realística o comportamento de um tutor humano, propiciaram uma grande mudança de paradigma, desenvolvendo plataformas que possibilitam explorar novas estratégias de ensino centradas no estudante.

Pesquisadores em educação têm buscado apoio na tecnologia como uma fonte para aprimoramento dos seus objetivos, mirando um ambiente de aprendizagem interativo que se adeque as necessidades cognitivas pessoais de cada estudante. Dentre estas tecnologias, os Sistemas Tutores Inteligentes (STI) mostram-se bastante proeminentes, o que pode ser justificado em virtude da sua interatividade e também da capacidade de supervisionar as ações do aluno, tomando para si a responsabilidade de conduzir toda a sessão de ensino, propondo ações que visam alcançar os objetivos definidos pela disciplina. A efetividade da aplicação de sistemas tutores foi comprovada em diversos domínios (Matsuda e VanLehn,2000; Rafferty e al., 2011; Seffrin, 2015).

A aplicação de Sistemas Tutores Inteligentes torna-se ainda mais eficiente quando se considera uma tutoria adaptativa, interativa um-para-um. Estudos que comparam diferentes formas de ensino, demostram que este tipo de tutoria pode conseguir melhores resultados que um ensino linear em sala de aula (Wang 2014). Em seu experimento, **??**), observou que aprendizes que receberam ensino individualizado tiveram em média, um desempenho na ordem de dois desvios-padrões superior comparados a outros aprendizes

que receberam instruções de maneira convencional. Por meio da tecnologia dos agentes, os Sistemas Tutores Inteligentes, viabilizam a construção de ambientes de aprendizagem dinâmicos e adaptativos, tornando-os mais propícios ao desenvolvimento do educando, principalmente por privilegiarem um modelo de ensino centrado no aluno, que por sua vez passa a ter uma participação mais ativa no processo de aprendizagem (SILVA 2006).

Um aspecto fundamental que deve ser considerado na construção de um Sistema de Apoio Educacional (como um STI por exemplo), refere-se a escolha do algoritmo responsável pela geração das ações instrucionais. Para isto diversos sistemas fazem uso de algoritmos de planejamento automatizado, que é uma subárea da Inteligência Artificial que busca antecipar um conjunto de ações que quando aplicadas a um estado corrente possibilitem que um dado objetivo seja alcançado. Existem diversas abordagens em planejamento considerando os diversos aspectos do ambiente (Russell e Norvig 2016). Neste trabalho é dado ênfase ao planejamento probabilístico que faz uso de Processos de Decisão Markovianos Parcialmente Observáveis(POMDP, acrônimo do equivalente em inglês *Partially Observable Markov Decision Process*), um modelo iterativo e sequencial que é estruturado como um conjunto de estados que são percebidos através de observações do ambiente e ações que provocam transições não-determinísticas entre estes estados. Um agente baseado em um modelo POMDP elabora um plano de ações visando alcançar estados mais promissores. Trata-se, portanto, de um problema de tomada de decisão em intervalos de tempo e aplicados a ambientes em que os resultados das ações não podem ser definidos de forma determinística. No contexto educacional, o modelo se aproxima a de uma sessão de tutoria, quando são consideradas as incertezas sobre o estado de conhecimento real do estudante, a sua percepção e o efeito não-determinístico produzido como resultado das ações instrucionais.

Um sistema educacional adaptativo tem como responsabilidade implementar estratégias para supervisionar as ações do aluno, e ao mesmo tempo decidir dentre um conjunto de instruções pedagógicas, aquelas que resultem em um melhor aproveitamento durante o curso da disciplina. Sabe-se que a escolha apropriada das ações em uma sessão de tutoria reflete no engajamento do estudante, podendo promover emoções que são mais ou são menos receptivas a novos conhecimentos (Zhang 2013). O objetivo geral deste trabalho é determinar um procedimento para a escolha de um planejador probabilístico para compor o Módulo Instrucional de um sistema educacional adaptativo. Um planejador probabilístico é um componente de *software* autônomo que implementa um algoritmo de planejamento e é dotado da capacidade de antecipar ações futuras, que no caso do Módulo Instrucional significa determinar um conjunto de ações pedagógicas sequenciais a serem apresentadas a um aluno. Embora existam muitos planejadores que realizam esta tarefa, um consenso na área de estudos sobre Planejamento é que não existe um planejador que seja superior a todos os demais em todos os domínios de problemas (9). Neste sentido, é necessário que a escolha de um planejador seja realizada segundo algum critério objetivo,

tal como as características particulares que compõe a especificação do problema para um determinado domínio. Este trabalho tem como um de seus objetivos contribuir para a escolha de tal planejador e aborda uma solução com base em um portfólio de planejadores probabilísticos autônomos e no uso de algoritmos de Aprendizagem de Máquina para auxiliar neste processo.

Para validar o mérito da seleção do planejador adequado foi desenvolvido um modelo educacional em um formalismo de planejamento probabilístico. A especificação deste modelo implementa a técnica *Knowledge tracing* para manter a rastreabilidade da aquisição sobre o conhecimento do estudante e dessa forma prover ações instrucionais personalizadas. Este modelo educacional considera o nível de proficiência do estudante em um conjunto de conceitos e em seus pré-requisitos para escolher as ações pedagógicas a serem aplicadas. A proposta foi implementada em um protótipo de um sistema de apoio ao ensino, obtendo resultados satisfatórios.

1.2 Objetivos e Desafios da Pesquisa

O objetivo principal do trabalho descrito nessa dissertação foi propor e implementar um módulo de planejamento instrucional apoiado em Inteligência Artificial com base em Processos de Decisão Markovianos Parcialmente Observáveis a ser empregado em um sistema de aprendizagem. Dessa forma o trabalho desenvolvido abordar os principais aspectos relacionados ao módulo de planejamento das ações de um sistema educacional online inteligente, que leva em consideração as necessidades do acompanhamento cognitivo de cada aprendiz para promover uma educação personalizada e com qualidade. Este sistema tem a capacidade de inferir o conhecimento que o estudante possui e realizar um planejamento para que todas as habilidades desejadas sejam ensinadas ao estudante, considerando as dependências entre os conceitos a serem lecionados. A adoção de POMDPs justifica-se por sua capacidade de deliberar em ambientes onde as informações são incompletas ou imprecisas. Em um processo de decisão Markoviano a informação necessária para a tomada de decisão depende única e exclusivamente do estado corrente, não importando o histórico dos estados e ações anteriores. A implementação de um sistema de aprendizagem com esta abordagem, implica em uma série de desafios enumerados abaixo:

- É necessário representar o conhecimento do estudante. Esta representação não é trivial, visto que o conhecimento é um conceito abstrato. O processo de abstraí-lo na forma de uma representação conceitual é uma das características chave do sistema, pois o estado de conhecimento do estudante é o conceito discriminatório que norteia as decisões que são tomadas na condução do ensino.

- Devido a dificuldade de se determinar precisamente o grau de compreensão que um estudante possui sobre cada um dos conceitos que compõe uma disciplina de

estudo, o sistema deverá trabalhar com inferências. As inferências sobre os estados são conhecidas no modelo POMDP como estado de crença. Os estados de crença são observados pelo tutor após a aplicação de uma ação, que resulta em um novo estado. Na prática, o resultado de uma ação provoca o ganho de novos conhecimentos, que poderão ser observados pelo tutor e usados para atualizar a estrutura de dados adotada para representar o conhecimento do estudante. Portanto, as observações também representam um elemento-chave para o modelo.

- Na Arquitetura de Sistemas Tutores Inteligente, um dos módulos contemplados é o Módulo Instrucional, responsável pela escolha das ações pedagógicas apropriadas. Este módulo normalmente engloba planejadores autônomos que já foram avaliados em competições. A escolha de um desses planejadores é fundamental para que os objetivos do ensino sejam alcançados com êxito, evitando escolhas de ações instrucionais apropriadas em razão de desconhecimento dos pré-requisitos necessários para a compreensão de cada um dos conceitos. Como a eficácia dos planejadores depende de uma série de características do domínio do problema, é desejável que se tenha uma metodologia que possa auxiliar na escolha deste planejador.

- O trabalho busca ainda fazer a integração entre a parte teórica dos conceitos de tomadas de decisão complexas em uma abordagem prática, avaliando e pontuando de forma discreta os resultados obtidos, e realizando comparações com os processos de ensino tradicional.

1.3 Contribuições

As principais contribuições deste trabalho são:

- A especificação em uma linguagem de formalismo que descreve um domínio educacional, onde as ações pedagógicas são sugeridas considerando os elementos cognitivos do estudante, como a sua proficiência nos conceitos ensinados. Estes elementos não são de fácil observação, e devido as incertezas presentes sobre o estado real de conhecimento do estudante, o modelo POMDP se apresenta como uma escolha promissora. Este domínio pode ser utilizado para a construção de um módulo tutor.

- Uma metodologia para a seleção de um planejador probabilístico que poderá ser aproveitada no planejamento instrucional no desenvolvimento de um sistema educacional, tal como um STI. A importância desta metodologia decorre do fato de que não existe um algoritmo de planejamento que seja superior a todos os demais. Portanto, para que o planejamento seja efetivo, é necessário considerar as características do domínio de planejamento.

- ❑ Um protótipo de um sistema de ensino adaptativo que seleciona as suas ações instrucionais em conformidade com o desempenho do estudante avaliado durante a sessão de tutoria.

1.4 Organização da Dissertação

A dissertação está estruturada da seguinte forma: no Capítulo 2 são identificados trabalhos relacionados à proposta desta pesquisa. O Capítulo 3 propicia os fundamentos teóricos necessários para a compreensão da proposta apresentada. O Capítulo 4 detalha os elementos da especificação formal para um domínio educacional que considera o estado cognitivo do estudante. Os aspectos da metodologia para seleção de um planejador autônomo são discutidos no Capítulo 5. O Capítulo 6 dispõe sobre os resultados alcançados pela metodologia e, por fim, no Capítulo 7 tem-se a conclusão do trabalho.

Capítulo 2

Revisão da Literatura

Este capítulo apresenta alguns trabalhos prévios que foram considerados na elaboração da proposta, e que foram classificados como a seguir: propostas relacionados a modelos POMDP aplicados a domínios educacionais são apresentados na Seção 2.1. Na Seção 2.2 são discutidos trabalhos com ênfase na modelagem e rastreabilidade do conhecimento. Por fim, a Seção 2.3 apresenta algumas importantes contribuições que se baseiam em características do problema para a seleção de planejadores probabilísticos.

2.1 Sistema Tutores e modelagem POMDP

A utilização de POMDPs na construção de sistemas tutores inteligentes é objeto de estudo de diversos trabalhos e tem logrado bons resultados, em virtude do modelo apresentar uma sólida abordagem teórica e, por tal razão, tem sido utilizada para modelar o conhecimento do estudante, permitindo a formulação de estratégias adequadas de ensino. **??**) demonstra com aprendizes humanos que o tempo de aprendizagem para a assimilação de conteúdos acadêmicos pode ser acelerado quando utilizado sistemas modelados com POMDP. Em seu experimento, eram designados mapeamentos entre letras e números, e os participantes foram divididos entre aqueles que recebiam instruções baseadas em uma política fornecidas por um planejamento POMDP e aqueles participantes que recebiam instruções aleatórias. O objetivo era descobrir a associação entre as letras e os números no menor tempo possível. Usando um teste de Kruskal-Wallis, os autores concluíram que o tempo médio de ensino eram reduzidos em aproximadamente 28% quando era realizado através de um planejamento POMDP. Um estudo inicial sobre recomendações de ações pedagógicas ótimas através de processos Markovianos também foi realizado em (Meza et al. 2016). A principal contribuição deste trabalho é a comparação dos resultados entre algoritmos que proporcionam soluções exatas e aproximadas.

Com o objetivo de ensinar habilidades de argumentação, **??**) desenvolveram um planejador instrucional baseado em processos de decisão Markoviano para ser aplicado em um STI, onde aos estudantes eram apresentados hipóteses e estes por sua vez, eram con-

vidados a desenvolver argumentos ou expor evidências que sustentassem o seu ponto de vista sobre a hipótese proferida pelo sistema. O estudo tinha como foco o planejamento instrucional, que tenta encontrar uma sequência de instruções de forma a maximizar os resultados esperados pelo ensino, aplicando as ações pedagógicas segundo uma ordem apropriada. A abordagem sugerida pelos autores considera o planejamento instrucional como um problema de decisão de planejamento teórico baseado em processos de decisão Markovianos. Algumas dificuldades são levadas em consideração na obtenção deste tipo de planejamento: a imprecisão no modelo do estudante, respostas inesperadas e compreensão equívocada de conceitos que foram explicado pelo STI. Todas estas dificuldades são representadas pelo modelo POMDP, comprovando dessa forma a sua aplicabilidade na construção de um planejador instrucional como um agente de decisão sequencial, com capacidade para lidar com as incertezas presentes em um ambiente de tutoria.

Zhang (Zhang 2013) aborda estratégias de ensino para um STI que ensina uma disciplina que contém conceitos, e estes por sua vez pode ter outros pré-requisitos que são necessários para a sua compreensão. Por estratégia de ensino compreende-se, dentre esta cadeia de conceitos e seus pré-requisitos, o ponto de partida pelo qual o tutor deverá iniciar a tutoria considerando o estado de conhecimento do estudante. O desafio de se alcançar a estratégia de ensino adequada deriva do fato de que o estado de conhecimento do estudante não é totalmente observável pelo sistema tutor, e em decorrência dessas incertezas, a modelagem do STI como um POMDP é utilizada para auxiliar nas tomadas de decisão.

Uma dificuldade encontrada na implementação de modelos POMDP consiste no tamanho do conjunto de espaço de estados a ser explorado, que cresce exponencialmente a medida que aumentam os conceitos a serem ensinados. Tal problema é comumente conhecida na literatura especializada como *"Curse of Dimensionality"* (Maldição da dimensionalidade). Com o objetivo de aplicar modelos POMDP em um espaço de estados de tamanho razoável , Folsom-Kovarik (Folsom-Kovarik et al. 2010) aborda duas representações alternativas que simplificam o tamanho do modelo e a quantidade de informações a serem armazenadas pelo modelo do estudante, resultando em uma performance superior quando a complexidade dos dados armazenados neste modelo cresce.

No trabalho de (Theocharous et al. 2009), um STI chamado SPAIS(*Socially and Physically Aware Interaction Systems*) utiliza moedas virtuais para ensinar conceitos de ordenação. Este sistema é modelado como um POMDP onde variáveis físicas (como o progresso da tarefa e o comportamento atual do estudante) representam os estados e as probabilidades de transição são definidas por variáveis sociais (especialidade do estudante, compreensão, etc). O problema da maldição de dimensionalidade também se apresenta neste domínio e é solucionado pela quebra do problema em soluções menores, cada qual representando a melhor forma de ensinar diferentes tipos de estudantes.

2.2 Modelos de rastreabilidade do conhecimento

Em um sistema educacional adaptativo é necessário prover um modelo que permita realizar o acompanhamento cognitivo, considerando os ganhos de conhecimentos providos pelas ações instrucionais fornecidas pelo tutor. A primeira tentativa de se estabelecer tal acompanhamento foi realizado por Corbett em (Corbett e Anderson 1994), que desenvolveu um modelo de estimativas probabilísticas sobre a capacidade que o estudante possuia de aplicar regras de produção na linguagem de programação Lisp. A avaliação sobre a corretude da aplicação das regras de produção foi construída por um modelo de estudante ideal, uma base de conhecimento que fazia a comparação das respostas fornecidas pelo estudante com as regras geradas pelo seu próprio motor de produção de regras, provendo assistência quando necessário. O resultado desta comparação era utilizado para manter a rastreabilidade do processo de aprendizagem. O modelo de aquisição cognitivo que mantinha a estimativa de probabilidade sobre as habilidades do estudante foi chamado pelos autores de *Knowledge Tracing*. Neste trabalho criamos uma especificação de planejamento que faz uso das equações do modelo proposto por Corbett. Baseado no modelo *Knowledge Tracing*, derivaram-se diversas outras técnicas que fazem uso de Inteligência Artificial. Em **??**), por exemplo, os autores exploram redes neurais para modelar a construção da aprendizagem. Outra representação comum é conhecida por *BKT Bayesian Knowledge Tracing*, e utiliza modelos Markovianos. Desafios relacionados a esta representação é explorada em trabalhos como os de **??**) e **??**).

2.3 Planejamento e extração de características

Para a obtenção de uma boa estratégia de ensino, e consequentemente, prover ações instrucionais apropriadas, um STI deve incluir um módulo de planejamento que fará uso do conhecimento do domínio e também do modelo do estudante para planejar as ações futuras. Entretanto, a escolha do planejador não é simples. Encontrar a solução que seja mais adequada para resolver um dado problema pode ser facilitado pelo uso de um portfólio de algoritmos. Com base em um conjunto de problemas especificados mediante uma linguagem de representação e em um portfólio, pode-se desenvolver um sistema que aprenda o melhor algoritmo a ser empregado para se resolver um dado problema. Nessa linha de pesquisa, destacam-se trabalhos como o de Vrakas (Vrakas et al. 2003), que aplica técnicas de aprendizagem de máquina para descobrir associações entre características de problemas de planejamento e parâmetros de configuração para otimização em tempo real do planejador HAP, desenvolvido pelos próprios autores. As associações descobertas por este procedimento são embutidas no planejador na forma de um sistema baseado em regras. Dessa forma, o planejador possui a capacidade de dinamicamente alterar os seus próprios parâmetros, para que possa obter um desempenho mais satisfatório de acordo

com as características particulares de cada problema.

Roberts, Howe e Flom (42) realizaram um estudo profundo sobre a capacidade de avaliação de performances de modelos preditivos. O trabalho consistia em verificar se modelos preditivos gerados a partir de características da especificação eram precisos e se estes modelos poderiam ser utilizados para aprimorar a performance dos planejadores. Os autores analisaram os resultados da execução de 28 planejadores e selecionaram dentro de um conjunto de domínios de problemas especificados na linguagem STRIPS PDDL, aqueles que eram os mais difíceis de serem resolvidos. Utilizando algoritmos de aprendizagem de máquina, identificaram 32 características aplicáveis ao planejamento clássico e geraram modelos preditivos. Por fim, foi realizada a comparação dos resultados da execução dos planejadores com os previstos pelo modelo.

O presente trabalho se distingue de Roberts em dois pontos: (i)os critérios adotados para avaliação de performance. Os autores avaliam o desempenho por dois critérios, sucesso na geração de um plano de execução válido e o tempo para a obtenção deste plano, enquanto avaliamos a performance sobre a perspectiva da obtenção de recompensas obtidas pela execução do plano. (ii) Roberts realiza o seu experimento considerando as peculiaridades do planejamento clássico, que se distingue sobretudo na identificação das características da especificação do planejamento probabilístico o qual focamos neste estudo.

Alhossaini (Alhossaini e Beck 2013) também extraiu caracterısticas de alguns problemas tentando definir o melhor planejador para resolver um determinado problema, porém, diferentemente de **??**), foram extraídas características específicas dos domínios, como quantidade de carros em problemas de transporte, ou quantidade de blocos no problema do mundo dos blocos. Sua abordagem, por selecionar características específicas do problema, não obteve êxito quando aplicado em um planejador de propósito geral.

Capítulo 3

Aspectos Teóricos

Neste capítulo é discutido os aspectos teóricos relacionados a proposta deste trabalho, que possibilitarão uma melhor compreensão do tema. O capítulo foi estruturado da seguinte forma: a Seção 3.1 apresenta uma definição de Sistemas Tutores Inteligentes e a arquitetura dos seus principais módulos componentes; na seção 3.2 são apresentados conceitos de planejamento, classificação e suas diferentes estratégias de execução; paradigmas de aprendizagem e problemas de decisão sequencial são discutidos na seção 3.3; nas seções 3.4 e 3.5 são discutidos os problemas de decisão Markovianos e os problemas de decisão Markovianos Parcialmente Observáveis, respectivamente.

3.1 Sistemas Tutores Inteligentes

Sistemas Tutores Inteligentes (STIs) são sistemas instrucionais baseados em computador que incorpora técnicas de Inteligência Artificial para simular o processo do pensamento humano na resolução de problemas ou em tomadas de decisão (Fowler 1991).

Segundo Jonassen (Jonassen e Wang 1993), um STI deve passar em três testes antes de ser considerado inteligente :

1. Conteúdo do tema ou especialidade deve ser codificado de modo que o sistema possa acessar as informações, fazer inferências ou resolver problemas.

2. Sistema deve ser capaz de avaliar a aquisição deste conhecimento pelo aluno.

3. As estratégias tutoriais devem ser projetadas para reduzir a discrepância entre o conhecimento do especialista e o conhecimento do aluno.

As características mais importantes de um STI segundo Lonaiz (Loinaz 2001) são:

1. O conhecimento do domínio estar restrito e claramente articulado.

2. Possuir conhecimento do aluno de forma que lhe permita dirigir e adaptar o ensino.

3. A sequência do ensino não deve ser predeterminada pelo designer instrucional.

4. Realizar processos de diagnóstico mais adaptados ao aluno e mais detalhados.

5. Permitir que o aluno realize perguntas ao agente tutor.

Genericamente, os STI conseguem representar separadamente a matéria que se ensina (modelo do domínio) e as estratégias para ensiná-la (modelo pedagógico). Além disso, possuem a capacidade de mapear as características específicas de cada estudante (modelo do aluno). Tais características incluem: (i) os conceitos aprendidos por ele, ou seja, suas habilidades; (ii) o estilo de aprendizagem deste aluno, se ele prefere que o conteúdo seja apresentado em forma de texto ou em forma de vídeo; (iii) os seus estados afetivos (JAQUES et al.,2011), entre outros. Tem ainda a capacidade de inferir o conhecimento e outras características dos alunos através de técnicas matemáticas como cálculos de probabilidade (Corbett e Anderson 1994) ou utilizando recursos da Inteligência Artificial como as redes Bayesianas (Manske e Conati 2005). Outra característica marcante é a necessidade da interface de comunicação ser um módulo bem planejado, de fácil manipulação, e que favoreça o processo de comunicação tutor-aluno (Seffrin 2015).

3.1.1 Arquitetura de um STI

Existem várias abordagens alternativas de arquitetura para STIs. Entre as mais populares destacam-se as arquiteturas proposta por Clancey (William 1987),Wenger (Wenger 1987), Kaplan (Kaplan e Rock 1995), Self (Self 1998) e McTaggart (McTAGGART 2001). Neste trabalho é descrita apenas a abordagem clássica (também conhecida como função tripartida ou arquitetura tradicional de STI) por ser recorrentemente adotada. Esta arquitetura é estruturada em quatro módulos funcionais:

- Modelo do estudante: este módulo armazena informações específicas para cada estudante de forma individual, tais como o seu nível de compreensão sobre os conceitos em uma disciplina e/ou as suas habilidades cognitivas.

- Modelo do Domínio: representa o conhecimento a ser lecionado ao estudante. Pode ser usado de diversas maneiras tais como: responder a perguntas dos alunos, gerar problemas (ou exemplos, sugestões) a serem repassados para o aluno e diagnosticar o modelo de estudante.

- Modelo Pedagógico: também conhecido como Planejador Instrucional ou Módulo Tutor, é utilizado para obter a próxima instrução instrucional a ser repassada ao aluno.

- Interface: fornece ao aluno canais para se comunicar com o STI.

Uma representação da interação entre os módulos na arquitetura clássica de um STI pode ser visualizada na Figura 1.

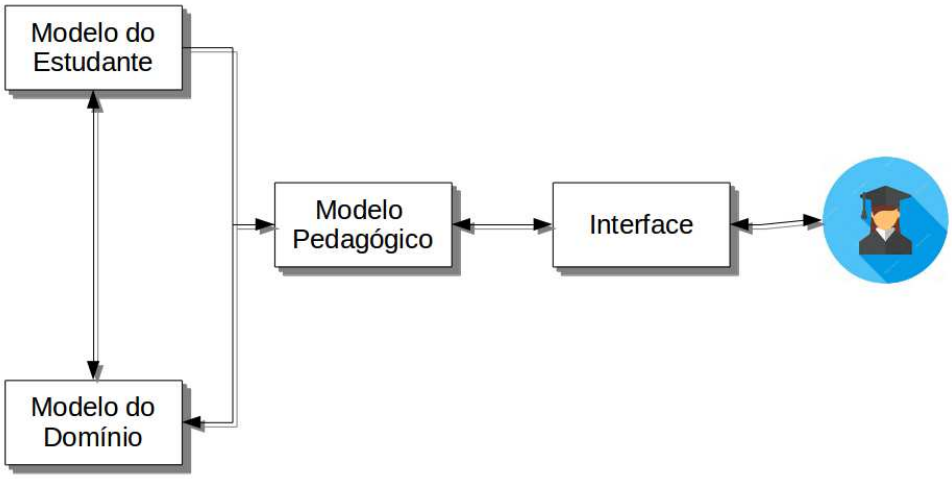

Figura 1 – Arquitetura Clássica de um STI

3.1.1.1 Modelo do Domínio

O Modelo do Domínio é fundamentalmente uma base de conhecimento contendo informações de um determinado domínio, organizada de alguma maneira para representar o conhecimento de um especialista ou professor (Pozzebon et al. 2003). Por domínio compreende-se qualquer área de conhecimento, como Cálculo Diferencial ou Álgebra Linear.

Um desafio na definição do Modelo do Domínio consiste na representação da informação. A dificuldade provém do fato de que, mais do que ser simplesmente uma fonte de dados, o Modelo do Domínio desempenha a função de gerar soluções para problemas seguindo o mesmo procedimento que um estudante deveria realizar. As respostas do estudante são comparadas com as etapas envolvidas na resolução da tarefa, dessa forma identificando os pontos de falhas conceituais no Modelo do Estudante. Além disso, a informação deve considerar o contexto em que o STI está inserido, contendo somente conteúdos que são relevantes para o modelo. Em outras palavras, é necessário que o domínio armazenado seja simplificado, evitando dificuldades além das exigidas, como por exemplo inserir equações envolvendo números complexos ou funções trigonométricas em uma sessão de tutoria cujo público alvo são estudantes do ensino fundamental.

De acordo com Rickel (Rickel 1989), não existe uma forma geral adequada para representar o conhecimento, mas tipos diferentes de raciocínio e de conhecimento, requerem diferentes representações para um uso eficiente e eficaz. Portanto, a escolha da representação de conhecimento em um sistema tutorial depende do tipo de conhecimento a ser

armazenado e da utilização pretendida. Alguns métodos de IA utilizados para representar o conhecimento do domínio, inclui o desenvolvimento de rede semânticas, a aplicação de regras de produção, representações procedimentais, e a construção de molduras (”frames”) e roteiros (”scripts”) (Pozzebon et al. 2003).

3.1.1.2 Modelo do Estudante

O Modelo do Estudante é o módulo do STI que armazena todas as informações relevantes sobre cada estudante. É através do Modelo do Estudante que o STI consegue fornecer uma tutoria adaptativa, pois leva em consideração cada peculiaridade que é importante considerar em uma sessão de ensino. A partir desse modelo e do conteúdo a ser ensinado, o sistema deve ser capaz de inferir a melhor estratégia de ensino a ser utilizada.

Dentre as informações que o Modelo do Estudante pode armazenar, incluem: nível de conhecimento, capacidade de aprendizagem, preferências, adequações de estilos de aprendizagem, etc. Muitos STIs também consideram os estados afetivos (emoções), como motivação, engajamento e confiança e as habilidades sociais. A Figura 2 representa algumas destas características obtidas a partir de estudantes humanos que podem ser utilizadas pelo STI para atualizar o seu Modelo de Estudante.

Figura 2 – Representação do Modelo do Estudante (Silverio 2017)

O STI infere o conhecimento do estudante quando este responde às avaliações propostas. É neste momento que o aluno demonstra o seu conhecimento ao sistema, sendo os dados da resolução das tarefas as únicas evidências que o sistema obtém (Vanlehn 2006). Essas evidências atualizam o modelo de inferência do sistema, que retorna informações sobre quais habilidades foram exercitadas.

Existem algumas técnicas que possibilitam inferir o conhecimento do estudante. Uma delas é a *Knowledge Tracing* e foi proposta por Corbett (Corbett e Anderson 1994). Neste trabalho, o autor utiliza um modelo de regras de produção para ensinar a linguagem de programação LISP usando um sistema tutor que infere a probabilidade que o estudante tem de conhecer uma regra através de equações matemáticas baseadas em evidências, que correspondem a ações corretas e incorretas realizadas em um dado momento. Estas

equações serão explicadas com mais detalhes no Capítulo 4, onde será discutido o modelo utilizado para realizar o planejamento automatizado das ações pedagógicas.

3.1.1.3 Modelo Pedagógico

Essencialmente um STI deve recomendar ações pedagógicas apropriadas ao Modelo do Estudante. A responsabilidade pelas estratégias que resultam na escolha dessas ações pertence ao Modelo Pedagógico. Ele também atua como um controlador que se comunica com todos os outros demais módulos do sistema: O Modelo Pedagógico é acessado pela interface com o usuário, que fornece respostas aos exercícios propostos. As respostas são comparadas com a resolução das tarefas, de conhecimento do Modelo do Domínio. Da comparação resulta na atualização da inferência de conhecimento no Modelo do Estudante. Por fim, um *feedback* é fornecido pela interface.

O Tutor pode interferir uma vez que ele detecte uma quantidade expressiva de erros em uma determinada habilidade, significando que algum conteúdo anterior não foi compreendido e, portanto, necessita de revisão. O Tutor, então, poderá apresentar materiais instrucionais que reforcem o aprendizado desta habilidade.

A eficiência de um Sistema Tutor Inteligente depende das estratégias pedagógicas utilizadas. Através destas estratégias as respostas são adaptadas individualmente conforme o estilo de aprendizagem presente no modelo do estudante. As estratégias ainda são responsáveis por determinar sobre quando é necessário intervir na aprendizagem, com base em características tais como o conhecimento presumido do estudante e as suas emoções. Elas devem ser personalizadas conforme as necessidades individuais: para um estudante pode ser necessário estimular a sua motivação, um outro pode necessitar de alguma ajuda cognitiva e para um terceiro, pode ser necessário treinar habilidades básicas. No livro de Woolf (Woolf 2007)), as estratégias de ensino são classificadas em três grupos distintos: estratégias baseadas no ensino humano, estratégias baseadas na teoria da aprendizagem e agentes pedagógicos animados.

Os modelos de estratégias baseados no ensino humano são empíricos e tem como referência as observações de ensino com professores humanos. Algumas estratégias deste modelo incluem:

- **treinamento de aprendizagem**: o princípio básico que norteia essa abordagem se caracteriza como um expert que monitora o desempenho do estudante, fornecendo dicas sob demanda e suportando diversas soluções válidas. A responsabilidade do estudo é delegada ao estudante e o tutor só intervém quando requisitado. A solução não exige uma sequência de passos determinados, podendo ser obtida por diversos caminhos. Exemplos: Medicina, Pilotagem, etc

- **resolução de problemas**: é aplicado para domínios complexos onde a solução da tarefa exige muitos passos bem-definidos. Exemplos: matemática, física, linguagem

de programação, etc.

O segundo grupo compreende as Estratégias baseadas em Teorias de Aprendizagem. Estas são teorias desenvolvidas através da colaboração entre cientistas cognitivos, educadores naturalistas e filósofos, e apresentam novas possibilidades de ensino (Woolf 2007). O uso de tais teorias em STIs permite a verificação da sua eficácia. A seguir, são descritas cinco teorias.

