Avaliação de Políticas Abstratas na Transferência de Conhecimento em Navegação Robótica

R. L. Beirigo, F. A. Pereira, M. L. Koga, T. Matos, V. F. da Silva, A. H. Reali Costa

***Abstract*— This paper presents a new approach to the problem of solving a new task by the use of previous knowledge acquired during the process of solving a similar task in the same domain, robot navigation A new algorithm, Qab-Learning is proposed to obtain the abstract policy that will guide the agent in the task of reaching a goal location from any other location in the environment, and this policy is compared to the policy derived from another algorithm, ND-TILDE. The policies are applied in a number of different tasks in two environments. The results show that the policies, even after the process of abstraction, present a positive impact on the performance of the agent.**

***Keywords*— relational representation, abstract policy, reinforcement learning, inductive logic programming, robotics.**

I. Introdução

ATUALMENTE robôs móveis encontram-se presentes nos mais diversos locais, auxiliando em serviços de entrega e limpeza em hospitais, escritórios e residências. Entretanto, a difusão de seu uso e completa aceitação só se dará, de fato, se os robôs forem capazes de rapidamente aprenderem novas tarefas e de reutilizarem o conhecimento previamente adquirido em novas atividades, ou seja, se demonstrarem boas autonomia e capacidade de adaptação a novas situações. Apesar da diversidade das soluções encontradas na literatura para a navegação autônoma de robôs móveis [1, 2, 3], elas tipicamente não consideram soluções previamente encontradas para problemas similares. Isso impede que o robô tenha a habilidade de melhorar seu desempenho por meio do aprendizado decorrente de experiências passadas em tarefas similares. Além disso, os planos de navegação gerados por estas soluções não são facilmente comunicados a humanos, tornando menos eficiente a interação entre humanos e robôs. Representações baseadas em Lógica de Primeira Ordem podem facilitar tal comunicação [4].

Tem havido um grande e recente interesse em abordar o problema de adaptação em tarefas de navegação e planejamento em robôs móveis, utilizando uma linguagem mais expressiva. Diversas técnicas têm sido exploradas, desde a construção de descrições relacionais de macroações (planos parciais) para transferência e reuso no aprendizado de novas políticas de atuação [5] até a utilização de aprendizes simbólicos proposicionais para generalizar a política ótima para outros problemas [6].

Este artigo tem por objetivo propor uma técnica de abstração do conhecimento adquirido por um robô na solução de um determinado problema, para permitir sua transferência e uso em um outro novo problema, similar ao anterior. Além disso, visa avaliar comparativamente esta nova técnica com uma proposta anteriormente na literatura [7] [8], em aplicações de navegação robótica em ambientes internos de edifícios.

O artigo está organizado da seguinte forma. A seção II apresenta os conceitos de Processos Markovianos de Decisão e Lógica de Primeira Ordem. As seções III e IV descrevem duas técnicas para se aprender políticas abstratas, ND-TILDE e Qab-Learning. A seção V descreve como se dá a aplicação das políticas em um caso de navegação robótica e os resultados obtidos dos experimentos são discutidos na seção VI. Finalmente, na seção VII estão as conclusões.

II. Representação Relacional

Uma abordagem tradicional de formalização para problemas estocásticos de decisão sequencial consiste no uso de Processos Markovianos de Decisão (MDP – *Markov Decision Process*), que podem ser expandidos para MDPs relacionais, ou RMDPs (*Relational Markov Decision Process*), a fim de permitir descrições através de Lógica de Primeira Ordem.

Um MDP pode ser definido formalmente pela quádrupla <S, A, T, R> [4, 9], onde S e A são conjuntos finitos de estados do ambiente e ações que o agente pode realizar, respectivamente, sendo o conjunto de ações aplicáveis no estado $s \in S$ denotado por A(s). $T: S \times A \times S \rightarrow [0, 1]$ é a função de transição de estado, com $T(s_t, a_t, s_{t+1})$ definindo a probabilidade de realizar a transição do estado s_t para o estado s_{t+1} através da execução da ação a_t e $R: S \times A \times S \rightarrow \Re$ é a função de recompensa, com $r_t = R(s_t, a_t, a_{t+1})$. Dessa forma, uma transição do estado $i \in S$ para $j \in S$, decorrente da execução de alguma ação $a \in A(i)$, ocorre com probabilidade T(i, a, j), e então uma recompensa R(i,a,j) é recebida.

Em um MDP, pode-se utilizar Lógica de Primeira Ordem na descrição de estados e ações, aplicando variáveis para evidenciar as relações entre os objetos, transformando-o em um RMDP [10, 11]. Na lógica relacional, expressões da forma $p(t_1, t_2, ..., t_m)$ são átomos, com p sendo um símbolo de relação entre os termos t_i. Por exemplo, *con*(r_1, r_2) indica que os

R. L. Beirigo, Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brasil, rafaelbeirigo@usp.br.

F. A. Pereira, Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brasil, fernando.pereira2@poli.usp.br.

M. L. Koga, Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brasil, mlk@usp.br.

T. Matos, Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brasil, tiago.matos@poli.usp.br.

V. F. da Silva, EACH, Universidade de São Paulo, São Paulo, SP, Brasil, valdinei.freire@usp.br.

A. H. Reali Costa, Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brasil, anna.reali@poli.usp.br.

cômodos r_1 e r_2 estão conectados (são adjacentes e mutuamente acessíveis). Um termo pode consistir em uma variável (iniciando com letra maiúscula) ou uma constante (iniciando com letra minúscula, como r_1). Conjunções de átomos lógicos representam estados. Assim, um exemplo de estado $z \in S$ seria:

$$z \equiv in(r_1), con(r_1, r_2), room(r_1), room(r_2).$$

indicando que o robô está no cômodo r_1, o qual é conectado ao cômodo r_2. Da mesma forma, define-se o conceito de estados abstratos como conjunções de átomos lógicos que contêm variáveis. Por exemplo, o estado abstrato

$$Z \equiv in(R), con(R,R'), room(R), room(R')$$

representa estados nos quais o robô está no cômodo R, o qual é conectado a um outro cômodo R'. Neste caso, o estado z seria uma instância do estado abstrato Z. Uma substituição θ atribui termos às variáveis de forma que $Z\theta \subseteq z$. Assim, para o exemplo acima, a instância z seria alcançada com a substituição $\theta = \{R/r_1, R'/r_2\}$. Um estado é denominado concreto se não contiver variáveis. Conjunções com variáveis são consideradas existencialmente quantificadas. Uma conjunção Z é subsumida pela conjunção W, isto é, $Z \leq_\theta W$ se existir θ tal que $W\theta \subseteq Z$. Analogamente, ações abstratas podem ser representadas por átomos tendo variáveis como argumentos. Assim, *goto*(R,R') é uma ação abstrata que indica que o robô navega de um cômodo R para um cômodo R', e *goto*(r_1, r_2) é uma ação concreta com a substituição $\theta = \{R/r_1, R'/r_2\}$.

