UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

CLEITON ALVES DA SILVA

**Seleção de abstração espacial no Aprendizado por Reforço avaliando o
processo de aprendizagem**

São Paulo

2018

CLEITON ALVES DA SILVA

**Seleção de abstração espacial no Aprendizado por Reforço avaliando o
processo de aprendizagem**

Dissertação apresentada à Escola de Artes,
Ciências e Humanidades da Universidade de
São Paulo para obtenção do título de Mestre
em Ciências pelo Programa de Pós-graduação
em Sistemas de Informação.

Área de concentração: Metodologia e
Técnicas da Computação

Versão corrigida contendo as alterações
solicitadas pela comissão julgadora em 14 de
junho de 2017. A versão original encontra-se
em acervo reservado na Biblioteca da
EACH-USP e na Biblioteca Digital de Teses
e Dissertações da USP (BDTD), de acordo
com a Resolução CoPGr 6018, de 13 de
outubro de 2011.

Orientador: Prof. Dr. Valdinei Freire da Silva

São Paulo

2018

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

CATALOGAÇÃO-NA-PUBLICAÇÃO

(Universidade de São Paulo. Escola de Artes, Ciências e Humanidades. Biblioteca)

Silva, Cleiton Alves da

Seleção de abstração espacial no Aprendizado por Reforço avaliando o processo de aprendizagem / Cleiton Alves da Silva ; orientador, Valdinei Freire da Silva. – São Paulo, 2018.

78 p. : il

Dissertação (Mestrado em Ciências) - Programa de Pós-Graduação em Sistemas de Informação, Escola de Artes, Ciências e Humanidades, Universidade de São Paulo.

Versão corrigida

1. Inteligência artificial. 2. Aprendizagem - Processos. I. Silva, Valdinei Freire da, orient. II. Título

CDD 22.ed.– 006.3

Dissertação de autoria de Cleiton Alves da Silva, sob o título “**Seleção de abstração espacial no Aprendizado por Reforço avaliando o processo de aprendizagem**”, apresentada à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo, para obtenção do título de Mestre em Ciências pelo Programa de Pós-graduação em Sistemas de Informação, na área de concentração Metodologia e Técnicas da Computação, aprovada em 14 de junho de 2017 pela comissão julgadora constituída pelos doutores:

Prof. Dr. Valdinei Freire da Silva

Instituição: EACH-USP

Presidente

Profa. Dra. Karina Valdivia Delgado

Instituição: EACH-USP

Profa. Dra. Anna Helena Reali Costa

Instituição: Poli-USP

Prof. Dr. Reinaldo Augusto da Costa Bianchi

Instituição: Centro Universitário FEI

Dedico à Clara e ao Lucas, meus filhos, e à Valéria Espanha, minha esposa e amiga.

Agradecimentos

À minha esposa, o maior de todos os agradecimentos.

À minha família, pelo apoio e incentivo.

Eterna gratidão ao professor Dr. Valdinei Freire da Silva, que desde o primeiro contato, incentivou, ensinou e não desistiu, mantendo-se confiante na elaboração deste trabalho e no amadurecimento acadêmico deste mestrando.

para $i = 1$ até 10^6

escreva (“Obrigado!”)

fim para

Agradeço a todos os professores do programa pelo conhecimento compartilhado. Oportunidades proporcionadas para o descobrimento de assuntos que jamais imaginei conhecer. Um agradecimento em especial à professora Dra. Karina Valdivia Delgado que, desde cedo, acompanhou o desenvolvimento deste aprendiz.

Sou grato pelas amizades conquistadas, Pedro Yuri, Ricardo Feitosa, Alexandre Lima e Bruno César Lima, obrigado pelos ensinamentos, incentivos e momentos felizes.

Deus, sem Você dentro de mim jamais seria quem eu sou.

*“Qualquer coisa que você possa fazer ou sonhar, você pode começar. A ousadia tem
genialidade, poder e magia em si.”
(Johann Goethe)*

Resumo

SILVA, Cleiton Alves. **Seleção de abstração espacial no Aprendizado por Reforço avaliando o processo de aprendizagem**. 2018. 78 f. Dissertação (Mestrado em Ciências) – Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, São Paulo, 2017.

Agentes que utilizam técnicas de Aprendizado por Reforço (AR) buscam resolver problemas que envolvem decisões sequenciais em ambientes estocásticos sem conhecimento *a priori*. O processo de aprendizado desenvolvido pelo agente em geral é lento, visto que se concretiza por tentativa e erro e exige repetidas interações com cada estado do ambiente e como o estado do ambiente é representado por vários fatores, a quantidade de estados cresce exponencialmente de acordo com o número de variáveis de estado. Uma das técnicas para acelerar o processo de aprendizado é a generalização de conhecimento, que visa melhorar o processo de aprendizado, seja no mesmo problema por meio da abstração, ao explorar a similaridade entre estados semelhantes ou em diferentes problemas, ao transferir o conhecimento adquirido de um problema fonte para acelerar a aprendizagem em um problema alvo. Uma abstração considera partes do estado e, ainda que uma única não seja suficiente, é necessário descobrir qual combinação de abstrações pode atingir bons resultados. Nesta dissertação é proposto um método para seleção de abstração, considerando o processo de avaliação da aprendizagem durante o aprendizado. A contribuição é formalizada pela apresentação do algoritmo **REPO**, utilizado para selecionar e avaliar subconjuntos de abstrações. O algoritmo é iterativo e a cada rodada avalia novos subconjuntos de abstrações, conferindo uma pontuação para cada uma das abstrações existentes no subconjunto e por fim, retorna o subconjunto com as abstrações melhores pontuadas. Experimentos com o simulador de futebol mostram que esse método é efetivo e consegue encontrar um subconjunto com uma quantidade menor de abstrações que represente o problema original, proporcionando melhoria em relação ao desempenho do agente em seu aprendizado.

Palavras-chaves: Aprendizado por Reforço. Seleção de Abstração. Transferência do Conhecimento.

Abstract

SILVA, Cleiton Alves. **Selection of spatial abstraction in Reinforcement Learning by learning process evaluating**. 2018. 78 p. Dissertation (Master of Science) – School of Arts, Sciences and Humanities, University of São Paulo, São Paulo, 2017.

Agents that use Reinforcement Learning (RL) techniques seek to solve problems that involve sequential decisions in stochastic environments without a priori knowledge. The learning process developed by the agent in general is slow, since it is done by trial and error and requires repeated iterations with each state of the environment and because the state of the environment is represented by several factors, the number of states grows exponentially according to the number of state variables. One of the techniques to accelerate the learning process is the generalization of knowledge, which aims to improve the learning process, be the same problem through abstraction, explore the similarity between similar states or different problems, transferring the knowledge acquired from A source problem to accelerate learning in a target problem. An abstraction considers parts of the state, and although a single one is not sufficient, it is necessary to find out which combination of abstractions can achieve good results. In this work, a method for abstraction selection is proposed, considering the evaluation process of learning during learning. The contribution is formalized by the presentation of the REPO algorithm, used to select and evaluate subsets of features. The algorithm is iterative and each round evaluates new subsets of features, giving a score for each of the features in the subset, and finally, returns the subset with the most highly punctuated features. Experiments with the soccer simulator show that this method is effective and can find a subset with a smaller number of features that represents the original problem, providing improvement in relation to the performance of the agent in its learning.

Keywords: Reinforcement Learning. Abstraction Selection. Transfer Learning.

Lista de figuras

Figura 1 – Interação entre agente e ambiente no Aprendizado por Reforço.	28
Figura 2 – <i>Tile Coding</i> em um espaço bidimensional. Vista superior à esquerda e lateral à direita. Nota-se que o <i>tiling</i> 2 está ligeiramente deslocada no espaço 2D em relação ao <i>tiling</i> 1.	34
Figura 3 – Simulador de Futebol D2DSS.	39
Figura 4 – Curvas de aprendizagem. Resultado obtido com um agente Sarsa, executando ações primitivas e utilizando 5 diferentes subconjuntos de abstrações.	43
Figura 5 – Curvas de aprendizagem. Resultado obtido com um agente Sarsa, executando ações comportamentais e utilizando 5 diferentes subconjuntos de abstrações.	44
Figura 6 – Comparação entre as melhores abstrações e o melhor subconjunto. Figura a esquerda exibe a média de recompensas acumuladas em cada rodada, utilizando $N_E = 16$. Figura a esquerda, exibe resultado do t-teste.	54
Figura 7 – Comparação entre as melhores abstrações e o melhor subconjunto. Figura a esquerda exibe a média de recompensas acumuladas em cada rodada, utilizando $N_E = 32$. Figura a esquerda, exibe resultado do t-teste.	55
Figura 8 – Comparação entre três diferentes abordagens de distribuição. Na Figura a esquerda: $N_E = 16$, exibe-se as curvas de aprendizagem com a médias de recompensa acumulada em cada rodada, as linhas tracejadas mostram a avaliação dos subconjuntos avaliados em cada rodada, enquanto linhas sólidas exibem a avaliação das melhores abstrações em cada rodada. A Figura a direita, exibe o resultado do teste- t parelhado das amostras em cada rodada.	56

Figura 9 – Comparação entre três diferentes abordagens de distribuição. Na Figura a esquerda: $N_E = 32$, exibe-se as curvas de aprendizagem com a médias de recompensa acumulada em cada rodada, as linhas tracejadas mostram a avaliação dos subconjuntos avaliados em cada rodada, enquanto linhas sólidas exibem a avaliação das melhores abstrações em cada rodada. A Figura a direita, exibe o resultado do teste- t parelhado das amostras em cada rodada.	56
Figura 10 – Comparação entre o algoritmo REPO e construção incremental. Figura esquerda exibe a recompensa média acumulada para diferentes tamanhos de subconjuntos. Figura direita exibe um teste- t de amostra pareada para cada tamanho do subconjunto.	58
Figura 11 – Comparação entre o algoritmo REPO e abordagem <i>Log Hard Selection</i> . A Figura exibe a recompensa média acumulada para diferentes tamanhos de subconjuntos e versões do algoritmo REPO.	59
Figura 12 – Avaliação de subconjuntos de abstrações com tamanhos diferentes e transferência de abstrações ranqueadas para problemas destino. Os experimentos variam o tamanho do campo do simulador, o conjunto de ações do agente de AR e o parâmetro ε do Agente de Planejamento. . .	60
Figura 13 – Comparação entre três diferentes abordagens de distribuição com $N_E = 32$ e parâmetro $\varepsilon = 1.0$. A Figura exibe as curvas de aprendizagem com as médias de recompensa acumulada em cada rodada, as linhas tracejadas mostram a avaliação dos subconjuntos avaliados em cada rodada, enquanto linhas sólidas exibem a avaliação das melhores abstrações em cada rodada.	62
Figura 14 – Comparação entre três diferentes abordagens de distribuição com $N_E = 32$ e parâmetro $\varepsilon = 0.5$. A Figura exibe as curvas de aprendizagem com as médias de recompensa acumulada em cada rodada, as linhas tracejadas mostram a avaliação dos subconjuntos avaliados em cada rodada, enquanto linhas sólidas exibem a avaliação das melhores abstrações em cada rodada.	63

- Figura 15 – Comparação entre três diferentes abordagens de distribuição com $N_E = 32$ e parâmetro $\varepsilon = 0$. A Figura exibe as curvas de aprendizagem com as médias de recompensa acumulada em cada rodada, as linhas tracejadas mostram a avaliação dos subconjuntos avaliados em cada rodada, enquanto linhas sólidas exibem a avaliação das melhores abstrações em cada rodada. 64
- Figura 16 – Avaliação de subconjuntos de abstrações transferidos do problema origem, comparando as três diferentes abordagens de distribuição com $N_E = 32$ e parâmetro $\varepsilon = 1$ e campo do simulador em 33×21 . A Figura exibe as curvas de aprendizagem com as médias de recompensa acumulada em cada rodada, as linhas exibem a avaliação das melhores abstrações em cada rodada. 65
- Figura 17 – Avaliação de subconjuntos de abstrações transferidos do problema origem, comparando as três diferentes abordagens de distribuição com $N_E = 32$ e parâmetro $\varepsilon = 0.5$ e campo do simulador em 33×21 . A Figura exibe as curvas de aprendizagem com as médias de recompensa acumulada em cada rodada, as linhas exibem a avaliação das melhores abstrações em cada rodada. 66
- Figura 18 – Avaliação de subconjuntos de abstrações transferidos do problema origem, comparando as três diferentes abordagens de distribuição com $N_E = 32$ e parâmetro $\varepsilon = 0$ e campo do simulador em 33×21 . A Figura exibe as curvas de aprendizagem com as médias de recompensa acumulada em cada rodada, as linhas exibem a avaliação das melhores abstrações em cada rodada. 67
- Figura 19 – Avaliação de subconjuntos de abstrações transferidos do problema origem, comparando as três diferentes abordagens de distribuição com $N_E = 32$ e parâmetro $\varepsilon = 1$ e campo do simulador em 23×15 . A Figura exibe as curvas de aprendizagem com as médias de recompensa acumulada em cada rodada, as linhas exibem a avaliação das melhores abstrações em cada rodada. 68

Figura 20 – Avaliação de subconjuntos de abstrações transferidos do problema origem, comparando as três diferentes abordagens de distribuição com $N_E = 32$ e parâmetro $\varepsilon = 0.5$ e campo do simulador em 23×15 . A Figura exhibe as curvas de aprendizagem com as médias de recompensa acumulada em cada rodada, as linhas exibem a avaliação das melhores abstrações em cada rodada.	69
Figura 21 – Avaliação de subconjuntos de abstrações transferidos do problema origem, comparando as três diferentes abordagens de distribuição com $N_E = 32$ e parâmetro $\varepsilon = 0$ e campo do simulador em 23×15 . A Figura exhibe as curvas de aprendizagem com as médias de recompensa acumulada em cada rodada, as linhas exibem a avaliação das melhores abstrações em cada rodada.	70
Figura 22 – Transferência de abstrações entre ambientes diferentes.	71

Lista de algoritmos

Algoritmo 1 – Algoritmo Q-learning	30
Algoritmo 2 – Algoritmo Sarsa	31
Algoritmo 3 – Algoritmo Sarsa com aproximador de função	32
Algoritmo 4 – Algoritmo REPO.	48
Algoritmo 5 – Algoritmo UpdateEval.	49
Algoritmo 6 – Algoritmo para seleção de abstração baseado em pontuação e busca.	50
Algoritmo 7 – Algoritmo para seleção de abstrações baseado em <i>Backward selection</i>	51

Lista de abreviaturas e siglas

AR	Aprendizado por Reforço
CMAC	<i>Cerebellar Model Articulator Controller</i>
IV	Iteração de Valor
PMD	Processo Markoviano de Decisão
TC	Transferência do Conhecimento
REPO	<i>Ranking Evolved Population Optimization</i>
D2DSS	<i>Discrete 2D Soccer Simulator</i>

