UNIVERSIDADE FEDERAL DO RIO GRANDE
CENTRODE CIÊNCIAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM COMPUTACIONAL

Modelagem de Emoções utilizando Redes Bayesianas

por
Felipe Neves da Silva

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Modelagem Computacional da Universidade Federal do Rio Grande - FURG

Orientadora: Profª. Drª. Diana Francisca Adamatti
Coorientador: Prof. Dr. Adriano Velasque Werhli

Rio Grande, abril de 2014

Agradecimentos

Agradeço a minha família e principalmente a minha mãe Neide e minha irmã Paula que se estiveram ao meu lado vibrando com os bons momentos e apoiando nos momentos ruins.

Agradeço, também, aos meus amigos que ajudaram a atravessar essa longa jornada, em especial aos bons amigos Josimara Silveira e Leonardo Rodrigues, os quais conheci ao iniciarmos o curso de engenharia de computação nesta universidade e que também juntos, hoje, alcançamos uma grande vitória a concluirmos este curso.

Por fim, mas não menos importante, agradeço a todos meus professores pelos ensinamentos e principalmente pela confiança depositada em minhas capacidades. Em especial, agradeço a meus orientadores, professora Diana Adamatti e professor Adrianao Werhli, os quais sempre estiveram disponíveis a me ajudar sempre que precisei e que foram fundamentais para que eu pudesse concluir este trabalho, confiando em meu trabalho e esforço e tendo paciência para corrigir meus erros.

Resumo

As emoções são consideradas a regra central de nossas vidas, tendo grande impacto na tomada de decisões, ações, memória, atenção, etc. Sendo assim, existe grande interesse em simulá-las em ambientes computacionais, possibilitando que situações do cotidiano humano possam ser estudadas em ambientes controlados.

Embora existam modelos teóricos para o funcionamento de emoções, estes por si só são insuficientes para uma simulação precisa em meios computacionais. Tendo como base um destes modelos, o modelo OCC, essa dissertação propõe a simulação de emoções em ambientes mutiagentes através da criação de uma rede Bayesiana capaz de traduzir estímulos gerados neste ambiente em emoções.

A utilização de redes Bayesianas combinadas à estrutura do modelo OCC busca a adição de imprevisibilidade ao modelo, além de fornecê-lo uma estrutura computacional. A aplicação do modelo proposto a um sistema multiagentes proporciona o estudo da influência das emoções sobre as ações e comportamento dos agentes, possibilitando um estudo de comparação entre os resultados obtidos ao se realizar uma simulação multiagentes clássica e uma simulação multiagentes contendo emoções.

De forma a validar e avaliar seu funcionamento, é apresentado o estudo da aplicação da rede Bayesiana de emoções sobre um modelo multiagentes exemplo, observando as variações que as emoções provocam sobre o comportamento dos agentes.

Palavras-chave: Emoções, Inteligência Artificial, Sistemas Multiagentes, Redes Bayesianas.

Abstract

Emotions are considered the central rule of our lives, having great impact in decision-making, actions, memory, attention, etc. Thus, there is great interest in simulating them in computing environments, thus enabling the day-to-day human situations be studied in controlled environments.

Although are theoretical models for the operation of emotions does exist, these alone are insufficient for accurate simulation on computational means. Based on one of these models, the OCC model, this thesis proposes the simulation of emotions in multiagents environments by creating a Bayesian network able to translate stimulus generated in this environment on emotions.

The combined use of Bayesian networks in the structure of the OCC model seeks to add unpredictability to the deterministic model, besides providing a computational structure to the model. The application of the proposed model to a multiagent system provides the study of the influence of emotions on the actions and behavior of agents, thus allowing a study comparing the results obtained by performing a classical multi-agent simulation and multi-agent simulation having emotions.

In order to validate and evaluate its functioning, the study of the application of the Bayesian network of emotions over a multiagent model example is presented, noting the changes that emotions provoke on agents' behavior.

Keywords: Emotions, Artificial Intelligence, Multiagent Systems, Bayesians Networks.

Sumário

Lista de Figuras

Lista de Tabelas

1. Introdução

O ser humano sempre foi alvo de diversos estudos e serviu inspiração nas mais diversas áreas do conhecimento, entre elas a computação. A Inteligência Artificial (IA), por exemplo, possui algumas técnicas inspiradas tanto na fisiologia como no comportamento individual e social do ser humano. É o caso das Redes Neurais Artificiais e dos Sistemas Multiagentes.

Os Sistemas Multiagentes oferecem estruturas próprias para a simulação das mais diversas situações, os agentes, capazes de interagir entre si e com o ambiente ao qual estão inseridos. Com essas características, se tornaram a ferramenta preferida pelos desenvolvedores para realização da simulação das relações humanas em meios computacionais.

Entretanto o comportamento humano é determinado por diversas variáveis, muitas delas sem um método de simulação computacional definida, como as emoções, que interferem de forma decisiva no comportamento humano, influenciando a tomada de decisões, ações, memória, atenção, etc. (GRATCH; MARSELHA, 2001).

Existe um grande interesse em se simular emoções em ambientes computacionais, tanto para que seja possível a simulação do comportamento humano em aplicações como jogos e simulações multiagentes, quanto para que seja possível melhorar a interface entre aplicações e usuários, permitindo que a partir de uma análise das ações do usuário, um software possa responder de forma mais adequada ao estado emocional do usuário.

Existem modelos teóricos que tentam formalizar o funcionamento das emoções, dentre os quais se destaca o modelo OCC (ORTONY *et al.*, 1988), que relaciona as emoções aos eventos que as geram. Composto de 22 emoções, o modelo admite três formas de estímulo, eventos do ambiente, ações de indivíduos e objetos, bem como a reação do individuo simulado a estes estímulos, positivo ou negativo.

Por possuir uma estrutura fácil de traduzir para o meio computacional, o modelo OCC se tornou o modelo de emoções mais utilizado quando se deseja simular emoções. Entretanto, o modelo gera sempre as mesmas emoções a partir de um mesmo estímulo, o que não reflete com exatidão a condição

humana, onde muitas vezes ocorrem reações emocionais diferentes para uma mesma ocasião.

Sendo assim, para se adicionar a imprevisibilidade característica do ser humano ao modelo OCC, este trabalho apresenta a proposta de se construir uma rede Bayesiana que traduza a estrutura do modelo adicionando sua característica principal, o trabalho com a incerteza.

As redes Bayesianas são uma excelente ferramenta para modelagem de problemas reais por utilizarem o raciocínio probabilístico, que se diferencia do raciocínio lógico, utilizado pela maioria das ferramentas computacionais, por permitir o trabalho com informações incompletas do ambiente, situação comum neste tipo de problema, seja pela dificuldade, imprecisão ou, até mesmo, impossibilidade de coleta destas informações.

Existem diversos trabalhos que utilizam redes Bayesianas em conjunto com o modelo OCC de emoções. Entretanto, a maioria destes trabalhos os utiliza em sequência, isto é, utiliza a rede bayesiana como um pré-processamento para o modelo OCC, oferecendo para este a alimentação necessária, dividindo as ocorrências do ambiente nos estímulos trabalhados pelo modelo, eventos, ações e objetos.

Já nesta dissertação é apresentada a possibilidade de se converter o modelo OCC em uma rede Bayesiana acrescentando, assim, diferentes características ao modelo, como variáveis e valores probabilísticos. Esta nova estrutura possibilita que a partir de modificações para os valores de inicialização para rede apresentada, seja possível o trabalho com emoções para diferentes perfis de indivíduos.

Outro ponto importante a ressaltar é a portabilidade do modelo proposto, tendo em vista que a rede proposta pode ser aplicada diretamente a qualquer aplicação computacional, permitindo, portanto, que o trabalho com emoções se torne mais simples nestes softwares.

Para a utilização prática e avaliação dos resultados gerados da união do modelo OCC com as redes Bayesianas, propõe-se ainda que este modelo híbrido seja inserido em um sistema multiagentes, utilizando um exemplo para que se possam avaliar as mudanças de comportamento dos agentes devido a emoções geradas a partir de estímulos do ambiente, permitindo, desta forma, que os agentes tornem-se ainda mais próximos à realidade humana.

1.1. Objetivo

O trabalho tem como objetivo desenvolver uma rede Bayesiana inspirada no modelo OCC de emoções e aplicá-la em um ambiente multiagentes, isto é, apresenta-se uma maneira de simular emoções em meios computacionais, uma rede Bayesiana de emoções, a qual é utilizada em um ambiente multiagentes de forma a demonstrar sua efetividade.

Para alcançar este objetivo, algumas etapas foram executadas. Inicialmente, realizou-se um estudo sobre a modelagem de emoções, em especial sobre o modelo OCC, sobre as redes Bayesianas e sobre os ambientes de simulação multiagentes.

A partir destes estudos, inspirada na estrutura do modelo OCC foi construída a rede Bayesiana de emoções, a qual, em seguida, foi aplicada a um ambiente multiagentes, o Jason (BORDINI *et al.*, 2007).

Um dos exemplos disponíveis nesta ferramenta foi escolhido para ser modificado de forma que um de seus agentes passasse a possuir a rede Bayesiana de emoções. Uma série de estudos foi realizada sobre este exemplo de forma a demonstrar a funcionalidade da rede e seus efeitos sobre o comportamento deste agente.

1.2. Estrutura do Texto

O Capítulo 2 apresenta uma revisão sobre as três técnicas utilizadas na realização da modelagem, as Redes Bayesianas, os Modelos de Emoções e os Sistemas Multiagentes.

Uma visão geral sobre os trabalhos que vem sendo na área de modelagem computacional de emoções é apresentada no capitulo 3.

Em seguida, a construção da rede Bayesiana de emoções, bem como sua aplicação a um sistema multiagentes e o estudo de um exemplo avaliando suas características são mostrados no Capítulo 4. Os resultados obtidos a partir da criação do exemplo são discutidos no Capítulo 5, enquanto o Capítulo 6 apresenta algumas conclusões a cerca do trabalho realizado e algumas sugestões para a utilização da rede Bayesiana de emoções em futuros trabalhos.

Em anexo, estão disponíveis a definição da rede Bayesiana de emoções desenvolvida (ANEXO A), os planos e crenças dos agentes pertencentes ao exemplo trabalhado (ANEXO B), as probabilidades para as emoções de uma instância deste exemplo (ANEXO C), uma comparação entre o código original e o modificado para função *nextSlot* pertencente ao exemplo trabalhado (ANEXO D) e o código utilizado para definição do ambiente mutiagentes deste exemplo (ANEXO E).

2. Revisão Bibliográfica

A seguir será apresentada uma revisão sobre as três grandes áreas envolvidas no desenvolvimento deste trabalho, redes Bayesianas, modelagem de emoções e sistemas multiagentes.

2.1. Redes Bayesianas

Um grande problema para modelagem de problemas reais é a falta de informações completas sobre seu ambiente, seja pela dificuldade, imprecisão ou, até mesmo, impossibilidade de sua coleta. Nestes casos, a utilização do raciocínio probabilístico, bem como dos métodos que o utilizam se apresenta como uma boa alternativa.

"A principal vantagem do raciocínio probabilístico sobre raciocínio lógico é fato de que agentes podem tomar decisões racionais mesmo quando não existe informação suficiente para se provar que uma ação funcionará" (RUSSEL; NORVIG, 2003).

Uma importante ferramenta para modelagem de ambientes de incerteza são as chamadas redes Bayesianas que podem ser definidas como uma combinação da Teoria Probabilística e a Teoria de Grafos. Elas oferecem uma representação gráfica das relações probabilísticas existentes entre os diversos componentes do ambiente a ser modelado.

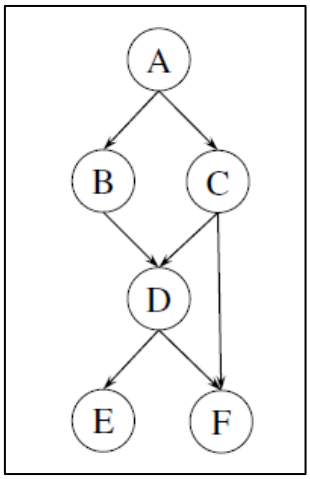


Figura 1 - Rede Bayesiana composta pelos nós N={A, B, C, D, E, F} e pelas arestas €={(A,B),(A,C),(B,D),(C,D),(D,E),(D, F),(C, F)} (WERHLI, 2007).

A Figura 1 apresenta a estrutura básica de uma rede Bayesiana, um grafo acíclico direcionado, ou seja, um grafo em que todas suas arestas são direcionadas e não existem ciclos.

2.1.1. Probabilidades em Redes Bayesianas

Redes Bayesianas utilizam a Teoria Probabilística para modelar as incertezas existentes em um determinado ambiente. A Teoria Probabilística é um campo da matemática que estuda e analisa a ocorrência de fenômenos aleatórios, os quais são experimentos repetidos sob as mesmas condições produzem resultados que não se pode prever com certeza (MORGADO *et al.*, 2001).

Dois tipos de probabilidades devem ser destacados: condicional e incondicional. O segundo, mais simples, é a probabilidade que independe de acontecimentos anteriores, o contrário do primeiro, o qual é amplamente utilizado para modelagem de redes Bayesianas.

Representa-se uma probabilidade condicional por $P(A|B)$, que significa a probabilidade de que o evento A ocorra dado a ocorrência de um evento B.

Considerando novamente a rede Bayesiana hipotética apresentada na Figura 1, sua estrutura é composta pelo grupo de nós N={A, B, C, D, E, F} e pelo conjunto de dependências entre nós representado pelo grupo de arestas €={(A, B), (A, C), (B, D), (C, D), (D, E), (D, F), (C, F)}. Existindo uma aresta direcionada do vértice A paro o nó B, então A é chamado pai de B, assim como B é dito filho ou descendente de A.

Uma rede Bayesiana é caracterizada por possuir uma regra simples e única capaz de expandir seu conjunto de probabilidades em termos de simples probabilidades condicionais. Esta regra segue a propriedade local de Markov, a qual diz que um nó é condicionalmente independente de todos seus não descendentes dados seus pais, e pode ser demonstrada na Equação 1:

$$P(X_1, X_2, \dots, X_n) = \prod_{i=1}^{n} P\left(X_i \middle| \pi_M(X_i)\right) \qquad \text{(Eq. 1)}$$

Primeiramente, é importante notar que (X_i) é utilizado para representar tanto um nó como também a variável aleatória que ele representa. Da mesma maneira, ($\pi_M(X_i)$) é o conjunto de nós pais e as variáveis aleatórias por eles

representada. Assim, $X_1, X_2, \ldots, X_n$ são variáveis aleatórias representadas por nós $X_i \in 1, \ldots, n$ e $\pi_M(X_i)$ é o conjunto de variáveis aleatórias representadas pelo conjunto de nós $\pi_M(X_i)$, os quais são pais do nó X_i em uma rede definida M.

Aplicando a Equação 1 para rede Bayesiana apresentada na Figura 1, obtém-se:

$$P(A,B,C,D,E,F) = P(A)P(B|A)P(C|A)P(D|B,C)P(E|D)P(F|D,C) \quad \text{(Eq. 2)}$$

A Equação 2 demonstra que para se obter o conjunto de probabilidades da rede Bayesiana apresentada na Figura 1 é necessário determinar cada uma das probabilidades condicionais existentes entre seus nós, apresentando uma visão clara da regra básica geral das redes Bayesianas.

2.1.2. Inferência em Redes Bayesianas

Após se definir uma rede Bayesiana, com suas variáveis e probabilidades, pode-se extrair conhecimento nela representado através de um processo de inferência.

Segundo (RUSSEL; NORVIG, 2003) existem quatro maneiras distintas de se realizar inferências sobre redes Bayesianas: diagnósticos, onde parte-se dos efeitos para se obter as causas; causa, partindo-se das causas em busca dos efeitos; intercausal, entre causas de efeito comum; mistas, combinação de dois ou mais tipos de inferência.

Ainda segundo (RUSSEL; NORVIG, 2003) métodos de inferência são utilizados visando diversos objetivos: tomar decisões baseadas em probabilidades; decidir quais evidências adicionais devem ser observadas, a fim de obter total conhecimento do domínio; realizar uma análise sensitiva para entender quais aspectos do modelo tem maior impacto sobre determinadas variáveis; e explicar os resultados de uma inferência probabilística ao usuário.

Segundo (HRUSCHKA JR., 2003) existem diversos métodos para realização de inferência, dentre os quais se podem destacar o método de inferência por eliminação de variáveis e o método do agrupamento.

2.1.2.1. Método da Eliminação de Variáveis

Observando a Equação 3, a maneira convencional para se avaliá-la é partir do valor mais a esquerda indo em direção ao mais a direita, como se partíssemos do nó raiz (topo) em direção a base da rede. Tratando do meio prático de uso de redes Bayesianas, os sistemas computacionais, um problema surge ao se usar este método de avaliação: a cada nova variável inserida na rede ocorre um crescimento no tempo de processamento da avaliação na ordem de 2^n, onde n é o numero de variáveis, ou seja, na prática, este método só pode ser utilizado em redes compostas por poucos nós.

$$P(b|j,m) = \alpha P(b) \sum_e P(e) \sum_\alpha P(a|b,e) P(j|a) P(m|a) \quad \text{(Eq. 3)}$$

Para contornar o problema anteriormente citado, gerado pela múltipla inferência de subexpressões, pode-se avaliar a equação da direita para esquerda (*botton - up*), armazenando os valores intermediários e eliminando da somatória os valores que independem da variável analisada. Este é o chamado método da eliminação de variáveis.

Assim, para iniciar a avaliação atribui-se um rótulo, uma letra maiúscula para o exemplo, para cada parte da equação: $M = P(m|a)$, $J = P(j|a)$, $A = P(a|b,e)$, $E = P(e)$, $B = P(b)$.

Para cada rótulo atribuído associa-se um fator que armazenará a probabilidade da variável representada através de uma matriz como no exemplo a seguir:

$$f_M(A) = \begin{pmatrix} P(m|a) \\ P(m|\neg a) \end{pmatrix} \quad \text{(Eq. 4)}$$

Após gerar todos os fatores, inicia-se o processo de somatório:

$$f_{\bar{A}JM}(B|E) = \sum_a f_A(a,B,E) \times f_J(a) \times f_M(a)$$

$$\boldsymbol{f_{\bar{A}JM}(B|E) = f_A(a,B,E) \times f_J(a) \times f_M(a) + f_A(\neg a,B,E) \times f_J(\neg a) \times f_M(\neg a)}$$

(Eq. 5)

Realizando-se o mesmo processo para E obtém-se:

$$f_{\bar{E}\bar{A}JM}(B) = f_E(e) \times f_{\bar{A}JM}(B|Ee) + f_E(\neg e) \times f_{\bar{A}JM}(B|E\neg e) \qquad \text{(Eq. 6)}$$

Por fim, agrupando-se os valores obtidos com o fator de $P(b)$ obtém-se a Equação 7:

$$P(B|j,m) = \alpha f_B(B) \times f_{\bar{E}\bar{A}JM}(B) \qquad \text{(Eq. 7)}$$

Com a equação reduzida, é necessário apenas realizar a multiplicação dos fatores restantes. Entretanto, o processo utilizado não será o de multiplicação de matrizes e sim o processo conhecido por *pointwise product*, que realiza a união de fatores. Por exemplo, para $f_1 \times f_2$ onde $f_1(A,B)$ e $f_2(B,C)$, tem-se $f_3(A,B,C)$ como demonstrado na Tabela 1.

Tabela 1 - Pointwise Product. A, B e C representam variáveis em uma rede Bayesiana, onde V representa um valor verdadeiro para variável e F um valor falso. Fonte: autor.

A	B	$f_1(A,B)$	B	C	$f_2(B,C)$	A	B	C	$f_3(A,B,C)$
V	V	0.3	V	V	0.2	V	V	V	0.3 x 0.2
V	F	0.7	V	F	0.8	V	V	F	0.3 x 0.8
F	V	0.9	F	V	0.6	V	F	V	0.7 x 0.6
F	F	0.1	F	F	0.4	V	F	F	0.7 x 0.4
						F	V	V	0.9 x 0.2
						F	V	F	0.9 x 0.8
						F	F	V	0.1 x 0.6
						F	F	F	0.1 x 0.4

Observando-se a Tabela 1 nota-se que para inferir a probabilidade de ocorrência dos eventos A, B e C derivada de suas devidas dependências é necessário apenas a realização de multiplicações simples, o que reduz o tempo computacional de execução da inferência em grande escala, tornando-o

dominado pelo tamanho do maior fator construído durante a execução do processo para uma determinada rede Bayesiana.

2.1.2.2. Método do Agrupamento

O método de eliminação de variáveis é simples e eficiente para se realizar inferências individuais, entretanto, para se calcular probabilidades posteriores para todas as variáveis em uma rede, ele pode ser menos eficiente. Em uma rede de 'n' elementos, por exemplo, seriam necessárias realizar n consultas sob um custo computacional de tempo na ordem de O(n), que gera um custo total na ordem de $O(n^2)$. Ao se utilizar o método de Agrupamento, também conhecido como *Junction Tree*, O tempo pode ser reduzido para ordem de O(n).

Como o nome sugere, a ideia básica do método é juntar nós individuais da rede de modo a formar nós agrupados. Um exemplo pode ser visto na Figura 2, onde a esquerda observa-se a rede inicial e a direita a rede após a união dos nós *Sprinkler* e *Rain*, formando o nó agrupado *Sprinkler+Rain*. Além da junção gráfica dos nós, ocorre também a união de suas tabelas booleanas, apresentando agora uma combinação de seus valores.

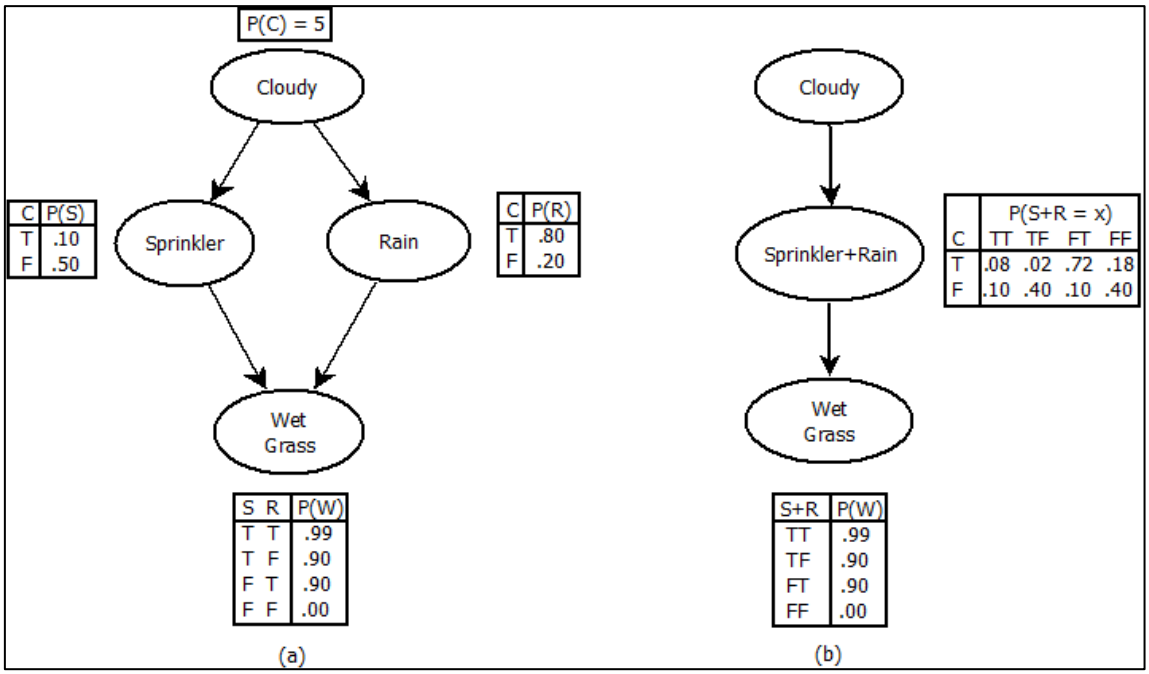


Figura 2 - Exemplo de aplicação do método do agrupamento. (a) Rede Bayesiana inicial. (b) Rede Bayesiana após a realização do agrupamento dos nós *Sprinkler* e *Rain*. Fonte: autor.

Após se realizar o agrupamento de todos os nós possíveis, um algoritmo específico de inferência é aplicado. Esse algoritmo é capaz de calcular probabilidades posteriores para todos os nós na rede em tempo O(n), onde ‘n’ é agora o tamanho da rede modificada.

2.2. Modelagem de emoções

Emoções são consideradas a regra central de nossas vidas, tendo grande impacto na tomada de decisões, ações, memória, atenção, etc. Existem diversas propostas para a modelagem de emoções, de forma a tentarem apresentar uma explicação melhor sobre como estas funcionam. Além disso, estas propostas oferecem o modelo básico para que emoções sejam simuladas em máquina (GRATCH; MARSELHA, 2001).

Porém, realizar a simulação de emoções em máquina não é uma tarefa fácil. Em tarefas onde as emoções exercem um papel fundamental, como processos de tomada de decisão, diversos fatores, tanto sociais quanto fisiológicos, tornam a realização da modelagem e simulação do processo bastante complexa.

Muitas das dificuldades ocorrem pelo fato de que as emoções são o centro das motivações humanas, sendo tanto precursor quanto resultado final em muitos empreendimentos, isto é, as emoções interferem em nosso comportamento, ações, na forma como nos relacionamos com outras pessoas, etc. Por isso, emoções são objeto de estudo desde os tempos mais remotos, com estudos de Lao-Tzu e Sócrates (BERCHT, 2001).