Resolver um problema modelado por um (R)MDP consiste em computar uma política π que especifica quais ações $a \in A$ devem ser executadas quando o agente está no estado $s \in S$. Se a política for não determinística, tem-se que π: $S \rightarrow 2^A$; entretanto, se a política for determinística, tem-se que π: $S \rightarrow A$, ou seja, π é determinística se leva cada estado em S a uma única ação em A. Dada a política π e um fator de desconto $\gamma \in [0, 1]$ que desconta recompensas futuras, a função de valor de estado V^π: $S \rightarrow \Re$ representa o valor de estar em um estado seguindo a política π, isto é, representa as recompensas esperadas. A política ótima π^* é a política que satisfaz a condição: $V^{\pi^*} \geq V^{\pi'}$, $\forall s \in S$ e $\forall \pi'$. A função valor ótima é denotada por V^*. Assim, o interesse consiste em computar uma política que melhor aproxime, ou, idealmente, iguale a política ótima π^*.

Da mesma forma que se definiu estados e ações abstratas no RMDP, pode-se definir uma política abstrata π_a [11] como uma lista ordenada de regras de ações abstratas na forma i: $\sigma \Rightarrow \{\alpha\}$, sendo σ um estado abstrato; $\{\alpha\}$, um conjunto de ações abstratas e i um inteiro representando a ordenação da regra.

Existem diversas formas para encontrar a política que soluciona um (R)MDP. Um exemplo é a utilização de algoritmos de planejamento de atividades, como o SPUDD [12], ou técnicas de aprendizado que utilizam repetidas iterações do agente com o ambiente, como no aprendizado por reforço [9]. Por sua vez, uma política abstrata pode ser obtida tanto por meio da generalização de uma política concreta conhecida quanto pelo aprendizado direto. Neste artigo, essas duas formas são exploradas. A primeira utiliza uma política concreta conhecida para gerar exemplos, que são agregados por meio de aprendizado supervisionado usando o algoritmo ND-TILDE [7], descrito na seção III. Já a segunda forma utiliza aprendizado por reforço por meio da abordagem proposta nesse artigo, descrita na seção IV.

III. Aprendendo políticas abstratas com aprendizado indutivo.

Esta seção descreve um método de obtenção de uma política abstrata através da indução de uma árvore de decisão em Lógica de Primeira Ordem [8] a partir de exemplos gerados por uma política concreta conhecida.

Dado um conjunto de exemplos E, onde cada exemplo consiste em um par estado-ação obtido durante a execução de uma política concreta conhecida, pode-se aplicar o algoritmo ND-TILDE, descrito em pseudo-código no Algoritmo 1. Ele induz a política abstrata a partir de E, gerando uma representação através de uma árvore binária de decisão em Lógica de Primeira Ordem [7].

Algoritmo 1: Algoritmo ND-TILDE

```
1:  função ND-TILDE (E: cj. de exemplos): retorna uma
    árvore binária de decisão
2:      T ← GERAR_TESTES(E)
3:      τ ← DIVISAO_OTIMA(T,E)
4:      E_e ← TESTE_VERDADEIRO(E,τ)
5:      E_d ← TESTE_FALSO(E,τ)
6:      se CRITERIO_PARADA(E,E_e,E_d) então
7:          retorna folha(INFO(E))
8:      senão
9:          t_e ← ND-TILDE(E_e)
10:         t_d ← ND-TILDE(E_d)
11:         retorna nó_interno(τ,t_e,t_d)
12:     fim se
13: fim função
```

ND-TILDE utiliza um conjunto de exemplos de treinamento E para criar uma árvore. Os candidatos aos testes de decisão são criados a partir de um conjunto de operadores de refinamento [8] definidos previamente por um especialista (passo 2 do Algoritmo 1). Cada um desses operadores gera um conjunto de literais de primeira ordem que é candidato para a divisão do conjunto de exemplos E fornecido. O melhor teste que pode ser aplicado para dividir o conjunto E é aquele que apresenta a maior redução de entropia. O teste ótimo é escolhido utilizando-se a taxa de ganho padrão. Este teste particiona o conjunto E de exemplos em dois outros conjuntos: E_e que contém os exemplos onde o teste τ é verdadeiro (passo 4); e E_d que contém os exemplos onde o teste τ é falso (passo 5).

Se a partição induzida em E indica que a divisão deva

parar (procedimento CRITERIO_PARADA no passo 6), uma folha contendo sentenças atômicas que representam ações abstratas é então criada na árvore (passo 7). Caso mais de uma sentença atômica representando uma ação esteja associada aos exemplos remanescentes na folha, todas elas são adicionadas pelo algoritmo a esta folha, gerando uma política não-determinística. Dessa forma, todas as ações abstratas indicadas pelos exemplos de E presentes nessa partição são associadas ao estado abstrato que corresponde à folha gerada. Se a partição induzida em E não indica que o processo deva parar, então para cada um dos elementos da partição (conjuntos E_e e E_d), a função ND-TILDE é chamada recursivamente (passos 9 e 10). Um nó interno é então criado (passo 11) usando o teste ótimo τ como teste e tendo como filhos as duas sub-árvores (τ_e e τ_d) criadas em cada chamada de ND-TILDE.

Ao término do algoritmo, cada caminho na árvore de decisão gerada da raiz até uma folha (excluindo esta) determina um estado abstrato, e cada folha possui um conjunto de ações abstratas relacionadas àquele estado. Percorrendo essa árvore sistematicamente seguindo a estratégia de busca em profundidade, obtém-se a sequência de regras ordenadas que compõem a política abstrata. A Seção V apresenta um exemplo de política abstrata para o problema da navegação em robótica obtida através da aplicação do ND-TILDE.