- Baseada na teoria socrática, uma forma de ensino grega e antiga, baseia-se no princípio de que cada pessoa possui a essência das ideias e respostas para todos os problemas do universo (WOOLF; MCDONALD, 1984). A técnica baseia-se no diálogo e consiste em uma discussão entre duas pessoas, na qual o tutor irá propor questões, relacionadas a um tema central, ao aluno. Tais questões tem o objetivo de fazer o aluno refletir sobre o tema e tirar suas próprias conclusões e com isso aprender sobre o tema.

- Teoria Cognitiva: Teoria adotada por STIs de sucesso como o Andes e o *Cognitive Tutor*, ela visa ao ensino da forma mais eficiente possível, baseando-se na identificação de processos mentais. Para isso, cada tarefa é subdividida em unidades simples (etapas), e a partir delas são elaboradas instruções que guiam o aluno (WOOLF, 2008), frente a novas informações, comparam-na com o conhecimento já adquirido e buscam estabelecer relações entre elas. Este processo ocorre em três estágios: (i) Recebimento de informação pelos sensores de entrada; (ii) Armazenamento da mesma na memória de curto prazo e (iii) Transferência da informação da memória de curto prazo para a memória de longo prazo, seja através de memorização ou através de prática. A implementação desta técnica exige a definição e a implementação das habilidades cognitivas no STI.

- Teoria Construtivista: Desenvolvida por Piaget, define o processo de aprendizado como um conjunto de disciplinas, que toda pessoa atravessa ao longo da vida. Em cada etapa, o processo de ensino deve condizer com a idade dos alunos. Por exemplo, segundo a teoria, dos 8 aos 11 anos, a criança deve lidar com objetos concretos, dessa forma, frações podem ser ensinadas através de blocos e figuras. Quando ela atinge os 12 anos, entra o pensamento abstrato, então, frações podem ser ensinadas através de fórmulas e números decimais. Esta teoria defende que o aluno deve buscar o conhecimento por si, através de relações sociais e troca de ideias.

- Aprendizado Situado: O processo de aprendizagem é influenciado pelo contexto em que o aluno está inserido (contexto cultural e social). Ocorre através da troca de experiências entre os indivíduos mais novos e os mais experientes. Esta estratégia contrasta com o ensino em sala de aula que é, muitas vezes, abstrato e fora do contexto do alunos. Tutores que implementam esta estratégia são muito utilizados

no meio militar, em programas de treinamento. Nestes sistemas, é apresentado ao aluno um ambiente, e o tutor deverá ensiná-lo a interagir com esse ambiente. Uma vez treinado, quando o aluno estiver em uma situação real, ele saberá o que fazer.

- Interação Social e Zona de Desenvolvimento Proximal: Definida por Vygotsky e, assim como o construtivismo e o aprendizado situado, também tem como ponto fundamental a interação social. É combinada com a Zona de Desenvolvimento Proximal (ZPD), que define um nível que as crianças atingem quando estão engajadas em um aprendizado colaborativo. Apresenta um ganho de aprendizado maior do que um aprendizado sem colaboração, sendo similar ao aprendizado situado. O objetivo dos STIs que utilizam esta estratégia é manter os alunos dentro da ZPD. O aluno pode sair por diversos fatores, seja por ele estar entediado, confuso ou não conseguir prosseguir devido à alguma dificuldade. Cabe ao STI adaptar o seu conteúdo de modo a manter o aluno interessado, por exemplo, provendo exercícios nem muito fáceis, de modo a evitar o tédio, e nem muito difíceis, que impeçam o aluno de prosseguir ou que possam confundi-lo.

Por fim, há as estratégias que têm como base o emprego da tecnologia. Nesse contexto, encontram-se os Agentes Pedagógicos Animados, que são personagens animados com o propósito de motivar o aluno durante o seu estudo. Eles fazem uso da IA para guiar as suas ações e possuem os seus próprios objetivos e crenças (Woolf 2007). Interagem com o ambiente em que estão situados, como a interface do STI, podendo exibir mensagens na tela, como os *feedbacks* ou as dicas providas pelo Tutor.

3.1.1.4 Interface

A interface é o módulo de comunicação entre o estudante com o sistema. Para que o ensino não seja desestimulante, nem provoque reações que não sejam propícias ao ensino, a interface deve ser simples e intuitiva. Se a interface tem muitas restrições de recursos ou demasiada complexidade, o aluno terá que gastar um tempo considerável para aprender a utilizar o sistema, tempo este em que ele poderia estar interagindo com o Tutor e aprendendo (Polson e Richardson 2013).

A comunicação visual desempenha um importante papel no processo de aprendizagem. O design das Interfaces Gráficas dos Usuários (GUIs, do equivalente em inglês *Graphical User Interface* podem exercer grande influência no desempenho do aluno (Woolf 2007). Por exemplo, se utilizar botões com ícones, eles devem transmitir a ideia exata da funcionalidade. Uma interface intuitiva reduz a chamada carga cognitiva, a quantidade de informação que o aluno necessita memorizar para interagir com o sistema (Reis et al. 2012)

Alguns dos itens que se seguem estão numa lista de recomendação exposto por Gruszynski (Gruszynski 2008), e que devem ser considerados em projeto visual para sistemas baseados em web:

- ❏ Economia no uso de diferentes fontes tipográficas;
- ❏ Utilização de um sistema de grid ou similar que assegure a ordenação racional do projeto de modo a garantir sua unidade;
- ❏ Legibilidade, clareza, hierarquia (ordenação) e facilidade de decodificação pela repetição dos signos utilizados, permitindo o rápido entendimento por parte do leitor / receptor.

3.2 Planejamento

Planejamento é um processo de deliberação explícita que escolhe e organiza ações, antecipando os resultados esperados. Esta deliberação visa atingir da melhor forma possível, alguns objetivos (14). A habilidade de planejar tarefas é um aspecto fundamental do comportamento inteligente e sua automatização tem sido um dos principais objetivos da pesquisa realizada na área de Inteligência Artificial (Russell e Norvig 2016).

Um plano é uma sequência de ações que, ao aplicadas em um estado inicial, o altera de forma que ao fim da aplicação dessas ações atenda a meta do problema atual. As transições entre estados intermediários são o resultado da execução de cada ação, que consome recursos existentes e produz novos recursos. Um sistema que gera planos para um problema é chamado planejador. (14).

Um problema de planejamento geralmente é definido a partir da especificação do estado inicial do problema, da meta a ser satisfeita e das ações que podem ser aplicadas a um estado do problema. Uma ação é caracterizada por pré-condições e efeitos. Quando um estado do problema satisfaz as pré-condições de uma ação isto torna possível sua execução resultando num estado em que os efeitos associados a ela se tornam verdadeiros (Weld 1999). Formalmente, sua definição é composta por uma 5-upla (Branquinho et al. 2009):

1. S : conjunto finito dos estados;
2. i ∈ S: estado inicial;
3. A: conjunto finito de ações;
4. g: meta;
5. P: solução do problema ou plano;

Considerando esta definição, um estado é representado por um conjunto de proposições verdadeiras P. Uma ação $a \in$ A é representada por uma tupla(*pre(a),add(a),del(a)*), onde *pre(a)* é um conjunto de pré-condições: proposições que devem ser verdadeiras para que a possa ser executada, *add(a)* é um conjunto de proposições que se tornarão verdadeiras

após a execução de *a* e *del(a)* é o conjunto de proposições que se tornarão falsas após a execução de *a*.

Existem dois campos de estudos relacionados ao planejamento. No planejamento clássico, assume-se que o ambiente é estático, determinístico e totalmente observável. Nessas condições, a avaliação de desempenho dos planos costuma ser restritas a questões temporais. No planejamento não-clássico (dentre eles o planejamento probabilístico, que especifica probabilidades associadas as transições de estados) por sua vez, são considerados ambientes parcialmente observados ou estocásticos e envolve conjuntos de algoritmos e agentes.

Quanto as estratégias de execução, o planejamento pode ser *offline* ou *online*. Quando o agente segue uma estratégia de execução *offline*, todo o plano de ação é executado *a priori*, delegando a sua execução somente após a conclusão de todo o plano. Devido a necessidade de antecipar todas as ações possíveis, a elaboração do plano de execução costuma ser muito demorada, despendindo um tempo significante para resolver problemas maiores. Além disso, o planejamento é sensível a qualquer mudança no ambiente, requerindo que toda a política de planejamento seja recalculada quando esta situação ocorre.

A estratégia de execução *online* intercala o planejamento e a execução. Por considerar um horizonte de planejamento menor em cada fase, consegue-se obter uma drástica redução no tempo de planejamento quando comparada à abordagem *offline*. Uma mudança na dinâmica do ambiente impacta apenas a fase de planejamento em execução, o que torna a sua utilização mais propensa para ambientes não-estáticos. Uma desvantagem do planejamento *online* é que restrições de tomadas de decisões em tempo-real podem ser um empecilho para a sua aplica (Ross et al. 2008). Combinações das duas abordagens são frequentemente encontradas em planejadores autônomos.

A adoção de um formalismo comum para descrever domínios de planejamento promove uma maior reutilização das pesquisas e permite a comparação mais direta de sistemas e abordagens de planejamento, ao mesmo tempo que possibilita uma evolução dos planejadores. A representação de problemas de planejamento deve ser feita por uma linguagem que seja eficiente e que possa ser utilizada para representar diversos tipos de problemas (Russell e Norvig 2016). Para que uma linguagem seja eficiente, ela deve ser:

- ❏ suficientemente expressiva para descrever uma grande variedade de problemas.
- ❏ suficientemente restritiva para permitir que algoritmos eficientes operem sobre ela.

3.2.1 Evolução dos Formalismos de Planejamento

A seguir é descrito brevemente o histórico de algumas das principais linguagens de especificação de planejamento autônomo. Cada uma delas é explicada utilizando como exemplo o problema do transporte aéreo de carga. Neste problema temos duas cargas,

C1 localizada no aeroporto de São Francisco (SFO) e C2 localizada no aeroporto John F. Kennedy (JFK). C1 deve ser transportada para JFK e C2 deve ser transportada para SFO, por dois aviões, cada um localizado em um aeroporto.

A primeira linguagem formal para descrever domínios e problemas de planejamento foi a linguagem STRIPS (***ST****anford* ***R****esearch* ***I****nstitute* ***P****roblem* ***S****olver*)(Fikes e Nilsson 1971), derivada do cálculo proposicional. Uma instância neste formalismo é composto por:

- ❏ um estado inicial.
- ❏ um ou mais ou estados objetivos.
- ❏ um conjunto de ações, cada uma delas incluindo pré-condições e pós-condições.

Os estados são representados como conjunções de literais positivos, que podem ser literais proposicionais (por exemplo: Azul ∧ Desconhecido) ou literais sem variáveis e sem funções (por exemplo: Em(Aviao1, JFK) ∧ Em(Aviao2,SFO)). Assume-se a premissa do mundo fechado, isto é, qualquer condição não mencionada por um estado é considerada falsa. A especificação do problema de Transporte de Carga Aérea na linguagem STRIPS é demonstrada na listagem 3.1.

```
Init (Em(C1, SFO) ∧ Em(C2, JFK) ∧ Em(P1, SFO) ∧ Em(P2, JFK) ∧
   Carga(C1) ∧ Carga(C2) ∧ Aviao(P1) ∧ Aviao(P2) ∧ Aeroporto(JFK)
    ∧ Aeroporto(SFO))
Goal (Em(C1, JFK) ∧ Em(C2, SFO))
Action (
    Carregar(c, p, a),
    PRECOND: At(c, a) ∧ At(p, a) ∧ Cargo(c) ∧ Aviao(p) ∧
       Aeroporto(a)
    EFFECT: ¬ At(c, a) ∧ In(c, p)
    )

Action (
    Descarregar(c, p, a),
    PRECOND: Dentro(c, p) ∧ Em(p, a) ∧ Carga(c) ∧ Aviao(p) ∧
       Aeroporto(a)
    EFFECT: Em(c, a) ∧ ¬Dentro(c, p)
    )

Action (
    Voar(p, origem, destino),
    PRECOND: Em(p, origem) ∧ Aviao(p) ∧ Aeroporto(origem) ∧
       Aeroporto(destino)
```

```
    EFFECT: ¬ Em(p, origem) ∧ Em(p, destino)
    )
```

Listagem 3.1 – Especificação do problema de Transporte de Carga Aérea na linguagem STRIPS

A linguagem Action Description Language (ADL) (Pednault 1989) foi desenvolvida como uma evolução da linguagem STRIPS. A ADL é menos restritiva que a sua antecessora , possibilitando que os estados sejam especificando com disjunções, negações e também quantificadores lógicos. Na ADL,assume-se a premissa do mundo aberto, onde os literais não mencionados são interpretados como desconhecidos. A especificação do problema de Transporte de Carga Aérea na linguagem ADL é demonstrada na listagem 3.2.

```
        Init (Em(C1, SFO) ∧ Em(C2, JFK) ∧ Em(P1, SFO) ∧ Em(P2,
           JFK) ∧ (C1:Carga) ∧ (C2:Carga) ∧ (P1:Aviao) ∧ (P2:
           Aviao) ∧ (JFK:Airport) ∧ (SFO:Airport) ∧ (SFO ≠ JFK))
Goal (Em(C1, JFK) ∧ Em(C2, SFO))

Action (
    Carregar(c: Carga, p: Aviao, a: Aeroporto),
    PRECOND: Em(c, a) ∧ Em(p, a)
    EFFECT: ¬ Em(c, a) ∧ Dentro(c, p)
)
Action (
    Descarregar(c: Carga, p: Aviao, a: Aeroporto),
    PRECOND: Dentro(c, p) ∧ Em(p, a)
    EFFECT: Em(c, a) ∧ ¬ Dentro(c, p)
)
Action (
    Voar(p: Aviao, origem: Aeroporto, destino: Aeroporto),
    PRECOND: Em(p, origem) ∧ (origem ≠ destino)
    EFFECT: ¬ Em(p, origem) ∧ Em(p, destino)
)
```

Listagem 3.2 – Especificação do problema de Transporte de Carga Aérea na linguagem ADL

A linguagem Planning Domain Definition Language (PDDL) (McDermott et al. 1998)foi desenvolvida com o intuito de padronizar as especificações de linguagem de planejamento clássico e tem sido adotada nas principais competições do gênero, entre elas o International Planning Competition (IPC) (Long e Fox 2003). Problemas especificados na linguagem PDDL são divididos em dois arquivos separados: um arquivo de domínio (contendo tipos,

predicados, funções e ações) e outro arquivo de problema (contendo os objetos, estado inicial e a especificação do objetivo). As definições de domínio e de problema são apresentadas nas listagens 3.3 e 3.4, respectivamente.

```
(define (domain air-cargo)
  (:requirements :strips)
  (:predicates (Dentro ?obj ?localizacao)
           (Em ?obj ?localizacao)
           (Carga ?obj)
           (Aviao ?obj)
           (Aeroporto ?obj))

  (:action carregar
     :parameters (?c ?p ?a)
     :precondition (and (Em ?c ?a) (Em ?p ?a)
             (Carga ?c) (Aviao ?p) (Aeroporto ?a))
     :effect (and (Dentro ?c ?p) (not (Em ?c ?a))))

  (:action descarregar
     :parameters (?c ?p ?a)
     :precondition (and (Dentro ?c ?p) (Em ?p ?a)
             (Carga ?c) (Aviao ?p) (Aeroporto ?a))
     :effect (and (Em ?c ?a) (not (Dentro ?c ?p))))

  (:action voar
     :parameters (?p ?from ?to)
     :precondition (and (Em ?p ?from)
             (Aviao ?p) (Aeroporto ?from) (Aeroporto ?to))
     :effect (and (Em ?p ?to) (not (Em ?p ?from))))
  )
```

Listagem 3.3 – Especificação de um domínio de Transporte de Carga Aérea na linguagem PDDL

```
(define (problem pb1)
  (:domain air-cargo)
  (:objects C1 C2
        P1 P2
        SFO JFK)
  (:init
   ;; tipos
   (Carga C1) (Carga C2)
   (Aviao P1) (Aviao P2)
   (Aeroporto SFO) (Aeroporto JFK)

   ;; locations
   (Em C1 SFO) (Em C2 JFK) (Em P1 SFO) (Em P2 JFK))

  (:goal
(and (Em C1 JFK) (Em C2 SFO))))
```

Listagem 3.4 – Especificação de um problema de Transporte de Carga Aérea na linguagem PDDL

No capítulo 4 é descrito um modelo de tutor que poderá ser utilizado como referência para um sistema educacional. Em sua especificação, foi feita a opção pela linguagem Relational Dynamic Influence Diagram Language (RDDL) (Sanner 2010). Esta linguagem especifica estados, ações e observações como variáveis parametrizáveis e a evolução do mundo é condicionado somente aos seus valores atuais. Semanticamente a RDDL é simplesmente uma rede bayesiana dinâmica(*DBN*). A opção pelo formalismo RDDL justifica-se por duas razões: (i) o modelo de transições estocásticas e de observações são mais naturais nesta linguagem do que em suas precedentes (ii) a linguagem tem sido adotada pelas principais competições de planejamento não-clássico, como o IPPC *International Probabilistic Planning Competition*. Em razão de nossa escolha, descreveremos o formalismo RDDL em mais detalhes na subseção 3.2.2.

3.2.2 O formalismo RDDL

A RDDL é uma linguagem uniforme de especificação, desenvolvida com intuito de resolver problemas de planejamento probabilístico e que também permite criar bons modelos de representação do tipo POMDP. Ela se diferencia de outras linguagens com a mesma finalidade, como os da família PDDL , por possuir mecanismos que permitem especificar características que são de difícil modelagem em outras linguagens, como por exemplo: eventos independentes, parcialmente observáveis, concorrentes, estocásticos e exógenos.

Um arquivo de especificação RDDL é categorizado em três seções principais: *Domains*,

onde os detalhes sobre o domínio são declarados; *Non-fluents*, que apresentam parâmetros não-fluentes (parâmetros cujo os valores não são alterados durante a execução do planejamento) ; e *Instances*: que representa uma instância aplicável ao domínio. As três seções serão detalhadas a seguir:

3.2.2.1 Domains

O arquivo de domínio é usualmente descrito em um arquivo separado dos demais. O molde desta subseção é especificado da seguinte forma:

```
domain <nome_dominio> {
        requirements = { <req_1> , ... , <req_n> } ;
        types {
                <nome_tipo> : <tipo_antecessor> ;
        }
        pvariables {
                <nome_var> : { <tipo_var> , <tipo_dados>
                   ,
                        default = <default> } ;
        }
        cpfs {
                <nome_cpf> ( <parametros> ) = <funcao> ;
        }
        reward = <funcao> ;
        state-action-constraints  = <funcao>{
        }
}
```

Inicialmente é necessário nomear o domínio. As subseções seguintes são:

- *requirements*: parâmetros exigidos para que o parser RDDL possa interpretar o arquivo. Ex: *reward-deterministic* (indica que avaliação dos resultados é baseada em uma função de recompensa), concurrent (para domínios tenha concorrência entre ações), constrained-state (estados que possuem algum tipo de restrição) e partially-observed (ambientes parcialmente observáveis), etc.
- *types*: declara os tipos das variavés e seus valores padrão.
- pvariables: variavés parametrizadas, podendo ser variáveis non-fluents (quando representam fatos estáticos), state-fluent(quando representam fatos dinâmicos) , action-fluent (operadores que controlam o comportamento das funções) e observ-fluent (quando representam as possíveis observações).
- cpfs: acrônimo para conditional probabilistic functions. São definições das funções a serem aplicadas ao problema.

- *reward*: Função que define o valor de recompensa.
- *state-action-constraints*: restrições aplicáveis aos estados (opcional).

3.2.2.2 Non-Fluents

A subseção non-fluent define valores para as variáveis que não sofrem alteração durante a execução do algoritmo no tempo das decisões. A sua representação segue a seguinte estrutura:

```
non-fluents <nome_nf> {
        domain = <nome_dominio> ;
        objects {
                <tipo_objeto> : { <nome_objeto> } ;
        };
        non-fluents {
                // Aqui sao listados os fatos estaticos.
        };
}
```

Na declaração da seção non-fluents, são definidos os valores de nome e domínio a que pertence. A seguir, são listadas duas subseções: objects, contendo a listagem de todos os objetos que serão manipulados durante o problema e non-fluents, que apresenta a lista de fatos estáticos, como as constantes.

3.2.2.3 Instances

A seção instance apresenta uma instância a ser aplicada ao domínio. Ela referencia as duas outras seções anteriores e configura parâmetros de execução, como o alcance do horizonte e fator de desconto. Definido da seguinte forma:

```
instance <nome_instancia> {
        domain = <nome_dominio> ;
        non-fluent = <nome_nf> ;
        init-state {
                // Aqui sao listados os fatos dinamicos,
                // e tambem os estaticos, caso a secao
                // non-fluents nao seja declarada.
        }
        max-nondef-actions = <valor> ;
        horizon = <valor> ;
        discount = <valor> ;
}
```

- domain: nome do domínio a que pertence.

- non-fluent: referência a seção non-fluent.
- init-state: necessário apenas quando o non-fluent não for especificado. Determina os fatos dinâmicos e estáticos.
- max-nondef-actions: declara o número de ações concorrentes.
- horizon: define o tamanho do horizonte, ou seja, a quantidade de iterações no tempo.
- discount: configura o fator de desconto aplicável a recompensa ao longo do tempo.

3.3 Paradigmas de Aprendizagem e Processos de decisão sequencial

Um processo de decisão sequencial compreende um agente que interage de forma síncrona com um ambiente externo. O objetivo deste agente é maximizar a recompensa escolhendo ações apropriadas. Estas ações e a história dos estados determinam a distribuição de probabilidade sobre os próximos estados possíveis. Assim sendo,a sequência de estados do sistema pode ser modelada como um processo estocástico (Braziunas 2003), o que significa que o efeito das ações não é certo e nem determinado. Problemas de decisão sequencial incorporam utilidades, incerteza e percepção, e incluem os problemas de busca e planejamento como casos especiais (Russell e Norvig 2016).

Um problema de decisão sequencial normalmente envolve os seguintes passos:

1. Observar o estado do sistema;
2. Escolher e realizar uma ação (Sistema evolui para um novo estado);
3. Observar um reforço imediato;
4. Repetir os passos 1 a 3;

Em decorrência das ações serem essencialmente estocásticas, o ambiente nem sempre responderá como esperado. O agente deverá buscar um estado-meta baseado na imprevisibilidade do resultado de suas ações. Diante desta situação, o plano de execução das ações é o que chamamos de planejamento probabilístico.

A técnica de aprendizado por reforço baseia-se na idéia de que se uma ação é seguida de estados satisfatórios, ou por melhoria no estado, então a tendência para produzir esta ação é aumentada, isto é, reforçada. Uma forma de modelar problemas de aprendizado por reforço é utilizar o Processo de Decisão de Markov, onde um agente toma suas decisões com base em um sinal do ambiente, chamado de Estado do ambiente. Processos de Decisão Markovianos e Processos de Decisão Marlovianos Parcialmente Observáveis são modelos teóricos com capacidade para lidar com decisões sequencias.

A Figura 3 mostra como é realizada a interação de um agente com um ambiente quando utilizado a técnica de aprendizado por reforço. Em cada iteração do tempo, o ambiente se encontra em um estado s_n. O agente então escolhe uma ação a_n e o ambiente evolui para um novo estado s_{n+1}. Além disso, o agente obtém uma recompensa r_n pela execução de a_n no estado s_n que representa um estímulo quantitativo que indica o valor da decisão executada.

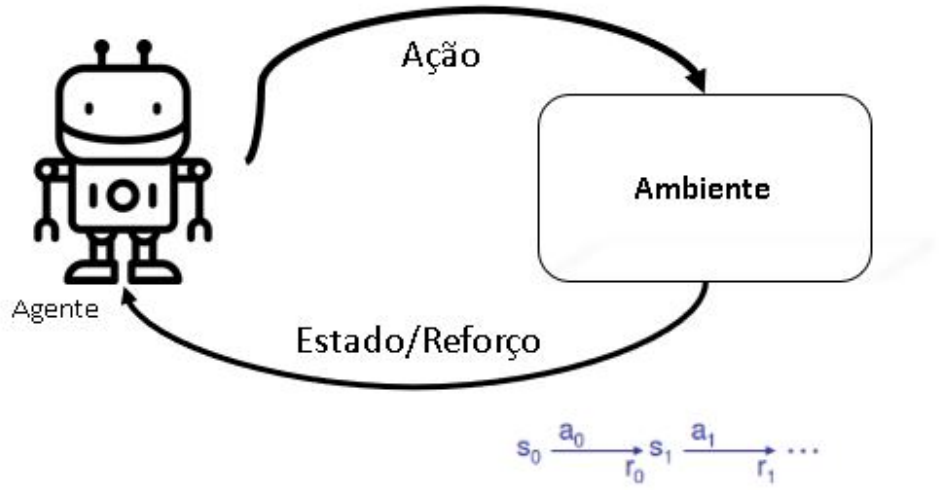

Figura 3 – Representação de Aprendizagem por Reforço

3.4 Processos de Decisão Markovianos (MDP)

Um modelo de representação comumente utilizado para tratar problemas de decisão sequencial, em que o ambiente é completamente observável, são os Markov Decision Process (MDP).

Mais precisamente, MDP é um processo de controle estocástico discreto no tempo onde um agente atua sobre um ambiente que lhe é visível e busca alcançar recompensas mais promissoras. Formalmente o modelo se caracteriza por uma 5-upla (S,A,T,R, γ), onde os elementos são definidos da seguinte forma:

- **S** : é o conjunto finito de todos os estados do ambiente.
- **A**: é o conjunto finito de ações possíveis.
- **T**: é o conjunto de todas as transições da forma t = (s'|s,a), t ∈ T, s ∈ S, s' ∈ S e a ∈ A. Ou seja, é a probabilidade de alcançar o estado s' executando a ação a no estado s.

- **R**: corresponde a uma função com imagem em $\mathbb{R}$ que atribui um valor de recompensa para cada ação executada em cada estado.

- γ: fator de desconto.

Um estado é uma descrição do ambiente em determinado momento do tempo (Braziunas 2003). Para que um processo seja considerado Markoviano, as transições devem depender apenas do estado corrente. Isto significa que o histórico dos estados do sistema é irrelevante para o futuro. Em termos matemáticos, tal afirmação pode ser descrita conforme a Equação 1.

$$Pr(S^{t+1}|S^{t}, S^{t-1}, ..., S^{0}) = Pr(S^{t+1}|S^{t}) \tag{1}$$

Em cada instante de tempo, o agente escolhe uma ação que poderá provocar uma mudança no estado de ambiente segundo uma probabilidade pré-definida P(s'|s,a), s $\in$ S, s'$\in$ S, a $\in$ A.

Podemos representar as probabilidades de transições de estado, através de um conjunto de matrizes de probabilidades, uma para cada ação e cada uma delas definida por $M_{|s|x|s|}$, onde cada elemento $M_{i,j}$ simbola a probabilidade de transição do estado S_i para o estado S_j (Braziunas 2003). Uma outra representação comumente utilizada são as redes bayesianas dinâmicas (Russell e Norvig 2016).

A história (também chamada de trajetória) compreende os registros das informações relevantes das decisões sequencias ao longo do tempo. Por exemplo, para se calcular as recompensas, o histórico dos estados visitados e das ações é suficiente, uma vez que o cálculo da recompensa baseia-se apenas nestes dados.Logo:

$$(s^0, a^0), (s^1, a^1), ..., (s^t, a^t) \tag{2}$$

Para completar a definição do ambiente, faz-se necessário determinar uma função de valor V (V: H $\rightarrow$ S) , correspondente a expectativa total das recompensas dado uma sequência finita do histórico do ambiente.

A solução de um MDP é encontrar um mapeamento, que forneça ações ótimas para cada estado do ambiente. Ações ótimas são aquelas que maximizam o potencial de recompensas dado as características definidas na definição do problema. Uma política π consiste em um plano de execução, que fornece uma decisão (uma ação) para cada estado s $\in$ S. Uma política π é considerada ótima se, e somente se, as ações definidas para cada estado também são ótimas.

3.5 Processos de Decisão Markovianos Parcialmente Observáveis (POMDP)

Processos de Decisão Markovianos Parcialmente Observáveis (POMDP) permite especificar modelos de problemas em que um agente autônomo deve planejar e executar ações sequenciais em ambientes parcialmente observáveis. Assim como no MDP, a finalidade consiste em obter recompensas ou, em alguns casos, reduzir penalidades. Trata-se de um modelo genérico, que incorpora problemas de decisão sequencial em ambientes de incerteza (Russell e Norvig 2016). POMDPs é uma extensão mais realista dos MDPs que foram estudados na Seção 3.4.

Quando o ambiente é parcialmente observável, os estados do ambiente não são diretamente acessíveis. O agente não conhece os estados verdadeiros, mas recebe informações parciais na forma de observações que podem ser capturados pelos seus sensores.

Formalmente, o modelo é descrito por uma 7-upla S,A,T,R,γ,O,Z onde:

- **S** : é o conjunto finito de todos os estados do ambiente.
- **A**: é o conjunto finito de ações possíveis.
- **T**: é o conjunto de todas as transições da forma t = (s'|s,a), t ∈ T, s ∈ S, s' ∈ S e a ∈ A. Ou seja, é a probabilidade de alcançar o estado s' executando a ação a no estado s.
- **R**: corresponde a uma função com imagem em $\mathbb{R}$ que atribui um valor de recompensa para cada ação executada em cada estado.
- γ: fator de desconto.
- **O**: é o conjunto finito de todas as observações permitidas.
- **Z**: é o conjunto de probabilidade de observação da forma z = (o'| s', a), $z \in$ Z, $o \in$ O, $s' \in$ S e $a \in$ A. De forma textual, representa a probabilidade do agente observar o, após executar a ação a e alcançar o estado s'.

Os estados, ações, transições, recompensas e fator de desconto são análogos aqueles estudados na Seção que tratava sobre os MDPs. As observações correspondem a forma com que o agente percebe o seu ambiente. Determinar em qual estado o ambiente está pode ser problemático, uma vez que as mesmas observações podem ser observadas em estados distintos (Ross et al. 2008).

A Figura 4 contém uma representação intuitiva do modelo POMDP. A porção acinzentada mostra como ocorre a evolução do ambiente em função do tempo. Por sua vez, o agente somente tem conhecimento da porção mais clara. Este agente mantém um vetor de probabilidade conforme a sua crença sobre em qual estado real o ambiente está. Esta

crença é atualizada conforme o agente executa as suas ações e percebe os estímulos do ambiente na forma de observações.