Outro ponto que dificulta o trabalho com emoções é a dificuldade de compreensão do que são e como funcionam. Não existe um consenso sobre a definição de emoções, embora alguns autores sugiram suas versões: Del Nero (1997) afirma que emoção é um processo consciente e que em conjunto com o pensamento e a vontade formam os protagonistas principais para o palco que é a mente. Para Damásio (2000), a emoção é um rótulo que designa um conjunto de fenômenos ou comportamentos. Ele divide as emoções em primárias (medo, alegria, tristeza, raiva, etc.) e as secundárias (ciúme, culpa, orgulho,

etc.). Moffat *et al.* (2000) afirmam que as emoções são funcionais, isto é, elas possuem um valor adaptativo para o organismo, contradizendo a convenção existente de que as emoções não são racionais. Sloman (2001) também conclui que não há uma definição única de emoção, pois esta depende de como se analisa quais são as concepções individuais dos seres humanos ou outros animais em relação ao assunto.

Quando se trata sobre o funcionamento das emoções, devem-se levar em conta duas características: as emoções são processos fisiológicos de difícil mensuração, são coisas que sentimos; emoções são geradas por estímulos, entretanto é impossível afirmar que um mesmo estímulo irá gerar a mesma emoção em dois indivíduos diferentes. Isto se deve a diversos fatores que, em resumo, definem cada pessoa como um ser diferente.

A fim de um melhor entendimento do funcionamento das emoções, foram propostos diversos modelos para a estruturação das emoções, cada uma visando aspectos diferentes do ser humano. Alguns de cunho psicológico, como percepção, sentimentos, experiências, cognição e comportamento (MOFFAT; FRIJDA, 2000) (CAÑAMERO; VAN DE VELDE, 1999) (ORTONY *et al.*, 1988), e outros de cunho fisiológico, como batimentos cardíacos, pressão arterial e sudorese (SLOMAN, 1999) (PICARD, 1997).

Dentre os modelos propostos, destaca-se o proposto por Ortony, Clore e Collins. Conhecido como OCC, o modelo oferece uma estrutura para se descobrir quais emoções podem ser geradas a partir de um determinado evento (ORTONY *et al.*, 1988). Este modelo é descrito em maiores detalhes a seguir.

2.2.1. O Modelo OCC

Em seu livro "*The Cognitive Structure of Emotions*" (ORTONY *et al.*, 1988), Ortony, Clore e Collins propuseram um modelo de emoções capaz de identificar, a partir de estímulos gerados em um ambiente arbitrário, quais emoções seriam geradas, dentro de um conjunto predeterminado de emoções. Este é um dos modelos mais utilizados no ramo da computação, seja para

adicionar emoções a agentes artificiais ou para se trabalhar com a tomada de decisão influenciada pelas emoções.

O modelo se baseia no princípio da diferenciação entre reações de valência positivas e negativas, ou seja, a partir de um estímulo do ambiente, variáveis são atribuídas de forma a determinar se o evento proporciona sentimentos positivos ou negativos para o indivíduo modelado.

No modelo OCC são considerados três geradores de estímulos para emoções: eventos, os quais têm suas consequências analisadas de forma a gerar emoções; agentes, onde se analisam suas ações; e objetos, em que se analisam seus aspectos e propriedades.

Toda emoção gerada no modelo é uma reação a um ou mais aspectos do ambiente. Entretanto, para indivíduos distintos, um determinado estímulo pode gerar emoções diferentes. Esta diferenciação ocorre no modelo a partir da atribuição de um valor positivo ou negativo como reação do indivíduo para uma determinada ocorrência. Este conceito se torna mais claro ao se observar a estrutura do modelo na Figura 3.

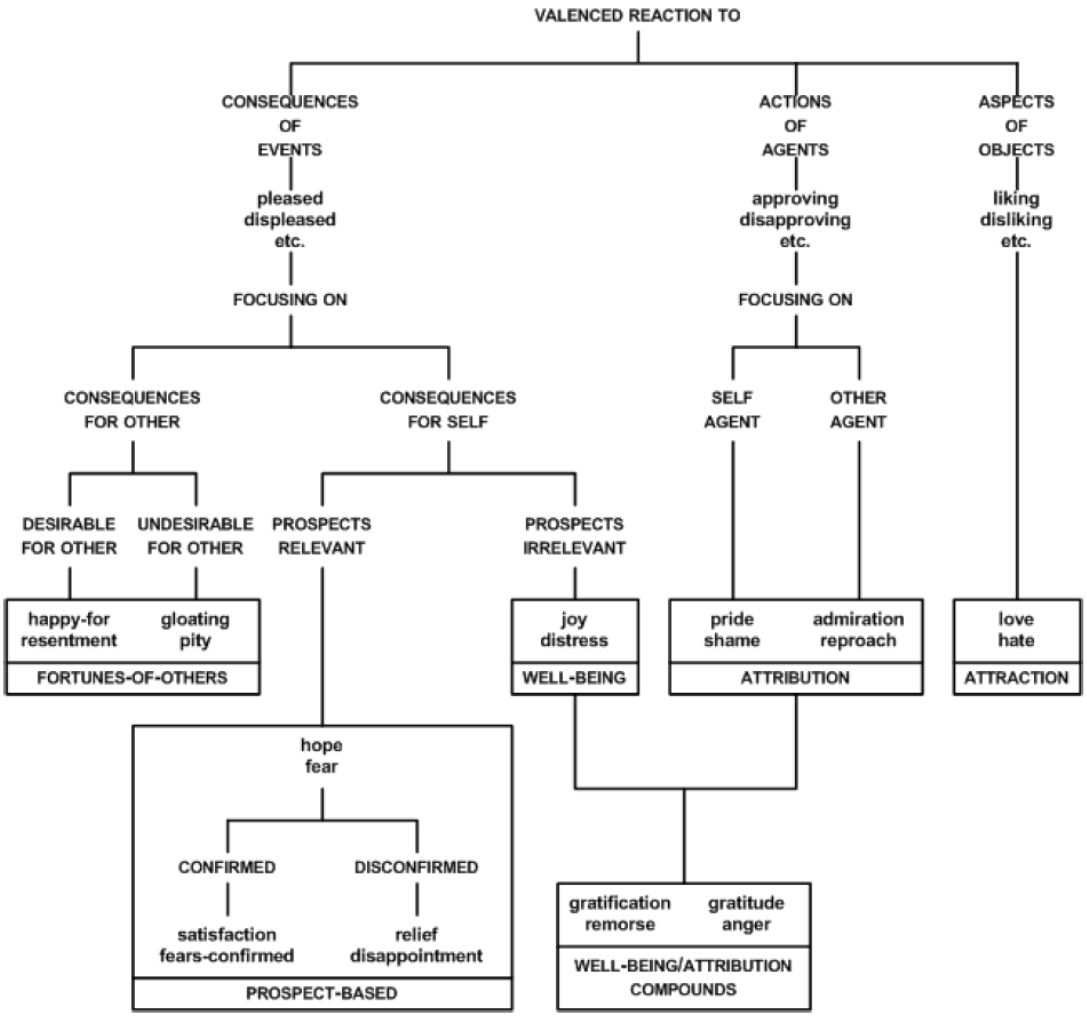


Figura 3 - Estrutura do Modelo OCC (ORTONY *et al.*, 1988).

A estrutura do modelo pode ser dividida em três ramos principais, cada um correspondendo a um tipo de estímulo gerado pelo ambiente. O ramo mais a esquerda refere-se ao desenvolvimento de emoções geradas a partir de eventos, o ramo central das emoções geradas por ações e o ramo a direita refere-se às emoções produzidas a partir de objetos. É importante ressaltar que a estrutura apresenta uma descrição lógica para a geração de emoções e não uma sequência temporal.

A parte inferior do modelo apresenta o conjunto de emoções trabalhadas, as quais totalizam 22 emoções distintas. Para manter a integridade do modelo, neste trabalho será utilizada a nomenclatura inglesa para as emoções.

Observando a Figura 3, notam-se seis diferentes subgrupos de emoções destacados em retângulos. Cada um deles é composto por emoções que dependem dos mesmos tipos de estímulos e compartilham características semelhantes.

Partindo da esquerda para direita, o primeiro subgrupo compreende as emoções geradas a partir de reações a eventos e que consideram um aspecto em comum, a consequência deste evento para outro indivíduo (*consequences for other*).

Este grupo compreende quatro emoções: *happy-for*, é gerada quando ocorre um evento bom tanto para o próprio indivíduo quanto para outro, o qual é alvo da emoção; *resentment*, ocorre quando o evento é desejável para outro agente mas é ruim para o indivíduo analisado; *gloating*, é gerada a partir de um evento bom para o agente analisado porém indesejável para outro indivíduo; *pity*, ocorre a partir de um evento ruim tanto para o agente analisado quanto para o agente alvo da emoção.

A seguir, está o subgrupo formado pelas emoções geradas a partir de eventos e que se referem às consequências futuras deste para o próprio indivíduo. Compreende seis emoções: em um primeiro nível estão *hope*, emoção gerada a partir de um evento considerado bom para o indivíduo, e *fear*, a qual é consequência de um evento ruim para o agente. Em um nível posterior encontram-se emoções derivadas das duas iniciais, diferenciando-se pela confirmação ou não das expectativas futuras do evento. Neste nível estão: *satisfaction*, que ocorre quando *hope* se confirma; *fears-confirmed*, gerada quando *fear* é confirmado; *relief*, que ocorre quando *fear* não se confirma; e *disapointment*, gerada quando *hope* não se confirma.

No próximo subgrupo estão as emoções geradas a partir de evento, em que se avaliam as consequências momentâneas para o indivíduo analisado. Duas emoções fazem parte deste subgrupo: *Joy*, gerada a partir de um evento considerado bom para o agente e *Distress*, gerada pela ocorrência de um evento ruim para o indivíduo.

No subgrupo a seguir estão às emoções geradas a partir de ações ocorridas no ambiente, compreendendo ações realizadas pelo próprio agente quanto por outros. Quatro emoções fazem parte deste subgrupo: *pride*, gerada pela execução de uma boa ação pelo próprio indivíduo; *shame*, que ocorre a

partir de uma ação ruim realizada pelo próprio agente; *admiration*, a qual é gerada pela execução de uma boa ação por outro agente; e *reproach*, gerada a partir da realização de uma ação ruim por outro indivíduo.

Na parte inferior do modelo existe um subgrupo especial, formado por emoções geradas a partir das emoções que compreendem os dois últimos grupos retratados, ou seja, emoções geradas a partir de uma ação e um evento relacionado a esta ação. Quatro emoções compõem este grupo: *gratification*, gerada quando o agente realiza uma boa ação que gera um evento satisfatório para o próprio agente; *remorse*, que ocorre a partir de uma ação ruim do próprio indivíduo que gera um evento também ruim para ele; *gratitude*, gerada quando outro agente realiza uma boa ação que gera um evento satisfatório para o indivíduo analisado; e *anger*, gerada quando um evento ruim para o agente analisado é gerado a partir de uma ação ruim de outro indivíduo.

Por fim, existe um subgrupo que compreende as duas emoções geradas a partir de objetos: *love*, gerada a partir de um objeto atraente para o indivíduo e *hate*, a qual é gerada a partir de um objeto não atraente para o agente.

Um resumo das relações existentes nestes subgrupos pode ser visualizado na Tabela 2, a qual apresenta a relação entre cada emoção e as condições para seu disparo no modelo OCC.

Tabela 2 - Condições para disparo de uma emoção no modelo OCC (ORTONY *et al.*, 1988).

Emoção	Condições
Joy	(contente com) um evento desejável
Distress	(descontente com) um evento indesejável
Happy-for	(contente com) um evento que se presume desejável para outro indivíduo
Pity	(descontente com) um evento que se presume indesejável para outro indivíduo
Gloating	(contente com) um evento que se presume indesejável para outro indivíduo
Resentment	(descontente com) um evento que se presume indesejável para outro indivíduo
Hope	(contente com) a perspectiva de um evento desejável
Fear	(descontente com) a perspectiva de um evento indesejável
Satisfaction	(contente com) a confirmação da perspectiva de um evento desejável
Fears-confirmed	(descontente com) a confirmação da perspectiva de um evento indesejável
Relief	(contente com) a não confirmação da perspectiva de um evento indesejável
Disappointment	(descontente com) a não confirmação da perspectiva de um evento desejável

Pride	(aprovação de) uma ação louvável do próprio indivíduo
Shame	(desaprovação de) uma ação condenável do próprio indivíduo
Admiration	(aprovação de) uma ação louvável de outro indivíduo
Reproach	(desaprovação de) uma ação condenável de outro indivíduo
Gratification	(aprovação de) uma ação louvável do próprio indivíduo e (estar contente com) um evento desejável relacionado
Remorse	(desaprovação de) uma ação condenável do próprio indivíduo e (estar descontente com) um evento indesejável relacionado
Gratitude	(aprovação de) uma ação louvável de outro indivíduo e (estar contente com) um evento desejável relacionado
Anger	(desaprovação de) uma ação condenável de outro indivíduo e (estar descontente com) um evento indesejável relacionado
Love	(gostar de) um objeto atraente
Hate	(não gostar de) um objeto desagradável

2.3. Sistemas Multiagentes

Quando se utilizam Sistemas Multiagentes (SMA), está se utilizando um modelo de Inteligência Artificial Distribuída (IAD) que, ao contrário do modelo de inteligência baseado no comportamento individual humano empregado pela inteligência artificial clássica, baseia-se no comportamento social, com ênfase em ações e interações entre agentes.

Da mesma forma como nas organizações humanas as atividades, muitas vezes, são realizadas por um grupo de pessoas que trabalham de modo cooperativo, onde existem decisões individuais que afetam o grupo, em sistemas multiagentes as pessoas são representadas por agentes artificiais, os quais se relacionam em um ambiente de forma a buscar soluções para problemas de forma cooperativa, compartilhando informações, evitando conflitos e coordenando a execução de atividades (ADAMATTI, 2003).

Por poderem ser utilizados sob vários contextos distintos, não existe um consenso sobe a definição de agente, embora a literatura apresente algumas possibilidades: para Russel *et al.* (2003) “um agente pode perceber seu ambiente e, através de sensores e ações, atuar sobre este. Humanos têm seu corpo para atuar sobre o ambiente. Robôs têm câmeras com vários tipos de sensores para atuar sobre o ambiente. Um agente de software tem um “código” de bits para perceber e atuar sobre este ambiente”.

Já para Ferber *et al.* (1991), agentes são “Uma entidade real ou virtual, capaz de agir num ambiente, de se comunicar com outros agentes; que possui recursos próprios; que é capaz de perceber seu ambiente (de modo limitado); que dispõe uma representação parcial deste ambiente; que possui competência e oferece serviços; que pode eventualmente se reproduzir e cujo comportamento tende a atingir seus objetivos utilizando as competências e os recursos que dispõe levando em conta os resultados de suas funções de percepção e comunicação, bem como suas representações internas”.

De maneira geral, pode-se dizer que um agente artificial em um sistema mutiagentes é uma entidade dotada de certa autonomia e inteligência inserida em um ambiente virtual, sendo capaz de perceber e interagir com os componentes deste ambiente, incluindo os possíveis demais agentes que o habitam.

Os agentes podem ser classificados segundo algumas características: quanto ao foco, em agentes estruturais e agentes comportamentais; quanto à atuação, em agentes isolados e agentes sociais; quanto ao ambiente, em agentes de *desktop* e agentes de internet; e quanto à cognição, em agentes cognitivos e agentes reativos (REZENDE, 2001).

Esta última classificação é a mais utilizada e, também por isso, a mais importante. Agentes reativos funcionam de acordo com um modelo estímulo-resposta, não possuem uma representação explícita de seu ambiente, memória das ações executadas no passado e nem previsão das ações a serem executadas no futuro. Eles apenas reagem às modificações do ambiente.

Já agentes cognitivos possuem uma representação explícita de seu ambiente, de outros agentes e deles mesmos. São baseados nos modelos humanos, possuem raciocínio e capacidade de planejamento de suas ações.

Sendo a interação entre os agentes a principal característica dos SMA, esta apresenta algumas características, dentre as quais se destaca a interferência social, que é a influência que as ações de um agente provocam sobre a tentativa de outro de alcançar seus objetivos. A partir desta característica dois ou mais agentes podem desenvolver uma relação de autonomia, delegação, adoção, compromisso ou cooperação, que é negociada a partir de definições existentes nos protocolos de iteração do sistema multiagentes vigente. Para alguns tipos de relação, uma coordenação entre as

ações executadas pelos agentes se faz necessária (ALVAREZ; SICHMAN, 1997).

Assim como os agentes, os sistemas multiagentes podem ser classificados de acordo com algumas características: quanto à perspectiva, de simulação ou resolução social; quanto à abertura, aberto ou não; quanto à granularidade, baixa ou alta granularidade; quanto à composição, homogênea ou heterogênea; e quanto à interação, envolvendo neutralismo, competição, amensalismo, parasitismo, predação, comensalismo, proto-cooperação ou simbiose (REZENDE, 2001).

2.3.1. Simulação e Sistemas Multiagentes

Simulações são utilizadas com grande sucesso como elemento auxiliar na tomada de decisões, principalmente no planejamento a médio e longo prazo e em situações que envolvem custos e riscos elevados. Os modelos de simulação são muito eficazes e versáteis no estudo dos mais diferentes problemas. Seu emprego permite o exame de detalhes específicos com grande precisão (REBONATTO, 2000).

Por envolverem uma grande quantidade de dados e exigirem um grande processamento sobre eles, os processos de simulação são empregados quase que exclusivamente através de um meio computacional. Sua utilização significa construir programas de computador (software) que ‘representem’ o sistema do mundo real em questão e ‘imitem’ seu funcionamento (REBONATTO, 2000).

Um processo de simulação segue três etapas básicas: etapa de modelagem, onde se constrói o modelo do fenômeno a ser estudado; etapa de experimento, na qual são aplicadas variações sobre o modelo construído, alterando parâmetros que influam no processo de resolução; e a etapa de validação, onde se comparam dados experimentais obtidos com o modelo e a realidade, ou seja, é a análise dos resultados.

Um modelo demonstrando as etapas de um processo de simulação pode ser visualizado na Figura 4.

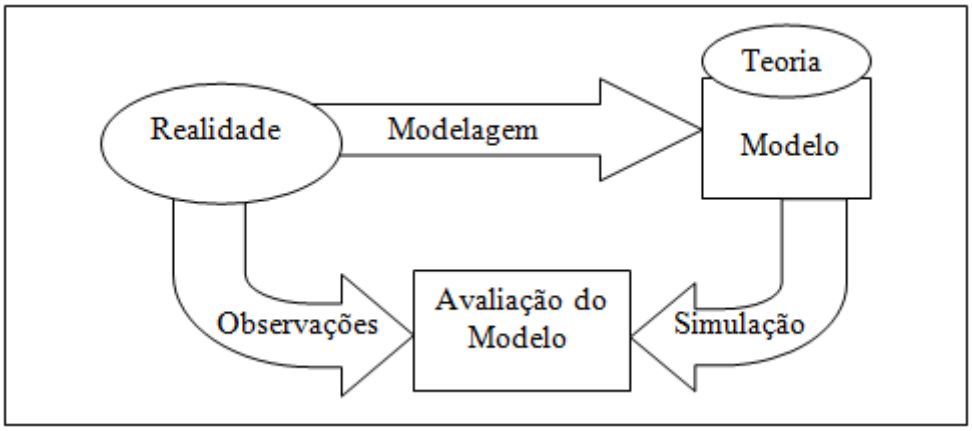


Figura 4 - Etapas do processo de simulação (FROZZA, 1997).

Na maior parte das vezes, uma aplicação desenvolvida sobre um SMA tem o objetivo de simular alguma situação da realidade. Em um sistema multiagentes, o fenômeno real é decomposto em um conjunto de elementos e em suas interações. Para cada elemento é modelado como um agente, resultando em um modelo geral onde os agentes interagem entre si e com o ambiente.

O processo de modelagem de um fenômeno real em um sistema multiagentes também segue algumas etapas: primeiramente, se decompõe o fenômeno a ser modelado em um conjunto de elementos autônomos; a seguir, modela-se cada um dos elementos como um agente, definindo seu conhecimento, funções comportamento e modos de iteração; o passo seguinte é definir o ambiente no qual os agentes criados habitarão; devem-se definir, também, quais agentes possuem a capacidade de ação e comunicação (FROZZA, 1997).

Para Russel *et al. (*2003), a modelagem de sistemas multiagentes e sua simulação exigem algumas características: os agentes devem ser autônomos; o comportamento dos agentes deve ser representado em alto nível de abstração; agentes devem ser flexíveis, tendo características de comportamento pró-ativo e reativo; agentes podem executar tarefas que exijam desempenho de tempo real; agentes se encaixam em aplicações distribuídas; agentes devem possuir habilidade de trabalhar cooperativamente.

Além disso, alguns cuidados devem ser tomados para se desenvolver uma boa simulação em um SMA. Em primeiro lugar, os agentes devem ser

claramente identificados, bem como suas atividades e comportamentos dentro do ambiente. Para determinar as atividades, devem-se observar os objetivos específicos que os agentes possuem.

Deve-se lembrar de que agentes possuem um comportamento sem padrões, assim, agentes mais simples são implementados de forma mais eficiente em uma linguagem orientada a agentes. Quanto mais simples a estrutura de cada agente, melhor funciona o sistema, tendo em vista que o foco da modelagem deve ser o comportamento e as iterações dos agentes, e não suas habilidades internas.

Por fim, deve haver uma descrição do sistema do agente e do ambiente em que o agente se posiciona, pois isso facilita a implementação e a solução de possíveis erros (ADAMATTI, 2003).

3. Trabalhos Relacionados

Ao se observar o capitulo anterior, nota-se o grande potencial que cada uma das técnicas, redes Bayesianas, sistemas multiagentes e modelos estruturais de emoções, possuem individualmente. Entretanto, para obter aplicações mais realísticas, se faz necessário a união de suas capacidades em prol de realizar determinado objetivo.

Os modelos de emoções, por exemplo, são modelos teóricos que necessitam de um meio experimental para que tenham uma utilidade prática. Os sistemas multiagentes e sua capacidade de criar simulações da realidade podem oferecer o meio perfeito para a aplicação de um modelo de emoções, de forma que estas possam ser agregadas a um ambiente computacional propiciando uma simulação ainda mais próxima da realidade.

Em Conati *et al.* (2010) utilizam-se do expediente de unir diferentes técnicas de forma a alcançar seu objetivo, criar um modelo probabilístico que permita a um agente inteligente perceber as emoções do usuário durante a execução de um jogo educacional.

Neste trabalho, os autores utilizam tanto o modelo OCC como um sistema multiagentes e uma rede Bayesiana. Partindo dos eventos geradores de emoções apresentados pelo modelo OCC e a observação por meio de sensores de efeitos físicos provocados por elas, os autores desenvolveram uma rede Bayesiana dinâmica, ou seja, uma rede Bayesiana que apresenta diferentes instâncias durante o tempo, as quais se relacionam entre si, capaz de alimentar o modelo OCC com as informações sobre a origem da emoção, eventos, agentes e/ou objetos, e seu efeito para os objetivos do usuário, positivo ou negativo, de forma a permitir, através da análise do modelo, a descoberta das emoções as quais o usuário está sentindo em um determinado momento.

Em seguida, a informação sobre quais emoções o usuário está experimentando é utilizada por um agente inteligente, que interage com o usuário durante a execução do jogo educacional. A capacidade de observar as emoções do usuário fornece ao agente a competência de racionalizar o porquê de uma ação ser realizada pelo usuário, possibilitando uma resposta mais

adequada a ela. Esta interação mais próxima às necessidades do usuário melhora a relação de confiança deste com o sistema.

Assim, se, por exemplo, uma emoção negativa gerada por uma ação do próprio usuário é reconhecida, o agente pode oferecer ‘dicas’ de forma a melhorar o desempenho do usuário no jogo. Da mesma forma, caso essa emoção negativa tenha sido gerada a partir de uma ação anterior do agente, este pode realizar ações de forma a tentar ‘fazer as pazes’ com o usuário.

Este trabalho demonstra a importância de se buscar novas formas de integrar ferramentas para que se possam obter novas funcionalidades em um sistema computacional. Como as emoções são um importante fator na tomada de decisão das pessoas, permitir que um sistema computacional possa de alguma maneira detectá-las, possibilitando que novas formas de interação entre os usuários e os mais diversos sistemas sejam desenvolvidas.

Um trabalho semelhante foi realizado por Yoon *et al.* (2010), onde os autores propõem um agente inteligente usado em smartphones, capaz de moldar seu comportamento de acordo com as necessidades do usuário.

O comportamento do agente baseia-se em três módulos: o de percepção, em que informações sobre o usuário são lidas a partir do smartphone, como contatos, marcas em calendários, registro de chamadas, etc., de forma a se obter uma visão aproximada do estado emocional do usuário. Embora o modelo não obtenha emoções específicas, ele recupera níveis de valência e excitação a partir das informações.

O segundo módulo, de emoções, é uma simplificação do modelo OCC, compreendendo 14 emoções e é utilizado para gerar emoções para o agente inteligente. Mesmo modificado, o modelo ainda baseia-se nas três fontes básicas para geração de emoções do modelo original, eventos, agentes e objetos.

Por fim, no módulo de motivações estão os objetivos a serem realizados pelo agente. Se a bateria do smartphone estiver perto de acabar, por exemplo, o objetivo de avisar o usuário sobre a situação do aparelho é ativado.