IV. Aprendendo políticas abstratas com aprendizado por reforço.

Nessa seção é proposto o uso de aprendizado por reforço (RL – *Reinforcement Learning*) como uma alternativa para a determinação da política abstrata que generaliza o conhecimento da solução do problema defrontado pelo robô. No RL, o aprendizado se dá por meio da interação direta do agente com o ambiente, em intervalos de tempos discretos contendo ciclos alternados de percepção e ação. Tido como o mais popular algoritmo de RL, o algoritmo Q–Learning é um algoritmo de aprendizado livre de modelos, com o qual uma política ótima π^* é aprendida iterativamente quando o modelo do sistema não é conhecido, i.e., as funções T e/ou R do (R)MDP são desconhecidas [9]. Fazendo $V^*(s) = \max_a Q^*(s,a)$, o algoritmo Q–Learning iterativamente aproxima a estimativa de Q*(s,a), da seguinte forma:

$$Q_{t+1}(s_t, a_t) \leftarrow (1-\alpha) Q_t(s_t, a_t) + \alpha \left(r_t + \gamma \max_a Q_t(s_{t+1}, a) \right), \quad (1)$$

sendo s_t o estado atual, a_t a ação realizada em s_t, r_t o reforço recebido após realizar a_t em s_t e atingir s_{t+1}, s_{t+1} o novo estado, $\gamma \in [0, 1]$ o fator de desconto e $\alpha \in]0, 1]$ a taxa de aprendizagem. O aprendizado aqui conduzido segue uma política ε-*greedy*, que consiste em aplicar uma ação totalmente aleatória quando o valor de uma variável aleatória for menor que ε (correspondendo à fase de exploração), e aplicar a política *greedy*, isto é, $\pi(s) = \arg\max_a Q(s, a)$, no caso contrário (correspondendo à fase de explotação).

Algoritmo 2: Qab-Learning

1: **função** Qab-Learning (**σ**: cj. de estados abstratos, **α**: cj. de ações abstratas, ε: fator exploração): retorna uma política abstrata
2: INICIALIZAR Qab_0
3: Observar o estado concreto s_t
4: Determinar o estado abstrato σ_t que subsume s_t, isto é, $s_t \leq_{\theta t} \sigma_t$
5: **repetir até** CONDIÇÃO_PARADA
6: **se** um número aleatório $\in [0,1] \leq \varepsilon$ **então**
7: Determinar a ação concreta a_t = RANDOM($A(s_t)$)
8: **senão**
9: Determinar ação abstrata: $\alpha_t \leftarrow \arg\max_\alpha Qab(\sigma_t, \alpha)$
10: Determinar o cj. de ações concretas subsumidas por α_t com a substituição θ_t: $\{a_t\} \leq_{\theta t} \alpha_t$
11: Fazer a_t = RANDOM($\{a_t\}$)
12: **fim se**
13: Executar a ação concreta a_t
14: Observar o novo estado concreto s_{t+1} e o reforço recebido r_t
15: Atualizar $Qab(\sigma_t, \alpha_t)$ usando a equação (1), com r_t e σ_{t+1} que subsume s_{t+1}
16: $s_t \leftarrow s_{t+1}$; $\sigma_t \leftarrow \sigma_{t+1}$
17: **fim repetir**
18: **laço para cada** $\sigma \in$ **σ faça**
19: $\pi_{abs}(\sigma) \leftarrow \arg\max_\alpha Qab(\sigma, \alpha)$
20: **fim laço**
21: **retornar** (π_{abs})
22: **fim função**

O Algoritmo 2 apresenta o pseudo-código do algoritmo Qab-Learning, que é uma modificação do algoritmo Q-Learning para aprender diretamente uma política abstrata. O algoritmo Qab-Learning consiste basicamente no Q-Learning tradicional usando a política ε-*greedy*, porém o aprendizado se dá no nível abstrato. Observa-se o estado concreto atual (passo 3) e determina-se o estado abstrato que subsume esse estado concreto e a correspondente substituição θ_t (passo 4). O ciclo de percepção e ação prossegue até atingir uma condição de parada (passo 5). Se na iteração atual uma exploração deve ser feita (passo 6), então deve-se determinar uma ação aleatória concreta do conjunto de ações aplicáveis no estado concreto atual (passo 7). Se não, fazer explotação determinando a ação abstrata recomendada pela política *greedy* para o estado abstrato que subsume o estado concreto atual (passo 9) e determinar o conjunto de ações concretas subsumidas por esta ação abstrata (passo 10). Escolher aleatoriamente uma ação concreta deste conjunto (passo 11). Executar a ação definida pela fase de exploração ou de explotação (passo 13), observar o reforço recebido e o novo estado concreto (passo 14). Determinar o estado abstrato que subsume o novo estado concreto observado e atualizar a tabela Qab no nível abstrato com esses dados (passo 15). Fazer o estado concreto e o correspondente abstrato atual como sendo o estado observado (passo 16) e repetir o ciclo. No final, retornar a política aprendida no nível abstrato. Na

próxima seção este algoritmo é aplicado no problema de navegação robótica, gerando uma política abstrata que busca generalizar a solução do problema.

V. Aplicando na Navegação Robótica

Considere agora um problema de navegação de robôs, cujo ambiente é o ilustrado na Fig. 1 (doravante ambiente 1).

Os estados deste problema podem ser descritos de acordo com o seguinte conjunto de predicados:
{*corridor*/1, *room*/1, *center*/1, *nearDoor*/1, *in*/1, *con*/2}, e o conjunto de ações com: {*gotoRDRD*/2, *gotoCDCD*/2, *gotoCCCC*/2, *gotoRCRD*/2, *gotoRDRC*/2, *gotoRDCD*/2, *gotoCDRD*/2, *gotoCCCD*/2, *gotoCDCC*/2}.

O significado de cada predicado é literal: por exemplo, se o agente estiver ocupando a localização 1 no mapa (representado como objeto l1), pode-se descrever o estado como:
s_{l1} = {*in(l1), corridor(l1), center(l1), con(l1,l5), corridor(l5), nearDoor(l5)*}

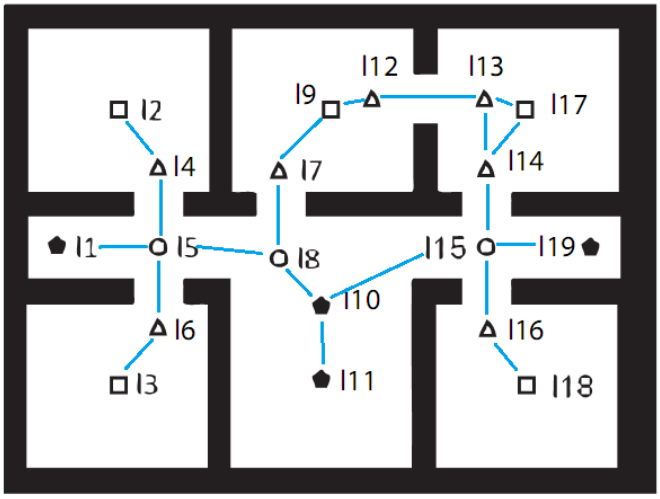


Figura 1 – Mapa do ambiente usado no aprendizado de políticas (ambiente 1). Centros e portas de cômodos são representados por quadrados e triângulos, respectivamente; círculos representam locais que estão perto de portas em corredores e pentágonos pretos representam locais nos corredores que não estão próximos de portas. As adjacências entre lugares estão indicadas por um grafo sobreposto no mapa, onde os lugares são vértices e as arestas indicam adjacências.