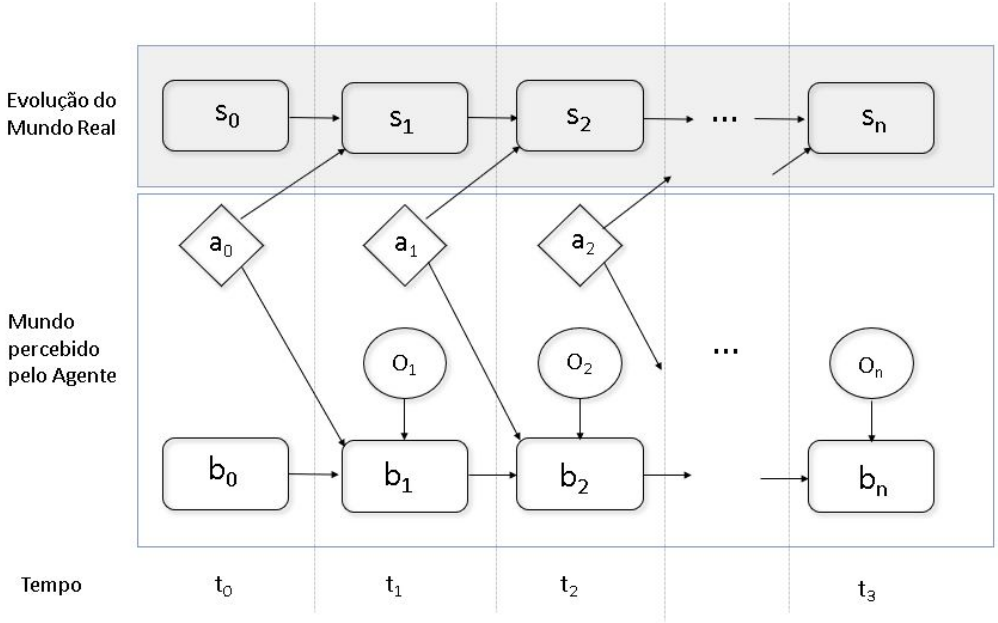

Figura 4 – Representação Gráfica de um POMDP

Por ser um modelo baseado em percepções, as soluções de um POMDP necessitam calcular o histórico de observações H_o, definido como o registro das ações realizadas e das observações percebidas ao longo do tempo.

$$(a^0, o^1), (a^1, o^2), ..., (a^{t\text{-}1}, o^t) \tag{3}$$

O custo de representar todo histórico de ações e observações pode ser indesejável. Uma representação mais usual considera uma distribuição de probabilidade sobre todo o espaço de estados S, o que é chamado de estado de crença. Em um dado tempo t qualquer, um estado de crença $b_t(s)$ é definido a partir do histórico das ações e observações, e também do estado de crença inicial, conforme a equação 4.

$$b_t(s) = P(s_t = s|h_t, b_0) \tag{4}$$

Em cada instante do tempo t, o agente atualiza a ação em um tempo prévio *t-1* e a observação z no tempo corrente t para atualizar a sua crença sobre o ambiente. A responsabilidade por manter atualizada a crença sobre o ambiente é chamada de *função de atualização de estado de crença*(Γ) e é definida segundo a Equação 5:

$$b_t(s') = \Gamma(b_{t\text{-}1}, a_{t\text{-}1}, o_t) = \frac{Z(s', a_{t\text{-}1}, o_t)}{Pr(o_t|b_{t\text{-}1}, a_{t\text{-}1})} = \sum_{s \in S} T(s, a_{t\text{-}1}, s')b_{t\text{-}1}(s) \tag{5}$$

onde $Pr(o|b,a)$ é a probabilidade do agente realizar a observação o depois de executar a ação a no estado de crença b, e pode ser definido como:

$$Pr(o|b,a) = \sum_{s' \in S} Z(s',a,o) \sum_{s \in S} T(s,a,s')b(s) \tag{6}$$

Agora que os estados de crença são estimados pelo agente, é possível criar uma política (π) que defina uma ação para um dado estado de crença fornecido. A solução de um POMDP consiste em encontrar esta política de forma a maximizar a soma total das recompensas imediatas. Mais formalmente, uma política ótima π^* pode ser definida pela seguinte Equação 7.

$$\pi^* = argmax_{\pi \in \Pi} E[\sum_{t=0}^{\infty} \gamma^t \sum_{s \in S} b_t(s) \sum_{a \in A} R(s,a)\pi(b_t,a)|b_0] \tag{7}$$

onde $\pi(b_t,a)$ é a probabilidade da ação a ser executada em um estado de crença b_t.

O retorno esperando seguindo essa política, pode ser definido pela função valor $V^{\pi}(b)$, conforme a Equação 8.

$$V^{\pi}(b) = \sum_{a \in A} \pi(b,a)[R_B(b,a) + \gamma \sum_{z \in Z} Pr(o|b,a)V^{\pi}(b)()] \tag{8}$$

3.5.1 Algoritmos Aproximados

Soluções baseadas em Algoritmos Exatos (que consideram a função de utilidade real de cada estado) inevitavelmente resultam em um problema conhecido como "maldição de dimensionalidade". A maldição da dimensionalidade é um termo introduzido por Bellman para descrever o problema causado pelo crescimento exponencial associado à adição de dimensões extras ao espaço euclidiano (Bellman 2013). No contexto de soluções de problemas do tipo POMDP, a maldição de dimensionalidade incorre do crescimento exponencial do espaço de crença, na mesma medida em que incrementalmente aumenta-se o número de estados, tornando as soluções impraticáveis mesmo quando o espaço de estados é relativamente pequeno. Consideremos, por exemplo, o problema de navegação de um robô modelado em um grid 10x10. Neste cenário, o espaço de crença resultante é de 100 dimensões (22).

Uma possível solução para contornar esta dificuldade é utilizar soluções de algoritmo POMDP baseadas em pontos. Este tipo de solução tem feito um progresso significativo no cálculo de boas soluções envolvendo espaços de estados maiores, como demonstrado em (Smith e Simmons 2012), (49) e (17).

A ideia-chave dessa solução é amostrar apenas um conjunto de pontos dentro do espaço de crença, ao invés do espaço inteiro (geralmente, muito maior), dessa forma reduzindo o espaço de armazenamento em memória. A maioria das soluções mais recentes amostram apenas R(b_0), o subconjunto de pontos de crenças alcançáveis a partir de um ponto inicial

b_0, seguindo uma sequência arbitrária de ações. O objetivo de uma solução POMDP é equivalente a encontrar $R^*(b_0)$, que representa o subconjunto de pontos alcançáveis apartir de $_0$ obedecendo uma sequência ótima de ações.

A Figura 5 ilustra a idéia da amostragem de pontos utilizadas por algoritmos aproximados. Na figura, B é o espaço de crenças, $R(b_0)$ é o espaço alcançável e $R^*(b_0)$ corresponde ao espaço ótimo alcançável. É importante observar que $R^*(b_0) \subseteq R(b_0) \subseteq B$.

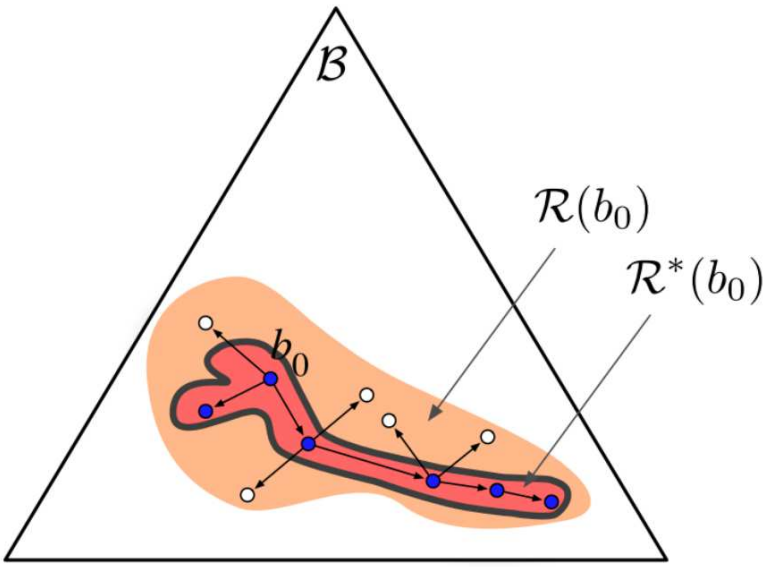

Figura 5 – Soluções de amostragem por pontos (22).

As próximas seções apresentam algoritmos aproximados para resolver POMDP. Os algoritmos que descrevemos aqui são aqueles que foram utilizados em nosso portfólio de planejadores no capítulo 5. Na seção 3.5.2 é descrito o algoritmo PBVI. Na seção 3.5.5 é apresentado o algoritmo Despot. Já a seção 3.5.4 descreve o algoritmo POMCP. Por fim, o algoritmo Sarsop é explicado na seção 3.5.3.

3.5.2 PBVI

O algoritmo PBVI foi o primeiro algoritmo baseado em pontos que obteve uma boa performance em um POMDP de larga escala (22). Ele recebe como parâmetros o estado de crença inicial B_{Init} , o conjunto inicial de soluções Γ_0, o número desejado de expansões N e o horizonte de planejamento T. O parâmetro Γ_0 é usando quando se deseja continuar o planejamento partindo-se de um conjunto soluções anterior e normalmente é iniciado utilizando-se um valor bem baixo, para que seja superado logo. O algoritmo acaba quando o número de expansões N é completado. Também é possível utilizar um critério de parada no algoritmo, quantidade limite de estados de crença, tempo ou quando um erro mínimo aceitável é alcançado.

O algoritmo principal do PBVI é bem simples. Durante N iterações que serão responsáveis pelas expansões do conjunto de estados de crença, ele realiza o cálculo das utilidades e geração da política ótima para um horizonte T.

A rotina de *backup* é normalmente encontrada em diversos algoritmos para resolução de POMDPs que calculam uma solução aproximada e serve para auxiliar a função utilidade, tornando seu valor mais exato a cada iteração T (Pineau et al. 2003). Ela é responsável pelo calculo da função utilidade, para um conjunto de crenças B e um conjunto de soluções Γ atual.

Durante a rotina *backup*, calcula-se todos os valores de α para cada uma das ações a $\in$ A e observações o $\in$ O. Em seguida, seleciona-se a melhor ação a para o estado de crença b, maximizando a função utilidade. Ao final, retorna o conjunto Γ t contendo apenas os vetores α das ações que maximizam os estados de crença.

A rotina *expand* seleciona os novos estados de crença a serem adicionados no conjunto finito de estados de crença, para um conjunto de crenças B e um conjunto de soluções Γ atual.

3.5.3 Sarsop

SARSOP(*Successive Approximations of the Reachable Space under Optimal Policies*) (22) é um eficiente algoritmo de planejamento offline baseado em pontos. O algoritmo utiliza árvores como estrutura de dados para representar o conjunto de pontos de crença, sendo a raiz da árvore a crença inicial. Orientado por uma heurística, o algoritmo realiza amostragem de novas crenças escolhendo uma ação a $\in$ A e uma observação o $\in$ O. O principal ponto do algoritmo é de focar em nós promissores em termos de recompensas. São usados os limites superiores $\bar{V}$ e inferiores $\underline{V}$ na função de valor para orientar o processo de aprendizagem. Para amostrar novos pontos de crença,o SARSOP define um tamanho de intervalo entre os limites superiores e inferiores no nó raiz e caminha para baixo na árvore. Em cada nó, uma ação é escolhida com o limite superior mais alto enquanto a observação escolhida é aquela que faz a maior contribuição para o espaço no nó da raiz da árvore. O algoritmo termina quando o intervalo dos limites inferiores é menor que um dado limiar (Branquinho 2017).

3.5.4 POMCP

POMCP (Silver e Veness 2010), abreviação para *Partially Observable Monte-Carlo Planning*, é um algoritmo de planejamento online que constrói dinamicamente uma árvore de busca, onde cada nó armazena um valor correspondente a uma estimativa.

O agente usa um simulador G como um modelo generativo sobre a árvore de busca para avaliar cada nó da árvore seguindo uma política de busca *"melhor-primeiro"*. Tomando como entrada o estado e uma ação, o modelo generativo fornece uma amostra do estado

sucessor, observação e recompensa, (s',o', r') $\sim$ G(s,a). O simulador é usado para gerar sequências de estados, observações e recompensas, e também para atualizar a função de valor para o estado corrente. A função de valor é estimada pela média dos retornos de N execuções a partir do estado de crença b, sendo V(b,a) = $\frac{1}{N} sum_{i=1}^{N} R^i$, onde R^i é o valor retornado na ith simulação. Todas as ações a $\in$ A são avaliadas e aquela ação que contém o maior valor V(b,a) é selecionada. Na prática, centenas de milhares de simulações podem ser executadas em menos de um segundo, permitindo que sejam construídas árvores de buscas extensas (Silver e Veness 2010).

Em problemas de decisão complexo, a expansão de busca de estados de solução deve considerar o equilíbrio entre diversificação e intensificação. A intensificação, é uma estratégia conservadora que consiste em expandir uma região limitada (mas promissora) do espaço de busca conhecido S com objetivo de encontrar melhores soluções. Trata-se, portanto, de uma busca local na vizinhança do estado S. A diversificação, por sua vez, consiste em sondar uma porção maior do espaço de busca, além da região do espaço S, com a esperança de encontrar outras soluções mais promissoras, evitando dessa forma um ponto de máximo local.É, portanto, uma estratégia de busca global. Para balancear a escolha entre diversificação e intensificação, o POMCP faz uso do algoritmo UCT (Kocsis e Szepesvári 2006).

O POMCP é ideal para problemas que são muito largo ou muito complexos, e uma definição explícita das distribuições de probabilidades são impraticáveis. O diferencial desta solução é que ele supera as maldições de dimensionalidade e de história, amostrando transições de estados, ao invés de considerar todas as possíveis transições de estados.

3.5.5 Despot

Despot (*Determinized Sparse Partially Observable Trees*) é um algoritmo de planejamento online, que assim como o algoritmo POMCP, busca solucionar a maldição de dimensionalidade e a maldição de história. O diferencial é que, por usar o algoritmo UCT, o algoritmo POMCP algumas vezes age de forma excessivamente gulosa, resultando em uma complexidade indesejável no pior cenário (Somani et al. 2013).

O DESPOT é uma árvore, semelhante a árvore de crença padrão, contendo todos os ramos de ações, mas somente ramos de observações dos cenários amostrados. Ela é construída pela aplicação de uma função chamada *modelo simulativo determinista* g: S X A X R $\rightarrow$ S X Z, para todas as ações possíveis em K cenários a partir de um estado de crença inicial b_0.

Enquanto que uma árvore de crença padrão captura a execução de todas as políticas em todos os cenários, o DESPOT também executa todas as políticas mas amostra apenas alguns cenários. Um cenário é uma simulação de uma trajetória iniciando-se de um estado inicial s_0. Para uma crença b, o cenário consiste em uma sequência $\phi = (s_0, \phi_1, \phi_2...)$,

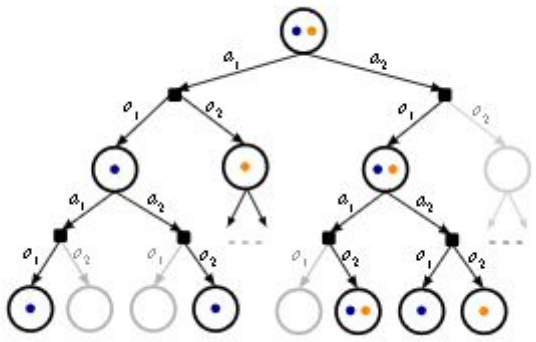

Figura 6 – Representação de uma árvore de crença e sua correspondente DESPOT. (Somani et al. 2013)

onde s_0 é amostrado de acordo com b e cada ϕ_i representa um valor real no intervalo [0,1], calculado de forma independente e uniforme.

Quando é realizada uma simulação sequencial de ações em um cenário, o resultado da aplicação do modelo simulativo determinista é uma trajetória (s_0,a_1,s_1,z_1,a_2,s_2,z_2,...), no qual O caminho (a_1,z_1,a_2,z_2,...) obtido dessa forma é inserido na DESPOT e cada nó da DESPOT armazena os cenários que lhe são alcançáveis.

A Figura 6 exibe uma árvore de crença (em cinza) de profundidade D=2 e a representação de e a sua correspondente DESPOT (em preto), obtida com 2 cenários (pontos coloridos). Cada um dos nós desta árvore representa um estado de crença.

A árvore DESPOT é uma versão simplificada da árvore de crença, com algumas ramificações de observações removidas. Enquanto que uma árvore de crença tradicional de altura D tem O($|A|^D$,$|Z|^D$) nós, a árvore DESPOT tem apenas O($|A|^D$,K) nós (Somani et al. 2013). Por manter uma estrutura de dados mais simples em memória, o DESPOT é uma alternativa eficaz para lidar com as indesejáveis sobrecargas de espaço encontradas em POMDPs com largo conjunto de estados.

Capítulo 4

Especificação de um Módulo Pedagógico

Neste capítulo será descrito um modelo pedagógico de um sistema de apoio ao ensino automatizado que tem por referência as técnicas de *Knowledge Tracing*, com o objetivo de manter atualizadas as estimativas das capacidades cognitivas do estudante. A especificação aqui descrita encontra-se disponível no Apêndice A.

4.1 O Modelo AIT

Um desafio encontrado no desenvolvimento de sistemas educacionais adaptativos é prover as instruções de ensino adequadas ao conhecimento do estudante, considerando o conjunto de pré-requisitos necessários para a sua compreensão. Através do formalismo da linguagem RDDL é especificado um domínio que será chamado de Autonomous Intelligent Tutor (AIT), onde um agente autônomo deve elaborar uma estratégia de ensino apropriada, fornecendo sequencialmente um conjunto de ações pedagógicas condizente com o nível de conhecimento do estudante.

Na parte inicial da especificação é declarado o nome do domínio, os seus requisitos e os tipos de objetos que serão manipulados e que possuem a capacidade de alterar os estados. Como requisito, o domínio é caracterizado como sendo *reward-deterministic* (o planejamento deverá considerar o acúmulo de recompensas e que estas, por sua vez, não são estocásticas), *partially-observed* (informando que o ambiente não é completamente observável pelo agente) e *continuos* (por fazer uso de variáveis parametrizadas fluentes com valores reais). Como objeto, a especificação lida com *skills*, que representam conceitos dentro de um domínio de conhecimento mais amplo.

```
domain AutonomousIntelligentTutor {

requirements = {
reward-deterministic,
partially-observed,
continous
};

types  {
skill : object;
};
```

As subseções seguintes serão apresentadas de acordo com os elementos do modelo POMDP.

4.1.1 Os estados

Os estados são caracterizados pelas configurações das variáveis de estado fluentes que determinam os objetos *skills*. Portanto, para cada configuração em particular das instâncias dos objetos temos um estado distinto e o conjunto de todas essas configurações determinam o espaço de estado. As variáveis definidas para a especificação do AIT serão explicadas a seguir.

Assim como a maioria dos STIs e outros sistemas de apoio ao ensino, as interações do estudante com o sistema ocorrem em turnos intercalados. Para cada avaliação que o tutor realiza, o estudante responde fornecendo uma ação-resposta, que por sua vez é observada pelo módulo pedagógico do sistema para realizar a atualização do modelo de estudante e recomendar a próxima ação instrucional. No modelo do AIT, os conceitos são avaliados um de cada vez, não sendo possível a avaliação de muitos conceitos simultaneamente. A variável responsável por realizar o controle do conceito que está sendo avaliado pela ação do tutor é a variável *updateTurn*, que é habilitada quando o tutor está aguardando a ação-resposta do estudante após a execução de uma ação instrucional que avalia um conceito específico e é desabilitada no turno seguinte. KronDelta corresponde a função matemática *Kronecker delta*, cuja entrada são dois valores inteiros i e j. O seu valor de saída é 1, se os elementos são iguais, ou 0 se i e j são diferentes (Equação 9).

$$KroneckerDelta = \begin{cases} 1, & \text{se i = j} \\ 0, & \text{caso contrário} \end{cases} \tag{9}$$

No RDDL, a função KronDelta recebe como entrada cláusulas booleanas e retorna um valor booleano correspondente a interpretação destas cláusulas. Ela é utilizada quando as transições são determinísticas.

```
updateTurn'(?s) =
       KronDelta( [forall_{?s2: skill} ~updateTurn(?s2)] ^ (
          evaluate(?s)));
```

Listagem 4.1 – Váriavel updateTurn

Como resultado de uma ação pedagógica, o estudante interage com o sistema retornando uma resposta válida. O agente interpreta se a resposta está correta, e atualiza a variável *answeredRight* considerando três condições: a primeira condição analisa qual conceito está sendo avaliado. Quando a variável *updateTurn* está desabilitada para um conceito, ele não está sob avaliação e consequentemente a variável *answeredRight* também não será habilitada para este conceito. Na segunda condição, o agente utiliza a distribuição de Bernoulli com probabilidade de sucesso p = GUESS(?s) , onde GUESS(?s) é uma variável não-fluente e representa a probabilidade do estudante acertar o conceito s quando ele não tem fluência nos pré-requisitos necessários para a sua compreensão. Quando nenhuma das condições anteriores é validada, o agente calcula a probabilidade do estudante acertar uma questão através da equação 10, que é baseada no modelo de rastreabilidade de conhecimento definida por (Corbett e Anderson 1994).

$$p(C_{\text{is}}) = p(L_{\text{rs}}) * (1 - p(S_r)) + (1 - p(L_{\text{rs}})) * p(Gr) \tag{10}$$

Em outras palavras, $p(C_i s)$ representa a probabilidade do estudante s executar uma ação correta para um objetivo i, e é definido pela soma de dois produtos: (i) a probabilidade do estudante s conhecer o conceito abordado ($p(L_r s)$) e não errar a resposta por distração (S_r) e (ii) a probabilidade do estudante não compreender o conceito ($1-p(L_r s)$) e adivinhar a resposta correta ($p(Gr)$).

O trecho de código que determina as condições para atualização da variável answeredRight é exibido na Listagem 4.2.

```
answeredRight'(?s) =

if(~updateTurn(?s))
       then KronDelta (false)

else if (forall_{?s2: skill} [PRE_REQ(?s2, ?s) ^ ~
   proficiencyFluent(?s2)])
       then Bernoulli(GUESS(?s))

else Bernoulli(knowledgeProb(?s) * (1 - SLIP(?s))
       + (1 - knowledgeProb(?s)) * GUESS(?s));
```

Listagem 4.2 – Váriavel answeredRight

Para realizar uma tutoria adaptativa, o agente deve prover a capacidade de gerenciar o estado de conhecimento do estudante e atualiza-lo sempre que perceber que por con-

sequência de uma ação resultar na aquisição de novos conhecimentos. Por se tratar de um ambiente parcialmente observável, o agente não tem a capacidade de saber precisamente o grau de conhecimento do estudante. Ao invés disso, ele mantém um vetor de probabilidade, que no modelo POMDP é chamado de estado de crenças. Com base no estado de crença inferido, o sistema provê a capacidade de fornecer instruções pedagógicas apropriadas ao conhecimento que o estudante possui sobre o domínio.

No AIT é utilizado como referência a Equação 11, que no *Knowledge tracing* atualiza a estimativa do estado de conhecimento do modelo do estudante.

$$p(L_n) = p(L_{\text{n-1}}|evidencia) + (1 - p(L_{\text{n-1}}|evidencia)) * p(T) \tag{11}$$

A equação pode ser interpretada como: a probabilidade de que um conceito tenha sido compreendido em um dado tempo aleatório n é o somatório de duas parcelas: (1) a probabilidade de que o conceito já tenha sido compreendido em um tempo imediatamente anterior n-1 e (2) a probabilidade de que o conceito ainda não tenha sido aprendido multiplicado por um fator p(T) $\in$ [0,1], que representa a probabilidade da obtenção de um conhecimento novo.

Na especificação do AIT, a variável *knowledgeProb* suporta a capacidade de inferir o estado de conhecimento do estudante . A listagem 4.3 apresenta a definição das regras para atualização dos valores da variável. Nela observa-se as seguintes condições para a atualização dos seus valores:

1. A condição inicial verifica qual é o conceito que está sendo avaliado.Assim como para a variável , a atualização dos valores é somente para o conceito que está sendo avaliado

2. A condição seguinte baseia-se na equação 11 para atualizar o valor de *knowledgeProb* quando a ação-resposta fornecida pelo estudante está correta. A variável não-fluente TRANSICTION_PROB é equivale a p(T), porém com uma diferenciação: no modelo de Corbett cada conceito ou está no estado *Learned* (aprendido) ou está no estado *Unlearned* (não-aprendido) . Assim, p(T) representa a probabilidade de transição do estado *Unlearned* para o estado *Learned*. No modelo do AIT, porém, *knowledgeProb* pertence ao domínio probabilístico. É portanto, um valor contínuo e neste caso TRANSICTION_PROB não é meramente uma probabilidade de transição, mas assim um fator de aprendizagem.

3. A terceira condição designa um valor para a variável *knowledgeProb* quando o estudante fornece uma resposta incorreta para uma questão. Novamente há uma distinção entre o modelo proposto por Corbett e o modelo do AIT: enquanto que no primeiro não existe transições Unlearned-Learned, o segundo possibilita a redução sobre a inferência de compreensão com a mesmo fator de aprendizagem TRANSICTION_PROB usado na aquisição de conhecimento.

```
knowledgeProb'(?s) =

if (~updateTurn(?s))
        then knowledgeProb'(?s)

else if(skillTaughtDelayVar(?s))
        then PROFICIENCY_AFTER_TEACH(?s)

else if (answeredRightObs(?s))
        then knowledgeProb(?s) + (1 - knowledgeProb(?s)) *
           TRANSICTION_PROB(?s)

else (1 - knowledgeProb(?s)) +  (knowledgeProb(?s) *
   TRANSICTION_PROB(?s));
```

Listagem 4.3 – Váriavel knowledgeProb

As habilidadades são divididas em níveis de proficiência, podendo ser categorizadas em *proficiencyFluent*, *proficiencyHigh*, *proficiencyMed* e *proficiencyLow*, que correspondem aos níveis de proficiência fluente, alta, média e baixa, respectivamente. Um nível de proficiência é alcançado quando *knowledgeProb* atinge os valores que são configurados pelas variáveis não-fluentes LEVEL_FLUENT, LEVEL_HIGH, LEVEL_MED e LEVEL_LOW. Estas variáveis não-fluentes são customizáveis, podendo ser adaptadas a classificação de ensino em que o sistema será utilizado. Por exemplo, nos Estados Unidos, o sistema de notas utilizado com maior frequência no *High School* segue a classificação apresentada na Figura 7 (National Center for Educational Statistics. U.S. Department of Education 2012). Neste contexto, o AIT poderia ser utilizado fazendo uma correspondência entre o sistema de notas (A-F) e os níveis de proficiência.

Letter Grade	Percentage	GPA
A	90%-100%	4.0
B	80%-89%	3.0
C	70%-79%	2.0
D	60%-69%	1.0
F	0%-59%	0.0

Figura 7 – Sistema de notas comumente adotado no *High school*

O código que condiciona a habilitação da variável *proficiencyFluent* é apresentado na listagem 4.4. Para que tal proficiência seja atingida, o conhecimento inferido na habilidade s deve ser igual ou superior a variável não fluente LEVEL_FLUENT. O código dos demais

níveis de proficiência são análogos a este.

```
proficiencyFluent'(?s) =

if (knowledgeProb(?s) >= LEVEL_FLUENT(?s))
        then KronDelta(true)

else KronDelta(false);
```

Listagem 4.4 – Váriavel proficiencyFluent

4.1.2 As ações

O processo de inferência de conhecimento ocorre durante a resolução das tarefas. Isto porque é nesse momento que o aluno demonstra o seu conhecimento ao sistema, sendo os dados da resolução das tarefas as únicas evidências que o sistema obtém (Vanlehn 2006).

Em uma sessão de tutoria, as ações de um modelo de processo markovianos são representadas por um conjunto de ações pedagógicas implementadas pelo sistema tutor. A ação *evaluate(skill)* avalia a capacidade do estudante em solucionar um problema e representa genericamente ações pedagógicas como dar dicas, perguntar sobre um conceito, etc. Embora os diferentes tipos de ações sejam tratados uniformemente na especificação, é importante salientar que já foram realizados estudos comparando a evolução do aprendizado quando ações distintas foram aplicadas. Em (Rafferty et al. 2011) por exemplo, foi realizado um experimento onde os participantes deveriam descobrir o mapeamento de letras para números, onde ambos (letras e números) eram apresentados na forma de um sistema de equações. Na experiência, um sistema tutor fornecia aleatoriamente três ações diferentes: exemplos, *quizzes* e perguntas com *feedback*. Como resultado do experimento, verificou-se que o tempo médio para que os participantes descobrissem o mapeamento quando perguntas com feedback eram realizadas foi de 12s, ao passo que exemplos e *quizz* tiveram média de 7s e 6.6s, respectivamente.

4.1.3 As observações

Dado que o conhecimento é um elemento abstrato, o agente não possui a capacidade de mensurar precisamente o nível de conhecimento que o estudante possui. Em um modelo POMDP, o agente mantém um vetor de probabilidades sobre cada estado físico, chamado de estado de crença. Após realizar uma transição de estado de s para s', o agente tem uma probabilidade p(o|s',a) de realizar a observação o.

No modelo criamos uma única variável *knowledgeProbObs* que simboliza a estimativa sobre o conhecimento em um dado conceito. Na listagem 4.5 é apresentada a definição da variável. Ela utiliza a função *DiracDelta*, inspirada na função matemática *Delta de Dirac*, e que na linguagem RDDL exerce o mesmo papel da função *KronDelta* porém aplicada

ao domínio de valores reais.

```
knowledgeProbObs(?s) =
        DiracDelta( knowledgeProb'(?s) );
```

Listagem 4.5 – Váriavel knowledgeProbObs

4.1.4 As transições de estados

O sistema quando executa uma ação instrucional pode provoca uma transição de estado, concebido como a aquisição de um novo conhecimento ou pela percepção que o estudante não tem o domínio sobre o conceito que acreditava-se que ele deveria ter. No modelo POMDP o efeito das ações é estocástico, o que significa dizer que a aplicação de uma mesma ação sobre um dado estado pode resultar em transições para estados distintos. Assim, os efeitos de uma ação instrucional podem ser representados como uma distribuição de probabilidade sobre os estados sucessores. Nos casos onde o tempo é discreto e os estados e as ações são ambos discretos e finitos, um POMDP é tipicamente representado por tabelas de probabilidade condicional (CPT, do inglês *conditional probability tables*) especificando probabilidades de transição e recompensas esperadas para estados e ações (Sallans 2000). Na especificação RDDL, os valores numéricos que representam as probabilidade de transições de estados não são declarados explicitamente, e por consequência é responsabilidade do planejador extrair estas probabilidades com base nas condições que habilitam ou desabilitam as variáveis de estados fluentes.

4.1.5 A recompensa

Processos de decisão Markovianos tem como meta a obtenção de recompensas (ou redução de custo) que são resultantes de

Para o modelo do AIT definimos recompensa como uma função da soma de cada habilidade s que o estudante atingiu um nível de proficiência multiplicado pela recompensa em atingir tal proficiência e o peso w(s) atribuído a aquisição de s. A equação 12 descreve matematicamente a função de recompensa.

$$R_{\text{total}} = \sum_{nivelProf} \sum_{s \in S} w(s) * nivelProf(s) * reward_{\text{nivelProf}}(s) \tag{12}$$

Na especificação RDDL a recompensa é descrita na seção *reward*. A listagem 4.6 descreve a função de recompensa da equação 12 para o AIT.

```
reward =  [sum_{?s : skill} [SKILL_WEIGHT(?s) * proficiencyLow(?s
   )
* ~proficiencyMed(?s) * ~proficiencyHigh(?s) * ~proficiencyFluent
   (?s)
* REWARD_PROFICIENCY_LOW(?s)]] +

[sum_{?s : skill} [SKILL_WEIGHT(?s) * proficiencyMed(?s) * ~
   proficiencyHigh(?s)
        * ~proficiencyFluent(?s) * REWARD_PROFICIENCY_MED(?s)]] +

[sum_{?s : skill} [SKILL_WEIGHT(?s)
* proficiencyHigh(?s) * ~proficiencyFluent(?s)
        * REWARD_PROFICIENCY_HIGH(?s)]] +

[sum_{?s : skill} [SKILL_WEIGHT(?s)
* proficiencyFluent(?s) * REWARD_PROFICIENCY_FLUENT(?s)]];
```

Listagem 4.6 – Equação de recompensa

Esta equação de recompensa é flexível por permitir que recompensas sejam atribuídas somente para os níveis de proficiência desejados.Por exemplo, se o intuito fosse atribuir recompensa apenas às habilidades em que o estudante seja fluente, bastaria configurar REWARD_PROFICIENCY_LOW = REWARD_PROFICIENCY_MED = REWARD_PROFICIENCY_HIGH = 0.