Lista de símbolos

α	Taxa de aprendizagem
ϵ	Probabilidade associada a escolha de uma ação aleatória na estratégia $\epsilon - greedy$
π	Política
π^*	Política ótima, ou seja, política que produz o valor máximo em cada estado
γ	Fator de desconto associado a recompensas futuras
θ	Vetor de parâmetros
$\mathcal{F}$	Conjunto de abstrações de estados
$\mathcal{F}'$	Subconjunto ótimo de abstrações
a	Ação
A	Conjunto de ações
F	Vetor de abstrações
Q	Função valor-ação
Q^*	Função valor-ação ótima
r	Recompensa
R	Função recompensa
s	Estado
S	Conjunto de estados
T	Função de transição
V	Função valor
V^π	Função valor associada à política π

V^*	Função valor-estado ótima
τ	Parâmetro de temperatura na distribuição de Boltzmann
$\mathbf{e}$	Vetor com avaliação abstrações

Sumário

1	Introdução	19
1.1	<i>Objetivo geral</i>	21
1.2	<i>Organização do Texto</i>	22
2	Revisão da Literatura	23
3	Fundamentação teórica	27
3.1	<i>Aprendizado por Reforço</i>	27
3.2	<i>Algoritmos no AR</i>	29
3.3	<i>Accelerando o aprendizado</i>	31
3.4	<i>Abstração Espacial: Tile Coding</i>	33
3.5	<i>Transferência de Conhecimento</i>	34
3.6	<i>Seleção de atributos</i>	36
4	Problema estudado e sua representação	38
4.1	<i>Simulador de futebol</i>	38
4.2	<i>Agente comportamental e atômico</i>	40
4.3	<i>Projetando abstrações espaciais</i>	41
4.4	<i>Estudo de caso</i>	43
5	Proposta	45
5.1	<i>Seleção de abstração por meio da avaliação do processo de aprendizagem</i>	45
5.2	<i>Algoritmo REPO</i>	47
5.3	<i>Tamanho dos subconjuntos de abstrações</i>	50
6	Experimentos	52
6.1	<i>Resultados - Avaliação do algoritmo REPO</i>	52
6.2	<i>Configurações do simulador de futebol</i>	52
6.2.1	<i>Comparação entre melhores abstrações e melhor subconjunto</i>	53
6.2.2	<i>Abordagens de distribuição de probabilidade</i>	54
6.2.3	<i>Dependência do tamanho fixo dos subconjuntos</i>	56
6.2.4	<i>Transferência de conhecimento</i>	59

6.3	<i>Novos experimentos com uso do agente atômico</i>	60
6.3.1	Abordagens de distribuição de probabilidade	61
6.3.2	Transferência de conhecimento	64
6.3.3	Transferência de conhecimento: agente comportamental e atômico .	70
6.3.4	Análise qualitativa das abstrações	71
7	Conclusão	73
7.0.1	Trabalhos futuros	73
7.0.2	Trabalhos publicados	74
	Referências ¹	75

¹ De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.

1 Introdução

Problemas em que um agente está inserido em um ambiente estocástico e toma decisões sequenciais podem ser resolvidos por técnicas de Aprendizado por Reforço (AR) (SUTTON; BARTO, 1998), que consideram o processo de aprendizado por meio de tentativa e erro e de repetidas interações do agente com o ambiente. Nesse contexto, o agente deve utilizar uma estratégia que realize um compromisso entre escolher ações que possam resultar em mais conhecimento sobre ambiente e usar o conhecimento adquirido para obter recompensas, para aprender uma política de ações adequada. Os algoritmos tradicionais utilizados no AR, como Q-learning (WATKINS; DAYAN, 1992) e Sarsa (RUMMERY; NIRANJAN, 1994) estimam um valor de cada estado visitado com base na experiência adquirida e, em suas versões tabulares, o aprendizado consiste em aprender uma função valor $Q : S \times A \rightarrow \mathbb{R}$, envolvendo $|S \times A|$ parâmetros. As condições para convergência implicam em visitar infinitas vezes cada um dos estados, embora uma política adequada possa ser obtida em um tempo finito. No entanto, o tamanho do espaço de estado aumenta exponencialmente com o número de variáveis de estado e dependendo da quantidade de variáveis de estado, o aprendizado pode se tornar inviável.

Para lidar com situações em que o espaço de estado é arbitrariamente grande e acelerar o aprendizado, o agente deve ser capaz de generalizar o conhecimento adquirido em outras experiências e compartilhá-lo entre estados semelhantes (SUTTON, 1996). Uma abordagem comum para se obter generalização é aproximar a função valor Q por uma função q parametrizada por $\theta \in \mathbb{R}^{N_c}$, onde $n \ll |S \times A|$ diminuindo a quantidade de parâmetros a serem aprendidos. Uma forma de criar uma função q é utilizar abstração de estados, que agrega estados com características semelhantes e por consequência diminui a quantidade de estados a serem considerados, permitindo que a função q possa ser representada por uma tabela. Embora uma abstração com poucos estados abstratos possa acelerar o aprendizado, essa abstração também pode limitar a qualidade da política que será aprendida. Uma forma de lidar com esse problema é considerar um conjunto $\mathcal{F}$ de abstrações espaciais e aproximar a função valor Q combinando as abstrações $F \in \mathcal{F}$, de modo a garantir que o aprendizado se torne viável e que a qualidade da política aprendida não seja deteriorada. Uma pergunta de pesquisa interessante é: como selecionar um subconjunto ótimo de abstrações espaciais $\mathcal{F}' \subset \mathcal{F}$?

Outra forma de acelerar o aprendizado no AR é considerar um ou mais ambientes origem, em que o agente pode experimentar e adquirir conhecimento e, posteriormente, transferir esse conhecimento acumulado para um ambiente alvo geralmente mais complexo (TAYLOR; STONE, 2009; TORREY; SHAVLIK, 2009). Nesse caso, em vários cenários, desconsidera-se o tempo gasto para obter conhecimento no ambiente origem e busca-se acelerar o aprendizado no ambiente alvo. O conhecimento adquirido e transferido difere de abordagem para abordagem; enquanto vários trabalhos consideram a transferência da função valor Q (TAYLOR; STONE, 2005) ou política (FERNÁNDEZ; VELOSO, 2006; TAYLOR; WHITESON; STONE, 2006; TORREY et al., 2006), que aconselham como o agente deve agir no ambiente alvo, poucos são os trabalhos que consideram a transferência de abstrações espaciais (WALSH; LI; LITTMAN, 2006).

Para selecionar um subconjunto ótimo de abstrações espaciais $\mathcal{F}^*$ é essencial definir como cada subconjunto $\mathcal{F}' \subset \mathcal{F}$ é avaliado. No AR, usualmente, avalia-se um subconjunto pelo seu poder de expressão, isto é, quão bem uma função q obtida do subconjunto $\mathcal{F}'$ aproxima a função valor Q e alguma penalidade baseada no tamanho de $\mathcal{F}'$ e na quantidade de parâmetros utilizados pelo subconjunto de abstrações, garantindo generalização de conhecimento. Por outro lado, no arcabouço de transferência de conhecimento, busca-se a qualidade do processo de aprendizagem no ambiente alvo, o poder de expressão e o tamanho do subconjunto $\mathcal{F}'$ funcionam como indicadores heurísticos para tal qualidade. Dessa forma, uma avaliação direta da influência de um subconjunto de abstrações $\mathcal{F}'$ no processo de aprendizagem pode resultar na escolha de um subconjunto de abstrações melhor. Por outro lado, se para avaliar um subconjunto de abstrações deve-se incorrer no processo de aprendizagem, a avaliação obtida só será útil quando da necessidade de um novo aprendizado, isto é, no ambiente alvo; além disso, vários subconjuntos de abstrações devem ser avaliados, sendo necessário um uso ilimitado de um ambiente origem ou uma alta eficiência na extração de informação da avaliação de qualquer subconjunto de abstrações.

Cada uma das abstrações no conjunto $\mathcal{F}$ pode ser pensada como um atributo, de modo que o problema de seleção de abstrações pode ser pensado como um problema de seleção de atributos, o qual dispõe de uma extensa literatura. Pode-se caracterizar os métodos de seleção de atributos em três principais abordagens: *Filter*, *Wrapper* e *Embedded* (TANG; ALELYANI; LIU, 2013). A abordagem *Wrapper* explora o espaço de atributos e gera vários subconjuntos candidatos e, após avaliar cada subconjunto, seleciona aquele com melhor desempenho. A abordagem *Filter* estima um índice de relevância para cada

atributo segundo algum critério estatístico e, em seguida, classifica o atributo de acordo com tal relevância, geralmente independente do processo de aprendizagem. Na abordagem *Embedded* um algoritmo de aprendizado executa a seleção de atributos durante o processo de aprendizagem, ajustando o modelo e a seleção de atributos simultaneamente, esta abordagem é comumente utilizada no Aprendizado por Reforço. Para considerar o processo de aprendizagem como avaliação a abordagem *Wrapper* é mais adequada, a abordagem *Filter* permite extrair informações de forma mais eficiente das avaliações realizadas.

1.1 Objetivo geral

Considerando as hipóteses: (i) avaliar o efeito de um subconjunto de abstrações no processo de aprendizagem é melhor do que avaliar o poder de representação desse subconjunto; e (ii) o conhecimento obtido pode ser transferido para um ambiente alvo; o objetivo deste trabalho é propor um método para seleção de um subconjunto de abstrações que considere a avaliação do processo de aprendizagem no AR.

E, de maneira mais restrita, geraram os **objetivos específicos**:

- Especificar uma regra de pontuação para abstrações com base na avaliação de subconjuntos de abstrações;
- Especificar uma regra para escolha de subconjuntos a serem avaliados;
- Definir um conjunto de abstrações espaciais $\mathcal{F}$ para o ambiente de simulador de futebol de robô;
- Implementar uma solução da literatura (*Forward e Backward Selection*) e
- Comparar os resultados da solução da literatura com os dos métodos propostos.
- Realizar uma análise empírica dos métodos propostos.

Ao considerar como método de avaliação o processo de aprendizagem, alguns problemas devem ser abordados. Primeiro, como cada avaliação é custosa, a seleção de um bom subconjunto de abstrações deve ocorrer após poucas avaliações. Segundo, como o processo de aprendizagem é estocástico, a cada avaliação de um mesmo subconjunto de abstrações obtém-se diferentes resultados. Terceiro, a interação entre as abstrações elegíveis influencia a qualidade do processo de aprendizagem.

O método proposto utiliza uma abordagem híbrida baseada nas abordagens *Wrapper* e *Filter*. Como na abordagem *Wrapper*, uma busca no espaço de subconjunto de abstrações

é realizada, experimentando subconjuntos de abstrações para realizar um processo de aprendizagem; como na abordagem *Filter*, após o teste de alguns subconjuntos de abstrações, um ranqueamento entre as abstrações é realizado. O ranqueamento de abstrações é utilizado de duas maneiras: (i) selecionando o melhor subconjunto dada uma quantidade de abstrações; e (ii) guiando os próximos subconjuntos a serem experimentados. Os algoritmos propostos evoluem por rodadas, repetindo os seguintes passos: (i) sorteia-se n subconjuntos de abstrações; e (ii) avalia a relevância de cada abstração espacial em $\mathcal{F}$ e utiliza essa informação na próxima rodada.

1.2 Organização do Texto

Esta dissertação está organizada da seguinte forma: O capítulo 2 apresenta uma revisão da literatura científica associada ao trabalho desenvolvido nesta dissertação. O capítulo 3 contempla os fundamentos relacionados ao Aprendizado por Reforço e fornece informações sobre os tradicionais algoritmos utilizados. O capítulo 4 mostra o ambiente utilizado nos experimentos e a motivação para o problema de seleção de abstrações. O capítulo 5 apresenta a proposta da dissertação, enquanto o capítulo 6 traz os experimentos e resultados alcançados. Finalmente, o capítulo 7 apresenta as conclusões.

2 Revisão da Literatura

As técnicas utilizadas por agentes em problemas de Aprendizado por Reforço (AR) utilizam algoritmos que buscam resolver problemas de tomada de decisão sequencial. Em problemas relacionados, o agente tenta maximizar a recompensa acumulada que recebe do ambiente (SUTTON; BARTO, 1998), desenvolvendo a capacidade de otimizar seu comportamento em ambientes estocásticos. Como o processo de aprendizado é realizado por tentativa e erro e repetidas iterações com o ambiente, o agente pode consumir muito tempo antes de aprender a tomar as decisões mais adequadas, tornando a convergência lenta.

Uma configuração comum em problemas no AR, envolve o uso de uma vasta quantidade de variáveis de estados que gera, por consequência, a maldição da dimensionalidade. O agente precisa aprender um número de parâmetros que cresce exponencialmente de acordo com o tamanho do espaço de estados e, conseqüentemente, o aprendizado é inviabilizado e o custo computacional é muito elevado. Nesse contexto, lidar com esse tipo de situação e aplicar algoritmos de AR, requer técnicas que exploram a similaridade dos problemas e buscam obter uma representação compacta daquilo que precisa ser aprendido, removendo, por vezes, dados irrelevantes ao aprendizado.

Algoritmos de aprendizado podem ser adaptados para que explorem alguma forma compacta de aproximação e possam aprender diretamente a função valor ou políticas. Essas soluções foram aplicadas com sucesso em domínios para controle e robótica e, exemplos bem sucedidos, incluem aproximação da função valor (SUTTON; BARTO, 1998), abstração de estado (LI; WALSH; LITTMAN, 2006). Outros métodos se concentram na seleção da melhor abstração, um subconjunto de componentes do estado, a partir de um conjunto de abstrações candidatas (DIUK; LI; LEFFLER, 2009; KONIDARIS; BARTO, 2009). A ideia geral é definir um tamanho adequado das abstrações, sendo menor do que o espaço de estado completo.

A abordagem mais popular para obter um subconjunto de abstrações consiste em uma construção incremental (GUYON; ELISSEEFF, 2003). Em cada rodada, considera-se um subconjunto de abstrações selecionadas $\mathcal{F}$ e o seu complemento $\overline{\mathcal{F}}$, então os subconjuntos $\mathcal{F} \cup \{F\}$ para todo $F \in \overline{\mathcal{F}}$ são avaliados e o subconjunto melhor avaliado torna-se o novo subconjunto de abstrações selecionadas $\mathcal{F}$; o processo continua enquanto há uma

melhoria na avaliação do subconjunto de abstrações $\mathcal{F}$ contra uma penalização no tamanho de $\mathcal{F}$. No AR, essa abordagem é tanto utilizada após o processo de aprendizado para encontrar um subconjunto de abstração que represente adequadamente a função valor (LI; WALSH; LITTMAN, 2006; PAINTER-WAKEFIELD; PARR, 2012) durante o aprendizado, considerando usualmente a diferença temporal como medida de avaliação (ZHAO et al., 2014; GERAMIFARD et al., 2011; GERAMIFARD; DANN; HOW, 2013). Embora não utilizado no AR, a abordagem incremental também pode ser feita na direção contrária, isto é, em cada rodada uma abstração F é eliminada do subconjunto de abstrações selecionadas $\mathcal{F}$.