Se o agente faz a escolha de ir da localização *l1*, que é *corridor(l1)* e *center(l1)*, para a localização *l5*, que é *corridor(l5)* e *nearDoor(l5)*, descreve-se essa ação como *gotoCCCD(l1,l5)*. As abreviações CC e CD indicam *corridor* e *center*, *corridor* e *door*, respectivamente, e as demais seguem o mesmo padrão.

A precondição de *gotoXXYY*(Li,Lj) é (*in(Li)* ^ *con(Li,Lj)*) e ambas localizações envolvidas na ação são representadas pelas variáveis Li e Lj, as quais atendem às condições *XX* e *YY*.

A. Definindo as Políticas Abstratas

O problema que o agente deve resolver é episódico e consiste em atingir um estado-meta. Esse estado-meta foi modelado como sendo o único estado para o qual o agente recebe um reforço positivo, independentemente da posição inicial.

Dado este domínio, a tarefa utilizada para a indução das duas políticas abstratas foi a de ir de qualquer local no ambiente 1 para a localidade *l2*. Ou seja, o objetivo é chegar a *l2*, partindo de qualquer outro lugar no mapa. A primeira política, π_{ND}, foi criada com o algoritmo ND-TILDE, fornecido na Seção III (Algoritmo 1), e a segunda, π_{Qab}, com o algoritmo Qab-Learning, fornecido na Seção IV (Algoritmo 2).

Para a política π_{ND}, em primeiro lugar descobriu-se a política ótima concreta para o ambiente 1. Dada essa política conhecida, aplicou-se o ND-TILDE para a extração da política abstrata π_{ND} em forma de árvore de decisão. A árvore pode também ser descrita como um conjunto de regras ordenadas, que estão ilustradas na Tabela I.

Nota-se que existem 8 estados abstratos nessa política, pois esses foram os estados que o algoritmo julgou serem as mais relevantes partições do problema. Além disso, cada estado aponta para um conjunto de ações abstratas. Na implementação dos experimentos deste artigo, descritos na Seção VI, a escolha entre uma das ações deste conjunto é feita aleatoriamente.

Já a segunda política, π_{Qab}, foi construída de forma diferente. Cada descrição de estado concreto (a descrição de um estado só contempla conexões com localidades vizinhas) foi transformada em um estado abstrato simplesmente substituindo por variável cada constante presente nos termos dos predicados, resultando em 12 estados abstratos diferentes, que generalizam os 19 estados concretos do problema. Em posse desse conjunto de estados abstratos e do conjunto de ações abstratas, o algoritmo Qab-Learning (Algoritmo 2) foi usado para gerar uma política abstrata π_{Qab}. Para o estado abstrato em que i=7, foram encontradas duas ações abstratas ótimas, sendo os experimentos executados com a aplicação de ambas. O resultado pode ser verificado na Tabela II.

Tabela I
Política Abstrata construída com o ND-TILDE

i	Estado Abstrato	Ações Abstratas
1	*in(A), con(A,B), center(B), corridor(B), center(A), corridor(A), con(A,C), nearDoor(C), corridor(C)*	*{gotoCCDC(A,C)}*
2	*in(A), con(A,B), center(B), corridor(B), center(A), corridor(A)*	*{gotoCCCC(A,B)}*
3	*in(A), con(A,B), center(B), corridor(B), con(A,C), nearDoor(C), corridor(C)*	*{gotoCDRD(A,D), gotoCDCD(A,C)}*
4	*in(A), con(A,B), center(B), corridor(B)*	*{gotoCDCC(A,B)}*
5	*in(A), nearDoor(C), corridor(C), center(A,) corridor(A)*	*{gotoCCCD(A,C)}*
6	*in(A), nearDoor(C), corridor(C)*	*{gotoRDRC(A,D) gotoRDCD(A,C)}*
7	*in(A), room(A), nearDoor(A)*	*{gotoRDRC(A,D), gotoRDRD(A,D)}*
8	*in(A)*	*{gotoRCRD(A,D)}*

TABELA II
POLÍTICA ABSTRATA CONSTRUÍDA COM QAB-LEARNING

i	Estado Abstrato	Ações Abstratas
1	*in(A), con(A,B), con(A,C), con(A,D), con(A,E), nearDoor(A), corridor(A), center(B), corridor(B), nearDoor(C), room(C), nearDoor(D), room(D), nearDoor(E), corridor(E)*	*gotoCDRD(A,D)*
2	*in(A), con(A,B), con(A,C), con(A,D), con(A,E), nearDoor(), corridor(A), center(B), corridor(B), nearDoor(), room(C), nearDoor(D), room(D) , center(E), corridor(E)*	*gotoCDCC(A,B)*
3	*in(A), con(A,B), con(A,C), con(A,D), nearDoor(A), corridor(A), nearDoor(B), corridor(B), nearDoor(C), room(C), center(D), corridor(D)*	*gotoCDCD(A,B)*
4	*in(A), con(A,B), con(A,C), con(A,D), center(A), corridor(A), nearDoor(B), corridor(B), center(C), corridor(C), nearDoor(D), corridor(D)*	*gotoCCCD(A,B)*
5	*in(A), con(A,B), nearDoor(A), room(A), center(B), room(B), con(A,C), nearDoor(C), room(C), con(A,D), nearDoor(D), room(D)*	*gotoRDRD(A,C)*
6	*in(A), con(A,B), nearDoor(A), room(A), center(B), room(B), con(A,C), nearDoor(C), room(C), con(A,D), nearDoor(D), corridor(D)*	*gotoRDCD(A,D)*
7	*in(A), con(A,B), nearDoor(A), room(A), center(B), room(B), con(A,C), nearDoor(C), corridor(C)*	*gotoRDRC(A,D) ou gotoRDCD(A,D)*
8	*in(A), con(A,B), con(A,C), center(A), room(A), nearDoor(B), room(B), nearDoor(C), room(C)*	*gotoRCRD(A,B)*
9	*in(A), con(A,B), con(A,C), nearDoor(A), room(A), center(B), room(B), nearDoor(C), room(C)*	*gotoRDRC(A,B)*
10	*in(A), con(A,B), center(A), corridor(A), nearDoor(B), corridor(B)*	*gotoCCDC(A,B)*
11	*in(A), con(A,B), center(A), room(A), nearDoor(B), room(B)*	*gotoRCRD(A,B)*
12	*in(A), con(A,B), center(A), corridor(A), center(B), corridor(B)*	*gotoCCCC(A,B)*

B. Aplicando as Políticas Abstratas

O objetivo da construção dessas políticas abstratas é possibilitar seus usos em outros problemas similares, avaliando a extensão da transferência desse conhecimento adquirido. Para essa aplicação em robótica móvel, as políticas geradas foram testadas no mesmo ambiente 1, porém para tarefas distintas, e também em um outro ambiente (ambiente 2), maior que o anterior e cujo mapa pode ser observado na Fig. 2. Para a escolha deste problema "similar", foram consideradas duas características para serem mantidas fixas entre os problemas: o conjunto de predicados que descrevem os estados e o conjunto de ações abstratas.