4.1.6 Aplicação do modelo em um ambiente simulado

A partir da especificação do modelo descrito, foi desenvolvido um protótipo de um sistema educacional que utiliza os recursos desta especificação para nortear a suas decisões. O domínio de enconhecimento escolhido foi a disciplina de Cálculo Diferencial, mais especificamente os conceitos relacionados ao ensino de derivadas. Considera-se que para que os objetivos sejam alcançados em sua completude, os conceitos "Fundamentos de derivada","Regra da Cadeia","Derivadas de Funções Trigonométricas","Derivadas de Funções Exponenciais e Logarítmicas","Derivada das Funções Hiperbólicas"e "Derivada de Alta Ordem"deverão ser lecionados obedecendo a cadeia de pré-requisitos, ordenados aqui da esquerda para a direita conforme a sua citação. Os atributos não-fluentes foram mantidos com os seus respectivos valores padrão.

O modelo do estudante é uma representação lógica que compreende todas as estimativas a cerca do estado de conhecimento deste estudante em cada um dos conceitos citados. Embora existam técnicas em computação para se definir o ponto de partida do estudo (como em (Zhang 2013)), para fins de exemplificação o nosso estudante hipotético não conhece nenhum dos conceitos abordados por este domínio.

Referente aos aspectos computacionais, o protótipo apresentado foi desenvolvido e executado em um ambiente Linux Mint 18.3 x64 em um computador com processador Intel i3-7100 (2.4 GHz, 3MB L3 Cache) com uma memória de 8GB DDR4. A linguagem Java e o banco de dados MySQL foram utilizados para a integração como uma camada intermediário entre a interface gráfica do usuário e o planejador, e para a persistência do domínio de conhecimento do problema.

No remanescente deste capítulo descrevemos os passos de uma execução real de uma sessão de tutoria que faz uso de um planejador. A cada instante de iteração são demonstrados como são feitas as atualizações dos valores dos atributos que compõe o estado do ambiente.

Na iteração inicial da simulação, a ação escolhida pelo planejador instrucional foi ensinar "Fundamentos de derivada". Na imagem da Figura 8 é possível visualizar no protótipo a execução desta ação.

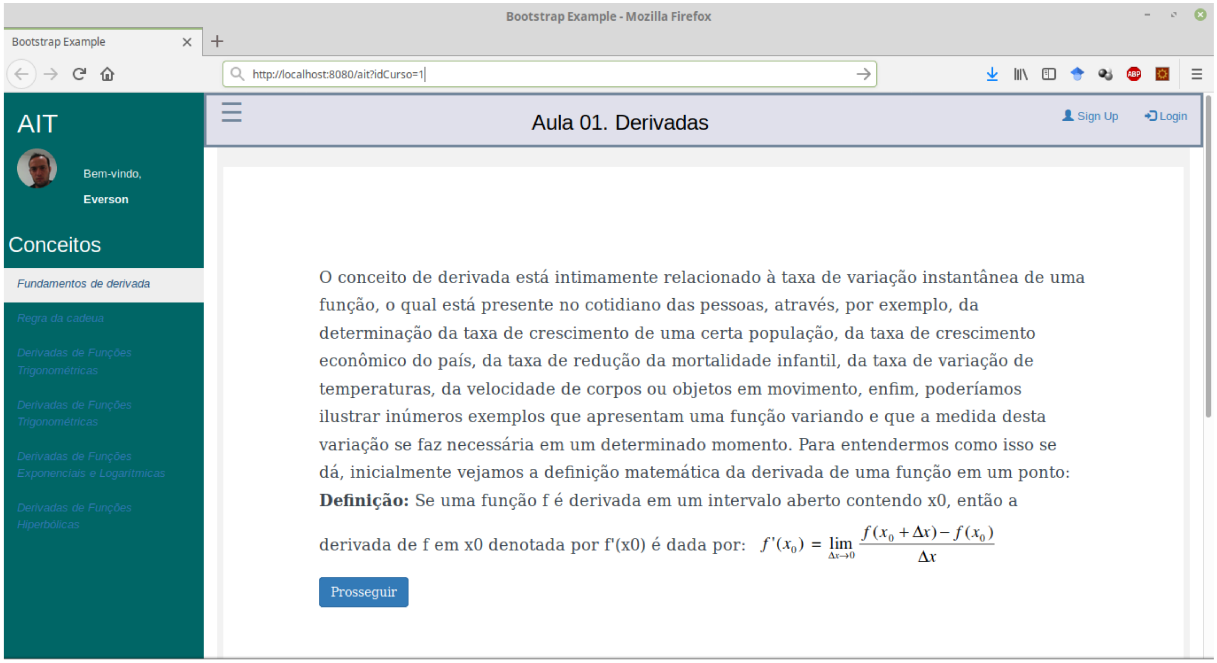

Figura 8 – Apresentação do conteúdo de derivada

Como consequência desta ação, a variável *knowledgeProb* assume para o conceito em questão, o valor definido pela variável não-fluente *knowledgeProb* com o valor padrão de 0.5(50%). Este valor classifica o estudante como tendo uma proficiência baixa (faixa de valores entre 45% e 60%), o que por sua vez induz a um ganho de recompensa imediata igual ao valor do peso do conceito (variável SKILL_WEIGHT = 1) multiplicado pela recompensa em se atingir tal proficiência (variável REWARD_PROFICIENCY_LOW = 1).

Após atualizar a inferência sobre o conhecimento no conceito *"Fundamentos de derivada"*, o agente tutor deve prosseguir com sessão de ensino, fornecedo a próxima ação.

Intuitivamente, neste momento os melhores cenário seriam melhorar a estimativa sobre *"Fundamentos de derivada"* ou explorar um novo conceito. Ambas abordagens cometem em acumulo de recompensas. O objetivo do planejador instrucional é decidir, dentre as alternativa a ação que possibilite melhores resultados a longo prazo. A opção por explorar um novo conceito pode ser considerada prematura, considerando que o conceito *"Fundamentos de derivada"* é pré-requisito para todos os demais conceitos e sua proficiência baixa pode resultar inclusive em uma redução da recompensa já adquirida.

A opção escolhida pelo sistema em sua segunda iteração, foi avaliar o conceito *"Fundamentos de derivada"*. Tal avaliação pode ocorrer de diversas formas. No modelo proposto por (Rafferty et al. 2011), valores diferenciados de recompensas foram atribuídos a tipos distintos de ações (*Quizzes*, *exemplos* e *perguntas com feedback*, de acordo com o tempo médio que levavam para que o estudante dominasse os conceitos. Nesta proposta, no entanto, não realizamos nenhuma distinção. A avaliação do conceito *"Fundamentos de derivada"* é mostrado na Figura 9.

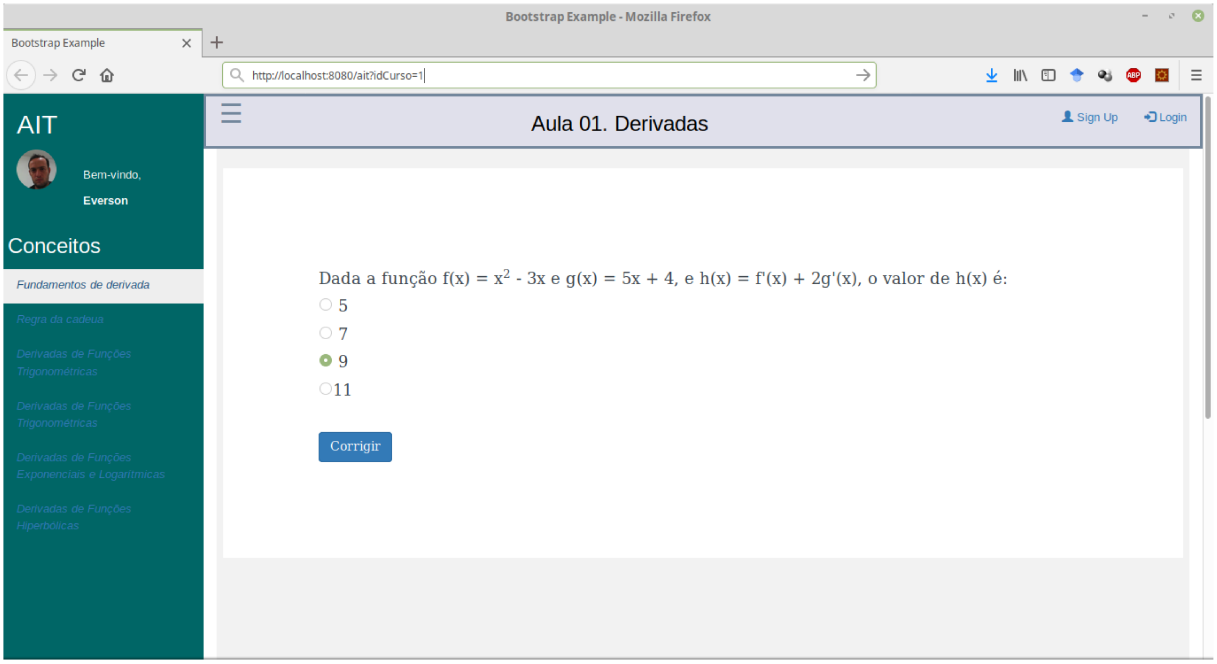

Figura 9 – Apresentação do conteúdo de derivada

O estudante neste momento pode fornecer uma resposta correta ou não, o que provoca uma nova atualização sobre as estimativas de conhecimento acerca do conceito. Uma redução pode fazer com que o sistema requeira uma nova avaliação ou mesmo explique o conceito novamente. Fornecendo uma resposta correta, o planejador opta por ensinar o conceito *Regra da Cadeia*, conforme demonstrado pela Figura 10.

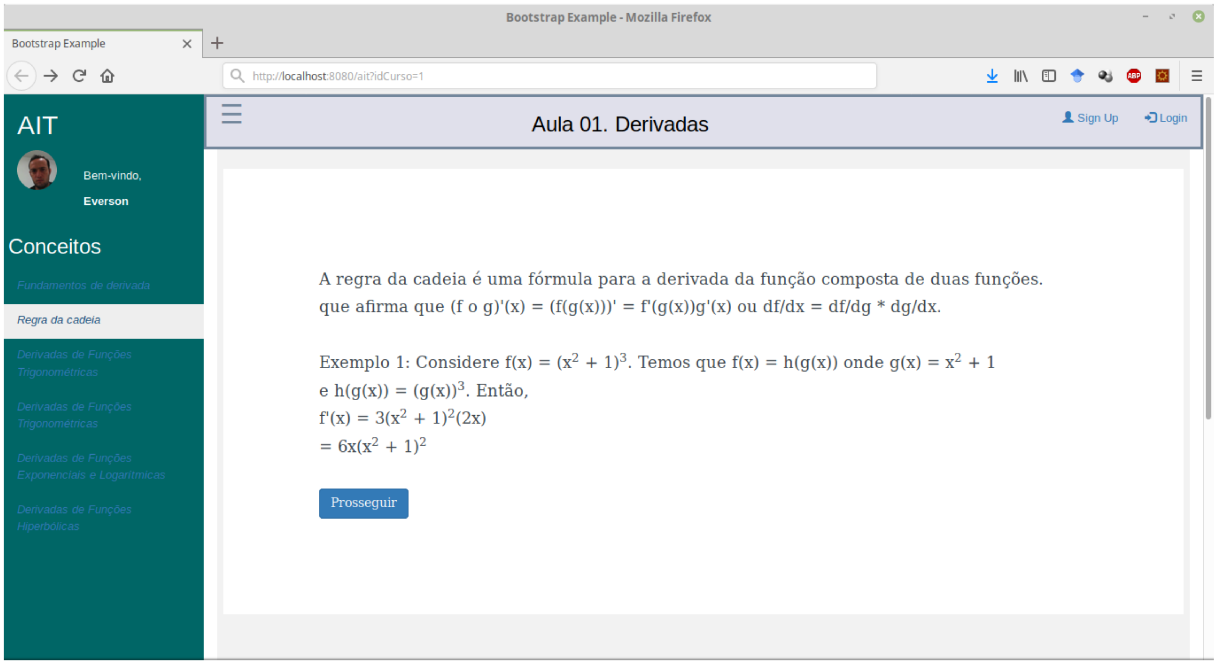

Figura 10 – Apresentação do conteúdo de derivada

O processo segue avaliando ou explicando conceitos até que a sessão termine e a soma das recompensas seja obtida. As escolhas das ações demonstradas neste exemplo podem variar conforme o plano instrucional gerado.

Capítulo 5

Metodologia para seleção de um planejador

Nesta seção são descritos todos os aspectos envolvidos na metodologia responsável pela seleção de um planejador probabilístico, incluindo o portfólio adotado, as características da especificação que são utilizados na classificação, os algoritmos de aprendizagem de máquina e a divisão do portfólio em dados de testes e treinamento.

5.1 Apresentação da Metodologia

Nesta seção é apresentada uma metodologia para a escolha de um planejador autônomo responsável pela escolha das ações pedagógicas, tais como avaliações, dicas ou explicações de conceitos. A escolha apropriada de tais ações é um objetivo primordial de um sistema adaptativo de apoio ao ensino e reflete diretamente no engajamento do estudante.

Embora existam uma grande quantidade de sistemas de planejamento automatizado, cada qual implementando um ou mais algoritmos como os que foram apresentados na seção 3.5.1, uma invariante dentro do planejamento probabilístico é que nenhum destes planejadores é capaz de se sobressair sobre todos os demais para qualquer domínio de problema existente (9). Tal fato é evidenciado pelo histórico das competições de planejamento, onde os vencedores não só se alternam entre edições distintas, mas também para combinações de domínios de problemas dentro da mesma edição. Portanto, faz-se necessário um método de seleção que seja especificamente aplicado a cada problema de forma individual.

A metodologia desenvolvida neste trabalho baseia-se em técnicas de aprendizagem de máquina e utiliza elementos da especificação RDDL (*Relational Dynamic Influence Diagram Language*) (Sanner 2010), uma linguagem de especificação uniforme desenvolvida para resolver problemas de planejamento probabilísticos e que também possibilita a criação de bons modelos de representação do tipo POMDP. A partir dos dados pertencentes a esta especificação, o resultado da execução do procedimento escolhe, dentre

diversos softwares de planejamento autônomo, aquele que possibilite a obtenção de valores de recompensas superiores quando comparado aos seus concorrentes. A decisão por tal planejador é apoiada em um portfólio de algoritmos de planejamento. Embora não haja um consenso na literatura acadêmica sobre a definição de portfólio, adota-se aqui a que a se refere a escolha justificada de um ou mais algoritmos de planejamento, dentro de um conjunto finito, com o intuito de melhorar os resultados esperados.

Apesar de o foco deste trabalho ser direcionado ao contexto educacional, a proposta tem um âmbito genérico e pode ser aplicado a outros domínios do planejamento probabilístico. A sequência dos passos utilizados por esta abordagem é apresentada na Figura 11.

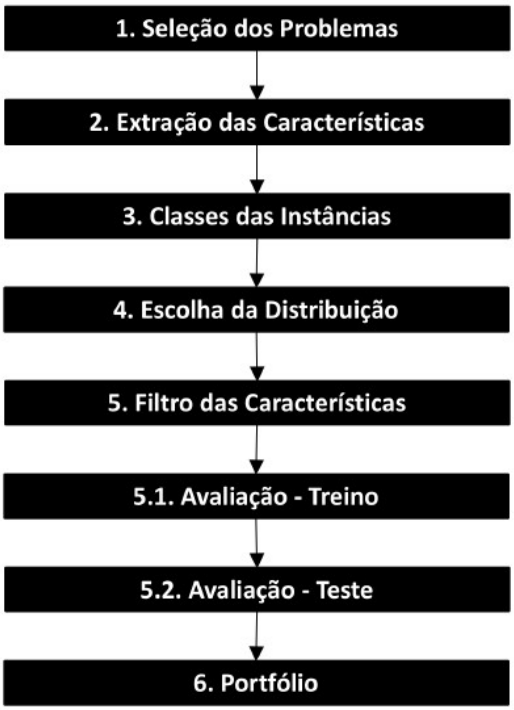

Figura 11 – Etapas da metodologia de seleção

A estratégia utilizada envolve técnicas de aprendizagem de máquina, mais especificamente de aprendizagem supervisionada, onde são fornecidos um conjunto de elementos rotulados com uma classe, e por meio de um ou mais classificadores, são gerados modelos preditivos que possibilitam a classificação de novas instâncias considerando suas características. Portanto, o processo pode ser dividido em duas etapas:

- Treinamento: a etapa inicial fornece um conjunto de elementos e a cada elemento é definido uma classe, ou seja, uma classificação segundo um contexto específico. Nesta etapa ressalta-se a importância dos dados de treinamento, a sua diversificação, os algoritmos de classificação e a escolha dos atributos que irão gerar os modelos preditivos.

- Teste: o teste é a classificação em si, quando novas instâncias deverão ser rotuladas a partir do modelo preditivo gerado na etapa anterior. A validação dos resultados é essencial para avaliar a qualidade do modelo preditivo.

Cada uma destas dificuldades elencadas serão discutidas nas próximas seções deste capítulo.

O objetivo do procedimento é que ao final seja gerado um comitê de classificadores que permitam predizer a escolha de um planejador probabilístico dentro de um portfólio. A idéia de um comitê de classificadores parte da metáfora de um time de especialistas que podem opinar sobre um determinado assunto. Assim, mesmo quando uma das opiniões não é a correta, tende-se a chegar a um consenso de alguma forma e, por consequência, tomar uma decisão.

5.2 Critérios de Avaliação

A proposta deste trabalho baseia-se nos regulamentos e critérios de avaliação utilizados na competição IPPC (*International Probabilistic Planning Competition*), mais especificamente na edição realizada em 2011, em razão da diversidade de domínios de problemas apresentados, que incorporam diversos desafios de planejamento e pela maior expressividade de participantes. Nesta edição, as regras estabeleciam que os competidores (os planejadores probabilísticos inscritos na competição) seriam executados em um conjunto de oito domínios de problemas, cada qual contendo dez instâncias. Cada uma destas instância deveriam ser executada trinta vezes para cada um dos competidores, em um ambiente simulado com um horizonte finito de quarenta iterações e sem penalização de fator de desconto. Como critério de avaliação, foi considerado a média das pontuações normalizadas, calculadas conforme a equação 13.

$$media\_pont\_norm(planejador) = \sum_{inst=1}^{inst=80} \frac{pont\_norm(planejador, inst)}{80} \qquad (13)$$

Na equação 13. *media_pont_norm* simboliza a média das pontuações normalizadas, *inst* representa cada uma das oitenta instâncias e a *pont_norm* é a pontuação normalizada, calculada separadamente para cada instância segundo a equação 14.

$$pont\_norm(planejador, inst) = max(0, \sum_{exec=1}^{exec=80} \frac{pont\_bruta(planejador, inst)/30 - minPont(inst)}{maxPont(inst) - minPont(inst)}) \qquad (14)$$

Na equação 14, *exec* é a variavél que itera sobre cada execução realizada para cada planejador em cada instância, *pont_bruta* é a pontuação bruta que corresponde ao valor da recompensa obtida durante a execução correspondente e minPont e maxPont são respectivamente, a pontuação máxima e a pontuação minima na instância, considerando

os valores de recompensas obtida por todos os planejadores e também por um política de seleção de ações aleatória e outra política estacionária *no-Op*, que nunca seleciona nenhuma ação.

5.3 Composição do Portfólio

Um portfólio deve compreender um conjunto amplo de planejadores, que incorporem soluções diversificadas para solucionar problemas contendo graus de dificuldade variados. Além disso, é necessário verificar se o planejador atende a todas os requisitos estabelecidos pelos domínios de problemas escolhidos. Nesse sentido, escolhemos para compor o portfólio apenas os planejadores que possuem a capacidade de resolver problemas de decisão em ambientes parcialmente observáveis e que possibilitem a interpretação da linguagem de especificação RDDL diretamente, ou indiretamente, quando os desenvolvedores do planejador disponibilizam ferramentas de tradução.

Os planejadores que fazem a composição deste portfólio são:

- **Symbolic Perseus**: Algoritmo de iteração de valor baseado em pontos que faz uso de diagramas de decisão algébricos (ADD) como estrutura de dados para lidar com POMDPs com elevada fatoração (Poupart 2005).

- **POMDPX NUS**: Planejador que integra dois algoritmos de planejamento: SARSOP (22), que é executado para pequenas instâncias e POMCP (Silver e Veness 2010), para instâncias maiores.

- **KAIST AILAB**: Planejador baseado em Heuristic Search Value Iteration simbólico (Symbolic HSVI) (Sim et al. 2008), modificado para lidar com POMDPs de horizonte finito.

- **McGill**: Planejador probabilístico desenvolvido na universidade de McGill que utiliza o algoritmo online POMCP.

- **POND-Hindsight**:A idéia deste planejador é usar planejamento clássico para avaliar futuros determinísticos, e então executar o plano (66) ou avaliar a média da qualidade de diversos futuros alternativos antes de selecionar uma ação (Yoon et al. 2008). Detalhes do algoritmo podem ser encontrados em (Olsen e Bryce).

- **HyPlan**: Assim como o POMDPX NUS, este planejador combina duas soluções, um offline baseada em pontos para cálculos de funções de valores para pequenos espaços de crença, que atua como uma heurística para a solução online, que é baseada em técnicas de amostragem do espaço de estado.

5.4 Domínios de Problemas

No contexto de planejamento existem diversos domínios de problemas que abrangem situações de dificuldades encontradas no mundo real. Muitas destas dificuldades advêm da robôtica, como a observação de obstáculos pelo agente. Outras porém, exploram a capacidade de paralelismo, da sincronização entre os agentes, restrições de estados e a dinâmica do ambiente.

A *IPPC* aborda uma série destas situações através de seus domínios de problemas, e portanto consideraremos este conjunto de problemas disponibilizados em nosso processo de seleção. Em sua edição 2011, os domínios propostos foram:

- *Crossing Traffic*: Um robô é posicionado em um grid, e deve chegar na meta desviando de diversos obstáculos que aparecem aleatoriamente. Se o obstáculo atinge o robô, esse não pode mais se mover. O objetivo é chegar na meta com o menor número de passos.

- *Elevators*: Um número de elevadores é fornecido e deve coletar todos os passageiros e transportá-los para o seu destino. Os locais de coleta de passageiros e destinos possíveis são apenas o primeiro e último andar de um prédio. O objetivo é levar os passageiros aos seus destinos sempre que um novo passageiro surgir.

- *Game of Life*: Neste problema é representado o *jogo da vida*, um autômato celular desenvolvido pelo matemático britânico John Horton Conway na decáda de 70. O problema é apresentado em forma de um grid, e as regras são simples: uma célula morre se tiver menos que duas ou mais que células vivas em seu lado, uma célula sobrevive se tiver duas ou três células vivas ao seu lado, e uma célula morta revive se tiver exatamente três células vivas ao seu lado. Seu objetivo é manter o maior número de células vivas ao final do horizonte.

- *Navigation*: Este problema assemelha-se ao *Crossing raffic*. Um robô disposto num tabuleiro com o objetivo de chegar até a meta. Porém, a cada célula é associada uma probabilidade que faz com que o robô desapareça se pisar nela.

- *Reconnaissance*: É necessário controlar um robô em um grid. Dispostos pelo grid temos a base, células contaminadas e objetos espalhados. O robô possui três ferramentas: uma para detectar água, outra para detectar seres vivos e outra para tirar fotos. Alguns objetos podem estar contaminados, e ao analisar objetos contaminados com uma ferramenta contaminamos a mesma. Para descontaminar a ferramenta, devemos ir até a base. O objetivo é encontrar o maior número de seres vivos e fotografá-los.

- *Skill Teaching*: Este domínio representa um agente que deve ensinar uma série de habilidades a um estudante, optando entre fornecer dicas e avaliar o conhecimento

através de perguntas. Cada estudante tem um nível de proficiência em cada uma das habilidades, e a probabilidade de acertar uma pergunta é condicionada ao grau de proficiência em dado conceito. A recompensa é calculada com base na soma de habilidades em que o estudante tenha adquirido uma proficiência média ou alta multiplicada pelo peso atribuído a tal habilidade.

- *SysAdmin*: O agente neste domínio é o administrador de um sistema que é responsável por manter uma rede de computadores ativa. Esta rede pode assumir diversas topologias e contém diversos computadores. Cada computador pode estar ou não em funcionamento. O objetivo é manter o máximo de computadores ativos na rede ao término da execução do problema.

- *Traffic*: Neste domínio devemos gerenciar um agente que controla o tráfego de veículos em cruzamentos de uma cidade. O objetivo é justamente diminuir o engarrafamento de carros em determinadas vias,desafogando ou desviando o trânsito para vias alternativas.

As pontuações brutas, normalizadas, máxima e mínima, obtida pelos planejadores em cada uma das instâncias dos domínios da competição estavam disponíveis na data da publicação deste trabalho no sitio (ICAPS IPPC-2011 log files 2011) .

5.5 Extração de características

Baseado no pressuposto que estabelece que não existe um planejador que se sobressaia sobre todos os demais para todos os tipos de problemas, é necessário que a seleção de um planejador seja fundamentada em algum critério objetivo. Neste trabalho adotamos a hipótese que as características que especificam instâncias e domínios de planejamento, podem ser aproveitadas na escolha do planejador que potencializa maiores resultados. Tal hipótese já foi adotada anterioremente com sucesso para o planejamento clássico em (Sousa et al. 2014) e (Vrakas et al. 2003), mas não há proposta semelhante para o planejamento probabilístico.

A identificação de características que podem influenciar na escolha do planejador probabilístico considera os elementos presentes no formalismo de especificação RDDL e também características que são comuns ao planejamento clássico e que já foram exploradas em outros trabalhos mencionados. De tal esforço, resultou-se na identificação de 29 características, que são apresentadas na Tabela 1.

As características C01 a C06, C10 a C28 são características dependentes apenas do domínio, e não varia entre as instâncias do mesmo problema, ao passo que as características C07 a C09 e C29 são dependentes da instância.

Tabela 1 – Características identificadas em uma especificação RDDL.

Característica	Nome
C01	Número de variavéis do tipo action-fluent
C02	Número de variavéis do tipo state-fluent
C03	Número de variavéis do tipo observ-fluent
C04	Número de variáveis do tipo interm-fluent
C05	Número de variavéis non-fluent
C06	Quantidade de restrições dos estados
C07	Número de iterações (tamanho do horizonte)
C08	Fator de desconto
C09	Quantidade de ações executadas em paralelo
C10	Quantidade de variavéis state-fluent que influenciam diretamente no cálculo da rec
C11	Número mínimo de parâmetros action-fluent que influenciam em estados sucessore
C12	Númeromáximo de parâmetros action-fluent que influenciam em estados sucessores
C13	Valor médio de parâmetros action-fluent que influenciam em estados sucessores
C14	Número mínimo de estados em t que influenciam os estados em t+1
C15	Númeromáximo de estados em t que influenciam em estados t+1
C16	Número médio de estados em t que influenciam os estados em t+1
C17	Número de estados multivalorados
C18	Quantidade de variavéis state-fluent do tipo booleana
C19	Quantidade de variavéis state-fluent do tipo inteiro
C20	Quantidade de variavéis state-fluent do tipo real
C21	Presença do requisito continous
C22	Presença do requisito multivalued
C23	Presença do requisito reward-deterministic
C24	Presença do requisito intermediate-nodes
C25	Presença do requisito constrained-state
C26	Presença do requisito partially-observed
C27	Presença do requisito concurrent
C28	Presença do requisito cpf-deterministic
C29	Quantidade de combinações de atributos possíveis para um estado

5.6 Divisão dos dados para treinamento

Uma decisão crucial em abordagens baseadas em aprendizagem de máquina, refere-se a divisão adotada das instâncias que comporão a base de treinamento dos modelos.

Quando os dados fornecidos para treinamento não são suficientemente abrangentes, o classificador possui uma baixa capacidade de generalização, isto é, quando ele não consegue boas taxas de classificação com novas instâncias, diz-se que ele sofreu um superajustamento aos dados de treinamento. Esse superajuste é denominado de *overfitting*.

O caso contrário ao *overfitting* é denominado de *underfitting*, o qual ocorre quando o modelo gerado pelo algoritmo não possui uma capacidade de classificação boa nem mesmo para o conjunto de treinamento. Nestes casos, considera-se que o algoritmo ainda não convergiu para uma solução ótima, mesmo que local.

Existem algumas abordagens comuns para realizar esta divisão de modo a se evi-

tar o *overfitting* e *underfitting*, e consequentemente se obter uma divisão de dados de treinamento suficientemente boa para gerar bons modelos. Algumas destas abordagens são explicadas no capítulo que trata sobre os classificadores em (Bouckaert et al. 2013). Optamos por duas estratégias distintas: *Leave One Domain Out* e *Percentage Split*.

A estratégia *Leave One Domain Out* consiste em aleatoriamente escolher um conjunto de domínios para compor os treinamentos e outro conjunto menor a ser utilizado para a avaliação da classificação. Utilizamos seis domínios para treinamento e os dois domínios restantes avaliamos a classificação do algoritmo de aprendizagem de máquina, mantendo dessa forma uma proporção razoável de 75% de dados para treinamento em relação ao conjunto total de instâncias ($\frac{dados-treinamento}{dados-totais}$). Repetimos o procedimento até que todos os oito domínios tenham sido utilizados na etapa de treinamento uma vez. A seleção aleatória dos domínios segue conforme disposto na tabela 2.

Tabela 2 – Divisão dos domínios para a estratégia Leave One Domain Out.

Divisão	Domínios selecionados para etapa de teste
div1	Skill Teaching e Elevators
div2	Navigation e Recconaissance
div3	Game of Live e Traffic
div4	Cross Traffic e SysAdmin

A estratégia alternativa, *Percentage Split*, realiza uma divisão de todo o conjunto de instâncias da base de dados, separando-os em treinamento e teste de acordo com uma percentagem pré-definida. Com o intuito de manter a coerência entre as estratégias, a proporção de 75% de dados para treinamento adota na estratégia *Leave One Domain Out* será utilizada também nesta abordagem.

5.7 Classificadores

A obtenção dos modelos preditivos ocorreu através da ferramenta WEKA (Witten et al. 2016). O WEKA possui uma extensa biblioteca de algoritmos de aprendizagem de máquina com a finalidade de aplicar a classificação de instâncias em classes. Para testar a eficiência dos classificadores, foram selecionados os seguintes algoritmos:

- Naive Bayes.
- Árvore de decisão J48.
- Tabelas de decisão.
- K-Nearest Neighbors (KNN).
- Support Vector Machines (SVM).