A informação gerada a partir dos três módulos é utilizada pelo agente para decidir qual das ações previamente definidas deve ser realizada. Esta avaliação ocorre através da rede de comportamento proposta por Maes (Maes, 1989), a qual relaciona comportamento, objetivos e ambiente através de

ligações associadas a níveis de ativação. Após a avaliação das informações obtidas a partir dos módulos em um determinado momento, observa-se o maior nível de ativação e realiza-se o comportamento apontado por ele.

Apesar de ser mais simples que o modelo apresentado por Conati *et al.* (2010) anteriormente apresentado, o modelo de Yoon *et al.* (2010) apresenta algumas características interessantes, sobre as quais se destaca a maior autonomia de seu agente inteligente, tendo em vista que seu comportamento não é definido tomando como base apenas as necessidades do usuário, mas também utiliza suas próprias necessidades e objetivos.

De modo geral, quando se deseja modelar emoções em sistemas computacionais, o modelo OCC é utilizado. Entretanto, para algumas aplicações a utilização de outros modelos pode ser benéfica, assim como é visto no trabalho de Sabourin *et al.* (2011), no qual os autores utilizam uma rede Bayesiana combinada com um modelo de emoções para tentar prever as emoções de uma pessoa em um processo de aprendizado.

Os autores argumentam que o modelo OCC não é o mais eficaz para simular emoções para a situação específica trabalhada, pelo fato de não apresentar em seu conjunto de emoções algumas consideradas importantes em processos de aprendizado.

Assim, o modelo de emoções escolhido foi o desenvolvido por Elliot *et al.* (2007), que oferece um conjunto de emoções específicas, baseadas em um conjunto de objetivos existentes em um processo de aprendizado. Neste modelo, as emoções são desenvolvidas a partir de valências de níveis de aprendizado e desempenho durante este processo.

O modelo trabalha com um conjunto de sete emoções, foco, curiosidade, frustração, excitação, confusão, tédio e ansiedade. Os autores submeteram 296 alunos da oitava série de uma escola norte americana a um jogo baseado em aprendizado chamado “*Crystal Island*”, no qual a cada sete minutos os usuários eram perguntados sobre qual das emoções anteriormente citadas eles se enquadravam no momento.

Possuindo este banco de dados, uma rede Bayesiana foi desenvolvida para se tentar prever as emoções desenvolvidas por usuários submetidos a processos de aprendizado. A rede baseia-se em três tipos de variáveis: de atributos pessoais, como consciência, afabilidade e abertura a novas ideias; de

observação do ambiente, baseadas em eventos do ambiente, no caso o jogo "*Crystal Island*"; e de avaliação, as quais são externas ao ambiente de aprendizado, e estão contidas no modelo de emoções de aprendizado.

A rede foi construída seguindo a estrutura do modelo de emoções de aprendizado utilizado e seus parâmetros foram desenvolvidos utilizando-se o algoritmo de aprendizado em redes Bayesianas *Expectation Maximization* (EM). Em seguida, uma segunda versão da rede foi criada, aproveitando-se a estrutura da rede e aplicando-se a teoria de redes Bayesianas dinâmicas, a qual adiciona o fator tempo em sua estrutura.

Por fim, os resultados da projeção gerada pelas redes foram comparados aos da experimentação realizada com os estudantes, revelando um grau de acerto razoável para as redes, nas quais a rede dinâmica obteve melhores resultados.

As emoções não influenciam apenas diretamente as ações dos agentes, como pode ser visto no trabalho de Bitencourt *et al.* (2013) onde é apresentado um modelo de confiança para agentes, o *TrustE,* que difere dos modelos mais utilizados por considerar emoções em sua deliberação.

Em um sistema multiagentes aberto, novos agentes podem juntar-se ao ambiente a qualquer momento. Esta característica torna difícil para os agentes que estão há mais tempo no sistema identificarem se os novos agentes são confiáveis, devido a falta de informação sobre eles.

Este problema de falta de informação é normalmente contornado através da utilização de modelos de confiança e reputação, numéricos e que, de modo geral, utilizam métricas de relevância de informação obtidas a partir de ações e comportamentos de outros agentes para determinar o quanto são confiáveis.

Embora bastante utilizados, este modelos acabam por dissociar a avaliação da confiança do agente de sua individualidade e do ambiente em que está inserido. Deste modo, o modelo TrustE utiliza para avaliação de confiança de agente informações diretamente relacionadas ao contexto ao qual o gente está inserido. Estas informações induzem sensações ou emoções que permitem ao agente associar avaliações qualitativas ou subjetivas a suas avaliações quantitativas ou racionais, dependendo do resultado de introspecção sobre situações vividas pelo agente.

Baseando-se no modelo *Regret* de avaliação de confiança e reputação (SABATER; SIERRA, 2003), o *TrustE* propõe a adição de emoções, determinadas a partir do modelo OCC, aos cálculos realizados neste modelo, utilizando valores de reputação social e reputação individual para obter um valor final de o quanto um agente é confiável.

Existem diversos outros trabalhos na área de simulação computacional de emoções. Uma compilação de alguns trabalhos foi realizada em Jaques *et al.* (2005), onde os autores apresentam uma breve explanação sobre emoções e sua simulação e em seguida listam uma série de trabalhos na área, demonstrando seus conceitos e funcionamento básico.

Observando os trabalhos contidos nesta compilação como os anteriormente apresentados, observa-se a grande gama de aplicações para os modelos de emoções, bem como a sua importância para o entendimento e simulação do comportamento humano.

A Tabela 3 Apresenta uma comparação resumida entre as técnicas utilizadas em cada um dos trabalhos apresentados e o trabalho proposto. As técnicas consideradas são as utilizadas neste trabalho: Redes Bayesianas, Modelagem de Emoções e Sistemas Multiagentes.

Tabela 3 - Tabela comparativa entre os trabalhos apresentados e o proposto. Fonte: autor.

Trabalho	Redes Bayesianas	Modelo de emoções	Sistemas Multiagentes
Conati *et al.* (2010)	Sim	OCC	Sim
Yoon *et al.* (2010)	Não	OCC Simplificado	Sim
Sabourin *et al.* (2011)	Sim	Modelo Criado por Elliot et al. (2007).	Não
Bitencourt *et al.* (2013)	Não	OCC	Sim
Modelo Proposto	Sim	OCC	Ambiente para testes

4. A Rede Bayesiana de Emoções

Este trabalho propõe a simulação de emoções em ambientes mutiagentes através da criação de uma rede Bayesiana capaz de traduzir estímulos gerados neste ambiente em emoções, isto é, apresenta-se uma maneira de simular emoções em meios computacionais aplicando-a a um ambiente multiagentes de forma a avaliar sua efetividade.

Quando se trata de emoções, alguns modelos de seu funcionamento já foram desenvolvidos, dentre os quais se destaca o modelo OCC, bastante abrangente e que possui uma estrutura de simples tradução computacional, características que propiciaram sua escolha como modelo base para rede Bayesiana de emoções proposta.

O modelo OCC por si só é um modelo que não considera a imprevisibilidade humana, determinando sempre o mesmo resultado emocional a partir de certo evento ou ação. Desta forma, propõe-se a construção de uma rede Bayesiana que possua uma estrutura equivalente ao modelo OCC de emoções e que possa, através de suas características básicas, adicionar a imprevisibilidade necessária ao modelo de simulação.

A utilização de redes Bayesianas para adição de imprevisibilidade em modelos de emoções já foi utilizada em diversos outros trabalhos, como os apresentados no Capitulo 3. Entretanto, a combinação destas técnicas ocorre, de maneira geral, de forma diferente à proposta neste trabalho, oferecendo um pré-processamento das informações contidas no ambiente de forma a propiciar a alimentação necessária ao modelo OCC.

Já o modelo proposto, utiliza uma rede Bayesiana que engloba tanto as funções proposta nestes trabalhos como também as funções desempenhadas pelo modelo OCC nestes, ou seja, a rede Bayesiana proposta é capaz de transportar os estímulos do ambiente diretamente para o modelo OCC, oferecendo de maneira direta as emoções em sua saída.

Dentre os trabalhos apresentados no Capítulo 3, o que mais se assemelha a proposta dessa dissertação é o de Conati *et al. (*2010). Um fluxograma de funcionamento deste trabalho pode ser observado na Figura 5a,

enquanto a Figura 5b apresenta o fluxograma de funcionamento para o modelo proposto.

Enquanto no trabalho de Conati *et al.* (2010), a rede bayesiana é utilizada para realizar um pré-processamento para os estímulos do ambiente de forma a propiciar alimentação para o modelo OCC clássico, estático e definido por uma sequência de desvios condicionais, o modelo proposto apresenta uma rede Bayesiana que redefine o modelo OCC, possuindo uma estrutura semelhante ao original, mas acrescentando uma série de características que o transformam em um modelo dinâmico, que apresenta diferentes resultados emocionais para os mesmos estímulos do ambiente em diferentes momentos.

Estas características propiciam, entre outras possibilidades, o trabalho com perfis comportamentais, onde dependendo das características do indivíduo simulado, a rede pode ser inicializada com valores diferentes, fazendo com que apresente emoções diferentes as de um indivíduo com outra inicialização, para as mesmas circunstâncias ambientais.

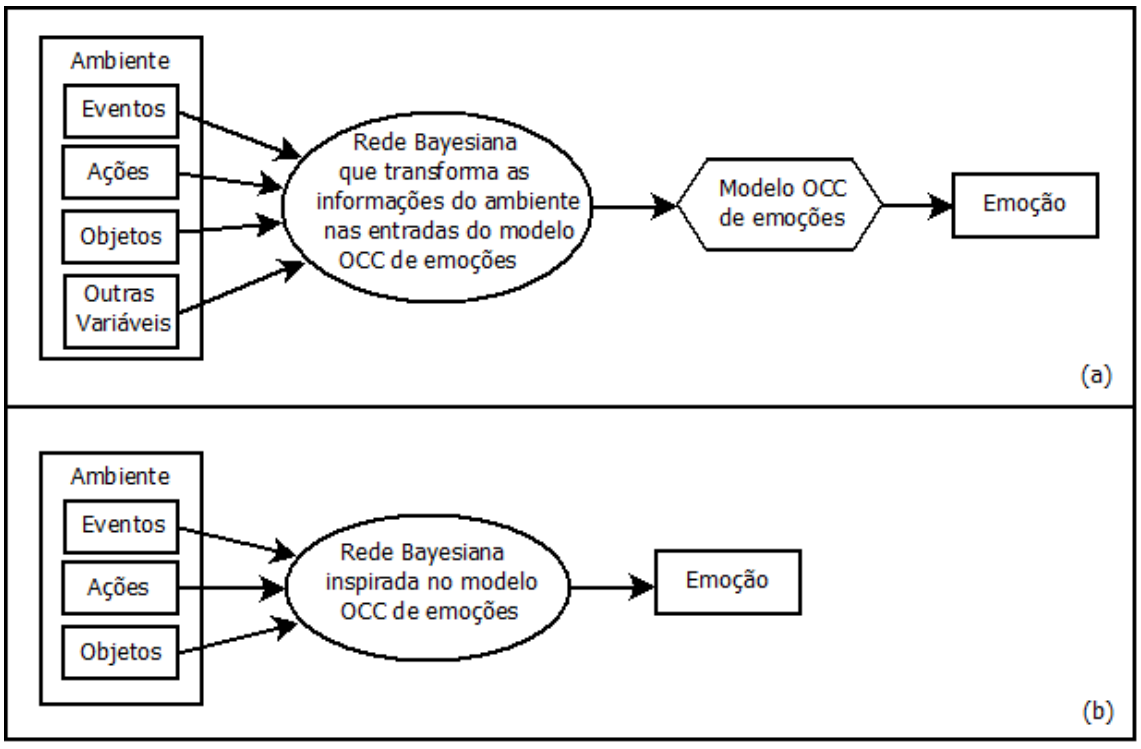


Figura 5 - Comparação entre as diferentes combinações entre redes Bayesianas e o modelo OCC. (a) Método mais utilizado onde a rede Bayesiana processa as informações do ambiente e cria as informações de entrada do modelo OCC. (b) Modelo proposto, onde o modelo OCC é substituído por uma rede Bayesiana de estrutura inspirada no modelo de emoções. Fonte: autor.

Para oferecer ao modelo as condições necessárias para seu funcionamento e validação é necessário aplicá-lo a um ambiente capaz de criar situações próximas à realidade humana, onde estímulos são gerados em um ambiente resultando em emoções nos indivíduos que o habitam. Assim, considerando as capacidades sociais e individuais dos agentes, propõe-se a aplicação da rede Bayesiana como uma estrutura capaz de gerar emoções para agentes em um ambiente multiagentes.

Para demonstrar a funcionalidade do modelo proposto, apresenta-se a simulação de diversas instâncias de um ambiente multiagentes exemplo, oferecendo dados suficientes para que se possa comparar comportamentos e efeitos da rede sobre os agentes que compõem o ambiente em diferentes situações.

4.1. Modelagem da Rede

Baseando-se na estrutura do modelo OCC (Figura 3), a rede Bayesiana de emoções tem por objetivo transformar estímulos de um ambiente em emoções de um individuo. Para sua construção e análise foi utilizado um software livre auxiliar chamado JavaBayes.

Desenvolvido por Cozman (2001), o JavaBayes permite a construção, visualização gráfica e análise de redes Bayesianas através de uma interface simples e de fácil acesso. O software possibilita a construção de redes com qualquer estrutura, com um numero indefinido de nós, arestas e variáveis. Além disso, oferece diversas funções, como, por exemplo, a seleção do algoritmo para o cálculo de probabilidades entre os métodos agrupamento e eliminação de variáveis e a possibilidade de exibir as probabilidades de todas as variáveis da rede a partir da observação de um de seus nós.

É importante ressaltar que além de ser um excelente software para o trabalho com redes Bayesianas, o JavaBayes, como o nome indica, foi desenvolvido na linguagem Java, sobre o regime de código aberto, possibilitando sua utilização sob a forma de API (*Application Programming*

Interface) no desenvolvimento de outros softwares. Desta forma, possibilita que suas classes e métodos sejam importados para uso em outros programas. Este é o processo utilizado neste trabalho e será discutido em maiores detalhes nas seções subsequentes.

Como o modelo OCC de emoções é constituído basicamente de uma sequencia de desvios condicionais que possibilitam a determinação de qual emoção é ativada a partir de determinadas condições, a rede Bayesiana baseada em sua estrutura foi construída traduzindo cada um destes desvios em um conjunto de nós, variáveis e arestas, onde cada nó representa um desvio, cada variável representa um dos estados que satisfazem o desvio e cada aresta liga cada desvio a próxima condição a ser analisada.

O resultado desta tradução é uma rede constituída de 34 nós e 49 arestas que pode ser visualizada na Figura 6.

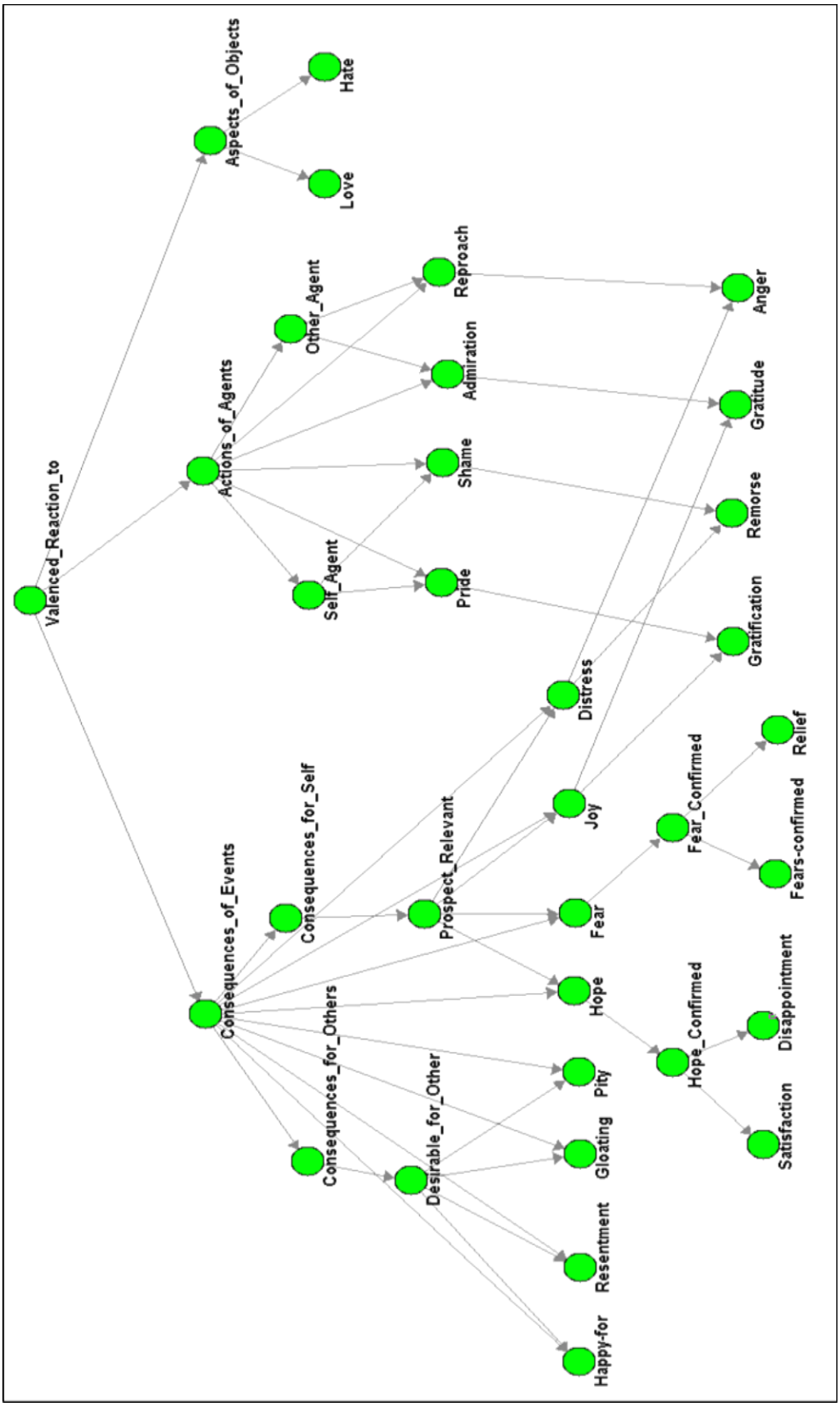


Figura 6 - Rede Bayesiana desenvolvida. Inspirada no modelo OCC de emoções, cada nó representa um desvio condicional no modelo original e as arestas representam as dependências pai-filho entre estes nós. Fonte: autor.

Ao observar a estrutura da rede é possível observar que, assim com ocorre com o modelo OCC de emoções, ela pode ser dividida em três áreas distintas de acordo com a fonte de estimulo do ambiente. Os nós na área à esquerda representam os estímulos gerados a partir de eventos ocorridos no ambiente, a área central compreende os nós relativos a estímulos gerados a partir das ações de outros indivíduos, enquanto a menor área, localizada a direita, refere-se às emoções geradas a partir do contado do individuo com diferentes objetos.

Na parte inferior da rede encontram-se os nós que representam cada uma das 22 emoções modeladas pelo sistema: *Happy-for, Resentment, Gloating, Pity, Hope, Fear, Joy, Distress, Satisfaction, Disappointment, Fears-confirmed, Relief, Pride, Shame, Admiration, Reproach, Love, Hate Gratification, Remorse, Gratitude* e *Anger.*

Como é característico das redes Bayesianas, cada nó da rede depende diretamente de seus nós pais, indicados pelas arestas, construídas na direção pai-filho, sendo permitido tanto um nó pai ter múltiplos nós filhos, como também um nó filho possuir múltiplos nós pai. Seguindo esta dinâmica, bem como as relações existentes entre emoções e estímulos existentes no modelo OCC, que podem ser vistos na Tabela 2, foram definidas tanto as variáveis de cada nó como também as probabilidades iniciais da rede.

Cada nó possui uma variável que pode assumir dois estados, sendo em geral, inclusive para os nós que compreendem as emoções, estes estados "*true"* e *"false"*, onde "*true"* representa um estado ativo, a confirmação de uma condição ou a efetividade de uma emoção; enquanto *"false"* significa o contrário. A exceção a esta regra são os três nós superiores a cada uma das áreas anteriormente citadas: *Consequences_of_Events, Actions_of_Agents* e *Aspects_of_Objects*.

Para estes três nós, suas variáveis possuem estados e funções diferentes dos demais: a variável do nó *Consequences_of_Events* pode assumir os estados *"Pleased"* e *"Displeased"* que, respectivamente, indicam se ocorreu um evento desejável ou indesejável para o indivíduo no ambiente.

Já no nó *Actions_of_Agents,* a variável pode assumir os valores *"Approving"* e *"Disapproving"*, que de maneira semelhante aos estados do nó

Consequences_of_Events indicam se o individuo esta de acordo ou não com a ação de um individuo que ativou o nó.

Por fim, a variável do nó *Aspects_of_Objects* pode assumir os estados *“Liking”* e *“Disliking”* que indicam se o indivíduo gosta ou não, respectivamente, do objeto que estimulou a rede Bayesiana.

Após definir os estados possíveis para as variáveis de cada um dos nós, foi necessário definir as probabilidades de ativação de cada um dos estados seguindo as dependências entre nós pais e filhos e respeitando as condições apresentadas na Tabela 2.

A rede apresenta duas situações, nós que possuem um ou dois pais. Para o primeiro caso, os estados da variável ainda podem possuir dependência fraca ou forte em relação ao nó pai. A dependência forte ocorre quando os estados do nó filho variam de acordo com as mudanças de estado do nó pai. Já a dependência fraca é a que ocorre quando, embora dois nós estejam estruturados em uma relação pai-filho, sua variação de estados não possui relação direta.

Assim, quando a dependência é fraca, suas probabilidades foram definidas como 50% de ocorrer qualquer um dos estados, seja qual for o estado do nó pai. Um exemplo desta situação é o nó *Consequences_for_Others* que tem seus estados sem relação direta aos estados de seu nó pai, *Consequenses_of_Events*. Este exemplo pode ser visualizado na Figura 7(a).

Um exemplo de nó que possui dependência forte com relação a seu nó pai é o nó *Satisfaction*, que possui seus estados dependentes diretamente dos estados de seu nó pai, *Hope_Confirmed*. Neste caso, um dos estados do pai faz com que o filho tenha 95% de chance de ter um determinado estado, enquanto seu estado contrário faz com que seu nó filho possua também 95% de chances de ter o estado contrário. Este exemplo pode ser visualizado na Figura 7(b).

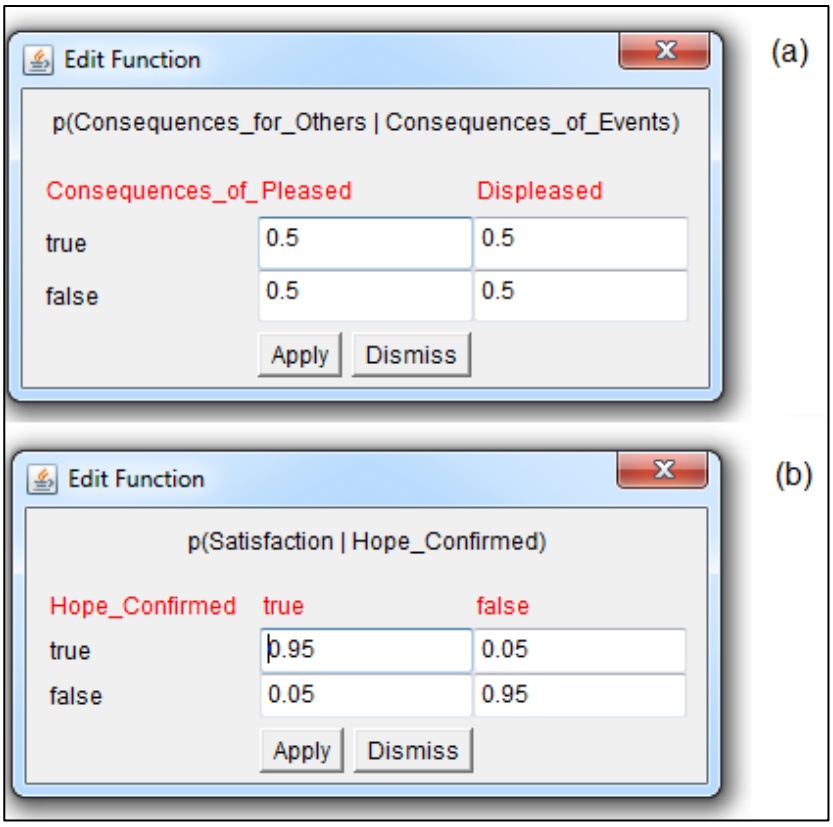


Figura 7 – Exemplo da determinação de probabilidades para estados de nós com um pai. (a) Nó Consequences_for_Others não depende dos estados de Consequences_of_Events, *"Pleased"* e *"Displeased"*. (b) Nó Stisfaction que depende dos estados de Hope_Confirmed, quando o pai é verdadeiro o filho tem 95% de chances de também ser, assim como o contrário. Fonte: autor.

Quando um nó possui dois nós pais ele sempre depende diretamente dos estados destes, sendo definido que o filho possui 95% de possuir um dos seus estados quando um determinado estado para cada um de seus pais. Quando os estados dos pais se invertem, o nó filho tem 95% de chance de também ter seu estado invertido. No caso de um dos pais terem seu estado invertido, o nó filho tem 50% de chance de ter cada um dos seus estados. Um exemplo deste tipo de nó filho é o nó Joy, o qual tem suas probabilidades demonstradas na Figura 8.