A aplicação da política abstrata em um problema ocorre da seguinte forma: primeiro, observa-se o estado concreto s_t no qual o agente se encontra. Busca-se nas regras que definem a política abstrata em uso (π_{ND} ou π_{Qab}), na ordem, qual estado abstrato σ_t (antecedente da regra) que subsume o estado s_t, isto é, $s_t \leq_\theta \sigma_t$, definindo a regra e a respectiva substituição θ, obtendo-se, assim, o correspondente conjunto de ações abstratas, já com a respectiva substituição θ aplicada. Em posse deste conjunto de ações abstratas instanciadas com a substituição θ, uma delas é escolhida aleatoriamente (chamada aqui de α_θ). Caso esta ação α_θ escolhida já seja uma ação concreta (devido à substituição aplicada), esta ação concreta a_t é executada em s_t; caso contrário, ela definirá um conjunto $A\alpha_\theta$ de ações concretas que poderão ser aplicadas em s_t (substituindo adequadamente todas as variáveis restantes em α_θ), com $A\alpha_\theta \subseteq A(s_t)$, e uma escolha aleatória é feita neste conjunto $A\alpha_\theta$ para definir a ação concreta a_t a ser executada em s_t. Para o caso de aplicar-se em um outro ambiente a política abstrata gerada para um determinado ambiente, é possível que o estado concreto atual s_t não seja subsumido por nenhum estado abstrato da política abstrata em uso; neste caso, a implementação corrente escolhe aleatoriamente uma ação a_t no conjunto $A(s_t)$ e executa a_t em s_t. Este processo repete-se para o novo estado s_{t+1} alcançado com a aplicação de a_t em s_t, até que s_{t+1} seja um estado-meta.

Para acelerar a execução da tarefa, quando o agente se depara com um estado no qual ele já passou anteriormente no mesmo episódio, todas as escolhas aleatórias citadas são feitas excluindo-se a(s) escolha(s) anterior(es). Ainda, caso o agente volte a um mesmo estado após ter tentado todas as ações possíveis indicadas pela sua política, então a política aleatória é empregada, para que o agente possa assim encontrar sua meta.

Para exemplificar este procedimento, considere que o ambiente seja o ambiente 1 (Fig.1), que esteja sendo aplicada a política abstrata π_{ND} (Tabela I) e que o agente se encontre no local *l8*, sendo

s_t = { *in(l8), con (l8,l10), center (l10), corridor (l10), con(l8, l5), nearDoor (l5), corridor (l5), con(l8, l7), nearDoor (l7), room(l7)*}.

Aplicando a política π_{ND}, verifica-se que o estado abstrato que subsume s_t é o estado da terceira regra da Tabela 1:

σ_t = { *in(A), con (A,B), center (B), corridor (B), con(A,C), nearDoor (C), corridor (C)*}

com substituição θ = { *A/l8, B/l10, C/l5*} e o conjunto de ações abstratas definido pela política π_{ND} é {*gotoCDRD(A,D), gotoCDCD(A,C)*}. Supondo que a escolha aleatória escolheu a ação abstrata *gotoCDRD(A,D)* que, com θ, resulta em $A\alpha_\theta$ = { *gotoCDRD(l8,D)* } = { *gotoCDRD(l8,l7)*}, já que *D=l7* é a única substituição possível neste caso. Assim, a ação concreta a_t = *gotoCDRD(l8,l7)* é executada em s_t, e o agente então observa o novo estado e repete o procedimento, até atingir a meta.

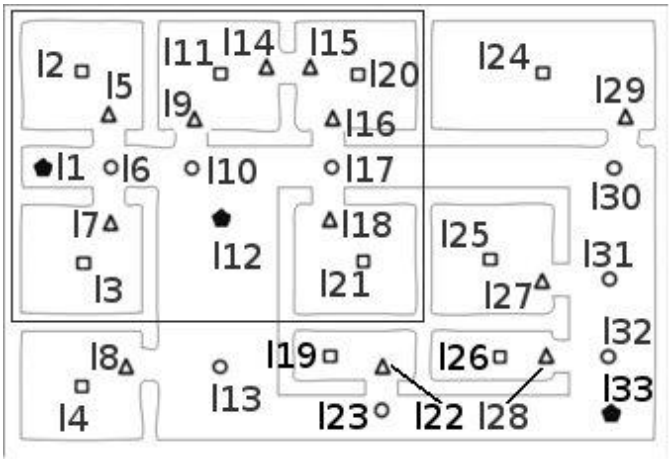


Figura 2 - Mapa do ambiente usado para o novo problema (ambiente 2). Quadrados, triângulos, círculos e pentágonos têm o mesmo significado que o explicado na Fig.1. A região marcada, por ser similar ao ambiente 1, foi utilizada nos experimentos em que a solução de um subproblema já resolvido auxiliava ou atrapalhava o agente na execução de uma nova tarefa.

VI. Experimentos e Análise comparativa.

Para avaliar as políticas abstratas aprendidas, uma série de experimentos foi realizada, usando tanto o ambiente 1, onde as políticas abstratas π_{ND} e π_{Qab} foram geradas, quanto o ambiente 2.

As três políticas utilizadas nos testas são: (i) uma política totalmente aleatória, que serve como referência, (ii) a política abstrata π_{ND} aprendida com um aprendizado supervisionado usando o ND-TILDE (Algoritmo 1) e (iii) a política abstrata π_{Qab} obtida através de aprendizado por reforço utilizando o Qab-Learning (Algoritmo 2). Os testes foram realizados com o objetivo de avaliar o desempenho das políticas na resolução de diversas tarefas.