Os pesquisadores Konidaris e Barto (KONIDARIS; BARTO, 2009), em seu trabalho, avaliam cada abstração a partir de um conjunto de abstrações candidatas, indicando seu valor. O uso de um critério padrão para a seleção de modelos *Bayesian Information Criterion* (BIC), penaliza claramente abstrações que geram alta dimensionalidade. O algoritmo proposto determina de forma iterativa o valor de BIC para habilidades representadas usando o *framework options*. Ainda, é proposto que o agente deve criar uma nova opção junto com uma abstração, a partir da escolha em uma biblioteca de abstrações (conjunto de abstrações candidatas), refinando a abstração selecionada por meio da experiência. Além disso, é citado pelos autores, que a seleção de abstrações abre a possibilidade de transferência de abstração, onde um agente que aprendeu um conjunto de habilidades pode se beneficiar das abstrações refinadas.

Enquanto a maioria dos trabalhos preocupa-se em encontrar um bom subconjunto de abstrações, devido ao caráter estocástico do AR, alguns trabalhos preocupam-se em como obter uma avaliação estável de um subconjunto de abstrações. A abordagem proposta por (SEIJEN; WHITESON; KESTER, 2013) avalia abstrações candidatas, determinando quais são as abstrações que produzem a melhor função valor. Isto é conseguido experimentando as diferentes abstrações e atualizando as ações correspondentes com o retorno produzido por essas abstrações. Como as abstrações são avaliadas com base na qualidade de sua política atual, as pequenas abstrações que aprendem rapidamente uma política adequada serão selecionadas na fase de aprendizado inicial, enquanto as abstrações que precisam de mais dados para obter uma política mais precisa, serão selecionadas posteriormente. Duas grandes restrições são impostas: todas as abstrações devem ser avaliadas e as abstrações devem apresentar a propriedade de Markov. A propriedade de Markov possibilita que todas as abstrações podem usufruir das experiências acumuladas, mesmo quando aquela

abstração não está em uso. (DIUK; LI; LEFFLER, 2009) consideram também o problema de escolher a melhor abstração entre abstrações candidatas, mas por não assumir a propriedade de Markov para as abstrações, cada uma das abstrações só pode ser avaliada separadamente, sendo que, neste caso, a maior preocupação é em tratar o problema de exploração e exploração na escolha de abstrações.

O trabalho de (BISHOP; MIIKKULAINEN, 2013) apresenta o *Temporal Difference Feature Evaluation* (TDFE), uma abordagem para o problema da avaliação de atributos por um agente online no AR. O *TDFE* combina a aprendizagem de função valor por métodos de diferença temporal com um algoritmo evolutivo que pesquisa o espaço de subconjuntos de atributos e produz um ranqueamento sobre todos os atributos de forma individual.

O algoritmo *Temporal Difference Feature Evaluation* (TDFE), utilizado em (BISHOP; MIIKKULAINEN, 2013) lida com problema da avaliação de atributos da seguinte forma: considera uma população de subconjuntos de atributos, ou seja, o *Feature Subset Population* (FSP) lida com vários subconjuntos de atributos diferentes em paralelo e o *Feature Subset Evaluator* (FSE) avalia cada subconjunto de atributos de acordo com o algoritmo de aprendizado usado pelo agente, atualizando estimativas estatísticas de forma incremental. Em seguida, a avaliação dos subconjuntos, combinar todas as estimativas estatísticas em uma única pontuação relacionada a qualidade do atributo e ranqueia todos os atributos. O *Subset Search GA*, pesquisa o espaço de subconjuntos de atributos para encontrar melhores combinações de atributos para a próxima iteração. O benefício causado pelo *Subset Search GA* é o de encontrar combinações de atributos que funcionem bem em subconjunto, o que melhora a qualidade do ranqueamento de funcionalidades do *TDFE*.

A estratégia de avaliar subconjuntos de abstrações e obter um ranqueamento de abstrações utilizada nesta dissertação também pode ser observada em outros trabalhos na literatura. O trabalho de (GAUDEL; SEBAG, 2010) formaliza a seleção de atributos como um problema de AR e utiliza Árvores de Monte Carlo para instanciar um algoritmo para seleção de atributos. O algoritmo constrói uma árvore, onde em cada nível da árvore um novo atributo é escolhido; expandir a árvore inteira significa avaliar todos os subconjuntos possíveis, no entanto, o algoritmo percorre caminhos na árvore que foram melhor sucedidos no passado para experimentar novas expansões. Nesta dissertação, o ranqueamento de abstrações permite escolher quais subconjuntos experimentar na próxima rodada, mas

com uma estrutura menos informativa que a apresentada por (SEIJEN; WHITESON; KESTER, 2013).

O trabalho apresentado por (BISHOP; MIIKKULAINEN, 2013) lida com o problema da avaliação de abstrações, em que considera uma população de subconjuntos de abstrações. Cada subconjunto de abstrações é avaliado de acordo com seu poder de representação, avaliado pela diferença temporal após aprendizado *off-line*, enquanto que uma medida de pontuação é utilizada para ordenar as abstrações com base na: ocorrência das abstrações em cada subconjunto, avaliação dos subconjuntos e importância da abstração na função aprendida. Diferentemente da abordagem nesta dissertação, a população de subconjuntos na próxima iteração depende apenas da população atual e das correspondentes avaliações, enquanto que o ranqueamento de abstrações é utilizado apenas para seleção final de abstrações.

A avaliação dos subconjuntos, combina todas as estimativas estatísticas em uma única pontuação relacionada a qualidade do atributo e os ranqueia. Por fim, realiza uma pesquisa no espaço de subconjuntos de atributos para encontrar melhores combinações entre os atributos existentes.

Desse modo, neste capítulo buscou-se mostrar uma visão geral e relatar alguns dos trabalhos disponíveis na literatura relacionados à avaliação de abstrações no Aprendizado por Reforço. O próximo capítulo apresenta os fundamentos relacionados ao Aprendizado por Reforço e os algoritmos utilizados, assim como discute sobre soluções para tratar problemas com grandes espaços de estados com o objetivo de acelerar o processo de aprendizado.

3 Fundamentação teórica

Métodos que utilizam Aprendizado por Reforço (AR) aprendem por tentativa e erro, o que significa que não é necessária qualquer informação antecipada sobre o ambiente. No AR, um agente repetidamente observa o estado de seu ambiente e, em seguida, escolhe e executa uma ação. Ao executar a ação o estado é alterado e o agente obtém uma recompensa numérica imediata como resultado. O agente deve aprender a escolher as ações de modo a buscar recompensas positivas e evitar as negativas.

O agente inicialmente não sabe o efeito que suas ações têm sobre o estado do ambiente, tampouco, não sabe qual é a melhor ação a ser escolhida. Em vez disso, ele tenta várias ações em vários estados e, gradualmente, aprende o que é o melhor em cada estado, de modo a maximizar sua recompensa a longo tempo. Esse processo de iteração é realizado por um algoritmo de aprendizado e resulta em uma política de ação.

Uma abordagem possível para aprender uma política consiste em experimentar várias ações em diferentes estados e aprender um modelo preditivo dos efeitos de suas ações, i.e., aprender o modelo de transição e quais recompensas estão associadas a cada estado do ambiente. Diante de tal modelo, o agente poderia planejar com antecedência e prever cursos alternativos para suas ações, estados e recompensas. No entanto, existem algoritmos que não se baseiam em modelos, mas apenas na construção de uma função valor, sendo assim, pode-se aprender uma política ideal mesmo sem representar os efeitos que suas ações produziram (WATKINS; DAYAN, 1992).

Este capítulo fornece informações que fundamentam a base teórica para o restante da dissertação. As seções 3.1 e 3.2, respectivamente, conceituam o AR e apresentam os algoritmos utilizados no AR. A partir da seção 3.3 são apresentadas as técnicas utilizadas nesta dissertação para lidar com espaço de estados arbitrariamente grande e acelerar o aprendizado.

3.1 Aprendizado por Reforço

A Figura 1 exibe o modelo conceitual do problema de AR, em que o agente interage com o ambiente, percebendo um estado s_t e selecionando uma ação a_t ; então o agente recebe do ambiente uma recompensa r_t e o ambiente transita para o novo estado s_{t+1} .

Figura 1 – Interação entre agente e ambiente no Aprendizado por Reforço.

Fonte: Adaptado de (WATKINS; DAYAN, 1992)

Como o objetivo do agente é acumular recompensas positivas e evitar as negativas, então a solução para o problema de AR consiste em uma estratégia para decidir qual ação será executada em qualquer estado do ambiente. Considera-se que essa é uma tarefa de otimização e que o agente deve aprender a escolher ações que maximizem as recompensas obtidas ao longo do tempo.

Problemas de AR são modelados por um Processo Markoviano de Decisão (PMD) (PUTERMAN, 1994) (*Markov Decision Process*). Um PMD é definido por uma tupla (S, A, T, R) , onde S é um conjunto de estados; A é um conjunto de ações; $T : S \times A \times S \rightarrow [0, 1]$ é uma função de transição, de tal modo que $T(s, a, s')$ indica a probabilidade de transição para um estado s' após executar uma ação a no estado s ; $R : S \times A \rightarrow \mathbb{R}$ é uma função de recompensa, que indica a recompensa imediata esperada após uma ação a ser executada em um estado s .

O agente acumula experiência $s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_n, a_n, r_n$, enquanto interage com o ambiente. Um algoritmo de AR deve resolver duas questões relacionadas ao problema em que está inserido: (i) como obter uma política de ação ideal? e (ii) como interagir com o ambiente, ou seja, como escolher ações? Uma política π mapeia cada estado possível $s \in S$ para uma ação $a \in A$, i.e., $\pi : S \rightarrow A$. Uma política π pode ser avaliada por uma função $V^\pi : S \rightarrow \mathbb{R}$ que atribui a cada estado $s \in S$ a soma da recompensa esperada quando se inicia em s , isto é, o valor de s . V^π é chamada de função valor da política π e no modelo horizonte infinito descontado é definida como:

$$V^\pi(s) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, \pi \right] \quad (1)$$

onde $\gamma \in (1, 0)$ é um fator de desconto.

A função valor $Q^*(s, a) = r(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^*(s')$ indica o valor de escolher a ação a no estado s e seguir com a política ótima deste momento em diante. A função valor $Q^*(\cdot)$ permite definir uma política ótima por $\pi^*(s) = \arg \max_{a \in A} Q^*(s, a)$.

Algoritmos bem conhecidos na literatura como Q-learning (WATKINS; DAYAN, 1992) e Sarsa (RUMMERY; NIRANJAN, 1994) estimam a função valor Q em cada iteração, realizando uma atualização com base na experiência atual $s_t, a_t, r_t, s_{t+1}, a_{t+1}$. Para que a estimativa da função Q convirja é necessário lidar com o dilema de exploração-exploração, definindo uma estratégia de seleção da ação que garanta um equilíbrio entre a exploração, que consiste na decisão de aprender mais sobre as partes desconhecidas do ambiente e a exploração, que utiliza o conhecimento adquirido para obter recompensas. Uma maneira simples e efetiva para lidar com o equilíbrio entre exploração e exploração é a estratégia ϵ -greedy, que considera uma taxa de exploração com probabilidade ϵ (ações selecionadas de forma aleatória e uniforme) e exploração com probabilidade $1 - \epsilon$. Usualmente, o parâmetro ϵ decai ao longo do tempo, isto é, utiliza-se a estratégia ϵ -greedy no tempo t ao utilizar $\epsilon(t)$.

3.2 Algoritmos no AR

Nesta subseção são apresentados mais alguns detalhes sobre os algoritmos tradicionais utilizados no AR, o primeiro algoritmo é o Q-learning e, na sequência, o algoritmo Sarsa.

Introduzido por Watkins (WATKINS; DAYAN, 1992), o algoritmo Q-learning é considerado *off-policy*, i.e., independe da política a ser seguida. O Q-learning considera uma estimativa Q_0 para a função valor ação ótima Q^* e, a cada passo de tempo, o agente observa o estado s_t e, em seguida, escolhe e executa uma ação a_t . O processo segue para o próximo estado s_{t+1} , o agente recebe uma recompensa r_t . A estimativa para Q^* é atualizada por:

$$Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \alpha \left[r_t + \gamma \max_{a' \in A} Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t) \right], \quad (2)$$

onde α é a taxa de aprendizado e γ é o fator de desconto. Se a taxa de aprendizado for definida apropriadamente, de modo que o valor de α decaia corretamente, o algoritmo converge.

Como observado no Q-learning, algoritmo 1, uma estratégia para seleção da ação deve ser especificada e, como comentado anteriormente, a estratégia ϵ -greedy é uma

possibilidade. A exploração pode ser feita simplesmente executando uma das ações com probabilidade uniforme e a exploração ser realizada, escolhendo a ação que maximize a estimativa atual de Q^* .

Algoritmo 1 Algoritmo Q-learning

```

1: Inicialize  $Q(s, a)$  arbitrariamente,  $\forall s, a \in S, A$ 
2: para todo episódio faça
3:     Inicialize o estado inicial  $s$ 
4:     repita
5:          $a \leftarrow$  escolha ação para ser executada
6:         execute a ação  $a$ , observe a recompensa  $r$  e o próximo estado  $s'$ 
7:          $Q(s, a) = Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$ 
8:          $s \leftarrow s'$ 
9:     até fim de episódio
10: fim

```

Fonte: Adaptado de (WATKINS; DAYAN, 1992)

O algoritmo Sarsa foi proposto por Rummerly e Niranjan (RUMMERLY; NIRANJAN, 1994) e diferencia-se do algoritmo Q-learning apenas na atualização da estimativa para Q^* . Enquanto o algoritmo Q-learning utiliza a estimativa de $V^*(s) = \max_{a' \in A} Q^*(s, a')$, independente da ação a ser executada no próximo passo, o algoritmo Sarsa utiliza a estimativa de $Q^*(s, a')$, em que a' é a ação executada no próximo estado. Como é um algoritmo *on-policy*, depende da política executada.

O nome Sarsa está relacionado ao modo como as atualizações são realizadas, usando a tupla $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$. O algoritmo Sarsa começa com uma inicialização arbitrária da função Q_t e a atualiza em cada iteração da seguinte forma:

$$Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \alpha(r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)), \quad (3)$$

e da mesma forma, α é a taxa de aprendizado e γ é o fator de desconto.

O operador máximo existente no Q-learning é substituído pela estimativa do valor da próxima ação de acordo com a estratégia de escolha de ações. A ação a_{t+1} selecionada não é necessariamente a ação com o maior valor, ou seja, aquela que maximiza $Q_t(s_{t+1}, a)$, mas é a ação escolhida de acordo com alguma estratégia implementada.

O algoritmo 2 descreve o algoritmo Sarsa e, como no algoritmo 1 Q-learning, uma estratégia para seleção da ação deve ser especificada da mesma forma, aplica-se a estratégia de escolha da ação ϵ -greedy.