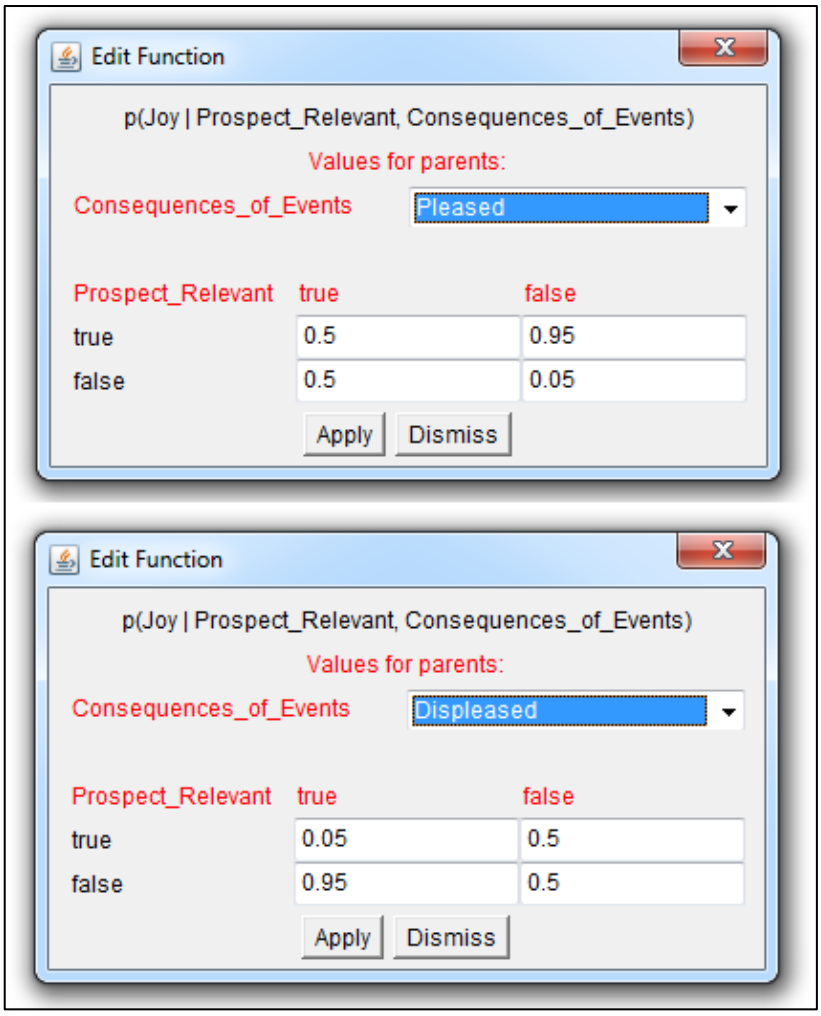


Figura 8 – Demonstração das probabilidades definidas para o nó Joy de acordo com os possíveis estados de seus nós pais, Consequences_of_Events e Prospect_Relevant. Fonte: autor.

A união de todas as probabilidades criadas permite que a rede em seu estado inicial, sem observações, possua 50% de chance de todas suas variáveis assumirem cada um de seus estados. No ANEXO A encontra-se a descrição da rede fornecida pelo software JavaBayes a qual descreve todos os nós da rede e suas correspondentes probabilidades.

4.2. O Ambiente Multiagentes

Jason foi o ambiente multiagentes escolhido. Ele é desenvolvido na linguagem Java e possibilita de forma simples a execução de simulações baseadas em sistemas multiagentes (BORDINI *et al.*, 2007).

O software trabalha com um ambiente baseado em um modelo de cognição fundamentado em três principais atitudes mentais que são as crenças, os desejos, e as intenções (abreviadas por BDI, *beliefs, desires* e *intentions*) (BORDINI *et al.*, 2007). Um esquema demonstrando o funcionamento de uma arquitetura BDI pode ser visualizado na Figura 9.

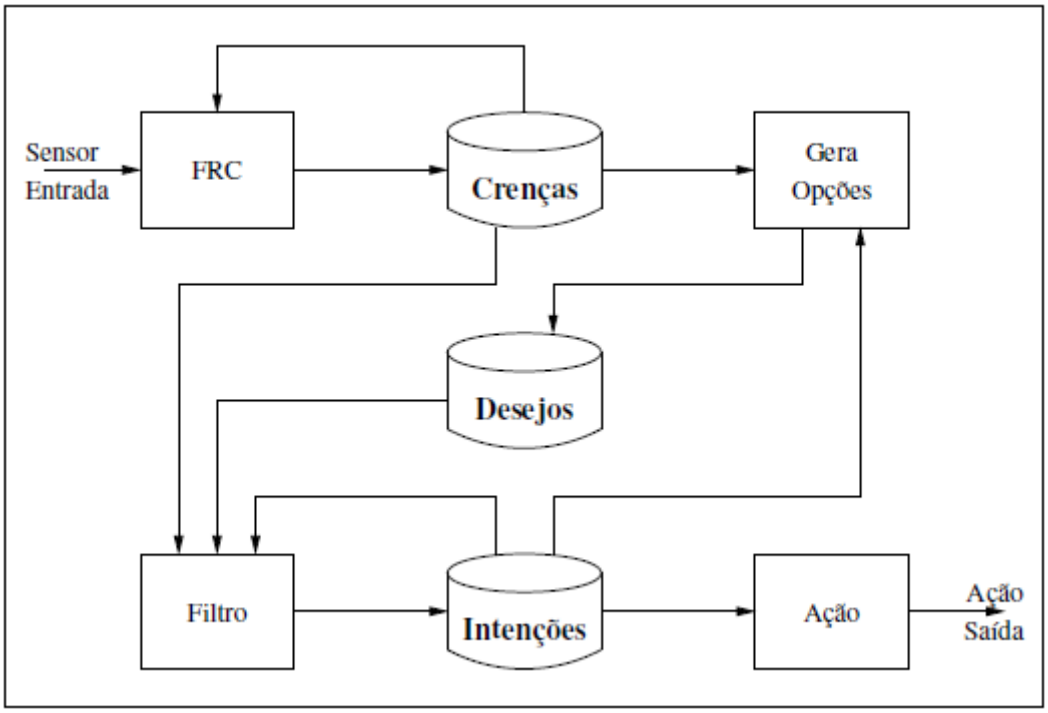


Figura 9 – Arquitetura BDI genérica (WOOLDRIDGE, 1999).

Em resumo, na arquitetura BDI as crenças representam aquilo que o agente sabe sobre o estado do ambiente e os agentes naquele ambiente (inclusive sobre si mesmo). Os desejos representam estados do mundo que o agente quer atingir. Já as intenções representam sequências de ações específicas que um agente se compromete a executar para atingir determinados objetivos.

A função de revisão de crenças (FRC) tem como entradas as informações sensoriais do agente e tem a função de atualizar suas crenças para deixá-las de acordo com o estado atual do ambiente. A função gera opções verifica os estados possíveis de se atingir no momento e avalia sua relevância para as intenções do agente.

Uma vez que o conhecimento e a motivação do agente foram atualizados por estas funções é necessário que se determine qual caminho o agente vai seguir de forma a tentar alcançar seus objetivos. Para isso, a função filtro atualiza o conjunto de intenções do agente de acordo com as crenças e desejos já atualizados, bem como nade acordo com as intenções já existentes. Este processo é conhecido como deliberação. Baseado nas novas intenções a função ação determina a ação a ser tomada (BORDINI *et al.*, 2007).

Para facilitar a programação, Jason utiliza a linguagem de programação AgentSpeak(L), a qual foi projetada para programação de agentes BDI na forma de sistemas de planejamento reativos, os quais são sistemas que estão em constante execução, reagindo a eventos que acontecem no ambiente em que estão situados através da execução de planos que se encontram em uma biblioteca de planos parcialmente instanciados.

Um agente definido por esta linguagem corresponde a uma especificação de um conjunto de crenças que formarão a base de crenças inicial, bem como um conjunto de planos. Um átomo de crença é um predicado de primeira ordem na notação lógica usual e literais de crença são átomos de crenças ou suas negações. A base de crenças de um agente é uma coleção de átomos de crenças (BORDINI *et al.*, 2007).

A linguagem AgentSpeak(L) trabalha com dois tipos de objetivos para os agentes: objetivos de realização (*achievement goals*) e objetivos de teste (*test goals*). Objetivos são predicados semelhantes as crenças, mas que possuem operadores fixados “!” e “?”, para os objetivos de realização e teste respectivamente.

Objetivos de realização expressam que o agente quer alcançar um estado no ambiente onde o predicado associado ao objetivo é verdadeiro. Já um objetivo de teste retorna a unificação do predicado de teste com uma crença do agente, ou falha caso não seja possível à unificação com nenhuma crença do agente. Um evento é dito ativador (*triggering event*) quando define quais eventos podem iniciar a execução de um plano. Um evento pode ser tanto interno, quando gerado pela execução de um plano em que um sub objetivo precisa ser alcançado, quanto externo, quando gerado pelas atualizações de crenças que resultam da percepção do ambiente. Eventos ativadores estão relacionados à adição e remoção de atitudes mentais (crenças

ou objetivos), as quais são representadas pelos operadores prefixados (‘+’) e (‘-’).

Planos fazem referência a ações básicas que um agente é capaz de executar em seu ambiente. Essas ações são definidas por predicados com símbolos predicativos especiais (chamados símbolos de ação) usados para distinguir ações de outros predicados. Um plano é formado por um evento ativador (denotando o propósito do plano), seguido de uma conjunção de literais de crença representando um contexto, o qual deve ser consequência lógica do conjunto de crenças do agente no momento em que o evento é selecionado pelo agente para o plano ser considerado aplicável. O resto do plano é uma sequência de ações básicas ou sub objetivos que o agente deve atingir ou testar quando uma instância do plano é selecionada para execução. A figura demonstra um exemplo de planos na linguagem AgentSpeak(L).

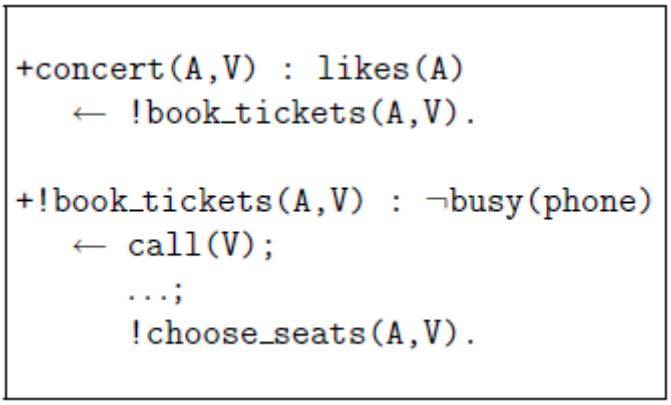


Figura 10 – Exemplo de planos na linguagem AgentSpeak(L) (BORDINI et al., 2007).

O primeiro plano do exemplo da Figura 10 especifica a situação de que um artista A irá se apresentar no local V, explicitado pela adição da crença concert(A,V) como consequência da percepção do ambiente. Se o agente gostar do artista A, então o agente terá o objetivo de realizar a reserva dos ingressos para o conserto.

Já o segundo plano especifica que ao adotar o objetivo de reservar ingressos, se a linha telefônica não estiver ocupada, então o agente deve executar o plano que consiste em ligar para o local do concerto, e após algumas outras ações não explicitadas no exemplo, representadas por “...;”,

deve encerrar a execução do sub plano escolhendo os assentos para o evento ao qual se deseja a reserva.

É importante ressaltar que embora a linguagem AgentSpeak(L) seja bastante útil para criação e principalmente análise de planos para agentes BDI, é necessário que todas as crenças utilizadas na criação dos planos, como no exemplo as crenças concert(A,V), book_tickets(A,V), etc., devem ser previamente definidas no ambiente, definindo o comportamento e ações do agente correspondentes a cada uma delas.

4.3. Exemplo Unificando o Ambiente Multiagentes e a Rede Bayesiana de Emoções

Em sua base de dados padrão, Jason oferece uma série de exemplos de modelos multiagentes que simulam as mais diversas situações. Dentre eles se encontra o chamado *cleaning_robots*, o qual foi utilizado como base para o estudo da funcionalidade e efetividade da rede Bayesiana de emoções em um ambiente multiagentes.

Neste exemplo, dois robôs R1 e R2 coletam e eliminam lixo no planeta Marte. O robô R1 anda sobre o solo do planeta procurando unidades de lixo. Ao encontrar uma unidade, o agente a recolhe e leva até o ponto onde está R2, em seguida retornando ao ponto onde encontrou a unidade para continuar a busca. O robô R2, por sua vez, está posicionado junto a um incinerador e ao receber uma unidade de lixo imediatamente a queima.

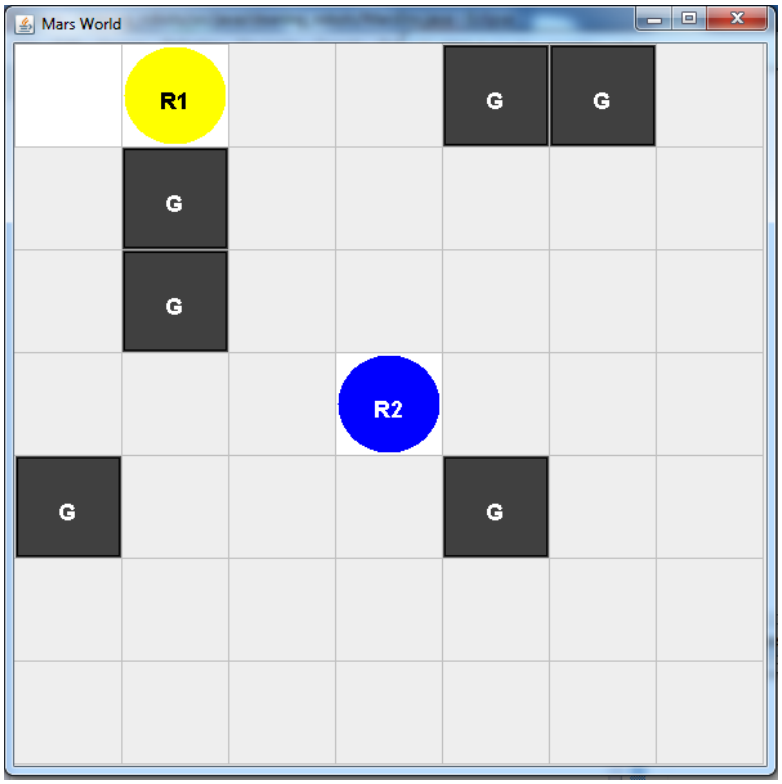


Figura 11 – Interface Gráfica do exemplo *cleaning_robots* na ferramenta Jason, mostrando os agentes R1 e R2. Fonte: autor.

A Figura 11 apresenta uma visão geral do ambiente de simulação. As unidades de lixo, representadas por G no mapa, são colocados em posições randômicas na grade de posições assim que a simulação se inicia. O agente R1 inicia sempre na posição superior mais a esquerda, realizando sua busca percorrendo todas as posições do mapa, linha a linha e sempre da esquerda para direita. O agente R2 inicia sempre na posição central do mapa e fica fixo todo o tempo de simulação nesta posição.

As crenças e planos responsáveis pelo comportamento dos agentes R1 e R2 no ambiente de simulação podem ser encontradas no ANEXO B.

Como dito anteriormente, a rede Bayesiana de emoções foi criada utilizando o software JavaBayes, desenvolvido na linguagem Java, a mesma utilizada para o desenvolvimento da ferramenta Jason, o que facilita a integração destes dois programas.

De forma a explorar as diversas funções oferecidas pelo *software* JavaBayes quanto a análise e simulação de redes Bayesianas, optou-se por integrar seu código ao utilizado na criação do ambiente exemplo estudado.

Para isso, foi necessário utilizá-lo sob a forma de API (Application Programming Interface).

Uma API é um conjunto de rotinas e padrões estabelecidos por um software para a utilização de suas funcionalidades por aplicativos que não pretendam envolver-se em detalhes de implementação do mesmo, ou seja, a utilização do JavaBayes sobre a forma de API possibilita a utilização de funções como a observação de nós e a consequente propagação de probabilidades sobre a rede sem a necessidade de que todo este processo tenha de ser implementado dentro do código onde a rede será introduzida.

Cada exemplo de sistema multiagentes criado no Jason possui, além dos códigos contendo os planos e crenças de cada agente (ANEXO B), um código mestre na linguagem Java que define o mundo a ser criado (ANEXO E). É neste código que são determinados todas as variáveis práticas do ambiente. No exemplo utilizado, são definidas como as ações dos agentes ocorrem e são visualizadas, onde as unidades de lixo serão criadas e como será sua representação no ambiente, o tamanho do ambiente, etc.

Sendo assim, se determinou que junto com a criação do ambiente fosse criada uma instância da rede de emoções para cada um dos agentes aos quais se deseja adicioná-las. Para o estudo deste exemplo apenas R1 recebeu a rede de emoções.

A criação da rede ocorre através dos mesmos métodos utilizados pelo programa JavaBayes e que estão disponíveis em sua API. Este processo ocorre da mesma maneira como ocorre quando se quer criar uma rede Bayesiana no software, embora sem sua representação visual: primeiramente são criados todos os nós com suas variáveis, em seguida são adicionadas as arestas que determinam as relações pai-filho entre os nós, definindo a estrutura da rede, por fim, adicionam-se as probabilidades para cada uma das variáveis da rede.

```
InferenceGraph rede_emocoes = new InferenceGraph();

InferenceGraphNode Valenced_Reaction_to = createNode ( rede_emocoes,  "Valenced_Reaction_to", "True", "False");
InferenceGraphNode Consequences_of_Events = createNode(rede_emocoes, "Consequences_of_Events", "Pleased", "Displeased");

rede_emocoes.create_arc(Valenced_Reaction_to, Consequences_of_Events);

setProbabilityValues(Valenced_Reaction_to, .5, .5);

setProbabilityValues(Consequences_of_Events, "True", .5, .5);
setProbabilityValues(Consequences_of_Events, "False", .5, .5);
```

Figura 12 – Código de definição dos nós *Valenced_Reactions_to* e *Consequences_of_Events* e a ligação entre eles. Fonte: autor.

A Figura 12 apresenta uma porção do código onde são criados os nós *Valenced_Reactions_to* e *Consequences_of_Events*. Na primeira linha, é definido que a nova rede será definida como *rede_emocoes.* Em seguida, são criados os nós a partir do método construtor *createNode*, o qual possui como argumentos a rede na qual o nó será criado, o rótulo que dá o nome do nó e os estados que ele pode estar.

O método *create_arc*, utilizado a seguir, cria uma aresta entre os dois nós que são seus argumentos, direcionada do primeiro para o segundo nó. Para o exemplo, é criada uma aresta indo do nó *Valenced_Reactions_to* para o nó *Consequences_of_Events,* definindo que *Valenced_Reactions_to* é um nó pai de *Consequences_of_Events.*

Com a estrutura criada, resta adicionar as devidas probabilidades iniciais aos nós, processo que ocorre pela utilização do método *setProbabilityValues*. Este método recebe como argumentos o nó ao qual se deseja determinar as probabilidades e, sendo o nó trabalhado um nó sem pais, os valores de suas probabilidades iniciais, dispostas na mesma ordem que seus estados foram definidos.

No caso de o nó possuir pais é necessário informar antes de suas probabilidades qual o estado do pai ao qual estas probabilidades são relativas. Para o exemplo, na última linha do código apresentado são definidas as probabilidades para os estados do nó *Consequences_of_Events* quando o seu nó pai, *Valenced_Reactions_to*, possui estado “*False*”.

Após a rede ser criada é necessário determinar como ela sofrerá as influências do ambiente. Para isso, é necessário analisar cada uma das ações

disponíveis no modelo e que estão definidas em seu código. O agente R1 realiza quatro ações diferentes: *nextSlot*, *moveTowards, pickGarb e dropGarb.* Já o agente R2 realiza apenas a ação burnGarb.

A ação *nextSlot* é responsável por mover R1 para a próxima posição de busca, seja esta a imediatamente a sua direita ou a primeira posição da próxima linha, o que ocorre quando R1 se encontra em uma posição do limite direito do mapa.

O método *moveTowards* faz com que o agente R1 se movimente para uma determinada posição. Está função é utilizada após o agente encontrar e pegar uma unidade de lixo para que ele se movimente tanto para a posição onde estão R2 e o incinerador quanto para fazer, logo em seguida, seu retorno a posição onde a unidade de lixo foi encontrada.

Mais simples, as funções *pickGarb, dropGarb* e *burnGarb* fazem, respectivamente, com que R1 pegue a unidade de lixo existente em sua posição atual, que ele solte a unidade de lixo em sua posse quando se encontra com R2 e que R2 queime a unidade de lixo levada até ele por R2.

Para criar os efeitos do ambiente sobre a rede é necessário que se avalie as possibilidades existentes no ambiente, determinadas pelas funções que nele ocorrem, e em seguida se realizem algumas suposições sobre o ambiente. Estas suposições devem englobar situações válidas para o modelo e que podem realizar efeitos sobre a rede.

Os efeitos sobre a rede são simples observações a alguns de seus nós. A função observação é fornecida pela API do JavaBayes e permite que se realize uma afirmação sobre o estado atual de uma variável da rede Bayesiana, alterando, por consequência, as probabilidades de outras variáveis da rede de assumir certos estados.

Para o exemplo trabalhado foram determinadas três suposições:

- Quando R1 realiza cinco vezes seguidas a função *nextSlot*, ou seja, realiza cinco passos consecutivos sobre o mapa e não encontra nenhuma unidade de lixo ele passa a pensar que o ambiente pode estar limpo e que, portanto, seu trabalho está sendo realizado em vão o que o desmotiva;

- Quando R1 encontra uma unidade de lixo o efeito é o contrário da proposta na suposição anterior, por estar realizando a função a qual completa seu objetivo o agente aumenta sua motivação;
- Quando R1 deposita a unidade de lixo recolhida na posição onde está R2 existe a possibilidade de este agradecer R1 pelo bom trabalho, o que também o motiva.

Após realizar as suposições é necessário traduzi-las em observações na rede Bayesiana de emoções de R1. Para primeira suposição, ocorre um evento do ambiente (a falta de unidades de lixo) que é indesejável para o agente, traduzido na observação da variável do nó *Consequences_of_Events* em estado *Displeased*. Além disso, este é um evento que tem consequências para R1, traduzida em a variável do nó *Consequences_for_Self* ser observada em estado *"true".*

A segunda suposição é semelhante à primeira e envolve observações dos mesmos nós. Quando o agente encontra uma unidade de lixo, ocorre um evento do ambiente (a existência de unidades de lixo no ambiente) que é desejável para o agente, já que possibilita que ele realize a função a qual está destinado. Assim, o nó *Consequences_of_Events* deve ser observado como *Pleased*, enquanto o nó *Consequences_for_Self* também deve ser observado para o estado *"true".*

Já a última suposição não trata de um evento do ambiente e sim uma ação realizada por outro agente, R2, a qual é aprovada por R1. Logo, o nó *Actions_of_Agents* deve ter sua variável observada para o estado *Approving.* Como a ação foi realizada por outro agente, o nó *Other_Agent* deve assumir o estado "*true".*

A partir destas modificações, a rede passa a sofrer uma variação nas probabilidades de seus nós, sendo estas que determinarão quando uma emoção está ou não realizando efeito sobre o agente. Estes valores podem ser trabalhados de três maneiras distintas: pode-se considerar a probabilidade nos nós correspondentes as emoções como um valor de probabilidade de realmente a emoção estar ativa.

Os valores podem ser considerados também como intensidade das emoções, sendo 50%, o valor inicial da emoção, valores maiores significando

que a emoção está ativa em maior intensidade e valores menores significando que a emoção não está realizando efeito sobre o agente.

Por fim, existe a maneira como os valores foram tratados neste trabalho: é considerado um conjunto de 11 emoções consideradas boas, *Happy_for, Pity, Hope, Joy, Satisfaction, Relief, Gratification, Gratitude, Pride, Admiration* e *Love.* Os valores de probabilidade dos nós destas emoções são somados e comparados com o valor somado das probabilidades das emoções restantes, consideradas emoções ruins.

Esta comparação de valores permite observar o estado emocional atual do agente. Se os valores são iguais, o agente é considerado em um estado emocional estável (este é o estado inicial do agente). Se o valor das probabilidades somadas das emoções boas for maior que a probabilidade das ruins, o estado emocional do agente é considerado bom, motivando o agente. Já quando a comparação indica o contrário, com a probabilidade das emoções ruins sendo maior que a das emoções boas, o estado emocional do agente é considerado ruim, o desmotivando.

Definido o modo como avaliar as saídas da rede é necessário criar o modo como elas influenciarão ações do agente. Assim como são necessárias suposições a serem realizadas sobre o ambiente multiagentes para criar os estímulos deste sobre a rede, são necessárias suposições sobre quais efeitos os estados emocionais causarão no agente.

Partindo do princípio que boas emoções motivam o agente e que o agente motivado trabalha melhor, decidiu-se que quando o estado emocional do agente for bom, ele deve aumentar sua eficiência na busca por novas unidades de lixo, enquanto seu desempenho piora quando seu estado emocional é ruim.

Entretanto, ao se observar as funções e características iniciais do exemplo trabalhado se nota que aumentar a eficiência do agente de uma forma que esta seja notada não é uma tarefa fácil, tendo em vista que a cada ciclo de execução o agente anda um espaço do mapa, a eficiência máxima de busca para o modo como os métodos foram desenvolvidos.