Um episódio corresponde a iniciar o robô em um local qualquer e aplicar a política que está sendo avaliada até que ele atinja a meta ou que um limite máximo de 1.000 passos seja atingido, a partir do qual se considera que o episódio não teve sucesso. Em todos os experimentos foram executados 10.000 episódios por meta, e a média entre os conjuntos de episódios de cada meta é que é apresentada em diversos gráficos. O eixo das abscissas em cada gráfico representa o número de passos **n**, enquanto o eixo das ordenadas representa a fração acumulada de episódios, dentre os 10.000, nos quais o robô atingiu a meta (em outras palavras, se definirmos que **N** é o número de passos usados para atingir a meta, essa fração representa a distribuição acumulada de N, i.e., P(N ≤ n)). Ou seja, supondo n=100 passos no gráfico da Fig. 3, tem-se que, usando a política aleatória, uma fração de cerca de 60% dos episódios alcançaram a meta em até 100 passos, dentre os episódios que tiveram sucesso (isto é, atingiram a meta em 1.000 passos ou menos) dos 10.000 episódios executados; usando a política π_{ND} e π_{Qab}, as frações foram respectivamente 78% e 80%.

O primeiro experimento consistiu em apenas aplicar as políticas abstratas (e aleatória) no próprio problema para o qual elas foram geradas, ou seja, o mesmo ambiente (ambiente 1) com a mesma meta (alcançar o local *l2*). Isso foi feito para verificar os efeitos e perdas causados pela abstração. Os resultados estão ilustrados na Fig. 3, onde se observa que o Qab-Learning mostrou melhor desempenho nesse caso. Ainda no mesmo ambiente, outro experimento avaliou a execução das políticas na tarefa de, saindo de qualquer local no ambiente 1, atingir um dos centros de cômodo (locais *l2*, *l3*, *l9*, *l17* ou *l18*), apresentando a média alcançada pela execução de todas estas tarefas (cada uma consistindo em 10.000 episódios com no máximo 1000 passos cada). O resultado deste segundo experimento está ilustrado na Fig. 4, onde se verifica que o Qab-Learning e ND-TILDE apresentam resultados similares e melhores do que a política aleatória.

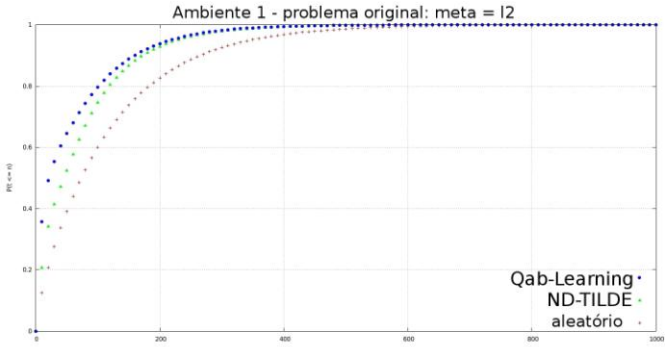


Figura 3 – Resultados da aplicação das políticas abstratas e aleatória no problema original: no ambiente 1, sair de qualquer lugar (exceto o local *l2)* e chegar a *l2*.

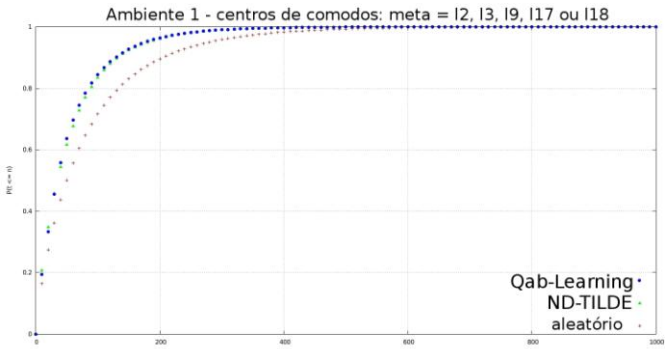


Figura 4 – Resultados da aplicação das políticas abstratas e aleatória no ambiente 1 para a tarefa de, alcançar cada um dos centros de cômodos do ambiente 1 a partir das demais localizações que não a que se pretende alcançar. O gráfico apresenta a média dos resultados obtidos tendo como metas em *l2*, *l3*, *l9*, *l17* e *l18*, em que cada experimento relacionado a cada uma das metas consistiu em 1000 episódios.

Outros três experimentos foram realizados no mapa da Fig. 2 com o objetivo de avaliar o desempenho das políticas abstratas definidas para o ambiente 1. Nesses experimentos,

buscou-se avaliar a capacidade de transferência do conhecimento adquirido em um problema para um novo problema. O experimento cujo resultado está ilustrado na Fig. 5 mostra o desempenho das políticas abstratas e aleatória no ambiente 2, quando a tarefa é atingir *l2* partindo de cada uma das demais localizações. A Fig. 6 mostra o desempenho das políticas em uma tarefa similar, na qual a meta é *l25* a partir das demais localizações do ambiente 2. Já a Fig. 7 mostra o desempenho apresentado nas tarefas de, saindo de qualquer localização a menos da meta, atingir cada um dos centros das salas do ambiente 2.

Analisando as figuras, pode-se observar que, tanto ND-TILDE quanto Qab-Learning geram políticas que apresentam um melhor desempenho que o resultado pelo uso da política aleatória.

O desempenho superior da política π_{Qab} em relação à π_{ND} para a solução do problema que foi utilizado na obtenção de ambas (Fig. 3), sugere que π_{Qab} foi capaz de reter mais elementos necessários à solução do problema original do que $\pi_{ND,}$ isto é, a política π_{Qab} é mais especializada que π_{ND}. Entretanto, quando a tarefa é mais geral (Fig. 4), ou seja, atingir uma localidade que, apesar de possuir a mesma natureza (centro de cômodo), possui uma localização diferente e, consequentemente, uma caracterização topológica dada por sua vizinhança também diferente (Fig. 1), ambas as políticas apresentam um desempenho similar, superando a aleatória.

Esse resultado, aliado ao que foi discutido anteriormente, poderia apoiar uma interpretação positiva em relação à π_{Qab}, sugerindo que essa política, além de reter um relativo bom desempenho na solução do problema que foi treinada para resolver, possui também características que poderiam ser explicadas por uma capacidade de generalização, também apresentada pela π_{ND}. Isso lhes permitiria atingir um bom desempenho em uma tarefa distinta da presente em seus treinos, mas que pertence ao mesmo domínio.