- ❑ Redes Neurais do tipo Multilayer Perceptron (MLP).

A razão pela escolha dos algoritmos listados foi pautada na utilização frequente dos mesmos em outros trabalhos de classificação baseados em aprendizagem de máquina e pelo sucesso que demonstraram em seus resultados. Não é objetivo deste trabalho determinar a eficiência de qualquer um destes classificadores nem realizar qualquer tipo de comparação entre eles.

Os algoritmos foram testados com variações de parâmetros de execução, incluindo podas, normalizações e padronizações. No trabalho, será omitido essa etapa, sendo considerado apenas as configurações o qual obtivemos os melhores resultados.

Capítulo 6

Experimentos e Análise dos Resultados

Nesta seção serão descritos os resultadoss da aplicação da metodologia de seleção de planejador definida no capítulo 5. Em sequência, a escolha é validada comparando os resultados obtidos pela execução do planejador escolhido com os demais planejadores do portfólio, quando executados sobre o domínio do problema *AIT* que foi proposto no capítulo 4. Todos os experimentos aqui descritos foram realizados em um computador com processador Intel i3-7100U (2.4 GHz, 3MB L3 Cache) com dois pentes de memória de 4GB DDR4.

O experimento pode ser dividido em duas fases. A primeira fase consiste na formação do comitê de planejadores. Na fase subsequente os resultados dos planejadores pertencentes ao portfólio são analisados.

6.1 Resultados

Os experimentos para a criação do portfólio de planejamento levam em conta a avaliação dos resultados em relação ao percentual de acerto dos algoritmos de classificação para as instâncias dos problemas da IPPC. Para criar a base de dados são consideradas as instâncias dos problemas da competição IPPC e da forma da distribuição *Percentage Split* ou *Leave One Domain Out*.

O passo inicial consiste na obtenção da linha de base. Na ferramenta Weka, a linha de base é calculada aplicando o classificador ZeroR no conjunto de dados. O seu funcionamento é trivial: o classificador apenas pesquisa pela classe mais popular e prevê a mesma classificação para todas as execuções. Aplicado aos domínios de problemas propostos na Seção 5.4, o algoritmo classificou corretamente 41 instâncias do conjunto total de 80 instâncias, o que corresponde a uma taxa de acerto de 51,25%. Os classificadores cuja precisão se situam abaixo da linha de base tem um desempenho bastante negativo, e portanto, são descartados na composição dos pesos na escolha do portfólio.

Na etapa seguinte, foi realizado um refinamento do conjunto de atributos descritos na Seção 5.5.A seleção de um subconjunto de atributos com o objetivo visa diminuir o

número de atributos utilizados na representação dos exemplos. São duas as principais razões para a sua realização. A primeira razão é que a maioria dos algoritmos de Aprendizagem de Máquinas, computacionalmente viáveis, não trabalham bem na presença de vários atributos. A segunda razão é que, um grande número de atributos não necessariamente resulta em melhores classificações. Dessa forma, atributos não-relevantes para a classificação podem prejudicar o resultado final.

A ferramenta WEKA dispões de vários mecanismos para auxiliar na seleção de um subconjunto de atributos. Neste experimento foi utilizados o método *Wrapper*. Este método utiliza validações cruzadas iterativamente para selecionar o melhor atributo para adicionar ou remover da iteração anterior. São realizadas entre 2 a 5 iterações por padrão, e o algoritmo pára quando o desvio-padrão entre as execuções é menor que um dado limiar. O método Wrapper é definido pelo método de pesquisa, que pode ser melhor-primeiro ou avaliação gulosa. Quando a avaliação é melhor-primeiro, a busca deve ser direcionada para frente, para trás ou em ambas as direções. Na aplicação do método Wrapper foi mantida constante as configurações de critério de pesquisa para evitar-se mínimos locais com o valor-padrão igual a 5 e o classificador J48 para a avaliação do atributos. Todas as combinações de métodos e direções de pesquisa foram avaliadas e são apresentadas pelos gráfico das Figuras 12,13, 14 e 15. Nos gráficos, o eixo das abscissas representa cada uma das características e o eixo das ordenadas a quantidade de vezes em que a característica fora selecionada pelo método de busca nas validações cruzada.

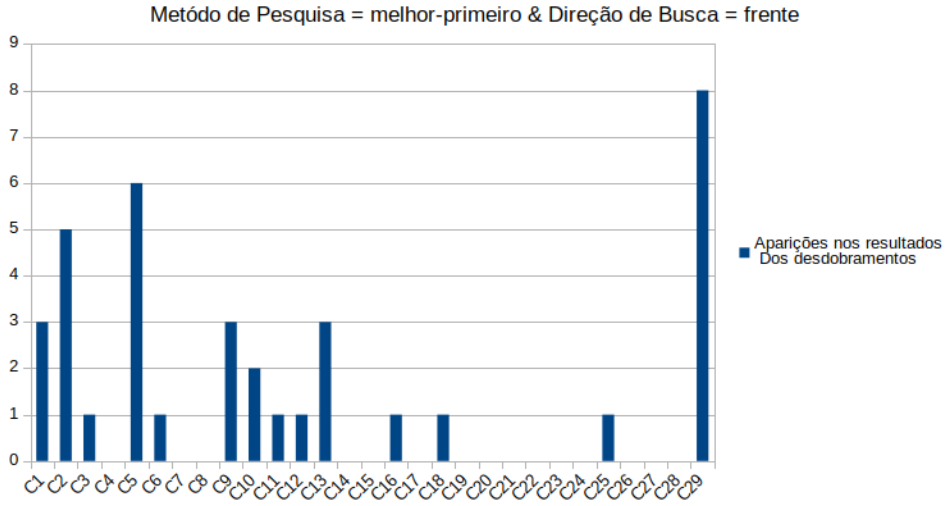

Figura 12 – Seleção de atributos melhor primeiro e busca pra frente

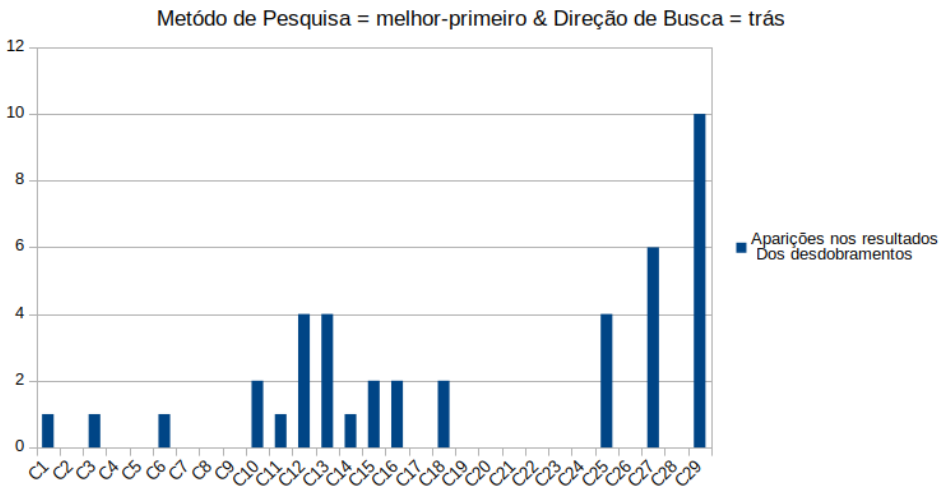

Figura 13 – Seleção de atributos melhor primeiro e busca pra trás

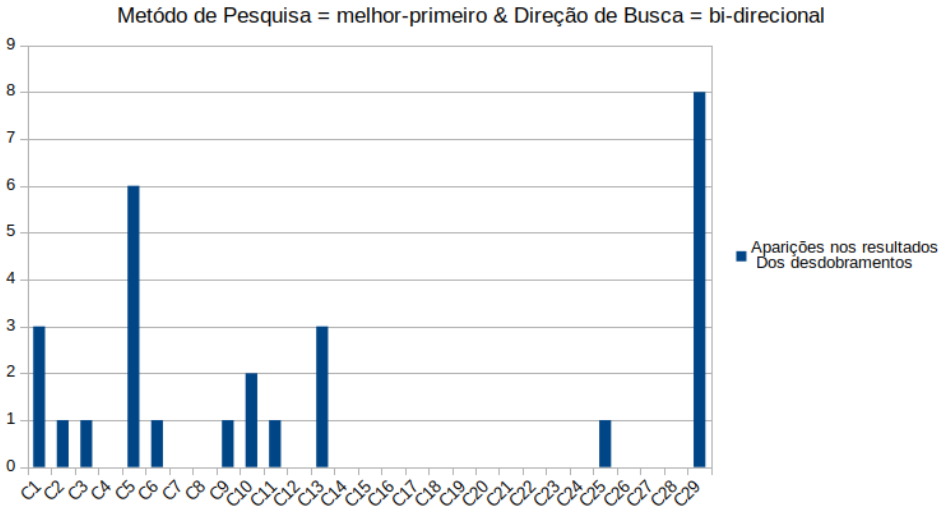

Figura 14 – Seleção de atributos melhor primeiro e busca bi-direcional

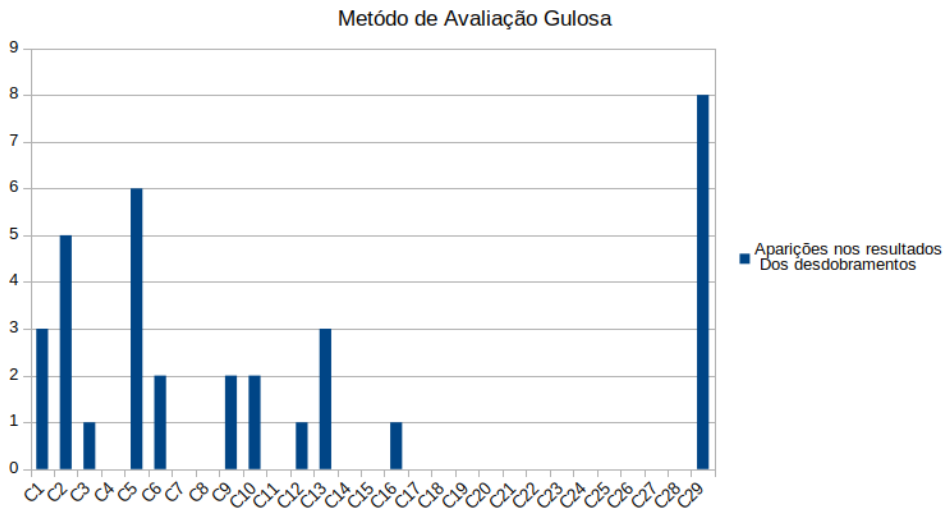

Figura 15 – Seleção de atributos utilizando avaliação gulosa

O resultado da aplicação atribuiu relevância na composição do modelo preditivo apenas a um sub-conjunto de apenas 17 atributos, composto pelas características $C1, C2, C3, C5, C6, C9, C10, C11, C12, C13, C14, C15, C16, C18, C25, C27, C29$. Este subconjunto será adotado nas demais etapas do procedimento. Da análise deste subconjunto resultante, observa-se que exceto a característica $C29$, todas as demais são relativas ao domínio e não a instância.

Outro aspecto essencial ao processo consiste na obtenção das melhores configurações dos algoritmos de Aprendizagem de Máquina que foram descritos na Seção 5.7. Para cada um dos algoritmos foram testadas diversas combinações de configurações, sendo que cada combinação foi executada 100 vezes para um conjunto de dados segundo a abordagem *Percentage Split* gerando sementes aleatórias únicas em cada execução. As configurações que obtiveram os melhores resultados são as listadas abaixo:

A taxa de acerto por algoritmo da aplicação da abordagem *Leave One Domain Out* é apresentada na Tabela 3 .

Tabela 3 – Resultados da abordagem Leave One Domain Out

	Naive Bayes	Árvore de Decisão	Tabela de Decisão	KNN	SVM	MLP
div1	75	75	65	65	75	75
div2	45	70	45	70	35	45
div3	30	30	30	80	80	30
div4	45	80	30	80	80	40
média	48,75	63,75	42,5	73,75	67,5	47,5

Em decorrência de grande parte das características identificadas pertencerem a definição do domínio e não da instância, muitos atributos se repetem para instâncias diferentes. Este fato se torna especialmente prejudicial na abordagem *Leave One Domain Out*, onde 10 registros de cada domínio contendo muitos atributos em comum são desconsiderados na etapa de treinamento. Como consequência, os dados para treinamento podem se tornar

pouco diversificados e o classificador resultante pode ter baixa capacidade de generalização. Esta situação pode ser verificada na tabela na execução da div3 do classificador gerado pela árvore de decisão, na div1 do KNN e também na div2 para o SVM. Entretanto, apesar deste problema, optou-se por se manter esta abordagem na decisão final do planejador, visto que os resultados obtidos por 4 dos classificadores obtiveram um desempenho superior a linha de base.

Na estratégia *Percentage Split* foram realizados experimentos para cada classificador utilizando uma divisão de 75% da totalidade dos dados como treinamento. Este procedimento foi realizado para obter-se as melhores configurações dos classificadores, conforme mencionado anteriormente. A média e o desvio padrão dos resultados das amostras é apresentado na Tabela 4.

Tabela 4 – Resultados da abordagem Percentage Split

Classificador	média	desvio-padrão
Naive Bayes	62,04	10,57
Árvore de Decisão	77,53	7,71
Tabela de Decisão	64,94	10,09
KNN	78,16	7,47
SVM	75,32	10,88
MLP	77,63	9,24

Nesta abordagem, as combinações de configurações dos algoritmos no WEKA que obtiveram os melhores resultados foram as seguintes:

- Naive Bayes: `meta.AttributeSelectedClassifier '-E \"WrapperSubsetEval -B bayes.N`
- Árvore de Decisão: `:trees.J48 '-C 0.25 -M 2'`
- Tabela de Decisão: `DecisionTable -X 1 -S "weka.attributeSelection.BestFirst -D 1`
- KNN: `lazy.IBk '-K 2 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"`
- SVM: `functions.SMO '-C 2.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K \"functions.`
- MLP: `functions.MultilayerPerceptron '-L 0.2 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a`

Na etapa seguinte foi realizada a normalização dos resultados. A normalização é calculada pela divisão das diferenças entre o valor obtido pelo algoritmo e o valor mínimo dentre todos os demais e as diferenças entre os valores máximos e mínimos considerando todos os algoritmos. Foi atribuída uma importância de 50% para cada uma das duas abordagens. Os valores obtidos na tabela foram usados como peso para as decisões dos algoritmos na etapa final. O resultado da normalização é apresentado na Tabela 5.

Como o objetivo de validar o procedimento, foram aplicados os modelos preditivos para selecionar um planejador para o domínio do AIT. Os valores correspondentes a cada uma das características na especificação são descritas na tabela 6.1.

Tabela 5 – Resultados Normalizados para as duas estratégias

	Percentage Split	Leave One Domain Out	Total
Naive Bayes	14,24	0,00	7,12
Árvore de Decisão	17,80	23,61	20,70
Tabela de Decisão	14,91	0,00	7,45
KNN	17,94	27,31	22,63
SVM	17,29	25,00	21,15
MLP	17,82	0,00	8,91

Tabela 6 – características e seus valores correspondentes para a especificação

Característica	Valor
C01	2
C02	8
C03	1
C05	10
C06	0
C09	1
C10	4
C11	0
C12	2
C13	0,375
C14	1
C15	4
C16	1,625
C18	7
C25	false
C27	false
C29	*dependente da instância

A característica C29 é a única que depende da instância. Para realizarmos os testes geramos 10 instâncias para o domínio (ver Apêndice B), cada uma delas contendo combinações bem distintas de habilidades e pré-requisitos, de forma a ter uma boa variabilidade dos dados. Os valores para a característica C29 em cada uma das instâncias são: 2,2,4,4,6,6,7,7,8 e 8.

A partir dos modelos preditivos gerados anteriormente e tendo como referência os pesos normalizados, foi realizado o teste para cada uma das instâncias, obtendo como resultado final a escolha pelo planejador POMDPX-NUS.

A etapa final consiste em validar a escolha realizada pelo procedimento descrito com o resultado das execuções dos planejadores probabilísticos. A validação do resultado do método foi realizado usando apenas planejadores POMDPX-NUS, Symbolic Perseus e KAIST AILAB, uma vez que os planejadores McGill, POND-Hindsight e HyPlan não tem os seus códigos-fontes disponibilizados, sendo impossível a sua avaliação. Também foram executadas as políticas aleatória e Noop, de forma similar aos critérios da competição *IPPC*.

Todos os testes foram realizados utilizando o ambiente de simulação *RDDLSim* (Sanner 2010) para a linguagem RDDL. O RDDLSim é software que implementa a arquitetura cliente-servidor e permite a fácil integração com os planejadores que estão sendo avaliados. Nas simulações o RDDLSim exerce o papel do estudante, fornecendo respostas para as ações pedagógicas que são fornecidas.

Os planejadores POMDPX-NUS, Symbolic Perseus e KAIST AILAB não interpretam a linguagem RDDL nativamente, sendo necessária realizar a conversão do domínio para os formatos POMDPX (para o planejador POMDPX-NUS) e SPUDD (para os planejadores Symbolic Perseus e KAIST AILAB). As traduções para os formatos POMDPX e SPUDD estavam disponíveis na data deste trabalho nos sítios (Approximate POMDP Planning Toolkit 2011) e (Hoey Jesse 2011), respectivamente. Durante a fase de tradução da especificação RDDL para o formato POMDPX, foi verificado que o planejador POMDPX NUS não suportava variavéis fluentes com domínio de valores do tipo real. Para resolver este problema foi realizado um processo discretização do valor da variável *KnowledgeProb* em 20 variáveis fluentes booleanas, cada uma representando intervalos de probabilidades de tamanho igual a 5. Assim, uma inferência em um conceito no modelo contínuo igual a 77%, no modelo discreto corresponderia a uma variável booleana simbolizando o intervalo 75% a 79,99%. Assume-se que este processo de discretização incorre em perda de precisão. No entanto, este procedimento é necessário para implementar as estimativas de conhecimento segundo o modelo *Knowledge Tracing*. Verifica-se ainda que a perda de precisão é de apenas 1,25% no caso médio, o que em muitos contextos não representa um grande prejuízo quando se trata de uma estimativa de probabilidade sobre o conhecimento, e que poderia ser discretizado em intervalos de tempos menores caso uma precisão maior seja necessária. O domínio discretizado é disponibilizado no Apendice C.

Os planejadores e suas respectivas pontuações médias, normalizadas, mínima e máxima para a instância 10 são apresentados na tabela 7. Para não prejudicar a leitura, os resultados das demais instâncias são disponibilizados no Apêndice D.

Tabela 7 – Resultado das execuções dos planejadores

	Pont. Média	Pont. Normalizada	Pont. Mínima	Pont. Máxima
Symbolic Perseus	1228,18	0,994	367,81	1064,66
POMDPX NUS	1333,40	1	715,43	1964,31
KAIST AILAB	1215,80	0,991	814,20	1774,18
RandonBoolPolicy	1027,06	0,540	615,58	1611,30
NoopPolicy	0	0	0	0

A política Noop não executa nenhuma ação. Como consequência, o nível de proficiência do estudante não é alterado. Como as recompensas são condicionadas a aquisição de conhecimento, a pontuação para esta política é sempre nula.

Os resultados demonstrados na Tabela 7 e das demais instâncias comprovam que para o domínio educacional que foi proposto, o planejador POMDPX NUS apontado como a

escolha ideal considerando as características de sua especificação e o método descrito, foi a escolha mais apropriada.

Capítulo 7

Conclusão

Neste trabalho foi apresentada a proposta de um modelo que realiza o planejamento instrucional de um sistema educacional. O objetivo é que este sistema tenha a capacidade de superar as dificuldades que são encontradas em uma sessão de ensino e propor um conjunto de ações que possibilitem melhorias no processo de ensino-aprendizagem. Dentre estas dificuldades, encontram-se as incertezas presentes sobre o conhecimento do estudante e também sobre os resultados das ações pedagógicas recomendadas. Para lidar com estas incertezas inerentes ao ensino, optou-se por uma abordagem baseada nos processos de decisão Markovianos Parcialmente Observáveis (POMDP), pela sua comprovada eficiência na representação de domínios de decisões sequencias onde o agente não contempla o ambiente em sua totalidade.

O modelo foi implementado através da linguagem RDDL, um formalismo que tem sido adotado recentemente pelas principais competições que envolvem planejadores probabilísticos. Cada elemento da implementação foi explicado e o código fonte disponibilizado em anexo, permitindo a sua incorporação no desenvolvimento de um sistema tutor, independente do domínio ou da linguagem de programação. Uma sessão de tutoria também foi apresentada, mostrando o comportamento do sistema conforme as iterações com o estudante.

O modelo faz uso de técnicas de rastreabilidade de conhecimento conhecidas para estimar o grau de proficiência do estudante em cada conceito curricular, conforme as respostas as ações instrucionais são fornecidas, permitindo dessa forma um acompanhamento mais próximo do aluno.

Como existem diversos planejadores probabilísticos, a qualidade das políticas de execução geradas são variáveis. Em virtude deste fato, realizamos a proposta de uma abordagem baseada em aprendizagem de máquina para auxiliar na escolha do planejador, levando em consideração as características do modelo especificado. Foi realizada uma filtragem destas características, eliminando aquelas que não contribuíam com o resultado final. Também foram realizados testes sobre os diversos parâmetros dos classificadores que foram escolhidos, com o intuito de se obter a melhor configuração possível durante o

experimento.

Os resultados do experimento para a seleção do planejador foram satisfatórios, em razão das seguintes observações realizadas. Primeiramente foi provado que os valores de recompensas esperados pela execução de um portfólio são superiores do que a escolha individual do melhor planejador, esta última determinando uma linha de base. Tal fato é verificado nas tabelas na seção de experimentos, que indicam o planejador ideal para cada instância do conjunto de problemas apresentado.

Por fim, foi realizada a validação dos resultados do experimento, aplicando-o ao modelo educacional proposto. Dentre as dificuldades encontradas, ressalta-se que nem todos os planejadores possuíam o seu código fonte disponibilizado. Diante desse fato, a validação foi restrita, incluindo somente os planejadores KAINST-AILAB, POMDPX-NUS e Symbolic Perseus. Estes planejadores não executam a especificação RDDL nativamente, sendo necessária a tradução do modelo para as linguagens SPUDD e POMDPX.

7.1 Trabalhos Futuros

Em razão da grande adesão de planejadores na edição 2011 do *IPPC* adotamos como critério para composição do portfólio aqueles que fizeram parte desta edição. Um possível ponto de melhoria é a ampliação do portfólio, incluindo não somente os algoritmos de planejamento mais recentes, bem como outros domínios de problemas com características que não foram explorados pela competição. Além disso, a competição não incluía fatores de desconto e o horizonte de planejamento limitava-se a 40 iterações. Estas configurações poderão ser ajustadas para a validação dos resultados obtidos pela classificação. Pretende-se ainda realizar experimentos com estudantes humanos para monitorar o comportamento e os ganhos do sistema quando aplicados a um contexto educacional real.

Referências

ALHOSSAINI, M. A.; BECK, J. C. Instance-specific remodelling of planning domains by adding macros and removing operators. In: **SARA**. [S.l.: s.n.], 2013.

Approximate POMDP Planning Toolkit. **Approximate POMDP Planning Toolkit**. 2011. Access date: 1 jun. 2011. Disponível em: <http://www.computing.dundee.ac.uk/staff/jessehoey/spudd/index.html>.

BELLMAN, R. **Dynamic programming**. [S.l.]: Courier Corporation, 2013.

BLOOM, B. S. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. **Educational researcher**, Sage Publications Sage CA: Thousand Oaks, CA, v. 13, n. 6, p. 4–16, 1984. Disponível em: <https://doi.org/10.3102/0013189X013006004>.

BOUCKAERT, R. R. et al. Weka manual for version 3-7-8, 2013. **Available at:. Accessed July**, v. 21, 2013.

BRANQUINHO, A. A. B. **Uma Abordagem apoiada em Portfólio de Planejadores Probabilísticos aplicada ao Mercado Financeiro**. Tese (Doutorado) — Universidade Federal de Uberlândia, 2017.

BRANQUINHO, A. A. B. et al. Planejamento de recursos para jogos de estratégia em tempo real. Universidade Federal de Uberlândia, 2009.

BRAZIUNAS, D. Pomdp solution methods. **University of Toronto**, 2003.

CENAMOR, I.; ROSA, T. D. L.; FERNÁNDEZ, F. The ibacop planning system: Instance-based configured portfolios. **Journal of Artificial Intelligence Research**, v. 56, p. 657–691, 2016. Disponível em: <https://doi.org/10.1613/jair.5080>.

CORBETT, A. T.; ANDERSON, J. R. Knowledge tracing: Modeling the acquisition of procedural knowledge. **User modeling and user-adapted interaction**, Springer, v. 4, n. 4, p. 253–278, 1994. Disponível em: <https://doi.org/10.1007/BF01099821>.

FIKES, R. E.; NILSSON, N. J. Strips: A new approach to the application of theorem proving to problem solving. **Artificial intelligence**, Elsevier, v. 2, n. 3-4, p. 189–208, 1971. Disponível em: <https://doi.org/10.1016/0004-3702(71)90010-5>.

FOLSOM-KOVARIK, J. T. et al. Scalable pomdps for diagnosis and planning in intelligent tutoring systems. In: **AAAI Fall Symposium: Proactive Assistant Agents**. [S.l.: s.n.], 2010.

FOWLER, D. G. A model for designing intelligent tutoring systems. **Journal of medical systems**, Springer, v. 15, n. 1, p. 47–63, 1991. Disponível em: <https://doi.org/10.1007/BF00993880>.

GHALLAB, M.; NAU, D.; TRAVERSO, P. **Automated Planning: theory and practice**. Elsevier, 2004. Disponível em: <https://doi.org/10.1016/B978-155860856-6/50020-X>.

GRUSZYNSKI, A. C. **Design gráfico: do invisível ao ilegível**. [S.l.]: Rosari, 2008.

Hoey Jesse. **SPUDD Translate**. 2011. Access date: 1 jun. 2011. Disponível em: <https://www.computing.dundee.ac.uk/staff/jessehoey/spudd/index.html>.

HSU, D.; LEE, W. S.; RONG, N. A point-based pomdp planner for target tracking. In: IEEE. **Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on**. 2008. p. 2644–2650. Disponível em: <https://doi.org/10.1109/ROBOT.2008.4543611>.

ICAPS IPPC-2011 log files. **Archive of log files and tabulated results**. 2011. Access date: 1 jun. 2011. Disponível em: <users.cecs.anu.edu.au/~ssanner/IPPC_2011/IPPC_2011_Data.tgz>.

JONASSEN, D. H.; WANG, S. The physics tutor: Integrating hypertext and expert systems. **Journal of Educational Technology Systems**, SAGE Publications Sage CA: Los Angeles, CA, v. 22, n. 1, p. 19–28, 1993. Disponível em: <https://doi.org/10.2190/7L7F-57H5-APYU-AAPA>.

KAPLAN, R.; ROCK, D. New directions for intelligent tutoring. **AI Expert**, [San Francisco, CA: CL Publications, Inc., c1986-, v. 10, n. 2, p. 30–40, 1995.

KOCSIS, L.; SZEPESVÁRI, C. Bandit based monte-carlo planning. In: SPRINGER. **European conference on machine learning**. 2006. p. 282–293. Disponível em: <https://doi.org/10.1007/11871842_29>.

KURNIAWATI, H.; HSU, D.; LEE, W. S. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In: ZURICH, SWITZERLAND. **Robotics: Science and systems**. 2008. v. 2008. Disponível em: <https://doi.org/10.15607/RSS.2008.IV.009>.

LOINAZ, M. U. Sistemas inteligentes en el ámbito de la educación. **Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial**, Asociación Española para la Inteligencia Artificial, v. 5, n. 12, 2001. Disponível em: <https://doi.org/10.4114/ia.v5i12.703>.

LONG, D.; FOX, M. The 3rd international planning competition: Results and analysis. **J. Artif. Int. Res.**, v. 20, n. 1, p. 1–59, dez. 2003. ISSN 1076-9757.

MANSKE, M.; CONATI, C. Modelling learning in an educational game. In: **AIED**. [S.l.: s.n.], 2005. p. 411–418.

MATSUDA, N.; VANLEHN, K. Decision theoretic instructional planner for intelligent tutoring systems. In: **Proceedings Workshop on Modeling Human Teaching Tactics and Strategies (ITS2000, pp. 72-83)**. [S.l.: s.n.], 2000.

MCDERMOTT, D. et al. Pddl-the planning domain definition language. 1998.

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

MEZA, H. V. et al. Aplicación de procesos markovianos para recomendar acciones pedagógicas óptimas en tutores inteligentes. **Research in Computing Science**, v. 111, p. 33–45, 2016.

National Center for Educational Statistics. U.S. Department of Education. **How is Grade Point Average Calculated**. 2012. Access date: 1 jun. 2011. Disponível em: <https://nces.ed.gov/nationsreportcard/hsts/howgpa.aspx>.

OLSEN, A.; BRYCE, D. Pond-hindsight: Applying hindsight optimization to pomdps.

PARDOS, Z. A.; HEFFERNAN, N. T. Modeling individualization in a bayesian networks implementation of knowledge tracing. In: SPRINGER. **International Conference on User Modeling, Adaptation, and Personalization**. 2010. p. 255–266. Disponível em: <https://doi.org/10.1007/978-3-642-13470-8_24>.

PEDNAULT, E. P. Adl: Exploring the middle ground between strips and the situation calculus. **Kr**, v. 89, p. 324–332, 1989.

PIECH, C. et al. Deep knowledge tracing. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 2015. p. 505–513.

PINEAU, J. et al. Point-based value iteration: An anytime algorithm for pomdps. In: **IJCAI**. [S.l.: s.n.], 2003. v. 3, p. 1025–1032.

POLSON, M. C.; RICHARDSON, J. J. **Foundations of intelligent tutoring systems**. Psychology Press, 2013. Disponível em: <https://doi.org/10.4324/9780203761557>.

POUPART, P. **Exploiting structure to efficiently solve large scale partially observable Markov decision processes**. [S.l.]: Citeseer, 2005.

POZZEBON, E. et al. Tutor inteligente adaptável conforme as preferências do aprendiz. Florianópolis, SC, 2003.

RAFFERTY, A. N. et al. Faster teaching by pomdp planning. In: SPRINGER. **International Conference on Artificial Intelligence in Education**. 2011. p. 280–287. Disponível em: <https://doi.org/10.1007/978-3-642-21869-9_37>.

REIS, H. M. et al. Towards reducing cognitive load and enhancing usability through a reduced graphical user interface for a dynamic geometry system: An experimental study. In: IEEE. **Multimedia (ISM), 2012 IEEE International Symposium on**. 2012. p. 445–450. Disponível em: <https://doi.org/10.1109/ISM.2012.91>.

RICKEL, J. W. Intelligent computer-aided instruction: A survey organized around system components. **IEEE Transactions on Systems, Man, and Cybernetics**, IEEE, v. 19, n. 1, p. 40–57, 1989. Disponível em: <https://doi.org/10.1109/21.24530>.

ROBERTS, M.; HOWE, A.; FLOM, L. Learned models of performance for many planners. In: **ICAPS 2007 Workshop AI Planning and Learning**. [S.l.: s.n.], 2007. p. 36–40.