Como uma forma de modificar o nível de eficiência do agente e obter uma forma clara para avaliar o efeito das emoções sobre suas ações se propõe

a adição de uma nova característica ao agente R1, diferentes velocidades no processo de busca.

Para realizar este novo processo a velocidade precisa ser simulada, oque é realizado da seguinte maneira: quando o agente está com seu estado emocional estável ou a rede de emoções não está sendo considerada o agente realiza a função *nextSlot* a cada dois ciclos de execução (ou seja, um sim, outro não), isto é, a cada duas vezes que o agente tenta executar a função de seguir para o próximo local da busca, uma vez ele consegue avançar e na outra ele fica parado.

Quando o agente está em um estado emocional ruim o processo deve se tornar mais lento. Neste caso, a cada três vezes que o agente vai executar a ação, em duas tentativas ele acaba ficando parado só realizando a terceira.

Já quando o estado emocional do agente é bom, é necessário que ele se torne mais eficiente em realizar a busca do que quando está em um estado emocional equilibrado, o que significa realizar a busca com uma velocidade maior que, neste caso, pode ser apenas realizar a busca sem realizar paradas. As modificações realizadas sobre o método *nextSlot* podem ser vistas no ANEXO D.

A partir do momento em que a rede de emoções é inserida no código do programa e passa a influenciar nas ações do agente R1 pode-se dizer que existe uma mudança conceitual sobre o que está sendo simulado, tendo em vista que anteriormente R1 era apenas um robô que seguia suas ações estando focado apenas em seu objetivo, e agora ele é um agente que possui reações mais próximas a realidade, onde ações e eventos ocorridos no ambiente podem vir a alterar seu desempenho e objetivos.

5. Análise dos Resultados

Para avaliar os efeitos da rede Bayesiana de emoções sobre o exemplo trabalhado é interessante que primeiramente se observe o comportamento padrão do agente R1 sem emoções. A melhor forma de avaliar seu desempenho e obter um valor mensurável é observar quantos ciclos de execução são necessários para o agente realizar a busca por unidades de lixo em todo o mapa.

Os ciclos de execução considerados aqui são apenas os responsáveis pela execução da função *nextSlot,* pelo fato de esta ser a única função realmente afetada pelas emoções a partir do modelo proposto. Incluir todos os ciclos de execução poderiam tornar os resultados inconsistentes e mascarar o efeito das emoções, tendo em vista que, por exemplo, a posição inicial das unidades de lixo, a qual é aleatória, pode fazer a quantidade de ciclos aumentarem ou diminuírem, pois a quantidade de ciclos necessária para executar a função *moveTowards* varia de acordo com a distancia entre a unidade de lixo encontrada por R1 e o agente R2.

Deste modo, trabalhando apenas o tempo de execução da função *nextSlot,* é possível observar a crescente imprevisibilidade que o modelo assume de acordo com o aumento de suposições implementadas no sistema.

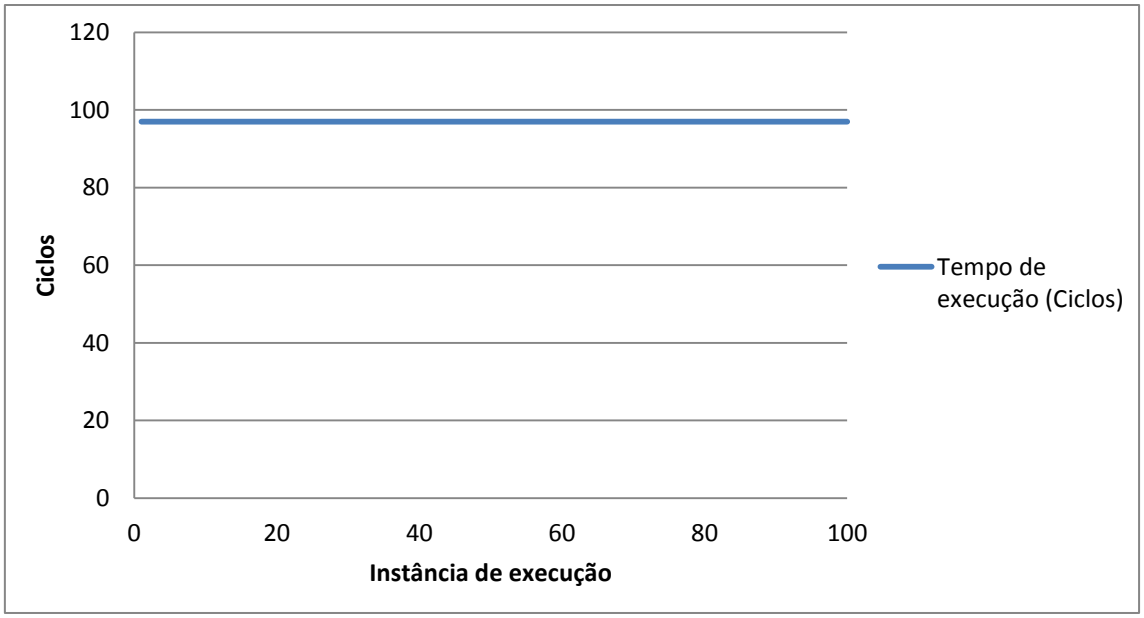


Figura 13 – Tempo de execução para 100 instancias do exemplo sem emoções. Fonte: autor.

A Figura 13 apresenta um gráfico dos valores obtidos para o tempo de execução em número de ciclos para 100 instâncias do exemplo sem utilizar a rede de emoções. Neste caso, o tempo de execução é sempre o mesmo, de 97 ciclos. O tempo é constante porque a execução da função *nextSlot* ocorre sempre da mesma maneira independente das variações do ambiente, inclusive da posição das unidades de lixo.

O mapa trabalhado possui um formato de sete posições em cada linha do mapa, o qual possui sete linhas, resultando um total de 49 posições. Sem emoções, a função *nextSlot* é executada sempre em dois ciclos, um para realizar a função em si e outro em que o agente fica parado, de modo a simular sua velocidade como explicado na Seção 4.3.

Possuindo *nextSlot* este tempo de execução de dois ciclos e o mapa o tamanho de 49 posições poderia se esperar que fossem necessários 98 ciclos para que o agente percorresse todo o mapa ao contrário dos 97 apresentados como resultado. Entretanto, na ultima execução da função o agente atinge a última posição do mapa no 97º ciclo de execução, devendo ficar parado no 98º, o qual não ocorre devido ao exemplo ser executado até que o agente atinja a última posição do mapa, o que ocorre após o 97º ciclo.

Ao acrescentar a primeira suposição proposta ao exemplo, de que o fato de o agente não encontrar unidades de lixo em certo espaço de tempo o

desmotiva e ativar a rede de emoções para que esta provoque variações no desempenho do agente, obtém-se um resultado bastante diferente do visto anteriormente, como pode ser visualizado no gráfico da Figura 14.

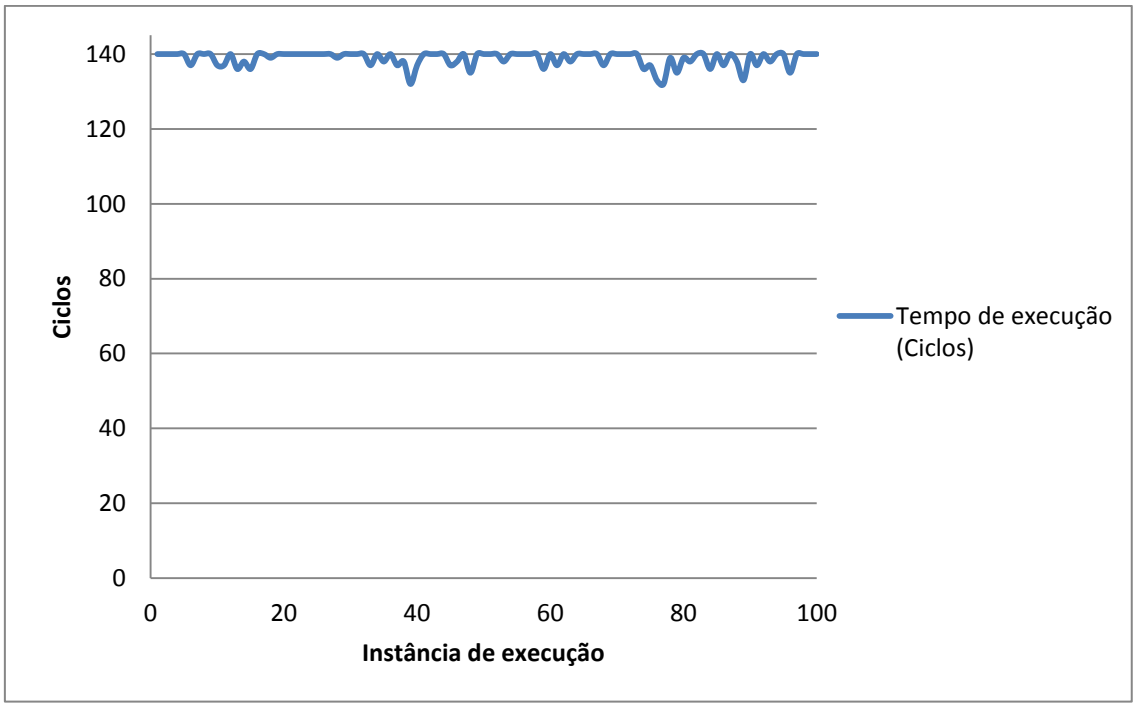


Figura 14 – Tempo de execução para 100 instancias do exemplo com emoções e uma suposição de efeito do ambiente sobre a rede Bayesiana de emoções. Fonte: autor.

O exemplo foi executado 100 vezes sob as novas condições, incluindo a rede Bayesiana de emoções. Porém, desta vez o desempenho do agente R1 foi bastante inferior, sendo que o tempo de execução do exemplo variou entre 132 e 140 ciclos, com um tempo médio de 138,77 ciclos de execução.

Ao serem comparados os valores obtidos a partir do exemplo com esta configuração aos resultados obtidos com o exemplo sem a rede Bayesiana de emoções é possível afirmar que a rede exerceu seu papel, tendo em vista que a única suposição utilizada no exemplo faz com que o agente se desmotive, piorando seu desempenho.

Entretanto, a condição diz que se após cinco passos no mapa o agente se desmotiva caso não tenha encontrado uma unidade de lixo. Essa suposição por si só faz com que na maioria das vezes o agente passe a maior parte de seu tempo de execução sob efeito desta suposição, pois quando o exemplo não apresenta uma unidade de lixo em uma de suas cinco primeiras posições

do agente, a rede é estimulada e, devido a falta de mais suposições, o agente fica preso a este estado pelo resto da execução.

O estudo sobre o exemplo e a rede de emoções se torna mais interessante ao acrescentar a segunda suposição ao ambiente, na qual o agente, quando encontra uma unidade de lixo sofre o efeito contrário ao apresentado na suposição anterior, ficando motivado por estar realizando a função a qual completa seu objetivo. Os resultados obtidos após a execução de 100 instâncias do exemplo sobre estas condições podem ser visualizados na Figura 15.

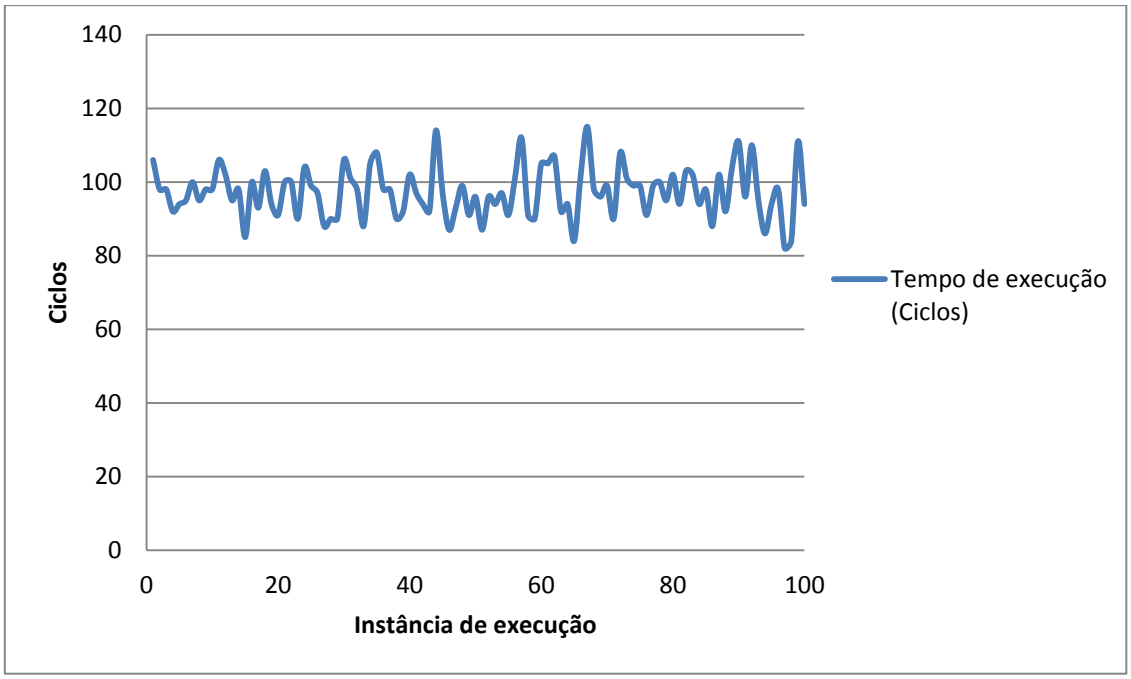


Figura 15 – Tempo de execução para 100 instancias do exemplo com emoções e duas suposições de efeitos do ambiente sobre a rede Bayesiana de emoções. Fonte: autor.

Para o exemplo com a rede de emoções ativada e duas suposições fornecedoras de estímulos determinadas a rede apresenta alguns resultados interessantes. Primeiramente, ocorre uma maior variação entre o tempo de execução observado, variando entre 82 e 115 ciclos. Estes valores demonstram que em algumas instâncias prevaleceu um estado emocional positivo sobre o agente, tornando estas execuções mais rápidas e eficientes, superando os resultados obtidos no exemplo sem emoções. Em contrapartida, em outras instâncias o tempo de execução foi bastante alto, revelando que nestas execuções as emoções negativas prevaleceram sobre o agente.

O tempo médio de execução foi de 97,23 ciclos, valor bastante próximo ao observado para o exemplo sem emoções. Isto indica que, de modo geral, ao se observar o exemplo como um todo, as duas regras propostas quase se anularam, o que se esperaria, tendo em vista que as suposição se baseiam nas mesmas ações, sendo opostas.

Estes dois fatores, a média próxima à vista no exemplo sem emoções e a variação observada nos resultados individuais, indicam que embora as suposições pareçam se anular, o desempenho do agente depende diretamente da configuração do ambiente, a qual é responsável por oferecer os estímulos que alimentam a rede de emoções. Esta é uma característica que também é observada em nosso dia a dia, onde ações e eventos que ocorrem a nossa volta provocam diferentes reações emocionais.

Para adicionar a terceira suposição ao exemplo (R2 pode motivar R1 quando este leva o as unidades de lixo para serem incineradas) é necessário determinar com qual frequência a iteração proposta ocorre. Supondo 50% de chance de o agente R2 interagir com R1, obtém-se o resultado visualizado na Figura 16.

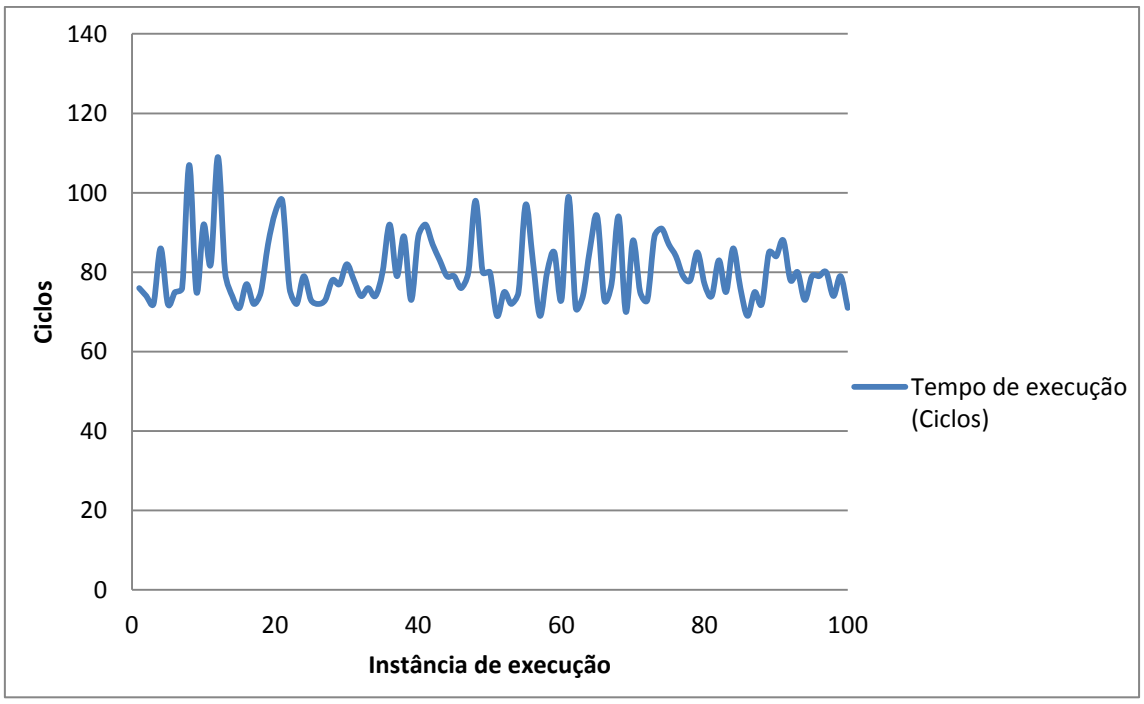


Figura 16 – Tempo de execução para 100 instancias do exemplo com emoções e três suposições de efeitos do ambiente sobre a rede Bayesiana de emoções. Fonte: autor.

Para esta configuração, o exemplo apresenta um tempo máximo de execução de 109 ciclos e um tempo mínimo de 69 ciclos, com uma média de tempo de 80,27 ciclos.

De modo geral, esta é uma configuração que favorece as emoções boas na rede, tendo em vista que existem duas suposições que motivam o agente contra apenas uma que o desmotiva, o que pode ser confirmado por sua baixa média de tempo de execução, existem instancias em que o desempenho é ruim, com destaque a duas execuções que consumiram 107 e 109 ciclos, o que indica que mesmo em um ambiente favorável as emoções tornam a previsão do desempenho do agente impossível.

Por envolver todas as suposições propostas é interessante fazer um estudo mais profundo nesta configuração do exemplo. Para isso se analisará uma instância do exemplo, a qual pode ser vista na Figura 17.

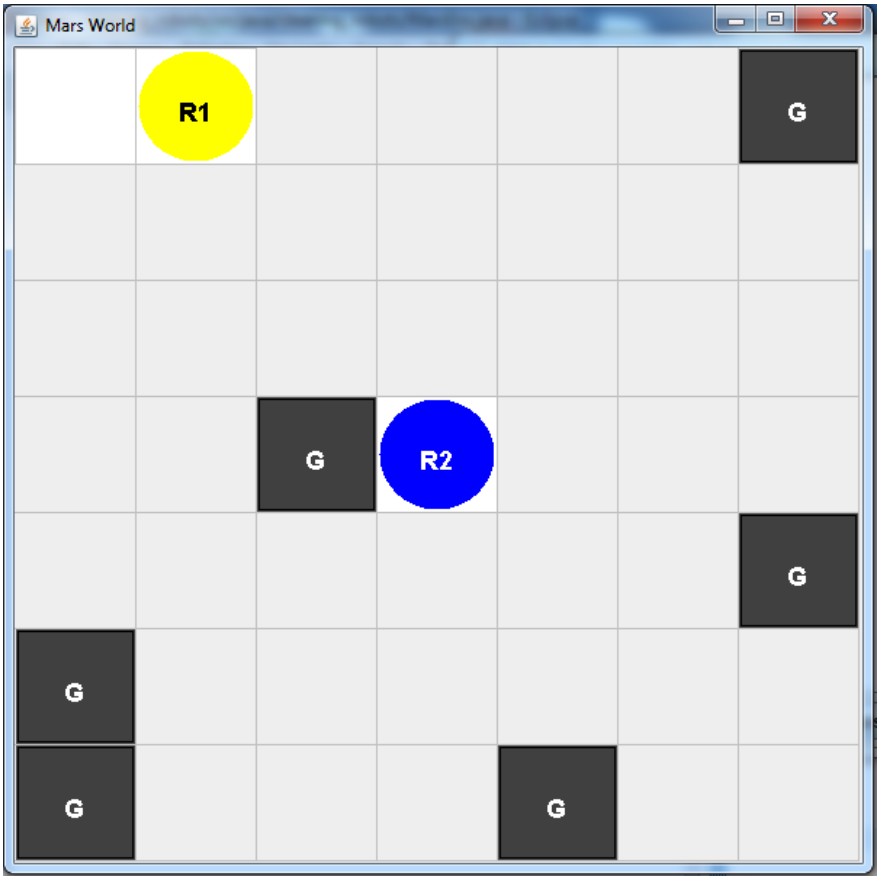


Figura 17 – Instância do exemplo com três suposições e a rede de emoções ativas. Fonte: autor.

Este exemplo compreende 76 ciclos de execução, nos quais o agente possui uma configuração de probabilidades em sua rede. Uma tabela contendo os valores das probabilidades dos nós referentes a cada uma das 22 emoções em cada ciclo do exemplo pode ser vista no ANEXO C. Uma versão simplificada (sem as emoções que não sofrem variação de probabilidade no exemplo) desta tabela é encontrada na Figura 18.

O exemplo se inicia com R1 em seu estado emocional estável, que se mantém pelos 10 primeiros ciclos do exemplo, os quais representam os cinco primeiros passos do agente. Sem encontrar uma unidade de lixo neste espaço, o agente se desmotiva, fruto da primeira suposição que estimula uma alteração nas probabilidades da rede, que passa a apresentar um estado emocional ruim para o agente, diminuindo sua eficiência.

Este estado prossegue até o ciclo 13, quando o agente R1 atinge a posição direita superior do mapa, satisfazendo a segunda suposição proposta, encontrando uma unidade de lixo. As novas entradas são adicionadas a rede, modificando o estado emocional do agente para bom.

Neste mesmo ciclo o agente se dirige ao centro do mapa, onde estão R2 e o incinerador. Ao chegar lá, R1 recebe “algumas palavras de incentivo” de R2, satisfazendo a terceira suposição do exemplo. Novas observações são realizadas sobre a rede que mantém o estado emocional do agente como bom.

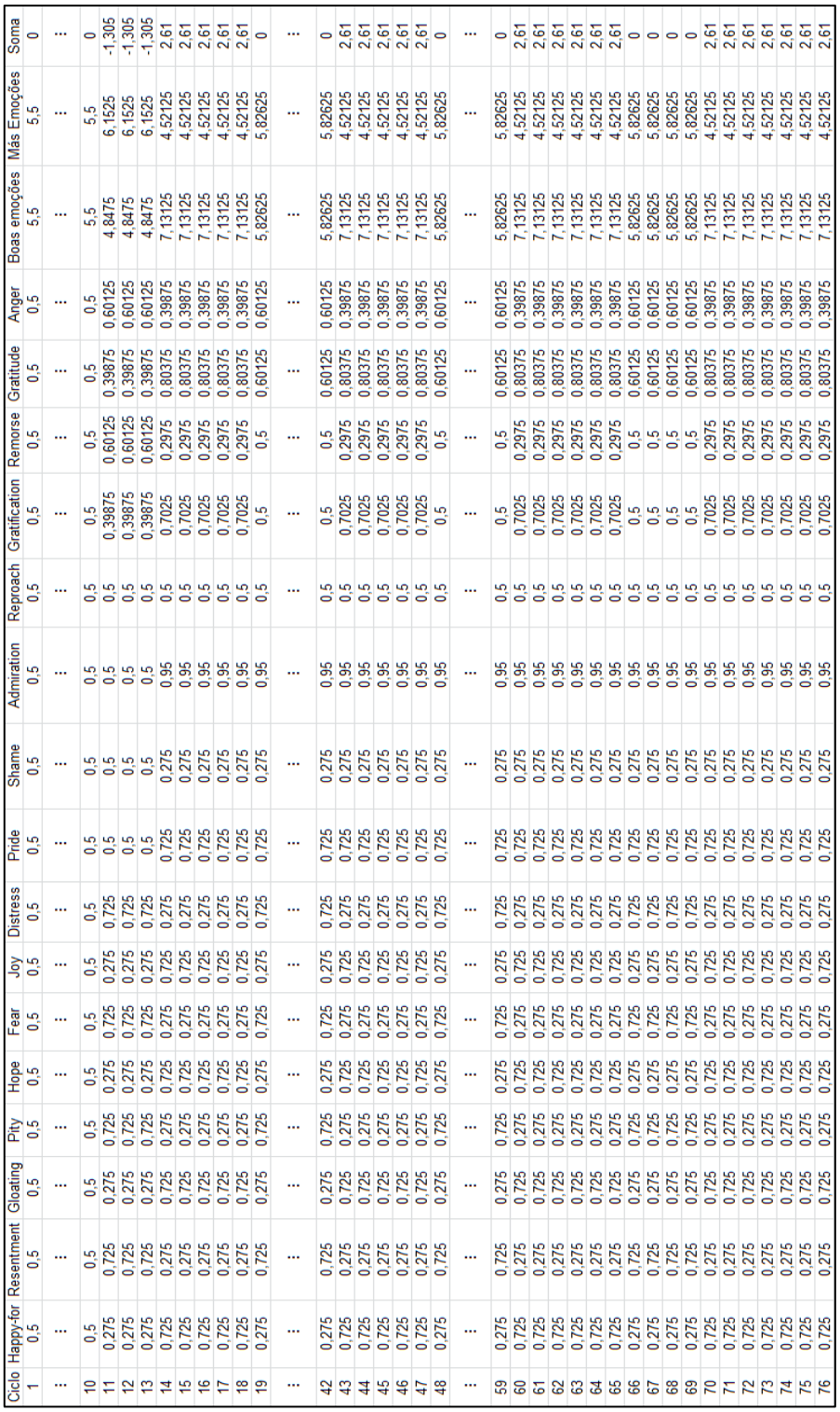
Ao retornar ao ponto superior direito do mapa, R1 continua a seu processo de busca de maneira bastante eficiente, já que se encontra em um estado emocional positivo. Pelos próximos cinco ciclos o estado se mantem.