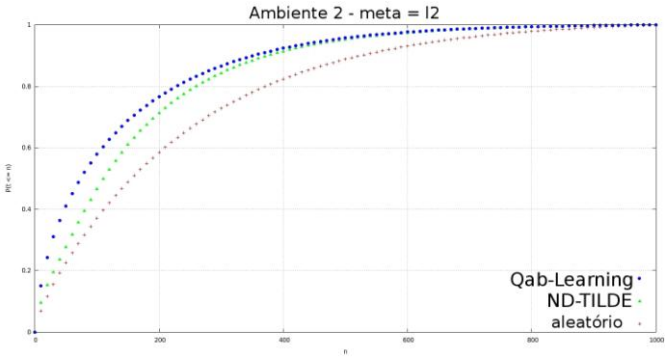


Figura 5 – Resultados da aplicação das políticas abstratas e aleatória no ambiente 2, na tarefa de, saindo de qualquer uma das demais localidades, chegar a *l2*.

Os resultados que compõem o cerne da proposta desse trabalho são ilustrados nas figuras 5, 6 e 7 e se referem à aplicação das políticas aprendidas no ambiente 1 na execução de tarefas no ambiente 2. No experimento cujos resultados são ilustrados na Fig. 5, o agente parte de cada uma das localizações e deve atingir a meta *l2* executando, no máximo, 1000 passos. É interessante notar que a localidade *l2* se encontra em uma região similar ao ambiente 1, correspondendo à meta "antiga". Pode-se ver que o desempenho das políticas é superior ao da aleatória, sendo que π_{Qab}, novamente, apresenta desempenho superior ao de π_{ND}. Esse fato corrobora a hipótese discutida acima de retenção de elementos do problema utilizado na obtenção da política, isto é, de maior especialização de π_{Qab} em relação à $\pi_{ND.}$

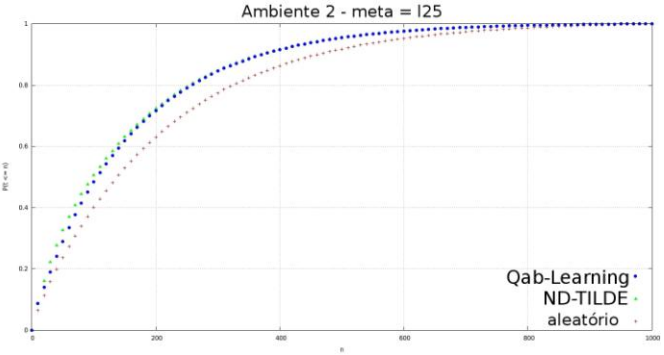


Figura 6 – Resultados da aplicação das políticas abstratas e aleatória no ambiente 2, na tarefa de sair de qualquer lugar e chegar ao local *l25*.

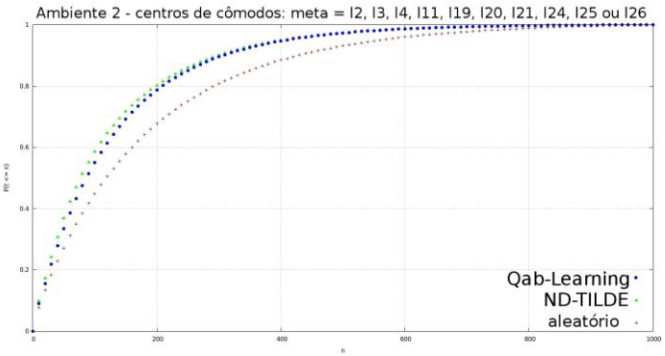


Figura 7 – Resultados da aplicação das políticas abstratas e aleatória no ambiente 2, na tarefa de sair de qualquer lugar e chegar a cada um dos 10 centros dos cômodos. O resultado apresentado é a média dos desempenhos apresentados em cada uma das 10 tarefas.

No experimento cujos resultados são representados na Fig. 6, o agente deveria atingir *l25*, que além de estar fora da região *conhecida* correspondente ao problema original, localiza-se em uma região oposta à *l2*, utilizada como meta na obtenção das políticas. Esperava-se com esse experimento verificar a capacidade de aplicação das políticas quando o problema de navegação consiste em atingir uma localidade que, apesar de possuir a mesma natureza (centro de cômodo), e com isso assemelhar-se à solução do problema original, possuísse uma caracterização topológica que dificultasse a aplicação direta da política na solução do problema (atingir uma meta em região oposta e com vizinhança ainda desconhecida). Nesse experimento, as políticas apresentaram um desempenho similar; entretanto, a π_{ND} foi ligeiramente melhor. Uma possível explicação seria a de que a suposta retenção de elementos do problema original por π_{Qab} seria acompanhada de especialização. Apesar de não ser significativa, pode-se utilizar o fato para analisar as características intrínsecas de obtenção das políticas, além da representação utilizada em cada uma, para que se possa inferir

quais elementos interferem (e com que magnitude) na generalização de uma política concreta obtida na solução de um problema prévio. Uma possível solução para o problema seria desenvolver e aplicar representações que permitissem a generalização da política abstrata para novos problemas, entretanto sem perder a característica desejável de alto desempenho na solução de um problema já resolvido.

Essa análise é respaldada pelos resultados apresentados na Fig. 7, em que o agente deve atingir os centros de cômodos presentes no ambiente 2 a partir de cada uma das demais localidades. A política gerada pelo ND-TILDE apresenta um desempenho ligeiramente superior, e ambas superam a política abstrata na solução do problema.

VII. Conclusões.

Neste artigo, foi proposto o algoritmo Qab-Learning como uma alternativa para a transferência do conhecimento obtido na solução prévia de um problema de navegação e aplicação desse conhecimento na solução de um novo problema de navegação através da geração de uma política abstrata.

A política abstrata gerada pelo Qab-Learning, através de uma modificação do algoritmo Q-learning em que estados e ações abstratos são utilizados no aprendizado, é comparada à política gerada pelo algoritmo ND-TILDE. Ambas as políticas geradas por estes algoritmos possuem um desempenho superior ao de uma política aleatória, i.e., uma escolha aleatória de ações para guiar o robô pelo ambiente. Isso pode ser visto como transferência do aprendizado entre problemas através da utilização das políticas abstratas geradas pelos algoritmos Qab-Learning e ND-TILDE.

Nos experimentos realizados foi possível verificar que o algoritmo proposto, Qab-Learning, criou políticas abstratas que obtiveram desempenho superior ao da política criada com ND-TILDE em determinadas situações, mas que ao mesmo tempo apresentavam desempenho similar ou inferior em outras. Em vista disto, futuramente pretende-se analisar os elementos presentes na obtenção de ambas, buscando verificar quais deles possuem maior e menor impacto na geração de políticas abstratas.