ROSS, S. et al. Online planning algorithms for pomdps. **Journal of Artificial Intelligence Research**, v. 32, p. 663–704, 2008. Disponível em: <https://doi.org/10.1613/jair.2567>.

RUSSELL, S. J.; NORVIG, P. **Artificial intelligence: a modern approach**. [S.l.]: Malaysia; Pearson Education Limited„ 2016.

SALLANS, B. Learning factored representations for partially observable markov decision processes. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2000. p. 1050–1056.

SANNER, S. Relational dynamic influence diagram language (rddl): Language description. **Unpublished ms. Australian National University**, p. 32, 2010.

SEFFRIN, H. M. Avaliando o conhecimento algébrico do estudante através de redes bayesianas dinâmicas: um estudo de caso com o sistema tutor inteligente pat2math. Universidade do Vale do Rio dos Sinos, 2015.

SELF, J. The defining characteristics of intelligent tutoring systems research: Itss care, precisely. **International Journal of Artificial Intelligence in Education (IJAIED)**, v. 10, p. 350–364, 1998.

SHANI, G.; BRAFMAN, R. I.; SHIMONY, S. E. Forward search value iteration for pomdps. In: **IJCAI**. [S.l.: s.n.], 2007. p. 2619–2624.

SILVA, A. P. Aplicações de sistemas tutores inteligentes na educação a distância: possibilidades e limites. **Anais do Seminário Nacional ABED de Educação a Distância. Brasília: ABED**, 2006.

SILVER, D.; VENESS, J. Monte-carlo planning in large pomdps. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2010. p. 2164–2172.

SILVERIO, F. Recomendações de ações pedagógicas considerando características comportamentais e emocionais utilizando planejamento probabilístico. In: . [S.l.: s.n.], 2017.

SIM, H. S. et al. Symbolic heuristic search value iteration for factored pomdps. In: **AAAI**. [S.l.: s.n.], 2008. p. 1088–1093.

SMITH, T.; SIMMONS, R. Point-based pomdp algorithms: Improved analysis and implementation. **arXiv preprint arXiv:1207.1412**, 2012.

SOMANI, A. et al. Despot: Online pomdp planning with regularization. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2013. p. 1772–1780.

SOUSA, J. L. d. et al. Aplicando técnicas de aprendizado de máquina em planejamento. Universidade Federal de Uberlândia, 2014.

THEOCHAROUS, G. et al. Tractable pomdp planning algorithms for optimal teaching in "spais". In: **IJCAI PAIR Workshop**. [S.l.: s.n.], 2009.

VANLEHN, K. The behavior of tutoring systems. **International journal of artificial intelligence in education**, IOS Press, v. 16, n. 3, p. 227–265, 2006.

VRAKAS, D. et al. Learning rules for adaptive planning. In: **ICAPS**. [S.l.: s.n.], 2003. p. 82–91.

WANG, F. Pomdp framework for building an intelligent tutoring system. In: **CSEDU (1)**. [S.l.: s.n.], 2014. p. 233–240.

WELD, D. S. Recent advances in ai planning. **AI magazine**, v. 20, n. 2, p. 93, 1999.

WENGER, E. Artificial intelligence and tutoring system: Computational and cognitive approaches to the communication of knowledge. **Califórnia: Morgan Kaufmann Publishers. Texto publicado na: Pátio-revista pedagógica Editora Artes Médicas Sul Ano**, v. 1, p. 19–21, 1987.

WILLIAM, J. C. **Knowledge based tutoring: the GUIDON program**. [S.l.]: MIT Press, Cambridge, MA, 1987.

WITTEN, I. H. et al. **Data Mining: Practical machine learning tools and techniques**. [S.l.]: Morgan Kaufmann, 2016.

WOOLF, B. P. **Building Intelligent Interactive Tutors: Student-centered Strategies for Revolutionizing e-Learning**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007. ISBN 9780080920047.

YOON, S. W.; FERN, A.; GIVAN, R. Ff-replan: A baseline for probabilistic planning. In: **ICAPS**. [S.l.: s.n.], 2007. v. 7, p. 352–359.

YOON, S. W. et al. Probabilistic planning via determinization in hindsight. In: **AAAI**. [S.l.: s.n.], 2008. p. 1010–1016.

YUDELSON, M. V.; KOEDINGER, K. R.; GORDON, G. J. Individualized bayesian knowledge tracing models. In: SPRINGER. **International Conference on Artificial Intelligence in Education**. 2013. p. 171–180. Disponível em: <https://doi.org/10.1007/978-3-642-39112-5_18>.

ZHANG, P. **Using POMDP-based reinforcement learning for online optimization of teaching strategies in an intelligent tutoring system**. Tese (Doutorado), 2013.

Apêndices

APÊNDICE **A**

Especificação RDDL para o modelo AIT

```
domain AutonomousIntelligentTutor {

    requirements = {
        reward-deterministic,
        partially-observed,
        integer-valued

    };

    types  {
        skill : object;
    };

    pvariables {

        ////////////////////////////
        //Variaveis non-fluent ///
        ////////////////////////////
        SKILL_WEIGHT(skill) : { non-fluent, real, default = 1.0
           };
        PRE_REQ(skill, skill) : { non-fluent, bool, default =
           false };
        TRANSICTION_PROB(skill) : { non-fluent, real, default =
           0.8 };
        PROFICIENCY_AFTER_TEACH(skill): { non-fluent, real,
           default = 0.5 };
        GUESS(skill) : { non-fluent, real, default = 0.05 };
```

```
        SLIP(skill) : { non-fluent, real, default = 0.05 };

        //Definicoes de proficiencias
        LEVEL_FLUENT(skill) : { non-fluent, real, default = 0.9
           };
        LEVEL_HIGH(skill) : { non-fluent, real, default = 0.75 };
        LEVEL_MED(skill) : { non-fluent, real, default = 0.6};
        LEVEL_LOW(skill) : { non-fluent, real, default = 0.45};

      //Recompensas por nivel de proficiencia
      REWARD_PROFICIENCY_FLUENT(skill) : { non-fluent, real,
         default = 4 };
      REWARD_PROFICIENCY_HIGH(skill) : { non-fluent, real,
         default = 3 };
      REWARD_PROFICIENCY_MED(skill) : { non-fluent, real,
         default = 2 };
      REWARD_PROFICIENCY_LOW(skill) : { non-fluent, real,
         default = 1 };


        ////////////////////////////
        //State-fluent /////////////
        ////////////////////////////
        updateTurn(skill) : {state-fluent, bool, default = false
           };
        skillTaughtDelayVar(skill) : {state-fluent, bool, default
            = false};
        knowledgeProb(skill) : {state-fluent, real, default =
           0.0};
        answeredRight(skill) : {state-fluent, bool, default =
           false};
        proficiencyFluent(skill) : {state-fluent, bool, default =
            false};
        proficiencyHigh(skill) : { state-fluent, bool, default =
           false };
        proficiencyMed(skill) : { state-fluent, bool, default =
           false };
        proficiencyLow(skill) : { state-fluent, bool, default =
           false };

        ////////////////////////////
        //Action-fluent ///////////
```

```
        ////////////////////////////
        teach(skill) : {action-fluent, bool, default = false};
        evaluate(skill) : {action-fluent, bool, default = false};


        ////////////////////////////
        //Observ-fluent ////////////
        ////////////////////////////

        //answeredRightObs(skill) : {observ-fluent, bool};
        knowledgeProbObs(skill) : {observ-fluent, bool};

    };

    cpfs {

        updateTurn'(?s) =
            KronDelta( [forall_{?s2: skill} ~updateTurn(?s2)] ^ (
               evaluate(?s)| teach(?s)));

        skillTaughtDelayVar'(?s) =
            KronDelta( [forall_{?s2: skill} ~updateTurn(?s2)] ^
               teach(?s));

        //p(Ln) = P(Ln-1| evidencia) + (1 - p(Ln-1 | evidencia))
           * p(T);
        knowledgeProb'(?s) =
            if (~updateTurn(?s))
                then knowledgeProb(?s)
            else if(skillTaughtDelayVar(?s))
                then PROFICIENCY_AFTER_TEACH(?s)
            else if (answeredRight(?s))
                then knowledgeProb(?s) + (1 - knowledgeProb(?s))
                   *  TRANSICTION_PROB(?s)
            else (1 - knowledgeProb(?s)) +  (knowledgeProb(?s) *
               TRANSICTION_PROB(?s));

        //p(Cis) = p(Lrs) * (1-p(Sr)) + (1-p(Lrs)) * p(Gr)
        answeredRight'(?s) =
            if(~updateTurn(?s))
                then KronDelta (false)
```

```
            else if (forall_{?s2: skill} [PRE_REQ(?s2, ?s) ^ ~
               proficiencyFluent(?s2)])
                then Bernoulli(GUESS(?s))
            else Bernoulli(knowledgeProb(?s) * (1 - SLIP(?s)) +
               (1 - knowledgeProb(?s)) * GUESS(?s));

         proficiencyLow'(?s) =
            if (knowledgeProb(?s) >= LEVEL_LOW(?s))
                then KronDelta(true)
            else KronDelta(false);

        proficiencyMed'(?s) =
            if (knowledgeProb(?s) >= LEVEL_MED(?s))
                then KronDelta(true)
            else KronDelta(false);

         proficiencyHigh'(?s) =
            if (knowledgeProb(?s) >= LEVEL_HIGH(?s))
                then KronDelta(true)
            else KronDelta(false);

           proficiencyFluent'(?s) =
            if (knowledgeProb(?s) >= LEVEL_FLUENT(?s))
                then KronDelta(true)
            else KronDelta(false);

      /// answeredRightObs(?s) =
            //KronDelta( answeredRight'(?s) );

        knowledgeProbObs(?s) =
            DiracDelta( knowledgeProb'(?s) );
   };


   reward =  [sum_{?s : skill} [SKILL_WEIGHT(?s) *
      proficiencyLow(?s) * ~proficiencyMed(?s) * ~proficiencyHigh
      (?s) * ~proficiencyFluent(?s) * REWARD_PROFICIENCY_LOW(?s)
      ]] +
              [sum_{?s : skill} [SKILL_WEIGHT(?s) *
                 proficiencyMed(?s) * ~proficiencyHigh(?s) * ~
                 proficiencyFluent(?s) * REWARD_PROFICIENCY_MED(?s
                 )]] +
```

```
                [sum_{?s : skill} [SKILL_WEIGHT(?s) *
                   proficiencyHigh(?s) * ~proficiencyFluent(?s) *
                   REWARD_PROFICIENCY_HIGH(?s)]] +
                [sum_{?s : skill} [SKILL_WEIGHT(?s) *
                   proficiencyFluent(?s) * REWARD_PROFICIENCY_FLUENT
                   (?s)]];
}
```

APÊNDICE **B**

Instâncias para o modelo

```
        non-fluents nf\_AutonomousIntelligentTutor\_inst\_pomdp\_
           \_1 {
        domain = AutonomousIntelligentTutor;
        objects {
                skill : {s0,s1};

        };
        non-fluents {

                SKILL\_WEIGHT(s0) = 1.1563843;
                SKILL\_WEIGHT(s1) = 1.0460582;

        };
}
instance AutonomousIntelligentTutor\_inst\_pomdp\_\_1 {
        domain = AutonomousIntelligentTutor;
        non-fluents = nf\_AutonomousIntelligentTutor\_inst\_pomdp
           \_\_1;
        max-nondef-actions = 1;
        horizon = 40;
        discount = 1.0;
}
```

Listagem B.1 – Instância AutonomousIntelligentTutor_inst_pomdp___1.rddl

```
        non-fluents nf\_AutonomousIntelligentTutor\_inst\_pomdp\_
           \_2 {
        domain = AutonomousIntelligentTutor;
        objects {
```

```
                skill : {s0,s1};

        };
        non-fluents {

                SKILL\_WEIGHT(s0) = 1.0921998;
                SKILL\_WEIGHT(s1) = 1.2478412;

        };
}
instance AutonomousIntelligentTutor\_inst\_pomdp\_\_2 {
        domain = AutonomousIntelligentTutor;
        non-fluents = nf\_AutonomousIntelligentTutor\_inst\_pomdp
           \_\_2;
        max-nondef-actions = 1;
        horizon = 40;
        discount = 1.0;
}
```

Listagem B.2 – Instância AutonomousIntelligentTutor_inst_pomdp___2.rddl

```

        non-fluents nf\_AutonomousIntelligentTutor\_inst\_pomdp\_
           \_3 {
        domain = AutonomousIntelligentTutor;
        objects {
                skill : {s0,s1,s2,s3};

        };
        non-fluents {

                PRE\_REQ(s0, s2);
                PRE\_REQ(s1, s2);
                PRE\_REQ(s0, s3);

                SKILL\_WEIGHT(s0) = 1.1262993;
                SKILL\_WEIGHT(s1) = 1.014086;
                SKILL\_WEIGHT(s2) = 2.069316;
                SKILL\_WEIGHT(s3) = 2.2972388;

        };
}
instance AutonomousIntelligentTutor\_inst\_pomdp\_\_3 {
```

```
        domain = AutonomousIntelligentTutor;
        non-fluents = nf\_AutonomousIntelligentTutor\_inst\_pomdp
           \_\_3;
        max-nondef-actions = 1;
        horizon = 40;
        discount = 1.0;
}
```

Listagem B.3 – Instância AutonomousIntelligentTutor_inst_pomdp___3.rddl

```
        non-fluents nf\_AutonomousIntelligentTutor\_inst\_pomdp\_
           \_4 {
        domain = AutonomousIntelligentTutor;
        objects {
                skill : {s0,s1,s2,s3};

        };
        non-fluents {

                PRE\_REQ(s0, s2);
                PRE\_REQ(s1, s2);
                PRE\_REQ(s2, s3);

                SKILL\_WEIGHT(s0) = 1.4267184;
                SKILL\_WEIGHT(s1) = 1.3916073;
                SKILL\_WEIGHT(s2) = 2.3499568;
                SKILL\_WEIGHT(s3) = 3.05222;

        };
}
instance AutonomousIntelligentTutor\_inst\_pomdp\_\_4 {
        domain = AutonomousIntelligentTutor;
        non-fluents = nf\_AutonomousIntelligentTutor\_inst\_pomdp
           \_\_4;
        max-nondef-actions = 1;
        horizon = 40;
        discount = 1.0;
}
```

Listagem B.4 – Instância AutonomousIntelligentTutor_inst_pomdp___4.rddl

```
        non-fluents nf\_AutonomousIntelligentTutor\_inst\_pomdp\_
           \_5 {
        domain = AutonomousIntelligentTutor;
        objects {
                skill : {s0,s1,s2,s3,s4,s5};

        };
        non-fluents {

                PRE\_REQ(s0, s2);
                PRE\_REQ(s1, s2);
                PRE\_REQ(s0, s3);
                PRE\_REQ(s1, s4);
                PRE\_REQ(s4, s5);

                SKILL\_WEIGHT(s0) = 1.1106293;
                SKILL\_WEIGHT(s1) = 1.4973819;
                SKILL\_WEIGHT(s2) = 2.1876209;
                SKILL\_WEIGHT(s3) = 2.0403035;
                SKILL\_WEIGHT(s4) = 2.4102356;
                SKILL\_WEIGHT(s5) = 3.1811306;

        };
}
instance AutonomousIntelligentTutor\_inst\_pomdp\_\_5 {
        domain = AutonomousIntelligentTutor;
        non-fluents = nf\_AutonomousIntelligentTutor\_inst\_pomdp
           \_\_5;
        max-nondef-actions = 1;
        horizon = 40;
        discount = 1.0;
}
```

Listagem B.5 – Instância AutonomousIntelligentTutor_inst_pomdp___5.rddl

```
        non-fluents nf\_AutonomousIntelligentTutor\_inst\_pomdp\_
           \_6 {
        domain = AutonomousIntelligentTutor;
        objects {
                skill : {s0,s1,s2,s3,s4,s5};

        };
```

```
        non-fluents {

                PRE\_REQ(s1, s2);
                PRE\_REQ(s0, s3);
                PRE\_REQ(s2, s3);
                PRE\_REQ(s0, s4);
                PRE\_REQ(s2, s5);

                SKILL\_WEIGHT(s0) = 1.3742342;
                SKILL\_WEIGHT(s1) = 1.138321;
                SKILL\_WEIGHT(s2) = 2.0038147;
                SKILL\_WEIGHT(s3) = 3.2604294;
                SKILL\_WEIGHT(s4) = 2.2573876;
                SKILL\_WEIGHT(s5) = 3.0533185;

        };
}
instance AutonomousIntelligentTutor\_inst\_pomdp\_\_6 {
        domain = AutonomousIntelligentTutor;
        non-fluents = nf\_AutonomousIntelligentTutor\_inst\_pomdp
           \_\_6;
        max-nondef-actions = 1;
        horizon = 40;
        discount = 1.0;
}
```

Listagem B.6 – Instância AutonomousIntelligentTutor_inst_pomdp___6.rddl

```
        non-fluents nf\_AutonomousIntelligentTutor\_inst\_pomdp\_
           \_7 {
        domain = AutonomousIntelligentTutor;
        objects {
                skill : {s0,s1,s2,s3,s4,s5,s6};

        };
        non-fluents {

                PRE\_REQ(s0, s2);
                PRE\_REQ(s2, s3);
                PRE\_REQ(s3, s4);
                PRE\_REQ(s2, s4);
                PRE\_REQ(s4, s5);
```

```
                PRE\_REQ(s0, s6);
                PRE\_REQ(s4, s6);

                SKILL\_WEIGHT(s0) = 1.3669298;
                SKILL\_WEIGHT(s1) = 1.0352057;
                SKILL\_WEIGHT(s2) = 2.2176514;
                SKILL\_WEIGHT(s3) = 3.1296713;
                SKILL\_WEIGHT(s4) = 4.3982325;
                SKILL\_WEIGHT(s5) = 5.3825264;
                SKILL\_WEIGHT(s6) = 5.3337073;



                TRANSICTION\_PROB(s4) = 0.75;
                TRANSICTION\_PROB(s5) = 0.7;
                TRANSICTION\_PROB(s6) = 0.7;
        };
}
instance AutonomousIntelligentTutor\_inst\_pomdp\_\_7 {
        domain = AutonomousIntelligentTutor;
        non-fluents = nf\_AutonomousIntelligentTutor\_inst\_pomdp
           \_\_7;
        max-nondef-actions = 1;
        horizon = 40;
        discount = 1.0;
}
```

Listagem B.7 – Instância AutonomousIntelligentTutor_inst_pomdp___7.rddl

```
        non-fluents nf\_AutonomousIntelligentTutor\_inst\_pomdp\_
           \_8 {
        domain = AutonomousIntelligentTutor;
        objects {
                skill : {s0,s1,s2,s3,s4,s5,s6};

        };
        non-fluents {

                PRE\_REQ(s1, s2);
                PRE\_REQ(s0, s3);
                PRE\_REQ(s2, s3);
                PRE\_REQ(s0, s4);
```

```
                PRE\_REQ(s3, s4);
                PRE\_REQ(s2, s4);
                PRE\_REQ(s1, s5);
                PRE\_REQ(s2, s5);
                PRE\_REQ(s5, s6);
                PRE\_REQ(s4, s6);


                TRANSICTION\_PROB(s4) = 0.75;
                TRANSICTION\_PROB(s5) = 0.7;
                TRANSICTION\_PROB(s6) = 0.7;
        };
}
instance AutonomousIntelligentTutor\_inst\_pomdp\_\_8 {
        domain = AutonomousIntelligentTutor;
        non-fluents = nf\_AutonomousIntelligentTutor\_inst\_pomdp
           \_\_8;
        max-nondef-actions = 1;
        horizon = 40;
        discount = 1.0;
}
```

Listagem B.8 – Instância AutonomousIntelligentTutor_inst_pomdp___8.rddl

```
        non-fluents nf\_AutonomousIntelligentTutor\_inst\_pomdp\_
           \_9 {
        domain = AutonomousIntelligentTutor;
        objects {
                skill : {s0,s1,s2,s3,s4,s5,s6,s7};

        };
        non-fluents {

                PRE\_REQ(s1, s2);
                PRE\_REQ(s1, s3);
                PRE\_REQ(s2, s4);
                PRE\_REQ(s1, s4);
                PRE\_REQ(s4, s5);
                PRE\_REQ(s5, s6);
                PRE\_REQ(s3, s7);

                SKILL\_WEIGHT(s0) = 1.0080537;
```

```
                SKILL\_WEIGHT(s1) = 1.1978351;
                SKILL\_WEIGHT(s2) = 2.1637115;
                SKILL\_WEIGHT(s3) = 2.3641436;
                SKILL\_WEIGHT(s4) = 3.2041914;
                SKILL\_WEIGHT(s5) = 4.039774;
                SKILL\_WEIGHT(s6) = 5.3715773;
                SKILL\_WEIGHT(s7) = 3.1539128;


                TRANSICTION\_PROB(s4) = 0.75;
                TRANSICTION\_PROB(s5) = 0.7;
                TRANSICTION\_PROB(s6) = 0.7;
                TRANSICTION\_PROB(s7) = 0.65;
        };
}
instance AutonomousIntelligentTutor\_inst\_pomdp\_\_9 {
        domain = AutonomousIntelligentTutor;
        non-fluents = nf\_AutonomousIntelligentTutor\_inst\_pomdp
           \_\_9;
        max-nondef-actions = 1;
        horizon = 40;
        discount = 1.0;
}
```

Listagem B.9 – Instância AutonomousIntelligentTutor_inst_pomdp___9.rddl

```
        non-fluents nf\_AutonomousIntelligentTutor\_inst\_pomdp\_
           \_10 {
        domain = AutonomousIntelligentTutor;
        objects {
                skill : {s0,s1,s2,s3,s4,s5,s6,s7};

        };
        non-fluents {

                PRE\_REQ(s1, s2);
                PRE\_REQ(s2, s3);
                PRE\_REQ(s0, s3);
                PRE\_REQ(s1, s3);
                PRE\_REQ(s1, s4);
                PRE\_REQ(s4, s5);
                PRE\_REQ(s1, s5);
```

```
                PRE\_REQ(s0, s5);
                PRE\_REQ(s4, s6);
                PRE\_REQ(s1, s6);
                PRE\_REQ(s2, s6);
                PRE\_REQ(s2, s7);

                SKILL\_WEIGHT(s0) = 1.1818901;
                SKILL\_WEIGHT(s1) = 1.4655554;
                SKILL\_WEIGHT(s2) = 2.3080103;
                SKILL\_WEIGHT(s3) = 3.375123;
                SKILL\_WEIGHT(s4) = 2.0841439;
                SKILL\_WEIGHT(s5) = 3.2145185;
                SKILL\_WEIGHT(s6) = 3.23505;
                SKILL\_WEIGHT(s7) = 3.047642;


                TRANSICTION\_PROB(s4) = 0.75;
                TRANSICTION\_PROB(s5) = 0.7;
                TRANSICTION\_PROB(s6) = 0.7;
                TRANSICTION\_PROB(s7) = 0.65;
        };
}
instance AutonomousIntelligentTutor\_inst\_pomdp\_\_10 {
        domain = AutonomousIntelligentTutor;
        non-fluents = nf\_AutonomousIntelligentTutor\_inst\_pomdp
           \_\_10;
        max-nondef-actions = 1;
        horizon = 40;
        discount = 1.0;
}
```