Como R1 não encontra, mais uma vez, uma unidade de lixo neste espaço, a primeira suposição do exemplo é novamente satisfeita gerando mais uma vez um estimulo negativo para rede. Entretanto, diferente do que ocorreu da primeira vez, quando o estado emocional do agente se tornou ruim, o agente R1 retorna a um estado emocional de equilíbrio.

Este estado só se modificará no ciclo 43, quando o agente encontrará novamente uma unidade de lixo, modificando seu estado emocional novamente para bom, aumentando sua velocidade. Cinco ciclos depois, o agente retorna a um estado emocional de equilíbrio por não encontrar uma unidade de lixo neste período.

No ciclo 60 o agente encontra outra unidade de lixo, causando mais um estímulo positivo para rede. Já em um estado emocional positivo, o agente tem este estímulo renovado no ciclo 61, quando outra unidade de lixo é encontrada. Este estado prossegue por mais cinco ciclos quando retorna a ser estável.

No ciclo 70 o agente volta a possuir um estado emocional positivo, encontrando a penúltima unidade de lixo. Este estado é renovado quando o agente encontra a última unidade de lixo, no ciclo 74, seguindo neste estado até finalizar a execução de sua função.



Ciclo	Happy-for	Resentment	Gloating	Pity	Hope	Fear	Joy	Distress	Pride	Shame	Admiration	Reproach	Gratification	Remorse	Gratitude	Anger	Boas emoções	Más Emoções	Soma
1	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	5,5	5,5	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
10	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	5,5	5,5	0
11	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,39875	0,60125	0,39875	0,60125	4,8475	6,1525	-1,305
12	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,39875	0,60125	0,39875	0,60125	4,8475	6,1525	-1,305
13	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,39875	0,60125	0,39875	0,60125	4,8475	6,1525	-1,305
14	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	7,13125	4,52125	2,61
15	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	7,13125	4,52125	2,61
16	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	7,13125	4,52125	2,61
17	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	7,13125	4,52125	2,61
18	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	7,13125	4,52125	2,61
19	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	5,82625	5,82625	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
42	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	5,82625	5,82625	0
43	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	7,13125	4,52125	2,61
44	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	7,13125	4,52125	2,61
45	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	7,13125	4,52125	2,61
46	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	7,13125	4,52125	2,61
47	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	7,13125	4,52125	2,61
48	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	5,82625	5,82625	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
59	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	5,82625	5,82625	0
60	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	7,13125	4,52125	2,61
61	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	7,13125	4,52125	2,61
62	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	7,13125	4,52125	2,61
63	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	7,13125	4,52125	2,61
64	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	7,13125	4,52125	2,61
65	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	7,13125	4,52125	2,61
66	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	5,82625	5,82625	0
67	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	5,82625	5,82625	0
68	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	5,82625	5,82625	0
69	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	5,82625	5,82625	0
70	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	7,13125	4,52125	2,61
71	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	7,13125	4,52125	2,61
72	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	7,13125	4,52125	2,61
73	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	7,13125	4,52125	2,61
74	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	7,13125	4,52125	2,61
75	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	7,13125	4,52125	2,61
76	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	7,13125	4,52125	2,61

Figura 18 – Probabilidades para a rede Bayesiana de emoções a cada ciclo do exemplo estudado. Fonte: autor.

O exemplo apresentado demonstra como as diferentes situações existentes no modelo proposto modificam o resultado final da execução. A Figura 19 apresenta a união de todos gráficos anteriores, facilitando algumas observações.

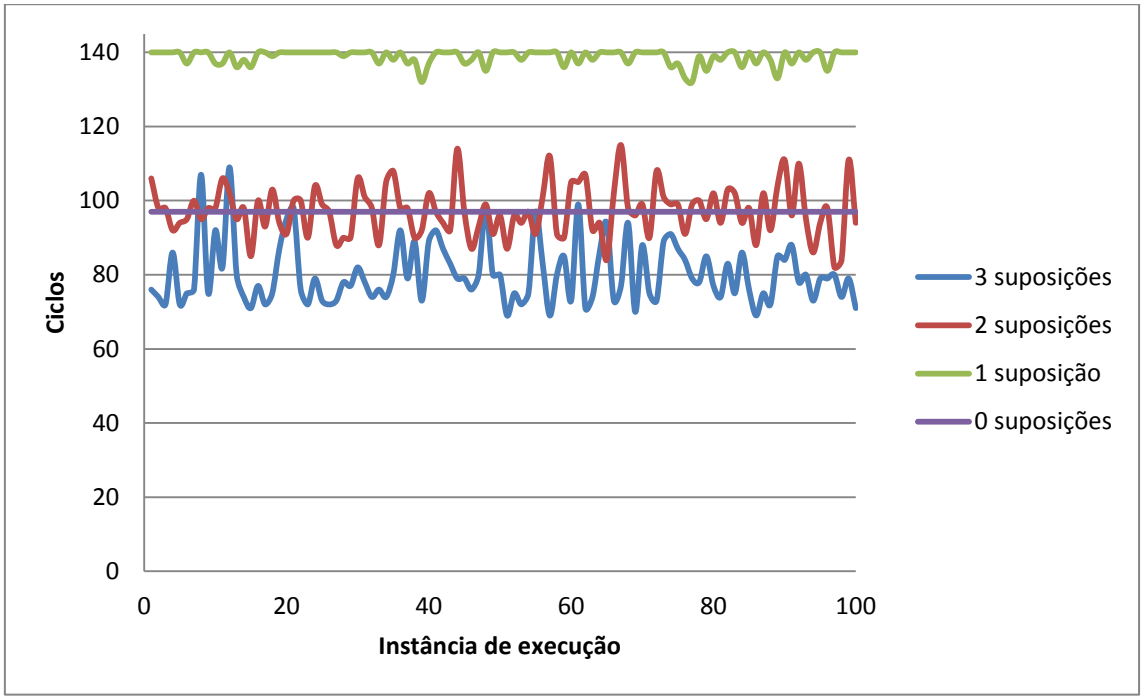


Figura 19 – Tempo de execução para 100 instancias do exemplo com quatro configurações distintas: sem suposições, com uma suposição, com duas suposições e com três suposições. Fonte: autor.

Primeiramente, nota-se que à medida que são adicionadas novas suposições ao exemplo, aumentando sua complexidade, o resultado se torna mais imprevisível. O exemplo sem suposições não variou seus resultados para as 100 execuções. Com uma suposição, o exemplo variou apenas oito ciclos entre seus resultados mais e menos eficientes. Já o exemplo com duas suposições, possuiu uma variação de 33 ciclos entre seus extremos. Por fim, o exemplo com três suposições possuiu a maior variação com 40 ciclos entre seu ápice de desempenho e seu pior resultado.

Outro ponto a se destacar é que os valores variam em torno de suas médias de desempenho, definidas pelas características de cada exemplo. O teste com apenas uma suposição ruim para o desempenho do agente obteve o pior resultado, com uma média de 138,77 ciclos por execução.

O teste com duas suposições circunda uma média de 97,23 ciclos por execução, enquanto a média do teste com três suposições, sendo estas numa proporção de duas motivacionais e uma de desmotivação para o agente, foi de 80,27 ciclos por execução.

Estes dois últimos casos, com duas e três suposições apresentam uma característica importante: apesar de suas médias se distanciarem por aproximadamente 17 ciclos, distancia bastante grande para um universo de execuções que em várias delas não supera 100 ciclos, têm seus resultados se cruzando em alguns pontos. Isto indica que embora as suposições indicassem que a versão com três suposições possuísse sempre um agente mais eficiente do que o exemplo com duas, a imprevisibilidade da rede da rede Bayesiana de emoções, bem como a aleatoriedade do ambiente fazem com que seja possível o agente no ambiente com três suposições ser pior do que o do exemplo com duas.

6. Conclusão e Trabalhos Futuros

Observando o funcionamento da rede Bayesiana de emoções é possível realizar algumas afirmações. Primeiramente, que a utilização de uma rede Bayesiana para a aplicação computacional do modelo OCC é possível, permitindo que através da manipulação de probabilidades em suas variáveis aumente-se a similaridade do modelo com a realidade, tendo em vista que o modelo oferece um método de decisão previsível quanto à quais emoções ocorrem de acordo com determinado evento, o que pode ser modificado em uma rede Bayesiana.

A utilização do modelo OCC sob a forma de uma rede Bayesiana apresenta outra característica importante, permitir uma visualização das relações existentes entre as diferentes emoções que compõe o modelo, relações estas que muitas vezes ficam escondidas em sua estrutura. Por exemplo, ao determinar-se na rede criada a existência da emoção "*Anger*", observa-se um aumento na probabilidade de que a emoção "*Fear*" também ocorra, o contrário de "*Happy-for*", que se torna menos provável.

Outro ponto a ser observado é a adição de imprevisibilidade ao ambiente multiagentes provocada pela utilização da rede Bayesiana de emoções por seus agentes. Este efeito pode ser observado nas variações de velocidades apresentadas no estudo do modelo exemplo. A adição de suposições incertas, como a do exemplo em que um agente pode ou não motivar o outro, fortalecem esta característica.

Embora o exemplo estudado seja simples, foi possível visualizar as características principais da rede Bayesiana de emoções, bem como sua influencia em um ambiente multiagentes. Entretanto, existem alguns pontos a serem observado de forma que em trabalhos futuros suas características sejam ainda mais aproveitadas.

Primeiramente, o modelo utilizado no exemplo, com estados emocionais, pode ser aplicado a um exemplo mais elaborado, com diversos agentes possuindo suas próprias redes Bayesianas de emoções. Este exemplo propiciaria mais interações entre os agentes e o ambiente a qual estiverem inseridos e, também, entre si, possibilitando, assim, uma maior quantidade de

estímulos à rede e a consequente maior variação nas emoções dos indivíduos, influenciando diretamente seus comportamentos.

O modelo trabalhado trata apenas estados emocionais, reduzindo os valores individuais das emoções a dois valores gerais, um compreendendo as probabilidades de todas as emoções consideradas boas e outro compreendendo as das emoções ruins. Este modelo pode ser expandido, por exemplo, ao se trabalhar com níveis de estados emocionais. Assim, quando o valor total para o somatório entre os valores positivos e negativos de probabilidade for muito alto, o agente deve ter um comportamento diferente de quando este valor não for tão alto.

Ao se definirem faixas de valores para o somatório das emoções é possível que, ao invés de haver apenas três tipos de reação à rede, estável, boa e ruim, possa existir diversas reações para o agente, tantas quantas forem as faixas de valores determinadas.

Outra alternativa é trabalhar com os valores individuais das emoções, seja avaliando-as apenas individualmente ou em conjunto a um sistema de estados emocionais. Embora bastante complexo, por exigir um conhecimento dos efeitos de cada emoção sobre a maneira como o agente encara as ações que deve realizar, permite que o modelo trabalhado se torne bastante completo e imprevisível, permitindo ao agente possuir diversas reações aos eventos ocorridos no ambiente a qual está inserido.

Todas estas propostas modificam o modo como tratar a rede e como esta sofre estímulos do ambiente mas não sua estrutura e valores. Embora não seja recomendado modificar sua estrutura, tendo em vista manter sua proximidade ao modelo OCC de emoções, seus valores podem ser trabalhados.

Outra modificação interessante pode ser a inserção de personalidade a agentes. Esta característica ocorreria a partir da modificação dos valores da rede Bayesiana de emoções individual de cada agente. Deste modo, por exemplo, ao se trabalhar com um agente otimista, talvez seja interessante modificar as probabilidades iniciais de sua rede de forma a possuir maior chance de ativar emoções boas do que ruins.

Todas estas alternativas de trabalhos futuros revelam o potencial da rede Bayesiana de emoções que pode, de maneira simples, adicionar emoções

simuladas a agentes e aumentar a proximidade destes a natureza humana, imprevisível e que reflete o ambiente em que vivem.

Bibliografia

ADAMATTI, D. F. **AFRODITE - Ambiente de Simulação Baseado em Agentes com Emoções**. Dissertação de Mestrado. Universidade Federal do Rio Grande do Sul. Porto Alegre. 2003.

ALVAREZ, L. O.; SICHMAN, J. Introdução aos Sistemas Multiagentes. **Jornada de Atualização em Informática**, Brasília, 1997. 1-38.

BERCHT, M. **Em Direção a Agentes Pedagógicos com Dimensões Afetivas.** Tese de Doutorado. Universidade Federal do Rio Grande do Sul. Porto Alegre. 2001.

BITENCOURT, G. K.; SILVEIRA, R. A.; MARCHI, J. TrustE An Emotional Trust Model for Agents. In: Proceedings of European Workshop on Multi-Agent Systems , EUMAS (2023), p 54-67.

BORDINI, R. H.; HÜBNER, J. F.; WOOLDRIDGE, M. **programming multi-agent systems in AgentSpeak using Jason**. England: Wiley, 2007.

CAÑAMERO, D.; VAN DE VELDE, W. Emotically Grounded Social Interaction. **Human Cognition and Social Agent Tecnology**, Amsterdam, 1999.

CONATI, C.; MACLAREN, H. **Empirically Building and Evaluating a Probabilistic Model of User Affect**. University of British Columbia. Vancouver. 2010.

COZMAN, F. G. Bayesian Networks in Java: User manual and download. **JavaBayes**, 2001. Disponivel em: <http://www.cs.cmu.edu/~javabayes/Home/index.html>. Acesso em: 9 Abril 2013.

DAMÁSIO, A. R. **O Mistério da Consciência:** do Corpo e das Emoções ao Conhecimento de Si. São Paulo: Companhia das Letras, 2000. 474 p.

DEL NERO, H. S. **O Sítio da Mente:** Pensamento, Emoção e Vontade no Cerebro Humano. São Paulo: Collegium Cognito, 1997. 510 p.

ELLIOT, A.; PEKRUN, R. Emotion in the Hierarchical Model of Aproach-Avoidance Achievement Motivation. **Emotion in Education**, London, 2007. 57-74.

FERBER, J.; GASSER, L. Intelligense Artificielle Distibuée. **Workshop on Expert Systems & their Applications.**, Avingnon, 1991. 10.

FROZZA, R. **SIMULA: Ambiente para Desenvolvimento de Sistemas Multiagentes Reativos**. Dissertação de Mestrado. Universidade Federal do Rio Grande do Sul. Porto Alegre. 1997.

GRATCH, J.; MARSELHA, S. Modeling Emotions in the Mission Rehearsal Exercice. **Conference on computer generated forces and behavioral representation**, 2001. 10.

HRUSCHKA JR., E. R. **Imputação Bayesiana no Contexto da Mineração de Dados**. Tese de Doutorado. Universidade Federal do Rio de Janeiro. Rio de Janeiro. 2003.

JAQUES, P. A.; VICCARI, R. M. Considering Student's Emotions in Computer-Mediated Learning Environments. **Web-based Intelligent e-Learning Systems: Technologies an Applications**, Hershey, 2005. 122-138.

MOFFAT, D. C.; FRIJDA, N. Functional Models of Emotions. **Affective Minds**, Amsterdam, 2000. 59-68.

MORGADO, A. C. et al. **Analise Combinatória e Probabilidade**. Rio de Janeiro: [s.n.], 2001.

ORTONY, A.; CLORE, G.; COLLINS, A. **The Cognitive Structure of Emotions**. Cambridge: Cambridge University Press, 1988.

PICARD, R. **Affective Computing**. Cambridge: MIT Press, 1997. 292 p.

REBONATTO, M. T. **Simulação Paralela de Eventos Discretos com Uso de Memória Compartilhada Distribuída**. Dissertação de Mestrado. Universidade Federal do Rio Grande do Sul. Porto Alegre. 2000.

REZENDE, S. O. **Sistemas Inteligentes - Fundamentos e Aplicações**. 1a. ed. São Paulo: Manole, 2001.

RUSSEL, S.; NORVIG, P. **Artificial Intelligence A Modern Approach**. 2a. ed. Upper Saddle River, New Jersey: Prentice Hall, 2003.

SABATER, J.; SIERRA, C. **Regret: reputation in gregarius societies.** In Proceedings of the fifth International Conference on Autonomous Agents, ACM (2001).

SABOURIN, J.; MOTT, B.; LESTER, J. **Modeling Learner Affect with Theoretically Grounded Dynamic Bayesian Networks**. North Carolina State University. Raleigh. 2011.

SLOMAN, A. Architectural Requirements for Human-like Agents Both Natural and Artificial (What sorts of machines ca love?). **Human Cognition and Social Agent Tecnology**, Amsterdam, 1999.

SLOMAN, A. Beyond Shallow Models of Emotions. **COGNITIVE PROCESSING**, 2001. 177-198.

WERHLI, A. V. **Reconstruction of Gene Regulatory Networks from Postgenomic Data**. Tese de Doutorado. University of Edinburgh. Edinburgh, p. 230. 2007.

WOOLDRIDGE, M. **Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence.** Cambridge: MIT Press, 1999.

YOON, J.; CHO, S.-B. **A Mobile Intelligent Synthetic Character with Natural Behavior Generation**. Yonsei University. Seoul. 2010.

Anexos

ANEXO A: Descrição da rede Bayesiana de emoções fornecida pelo software JavaBayes.

```
// Bayesian network
network "InternalNetwork" { //34 variables and 34 probability distributions
}
variable  "Consequences_of_Events" { //2 values
          type discrete[2] {  "Pleased"  "Displeased" };
          property "position = (360, 199)" ;
}
variable  "Actions_of_Agents" { //2 values
          type discrete[2] {  "Approving"  "Disapproving" };
          property "position = (876, 197)" ;
}
variable  "Aspects_of_Objects" { //2 values
          type discrete[2] {  "Liking"  "Disliking" };
          property "position = (1190, 203)" ;
}
variable  "Consequences_for_Others" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (220, 283)" ;
}
variable  "Consequences_for_Self" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (451, 265)" ;
}
variable  "Happy-for" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (28, 506)" ;
}
variable  "Resentment" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (119, 507)" ;
}
variable  "Gloating" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (227, 507)" ;
}
variable  "Pity" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (305, 506)" ;
}
```

```
variable  "Hope" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (382, 502)" ;
}
variable  "Fear" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (456, 503)" ;
}
variable  "Hope_Confirmed" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (314, 583)" ;
}
variable  "Fear_Confirmed" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (537, 583)" ;
}
variable  "Satisfaction" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (236, 658)" ;
}
variable  "Fears-confirmed" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (493, 668)" ;
}
variable  "Relief" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (631, 670)" ;
}
variable  "Disapoitment" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (349, 658)" ;
}
variable  "Joy" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (560, 498)" ;
}
variable  "Distress" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (663, 493)" ;
}
variable  "Self_Agent" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (758, 284)" ;
}
variable  "Other_Agent" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (1011, 269)" ;
}
variable  "Pride" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (770, 393)" ;
}
```

```
variable  "Shame" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (884, 394)" ;
}
variable  "Admiration" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (968, 398)" ;
}
variable  "Reproach" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (1065, 391)" ;
}
variable  "Gratification" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (714, 633)" ;
}
variable  "Remorse" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (836, 632)" ;
}
variable  "Gratitude" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (939, 635)" ;
}
variable  "Anger" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (1050, 637)" ;
}
variable  "Love" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (1149, 297)" ;
}
variable  "Hate" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (1260, 297)" ;
}
variable  "Valenced_Reaction_to" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (753, 55)" ;
}
variable  "Desirable_for_Other" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (202, 368)" ;
}
variable  "Prospect_Relevant" { //2 values
          type discrete[2] {  "true"  "false" };
          property "position = (455, 379)" ;
}
probability (  "Consequences_of_Events"  "Valenced_Reaction_to" ) { //2 variable(s) and 4
values
          table
                    0.5 0.5 0.5 0.5;
```

```
}
probability (  "Actions_of_Agents"  "Valenced_Reaction_to" ) { //2 variable(s) and 4 values
        table
                0.5 0.5 0.5 0.5;
}
probability (  "Aspects_of_Objects"  "Valenced_Reaction_to" ) { //2 variable(s) and 4 values
        table
                0.5 0.5 0.5 0.5;
}
probability (  "Consequences_for_Others"  "Consequences_of_Events" ) { //2 variable(s) and 4
values
        table
                0.5 0.5 0.5 0.5;
}
probability (  "Consequences_for_Self"  "Consequences_of_Events" ) { //2 variable(s) and 4
values
        table
                0.5 0.5 0.5 0.5;
}
probability (  "Happy-for"  "Consequences_of_Events"  "Desirable_for_Other" ) { //3 variable(s)
and 8 values
        table
                0.95 0.5 0.5 0.05 0.05 0.5 0.5 0.95;
}
probability (  "Resentment"  "Consequences_of_Events"  "Desirable_for_Other" ) { //3
variable(s) and 8 values
        table
                0.5 0.05 0.95 0.5 0.5 0.95 0.05 0.5;
}
probability (  "Gloating"  "Consequences_of_Events"  "Desirable_for_Other" ) { //3 variable(s)
and 8 values
        table
                0.5 0.95 0.05 0.5 0.5 0.05 0.95 0.5;
}
probability (  "Pity"  "Consequences_of_Events"  "Desirable_for_Other" ) { //3 variable(s) and 8
values
        table
                0.05 0.5 0.5 0.95 0.95 0.5 0.5 0.05;
}
probability (  "Hope"  "Consequences_of_Events"  "Prospect_Relevant" ) { //3 variable(s) and 8
values
        table
                0.95 0.5 0.5 0.05 0.05 0.5 0.5 0.95;
}
probability (  "Fear"  "Consequences_of_Events"  "Prospect_Relevant" ) { //3 variable(s) and 8
values
        table
                0.5 0.05 0.95 0.5 0.5 0.95 0.05 0.5;
}
probability (  "Hope_Confirmed"  "Hope" ) { //2 variable(s) and 4 values
        table
                0.5 0.0 0.5 0.0;
```

```
}
probability (  "Fear_Confirmed"  "Fear" ) { //2 variable(s) and 4 values
        table
                0.5 0.0 0.5 0.0;
}
probability (  "Satisfaction"  "Hope_Confirmed" ) { //2 variable(s) and 4 values
        table
                0.95 0.05 0.05 0.95;
}
probability (  "Fears-confirmed"  "Fear_Confirmed" ) { //2 variable(s) and 4 values
        table
                0.95 0.05 0.05 0.95;
}
probability (  "Relief"  "Fear_Confirmed" ) { //2 variable(s) and 4 values
        table
                0.05 0.95 0.95 0.05;
}
probability (  "Disapoitment"  "Hope_Confirmed" ) { //2 variable(s) and 4 values
        table
                0.05 0.95 0.95 0.05;
}
probability (  "Joy"  "Prospect_Relevant"  "Consequences_of_Events" ) { //3 variable(s) and 8
values
        table
                0.5 0.05 0.95 0.5 0.5 0.95 0.05 0.5;
}
probability (  "Distress"  "Consequences_of_Events"  "Prospect_Relevant" ) { //3 variable(s)
and 8 values
        table
                0.05 0.5 0.5 0.95 0.95 0.5 0.5 0.05;
}
probability (  "Self_Agent"  "Actions_of_Agents" ) { //2 variable(s) and 4 values
        table
                0.5 0.5 0.5 0.5;
}
probability (  "Other_Agent"  "Actions_of_Agents" ) { //2 variable(s) and 4 values
        table
                0.5 0.5 0.5 0.5;
}
probability (  "Pride"  "Self_Agent"  "Actions_of_Agents" ) { //3 variable(s) and 8 values
        table
                0.95 0.5 0.5 0.05 0.05 0.5 0.5 0.95;
}
probability (  "Shame"  "Self_Agent"  "Actions_of_Agents" ) { //3 variable(s) and 8 values
        table
                0.5 0.95 0.05 0.5 0.5 0.05 0.95 0.5;
}
probability (  "Admiration"  "Other_Agent"  "Actions_of_Agents" ) { //3 variable(s) and 8 values
        table
                0.95 0.5 0.5 0.05 0.05 0.5 0.5 0.95;
}
probability (  "Reproach"  "Other_Agent"  "Actions_of_Agents" ) { //3 variable(s) and 8 values
```

```
        table
                0.5 0.95 0.05 0.5 0.5 0.05 0.95 0.5;
}
probability (  "Gratification"  "Pride"  "Joy" ) { //3 variable(s) and 8 values
        table
                0.95 0.5 0.5 0.05 0.05 0.5 0.5 0.95;
}
probability (  "Remorse"  "Shame"  "Distress" ) { //3 variable(s) and 8 values
        table
                0.95 0.5 0.5 0.05 0.05 0.5 0.5 0.95;
}
probability (  "Gratitude"  "Admiration"  "Joy" ) { //3 variable(s) and 8 values
        table
                0.95 0.5 0.5 0.05 0.05 0.5 0.5 0.95;
}
probability (  "Anger"  "Distress"  "Reproach" ) { //3 variable(s) and 8 values
        table
                0.95 0.5 0.5 0.05 0.05 0.5 0.5 0.95;
}
probability (  "Love"  "Aspects_of_Objects" ) { //2 variable(s) and 4 values
        table
                0.95 0.05 0.05 0.95;
}
probability (  "Hate"  "Aspects_of_Objects" ) { //2 variable(s) and 4 values
        table
                0.05 0.95 0.95 5.0;
}
probability (  "Valenced_Reaction_to" ) { //1 variable(s) and 2 values
        table
                0.5     // p(true | evidence )
                0.5;    // p(false | evidence );
}
probability (  "Desirable_for_Other"  "Consequences_for_Others" ) { //2 variable(s) and 4
values
        table
                0.5 0.5 0.5 0.5;
}
probability (  "Prospect_Relevant"  "Consequences_for_Self" ) { //2 variable(s) and 4 values
        table
                0.5 0.5 0.5 0.5;
}
```