Para ocorrer a convergência a uma política ótima, é necessário que o parâmetro ϵ diminua no tempo proposto até chegar a zero. Observa-se que o algoritmo Sarsa proporciona

Algoritmo 2 Algoritmo Sarsa

```

1: Inicialize  $Q(s, a)$  arbitrariamente,  $\forall s, a \in S, A$ 
2: para todo episódio faça
3:   Inicialize o estado inicial  $s$ 
4:    $a \leftarrow$  escolha ação para ser executada com base em  $s$ 
5:   repita
6:     execute a ação  $a$ , observe a recompensa  $r$  e o próximo estado  $s'$ 
7:      $a' \leftarrow$  escolha a ação ser executada com base em  $s'$ 
8:      $Q(s, a) = Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$ 
9:      $s \leftarrow s'$ 
10:     $a \leftarrow a'$ 
11:   até fim de episódio
12: fim

```

Fonte: Adaptado de (WATKINS; DAYAN, 1992)

melhores características de convergência, quando aproximações são utilizadas na estimativa da função Q^* . Nesta dissertação, como formalizado na seção 3.4, o algoritmo Sarsa é combinado com o aproximador de função *Tile Coding*.

3.3 Acelerando o aprendizado

As condições para convergência dos algoritmos de AR em geral implicam em: (i) visitar todos estados possíveis infinitas vezes e (ii) executar todas as ações possíveis para cada estado infinitas vezes. Mesmo que boas soluções possam ser encontradas antes de “infinitas” visitas, quanto maior a quantidade de estados e ações disponíveis, mais longo é o tempo necessário para aprender boas soluções. Portanto, a estratégia empregada no AR faz com que o agente gaste muito tempo resolvendo determinado problema.

Nesse sentido, as abordagens utilizadas para transferir conhecimento no AR visam melhorar o processo de aprendizagem, seja no mesmo problema por meio de generalização, ao explorar a similaridade entre estados semelhantes ou em problemas diferentes do original, ao transferir conhecimento adquirido em um problema para o aprendizado em outros problemas.

A abstração de estados é uma forma de obter generalização, pois visa agregar os estados aos quais o agente deve tratar como semelhantes (JONG; STONE, 2005), permitindo reduzir a complexidade de um problema, desconsiderando algumas propriedades irrelevantes (PONSEN; TAYLOR; TUYLS, 2010) e preservando a informação relevante para resolver um determinado problema.

Nesta dissertação, considera-se que por meio de um conjunto de abstrações, seja permitido obter um aproximador de função que mapeie o espaço de estado original para um espaço de estado menor, assegurando que o algoritmo Sarsa junto ao aproximador de função, lide diretamente com os estados mapeados e com a quantidade de parâmetros a serem aprendidos.

A representação tabular precisa de $|S| \times |A|$ espaço em memória e, dessa forma, pode ser pensada como uma função com $|S| \times |A|$ parâmetros. Se uma classe de funções parametrizada $\mathcal{C}$ for escolhida para que se possa aproximar adequadamente a função $Q(s, a)$ e a quantidade de parâmetros $N_{\mathcal{C}}$ for muito menor que $|S| \times |A|$, isto é, $N_{\mathcal{C}} \ll |S| \times |A|$, pode-se aprender uma política adequada em um tempo menor.

Considere uma função $q : S \times A \times \mathbb{R}^{N_{\mathcal{C}}} \rightarrow \mathbb{R}$, onde $q(s, a; \theta)$ indica o valor do estado s quando a ação a é executada e assume-se o vetor de parâmetros $\theta \in \mathbb{R}^{N_{\mathcal{C}}}$. Considere ainda que a classe de funções $\mathcal{C}$ é derivável no vetor de parâmetros, isto é, pode-se calcular o gradiente $\nabla_{\theta} q(s, a; \theta)$. Note que agora aprender a função Q^* consiste em estimar o vetor de parâmetros θ mais apropriado.

A versão com aproximador de função do algoritmo Sarsa é apresentada no algoritmo 3. Note que a versão tabular desse algoritmo é um caso especial da versão aqui apresentada, uma vez que na versão tabular o gradiente consiste em um vetor de zeros com um único 1 presente, justamente o parâmetro que refere-se ao par (s, a) .

Algoritmo 3 Algoritmo Sarsa com aproximador de função

```

1: Inicialize  $\theta \in \mathbb{R}^{N_{\mathcal{C}}}$  arbitrariamente
2: para todo episódio faça
3:   Inicialize o estado inicial  $s$ 
4:    $a \leftarrow$  escolha ação para ser executada com base em  $s$ 
5:   repita
6:     execute a ação  $a$ , observe a recompensa  $r$  e o próximo estado  $s'$ 
7:      $a' \leftarrow$  escolha ação para ser executada com base em  $s'$ 
8:      $\theta = \theta + \alpha \nabla_{\theta} q(s, a; \theta) [r + \gamma Q(s', a') - Q(s, a)]$ 
9:      $s \leftarrow s'$ 
10:     $a \leftarrow a'$ 
11:   até fim de episódio
12: fim
```

Fonte: Adaptado de (WATKINS; DAYAN, 1992)

3.4 Abstração Espacial: *Tile Coding*

Até agora, os estados foram considerados apenas como elementos em um conjunto S , os quais podem ser enumerados: $s_1, s_2, \dots, s_{|S|}$. Embora existam técnicas que acelerem o aprendizado, mesmo considerando estados enumerados, sem considerar nenhuma estrutura no conjunto de estados, os resultados são limitados. Uma forma de estruturar o conjunto de estados é utilizar representação fatorada.

Considere n variáveis de estado $(X_1, X_2, X_3, \dots, X_n)$, um estado fatorado é um vetor $\vec{x} = (x_1, x_2, x_3, \dots, x_n)$ com $x_i \in R_i$, sendo que todo $s \in S$ pode ser expresso como $\vec{x}$. Considera-se implicitamente que exista uma função:

$$F : S \rightarrow \prod_{i=1}^n R_i,$$

onde R_i é o domínio do fator i .

Um caso típico encontrado em problemas reais ocorre quando os fatores de um estado são valores contínuos, isto é, $R_i = \mathbb{R}$. Uma solução para criar uma função de aproximação consiste na discretização de cada domínio, voltado para uma representação tabular (função de aproximação linear e gradiente de zeros com um único 1). Porém é preciso definir um equilíbrio entre: intervalos discretos grandes que aceleram o aprendizado, mas pioram a qualidade da aproximação, e ter intervalos discretos pequenos que melhoram a qualidade da aproximação, mas que deixa o aprendizado mais lento. Uma solução para lidar com esse problema é a utilização de *Tile Coding*.

A discretização de espaço de estado contínuo em várias camadas (*tilings*) foi introduzida pela primeira vez por Albus em 1971 como *Cerebellar Model Articulator Controller (CMAC)*, posteriormente foi nomeado para *Tile Coding* (SUTTON; BARTO, 1998) e tem sido utilizado como aproximador de função no AR. O *Tile Coding* é composto por várias partições diferentes do espaço de estado, chamadas *tilings* e cada partição é composta por subpartições denominadas *tiles*. Um campo receptivo em um *tiling* é chamado de *tile*.

A Figura 2 ilustra como um espaço contínuo bidimensional pode ser discretizado de forma uniforme ao longo de duas camadas. A figura da esquerda oferece uma vista de cima, enquanto a figura da direita ilustra o conceito da visão lateral. Cada *tile* representa uma característica binária. A Figura 2 mostra como *Tile Coding* mapeia um ponto a

Figura 2 – *Tile Coding* em um espaço bidimensional. Vista superior à esquerda e lateral à direita. Nota-se que o *tiling* 2 está ligeiramente deslocada no espaço 2D em relação ao *tiling* 1.

Fonte: Adaptado de (WATKINS; DAYAN, 1992)

partir do espaço de estado para um conjunto de características ativas, localizando a região correspondente (*tile* em destaque) dentro de cada camada.

A abordagem *Tile Coding* considera um conjunto de abstrações $\{F_1, \dots, F_k\}$ e aproxima os valores da função Q pela soma de cada função tabular (SUTTON; BARTO, 1998; WHITESON; TAYLOR; STONE, 2007; STONE; SUTTON; KUHLMANN, 2005). Formalmente o *Tile Coding* utiliza a seguinte estrutura:

$$q(s, a; \theta^1, \dots, \theta^k) = \sum_{i=1}^k \theta_{F^i(s), a}^i = \sum_{i=1}^k Q^i(F^i(s), a) \quad (4)$$

onde k é a quantidade de *tilings*, θ^i é o vetor de parâmetros associado ao *tiling* i , $F^i(s)$ é o número do *tile* afetado pelo estado s no *tiling* i e $Q^i(\cdot)$ representa a função tabular que representa o *tiling* i .

3.5 Transferência de Conhecimento

As técnicas aplicadas para Transferência de Conhecimento (TC) reutilizam o conhecimento previamente aprendido a fim de resolver novos problemas. Geralmente os problemas de aprendizado compartilham propriedades em comum, pois compete ao agente aprender a solucionar o problema origem e utilizar essa solução para ajudar a resolver outros problemas.

A TC é formalizada quando há um problema origem a partir do qual o conhecimento é extraído e um problema destino para o qual o conhecimento é transferido (TAYLOR; STONE, 2009). Reutilizar o conhecimento, quando feito de forma apropriada, permite acelerar o aprendizado no problema destino.

O conhecimento pode ser transferido na inicialização do processo de aprendizagem ou ao longo de todo o processo. Porém é necessário determinar “O que transferir” e “Como transferir” (PAN; YANG, 2010), assumindo implicitamente uma relação entre os problemas origem e destino. Os tipos de conhecimento transferidos entre problemas variam de acordo com as diferentes abordagens existentes, comumente são transferidos: função valor (TAYLOR; STONE, 2005), por exemplo, aprendendo funções valores por meio de subconjuntos de abstrações compartilhadas entre as tarefas (KONIDARIS, 2006); política (FERNÁNDEZ; VELOSO, 2006), na qual um mapeamento definido pelo usuário é usado para relacionar a nova tarefa com a tarefa na qual uma política já foi aprendida (TAYLOR; WHITESON; STONE, 2006) (TORREY et al., 2006); amostra de experiências (LAZARIC; RESTELLI; BONARINI, 2008), que podem ser utilizadas para atualizar a função valor Q ; uma abstração especial específica (WALSH; LI; LITTMAN, 2006). Normalmente, dois pressupostos podem ser considerados quando o agente interage com a tarefa de origem: (i) a tarefa de origem é mais simples do que a de destino e (ii) o agente interage com o ambiente por um longo tempo arbitrário. Após descobrir qual conhecimento pode ser transferido, os algoritmos de aprendizagem, no problema destino, são instruídos a utilizar o conhecimento adquirido.

Poucos trabalhos relacionados à transferência de conhecimento, como os de (WALSH; LI; LITTMAN, 2006) e (FROMMBERGER; WOLTER, 2010) abordam a temática sobre transferir uma abstração. Nesta dissertação, considerando que o problema de origem e de destino utilizam os mesmos tipos de abstrações, um subconjunto composto por abstrações especiais será transferido entre os problemas. As abstrações selecionadas no problema origem são consideradas melhores e, portanto, quando transferidas espera-se também acelerar o processo de aprendizagem no problema destino.

3.6 Seleção de atributos

Embora o uso de abstrações espaciais possam acelerar o AR, projetar uma abstração espacial adequada automaticamente não é uma tarefa trivial. Sob a suposição de que exista *a priori* um conjunto de abstrações espaciais $\mathcal{F}$ e que o agente aprende usando a representação do *Tile Coding*, o problema de projetar uma abstração espacial pode ser posto como um problema de seleção de atributos. Considere um estado s , pode-se representar um estado s por um vetor de atributos $(F_1(s), F_2(s), \dots, F_{|\mathcal{F}|}(s))$, onde $F_i \in \mathcal{F}$ e $F_i \neq F_j$ para $i \neq j$. Então, projetar uma abstração de estado apropriada é o mesmo que selecionar um subconjunto de atributos.

Métodos para seleção de atributos são utilizados com propósitos de identificar atributos de acordo com um critério de avaliação e importância, diminuir o espaço de estados após identificar os atributos relevantes que representem o problema original e melhorar substancialmente o desempenho do agente no processo de aprendizado (GAUDEL; SEBAG, 2010). Esses atributos devem representar adequadamente o problema de aprendizado e, de um modo geral, os métodos de seleção de atributos são classificados em três abordagens: *Filter*, *Wrapper* e *Embedded* (TANG; ALELYANI; LIU, 2013).

O processo de seleção de atributos deve ignorar os atributos irrelevantes ou aqueles considerados redundantes com outros atributos que fazem parte do mesmo conjunto. Atributos irrelevantes não contribuem de forma positiva na solução do problema e os atributos redundantes não fornecem novas informações para solução do problema, visto que, essas informações são fornecidas por outros atributos (KUMAR; MINZ, 2014). Problemas no AR se beneficiam com os resultados do processo de seleção de atributos, por exemplo, utilizar um subconjunto com um número menor de atributos reduz, por consequência, o tamanho do espaço de estado e permite lidar com problemas com alta dimensionalidade, usando menos recursos computacionais.

A abordagem *Filter* emprega uma função de classificação para escolher os melhores atributos, empregando uma pontuação de relevância com base em uma sequência de avaliações, aplicando uma medida estatística e concedendo a cada atributo uma pontuação que permitirá classificá-lo por relevância. Intuitivamente, quanto mais relevante for o atributo maior será sua classificação. Para selecionar os atributos que devem fazer parte de um subconjunto é possível determinar um número fixo de atributos mais bem classificados

ou um número de atributos acima de um valor limite predefinido na classificação. Os algoritmos mais utilizados são *Relief* (KIRA; RENDELL, 1992) e *Fisher score* (DUDA; HART; STORK, 2001).

A abordagem *Wrapper*, ao invés de avaliar cada atributo individualmente, avalia os diferentes subconjuntos de atributos (KOHAVI; JOHN, 1997), verificando as possíveis combinações, o subconjunto de atributos que oferece o melhor desempenho é selecionado para uso final. Observa-se que, se existirem N atributos no total, então existem 2^N possíveis subconjuntos para avaliar, fazendo com que essa abordagem seja desvantajosa em relação ao uso de recursos computacionais. Por ser considerado como um problema de otimização combinatória, algumas heurísticas como *Hill climbing* (HALL, 2000) e *mixing forward and backward selection* (ZHANG, 2008) são utilizadas.

A abordagem *Embedded* seleciona atributos com base em critérios que são gerados durante o processo de aprendizado. Além disso, a abordagem descreve o problema de seleção de atributos como um problema de otimização no qual o uso de atributos são penalizados e utilizam técnicas construtivas como árvores de classificação ou técnicas baseadas em regularização.

No capítulo 5, uma proposta híbrida entre a abordagem *Wrapper* e *Filter* será apresentada para seleção de abstrações no AR.

4 Problema estudado e sua representação

Este capítulo tem como objetivo apresentar o ambiente no qual os experimentos serão executados, um simulador de futebol, assim como os conjuntos de abstrações que serão utilizados *a priori*.

4.1 Simulador de futebol

O simulador *Discrete 2D Soccer Simulator* (D2DSS), implementado em MATLAB pelo autor, orientador e colaboradores, permite definir facilmente ambientes multiagentes, visto a possibilidade existente na definição de suas configurações. Embora um simulador discreto possa ser menos realista do que um simulador contínuo, os problemas tratados não ficam mais fáceis por esse motivo.