Listagem B.10 – Instância AutonomousIntelligentTutor_inst_pomdp___10.rddl

APÊNDICE C

Domínio AIT discretizado

```
    domain AutonomousIntelligentTutor {

requirements = {
    reward-deterministic,
    partially-observed,
    integer-valued

};

types  {
    skill : object;
};

pvariables {

    ////////////////////////////
    //Variaveis non-fluent ///
    ////////////////////////////
    SKILL_WEIGHT(skill) : { non-fluent, real, default = 1.0
       };
    PRE_REQ(skill, skill) : { non-fluent, bool, default =
       false };
    TRANSICTION_PROB(skill) : { non-fluent, real, default =
       0.8 };
    PROFICIENCY_AFTER_TEACH(skill): { non-fluent, real,
       default = 0.5 };
    GUESS(skill) : { non-fluent, real, default = 0.05 };
    SLIP(skill) : { non-fluent, real, default = 0.05 };
```

```
 //Definicoes de proficiencias
 LEVEL_FLUENT(skill) : { non-fluent, real, default = 0.9
    };
 LEVEL_HIGH(skill) : { non-fluent, real, default = 0.75 };
 LEVEL_MED(skill) : { non-fluent, real, default = 0.6};
 LEVEL_LOW(skill) : { non-fluent, real, default = 0.45};

//Recompensas por nivel de proficiencia
REWARD_PROFICIENCY_FLUENT(skill) : { non-fluent, real,
   default = 4 };
REWARD_PROFICIENCY_HIGH(skill) : { non-fluent, real,
   default = 3 };
REWARD_PROFICIENCY_MED(skill) : { non-fluent, real,
   default = 2 };
REWARD_PROFICIENCY_LOW(skill) : { non-fluent, real,
   default = 1 };


 ////////////////////////////
 //State-fluent /////////////
 ////////////////////////////
 updateTurn(skill) : {state-fluent, bool, default = false
    };
 skillTaughtDelayVar(skill) : {state-fluent, bool, default
     = false};
 knowledgeProb(skill) : {state-fluent, int, default =
    0.0};
 answeredRight(skill) : {state-fluent, bool, default =
    false};
 proficiencyFluent(skill) : {state-fluent, bool, default =
     false};
 proficiencyHigh(skill) : { state-fluent, bool, default =
    false };
 proficiencyMed(skill) : { state-fluent, bool, default =
    false };
 proficiencyLow(skill) : { state-fluent, bool, default =
    false };
 prob0-5(skill) : { state-fluent, bool, default = false };
 prob6-10(skill) : { state-fluent, bool, default = false
    };
 prob11-15(skill) : { state-fluent, bool, default = false
```

```
    };
prob16-20(skill) : { state-fluent, bool, default = false
    };
prob21-25(skill) : { state-fluent, bool, default = false
    };
prob26-30(skill) : { state-fluent, bool, default = false
    };
prob31-35(skill) : { state-fluent, bool, default = false
    };
prob36-40(skill) : { state-fluent, bool, default = false
    };
prob41-45(skill) : { state-fluent, bool, default = false
    };
prob46-50(skill) : { state-fluent, bool, default = false
    };
prob51-55(skill) : { state-fluent, bool, default = false
    };
prob56-60(skill) : { state-fluent, bool, default = false
    };
prob61-65(skill) : { state-fluent, bool, default = false
    };
prob66-70(skill) : { state-fluent, bool, default = false
    };
prob71-75(skill) : { state-fluent, bool, default = false
    };
prob76-80(skill) : { state-fluent, bool, default = false
    };
prob81-85(skill) : { state-fluent, bool, default = false
    };
prob86-90(skill) : { state-fluent, bool, default = false
    };
prob91-95(skill) : { state-fluent, bool, default = false
    };
prob96-100(skill) : { state-fluent, bool, default = false
     };

////////////////////////////
//Action-fluent ////////////
////////////////////////////
teach(skill) : {action-fluent, bool, default = false};
evaluate(skill) : {action-fluent, bool, default = false};
```

```
    ////////////////////////////
    //Observ-fluent ////////////
    ////////////////////////////
    knowledgePropObs(skill) : {observ-fluent, bool};


};

cpfs {

    updateTurn'(?s) =
        KronDelta( [forall_{?s2: skill} ~updateTurn(?s2)] ^ (
           evaluate(?s)| teach(?s)));

    skillTaughtDelayVar'(?s) =
        KronDelta( [forall_{?s2: skill} ~updateTurn(?s2)] ^
           teach(?s));



    //p(Cis) = p(Lrs) * (1-p(Sr)) + (1-p(Lrs)) * p(Gr)
    answeredRight'(?s) =
        if(~updateTurn(?s))
            then KronDelta (false)
        else if (forall_{?s2: skill} [PRE_REQ(?s2, ?s) ^ ~
           proficiencyFluent(?s2)])
            then Bernoulli(GUESS(?s))
        else Bernoulli(knowledgeProb(?s) * (1 - SLIP(?s)) +
           (1 - knowledgeProb(?s)) * GUESS(?s));

     proficiencyLow'(?s) =
        if (knowledgeProb(?s) >= LEVEL_LOW(?s))
            then KronDelta(true)
        else KronDelta(false);

    proficiencyMed'(?s) =
        if (knowledgeProb(?s) >= LEVEL_MED(?s))
            then KronDelta(true)
        else KronDelta(false);

     proficiencyHigh'(?s) =
```

```
        if (knowledgeProb(?s) >= LEVEL_HIGH(?s))
            then KronDelta(true)
        else KronDelta(false);

      proficiencyFluent'(?s) =
        if (knowledgeProb(?s) >= LEVEL_FLUENT(?s))
            then KronDelta(true)
        else KronDelta(false);

    prob0-5'(?s) =
        if(~updateTurn(?s) ^ prob0-5(?s) )
            then KronDelta(true)
        else if(~updateTurn(?s) ^ ~prob0-5(?s) )
            then KronDelta(false)
        else if(skillTaughtDelayVar(?s) ^ (
           PROFICIENCY_AFTER_TEACH(?s) >= 0.05))
            then KronDelta(true)
        else if(skillTaughtDelayVar(?s) ^ (
           PROFICIENCY_AFTER_TEACH(?s) < 0.05) )
            then KronDelta(false)
        else if (answeredRight(?s) ^ prob6-10(?s))
            then KronDelta(true)
        else if (answeredRight(?s) ^ ~prob6-10(?s) ^ (0.05 +
           (0.95 * TRANSICTION_PROB(?s))) >= 0.05)
            then KronDelta(true)

        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
            * TRANSICTION_PROB(?s))) >= 0.05)
            then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
            * TRANSICTION_PROB(?s))) < 0.05)
            then KronDelta(false)
        else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
           (0.95 * TRANSICTION_PROB(?s))) >= 0.05)
            then KronDelta(true)
        else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
           (0.95 * TRANSICTION_PROB(?s))) < 0.05)
            then KronDelta(false)
        else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
           (0.9 * TRANSICTION_PROB(?s))) >= 0.05)
            then KronDelta(true)
        else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
```

```
    (0.9 * TRANSICTION_PROB(?s))) < 0.05)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
    (0.85 * TRANSICTION_PROB(?s))) >= 0.05)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
    (0.85 * TRANSICTION_PROB(?s))) < 0.05)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
    (0.8 * TRANSICTION_PROB(?s))) >= 0.05)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
    (0.8 * TRANSICTION_PROB(?s))) < 0.05)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
    (0.75 * TRANSICTION_PROB(?s))) >= 0.05)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
    (0.75 * TRANSICTION_PROB(?s))) < 0.05)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
    (0.7 * TRANSICTION_PROB(?s))) >= 0.05)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
    (0.7 * TRANSICTION_PROB(?s))) < 0.05)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob61-65(?s) ^ (0.35 +
    (0.65 * TRANSICTION_PROB(?s))) >= 0.05)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob61-65(?s) ^ (0.35 +
    (0.65 * TRANSICTION_PROB(?s))) < 0.05)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob56-60(?s) ^ (0.4 +
    (0.6 * TRANSICTION_PROB(?s))) >= 0.05)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob56-60(?s) ^ (0.4 +
    (0.6 * TRANSICTION_PROB(?s))) < 0.05)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob51-55(?s) ^ (0.45 +
    (0.55 * TRANSICTION_PROB(?s))) >= 0.05)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob51-55(?s) ^ (0.45 +
```

```
            (0.55 * TRANSICTION_PROB(?s))) < 0.05)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob46-50(?s) ^ (0.5 +
            (0.50 * TRANSICTION_PROB(?s))) >= 0.05)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob46-50(?s) ^ (0.5 +
            (0.50 * TRANSICTION_PROB(?s))) < 0.05)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob41-45(?s) ^ (0.55 +
            (0.45 * TRANSICTION_PROB(?s))) >= 0.05)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob41-45(?s) ^ (0.55 +
            (0.45 * TRANSICTION_PROB(?s))) < 0.05)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob36-40(?s) ^ (0.6 +
            (0.40 * TRANSICTION_PROB(?s))) >= 0.05)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob36-40(?s) ^ (0.6 +
            (0.40 * TRANSICTION_PROB(?s))) < 0.05)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob31-35(?s) ^ (0.65 +
            (0.35 * TRANSICTION_PROB(?s))) >= 0.05)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob31-35(?s) ^ (0.65 +
            (0.35 * TRANSICTION_PROB(?s))) < 0.05)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob26-30(?s) ^ (0.7 +
            (0.3 * TRANSICTION_PROB(?s))) >= 0.05)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob26-30(?s) ^ (0.7 +
            (0.3 * TRANSICTION_PROB(?s))) < 0.05)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob21-25(?s) ^ (0.75 +
            (0.25 * TRANSICTION_PROB(?s))) >= 0.05)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob21-25(?s) ^ (0.75 +
            (0.25 * TRANSICTION_PROB(?s))) < 0.05)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob16-20(?s) ^ (0.8 +
            (0.2 * TRANSICTION_PROB(?s))) >= 0.05)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob16-20(?s) ^ (0.8 +
```

```
            (0.2 * TRANSICTION_PROB(?s))) < 0.05)
             then KronDelta(false)
         else if(~answeredRight(?s) ^ prob11-15(?s) ^ (0.85 +
            (0.15 * TRANSICTION_PROB(?s))) >= 0.05)
             then KronDelta(true)
         else if(~answeredRight(?s) ^ prob11-15(?s) ^ (0.85 +
            (0.15 * TRANSICTION_PROB(?s))) < 0.05)
             then KronDelta(false)
         else if(~answeredRight(?s) ^ prob6-10(?s) ^ (0.9 +
            (0.1 * TRANSICTION_PROB(?s))) >= 0.05)
             then KronDelta(true)
         else if(~answeredRight(?s) ^ prob6-10(?s) ^ (0.9 +
            (0.1 * TRANSICTION_PROB(?s))) < 0.05)
             then KronDelta(false)
         else KronDelta(false);

      prob6-10'(?s) =
         if(~updateTurn(?s) ^ prob6-10(?s) )
             then KronDelta(true)
         else if(~updateTurn(?s) ^ ~prob6-10(?s) )
             then KronDelta(false)
         else if(skillTaughtDelayVar(?s) ^ (
            PROFICIENCY_AFTER_TEACH(?s) >=0.1))
             then KronDelta(true)
         else if(skillTaughtDelayVar(?s) ^ (
            PROFICIENCY_AFTER_TEACH(?s) < 0.1))
             then KronDelta(false)
         else if (answeredRight(?s) ^ prob6-10(?s))
             then KronDelta(true)
         else if (answeredRight(?s) ^ ~prob6-10(?s) ^ (0.05 +
            (0.95 * TRANSICTION_PROB(?s))) >=0.1)
             then KronDelta(true)
         else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
             * TRANSICTION_PROB(?s))) >=0.1)
             then KronDelta(true)
         else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
             * TRANSICTION_PROB(?s))) < 0.1)
             then KronDelta(false)
         else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
            (0.95 * TRANSICTION_PROB(?s))) >=0.1)
             then KronDelta(true)
         else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
```

```
        (0.95 * TRANSICTION_PROB(?s))) < 0.1)
         then KronDelta(false)
      else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
        (0.9 * TRANSICTION_PROB(?s))) >=0.1)
         then KronDelta(true)
      else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
        (0.9 * TRANSICTION_PROB(?s))) < 0.1)
         then KronDelta(false)
      else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
        (0.85 * TRANSICTION_PROB(?s))) >=0.1)
         then KronDelta(true)
      else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
        (0.85 * TRANSICTION_PROB(?s))) < 0.1)
         then KronDelta(false)
      else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
        (0.8 * TRANSICTION_PROB(?s))) >=0.1)
         then KronDelta(true)
      else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
        (0.8 * TRANSICTION_PROB(?s))) < 0.1)
         then KronDelta(false)
      else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
        (0.75 * TRANSICTION_PROB(?s))) >=0.1)
         then KronDelta(true)
      else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
        (0.75 * TRANSICTION_PROB(?s))) < 0.1)
         then KronDelta(false)
      else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
        (0.7 * TRANSICTION_PROB(?s))) >=0.1)
         then KronDelta(true)
      else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
        (0.7 * TRANSICTION_PROB(?s))) < 0.1)
         then KronDelta(false)
      else if(~answeredRight(?s) ^ prob61-65(?s) ^ (0.35 +
        (0.65 * TRANSICTION_PROB(?s))) >=0.1)
         then KronDelta(true)
      else if(~answeredRight(?s) ^ prob61-65(?s) ^ (0.35 +
        (0.65 * TRANSICTION_PROB(?s))) < 0.1)
         then KronDelta(false)
      else if(~answeredRight(?s) ^ prob56-60(?s) ^ (0.4 +
        (0.6 * TRANSICTION_PROB(?s))) >=0.1)
         then KronDelta(true)
      else if(~answeredRight(?s) ^ prob56-60(?s) ^ (0.4 +
```

```
    (0.6 * TRANSICTION_PROB(?s))) < 0.1)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob51-55(?s) ^ (0.45 +
    (0.55 * TRANSICTION_PROB(?s))) >=0.1)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob51-55(?s) ^ (0.45 +
    (0.55 * TRANSICTION_PROB(?s))) < 0.1)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob46-50(?s) ^ (0.5 +
    (0.50 * TRANSICTION_PROB(?s))) >=0.1)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob46-50(?s) ^ (0.5 +
    (0.50 * TRANSICTION_PROB(?s))) < 0.1)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob41-45(?s) ^ (0.55 +
    (0.45 * TRANSICTION_PROB(?s))) >=0.1)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob41-45(?s) ^ (0.55 +
    (0.45 * TRANSICTION_PROB(?s))) < 0.1)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob36-40(?s) ^ (0.6 +
    (0.40 * TRANSICTION_PROB(?s))) >=0.1)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob36-40(?s) ^ (0.6 +
    (0.40 * TRANSICTION_PROB(?s))) < 0.1)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob31-35(?s) ^ (0.65 +
    (0.35 * TRANSICTION_PROB(?s))) >=0.1)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob31-35(?s) ^ (0.65 +
    (0.35 * TRANSICTION_PROB(?s))) < 0.1)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob26-30(?s) ^ (0.7 +
    (0.3 * TRANSICTION_PROB(?s))) >=0.1)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob26-30(?s) ^ (0.7 +
    (0.3 * TRANSICTION_PROB(?s))) < 0.1)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob21-25(?s) ^ (0.75 +
    (0.25 * TRANSICTION_PROB(?s))) >=0.1)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob21-25(?s) ^ (0.75 +
```

```
            (0.25 * TRANSICTION_PROB(?s))) < 0.1)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob16-20(?s) ^ (0.8 +
            (0.2 * TRANSICTION_PROB(?s))) >=0.1)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob16-20(?s) ^ (0.8 +
            (0.2 * TRANSICTION_PROB(?s))) < 0.1)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob11-15(?s) ^ (0.85 +
            (0.15 * TRANSICTION_PROB(?s))) >=0.1)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob11-15(?s) ^ (0.85 +
            (0.15 * TRANSICTION_PROB(?s))) < 0.1)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob6-10(?s) ^ (0.9 +
            (0.1 * TRANSICTION_PROB(?s))) >=0.1)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob6-10(?s) ^ (0.9 +
            (0.1 * TRANSICTION_PROB(?s))) < 0.1)
             then KronDelta(false)
        else KronDelta(false);

    prob11-15'(?s) =
        if(~updateTurn(?s) ^ prob11-15(?s) )
             then KronDelta(true)
        else if(~updateTurn(?s) ^ ~prob11-15(?s) )
             then KronDelta(false)
        else if(skillTaughtDelayVar(?s) ^ (
            PROFICIENCY_AFTER_TEACH(?s) >=0.15))
             then KronDelta(true)
        else if(skillTaughtDelayVar(?s) ^ (
            PROFICIENCY_AFTER_TEACH(?s) < 0.15))
             then KronDelta(false)
        else if (answeredRight(?s) ^ prob6-10(?s))
             then KronDelta(true)
        else if (answeredRight(?s) ^ ~prob6-10(?s) ^ (0.05 +
            (0.95 * TRANSICTION_PROB(?s))) >=0.15)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
             * TRANSICTION_PROB(?s))) >=0.15)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
```

```
        * TRANSICTION_PROB(?s))) < 0.15)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
       (0.95 * TRANSICTION_PROB(?s))) >=0.15)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
       (0.95 * TRANSICTION_PROB(?s))) < 0.15)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
       (0.9 * TRANSICTION_PROB(?s))) >=0.15)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
       (0.9 * TRANSICTION_PROB(?s))) < 0.15)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
       (0.85 * TRANSICTION_PROB(?s))) >=0.15)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
       (0.85 * TRANSICTION_PROB(?s))) < 0.15)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
       (0.8 * TRANSICTION_PROB(?s))) >=0.15)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
       (0.8 * TRANSICTION_PROB(?s))) < 0.15)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
       (0.75 * TRANSICTION_PROB(?s))) >=0.15)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
       (0.75 * TRANSICTION_PROB(?s))) < 0.15)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
       (0.7 * TRANSICTION_PROB(?s))) >=0.15)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
       (0.7 * TRANSICTION_PROB(?s))) < 0.15)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob61-65(?s) ^ (0.35 +
       (0.65 * TRANSICTION_PROB(?s))) >=0.15)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob61-65(?s) ^ (0.35 +
```

```
            (0.65 * TRANSICTION_PROB(?s))) < 0.15)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob56-60(?s) ^ (0.4 +
            (0.6 * TRANSICTION_PROB(?s))) >=0.15)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob56-60(?s) ^ (0.4 +
            (0.6 * TRANSICTION_PROB(?s))) < 0.15)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob51-55(?s) ^ (0.45 +
            (0.55 * TRANSICTION_PROB(?s))) >=0.15)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob51-55(?s) ^ (0.45 +
            (0.55 * TRANSICTION_PROB(?s))) < 0.15)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob46-50(?s) ^ (0.5 +
            (0.50 * TRANSICTION_PROB(?s))) >=0.15)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob46-50(?s) ^ (0.5 +
            (0.50 * TRANSICTION_PROB(?s))) < 0.15)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob41-45(?s) ^ (0.55 +
            (0.45 * TRANSICTION_PROB(?s))) >=0.15)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob41-45(?s) ^ (0.55 +
            (0.45 * TRANSICTION_PROB(?s))) < 0.15)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob36-40(?s) ^ (0.6 +
            (0.40 * TRANSICTION_PROB(?s))) >=0.15)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob36-40(?s) ^ (0.6 +
            (0.40 * TRANSICTION_PROB(?s))) < 0.15)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob31-35(?s) ^ (0.65 +
            (0.35 * TRANSICTION_PROB(?s))) >=0.15)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob31-35(?s) ^ (0.65 +
            (0.35 * TRANSICTION_PROB(?s))) < 0.15)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob26-30(?s) ^ (0.7 +
            (0.3 * TRANSICTION_PROB(?s))) >=0.15)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob26-30(?s) ^ (0.7 +
```

```
            (0.3 * TRANSICTION_PROB(?s))) < 0.15)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob21-25(?s) ^ (0.75 +
            (0.25 * TRANSICTION_PROB(?s))) >=0.15)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob21-25(?s) ^ (0.75 +
            (0.25 * TRANSICTION_PROB(?s))) < 0.15)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob16-20(?s) ^ (0.8 +
            (0.2 * TRANSICTION_PROB(?s))) >=0.15)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob16-20(?s) ^ (0.8 +
            (0.2 * TRANSICTION_PROB(?s))) < 0.15)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob11-15(?s) ^ (0.85 +
            (0.15 * TRANSICTION_PROB(?s))) >=0.15)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob11-15(?s) ^ (0.85 +
            (0.15 * TRANSICTION_PROB(?s))) < 0.15)
             then KronDelta(false)
        else KronDelta(false);

    prob16-20'(?s) =
        if(~updateTurn(?s) ^ prob16-20(?s) )
             then KronDelta(true)
        else if(~updateTurn(?s) ^ ~prob16-20(?s) )
             then KronDelta(false)
        else if(skillTaughtDelayVar(?s) ^ (
            PROFICIENCY_AFTER_TEACH(?s) >=0.2))
             then KronDelta(true)
        else if(skillTaughtDelayVar(?s) ^ (
            PROFICIENCY_AFTER_TEACH(?s) < 0.2))
             then KronDelta(false)
        else if (answeredRight(?s) ^ prob6-10(?s))
             then KronDelta(true)
        else if (answeredRight(?s) ^ ~prob6-10(?s) ^ (0.05 +
            (0.95 * TRANSICTION_PROB(?s))) >=0.2)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
             * TRANSICTION_PROB(?s))) >=0.2)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
```

```
        * TRANSICTION_PROB(?s))) < 0.2)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
       (0.95 * TRANSICTION_PROB(?s))) >=0.2)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
       (0.95 * TRANSICTION_PROB(?s))) < 0.2)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
       (0.9 * TRANSICTION_PROB(?s))) >=0.2)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
       (0.9 * TRANSICTION_PROB(?s))) < 0.2)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
       (0.85 * TRANSICTION_PROB(?s))) >=0.2)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
       (0.85 * TRANSICTION_PROB(?s))) < 0.2)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
       (0.8 * TRANSICTION_PROB(?s))) >=0.2)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
       (0.8 * TRANSICTION_PROB(?s))) < 0.2)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
       (0.75 * TRANSICTION_PROB(?s))) >=0.2)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
       (0.75 * TRANSICTION_PROB(?s))) < 0.2)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
       (0.7 * TRANSICTION_PROB(?s))) >=0.2)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
       (0.7 * TRANSICTION_PROB(?s))) < 0.2)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob61-65(?s) ^ (0.35 +
       (0.65 * TRANSICTION_PROB(?s))) >=0.2)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob61-65(?s) ^ (0.35 +
```

```
    (0.65 * TRANSICTION_PROB(?s))) < 0.2)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob56-60(?s) ^ (0.4 +
    (0.6 * TRANSICTION_PROB(?s))) >=0.2)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob56-60(?s) ^ (0.4 +
    (0.6 * TRANSICTION_PROB(?s))) < 0.2)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob51-55(?s) ^ (0.45 +
    (0.55 * TRANSICTION_PROB(?s))) >=0.2)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob51-55(?s) ^ (0.45 +
    (0.55 * TRANSICTION_PROB(?s))) < 0.2)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob46-50(?s) ^ (0.5 +
    (0.50 * TRANSICTION_PROB(?s))) >=0.2)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob46-50(?s) ^ (0.5 +
    (0.50 * TRANSICTION_PROB(?s))) < 0.2)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob41-45(?s) ^ (0.55 +
    (0.45 * TRANSICTION_PROB(?s))) >=0.2)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob41-45(?s) ^ (0.55 +
    (0.45 * TRANSICTION_PROB(?s))) < 0.2)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob36-40(?s) ^ (0.6 +
    (0.40 * TRANSICTION_PROB(?s))) >=0.2)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob36-40(?s) ^ (0.6 +
    (0.40 * TRANSICTION_PROB(?s))) < 0.2)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob31-35(?s) ^ (0.65 +
    (0.35 * TRANSICTION_PROB(?s))) >=0.2)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob31-35(?s) ^ (0.65 +
    (0.35 * TRANSICTION_PROB(?s))) < 0.2)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob26-30(?s) ^ (0.7 +
    (0.3 * TRANSICTION_PROB(?s))) >=0.2)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob26-30(?s) ^ (0.7 +
```

```
            (0.3 * TRANSICTION_PROB(?s))) < 0.2)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob21-25(?s) ^ (0.75 +
            (0.25 * TRANSICTION_PROB(?s))) >=0.2)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob21-25(?s) ^ (0.75 +
            (0.25 * TRANSICTION_PROB(?s))) < 0.2)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob16-20(?s) ^ (0.8 +
            (0.2 * TRANSICTION_PROB(?s))) >=0.2)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob16-20(?s) ^ (0.8 +
            (0.2 * TRANSICTION_PROB(?s))) < 0.2)
             then KronDelta(false)
        else KronDelta(false);

    prob21-25'(?s) =
        if(~updateTurn(?s) ^ prob21-25(?s) )
             then KronDelta(true)
        else if(~updateTurn(?s) ^ ~prob21-25(?s) )
             then KronDelta(false)
        else if(skillTaughtDelayVar(?s) ^ (
            PROFICIENCY_AFTER_TEACH(?s) >= 0.25))
             then KronDelta(true)
        else if(skillTaughtDelayVar(?s) ^ (
            PROFICIENCY_AFTER_TEACH(?s) < 0.25))
             then KronDelta(false)
        else if (answeredRight(?s) ^ prob6-10(?s))
             then KronDelta(true)
        else if (answeredRight(?s) ^ ~prob6-10(?s) ^ (0.05 +
            (0.95 * TRANSICTION_PROB(?s))) >= 0.25)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
             * TRANSICTION_PROB(?s))) >= 0.25)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
             * TRANSICTION_PROB(?s))) < 0.25)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
            (0.95 * TRANSICTION_PROB(?s))) >= 0.25)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
```

```
      (0.95 * TRANSICTION_PROB(?s))) < 0.25)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
      (0.9 * TRANSICTION_PROB(?s))) >= 0.25)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
      (0.9 * TRANSICTION_PROB(?s))) < 0.25)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
      (0.85 * TRANSICTION_PROB(?s))) >= 0.25)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
      (0.85 * TRANSICTION_PROB(?s))) < 0.25)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
      (0.8 * TRANSICTION_PROB(?s))) >= 0.25)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
      (0.8 * TRANSICTION_PROB(?s))) < 0.25)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
      (0.75 * TRANSICTION_PROB(?s))) >= 0.25)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
      (0.75 * TRANSICTION_PROB(?s))) < 0.25)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
      (0.7 * TRANSICTION_PROB(?s))) >= 0.25)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
      (0.7 * TRANSICTION_PROB(?s))) < 0.25)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob61-65(?s) ^ (0.35 +
      (0.65 * TRANSICTION_PROB(?s))) >= 0.25)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob61-65(?s) ^ (0.35 +
      (0.65 * TRANSICTION_PROB(?s))) < 0.25)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob56-60(?s) ^ (0.4 +
      (0.6 * TRANSICTION_PROB(?s))) >= 0.25)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob56-60(?s) ^ (0.4 +
```

```
            (0.6 * TRANSICTION_PROB(?s))) < 0.25)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob51-55(?s) ^ (0.45 +
            (0.55 * TRANSICTION_PROB(?s))) >= 0.25)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob51-55(?s) ^ (0.45 +
            (0.55 * TRANSICTION_PROB(?s))) < 0.25)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob46-50(?s) ^ (0.5 +
            (0.50 * TRANSICTION_PROB(?s))) >= 0.25)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob46-50(?s) ^ (0.5 +
            (0.50 * TRANSICTION_PROB(?s))) < 0.25)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob41-45(?s) ^ (0.55 +
            (0.45 * TRANSICTION_PROB(?s))) >= 0.25)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob41-45(?s) ^ (0.55 +
            (0.45 * TRANSICTION_PROB(?s))) < 0.25)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob36-40(?s) ^ (0.6 +
            (0.40 * TRANSICTION_PROB(?s))) >= 0.25)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob36-40(?s) ^ (0.6 +
            (0.40 * TRANSICTION_PROB(?s))) < 0.25)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob31-35(?s) ^ (0.65 +
            (0.35 * TRANSICTION_PROB(?s))) >= 0.25)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob31-35(?s) ^ (0.65 +
            (0.35 * TRANSICTION_PROB(?s))) < 0.25)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob26-30(?s) ^ (0.7 +
            (0.3 * TRANSICTION_PROB(?s))) >= 0.25)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob26-30(?s) ^ (0.7 +
            (0.3 * TRANSICTION_PROB(?s))) < 0.25)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob21-25(?s) ^ (0.75 +
            (0.25 * TRANSICTION_PROB(?s))) >= 0.25)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob21-25(?s) ^ (0.75 +
```

```
            (0.25 * TRANSICTION_PROB(?s))) < 0.25)
             then KronDelta(false)
        else KronDelta(false);

    prob26-30'(?s) =
        if(~updateTurn(?s) ^ prob26-30(?s) )
            then KronDelta(true)
        else if(~updateTurn(?s) ^ ~prob26-30(?s) )
            then KronDelta(false)
        else if(skillTaughtDelayVar(?s) ^ (
           PROFICIENCY_AFTER_TEACH(?s) >= 0.3))
            then KronDelta(true)
        else if(skillTaughtDelayVar(?s) ^ (
           PROFICIENCY_AFTER_TEACH(?s) < 0.3))
            then KronDelta(false)
        else if (answeredRight(?s) ^ prob6-10(?s))
            then KronDelta(true)
        else if (answeredRight(?s) ^ ~prob6-10(?s) ^ (0.05 +
           (0.95 * TRANSICTION_PROB(?s))) >= 0.3)
            then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
            * TRANSICTION_PROB(?s))) >= 0.3)
            then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
            * TRANSICTION_PROB(?s))) < 0.3)
            then KronDelta(false)
        else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
           (0.95 * TRANSICTION_PROB(?s))) >= 0.3)
            then KronDelta(true)
        else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
           (0.95 * TRANSICTION_PROB(?s))) < 0.3)
            then KronDelta(false)
        else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
           (0.9 * TRANSICTION_PROB(?s))) >= 0.3)
            then KronDelta(true)
        else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
           (0.9 * TRANSICTION_PROB(?s))) < 0.3)
            then KronDelta(false)
        else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
           (0.85 * TRANSICTION_PROB(?s))) >= 0.3)
            then KronDelta(true)
        else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
```

```
            (0.85 * TRANSICTION_PROB(?s))) < 0.3)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
            (0.8 * TRANSICTION_PROB(?s))) >= 0.3)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
            (0.8 * TRANSICTION_PROB(?s))) < 0.3)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
            (0.75 * TRANSICTION_PROB(?s))) >= 0.3)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
            (0.75 * TRANSICTION_PROB(?s))) < 0.3)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
            (0.7 * TRANSICTION_PROB(?s))) >= 0.3)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
            (0.7 * TRANSICTION_PROB(?s))) < 0.3)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob61-65(?s) ^ (0.35 +
            (0.65 * TRANSICTION_PROB(?s))) >= 0.3)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob61-65(?s) ^ (0.35 +
            (0.65 * TRANSICTION_PROB(?s))) < 0.3)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob56-60(?s) ^ (0.4 +
            (0.6 * TRANSICTION_PROB(?s))) >= 0.3)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob56-60(?s) ^ (0.4 +
            (0.6 * TRANSICTION_PROB(?s))) < 0.3)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob51-55(?s) ^ (0.45 +
            (0.55 * TRANSICTION_PROB(?s))) >= 0.3)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob51-55(?s) ^ (0.45 +
            (0.55 * TRANSICTION_PROB(?s))) < 0.3)
             then KronDelta(false)
           else if(~answeredRight(?s) ^ prob46-50(?s) ^ (0.5 +
            (0.50 * TRANSICTION_PROB(?s))) >= 0.3)
             then KronDelta(true)
           else if(~answeredRight(?s) ^ prob46-50(?s) ^ (0.5 +
```

```
            (0.50 * TRANSICTION_PROB(?s))) < 0.3)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob41-45(?s) ^ (0.55 +
            (0.45 * TRANSICTION_PROB(?s))) >= 0.3)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob41-45(?s) ^ (0.55 +
            (0.45 * TRANSICTION_PROB(?s))) < 0.3)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob36-40(?s) ^ (0.6 +
            (0.40 * TRANSICTION_PROB(?s))) >= 0.3)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob36-40(?s) ^ (0.6 +
            (0.40 * TRANSICTION_PROB(?s))) < 0.3)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob31-35(?s) ^ (0.65 +
            (0.35 * TRANSICTION_PROB(?s))) >= 0.3)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob31-35(?s) ^ (0.65 +
            (0.35 * TRANSICTION_PROB(?s))) < 0.3)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob26-30(?s) ^ (0.7 +
            (0.3 * TRANSICTION_PROB(?s))) >= 0.3)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob26-30(?s) ^ (0.7 +
            (0.3 * TRANSICTION_PROB(?s))) < 0.3)
             then KronDelta(false)
        else KronDelta(false);

    prob31-35'(?s) =
        if(~updateTurn(?s) ^ prob31-35(?s) )
             then KronDelta(true)
        else if(~updateTurn(?s) ^ ~prob31-35(?s) )
             then KronDelta(false)
        else if(skillTaughtDelayVar(?s) ^ (
            PROFICIENCY_AFTER_TEACH(?s) >= 0.35))
             then KronDelta(true)
        else if(skillTaughtDelayVar(?s) ^ (
            PROFICIENCY_AFTER_TEACH(?s) < 0.35))
             then KronDelta(false)
        else if (answeredRight(?s) ^ prob6-10(?s))
             then KronDelta(true)
        else if (answeredRight(?s) ^ ~prob6-10(?s) ^ (0.05 +
```

```
        (0.95 * TRANSICTION_PROB(?s))) >= 0.35)
         then KronDelta(true)
       else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
         * TRANSICTION_PROB(?s))) >= 0.35)
         then KronDelta(true)
       else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
         * TRANSICTION_PROB(?s))) < 0.35)
         then KronDelta(false)
       else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
        (0.95 * TRANSICTION_PROB(?s))) >= 0.35)
         then KronDelta(true)
       else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
        (0.95 * TRANSICTION_PROB(?s))) < 0.35)
         then KronDelta(false)
       else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
        (0.9 * TRANSICTION_PROB(?s))) >= 0.35)
         then KronDelta(true)
       else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
        (0.9 * TRANSICTION_PROB(?s))) < 0.35)
         then KronDelta(false)
       else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
        (0.85 * TRANSICTION_PROB(?s))) >= 0.35)
         then KronDelta(true)
       else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
        (0.85 * TRANSICTION_PROB(?s))) < 0.35)
         then KronDelta(false)
       else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
        (0.8 * TRANSICTION_PROB(?s))) >= 0.35)
         then KronDelta(true)
       else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
        (0.8 * TRANSICTION_PROB(?s))) < 0.35)
         then KronDelta(false)
       else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
        (0.75 * TRANSICTION_PROB(?s))) >= 0.35)
         then KronDelta(true)
       else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
        (0.75 * TRANSICTION_PROB(?s))) < 0.35)
         then KronDelta(false)
       else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
        (0.7 * TRANSICTION_PROB(?s))) >= 0.35)
         then KronDelta(true)
       else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
```

```
    (0.7 * TRANSICTION_PROB(?s))) < 0.35)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob61-65(?s) ^ (0.35 +
    (0.65 * TRANSICTION_PROB(?s))) >= 0.35)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob61-65(?s) ^ (0.35 +
    (0.65 * TRANSICTION_PROB(?s))) < 0.35)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob56-60(?s) ^ (0.4 +
    (0.6 * TRANSICTION_PROB(?s))) >= 0.35)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob56-60(?s) ^ (0.4 +
    (0.6 * TRANSICTION_PROB(?s))) < 0.35)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob51-55(?s) ^ (0.45 +
    (0.55 * TRANSICTION_PROB(?s))) >= 0.35)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob51-55(?s) ^ (0.45 +
    (0.55 * TRANSICTION_PROB(?s))) < 0.35)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob46-50(?s) ^ (0.5 +
    (0.50 * TRANSICTION_PROB(?s))) >= 0.35)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob46-50(?s) ^ (0.5 +
    (0.50 * TRANSICTION_PROB(?s))) < 0.35)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob41-45(?s) ^ (0.55 +
    (0.45 * TRANSICTION_PROB(?s))) >= 0.35)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob41-45(?s) ^ (0.55 +
    (0.45 * TRANSICTION_PROB(?s))) < 0.35)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob36-40(?s) ^ (0.6 +
    (0.40 * TRANSICTION_PROB(?s))) >= 0.35)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob36-40(?s) ^ (0.6 +
    (0.40 * TRANSICTION_PROB(?s))) < 0.35)
     then KronDelta(false)
else if(~answeredRight(?s) ^ prob31-35(?s) ^ (0.65 +
    (0.35 * TRANSICTION_PROB(?s))) >= 0.35)
     then KronDelta(true)
else if(~answeredRight(?s) ^ prob31-35(?s) ^ (0.65 +
```

```
            (0.35 * TRANSICTION_PROB(?s))) < 0.35)
             then KronDelta(false)
        else KronDelta(false);

    prob36-40'(?s) =
        if(~updateTurn(?s) ^ prob36-40(?s) )
             then KronDelta(true)
        else if(~updateTurn(?s) ^ ~prob36-40(?s) )
             then KronDelta(false)
        else if(skillTaughtDelayVar(?s) ^ (
            PROFICIENCY_AFTER_TEACH(?s) >= 0.40))
             then KronDelta(true)
        else if(skillTaughtDelayVar(?s) ^ (
            PROFICIENCY_AFTER_TEACH(?s) < 0.40))
             then KronDelta(false)
        else if (answeredRight(?s) ^ prob6-10(?s))
             then KronDelta(true)
        else if (answeredRight(?s) ^ ~prob6-10(?s) ^ (0.05 +
            (0.95 * TRANSICTION_PROB(?s))) >= 0.40)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
             * TRANSICTION_PROB(?s))) >= 0.40)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
             * TRANSICTION_PROB(?s))) < 0.40)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
            (0.95 * TRANSICTION_PROB(?s))) >= 0.40)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
            (0.95 * TRANSICTION_PROB(?s))) < 0.40)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
            (0.9 * TRANSICTION_PROB(?s))) >= 0.40)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
            (0.9 * TRANSICTION_PROB(?s))) < 0.40)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
            (0.85 * TRANSICTION_PROB(?s))) >= 0.40)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
```

```
      (0.85 * TRANSICTION_PROB(?s))) < 0.40)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
      (0.8 * TRANSICTION_PROB(?s))) >= 0.40)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
      (0.8 * TRANSICTION_PROB(?s))) < 0.40)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
      (0.75 * TRANSICTION_PROB(?s))) >= 0.40)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
      (0.75 * TRANSICTION_PROB(?s))) < 0.40)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
      (0.7 * TRANSICTION_PROB(?s))) >= 0.40)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
      (0.7 * TRANSICTION_PROB(?s))) < 0.40)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob61-65(?s) ^ (0.35 +
      (0.65 * TRANSICTION_PROB(?s))) >= 0.40)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob61-65(?s) ^ (0.35 +
      (0.65 * TRANSICTION_PROB(?s))) < 0.40)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob56-60(?s) ^ (0.4 +
      (0.6 * TRANSICTION_PROB(?s))) >= 0.40)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob56-60(?s) ^ (0.4 +
      (0.6 * TRANSICTION_PROB(?s))) < 0.40)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob51-55(?s) ^ (0.45 +
      (0.55 * TRANSICTION_PROB(?s))) >= 0.40)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob51-55(?s) ^ (0.45 +
      (0.55 * TRANSICTION_PROB(?s))) < 0.40)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob46-50(?s) ^ (0.5 +
      (0.50 * TRANSICTION_PROB(?s))) >= 0.40)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob46-50(?s) ^ (0.5 +
```

```
            (0.50 * TRANSICTION_PROB(?s))) < 0.40)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob41-45(?s) ^ (0.55 +
            (0.45 * TRANSICTION_PROB(?s))) >= 0.40)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob41-45(?s) ^ (0.55 +
            (0.45 * TRANSICTION_PROB(?s))) < 0.40)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob36-40(?s) ^ (0.6 +
            (0.40 * TRANSICTION_PROB(?s))) >= 0.40)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob36-40(?s) ^ (0.6 +
            (0.40 * TRANSICTION_PROB(?s))) < 0.40)
             then KronDelta(false)
        else KronDelta(false);

    prob41-45'(?s) =
        if(~updateTurn(?s) ^ prob41-45(?s) )
             then KronDelta(true)
        else if(~updateTurn(?s) ^ ~prob41-45(?s) )
             then KronDelta(false)
        else if(skillTaughtDelayVar(?s) ^ (
            PROFICIENCY_AFTER_TEACH(?s) >= 0.45))
             then KronDelta(true)
        else if(skillTaughtDelayVar(?s) ^ (
            PROFICIENCY_AFTER_TEACH(?s) < 0.45))
             then KronDelta(false)
        else if (answeredRight(?s) ^ prob6-10(?s))
             then KronDelta(true)
        else if (answeredRight(?s) ^ ~prob6-10(?s) ^ (0.05 +
            (0.95 * TRANSICTION_PROB(?s))) >= 0.45)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
             * TRANSICTION_PROB(?s))) >= 0.45)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
             * TRANSICTION_PROB(?s))) < 0.45)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
            (0.95 * TRANSICTION_PROB(?s))) >= 0.45)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
```

```
      (0.95 * TRANSICTION_PROB(?s))) < 0.45)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
      (0.9 * TRANSICTION_PROB(?s))) >= 0.45)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
      (0.9 * TRANSICTION_PROB(?s))) < 0.45)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
      (0.85 * TRANSICTION_PROB(?s))) >= 0.45)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
      (0.85 * TRANSICTION_PROB(?s))) < 0.45)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
      (0.8 * TRANSICTION_PROB(?s))) >= 0.45)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
      (0.8 * TRANSICTION_PROB(?s))) < 0.45)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
      (0.75 * TRANSICTION_PROB(?s))) >= 0.45)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
      (0.75 * TRANSICTION_PROB(?s))) < 0.45)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
      (0.7 * TRANSICTION_PROB(?s))) >= 0.45)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
      (0.7 * TRANSICTION_PROB(?s))) < 0.45)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob61-65(?s) ^ (0.35 +
      (0.65 * TRANSICTION_PROB(?s))) >= 0.45)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob61-65(?s) ^ (0.35 +
      (0.65 * TRANSICTION_PROB(?s))) < 0.45)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob56-60(?s) ^ (0.4 +
      (0.6 * TRANSICTION_PROB(?s))) >= 0.45)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob56-60(?s) ^ (0.4 +
```

```
            (0.6 * TRANSICTION_PROB(?s))) < 0.45)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob51-55(?s) ^ (0.45 +
            (0.55 * TRANSICTION_PROB(?s))) >= 0.45)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob51-55(?s) ^ (0.45 +
            (0.55 * TRANSICTION_PROB(?s))) < 0.45)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob46-50(?s) ^ (0.5 +
            (0.50 * TRANSICTION_PROB(?s))) >= 0.45)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob46-50(?s) ^ (0.5 +
            (0.50 * TRANSICTION_PROB(?s))) < 0.45)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob41-45(?s) ^ (0.55 +
            (0.45 * TRANSICTION_PROB(?s))) >= 0.45)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob41-45(?s) ^ (0.55 +
            (0.45 * TRANSICTION_PROB(?s))) < 0.45)
             then KronDelta(false)
        else KronDelta(false);

    prob46-50'(?s) =
        if(~updateTurn(?s) ^ prob46-50(?s) )
             then KronDelta(true)
        else if(~updateTurn(?s) ^ ~prob46-50(?s) )
             then KronDelta(false)
        else if(skillTaughtDelayVar(?s) ^ (
            PROFICIENCY_AFTER_TEACH(?s) >= 0.5))
             then KronDelta(true)
        else if(skillTaughtDelayVar(?s) ^ (
            PROFICIENCY_AFTER_TEACH(?s) < 0.5))
             then KronDelta(false)
        else if (answeredRight(?s) ^ prob6-10(?s))
             then KronDelta(true)
        else if (answeredRight(?s) ^ ~prob6-10(?s) ^ (0.05 +
            (0.95 * TRANSICTION_PROB(?s))) >= 0.5)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
             * TRANSICTION_PROB(?s))) >= 0.5)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
```

```
        * TRANSICTION_PROB(?s))) < 0.5)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
       (0.95 * TRANSICTION_PROB(?s))) >= 0.5)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
       (0.95 * TRANSICTION_PROB(?s))) < 0.5)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
       (0.9 * TRANSICTION_PROB(?s))) >= 0.5)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
       (0.9 * TRANSICTION_PROB(?s))) < 0.5)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
       (0.85 * TRANSICTION_PROB(?s))) >= 0.5)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
       (0.85 * TRANSICTION_PROB(?s))) < 0.5)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
       (0.8 * TRANSICTION_PROB(?s))) >= 0.5)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
       (0.8 * TRANSICTION_PROB(?s))) < 0.5)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
       (0.75 * TRANSICTION_PROB(?s))) >= 0.5)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
       (0.75 * TRANSICTION_PROB(?s))) < 0.5)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
       (0.7 * TRANSICTION_PROB(?s))) >= 0.5)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
       (0.7 * TRANSICTION_PROB(?s))) < 0.5)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob61-65(?s) ^ (0.35 +
       (0.65 * TRANSICTION_PROB(?s))) >= 0.5)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob61-65(?s) ^ (0.35 +
```

```
            (0.65 * TRANSICTION_PROB(?s))) < 0.5)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob56-60(?s) ^ (0.4 +
            (0.6 * TRANSICTION_PROB(?s))) >= 0.5)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob56-60(?s) ^ (0.4 +
            (0.6 * TRANSICTION_PROB(?s))) < 0.5)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob51-55(?s) ^ (0.45 +
            (0.55 * TRANSICTION_PROB(?s))) >= 0.5)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob51-55(?s) ^ (0.45 +
            (0.55 * TRANSICTION_PROB(?s))) < 0.5)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob46-50(?s) ^ (0.5 +
            (0.50 * TRANSICTION_PROB(?s))) >= 0.5)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob46-50(?s) ^ (0.5 +
            (0.50 * TRANSICTION_PROB(?s))) < 0.5)
             then KronDelta(false)
        else KronDelta(false);

    prob51-55'(?s) =
        if(~updateTurn(?s) ^ prob51-55(?s) )
             then KronDelta(true)
        else if(~updateTurn(?s) ^ ~prob51-55(?s) )
             then KronDelta(false)
        else if(skillTaughtDelayVar(?s) ^ (
            PROFICIENCY_AFTER_TEACH(?s) >= 0.55))
             then KronDelta(true)
        else if(skillTaughtDelayVar(?s) ^ (
            PROFICIENCY_AFTER_TEACH(?s) < 0.55))
             then KronDelta(false)
        else if (answeredRight(?s) ^ prob6-10(?s))
             then KronDelta(true)
        else if (answeredRight(?s) ^ ~prob6-10(?s) ^ (0.05 +
            (0.95 * TRANSICTION_PROB(?s))) >= 0.55)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
             * TRANSICTION_PROB(?s))) >= 0.55)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
```

```
        * TRANSICTION_PROB(?s))) < 0.55)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
       (0.95 * TRANSICTION_PROB(?s))) >= 0.55)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
       (0.95 * TRANSICTION_PROB(?s))) < 0.55)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
       (0.9 * TRANSICTION_PROB(?s))) >= 0.55)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
       (0.9 * TRANSICTION_PROB(?s))) < 0.55)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
       (0.85 * TRANSICTION_PROB(?s))) >= 0.55)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
       (0.85 * TRANSICTION_PROB(?s))) < 0.55)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
       (0.8 * TRANSICTION_PROB(?s))) >= 0.55)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
       (0.8 * TRANSICTION_PROB(?s))) < 0.55)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
       (0.75 * TRANSICTION_PROB(?s))) >= 0.55)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
       (0.75 * TRANSICTION_PROB(?s))) < 0.55)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
       (0.7 * TRANSICTION_PROB(?s))) >= 0.55)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
       (0.7 * TRANSICTION_PROB(?s))) < 0.55)
        then KronDelta(false)
    else if(~answeredRight(?s) ^ prob61-65(?s) ^ (0.35 +
       (0.65 * TRANSICTION_PROB(?s))) >= 0.55)
        then KronDelta(true)
    else if(~answeredRight(?s) ^ prob61-65(?s) ^ (0.35 +
```

```
            (0.65 * TRANSICTION_PROB(?s))) < 0.55)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob56-60(?s) ^ (0.4 +
            (0.6 * TRANSICTION_PROB(?s))) >= 0.55)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob56-60(?s) ^ (0.4 +
            (0.6 * TRANSICTION_PROB(?s))) < 0.55)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob51-55(?s) ^ (0.45 +
            (0.55 * TRANSICTION_PROB(?s))) >= 0.55)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob51-55(?s) ^ (0.45 +
            (0.55 * TRANSICTION_PROB(?s))) < 0.55)
             then KronDelta(false)
        else KronDelta(false);

    prob56-60'(?s) =
        if(~updateTurn(?s) ^ prob56-60(?s) )
             then KronDelta(true)
        else if(~updateTurn(?s) ^ ~prob56-60(?s) )
             then KronDelta(false)
        else if(skillTaughtDelayVar(?s) ^ (
            PROFICIENCY_AFTER_TEACH(?s) >= 0.60))
             then KronDelta(true)
        else if(skillTaughtDelayVar(?s) ^ (
            PROFICIENCY_AFTER_TEACH(?s) < 0.60))
             then KronDelta(false)
        else if (answeredRight(?s) ^ prob6-10(?s))
             then KronDelta(true)
        else if (answeredRight(?s) ^ ~prob6-10(?s) ^ (0.05 +
            (0.95 * TRANSICTION_PROB(?s))) >= 0.60)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
             * TRANSICTION_PROB(?s))) >= 0.60)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
             * TRANSICTION_PROB(?s))) < 0.60)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
            (0.95 * TRANSICTION_PROB(?s))) >= 0.60)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
```

```
      (0.95 * TRANSICTION_PROB(?s))) < 0.60)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
      (0.9 * TRANSICTION_PROB(?s))) >= 0.60)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
      (0.9 * TRANSICTION_PROB(?s))) < 0.60)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
      (0.85 * TRANSICTION_PROB(?s))) >= 0.60)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
      (0.85 * TRANSICTION_PROB(?s))) < 0.60)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
      (0.8 * TRANSICTION_PROB(?s))) >= 0.60)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
      (0.8 * TRANSICTION_PROB(?s))) < 0.60)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
      (0.75 * TRANSICTION_PROB(?s))) >= 0.60)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
      (0.75 * TRANSICTION_PROB(?s))) < 0.60)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
      (0.7 * TRANSICTION_PROB(?s))) >= 0.60)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
      (0.7 * TRANSICTION_PROB(?s))) < 0.60)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob61-65(?s) ^ (0.35 +
      (0.65 * TRANSICTION_PROB(?s))) >= 0.60)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob61-65(?s) ^ (0.35 +
      (0.65 * TRANSICTION_PROB(?s))) < 0.60)
       then KronDelta(false)
    else if(~answeredRight(?s) ^ prob56-60(?s) ^ (0.4 +
      (0.6 * TRANSICTION_PROB(?s))) >= 0.60)
       then KronDelta(true)
    else if(~answeredRight(?s) ^ prob56-60(?s) ^ (0.4 +
```

```
            (0.6 * TRANSICTION_PROB(?s))) < 0.60)
             then KronDelta(false)
        else KronDelta(false);

    prob61-65'(?s) =
        if(~updateTurn(?s) ^ prob61-65(?s) )
             then KronDelta(true)
        else if(~updateTurn(?s) ^ ~prob61-65(?s) )
             then KronDelta(false)
        else if(skillTaughtDelayVar(?s) ^ (
           PROFICIENCY_AFTER_TEACH(?s) >= 0.65))
             then KronDelta(true)
        else if(skillTaughtDelayVar(?s) ^ (
           PROFICIENCY_AFTER_TEACH(?s) < 0.65))
             then KronDelta(false)
        else if (answeredRight(?s) ^ prob6-10(?s))
             then KronDelta(true)
        else if (answeredRight(?s) ^ ~prob6-10(?s) ^ (0.05 +
           (0.95 * TRANSICTION_PROB(?s))) >= 0.65)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
            * TRANSICTION_PROB(?s))) >= 0.65)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
            * TRANSICTION_PROB(?s))) < 0.65)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
           (0.95 * TRANSICTION_PROB(?s))) >= 0.65)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
           (0.95 * TRANSICTION_PROB(?s))) < 0.65)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
           (0.9 * TRANSICTION_PROB(?s))) >= 0.65)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
           (0.9 * TRANSICTION_PROB(?s))) < 0.65)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
           (0.85 * TRANSICTION_PROB(?s))) >= 0.65)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
```

```
            (0.85 * TRANSICTION_PROB(?s))) < 0.65)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
            (0.8 * TRANSICTION_PROB(?s))) >= 0.65)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
            (0.8 * TRANSICTION_PROB(?s))) < 0.65)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
            (0.75 * TRANSICTION_PROB(?s))) >= 0.65)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
            (0.75 * TRANSICTION_PROB(?s))) < 0.65)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
            (0.7 * TRANSICTION_PROB(?s))) >= 0.65)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
            (0.7 * TRANSICTION_PROB(?s))) < 0.65)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob61-65(?s) ^ (0.35 +
            (0.65 * TRANSICTION_PROB(?s))) >= 0.65)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob61-65(?s) ^ (0.35 +
            (0.65 * TRANSICTION_PROB(?s))) < 0.65)
             then KronDelta(false)
        else KronDelta(false);

    prob66-70'(?s) =
        if(~updateTurn(?s) ^ prob66-70(?s) )
             then KronDelta(true)
        else if(~updateTurn(?s) ^ ~prob66-70(?s) )
             then KronDelta(false)
        else if(skillTaughtDelayVar(?s) ^ (
            PROFICIENCY_AFTER_TEACH(?s) >= 0.70))
             then KronDelta(true)
        else if(skillTaughtDelayVar(?s) ^ (
            PROFICIENCY_AFTER_TEACH(?s) < 0.70))
             then KronDelta(false)
        else if (answeredRight(?s) ^ prob6-10(?s))
             then KronDelta(true)
        else if (answeredRight(?s) ^ ~prob6-10(?s) ^ (0.05 +
```

```
        (0.95 * TRANSICTION_PROB(?s))) >= 0.70)
         then KronDelta(true)
       else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
         * TRANSICTION_PROB(?s))) >= 0.70)
         then KronDelta(true)
       else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
         * TRANSICTION_PROB(?s))) < 0.70)
         then KronDelta(false)
       else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
        (0.95 * TRANSICTION_PROB(?s))) >= 0.70)
         then KronDelta(true)
       else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
        (0.95 * TRANSICTION_PROB(?s))) < 0.70)
         then KronDelta(false)
       else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
        (0.9 * TRANSICTION_PROB(?s))) >= 0.70)
         then KronDelta(true)
       else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
        (0.9 * TRANSICTION_PROB(?s))) < 0.70)
         then KronDelta(false)
       else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
        (0.85 * TRANSICTION_PROB(?s))) >= 0.70)
         then KronDelta(true)
       else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
        (0.85 * TRANSICTION_PROB(?s))) < 0.70)
         then KronDelta(false)
       else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
        (0.8 * TRANSICTION_PROB(?s))) >= 0.70)
         then KronDelta(true)
       else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
        (0.8 * TRANSICTION_PROB(?s))) < 0.70)
         then KronDelta(false)
       else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
        (0.75 * TRANSICTION_PROB(?s))) >= 0.70)
         then KronDelta(true)
       else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
        (0.75 * TRANSICTION_PROB(?s))) < 0.70)
         then KronDelta(false)
       else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
        (0.7 * TRANSICTION_PROB(?s))) >= 0.70)
         then KronDelta(true)
       else if(~answeredRight(?s) ^ prob66-70(?s) ^ (0.30 +
```

```
            (0.7 * TRANSICTION_PROB(?s))) < 0.70)
             then KronDelta(false)
        else KronDelta(false);

    prob71-75'(?s) =
        if(~updateTurn(?s) ^ prob71-75(?s) )
            then KronDelta(true)
        else if(~updateTurn(?s) ^ ~prob71-75(?s) )
            then KronDelta(false)
        else if(skillTaughtDelayVar(?s) ^ (
           PROFICIENCY_AFTER_TEACH(?s) >= 0.75))
            then KronDelta(true)
        else if(skillTaughtDelayVar(?s) ^ (
           PROFICIENCY_AFTER_TEACH(?s) < 0.75))
            then KronDelta(false)
        else if (answeredRight(?s) ^ prob6-10(?s))
            then KronDelta(true)
        else if (answeredRight(?s) ^ ~prob6-10(?s) ^ (0.05 +
           (0.95 * TRANSICTION_PROB(?s))) >= 0.75)
            then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
            * TRANSICTION_PROB(?s))) >= 0.75)
            then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
            * TRANSICTION_PROB(?s))) < 0.75)
            then KronDelta(false)
        else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
           (0.95 * TRANSICTION_PROB(?s))) >= 0.75)
            then KronDelta(true)
        else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
           (0.95 * TRANSICTION_PROB(?s))) < 0.75)
            then KronDelta(false)
        else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
           (0.9 * TRANSICTION_PROB(?s))) >= 0.75)
            then KronDelta(true)
        else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
           (0.9 * TRANSICTION_PROB(?s))) < 0.75)
            then KronDelta(false)
        else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
           (0.85 * TRANSICTION_PROB(?s))) >= 0.75)
            then KronDelta(true)
        else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
```

```
            (0.85 * TRANSICTION_PROB(?s))) < 0.75)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
            (0.8 * TRANSICTION_PROB(?s))) >= 0.75)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
            (0.8 * TRANSICTION_PROB(?s))) < 0.75)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
            (0.75 * TRANSICTION_PROB(?s))) >= 0.75)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob71-75(?s) ^ (0.25 +
            (0.75 * TRANSICTION_PROB(?s))) < 0.75)
             then KronDelta(false)
        else KronDelta(false);

    prob76-80'(?s) =
        if(~updateTurn(?s) ^ prob76-80(?s) )
             then KronDelta(true)
        else if(~updateTurn(?s) ^ ~prob76-80(?s) )
             then KronDelta(false)
        else if(skillTaughtDelayVar(?s) ^ (
            PROFICIENCY_AFTER_TEACH(?s) >= 0.80))
             then KronDelta(true)
        else if(skillTaughtDelayVar(?s) ^ (
            PROFICIENCY_AFTER_TEACH(?s) < 0.80))
             then KronDelta(false)
        else if (answeredRight(?s) ^ prob6-10(?s))
             then KronDelta(true)
        else if (answeredRight(?s) ^ ~prob6-10(?s) ^ (0.05 +
            (0.95 * TRANSICTION_PROB(?s))) >= 0.80)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
             * TRANSICTION_PROB(?s))) >= 0.80)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
             * TRANSICTION_PROB(?s))) < 0.80)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
            (0.95 * TRANSICTION_PROB(?s))) >= 0.80)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
```

```
            (0.95 * TRANSICTION_PROB(?s))) < 0.80)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
            (0.9 * TRANSICTION_PROB(?s))) >= 0.80)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
            (0.9 * TRANSICTION_PROB(?s))) < 0.80)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
            (0.85 * TRANSICTION_PROB(?s))) >= 0.80)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
            (0.85 * TRANSICTION_PROB(?s))) < 0.80)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
            (0.8 * TRANSICTION_PROB(?s))) >= 0.80)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob76-80(?s) ^ (0.2 +
            (0.8 * TRANSICTION_PROB(?s))) < 0.80)
             then KronDelta(false)
        else KronDelta(false);

    prob81-85'(?s) =
        if(~updateTurn(?s) ^ prob81-85(?s) )
             then KronDelta(true)
        else if(~updateTurn(?s) ^ ~prob81-85(?s) )
             then KronDelta(false)
        else if(skillTaughtDelayVar(?s) ^ (
            PROFICIENCY_AFTER_TEACH(?s) >= 0.85))
             then KronDelta(true)
        else if(skillTaughtDelayVar(?s) ^ (
            PROFICIENCY_AFTER_TEACH(?s) < 0.85))
             then KronDelta(false)
        else if (answeredRight(?s) ^ prob6-10(?s))
             then KronDelta(true)
        else if (answeredRight(?s) ^ ~prob6-10(?s) ^ (0.05 +
            (0.95 * TRANSICTION_PROB(?s))) >= 0.85)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
             * TRANSICTION_PROB(?s))) >= 0.85)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
```

```
            * TRANSICTION_PROB(?s))) < 0.85)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
           (0.95 * TRANSICTION_PROB(?s))) >= 0.85)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
           (0.95 * TRANSICTION_PROB(?s))) < 0.85)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
           (0.9 * TRANSICTION_PROB(?s))) >= 0.85)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
           (0.9 * TRANSICTION_PROB(?s))) < 0.85)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
           (0.85 * TRANSICTION_PROB(?s))) >= 0.85)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob81-85(?s) ^ (0.15 +
           (0.85 * TRANSICTION_PROB(?s))) < 0.85)
             then KronDelta(false)
        else KronDelta(false);

    prob86-90'(?s) =
        if(~updateTurn(?s) ^ prob0-5(?s) )
             then KronDelta(true)
        else if(~updateTurn(?s) ^ ~prob0-5(?s) )
             then KronDelta(false)
        else if(skillTaughtDelayVar(?s) ^ (
           PROFICIENCY_AFTER_TEACH(?s) >= 0.9))
             then KronDelta(true)
        else if(skillTaughtDelayVar(?s) ^ (
           PROFICIENCY_AFTER_TEACH(?s) < 0.9))
             then KronDelta(false)
        else if (answeredRight(?s) ^ prob6-10(?s))
             then KronDelta(true)
        else if (answeredRight(?s) ^ ~prob6-10(?s) ^ (0.05 +
           (0.95 * TRANSICTION_PROB(?s))) >= 0.9)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
            * TRANSICTION_PROB(?s))) >= 0.9)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
```

```
            * TRANSICTION_PROB(?s))) < 0.9)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
           (0.95 * TRANSICTION_PROB(?s))) >= 0.9)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
           (0.95 * TRANSICTION_PROB(?s))) < 0.9)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
           (0.9 * TRANSICTION_PROB(?s))) >= 0.9)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob86-90(?s) ^ (0.1 +
           (0.9 * TRANSICTION_PROB(?s))) < 0.9)
             then KronDelta(false)
        else KronDelta(false);

    prob91-95'(?s) =
        if(~updateTurn(?s) ^ prob0-5(?s) )
             then KronDelta(true)
        else if(~updateTurn(?s) ^ ~prob0-5(?s) )
             then KronDelta(false)
        else if(skillTaughtDelayVar(?s) ^ (
           PROFICIENCY_AFTER_TEACH(?s) >= 0.95))
             then KronDelta(true)
        else if(skillTaughtDelayVar(?s) ^ (
           PROFICIENCY_AFTER_TEACH(?s) < 0.95))
             then KronDelta(false)
        else if (answeredRight(?s) ^ prob6-10(?s))
             then KronDelta(true)
        else if (answeredRight(?s) ^ ~prob6-10(?s) ^ (0.05 +
           (0.95 * TRANSICTION_PROB(?s))) >= 0.95)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
             * TRANSICTION_PROB(?s))) >= 0.95)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
             * TRANSICTION_PROB(?s))) < 0.95)
             then KronDelta(false)
        else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
           (0.95 * TRANSICTION_PROB(?s))) >= 0.95)
             then KronDelta(true)
        else if(~answeredRight(?s) ^ prob91-95(?s) ^ (0.05 +
```

```
            (0.95 * TRANSICTION_PROB(?s))) < 0.95)
             then KronDelta(false)
        else KronDelta(false);

    prob96-100'(?s) =
        if(~updateTurn(?s) ^ prob96-100(?s) )
            then KronDelta(true)
        else if(~updateTurn(?s) ^ ~prob96-100(?s) )
            then KronDelta(false)
        else if(skillTaughtDelayVar(?s) ^ (
           PROFICIENCY_AFTER_TEACH(?s) >= 1))
            then KronDelta(true)
        else if(skillTaughtDelayVar(?s) ^ (
           PROFICIENCY_AFTER_TEACH(?s) < 1))
            then KronDelta(false)
        else if (answeredRight(?s) ^ prob6-10(?s))
            then KronDelta(true)
        else if (answeredRight(?s) ^ ~prob6-10(?s) ^ (0.05 +
           (0.95 * TRANSICTION_PROB(?s))) >= 1)
            then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
            * TRANSICTION_PROB(?s))) >= 1)
            then KronDelta(true)
        else if(~answeredRight(?s) ^ prob96-100(?s) ^ (0 + (1
            * TRANSICTION_PROB(?s))) < 1)
            then KronDelta(false)
        else KronDelta(false);

    knowledgeProbObs(?s) =
        DiracDelta( knowledgeProb'(?s) );
};

// A recompensa eh a soma de todas as habilidades que estao
   no nivel de proficiencia Mastery-Learning
// multiplicado pelo peso
reward =  [sum_{?s : skill} [SKILL_WEIGHT(?s) *
   proficiencyLow(?s) * ~proficiencyMed(?s) * ~proficiencyHigh
   (?s) * ~proficiencyFluent(?s) * REWARD_PROFICIENCY_LOW(?s)
   ]] +
          [sum_{?s : skill} [SKILL_WEIGHT(?s) *
             proficiencyMed(?s) * ~proficiencyHigh(?s) * ~
             proficiencyFluent(?s) * REWARD_PROFICIENCY_MED(?s
```

```
                )]] +
            [sum_{?s : skill} [SKILL_WEIGHT(?s) *
                proficiencyHigh(?s) * ~proficiencyFluent(?s) *
                REWARD_PROFICIENCY_HIGH(?s)]] +
            [sum_{?s : skill} [SKILL_WEIGHT(?s) *
                proficiencyFluent(?s) * REWARD_PROFICIENCY_FLUENT
                (?s)]];
}
```