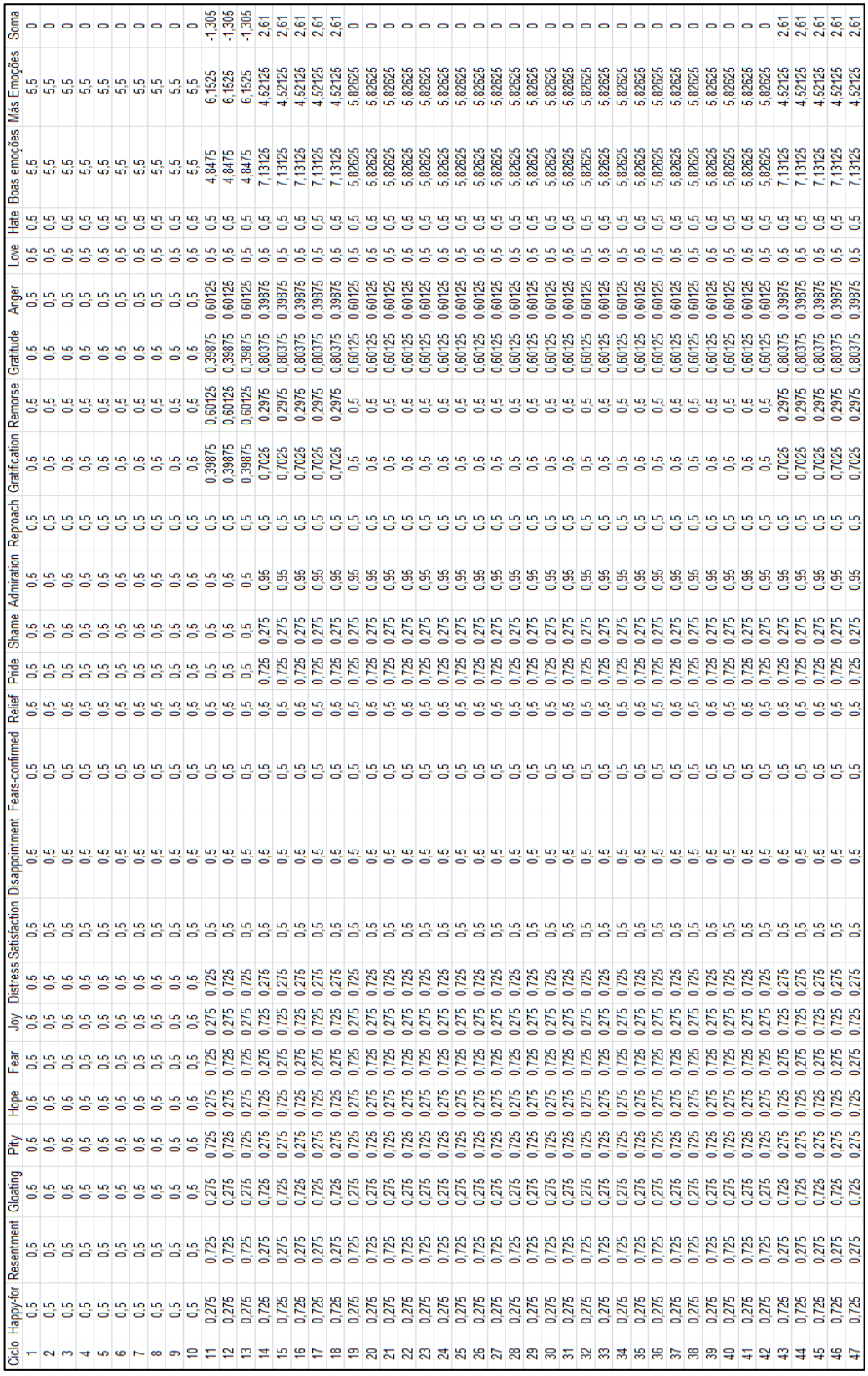
ANEXO B: Definição de planos e crenças para os agentes R1 e R2 componentes do exemplo cleaning_robots encontrado no software de modelagem e simulação de ambientes multiagentes Jason.

```
// mars robot 1

/* Initial beliefs */

at(P) :- pos(P,X,Y) & pos(r1,X,Y).

/* Initial goal */

!check(slots).

/* Plans */

+!check(slots) : not garbage(r1)
  <- next(slot);
     !!check(slots).

+!check(slots).

+garbage(r1) : not .desire(carry_to(r2))
  <- !carry_to(r2).

+!carry_to(R)
  <- // remember where to go back
     ?pos(r1,X,Y);
     -+pos(last,X,Y);
     // carry garbage to r2
     !take(garb,R);
     // goes back and continue to check
     !at(last);
     !!check(slots).

+!take(S,L) : true
  <- !ensure_pick(S);
```

```
    !at(L);
    drop(S).
+!ensure_pick(S) : garbage(r1)
  <- pick(garb);
    !ensure_pick(S).
+!ensure_pick(_).
+!at(L) : at(L).
+!at(L) <- ?pos(L,X,Y);
        move_towards(X,Y);
        !at(L).
```

```
// mars robot 2
+garbage(r2) : true <- burn(garb).
```

ANEXO C: Tabela contendo as probabilidades das emoções componentes da rede Bayesiana para um exemplo com três suposições.



Ciclo	Happy-for	Resentment	Gloating	Pity	Hope	Fear	Joy	Distress	Satisfaction	Disappointment	Fears-confirmed	Relief	Pride	Shame	Admiration	Reproach	Gratification	Remorse	Gratitude	Anger	Love	Hate	Boas emoções	Más Emoções	Soma
1	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	5,5	5,5	0
2	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	5,5	5,5	0
3	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	5,5	5,5	0
4	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	5,5	5,5	0
5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	5,5	5,5	0
6	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	5,5	5,5	0
7	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	5,5	5,5	0
8	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	5,5	5,5	0
9	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	5,5	5,5	0
10	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	5,5	5,5	0
11	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,39875	0,60125	0,39875	0,60125	0,5	0,5	4,8475	6,1525	-1,305
12	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,39875	0,60125	0,39875	0,60125	0,5	0,5	4,8475	6,1525	-1,305
13	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,39875	0,60125	0,39875	0,60125	0,5	0,5	4,8475	6,1525	-1,305
14	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	0,5	0,5	7,13125	4,52125	2,61
15	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	0,5	0,5	7,13125	4,52125	2,61
16	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	0,5	0,5	7,13125	4,52125	2,61
17	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	0,5	0,5	7,13125	4,52125	2,61
18	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	0,5	0,5	7,13125	4,52125	2,61
19	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
20	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
21	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
22	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
23	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
24	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
25	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
26	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
27	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
28	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
29	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
30	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
31	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
32	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
33	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
34	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
35	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
36	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
37	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
38	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
39	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
40	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
41	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
42	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
43	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	0,5	0,5	7,13125	4,52125	2,61
44	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	0,5	0,5	7,13125	4,52125	2,61
45	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	0,5	0,5	7,13125	4,52125	2,61
46	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	0,5	0,5	7,13125	4,52125	2,61
47	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	0,5	0,5	7,13125	4,52125	2,61

48	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
49	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
50	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
51	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
52	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
53	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
54	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
55	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
56	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
57	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
58	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
59	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
60	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	0,5	0,5	7,13125	4,52125	2,61
61	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	0,5	0,5	7,13125	4,52125	2,61
62	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	0,5	0,5	7,13125	4,52125	2,61
63	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	0,5	0,5	7,13125	4,52125	2,61
64	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	0,5	0,5	7,13125	4,52125	2,61
65	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	0,5	0,5	7,13125	4,52125	2,61
66	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
67	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
68	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
69	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,5	0,5	0,60125	0,60125	0,5	0,5	5,82625	5,82625	0
70	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	0,5	0,5	7,13125	4,52125	2,61
71	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	0,5	0,5	7,13125	4,52125	2,61
72	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	0,5	0,5	7,13125	4,52125	2,61
73	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	0,5	0,5	7,13125	4,52125	2,61
74	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	0,5	0,5	7,13125	4,52125	2,61
75	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	0,5	0,5	7,13125	4,52125	2,61
76	0,725	0,275	0,725	0,275	0,725	0,275	0,725	0,275	0,5	0,5	0,5	0,5	0,725	0,275	0,95	0,5	0,7025	0,2975	0,80375	0,39875	0,5	0,5	7,13125	4,52125	2,61

ANEXO D: Imagens comparativas dos códigos padrão e modificado do método nextSlot.

```
// ##############  Método nextSlot sem modifocações ###############

void nextSlot() throws Exception {
    Location r1 = getAgPos(0);
    r1.x++;
    if (r1.x == getWidth()) {
        r1.x = 0;
        r1.y++;
    }
    // finished searching the whole grid
    if (r1.y == getHeight()) {
        return;
    }
    setAgPos(0, r1);
    setAgPos(1, getAgPos(1)); // just to draw it in the view
}
```

```
// Método para que o agente avance uma posição no mapa

// ############# Método nextSlot modificado para trabalhar com a rede de emoções #############

void nextSlot() throws Exception{

    /**
     * #######  Obtendo as probabilidades atuais das emoções  #########
     */
    try{
        belief1 = getBelief(rede_emocoes, Happy_for);
        belief2 = getBelief(rede_emocoes, Pity);
        belief3 = getBelief(rede_emocoes, Hope);
        belief4 = getBelief(rede_emocoes, Joy);
        belief5 = getBelief(rede_emocoes, Satisfaction);
        belief6 = getBelief(rede_emocoes, Relief);
        belief7 = getBelief(rede_emocoes, Gratification);
        belief8 = getBelief(rede_emocoes, Gratitude);
        belief9 = getBelief(rede_emocoes, Pride);
        belief10 = getBelief(rede_emocoes, Admiration);
        belief11 = getBelief(rede_emocoes, Love);
        belief12= getBelief(rede_emocoes, Resentment);
        belief13 = getBelief(rede_emocoes, Gloating);
        belief14 = getBelief(rede_emocoes, Fear);
        belief15 = getBelief(rede_emocoes, Distress);
        belief16 = getBelief(rede_emocoes, Disappointment);
        belief17 = getBelief(rede_emocoes, Fears_Confirmed);
        belief18 = getBelief(rede_emocoes, Remorse);
        belief19 = getBelief(rede_emocoes, Anger);
        belief20 = getBelief(rede_emocoes, Shame);
        belief21 = getBelief(rede_emocoes, Reproach);
        belief22 = getBelief(rede_emocoes, Hate);
    } catch (NullPointerException e) {
        e.printStackTrace();
        init(count,ns_count);
    }

    double good_beliefs = belief1 + belief2 + belief3 + belief4 + belief5 + belief6 + belief7 + belief8 + belief9 + belief10 + belief11;
    double bad_beliefs = belief12 + belief13 + belief14 + belief15 + belief16 + belief17 + belief18 + belief19 + belief20 + belief21 + belief22;



    // teste para verificar se o agente possui estado emocional positivo
    if(good_beliefs - bad_beliefs > 0.1 ){
        stop = 1;
    }
    // teste para verificar se o agente possui estado emocional positivo
    if ((good_beliefs - bad_beliefs < -0.1) && (stop_bad == 0)){
        System.out.println("PARADO");
        stop_bad++;
        step_count++;
        return;
    }
    // teste para verificar se o agente deve realizar a função ou ficar parado para simular a velocidade
    if(stop == 0){
        System.out.println("PARADO");
        stop++;
    }else{
        step_count++;
        stop = 0;
        stop_bad = 0;
        //teste do número de passos sem encontrar uma unidade de lixo
        if (step_count>=5){
            // influências sobre a rede por não encontrar unidade de lixo
            Consequences_of_Events.set_observation_value("Displeased");
            Consequences_for_Self.set_observation_value("True");
        }

        Location r1 = getAgPos(0);
        r1.x++;
        if (r1.x == getWidth()) {
            r1.x = 0;
            r1.y++;
        }
        // finished searching the whole grid
        if (r1.y == getHeight()) {
            return;
        }
        setAgPos(0, r1);
        setAgPos(1, getAgPos(1)); // just to draw it in the view
    }

}
```

ANEXO E: Códidgo Java para definição do ambiente utilizado no exemplo estudado.

```
package cleaning_robots;

import jason.asSyntax.*;
import jason.environment.Environment;
import jason.environment.grid.GridWorldModel;
import jason.environment.grid.GridWorldView;
import jason.environment.grid.Location;

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.io.IOException;
import java.util.Random;
import java.util.Vector;
import java.util.logging.Logger;

import javabayes.BayesianNetworks.BayesNet;
import javabayes.InferenceGraphs.InferenceGraph;
import javabayes.InferenceGraphs.InferenceGraphNode;
import javabayes.InterchangeFormat.IFException;
import javabayes.QuasiBayesianInferences.QBInference;

import java.io.File;
import java.util.Date;
import jxl.*;
import jxl.read.biff.BiffException;
import jxl.write.*;
import jxl.write.Number;
import jxl.write.biff.RowsExceededException;

public class MarsEnv extends Environment {
```

```
public static final int GSize = 7; // grid size
public static final int GARB  = 16; // garbage code in grid model

public static final Term    ns = Literal.parseLiteral("next(slot)");
public static final Term    pg = Literal.parseLiteral("pick(garb)");
public static final Term    dg = Literal.parseLiteral("drop(garb)");
public static final Term    bg = Literal.parseLiteral("burn(garb)");
public static final Literal g1 = Literal.parseLiteral("garbage(r1)");
public static final Literal g2 = Literal.parseLiteral("garbage(r2)");

static Logger logger = Logger.getLogger(MarsEnv.class.getName());

private MarsModel model;
private MarsView  view;

@Override
public void init(String[] args)  {
   model = new MarsModel(0);
   view  = new MarsView(model);
   model.setView(view);
   updatePercepts();
}

public void init(int count, int ns_count)  {
```