No D2DSS, o estado do ambiente é totalmente observável e conta com uma representação fatorada, apresentada da seguinte forma: 4 fatores para a bola e 2 fatores por cada jogador. O estado da bola é representado por suas coordenadas verticais e horizontais, sua velocidade e sua direção; cada jogador é representado apenas pelas suas coordenadas verticais e horizontais. Por exemplo, se há um jogador na equipe *A* e um jogador na equipe *B*, então há 8 fatores no total.

No D2DSS, o agente é encarregado de controlar a equipe *A*, enquanto joga contra a equipe *B*. Cada agente da equipe *A* pode escolher entre 13 ações: mover para cada uma das quatro direções cardeais; chutar a bola para cada uma das quatro direções cardeais e, também, para as quatro direções intercardeais e realizar um *tackle*, que é tentar tomar a bola de outro agente. O chute é realizado somente quando o agente está em posse da bola; o *tackle* acontece somente quando o agente está perto da bola. Estas ações resultam em nenhum movimento para o agente.

Nos experimentos, o agentes da equipe *B* emprega um algoritmo de planejamento conhecido com A^* (A-Estrela), que considera uma dinâmica determinista do ambiente, heurística e um pequeno horizonte de planejamento. Esse agente é chamado de Agente de Planejamento e como nenhum aprendizado acontece no agentes de Planejamento, não há necessidade de preocupar-se com a teoria dos jogos.

O D2DSS apresenta muitas configurações quanto à estocasticidade, por exemplo, ajustes que dizem respeito à dinâmica do jogo. As configurações da dinâmica de jogo são definidas por meio das probabilidades de sucesso do agente carregar a bola, avançar sem a bola, acertar o chute, realizar um *tackle*, da bola fazer curva e a velocidade máxima da bola quando um agente acerta o chute. Finalmente, também é possível controlar o número de jogadores e o tamanho do campo.

A Figura 3 mostra um exemplo com configurações que definem o uso de dois agentes em um campo com dimensão onze por sete (11×7). Como o ambiente utiliza uma representação fatorada, obtêm-se os seguintes fatores e possíveis valores:

- f_1 - posição da bola no eixo x (11 valores);
- f_2 - posição da bola no eixo y (7 valores);
- f_3 - direção da bola (9 valores);
- f_4 - velocidade da bola (3 valores);
- f_5 - posição do jogador A no eixo x (11 valores);
- f_6 - posição do jogador A no eixo y (7 valores);
- f_7 - posição do jogador B no eixo x (11 valores);
- f_8 - posição do jogador B no eixo y (7 valores).

Figura 3 – Simulador de Futebol D2DSS.

Fonte: Cleiton Alves da Silva, 2017

A Figura 3 mostra a bola (quadrado vermelho) e dois agentes interagindo com o ambiente no simulador de futebol. Cada jogador está posicionado no campo de acordo com suas coordenadas verticais e horizontais e o jogador da equipe A (cinza claro) tem como

objetivo aprender a marcar mais gols do que o jogador da equipe B (cinza escuro). O estado observado pelo agente é representado pelas variáveis originais do problema, enquanto que o espaço de ações é dado pelas 13 ações disponíveis para o agente executar. Na Figura 3 obtêm-se informações sobre os valores dos fatores que representam o estado para a posição vertical e horizontal, direção e velocidade da bola, além das posições verticais e horizontais dos jogadores. Os fatores que representam o estado exatamente como observado na figura são, respectivamente 3, 4, 1, 1, 4, 4, 5 e 2. Por exemplo, o valor 3 é a posição da bola na coordenada horizontal e o valor 4 é posição da bola na coordenada vertical.

4.2 Agente comportamental e atômico

Os agentes aprendizes baseiam suas ações na descrição de estado fornecidas pelo ambiente do simulador. Nesta dissertação são utilizados dois tipos de agentes aprendizes: atômico e comportamental. A diferença entre os tipos de agentes diz respeito apenas às ações disponíveis. O agente atômico tem disponível em qualquer estado as 13 ações definidas no simulador e são enumeradas da seguinte forma:

- Movimentar: norte (1), sul (2), leste (3), oeste (4);
- Chutar: norte (5), sul (6), leste (7), oeste (8), noroeste (9), sudeste (10), nordeste (11), sudoeste (12) e
- Tomar a bola (*tacke*): (13).

O agente comportamental realiza ações com o propósito de facilitar a marcação de um gol e como há um conjunto de ações heurísticas dependendo do contexto, ou seja, do estado, as ações disponíveis variam da seguinte forma:

- Se o agente está longe da bola (distância Manhattan maior que 5), existe apenas uma ação possível: ir na direção da bola (1);
- Se o agente está em posse da bola existem 4 ações possíveis: driblar na direção do gol (2), chutar para frente (3), chutar na diagonal esquerda (4), chutar na diagonal direita (5) e
- Caso contrário (agente próximo a bola) existem 5 ações possíveis: mover para norte (6), sul (7), leste (8), oeste (9) e *tackle* (10).

O agente que implementa o algoritmo de planejamento considera uma dinâmica determinista do ambiente e o seu comportamento pode ser ajustado de forma a influenciar diretamente o modo como suas ações são executadas. O Agente de Planejamento pode ser configurado para atuar com ações ε -greedy, ou seja, com $1-\varepsilon$ seleciona uma ação heurística planejada a cada passo de tempo e seleciona uma ação aleatoriamente uniforme no tempo com ε . Esse ajuste é definido em três opções: (i) $\varepsilon = 1.0$, as ações são baseadas em planejamento, (ii) $\varepsilon = 0.5$, as ações são baseadas em planejamento e ações aleatórias, com a mesma proporção e (iii) $\varepsilon = 0.0$ ações totalmente aleatórias.

Observa-se que, quanto maior o valor do parâmetro ε , o agente de planejamento se torna menos estocástico e permanece a maior parte do tempo da execução em posse da bola e tem mais possibilidade de marcar gols comparado ao agente Sarsa. Esse comportamento influencia o agente atômico, por exemplo, ao executar ações para descobrir se o agente de planejamento está em posse da bola.

4.3 *Projetando abstrações espaciais*

Com a configuração onze por sete (11×7) e um jogador em cada equipe, o espaço de estados conta com uma quantidade de $1,2326 \times 10^7$ estados. Para gerar um ambiente com espaço de estados maior, no qual posteriormente um conjunto de abstração será transferido, a definição dos parâmetros para o tamanho do campo de jogo foi alterada em duas configurações: uma para um campo com dimensão vinte e três por quinze (23×15), gerando uma quantidade de $1,1087 \times 10^9$ estados e outra para trinta e três por vinte um (33×21), resultando em um espaço de estados com $8,9859 \times 10^9$ estados.

Nos experimentos relatados, o algoritmo Sarsa é combinado ao aproximador de função *Tile Coding* e considera um conjunto de funções tabulares Q^i e aproxima a função Q pelo somatório dessas funções, como apresentado na equação 4. Cada função Q^i considera uma abstração espacial F^i diferente, possibilitando a generalização do conhecimento de forma diferente em cada função e permitindo aproximar a função Q . Nesta dissertação são considerados três tipos de abstrações espaciais: Canônica, Delta e Joint.

A abstração Canônica contém as variáveis diretas do espaço de estados e representa os atributos originais do problema. Embora em alguns problemas os atributos originais possam ajudar na construção da função Q , independentemente, essa não é uma propriedade

geral. Assim considera-se duas formas de combinações entre abstrações Canônicas: Delta, a diferença entre duas abstrações Canônicas e Joint, o produto cartesiano entre duas abstrações que podem ser do tipo Canônica ou Delta.

Considere que um estado s seja definido por uma d -tupla $(f_1(s), f_2(s), \dots, f_n(s))$ e que cada função $f_i(s) : S \rightarrow R_i$ tenha como imagem um conjunto R_i finito. Formalmente as abstrações são definidas da seguinte maneira:

- A abstração canônica C é criada a partir da utilização dos fatores que representam o estado original do problema, isto é, $F_i^C(s) = f_i(s)$ com imagem R_i^C de tal modo que $|R_i^C| = |R_i|$. Considere ainda a abstração canônica especial $F_0^C(s) = cte$ com $|R_0^C| = 1$.
- A abstração Delta D é criada a partir da combinação existente entre a subtração de duas abstrações Canônicas, isto é, $F_{ij}^D(s) = F_i^C(s) - F_j^C(s)$ onde $i < j$ com imagem R_{ij}^D de tal modo que $|R_{ij}^D| = |R_i| + |R_j| - 1$. Considere ainda a abstração Delta especial $F_{0,0}^D(s) = cte$ com $|R_{0,0}^D| = 1$. Note que, a consideração de F_0^C faz com que o conjunto de abstrações do tipo canônica esteja contido no conjunto de abstrações do tipo Delta.
- A abstração Joint J é criada a partir do produto cartesiano dois a dois entre duas abstrações Deltas, isto é, $F_{ij \times kl}^J(s) = F_{ij}^D + |R_{ij}^D| \times F_{kl}^D$ onde $(i, j) < (k, l)$ com imagem $R_{ij \times kl}^J$ de tal modo que $|R_{ij \times kl}^J| = |R_{ij}^D| \cdot |R_{kl}^D|$. Note que, a consideração de $F_{0,0}^D$ faz com que o conjunto de abstrações do tipo Delta esteja contido no conjunto de abstrações do tipo Joint.

O tamanho das tabelas necessárias para o aprendizado depende do domínio de cada abstração e em cada tipo de abstração é diferente. Uma abstração do tipo Delta tem um domínio 2 vezes maior que o domínio de uma abstração Canônica, enquanto que uma abstração do tipo Joint tem um domínio que é o quadrado do domínio de uma abstração do tipo Delta.

As combinações dois a dois entre as abstrações Canônicas definem a quantidade de abstrações do tipo Delta. Seja c a quantidade de fatores na descrição do estado, então, a quantidade de abstrações do tipo Delta é definida por $\frac{c(c+1)}{2}$. Como há 8 fatores que definem as abstrações do tipo Canônicas, então são criadas $\frac{8(8+1)}{2} = 36$ abstrações do tipo Delta.

Nos experimentos realizados, considera-se uma quantidade de 8 abstrações Canônicas, resultando em 36 abstrações do tipo Delta. Por exemplo, uma abstração do tipo Delta, entre as 36 possíveis, pode ser criada ao combinar a diferença entre as abstrações Canônicas F_1^C , posição da bola no eixo x e F_2^C posição da bola no eixo y . Dessa forma, constrói-se uma nova abstração $F_{1,2}^D$, que informa a posição exata da bola em campo. Da mesma forma, seja d a quantidade de abstrações do tipo Delta, então a quantidade de abstrações do tipo Joint é dada por $\frac{d(d+1)}{2}$. Nos experimentos realizados há $\frac{36(36+1)}{2} = 666$ abstrações do tipo Joint, ou seja, combinações realizadas com as abstrações do tipo Delta.

4.4 Estudo de caso

Antes de prosseguir para a proposta, apresenta-se uma breve análise sobre o resultado do aprendizado desenvolvido por meio de um agente Sarsa utilizando alguns subconjuntos de abstrações no simulador de futebol D2DSS.

As Figuras 4 e 5 apresentam as curvas de aprendizagem para o agente atômico e para o agente comportamental respectivamente, ambos implementando o algoritmo Sarsa e utilizando 5 diferentes subconjuntos de abstrações. Abstrações Canônicas (8 abstrações), abstrações Deltas puras, isto é, sem incluir abstrações Canônicas (28 abstrações), abstrações Joint de abstrações Canônicas (28 abstrações), as 64 abstrações anteriores juntas e todas as 666 abstrações do tipo Joint.

Figura 4 – Curvas de aprendizagem. Resultado obtido com um agente Sarsa, executando ações primitivas e utilizando 5 diferentes subconjuntos de abstrações.

Fonte: Cleiton Alves da Silva, 2017

Figura 5 – Curvas de aprendizagem. Resultado obtido com um agente Sarsa, executando ações comportamentais e utilizando 5 diferentes subconjuntos de abstrações.

Fonte: Cleiton Alves da Silva, 2017

Nas Figuras 4 e 5 pode-se observar que o tempo para aprender do agente comportamental é quatro vezes mais rápido que o tempo do agente atômico (compare a escala das abscissas em ambos os gráficos), mesmo ciente que o aprendizado ainda está ocorrendo. As figuras também confirmam que o aprendizado não acontece se apenas as abstrações Canônicas são utilizadas. Para proporcionar que o aprendizado ocorra, o uso das abstrações do tipo Delta e/ou Joint realmente serão necessárias. No restante desta dissertação serão discutidas e avaliadas duas perguntas:

- Qual é o melhor subconjunto de abstrações para acelerar o processo de aprendizagem?
- Esse subconjunto de abstrações pode ser transferido entre problemas similares?

5 Proposta

Este capítulo apresenta a proposta da dissertação e os algoritmos empregados como parte da solução do problema. Aqui será apresentado um método para encontrar um bom subconjunto de abstrações espaciais baseado em ranqueamento.

5.1 Seleção de abstração por meio da avaliação do processo de aprendizagem

Parte essencial em problemas que envolvem seleção de abstrações está relacionada em como avaliar um subconjunto de abstrações. No AR, geralmente, considera-se um conjunto de abstrações $\mathcal{F}$ que pode representar a função valor e, por consequência, essa função pode ser aprendida por meio de uma representação com tais abstrações. A seleção de abstrações pode definir um subconjunto com uma quantidade menor de abstrações que, ainda assim, representem bem a função valor e que seja capaz de acelerar o aprendizado.

Esta dissertação propõe uma estratégia para avaliar um subconjunto de abstrações pela influência que promovem durante o processo de aprendizagem. Duas dificuldades principais surgem desta abordagem: (i) o processo de aprendizagem deve ser repetido inúmeras vezes e, por isso, considera-se que esta abordagem para seleção de abstrações é aplicada apenas para o cenário em que há transferência de conhecimento e (ii) não há uma maneira única de avaliar o processo de aprendizagem.

Também, avalia-se o processo de aprendizagem considerando a quantidade de recompensa acumulada durante o processo de aprendizado por um período fixo de iterações, mas opções diferentes podem ser consideradas. Por exemplo, o desempenho da política aprendida no final do processo de aprendizagem pode ser considerado ou quando possível avaliar, considera-se o tempo de convergência. De alguma forma, a recompensa acumulada combina o desempenho da política e o tempo para a convergência; o período fixo de iterações é que pondera a importância de cada critério. Se o período fixo é longo, a importância é dada ao desempenho da política aprendida e se o período é curto, a importância é dada ao tempo para a convergência. Nos experimentos realizados no capítulo 6, o período fixo para aprendizagem foi definido empiricamente.