Listagem C.1 – Domínio AIT discretizado

APÊNDICE **D**

Resultados das execuções dos planejadores

Tabela 8 – Resultado das execuções para a instância 1

	Pont. Média	Pont. Normalizada	Pont. Mínima	Pont. Máxima
Symbolic Perseus	145,63	0,94	116,99	177,51
POMDPX NUS	167,14	1	148,14	235,48
KAIST AILAB	198,42	0,97	133,49	221,18
RandonBoolPolicy	172,68	0,88	126,36	214,07
NoopPolicy	0	0	0	0

Tabela 9 – Resultado das execuções para a instância 2

	Pont. Média	Pont. Normalizada	Pont. Mínima	Pont. Máxima
Symbolic Perseus	193,08	0,98	139,87	248,48
POMDPX NUS	206,14	1	137,41	251,14
KAIST AILAB	200,80	0,92	134,18	218,60
RandonBoolPolicy	185,41	0,74	97,96	231.34
NoopPolicy	0	0	0	0

Tabela 10 – Resultado das execuções para a instância 3

	Pont. Média	Pont. Normalizada	Pont. Mínima	Pont. Máxima
Symbolic Perseus	552,25	0,97	367,81	689,55
POMDPX NUS	644,16	1	414,07	740,65
KAIST AILAB	608,29	0,99	460,48	688,73
RandonBoolPolicy	504,48	0,88	285,93	625,08
NoopPolicy	0	0	0	0

Tabela 11 – Resultado das execuções para a instância 4

	Pont. Média	Pont. Normalizada	Pont. Mínima	Pont. Máxima
Symbolic Perseus	724,18	0,96	455,7	874,62
POMDPX NUS	751,60	1	495,78	981,16
KAIST AILAB	604,70	0,86	467,75	886,49
RandonBoolPolicy	602,93	0,86	434,43	781,98
NoopPolicy	0	0	0	0

Tabela 12 – Resultado das execuções para a instância 5

	Pont. Média	Pont. Normalizada	Pont. Mínima	Pont. Máxima
Symbolic Perseus	771,46	0,98	579,22	1064,66
POMDPX NUS	785,16	1	590,01	1051,16
KAIST AILAB	779,16	0,99	532,16	1012,15
RandonBoolPolicy	613,64	0,73	464,21	943,97
NoopPolicy	0	0	0	0

Tabela 13 – Resultado das execuções para a instância 6

	Pont. Média	Pont. Normalizada	Pont. Mínima	Pont. Máxima
Symbolic Perseus	653,35	0,92	467,95	789,68
POMDPX NUS	745,26	1	514,17	840,75
KAIST AILAB	709,39	0,96	560,58	788,83
RandonBoolPolicy	605,58	0,88	385,03	725,18
NoopPolicy	0	0	0	0

Tabela 14 – Resultado das execuções para a instância 7

	Pont. Média	Pont. Normalizada	Pont. Mínima	Pont. Máxima
Symbolic Perseus	826,30	0,96	556,83	975,74
POMDPX NUS	852,70	1	596,79	1082,29
KAIST AILAB	705,80	0,86	568,85	987,59
RandonBoolPolicy	703,03	0,86	535,53	882,08
NoopPolicy	0	0	0	0

Tabela 15 – Resultado das execuções para a instância 8

	Pont. Média	Pont. Normalizada	Pont. Mínima	Pont. Máxima
Symbolic Perseus	872,56	0,98	670,32	1165,79
POMDPX NUS	886,33	1	691,14	1152,40
KAIST AILAB	890,28	0,99	634,29	1121,31
RandonBoolPolicy	730,76	0,73	565,34	1046,07
NoopPolicy	0	0	0	0

Tabela 16 – Resultado das execuções para a instância 9

	Pont. Média	Pont. Normalizada	Pont. Mínima	Pont. Máxima
Symbolic Perseus	1127,05	0,98	367,81	1064,66
POMDPX NUS	1232,27	1	614,30	1964,31
KAIST AILAB	1114,67	0,99	612,96	1673,05
RandonBoolPolicy	997,36	0,580	514,45	1510,18
NoopPolicy	0	0	0	0

Tabela 17 – Resultado das execuções para a instância 10

	Pont. Média	Pont. Normalizada	Pont. Mínima	Pont. Máxima
Symbolic Perseus	1228,18	0,994	367,81	1064,66
POMDPX NUS	1333,40	1	715,43	1964,31
KAIST AILAB	1215,80	0,99	814,20	1774,18
RandonBoolPolicy	1027,06	0,540	615,58	1611,30
NoopPolicy	0	0	0	0