// Porção de código responsável por adicionar os valores da execução anterior (número da execução e quantidade de ciclos) a planilha plan2.xls

```
try {
                Workbook workbook = Workbook.getWorkbook(new
File("plan2.xls"));
                WritableWorkbook copy =
Workbook.createWorkbook(new File("plan2.xls"), workbook);
```

```
                    WritableSheet sheet2 = copy.getSheet(0);
                    Number number = new Number(0, count, count);
                    sheet2.addCell(number);
                    Number number2 = new Number(1, count,
ns_count);
                    sheet2.addCell(number2);
                    copy.write();
                    copy.close();
                } catch (BiffException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (RowsExceededException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (WriteException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            model = new MarsModel(count);
            view  = new MarsView(model);
            model.setView(view);
            updatePercepts();
        }


        // Método responsável pelo escanolamento de ações, ativando o
método correspondente a ação a qual será realizada
        @Override
        public boolean executeAction(String ag, Structure action) {
```

```
    logger.info(ag+" doing: "+ action);
    try {
        if (action.equals(ns)) {
                model.nextSlot();
        } else if (action.getFunctor().equals("move_towards")) {
            int x = (int)((NumberTerm)action.getTerm(0)).solve();
            int y = (int)((NumberTerm)action.getTerm(1)).solve();
            model.moveTowards(x,y);
        } else if (action.equals(pg)) {
            model.pickGarb();
        } else if (action.equals(dg)) {
            model.dropGarb();
        } else if (action.equals(bg)) {
            model.burnGarb();
        } else {
            return false;
        }
    } catch (Exception e) {
        e.printStackTrace();
    }

    updatePercepts();

    try {
        Thread.sleep(200);
    } catch (Exception e) {}
    informAgsEnvironmentChanged();
    return true;
}

/** creates the agents perception based on the MarsModel */
void updatePercepts() {
    clearPercepts();
```

```
            Location r1Loc = model.getAgPos(0);
            Location r2Loc = model.getAgPos(1);

            Literal pos1 = Literal.parseLiteral("pos(r1," + r1Loc.x + "," + r1Loc.y
+ ")");
            Literal pos2 = Literal.parseLiteral("pos(r2," + r2Loc.x + "," + r2Loc.y
+ ")");

            addPercept(pos1);
            addPercept(pos2);

            if (model.hasObject(GARB, r1Loc)) {
                addPercept(g1);
            }
            if (model.hasObject(GARB, r2Loc)) {
                addPercept(g2);
            }
        }


        class MarsModel extends GridWorldModel {

            public static final int MErr = 2; // max error in pick garb
            int nerr; // number of tries of pick garb
            boolean r1HasGarb = false; // whether r1 is carrying garbage or not
            Random random = new Random(System.currentTimeMillis());
            int step_count = 0; //conta o número de ppassos do agente sem
encongtrar uma unidade de lixo
            int stop = 0; // inteiro que controla quando o gente deve parar para
simular a velocidade
            int stop_bad = 0; // inteiro que controla o momento em que o agente
deve ficar parado devido a uma emoção ruim
            int ns_count = 0; //conta o número de vezes que a função nextSlot é
chamada
```

```
int count;

// definição dos nós da rede de emoções
InferenceGraph rede_emocoes = new InferenceGraph();

InferenceGraphNode Valenced_Reaction_to;
        InferenceGraphNode Consequences_of_Events;
InferenceGraphNode Consequences_for_Others;
InferenceGraphNode Consequences_for_Self;
InferenceGraphNode Desirable_for_Other;
InferenceGraphNode Prospect_Relevant;
InferenceGraphNode Happy_for;
InferenceGraphNode Resentment;
InferenceGraphNode Gloating;
InferenceGraphNode Pity;
InferenceGraphNode Hope;
InferenceGraphNode Fear;
InferenceGraphNode Joy;
InferenceGraphNode Distress;
InferenceGraphNode Hope_Confirmed;
InferenceGraphNode Fear_Confirmed;
InferenceGraphNode Satisfaction;
InferenceGraphNode Disappointment;
InferenceGraphNode Fears_Confirmed;
InferenceGraphNode Relief;
InferenceGraphNode Actions_of_Agents;
InferenceGraphNode Self_Agent;
InferenceGraphNode Other_Agent;
InferenceGraphNode Pride;
InferenceGraphNode Shame;
InferenceGraphNode Admiration;
InferenceGraphNode Reproach;
InferenceGraphNode Gratification;
InferenceGraphNode Remorse;
```

```
InferenceGraphNode Gratitude;
InferenceGraphNode Anger;
InferenceGraphNode Aspects_of_Objects;
InferenceGraphNode Love;
InferenceGraphNode Hate;

// variáveis que receberão as probabilidades das emoções
double belief1, belief2, belief3, belief4, belief5, belief6, belief7, belief8, belief9, belief10, belief11, belief12, belief13, belief14, belief15, belief16, belief17, belief18, belief19, belief20, belief21, belief22;

// definições necessaria para impressão em um arquivo do tipo .xls
WritableWorkbook workbook;
WritableSheet sheet0;
Number number;


private MarsModel(int a) {
   super(GSize, GSize, 2);

   // initial location of agents
   try {
      setAgPos(0, 0, 0);

      Location r2Loc = new Location(GSize/2, GSize/2);
      setAgPos(1, r2Loc);
   } catch (Exception e) {
      e.printStackTrace();
   }

   // initial location of garbage

   int x, y, number_garb=0;
   Random gerador = new Random();
```

```
while(number_garb<6){
        x = gerador.nextInt(GSize);
        y = gerador.nextInt(GSize);
        if(x!=GSize/2 || y!=GSize/2){
                add(GARB, x, y);
                number_garb++;
        }
}
//add(GARB, 3, 0);
//add(GARB, GSize-1, 0);
//add(GARB, 1, 2);
//add(GARB, 0, GSize-2);
//add(GARB, GSize-1, GSize-1);

/**
 * ################### Criando a rede Bayesiana de emoções
################################
 */

Valenced_Reaction_to = createNode(rede_emocoes,
"Valenced_Reaction_to", "True", "False");

        Consequences_of_Events = createNode(rede_emocoes,
"Consequences_of_Events", "Pleased", "Displeased");
Consequences_for_Others = createNode(rede_emocoes,
"Consequences_for_Others", "True", "False");
Consequences_for_Self = createNode(rede_emocoes,
"Consequences_for_Self", "True", "False");
Desirable_for_Other = createNode(rede_emocoes,
"Desirable_for_Other", "True", "False");
Prospect_Relevant = createNode(rede_emocoes,
"Prospect_Relevant", "True", "False");
```

```
        Happy_for = createNode(rede_emocoes, "Happy_for", "True",
"False");
        Resentment = createNode(rede_emocoes, "Resentment", "True",
"False");
        Gloating = createNode(rede_emocoes, "Gloating", "True",
"False");
        Pity = createNode(rede_emocoes, "Pity", "True", "False");
        Hope = createNode(rede_emocoes, "Hope", "True", "False");
        Fear = createNode(rede_emocoes, "Fear", "True", "False");
        Joy = createNode(rede_emocoes, "Joy", "True", "False");
        Distress = createNode(rede_emocoes, "Distress", "True",
"False");

        Hope_Confirmed = createNode(rede_emocoes,
"Hope_Confirmed", "True", "False");
        Fear_Confirmed = createNode(rede_emocoes,
"Fear_Confirmed", "True", "False");
        Satisfaction = createNode(rede_emocoes, "Satisfaction", "True",
"False");
        Disappointment = createNode(rede_emocoes, "Disappointment",
"True", "False");
        Fears_Confirmed = createNode(rede_emocoes,
"Fears_Confirmed", "True", "False");
        Relief = createNode(rede_emocoes, "Relief", "True", "False");

        Actions_of_Agents = createNode(rede_emocoes,
"Actions_of_Agents", "Approving", "Disapproving");
        Self_Agent = createNode(rede_emocoes, "Self_Agent", "True",
"False");
        Other_Agent = createNode(rede_emocoes, "Other_Agent",
"True", "False");

        Pride = createNode(rede_emocoes, "Pride", "True", "False");
        Shame = createNode(rede_emocoes, "Shame", "True", "False");
```

```
		Admiration = createNode(rede_emocoes, "Admiration", "True",
"False");
		Reproach = createNode(rede_emocoes, "Reproach", "True",
"False");

		Gratification = createNode(rede_emocoes, "Gratification", "True",
"False");
		Remorse = createNode(rede_emocoes, "Remorse", "True",
"False");
		Gratitude = createNode(rede_emocoes, "Gratitude", "True",
"False");
		Anger = createNode(rede_emocoes, "Anger", "True", "False");

		Aspects_of_Objects = createNode(rede_emocoes,
"Aspects_of_Objects", "Liking", "Disliking");
		Love = createNode(rede_emocoes, "Love", "True", "False");
		Hate = createNode(rede_emocoes, "Hate", "True", "False");

		rede_emocoes.create_arc(Valenced_Reaction_to,
Consequences_of_Events);
		rede_emocoes.create_arc(Valenced_Reaction_to,
Actions_of_Agents);
		rede_emocoes.create_arc(Valenced_Reaction_to,
Aspects_of_Objects);

		rede_emocoes.create_arc(Consequences_of_Events,
Consequences_for_Others);
		rede_emocoes.create_arc(Consequences_of_Events,
Consequences_for_Self);
		rede_emocoes.create_arc(Consequences_of_Events,
Happy_for);
		rede_emocoes.create_arc(Consequences_of_Events,
Resentment);
		rede_emocoes.create_arc(Consequences_of_Events, Gloating);
```

```
rede_emocoes.create_arc(Consequences_of_Events, Pity);
rede_emocoes.create_arc(Consequences_of_Events, Hope);
rede_emocoes.create_arc(Consequences_of_Events, Fear);
rede_emocoes.create_arc(Consequences_of_Events, Joy);
rede_emocoes.create_arc(Consequences_of_Events, Distress);


rede_emocoes.create_arc(Consequences_for_Others,
Desirable_for_Other);
rede_emocoes.create_arc(Consequences_for_Self,
Prospect_Relevant);

rede_emocoes.create_arc(Desirable_for_Other, Happy_for);
rede_emocoes.create_arc(Desirable_for_Other, Resentment);
rede_emocoes.create_arc(Desirable_for_Other, Gloating);
rede_emocoes.create_arc(Desirable_for_Other, Pity);

rede_emocoes.create_arc(Prospect_Relevant, Hope);
rede_emocoes.create_arc(Prospect_Relevant, Fear);
rede_emocoes.create_arc(Prospect_Relevant, Joy);
rede_emocoes.create_arc(Prospect_Relevant, Distress);

rede_emocoes.create_arc(Hope, Hope_Confirmed);
rede_emocoes.create_arc(Fear, Fear_Confirmed);

rede_emocoes.create_arc(Hope_Confirmed, Satisfaction);
rede_emocoes.create_arc(Hope_Confirmed, Disappointment);

rede_emocoes.create_arc(Fear_Confirmed, Fears_Confirmed);
rede_emocoes.create_arc(Fear_Confirmed, Relief);

rede_emocoes.create_arc(Actions_of_Agents, Self_Agent);
rede_emocoes.create_arc(Actions_of_Agents, Other_Agent);
```

```
rede_emocoes.create_arc(Actions_of_Agents, Pride);
rede_emocoes.create_arc(Actions_of_Agents, Shame);
rede_emocoes.create_arc(Actions_of_Agents, Admiration);
rede_emocoes.create_arc(Actions_of_Agents, Reproach);

rede_emocoes.create_arc(Self_Agent, Pride);
rede_emocoes.create_arc(Self_Agent, Shame);

rede_emocoes.create_arc(Other_Agent, Admiration);
rede_emocoes.create_arc(Other_Agent, Reproach);

rede_emocoes.create_arc(Joy, Gratification);
rede_emocoes.create_arc(Joy, Gratitude);

rede_emocoes.create_arc(Distress, Remorse);
rede_emocoes.create_arc(Distress, Anger);

rede_emocoes.create_arc(Pride, Gratification);
rede_emocoes.create_arc(Admiration, Gratitude);
rede_emocoes.create_arc(Shame, Remorse);
rede_emocoes.create_arc(Reproach, Anger);

rede_emocoes.create_arc(Aspects_of_Objects, Love);
rede_emocoes.create_arc(Aspects_of_Objects, Hate);



/**
 *  ################################### Definindo as
probabilidades da rede ############################
 */

setProbabilityValues(Valenced_Reaction_to, .5, .5);
```

```
setProbabilityValues(Consequences_of_Events, "True", .5, .5);
setProbabilityValues(Consequences_of_Events, "False", .5, .5);
setProbabilityValues(Actions_of_Agents, "True", .5, .5);
setProbabilityValues(Actions_of_Agents, "False", .5, .5);
setProbabilityValues(Aspects_of_Objects, "True", .5, .5);
setProbabilityValues(Aspects_of_Objects, "False", .5, .5);

setProbabilityValues(Consequences_for_Others, "Pleased", .5,
.5);
setProbabilityValues(Consequences_for_Others, "Displeased",
.5, .5);
setProbabilityValues(Consequences_for_Self, "Pleased", .5, .5);
setProbabilityValues(Consequences_for_Self, "Displeased", .5,
.5);

setProbabilityValues(Desirable_for_Other, "True", .5, .5);
setProbabilityValues(Desirable_for_Other, "False", .5, .5);
setProbabilityValues(Prospect_Relevant, "True", .5, .5);
setProbabilityValues(Prospect_Relevant, "False", .5, .5);

setProbabilityValues(Happy_for, "True", "Pleased", .95, .05);
setProbabilityValues(Happy_for, "True", "Displeased", .5, .5);
setProbabilityValues(Happy_for, "False", "Pleased", .5, .5);
setProbabilityValues(Happy_for, "False", "Displeased", .05, .95);

setProbabilityValues(Resentment, "Pleased", "True", .5, .5);
setProbabilityValues(Resentment, "Displeased", "True", .95, .05);
setProbabilityValues(Resentment, "Pleased", "False", .05, .95);
setProbabilityValues(Resentment, "Displeased", "False", .5, .5);

setProbabilityValues(Gloating, "Pleased", "True", .5, .5);
setProbabilityValues(Gloating, "Displeased", "True", .05, .95);
setProbabilityValues(Gloating, "Pleased", "False", .95, .05);
setProbabilityValues(Gloating, "Displeased", "False", .5, .5);
```

```
setProbabilityValues(Pity, "True", "Pleased", .05, .95);
setProbabilityValues(Pity, "True", "Displeased", .5, .5);
setProbabilityValues(Pity, "False", "Pleased", .5, .5);
setProbabilityValues(Pity, "False", "Displeased", .95, .05);

setProbabilityValues(Hope, "True", "Pleased", .95, .05);
setProbabilityValues(Hope, "True", "Displeased", .5, .5);
setProbabilityValues(Hope, "False", "Pleased", .5, .5);
setProbabilityValues(Hope, "False", "Displeased", .05, .95);

setProbabilityValues(Fear, "Pleased", "True", .5, .5);
setProbabilityValues(Fear, "Displeased", "True", .95, .05);
setProbabilityValues(Fear, "Pleased", "False", .05, .95);
setProbabilityValues(Fear, "Displeased", "False", .5, .5);

setProbabilityValues(Joy, "Pleased", "True", .5, .5);
setProbabilityValues(Joy, "Displeased", "True", .05, .95);
setProbabilityValues(Joy, "Pleased", "False", .95, .05);
setProbabilityValues(Joy, "Displeased", "False", .5, .5);

setProbabilityValues(Distress, "True", "Pleased", .05, .95);
setProbabilityValues(Distress, "True", "Displeased", .5, .5);
setProbabilityValues(Distress, "False", "Pleased", .5, .5);
setProbabilityValues(Distress, "False", "Displeased", .95, .05);

setProbabilityValues(Hope_Confirmed, "True", .5, .5);
setProbabilityValues(Hope_Confirmed, "False", .0, .0);
setProbabilityValues(Fear_Confirmed, "True", .5, .5);
setProbabilityValues(Fear_Confirmed, "False", .0, .0);

setProbabilityValues(Satisfaction, "True", .95, .05);
setProbabilityValues(Satisfaction, "False", .05, .95);
setProbabilityValues(Disappointment, "True", .05, .95);
```

```
setProbabilityValues(Disappointment, "False", .95, .05);
setProbabilityValues(Fears_Confirmed, "True", .95, .05);
setProbabilityValues(Fears_Confirmed, "False", .05, .95);
setProbabilityValues(Relief, "True", .05, .95);
setProbabilityValues(Relief, "False", .95, .05);

setProbabilityValues(Self_Agent, "Approving", .5, .5);
setProbabilityValues(Self_Agent, "Disapproving", .5, .5);
setProbabilityValues(Other_Agent, "Approving", .5, .5);
setProbabilityValues(Other_Agent, "Disapproving", .5, .5);

setProbabilityValues(Pride, "Approving", "True", .95, .05);
setProbabilityValues(Pride, "Disapproving", "True", .5, .5);
setProbabilityValues(Pride, "Approving", "False", .5, .5);
setProbabilityValues(Pride, "Disapproving", "False", .05, .95);

setProbabilityValues(Shame, "Approving", "True", .5, .5);
setProbabilityValues(Shame, "Disapproving", "True", .95, .05);
setProbabilityValues(Shame, "Approving", "False", .05, .95);
setProbabilityValues(Shame, "Disapproving", "False", .5, .5);

setProbabilityValues(Admiration, "Approving", "True", .95, .05);
setProbabilityValues(Admiration, "Disapproving", "True", .5, .5);
setProbabilityValues(Admiration, "Approving", "False", .5, .5);
setProbabilityValues(Admiration, "Disapproving", "False", .05,
.95);

setProbabilityValues(Reproach, "Approving", "True", .5, .5);
setProbabilityValues(Reproach, "Disapproving", "True", .95, .05);
setProbabilityValues(Reproach, "Approving", "False", .05, .95);
setProbabilityValues(Reproach, "Disapproving", "False", .5, .5);

setProbabilityValues(Gratification, "True", "True", .95, .05);
setProbabilityValues(Gratification, "True", "False", .5, .5);
```

```
setProbabilityValues(Gratification, "False", "True", .5, .5);
setProbabilityValues(Gratification, "False", "False", .05, .95);

setProbabilityValues(Remorse, "True", "True", .95, .05);
setProbabilityValues(Remorse, "True", "False", .5, .5);
setProbabilityValues(Remorse, "False", "True", .5, .5);
setProbabilityValues(Remorse, "False", "False", .05, .95);

setProbabilityValues(Gratitude, "True", "True", .95, .05);
setProbabilityValues(Gratitude, "True", "False", .5, .5);
setProbabilityValues(Gratitude, "False", "True", .5, .5);
setProbabilityValues(Gratitude, "False", "False", .05, .95);

setProbabilityValues(Anger, "True", "True", .95, .05);
setProbabilityValues(Anger, "True", "False", .5, .5);
setProbabilityValues(Anger, "False", "True", .5, .5);
setProbabilityValues(Anger, "False", "False", .05, .95);

setProbabilityValues(Love, "Liking", .95, .05);
setProbabilityValues(Love, "Disliking", .05, .95);
setProbabilityValues(Hate, "Liking", .05, .95);
setProbabilityValues(Hate, "Disliking", .95, .05);


// Porção de código responsável por inicializar o arquivo plan.xls
        try {
                workbook = Workbook.createWorkbook(new
File("plan.xls"));
            sheet0 = workbook.createSheet("sheet", 0);

            Label step = new Label(0, 0, "Step");
            Label happy_for = new Label(1, 0, "Happy-for");
            Label resentment = new Label(2, 0, "Resentment");
            Label gloating = new Label(3, 0, "Gloating");
```

```
Label pity = new Label(4, 0, "Pity");
Label hope = new Label(5, 0, "Hope");
Label fear = new Label(6, 0, "Fear");
Label joy = new Label(7, 0, "Joy");
Label distress = new Label(8, 0, "Distress");
Label satisfaction = new Label(9, 0, "Satisfaction");
Label disappointment = new Label(10, 0, "Disappointment");
Label fears_confirmed = new Label(11, 0, "Fears-confirmed");
Label relief = new Label(12, 0, "Relief");
Label pride = new Label(13, 0, "Pride");
Label shame = new Label(14, 0, "Shame");
Label admiration = new Label(15, 0, "Admiration");
Label reproach = new Label(16, 0, "Reproach");
Label gratification = new Label(17, 0, "Gratification");
Label remorse = new Label(18, 0, "Remorse");
Label gratitude = new Label(19, 0, "Gratitude");
Label anger = new Label(20, 0, "Anger");
Label love = new Label(21, 0, "Love");
Label hate = new Label(22, 0, "Hate");
Label goodbeliefs = new Label(23, 0 ,"Good Beliefs");
Label badbeliefs = new Label(24, 0 ,"Bad Beliefs");
Label soma = new Label(25 , 0, "Soma");


        sheet0.addCell(step);
        sheet0.addCell(happy_for);
        sheet0.addCell(resentment);
        sheet0.addCell(gloating);
        sheet0.addCell(pity);
        sheet0.addCell(hope);
        sheet0.addCell(fear);
        sheet0.addCell(joy);
        sheet0.addCell(distress);
        sheet0.addCell(satisfaction);
```

```
                    sheet0.addCell(disappointment);
                    sheet0.addCell(fears_confirmed);
                    sheet0.addCell(relief);
                    sheet0.addCell(pride);
                    sheet0.addCell(shame);
                    sheet0.addCell(admiration);
                    sheet0.addCell(reproach);
                    sheet0.addCell(gratification);
                    sheet0.addCell(remorse);
                    sheet0.addCell(gratitude);
                    sheet0.addCell(anger);
                    sheet0.addCell(love);
                    sheet0.addCell(hate);
                    sheet0.addCell(goodbeliefs);
                    sheet0.addCell(badbeliefs);
                    sheet0.addCell(soma);

                } catch (RowsExceededException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (WriteException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }

            count = a;
        }

          /**
           * Helper function to create node since not as straightforward with
JavaBayes
```

```
     * to get a pointer back to the node that is being added
     */
    private InferenceGraphNode createNode(
                    InferenceGraph ig, String name, String trueVariable,
String falseVariable) {
        ig.create_node(0, 0);
        InferenceGraphNode node = (InferenceGraphNode)
ig.get_nodes().lastElement();

        node.set_name(name);
        ig.change_values(node, new String[] {trueVariable,
falseVariable});

        return node;
    }

    /**
     * Sets probabilities for a leaf node
     */
    private void setProbabilityValues(InferenceGraphNode node,
double trueValue, double falseValue) {
        node.set_function_values(new double[] {trueValue,
falseValue});
    }

    /**
     * Returns the index of the variable for the parent that has the
given variable
     */
    private int getVariableIndex(InferenceGraphNode node, String
parentVariable) {

        for (InferenceGraphNode parent :
(Vector<InferenceGraphNode>) node.get_parents()) {
```

```
                int variableIndex = 0;

                for (String variable : parent.get_values()) {
                    if (variable.equals(parentVariable)) {
                        return variableIndex;
                    }

                    variableIndex++;
                }
            }

            return 0;
        }

        /**
         * Returns the total number of values for the parent that has the given variable
         */
        private int getTotalValues(InferenceGraphNode node, String parentVariable) {
            for (InferenceGraphNode parent : (Vector<InferenceGraphNode>) node.get_parents()) {

                for (String variable : parent.get_values()) {
                    if (variable.equals(parentVariable)) {
                        return parent.get_number_values();
                    }
                }
            }

            return 0;
        }

        /**
```

```
        * Sets probabilities for a node that has a parent
        */
        private void setProbabilityValues(InferenceGraphNode node,
String parentVariable,
                double trueValue, double falseValue) {
            int variableIndex = getVariableIndex(node, parentVariable);
            int totalValues = getTotalValues(node, parentVariable);

            double[] probabilities = node.get_function_values();
            probabilities[variableIndex] = trueValue;
            probabilities[variableIndex + totalValues] = falseValue;
            node.set_function_values(probabilities);
        }

        /**
        * Sets probabilities for a node that has two parents
        */
        private void setProbabilityValues(InferenceGraphNode node,
String firstParentVariable,
                String secondParentVariable, double trueValue,
double falseValue) {

            int variableIndex = (getVariableIndex(node,
firstParentVariable) * 2) +
                    getVariableIndex(node,
secondParentVariable);
            int totalValues = getTotalValues(node, firstParentVariable)
+
                    getTotalValues(node, secondParentVariable);

            double[] probabilities = node.get_function_values();
            probabilities[variableIndex] = trueValue;
            probabilities[variableIndex + totalValues] = falseValue;
            node.set_function_values(probabilities);
```

```
        }

        /**
         * Gets the belief/true result from the inference of the given node
         */
        private double getBelief(InferenceGraph ig, InferenceGraphNode
node) {
                QBInference qbi = new QBInference(ig.get_bayes_net(),
false);
                qbi.inference(node.get_name());
                return qbi.get_result().get_value(0);
        }

      // Método para que o agente avance uma posição no mapa
      void nextSlot() throws Exception{

        ns_count++;

        /**
         * ############################  Obtendo as probabilidades
atuais das emoções  ###################################
         */


        try{
                belief1 = getBelief(rede_emocoes, Happy_for);
                belief2 = getBelief(rede_emocoes, Pity);
                belief3 = getBelief(rede_emocoes, Hope);
                belief4 = getBelief(rede_emocoes, Joy);
                belief5 = getBelief(rede_emocoes, Satisfaction);
                belief6 = getBelief(rede_emocoes, Relief);
                belief7 = getBelief(rede_emocoes, Gratification);
                belief8 = getBelief(rede_emocoes, Gratitude);
                belief9 = getBelief(rede_emocoes, Pride);
```

```
                belief10 = getBelief(rede_emocoes, Admiration);
                belief11 = getBelief(rede_emocoes, Love);
                belief12= getBelief(rede_emocoes, Resentment);
                belief13 = getBelief(rede_emocoes, Gloating);
                belief14 = getBelief(rede_emocoes, Fear);
                belief15 = getBelief(rede_emocoes, Distress);
                belief16 = getBelief(rede_emocoes, Disappointment);
                belief17 = getBelief(rede_emocoes, Fears_Confirmed);
                belief18 = getBelief(rede_emocoes, Remorse);
                belief19 = getBelief(rede_emocoes, Anger);
                belief20 = getBelief(rede_emocoes, Shame);
                belief21 = getBelief(rede_emocoes, Reproach);
                belief22 = getBelief(rede_emocoes, Hate);
        } catch (NullPointerException e) {
                        e.printStackTrace();
        }

         double good_beliefs = belief1 + belief2 + belief3 + belief4 +
belief5 + belief6 + belief7 + belief8 + belief9 + belief10 + belief11;
         double bad_beliefs = belief12 + belief13 + belief14 + belief15 +
belief16 + belief17 + belief18 + belief19 + belief20 + belief21 + belief22;

                        Location loc = getAgPos(0);
                        if (loc.x < GSize-1 || loc.y < GSize-1){
                                try {
                                        // imprimindo as probabilidades das
emoções no arquivo plan.xls
                        number = new Number(0,ns_count,ns_count);
                                sheet0.addCell(number);
                                number = new Number(1,ns_count,belief1);
                                sheet0.addCell(number);
                                number = new Number(2,ns_count,belief12);
                                sheet0.addCell(number);
                                number = new Number(3,ns_count,belief13);
```

```
sheet0.addCell(number);
number = new Number(4,ns_count,belief2);
sheet0.addCell(number);
number = new Number(5,ns_count,belief3);
sheet0.addCell(number);
number = new Number(6,ns_count,belief14);
sheet0.addCell(number);
number = new Number(7,ns_count,belief4);
sheet0.addCell(number);
number = new Number(8,ns_count,belief15);
sheet0.addCell(number);
number = new Number(9,ns_count,belief5);
sheet0.addCell(number);
number = new Number(10,ns_count,belief16);
sheet0.addCell(number);
number = new Number(11,ns_count,belief17);
sheet0.addCell(number);
number = new Number(12,ns_count,belief6);
sheet0.addCell(number);
number = new Number(13,ns_count,belief9);
sheet0.addCell(number);
number = new Number(14,ns_count,belief20);
sheet0.addCell(number);
number = new Number(15,ns_count,belief10);
sheet0.addCell(number);
number = new Number(16,ns_count,belief21);
sheet0.addCell(number);
number = new Number(17,ns_count,belief7);
sheet0.addCell(number);
number = new Number(18,ns_count,belief18);
sheet0.addCell(number);
number = new Number(19,ns_count,belief8);
sheet0.addCell(number);
number = new Number(20,ns_count,belief19);
```

```
                    sheet0.addCell(number);
                    number = new Number(21,ns_count,belief11);
                    sheet0.addCell(number);
                    number = new Number(22,ns_count,belief22);
                    sheet0.addCell(number);
                    number = new Number(23,ns_count,good_beliefs);
                    sheet0.addCell(number);
                    number = new Number(24,ns_count,bad_beliefs);
                    sheet0.addCell(number);
                    number = new Number(25,ns_count,good_beliefs - bad_beliefs);
                    sheet0.addCell(number);

                } catch (RowsExceededException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (WriteException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }

                }else{

                    try {
                        // imprimindo as probabilidades das emoções no arquivo plan.xls
                number = new Number(0,ns_count,ns_count);
                    sheet0.addCell(number);
                    number = new Number(1,ns_count,belief1);
                    sheet0.addCell(number);
                    number = new Number(2,ns_count,belief12);
                    sheet0.addCell(number);
```

```
number = new Number(3,ns_count,belief13);
sheet0.addCell(number);
number = new Number(4,ns_count,belief2);
sheet0.addCell(number);
number = new Number(5,ns_count,belief3);
sheet0.addCell(number);
number = new Number(6,ns_count,belief14);
sheet0.addCell(number);
number = new Number(7,ns_count,belief4);
sheet0.addCell(number);
number = new Number(8,ns_count,belief15);
sheet0.addCell(number);
number = new Number(9,ns_count,belief5);
sheet0.addCell(number);
number = new Number(10,ns_count,belief16);
sheet0.addCell(number);
number = new Number(11,ns_count,belief17);
sheet0.addCell(number);
number = new Number(12,ns_count,belief6);
sheet0.addCell(number);
number = new Number(13,ns_count,belief9);
sheet0.addCell(number);
number = new Number(14,ns_count,belief20);
sheet0.addCell(number);
number = new Number(15,ns_count,belief10);
sheet0.addCell(number);
number = new Number(16,ns_count,belief21);
sheet0.addCell(number);
number = new Number(17,ns_count,belief7);
sheet0.addCell(number);
number = new Number(18,ns_count,belief18);
sheet0.addCell(number);
number = new Number(19,ns_count,belief8);
sheet0.addCell(number);
```

```
                    number = new Number(20,ns_count,belief19);
                    sheet0.addCell(number);
                    number = new Number(21,ns_count,belief11);
                    sheet0.addCell(number);
                    number = new Number(22,ns_count,belief22);
                    sheet0.addCell(number);
                    number = new Number(23,ns_count,good_beliefs);
                    sheet0.addCell(number);
                    number = new Number(24,ns_count,bad_beliefs);
                    sheet0.addCell(number);
                    number = new Number(25,ns_count,good_beliefs - bad_beliefs);
                    sheet0.addCell(number);
                        workbook.write();
                        workbook.close();

                } catch (RowsExceededException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (WriteException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                    // porção de código responsável pela reinicialização do exemplo para a execução de uma nova instância
                    count++;
                    if (count < 100){
```

```
        System.out.println("################################## "+ count + "
##########################################");
                                        init(count, ns_count);
                                }else{
                                        System.exit(0);
                                }

                        }



                        // teste para verificar se o agente possui estado
emocional positivo
                if(good_beliefs - bad_beliefs > 0.1 ){
                        stop = 1;
                }
                        // teste para verificar se o agente possui estado
emocional positivo
                if ((good_beliefs - bad_beliefs < -0.1) && (stop_bad == 0)){
                        System.out.println("PARADO");
                                stop_bad++;
                                step_count++;
                                return;
                }
                        // teste para verificar se o agente deve realizar a
função ou ficar parado para simular a velocidade
                if(stop == 0){
                        System.out.println("PARADO");
                                stop++;
                }else{
                        step_count++;
                        stop = 0;
                        stop_bad = 0;
```

```
                    //teste do número de passos sem encontrar uma unidade
de lixo
                    if (step_count>=5){
                            // influências sobre a rede por não encontrar unidade
de lixo

        Consequences_of_Events.set_observation_value("Displeased");
                    Consequences_for_Self.set_observation_value("True");
                }

                Location r1 = getAgPos(0);
                r1.x++;
                if (r1.x == getWidth()) {
                    r1.x = 0;
                    r1.y++;
                }
                // finished searching the whole grid
                if (r1.y == getHeight()) {
                    return;
                }
                setAgPos(0, r1);
                setAgPos(1, getAgPos(1)); // just to draw it in the view
            }

        }

        //Método para que o agente caminhe até uma determinada posição
         void moveTowards(int x, int y) throws Exception {

            Location r1 = getAgPos(0);
            if (r1.x < x)
                r1.x++;
            else if (r1.x > x)
                r1.x--;
```

```
    if (r1.y < y)
        r1.y++;
    else if (r1.y > y)
        r1.y--;
    setAgPos(0, r1);
    setAgPos(1, getAgPos(1)); // just to draw it in the view

  }

//Método para que o agente pegue a unidade de lixo
  void pickGarb() {

    // r1 location has garbage
    if (model.hasObject(GARB, getAgPos(0))) {
        // sometimes the "picking" action doesn't work
        // but never more than MErr times
        if (random.nextBoolean() || nerr == MErr) {
            remove(GARB, getAgPos(0));
            nerr = 0;
            r1HasGarb = true;
        } else {
            nerr++;
        }
    }
    step_count=0;
   // influências sobre a rede por encontrar unidade de lixo
    Consequences_of_Events.set_observation_value("Pleased");
    Consequences_for_Self.set_observation_value("True");

  }

  //Método para que o agente largue a unidade de lixo
  void dropGarb() {
```

```
      int a;
      Random gerador = new Random();
      a = gerador.nextInt(100);

      // teste para o agente receber ou não motivação do outro agente
      if (a<50){
            System.out.println("R2 Motivou R1");
            //influências sobre a rede pelo agente ter sido motivado
            Actions_of_Agents.set_observation_value("Approving");
            Other_Agent.set_observation_value("True");
      }

       if (r1HasGarb) {
          r1HasGarb = false;
          add(GARB, getAgPos(0));
       }
    }

    //Método para queima de uma unidade de lixo
    void burnGarb() {

       // r2 location has garbage
       if (model.hasObject(GARB, getAgPos(1))) {
          remove(GARB, getAgPos(1));
       }
    }
}

class MarsView extends GridWorldView {

    /**
            *
            */
            private static final long serialVersionUID = 1L;
```

```
            public MarsView(MarsModel model) {
        super(model, "Mars World", 600);
        defaultFont = new Font("Arial", Font.BOLD, 18); // change default
font
        setVisible(true);
        repaint();
    }

    /** draw application objects */
    @Override
    public void draw(Graphics g, int x, int y, int object) {
        switch (object) {
            case MarsEnv.GARB: drawGarb(g, x, y);  break;
        }
    }

    @Override
    public void drawAgent(Graphics g, int x, int y, Color c, int id) {
        String label = "R"+(id+1);
        c = Color.blue;
        if (id == 0) {
            c = Color.yellow;
            if (((MarsModel)model).r1HasGarb) {
                label += " - G";
                c = Color.orange;
            }
        }
        super.drawAgent(g, x, y, c, -1);
        if (id == 0) {
            g.setColor(Color.black);
        } else {
            g.setColor(Color.white);
        }
```

```
            super.drawString(g, x, y, defaultFont, label);
        }

        public void drawGarb(Graphics g, int x, int y) {
            super.drawObstacle(g, x, y);
            g.setColor(Color.white);
            drawString(g, x, y, defaultFont, "G");
        }

    }
}
```