Outro ponto sobre a avaliação do processo de aprendizagem é a estocasticidade, que origina de dois lugares: (i) o próprio ambiente é estocástico e (ii) o dilema entre exploração e

exploração é estocástico. Para avaliar um subconjunto de abstrações adequadamente, médias de várias repetições do processo de aprendizagem devem ser consideradas, aumentando o tempo necessário para a seleção de abstrações. Para lidar com a estocasticidade propõe-se um método híbrido usando as abordagens *Wrapper* e *Filter*: (i) subconjuntos de abstrações são avaliados, (ii) cada abstração é avaliada separadamente com base na avaliação dos subconjuntos em que a abstração participa e (iii) subconjuntos de abstrações a serem avaliados são escolhidos com base na avaliação de cada abstração.

Nos algoritmos baseados na estratégia *Wrapper* pura, o subconjunto de abstrações selecionado como melhor é simplesmente o subconjunto de abstrações com a melhor avaliação dentre os avaliados. Como a quantidade de subconjuntos avaliados deve ser pequena, por conta do alto custo de avaliação, pode ser difícil encontrar um bom subconjunto de abstrações. Dessa forma uma pontuação m é obtida para cada abstração (abordagem *Filter*) utilizando a seguinte avaliação:

$$m(c) = \frac{Soma(c)}{N_{ocor}(c)} - \frac{Diff(c)}{N_E - N_{ocor}(c)},$$

onde $m(c)$ é a pontuação da abstração c , $Soma(c)$ é a soma total de recompensas obtidas pelos subconjuntos de abstrações que contêm a abstração c , $Diff(c)$ é a soma total de recompensas obtidas pelos subconjuntos de abstrações que não contêm a abstração c , $N_{ocor}(c)$ é a quantidade de subconjuntos de abstrações que contêm a abstração c e N_E é a quantidade total de subconjuntos de abstrações que foram avaliados. A pontuação $m(c)$ compara o desempenho dos subconjuntos de abstrações que utilizam a abstração c , contra os subconjuntos de abstrações que não utilizam a abstração c e é implementada no algoritmo 5 (`UpdateEval`).

As pontuações $m(c)$ permitem escolher um subconjunto de d abstrações, podendo d ser arbitrário, basta escolher as d primeiras abstrações segundo $m(c)$. O algoritmo aqui proposto considera um número de rodadas N_R e em cada rodada deve sortear número de subconjuntos N_E de abstrações para serem avaliados. Se os subconjuntos são escolhidos segundo uma distribuição uniforme, todas as abstrações possuem chances iguais de aparecerem e a pontuação $m(c)$ pode ser melhor estimada, por outro lado, é de interesse avaliar a relevância de uma abstração em bons subconjuntos de abstrações. Na próxima seção é apresentada a estratégia nomeada Otimização de População Evoluída por Ranqueamento (*Ranking Evolved Population Optimization* - **REPO**), nela considera-se que

nas primeiras rodadas subconjuntos de abstrações são escolhidos uniformemente, enquanto que nas rodadas finais os melhores subconjuntos são escolhidos através da pontuação $m(c)$.

5.2 Algoritmo *REPO*

A solução para o problema de selecionar um subconjunto de abstrações espaciais de tamanho fixo, o algoritmo *REPO*, é descrita no algoritmo 4. O algoritmo considera quatro parâmetros: N_F , N_R , N_E e T , sendo número de abstrações selecionadas, número de rodadas, número de avaliações por rodada e período de aprendizagem, respectivamente. O algoritmo *REPO* desenvolve-se em N_R rodadas e em cada rodada uma população com N_E subconjuntos de N_F abstrações espaciais é selecionada (linha 8). Para cada subconjunto selecionado $\mathbf{S}(i)$, um agente utiliza uma abstração de estado baseado no subconjunto $\mathbf{S}(i)$ e aprende durante T período de aprendizagem (linha 9), armazenando a recompensa acumulada (linha 10). A recompensa acumulada é usada para avaliar cada abstração nesse subconjunto de abstrações pelo algoritmo 5, *UpdateEval* (linha 12).

Para cada abstração em $\mathcal{F}$ (linha 7-14) o algoritmo *UpdateEval* acumula recompensas de experimentos quando a abstração foi considerada (linha 9) e quando a abstração não foi considerada (linha 12). Então a diferença da média de recompensas quando a abstração é considerada e a média de recompensas quando a abstração não é considerada é calculada (linha 20). Se uma abstração nunca aparece no subconjunto, a média é simplesmente calculada para ser 0 (linha 18) e se uma abstração aparece em todo subconjunto, a média é computada para ser a melhor média entre todas as abstrações (linha 25). Finalmente, a média anterior é combinada com a média da rodada atual (linha 27).

O algoritmo *REPO* define a probabilidade $Pr(F|\mathbf{e})$, considerando três abordagens diferentes: *Probabilistic Selection*, *Hard Selection* e *Uniform Selection*. Na primeira abordagem, *Probabilistic Selection*, escolhe-se um agendamento $(p_1, p_2, \dots, p_{N_R})$ onde $p_j \geq \frac{N_F}{|\mathcal{F}|}$ define uma probabilidade para cada rodada e, então, para cada rodada j

$$Pr(F_c|\mathbf{e}) = N_F \frac{\exp(\tau \mathbf{e}(c))}{\sum_{i=1}^{|\mathcal{F}|} \exp(\tau \mathbf{e}(i))},$$

onde o valor do parâmetro τ é escolhido para que $\max_{c \in \{1, \dots, |\mathcal{F}|\}} Pr(F_c|\mathbf{e}) = p_j$. Recomenda-se escolher um agendamento onde $p_i > p_j$ se $i > j$ e $p_1 = \frac{N_F}{|\mathcal{F}|}$. Por conseguinte, na primeira rodada todas as abstrações têm chances iguais de serem sorteadas, a partir da segunda

Algoritmo 4 Algoritmo REPO.

- 1: **Entrada:** Conjunto de abstrações $\mathcal{F}$, quantidade de abstrações selecionadas N_F , número de avaliações por rodada N_E , número de rodadas N_R , período de aprendizagem T
- 2: **Saída:** Subconjunto com N_F abstrações selecionadas
- 3: **para** $c = 1$ até $|\mathcal{F}|$
- 4: $\mathbf{e}(c) \leftarrow 0$
- 5: **fim para**
- 6: **para** $j = 1$ até N_R **faça**
- 7: **para** $i = 1$ to N_E **faça**
- 8: Selecione um subconjunto $\mathbf{S}(i) \subset \mathcal{F}$ com N_F abstrações considerando uma distribuição de probabilidade $Pr(F|\mathbf{e})$
- 9: Use um agente de Aprendizado por Reforço que desenvolva o aprendizado com as abstrações do subconjunto $\mathbf{S}(i)$ por T período
- 10: Armazene a recompensa em $\mathbf{r}(i)$
- 11: **fim para**
- 12: $\mathbf{e} \leftarrow \text{UpdateEval}(\mathbf{e}, \mathbf{r}, \mathbf{S})$
- 13: **fim para**
- 14: **retorne** as N_F abstrações segundo as avaliações $\mathbf{e}$

Fonte: Cleiton Alves da Silva, 2017

rodada o valor da probabilidade aumenta, permitindo mais chances para as abstrações com melhores avaliações fazerem parte do subconjunto.

Na segunda abordagem, *Hard Selection*, define-se a escolha do agendamento por $(n_1, n_2, \dots, n_{N_R})$, onde $n_j \in \{0, 1, \dots, N_F - 1\}$, definindo um número fixo de abstrações selecionadas para cada uma das rodadas, então em cada rodada j

$$Pr(F_c|\mathbf{e}) = \begin{cases} 1 & , \text{ se } c \text{ é } n_j\text{-melhor abstração} \\ Pr(F_c|\mathbf{e}) = \frac{N_F - n_j}{|\mathcal{F}| - n_j} & , \text{ caso contrário} \end{cases},$$

onde n_j -melhores abstrações significa que c é igual ou melhor que a posição n_j segundo a avaliação $\mathbf{e}$. Recomenda-se escolher um agendamento onde $n_i > n_j$ se $i > j$ e $n_1 = 0$. Então na primeira rodada todas as abstrações têm chances iguais de serem sorteadas e fazer parte do subconjunto. A partir da segunda rodada, pelo menos, uma nova abstração é selecionada, i.e., com certeza será retornada como uma abstração selecionada.

Na terceira abordagem, *Uniform Selection*, seleciona-se abstrações constantemente da mesma forma para cada rodada j :

$$Pr(F_c|\mathbf{e}) = \frac{N_F}{|\mathcal{F}|},$$

de modo a fazer com que cada abstração possua a mesma chance de participar do subconjunto.

Algoritmo 5 Algoritmo UpdateEval.

```

1: Entrada: Conjunto de abstrações  $\mathcal{F}$ , número de avaliações por rodada  $N_E$ , vetor com
   abstrações avaliadas anteriormente  $\mathbf{e}$ , vetor com recompensas acumuladas  $\mathbf{r}$ , conjunto
   com subconjunto de abstrações  $\mathbf{S}$ 
2: Saída: vetor atualizado com abstrações avaliadas  $\mathbf{e}$ 
3: para  $c = 1$  até  $|\mathcal{F}|$ 
4:    $Diff(c) \leftarrow 0, Soma(c) \leftarrow 0, N_{ocor}(c) \leftarrow 0$ 
5: fim para
6: para  $i = 1$  até  $N_E$  faça
7:   para  $c = 1$  até  $|\mathcal{F}|$  faça
8:     se  $c \in \mathbf{S}(i)$ 
9:        $Soma(c) \leftarrow Soma(c) + \mathbf{r}(i)$ 
10:       $N_{ocor}(c) \leftarrow N_{ocor}(c) + 1$ 
11:     senão
12:        $Diff(c) \leftarrow Diff(c) + \mathbf{r}(i)$ 
13:     fim se
14:   fim para
15: fim para
16: para  $c = 1$  até  $|\mathcal{F}|$  faça
17:   se  $N_{ocor}(c) = 0$ 
18:      $\mathbf{m}(c) \leftarrow 0$ 
19:   senão se  $N_{ocor}(c) < N_E$ 
20:      $\mathbf{m}(c) \leftarrow \frac{Soma(c)}{N_{ocor}(c)} - \frac{Diff(c)}{N_E - N_{ocor}(c)}$ 
21:   fim se
22: fim para
23: para  $c = 1$  até  $|\mathcal{F}|$  faça
24:   se  $N_{ocor}(c) = N_E$ 
25:      $\mathbf{m}(c) \leftarrow \max_{c \in \{1, \dots, |\mathcal{F}|\}} \mathbf{m}(c)$ 
26:   fim se
27:    $\mathbf{e}(c) \leftarrow \mathbf{e}(c) + \mathbf{m}(c)$ 
28: fim para
29: retorne o vetor atualizado com avaliações  $\mathbf{e}$ 

```

 Fonte: Cleiton Alves da Silva, 2017

Um ponto sensível do algoritmo REPO é a relação entre os parâmetros N_F e N_E e o tamanho do conjunto $\mathcal{F}$. Durante a primeira rodada, quando a seleção é feita segundo a distribuição uniforme, a ocorrência média de qualquer abstração é dada por:

$$N_E \frac{N_F}{|\mathcal{F}|}.$$

Se a quantidade de abstrações por subconjunto N_F é pequena em relação ao tamanho de $\mathcal{F}$, deve-se ter uma alta quantidade de avaliações N_E para garantir que todas abstrações sejam bem avaliadas. Nos experimentos apresentados no capítulo 6, N_E foi escolhido de modo a garantir que, pelo menos na primeira rodada, a ocorrência média de qualquer abstração seja 5.

5.3 Tamanho dos subconjuntos de abstrações

O algoritmo **REPO** considera como entrada uma quantidade de abstrações a serem selecionadas N_F e todas as avaliações são feitas com subconjuntos de tamanho N_F . No entanto, faz parte da solução para o problema de seleção de abstrações escolher o tamanho do subconjunto de abstrações. Aqui duas soluções são consideradas para solucionar esse problema: (i) baseada em pontuação e busca, e (ii) baseada em *backward selection*.

Embora o algoritmo **REPO** tenha o objetivo de encontrar o melhor subconjunto com N_F abstrações, durante esse procedimento uma pontuação $\mathbf{e}$ é calculada para toda abstração em $\mathcal{F}$. Essa pontuação pode ser utilizada para selecionar um subconjunto de abstrações com tamanho N arbitrário, basta escolher as N melhores abstrações segundo a pontuação $\mathbf{e}$. A solução baseada em pontuação e busca considera tamanhos de busca $n_1, n_2, \dots, n_k$; para cada tamanho de busca n_i , cria um subconjunto de abstrações $\mathbf{S}(i)$ com as n_i melhores abstrações segundo a pontuação $\mathbf{e}$ e avalia $\mathbf{S}(i)$ por N_E avaliações. Essa proposta é implementada no algoritmo 6.

Algoritmo 6 Algoritmo para seleção de abstração baseado em pontuação e busca.

- 1: **Entrada:** Conjunto de abstrações $\mathcal{F}$, vetor de pontuação $\mathbf{e}$, número de avaliações por subconjunto de abstrações N , tamanhos de busca $n_1, n_2, \dots, n_k$, período de aprendizagem T
- 2: **Saída:** Subconjunto de abstrações selecionadas
- 3: **para** $i = 1$ até k **faça**
- 4: Selecione o subconjunto $\mathbf{S}(i) \subset \mathcal{F}$ com as n_i melhores abstrações segundo a pontuação $\mathbf{e}$
- 5: Use N vezes um agente de Aprendizado por Reforço que desenvolva o aprendizado com as abstrações do subconjunto $\mathbf{S}(i)$ por T período
- 6: Armazene a recompensa acumulada nas N execuções em $\mathbf{r}(i)$
- 7: **fim para**
- 8: **retorne** o subconjunto de abstrações $\mathbf{S}(i)$ com o maior valor de $\mathbf{r}(i)$

Fonte: Cleiton Alves da Silva, 2017

O segundo método considera a abordagem de *Backward selection*, na qual abstrações são eliminadas gradualmente até se obter a quantidade de abstrações desejada ou enquanto houver melhoria no subconjunto de abstrações. O algoritmo *Backward selection* tem complexidade quadrática na quantidade de abstrações em $\mathcal{F}$. Por outro lado, se a quantidade de abstrações selecionadas no algoritmo **REPO** for alta, como por exemplo, próxima do tamanho de $\mathcal{F}$, poucas avaliações são necessárias para garantir que todas as abstrações sejam experimentadas e avaliadas. A proposta baseada em *Backward selection*

considera uma agenda $n_1, n_2, \dots, n_k$ com $n_i > n_j$ se $i < j$; então primeiro n_1 abstrações são selecionadas, utilizando o algoritmo REPO e em qualquer iteração n_i abstrações são selecionadas, considerando as n_{i-1} abstrações da rodada anterior. O algoritmo 7 descreve essa proposta.

Algoritmo 7 Algoritmo para seleção de abstrações baseado em *Backward selection*.