Entretanto, nas situações de superioridade do Qab-Learning, a diferença entre as políticas é notável, principalmente pelo fato do ND-TILDE agregar estados no nível abstrato, no intuito de tornar a política mais geral em detrimento da velocidade, e o Qab-Learning não (utiliza todos os estados abstratos existentes). Certamente, o intuito da construção dessas políticas abstratas não é utilizá-las como únicos guias do agente, mas sim para acelerar o aprendizado de novas tarefas através da transferência do conhecimento prévio. Desse modo, um dos aspectos a se trabalhar no futuro seria a identificação de subtarefas ou partições do espaço de estados nas quais a política abstrata é apropriada para um novo problema. Ou seja, definir quais partes da política apresentam conhecimento relevante e quando aplicá-las.

Além disso, outro aspecto a ser estudado é a definição e avaliação dos predicados que compõem a descrição do domínio. Os predicados possuem papel decisivo na construção da política abstrata e dependendo da forma como são definidos, podem levar à criação de políticas melhores ou piores.

Referências

[1] Dudek, G.; Jenkin, M. *Computational Principles of Mobile Robotics*. New York, Cambridge University Press, 2000.

[2] Choset, H. M; Lynch, K. M.; Hutchinson, S.; Kantor, G.; Burgard, W.; Kavraki, L. E.; Thrun, *S. Principles of robot motion: theory, algorithms, and implementation*. New York, MIT Press, 2005.

[3] Thrun, S.; Burgard, W.; Fox, D. *Probabilistic Robotics*. New York, MIT Press, 2005.

[4] Russell, S. J.; Norvig, P. *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ, Pearson Education, Inc., 2nd. edition, 2003.

[5] Torrey, L.; Shavlik, J.; Waker, T.; Maclin,R. *Relational macros for transfer in reinforcement learning*. 2008. In: ILP'07 – *Proc. Of the 17th Int. Conf. on Inductive Logic Programming*, 2008.

[6] Madden, M. G.; Howley, T. Transfer of experience between reinforcement learning environments with progressive difficulty. *Artificial Intelligence Review*, Kluwer Academic Publishers, Norwell, MA, USA, v. 21, p. 375-398, June 2004.

[7] Matos, T.; Bergamo, Y.; Silva, V. F. da; Costa, A. H. R. Stochastic Abstract Policies for Knowledge Transfer in Robotic Navigation Tasks. In: MICAI'2011 – *Mexican International Conference on Artificial Intelligence*, 2011.

[8] Blockeel, H.; Raedt, L.D. Top-down induction of first-order logical decision trees. *Artificial Intelligence*, Volume 101, Issues 1-2, p. 285-297. May 1998.

[9] Sutton, R.S., Barto, A.G.: *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA. 1998.

[10] Kersting,K.; Otterlo,M.V.; and Raedt,L.D. 2004. Bellman goes relational. In *Proc. of 21th Int. Conf. on Machine Learning*, 465–472.

[11] van Otterlo, M. *The Logic of Adaptative Behaviour*, IOS Press, Amsterdam, 2009.

[12] Hoey, J., St-Aubin, R., Hu, A.J., Boutilier, C.: Spudd: Stochastic planning using decision diagrams. In: UAI'99 – *Proc. of Uncertainty in Artificial Intelligence*, Stockholm, Sweden. 1999.

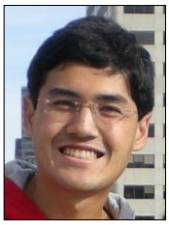
Agradecimentos

Esta pesquisa obteve o apoio da FAPESP (08/03995-5, 09/04489-9, 09/14650-1) e do CNPq (305512/2008-0).

Rafael agradece a Guilherme Renoldi pelo auxílio nas etapas de depuração do simulador e pelas ideias surgidas durante as construtivas discussões.

Rafael Lemes Beirigo é aluno do Curso de Ciências Moleculares, da Pró-Reitoria de Graduação da Universidade de São Paulo (CM-USP) e desenvolve sua pesquisa como aluno de Iniciação Científica no Laboratório de Técnicas Inteligentes (LTI), da Escola Politécnica (POLI-USP). Seus interesses em pesquisa em Inteligência Artificial recaem particularmente sobre o Aprendizado de Máquinas, especificamente Mapas Semânticos e Políticas Abstratas, considerando os processos de abstração de instâncias de problemas de navegação e os elementos envolvidos.

Fernando de Andrade Pereira é aluno do curso de Engenharia Elétrica com ênfase em Computação na Escola Politécnica da Universidade de São Paulo (EPUSP). Seus interesses na área de Inteligência Artificial voltam-se para os ramos do Aprendizado por Reforço e Políticas Abstratas.



Marcelo Li Koga graduou-se em Engenharia Elétrica com ênfase em Computação (2010) e atualmente é estudante de Mestrado em Inteligência Artificial, ambos pela Escola Politécnica da Universidade de São Paulo (EPUSP). Em 2008 atuou como representante discente do PCS (Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP) e em 2011 apresentou trabalho sobre inferência de gramáticas usando adaptatividade no WTA (Workshop de Tecnologias Adaptativas). Atualmente seus interesses de pesquisa se concentram nas áreas de Aprendizado por Reforço e Planejamento Probabilístico.

Tiago Matos possui mestrado em Engenharia Elétrica pela Universidade de São Paulo (2011) e é formado em Engenharia Elétrica - Ênfase Computação pela Escola Politécnica da Universidade de São Paulo (2008). Seus interesses em pesquisa se concentram nas áreas de Aprendizado por Reforço e Reaproveitamento do Conhecimento.

Valdinei Freire da Silva é Professor Doutor do curso de Sistemas de Informação da Escola de Artes, Ciências e Humanidades da Universidade de São Paulo (USP, 2011). Possui Doutorado em Engenharia Elétrica (USP em co-tutela com IST-UTL, 2009) e é formado em Engenharia da Computação (USP, 2002). Seus interesses em pesquisa abrangem vários aspectos de Aprendizado de Máquina e Tomadas de Decisões Sequenciais em ambientes probabilísticos.

Anna Helena Reali Costa é Professora Titular do Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo (USP, 2011). Possui Doutorado em Engenharia Elétrica (USP, 1994) e Mestrado em Engenharia (FEI, 1989). É formada em Engenharia Elétrica (FEI, 1983). De 1985 a 1988 e de 1991 a 1992 foi pesquisadora na Universidade de Karlsruhe, Alemanha. Em 1998 e 1999 foi pesquisadora visitante na Carnegie Mellon University, nos Estados Unidos. Seus interesses em pesquisa abrangem um grande espectro da Inteligência Artificial, incluindo particularmente o Aprendizado de Máquinas, Robótica Inteligente e Visão Computacional.