- 1: **Entrada:** Conjunto de abstrações $\mathcal{F}$, número de avaliações por rodada N_E , número de rodadas N_R por iteração, período de aprendizagem T , agenda $n_1, n_2, \dots, n_k$, número de avaliações por subconjunto de abstrações N
- 2: **Saída:** Subconjunto de abstrações selecionadas
- 3: $\mathbf{S}(0) \leftarrow \mathcal{F}$
- 4: **para** $i = 1$ até k **faça**
- 5: $\mathbf{S}(i) \leftarrow \text{REPO}(\mathbf{S}(i-1), n_i, N_E, N_R, T)$
- 6: Use N vezes um agente de Aprendizado por Reforço que desenvolva o aprendizado com as abstrações do subconjunto $\mathbf{S}(i)$ por T período
- 7: Armazene a recompensa acumulada nas N execuções em $\mathbf{r}(i)$
- 8: **fim para**
- 9: **retorne** o subconjunto de abstrações $\mathbf{S}(i)$ com o maior valor de $\mathbf{r}(i)$

Fonte: Cleiton Alves da Silva, 2017

6 Experimentos

Inicialmente, neste capítulo são apresentadas as definições das configurações do simulador e do algoritmo Sarsa. Na sequência, a partir da seção 6.2.1 são apresentados resultados alcançados por meio da validação da proposta, que busca avaliar se o método para seleção de abstração conseguiu encontrar um subconjunto com abstrações adequadas para promover uma melhora no desempenho do agente em relação ao aprendizado. Na seção 6.2.4, são apresentados os resultados referentes a transferência de conhecimento, na qual subconjuntos compostos com melhores abstrações são avaliados em dois novos ambientes.

6.1 Resultados - Avaliação do algoritmo *REPO*

Para investigar a qualidade do ranqueamento de abstrações produzido pelo algoritmo *REPO*, definiu-se um conjunto de configurações nos experimentos. Primeiro, avalia-se uma das hipóteses de que a seleção de abstrações com base no ranqueamento é melhor do que selecionar o melhor subconjunto de abstrações. Em segundo lugar, avalia-se qual das classes de distribuição produz melhores resultados. Em terceiro lugar, avaliamos a qualidade do ranqueamento de abstrações para construir subconjuntos de abstrações de qualquer tamanho. Finalmente, testa-se a hipótese de que as abstrações selecionadas podem ser transferidas.

6.2 Configurações do simulador de futebol

Os experimentos foram realizados em simulador de futebol D2DSS, que permite representar problemas com diferentes dificuldades. A configuração escolhida define dois agentes disputando uma partida de futebol, na qual um dos agentes implementa o algoritmo Sarsa com aproximador de função (agente comportamental) e o outro agente de Planejamento. O algoritmo Sarsa em todas as experiências relatadas nas seções 6.3.1 e 6.3.2, utiliza os seguintes parâmetros: $\alpha = 0.1$, $\gamma = 0.99$ e $\epsilon = 0.1$ que é considerado constante ao longo de todo aprendizado, enquanto o Agente de Planejamento atua de forma aleatória, ou seja, com parâmetro $\epsilon = 0.0$.

Os outros parâmetros definidos no algoritmo **REPO** estão relacionados ao número de abstrações selecionadas, na qual foi alterado entre (i) número de abstrações selecionadas $N_F = 16$ e número de avaliações por rodada $N_E = 20$, (ii) número de abstrações selecionadas $N_F = 32$ e número de avaliações por rodada $N_E = 10$. Observa-se que o número de abstrações N_E foi definido para que cada abstração tenha uma frequência esperada de 5 aparições sob uma distribuição uniforme. Os demais parâmetros foram definido de forma igual para os experimentos, ou seja, número de rodadas $N_R = 6$, período de aprendizagem $T = 10^6$ para tamanho do campo igual a 11×7 . O subconjunto F contém 64 abstrações (8 abstrações canônicas, 28 abstrações Delta e 28 abstrações Joint).

6.2.1 Comparação entre melhores abstrações e melhor subconjunto

O algoritmo **REPO** considera a estratégia de encontrar um ranqueamento para as abstrações utilizadas no processo de aprendizagem, o que possibilita obter uma média de informações dessas abstrações de acordo com uma quantidade de avaliações realizadas. Em relação as avaliações, uma estratégia considera escolher subconjuntos selecionando as melhores abstrações, o que pode resultar em um subconjunto que nunca foi avaliado. Outra estratégia consiste em encontrar um ranqueamento para os subconjuntos de abstrações simplesmente escolhendo o subconjunto melhor avaliado. No entanto, essa estratégia pode não funcionar bem, porque alguns subconjuntos são observados e avaliados apenas uma vez.

Nos experimentos, o algoritmo **REPO** com distribuição uniforme e comparado aos resultados de ambas as estratégias em cada rodada. Em cada rodada, o vetor de pontuação é usado para construir um subconjunto com as melhores N_F abstrações (Melhores abstrações). Além disso, o histórico de subconjuntos avaliados é usado para selecionar o subconjunto com a melhor avaliação (Melhores subconjuntos). Então, ambos os subconjuntos (as melhores abstrações e o melhor subconjunto) são avaliados pela média da recompensa acumulada em 5 processos de aprendizagem AR completos por um período de $T = 10^6$ etapas.

A Figura 6 e 7 exibem as curvas de aprendizagem comparando ambas as estratégias, ao utilizar $N_F = 16$ com $N_E = 20$ e $N_F = 32$ com $N_E = 10$ respectivamente. As curvas de aprendizagem são calculadas a partir da média de 30 execuções do algoritmo **REPO**.

No lado esquerdo, além de mostrar a avaliação de cada estratégia (melhores abstrações e melhor subconjunto), também exibem os subconjuntos avaliados ao executar o algoritmo REPO para ter uma linha de base para a avaliação esperada dos subconjuntos a partir de uma distribuição uniforme.

A Figura do lado direito, exibe o resultado de um teste de hipótese, na qual compara se os resultados das melhores abstrações são iguais ao resultados do melhor subconjunto e iguais aos subconjuntos avaliados. A estratégia de usar o ranqueamento de abstrações provou ser melhor em ambos os cenários, apresentando baixos valores- p ao usar $N_F = 16$. Pressupõe-se que ao ranquear conjuntos de abstrações de cardinalidade uniformemente pequenas, é mais difícil encontrar um bom subconjunto de abstrações. Por outro lado, o vetor de pontuações não sofre de um problema assim, uma vez que todas as abstrações foram avaliadas de alguma forma.

Figura 6 – Comparação entre as melhores abstrações e o melhor subconjunto. Figura a esquerda exibe a média de recompensas acumuladas em cada rodada, utilizando $N_E = 16$. Figura a esquerda, exibe resultado do t-teste.

Fonte: Cleiton Alves da Silva, 2017

6.2.2 Abordagens de distribuição de probabilidade

Os experimentos anteriores foram repetidos, para comparar as três abordagens relacionadas a distribuição de probabilidade descritas na seção 5.2: *Probabilistic Selection*, *Hard Selection* e *Uniform Selection*. Para a abordagem *Probabilistic Selection*, o parâmetro de modulação foi agendado como (0.25, 0.40, 0.55, 0.69, 0.84, 0.99) quando $N_E = 16$ e (0.50, 0.60, 0.70, 0.79, 0.89, 0.99) quando $N_E = 32$. Para a abordagem *Hard Selection*, o

Figura 7 – Comparação entre as melhores abstrações e o melhor subconjunto. Figura a esquerda exibe a média de recompensas acumuladas em cada rodada, utilizando $N_E = 32$. Figura a esquerda, exibe resultado do t-teste.

Fonte: Cleiton Alves da Silva, 2017

parâmetro de modulação foi agendado como (0, 3, 5, 8, 11, 13) quando $N_E = 16$ e (0, 5, 11, 16, 21, 27) quando $N_E = 32$. Para a abordagem *Uniform Selection* todas tem a mesma probabilidade de aparecerem nos subconjuntos.

As Figuras 8 e 9, respectivamente, exibem a comparação das três abordagens propostas para distribuição de probabilidade ao usar $N_F = 16$ ($N_E = 20$) e $N_F = 32$ ($N_E = 10$). Os resultados foram calculados a partir da média de 20 execuções do algoritmo REPO. No lado esquerdo, além de mostrar a avaliação de cada abordagem de distribuição, exibe-se também os subconjuntos avaliados (linhas tracejadas) ao executar o algoritmo REPO em cada abordagem de distribuição de probabilidade. Além de exibir a avaliação das melhores abstrações em cada rodada (linhas sólidas).

Observa-se que na primeira rodada utilizou-se distribuição uniforme, ou seja, para as três abordagens de distribuições de probabilidade foi a mesma. Na Figura a direita, exibe-se o resultado de teste t pareado para as amostras, na qual a hipótese de que a distribuição *Hard Selection* é igual à distribuição *Probabilistic Selection* e *Uniform Selection*. O uso da distribuição *Hard Selection* mostrou-se melhor em ambos os cenários, novamente, apresentando menores valores- p ao usar $N_F = 16$.

Observa-se que a abordagem *Hard Selection*, no final do processo obteve o melhor comportamento. Essa abordagem foi escolhida para desenvolvimento dos próximos experimentos.

Figura 8 – Comparação entre três diferentes abordagens de distribuição. Na Figura a esquerda: $N_E = 16$, exibe-se as curvas de aprendizagem com a médias de recompensa acumulada em cada rodada, as linhas tracejadas mostram a avaliação dos subconjuntos avaliados em cada rodada, enquanto linhas sólidas exibem a avaliação das melhores abstrações em cada rodada. A Figura a direita, exibe o resultado do teste- t parelhado das amostras em cada rodada.

Fonte: Cleiton Alves da Silva, 2017

Figura 9 – Comparação entre três diferentes abordagens de distribuição. Na Figura a esquerda: $N_E = 32$, exibe-se as curvas de aprendizagem com a médias de recompensa acumulada em cada rodada, as linhas tracejadas mostram a avaliação dos subconjuntos avaliados em cada rodada, enquanto linhas sólidas exibem a avaliação das melhores abstrações em cada rodada. A Figura a direita, exibe o resultado do teste- t parelhado das amostras em cada rodada.

Fonte: Cleiton Alves da Silva, 2017

6.2.3 Dependência do tamanho fixo dos subconjuntos

O algoritmo REPO considera um tamanho fixo para subconjuntos de abstrações em cada rodada. No entanto, uma vez que retorna um ranqueamento de abstrações, subconjuntos de abstrações de qualquer tamanho podem ser construídos a partir desse

ranqueamento. Embora seja bem conhecido na literatura que tal estratégia não é otimizada, muitas abordagens utilizam essa estratégia.

A abordagem mais popular para obter um subconjunto de abstrações é uma construção incremental, que pode ser feita como *forward* ou *backward* (GUYON; ELISSEEFF, 2003). A versão *forward* considera que em cada iteração um subconjunto de abstrações é selecionado $\mathcal{G}$ e seu complemento $\overline{\mathcal{G}}$, então, os subconjuntos $\mathcal{G} \cup \{F\}$ para todos os $F \in \overline{\mathcal{G}}$ são avaliados e o subconjunto melhor avaliado se torna o novo subconjunto das abstrações selecionadas $\mathcal{G}$. A versão *backward* é feita na direção oposta, ou seja, em cada iteração, uma abstração F é eliminada do subconjunto das abstrações selecionadas $\mathcal{G}$.

A fim de encontrar o tamanho adequado de abstrações por subconjunto, o algoritmo REPO foi comparado com a técnica de construção incremental. Porém, uma vez que o processo de avaliação é custoso, ou seja, há muita necessidade de tempo para processamento, ao invés de testar a inclusão ou remoção de uma única abstração, propõe-se testar a inclusão ou remoção de subconjuntos de abstrações.

Nos experimentos relatados nessa seção, define-se partições aleatórias com a quantidade de 4 abstrações para subconjunto $\overline{\mathcal{G}}$ para construção incremental *forward* ou para subconjunto $\mathcal{G}$ para construção incremental *backward*.

No cenário utilizando as 64 abstrações e partições com subconjuntos de 4 abstrações, a construção incremental *forward* ou *backward* precisa de 135 avaliações ($15 \frac{16+2}{2}$). Nessas configurações, o algoritmo REPO precisa de 60 avaliações quando $N_F = 32$ e 120 avaliações quando $N_F = 16$.

A Figura 10 exibe a comparação entre versões do algoritmo REPO e versões para construção incremental. Na Figura à esquerda, a avaliação do ranqueamento de cada algoritmo foi feita para diferentes tamanhos para subconjuntos de abstrações (4, 8, ..., 60, 64). Novamente, os subconjuntos são avaliados pela média da recompensa acumulada ao longo de 5 processos de aprendizagem no AR completos por um período de $T = 2.5 \times 10^5$. O algoritmo REPO utiliza a distribuição de probabilidade da abordagem *Hard Selection*, sob a configuração descrita anteriormente. Na Figura a direita, há o resultado de um teste- t de amostra pareada na hipótese em que o algoritmo REPO é melhor do que a construção incremental *forward* e *backward*. O algoritmo REPO com $N_F = 16$ apresentou o melhor desempenho, mostrando a dependência do parâmetro N_F no algoritmo REPO. No entanto, mesmo quando $N_F = 32$ o algoritmo REPO apresenta um melhor desempenho em torno do

mesmo tamanho do subconjuntos, ou seja, utilizando subconjuntos com tamanho de 32 abstrações.

Figura 10 – Comparação entre o algoritmo REPO e construção incremental. Figura esquerda exibe a recompensa média acumulada para diferentes tamanhos de subconjuntos. Figura direita exibe um teste- t de amostra pareada para cada tamanho do subconjunto.

Fonte: Cleiton Alves da Silva, 2017

A Figura 11, exibe a comparação entre versões do algoritmo REPO e versão *Log Hard Selection*. Observa-se que a curva representada pela versão do algoritmo REPO que utiliza 666 abstrações do tipo *Joint*, encontra o melhor subconjunto de abstrações para cada novo tamanho de subconjunto. Também utilizando uma construção incremental, em cada uma das repetições o tamanho do subconjunto avaliado diminui de acordo com o parâmetro *ratio*, definido como raiz de 2. Observa-se que ao final do processo, existe um subconjunto com 4 abstrações que é melhor do que a estratégia da versão *Log Hard Selection* com 64 abstrações e REPO com 16 abstrações.

Figura 11 – Comparação entre o algoritmo REPO e abordagem *Log Hard Selection*. A Figura exibe a recompensa média acumulada para diferentes tamanhos de subconjuntos e versões do algoritmo REPO.

Fonte: Cleiton Alves da Silva, 2017

6.2.4 Transferência de conhecimento

Para testar a hipótese de que as abstrações podem ser transferidas entre problemas semelhantes que utilizam as mesmas variáveis, avalia-se o ranqueamento obtido no algoritmo REPO com $N_F = 16$. Considera-se 4 variações nos experimentos, na qual altera-se o tamanho do campo do simulador, o conjunto de ações do agente AR e o parâmetro ε do Agente de Planejamento.

A Figura 22 exibe a avaliação de diferentes tamanhos de subconjuntos de abstrações (8, 16, ..., 56, 64). Novamente, os subconjuntos são avaliados pela média da recompensa acumulada em 5 processos completos de aprendizagem AR por um período de $T = 2.5 \times 10^5$, na qual as ações que o agente comportamental desempenha, utiliza heurísticas e tamanho do campo 11×7 . Em outro experimento, considera-se o período de $T = 10 \times 10^5$ e um agente que utiliza ações atômicas e tamanho do campo 23×15 . Considera-se o uso de 64 abstrações como base e percebe-se que a maioria dos tamanhos de subconjuntos apresentaram um melhor desempenho, em especial o subconjunto com tamanho 16, ou seja, $N_F = 16$, que mostra o melhor desempenho, como também aconteceu nos experimentos apresentados na Figura 10.

Figura 12 – Avaliação de subconjuntos de abstrações com tamanhos diferentes e transferência de abstrações ranqueadas para problemas destino. Os experimentos variam o tamanho do campo do simulador, o conjunto de ações do agente de AR e o parâmetro ε do Agente de Planejamento.

Fonte: Cleiton Alves da Silva, 2017

6.3 Novos experimentos com uso do agente atômico

Para obter novos resultados, um novo conjunto de experimentos foi realizado, alterando a configuração relacionada ao agente aprendiz para que realiza-se ações atômicas. Além disso, o algoritmo Sarsa em todas as experiências relatadas nas seções 6.3.1 e 6.3.2, novamente, utiliza os seguintes parâmetros: $\alpha = 0.1$, $\gamma = 0.99$ e $\epsilon = 0.1$ que é considerado constante ao longo de todo aprendizado, enquanto o Agente de Planejamento atua com o parâmetro ε de maneira diferente em cada conjunto de experimentos, ou seja, $\varepsilon = 1.0$, $\varepsilon = 0.5$ e $\varepsilon = 0.0$.

Os outros parâmetros definidos no algoritmo REPO estão relacionados ao número de abstrações selecionadas, na qual foi alterado para o número de abstrações selecionadas $N_F = 32$ e número de avaliações por rodada $N_E = 10$. Observa-se que o número de abstrações N_E foi definido para que cada abstração tenha uma frequência esperada de 5

aparições sob uma distribuição uniforme. Os demais parâmetros foram definidos de forma igual para os experimentos, ou seja, número de rodadas $N_R = 6$, período de aprendizagem $T = 10^6$ para tamanho do campo igual a 11×7 . O subconjunto F contém 64 abstrações (8 abstrações canônicas, 28 abstrações Delta e 28 abstrações Joint).

6.3.1 Abordagens de distribuição de probabilidade

Esta subseção apresenta os resultados para as três abordagens propostas: *Probabilistic Selection*, *Hard Selection* e *Uniform Selection*. Em cada rodada na abordagem *Probabilistic Selection* o agendamento é definido para (0.50, 0.59, 0.69, 0.79, 0.89 e 0.99), a abordagem *Hard Selection* usa agendamento (0, 5, 11, 16, 21, 27) e a abordagem *Uniform Selection* escolhe abstrações da mesma forma, por meio de uma distribuição uniforme, em cada rodada. Os resultados representam a média de 100 curvas de aprendizagem (10 repetições vezes os $N_E = 10$ subconjuntos) para cada rodada.

As Figuras 13, 14 e 15 exibem os gráficos relacionados ao desempenho do agente ao acumular recompensas em cada uma das rodadas. Em cada um dos gráficos, existem duas curvas referentes a cada abordagem, uma representa os resultados obtidos com os subconjuntos que foram selecionados e avaliados durante a execução do algoritmo REPO e outra os resultados dos subconjuntos com as melhores abstrações selecionadas. Em todas as execuções o agente atômico disputou partidas contra o agente que implementou o algoritmo de planejamento. Observa-se que nas Figuras 13, 14 e 15 que o agente de planejamento usa parâmetros $\varepsilon = 1.0$, $\varepsilon = 0.5$ e $\varepsilon = 0.0$, respectivamente.

Os resultados obtidos na Figura 13 mostram que a abordagem *Hard Selection* ao final da sexta rodada apresentou o melhor desempenho comparado as outras abordagens. Apesar da abordagem *Probabilistic Selection* apresentar melhor na quarta rodada, não conseguiu ao final obter o melhor rendimento. A abordagem *Uniform Selection* apresentou resultados superiores comparado ao uso das 64 abstrações a partir da quarta rodada. As abordagens *Probabilistic Selection* e *Hard Selection*, utilizando subconjuntos avaliados, melhoram muito rápido comparadas a *Uniform Selection*, mostrando que o agendamento apropriado pode gerar melhores resultados.

Figura 13 – Comparação entre três diferentes abordagens de distribuição com $N_E = 32$ e parâmetro $\varepsilon = 1.0$. A Figura exibe as curvas de aprendizagem com as médias de recompensa acumulada em cada rodada, as linhas tracejadas mostram a avaliação dos subconjuntos avaliados em cada rodada, enquanto linhas sólidas exibem a avaliação das melhores abstrações em cada rodada.

Fonte: Cleiton Alves da Silva, 2017

A abordagem *Probabilistic Selection* observada na Figura 14 melhorou até a terceira rodada, mas quando comparado seu comportamento na quarta rodada com *Hard Selection* e *Uniform Selection* o desempenho não foi o mesmo. Percebe-se que a curva referente a abordagem *Hard Selection* com as melhores abstrações, apresentou pouco crescimento no decorrer das rodadas quando comparada a *Probabilistic Selection* e *Uniform Selection*, mostrando que as abstrações selecionadas nas rodadas iniciais foram suficientes para desenvolver o aprendizado.

Figura 14 – Comparação entre três diferentes abordagens de distribuição com $N_E = 32$ e parâmetro $\varepsilon = 0.5$. A Figura exhibe as curvas de aprendizagem com as médias de recompensa acumulada em cada rodada, as linhas tracejadas mostram a avaliação dos subconjuntos avaliados em cada rodada, enquanto linhas sólidas exibem a avaliação das melhores abstrações em cada rodada.

Fonte: Cleiton Alves da Silva, 2017

Observa-se na Figura 15 que o resultado obtido na primeira rodada com o uso das melhores abstrações, manteve comportamento semelhante até chegar a ultima rodada. Mostrando que o processo de seleção de abstração conseguiu encontrar um subconjunto com as melhores abstrações na primeira rodada e que se manteve no decorrer das rodadas seguintes. Ainda, a partir da terceira rodada na abordagem *Hard Selection* com os conjuntos avaliados, tem comportamento semelhantes aos resultados dos subconjuntos com melhores abstrações, mostrando que o agendamento conseguiu encontrar melhores abstrações na metade das rodadas. Na sexta rodada a abordagem *Probabilistic Selection* com os conjuntos avaliados, conseguiu mesmo desempenho.

Figura 15 – Comparação entre três diferentes abordagens de distribuição com $N_E = 32$ e parâmetro $\varepsilon = 0$. A Figura exibe as curvas de aprendizagem com as médias de recompensa acumulada em cada rodada, as linhas tracejadas mostram a avaliação dos subconjuntos avaliados em cada rodada, enquanto linhas sólidas exibem a avaliação das melhores abstrações em cada rodada.

Fonte: Cleiton Alves da Silva, 2017

De modo geral, ao final das seis rodadas, todas as abordagens que utilizam as melhores abstrações, atingiram um resultado melhor comparado ao agente que utilizou as 64 abstrações.

6.3.2 Transferência de conhecimento

Com o conhecimento adquirido por intermédio do algoritmo REPO, nestes experimentos, avalia-se por meio da transferência de conhecimento o desempenho do agente atômico que utiliza subconjuntos com melhores abstrações obtidas no problema fonte.

As Figuras 16, 17 e 18 exibem curvas de aprendizagem relacionadas ao desempenho do agente ao acumular recompensas em cada uma das rodadas. As curvas exibidas em cada gráfico, estão relacionadas as abordagens *Probabilistic Selection*, *Hard Selection* e *Uniform Selection* e representam os resultados obtidos por um agente atômico que utiliza os subconjuntos com as melhores abstrações selecionadas no problema origem. Em todas as execuções o agente atômico disputou partidas contra o Agente de Planejamento. Observa-se nas Figuras 16, 17 e 18 que o agente de planejamento usa parâmetros $\varepsilon = 1.0$, $\varepsilon = 0.5$ e

$\varepsilon = 0.0$, respectivamente. O campo considerado tem dimensão 33×21 e a quantidade de abstrações utilizadas pelo agente atômico é de $N_F = 32$

Observa-se que o resultado apresentado na Figura ?? a abordagem *Uniform Selection* obteve comportamento mais coerente, sendo ascendente durante as rodadas e na ultima rodada, apresentou resultado superior ao agente que utiliza as 64 abstrações. A abordagem *Probabilistic Selection* em todas rodadas, mostrou comportamento inferior quando comparada ao agente com 64 abstrações. A abordagem *Hard Selection* obteve um bom desempenho na primeira rodada, mas sem coerência nas demais. Na rodada 4 obteve o pior desempenho do que qualquer outra abordagem, porém seu comportamento melhorou muito a partir da quinta rodada e manteve-se superior ao agente com 64 abstrações.

Figura 16 – Avaliação de subconjuntos de abstrações transferidos do problema origem, comparando as três diferentes abordagens de distribuição com $N_E = 32$ e parâmetro $\varepsilon = 1$ e campo do simulador em 33×21 . A Figura exhibe as curvas de aprendizagem com as médias de recompensa acumulada em cada rodada, as linhas exibem a avaliação das melhores abstrações em cada rodada.

Fonte: Cleiton Alves da Silva, 2017

Na Figura 16 a abordagem *Uniform Selection* manteve-se com rendimento inferior em três rodadas diferentes, comparada ao agente com 64 abstrações. A abordagem *Hard Selection* apresentou comportamento ascendente desde o início, acumulando recompensas em cada rodada. A abordagem *Probabilistic Selection* obteve um comportamento superior as outras abordagens na rodada dois, mas o rendimento diminui em seguida, mas melhorado

na ultima rodada. Observa-se que no final de todas as rodadas, as abordagens conseguiram superar o resultado do agente que utiliza as 64 abstrações.

Figura 17 – Avaliação de subconjuntos de abstrações transferidos do problema origem, comparando as três diferentes abordagens de distribuição com $N_E = 32$ e parâmetro $\varepsilon = 0.5$ e campo do simulador em 33×21 . A Figura exhibe as curvas de aprendizagem com as médias de recompensa acumulada em cada rodada, as linhas exibem a avaliação das melhores abstrações em cada rodada.

Fonte: Cleiton Alves da Silva, 2017

Observa-se na Figura 18 que a abordagem *Uniform Selection* conseguiu nas rodadas um e três acumular mais recompensa comparada as outras abordagens, porém não manteve o mesmo comportamento e na ultima rodada, terminou com desempenho pior. As abordagens *Probabilistic Selection* e *Hard Selection* tiveram comportamentos semelhantes e nas rodadas finais conseguiram melhorar bem seu desempenho. Comparando o resultado das abordagens na ultima rodada, nota-se que a *Hard Selection* foi a melhor. Ainda, as abordagens propostas conseguiram obter rendimento superior em todas as rodadas, superando o resultado do agente que utiliza as 64 abstrações.

Figura 18 – Avaliação de subconjuntos de abstrações transferidos do problema origem, comparando as três diferentes abordagens de distribuição com $N_E = 32$ e parâmetro $\varepsilon = 0$ e campo do simulador em 33×21 . A Figura exibe as curvas de aprendizagem com as médias de recompensa acumulada em cada rodada, as linhas exibem a avaliação das melhores abstrações em cada rodada.

Fonte: Cleiton Alves da Silva, 2017

Da mesma forma que no experimento anterior, o conhecimento adquirido no problema origem é transferido e utilizado nesse novo ambiente, que conta com dimensão de campo igual a 23×15 , a quantidade de abstrações utilizadas pelo agente atômico é de $N_F = 32$.

As Figuras 19, 20 e 21 exibem curvas de aprendizagem relacionadas ao desempenho do agente ao acumular recompensas em cada uma das rodadas. As curvas exibidas em cada gráfico, estão relacionadas as abordagens *Probabilistic Selection*, *Hard Selection* e *Uniform Selection* e representam os resultados obtidos por um agente atômico utilizando os subconjuntos com as melhores abstrações selecionadas no problema origem.

A Figura 19 mostra que a abordagem *Hard Selection* obteve um rápido crescimento na segunda e terceira rodada, mas não foi suficiente. No final da rodada seis, obteve o pior rendimento e não superou o desempenho do agente que utiliza as 64 abstrações. A abordagem *Probabilistic Selection* apresentou crescimento coerente até a quarta rodada, posteriormente o desempenho diminuiu, mas no final da sexta rodada conseguiu superar o agente com 64 abstrações. A *Uniform Selection* apresentou um bom desempenho nas rodadas iniciais, porém na rodada quatro apresentou-se pior do que qualquer outro método.

O melhor resultado foi obtido na rodada cinco, mantendo-se com o melhor desempenho comparada as outras abordagens e, novamente, superando o agente com 64 abstrações.

Figura 19 – Avaliação de subconjuntos de abstrações transferidos do problema origem, comparando as três diferentes abordagens de distribuição com $N_E = 32$ e parâmetro $\varepsilon = 1$ e campo do simulador em 23×15 . A Figura exhibe as curvas de aprendizagem com as médias de recompensa acumulada em cada rodada, as linhas exibem a avaliação das melhores abstrações em cada rodada.

Fonte: Cleiton Alves da Silva, 2017

A abordagem *Uniform Selection* observada na Figura 20 apresenta um comportamento não coerente no decorrer das rodadas, porém obtêm um bom rendimento, especialmente, na ultima rodada. A abordagem *Probabilistic Selection* apresenta bons resultados a partida da segunda rodada. A abordagem *Hard Selection* é a mais adequada, mostrando um comportamento ascendente durante todas as rodadas. No final obtêm o melhor resultado comparado as outras abordagens. Ainda, todas as abordagens quando comparadas ao uso das 64 abstrações, obtiveram comportamento superior.

Figura 20 – Avaliação de subconjuntos de abstrações transferidos do problema origem, comparando as três diferentes abordagens de distribuição com $N_E = 32$ e parâmetro $\varepsilon = 0.5$ e campo do simulador em 23×15 . A Figura exibe as curvas de aprendizagem com as médias de recompensa acumulada em cada rodada, as linhas exibem a avaliação das melhores abstrações em cada rodada.

Fonte: Cleiton Alves da Silva, 2017

Inicialmente, observa-se na Figura 21 que todas as abordagens apresentaram melhores comportamentos quando comparadas ao agente que utiliza as 64 abstrações. As abordagens *Probabilistic Selection* e *Hard Selection* apresentaram-se de maneira incoerente, acumulando a maior quantidade de recompensas a partir da primeira rodada, porém não melhoraram o rendimento. A abordagem *Uniform Selection* apresentou melhores resultados na primeira e quinta rodada. Nota-se que na primeira rodada, a seleção de abstrações conseguiu encontrar um subconjunto de abstrações que conseguiu obter melhores resultados.

Figura 21 – Avaliação de subconjuntos de abstrações transferidos do problema origem, comparando as três diferentes abordagens de distribuição com $N_E = 32$ e parâmetro $\varepsilon = 0$ e campo do simulador em 23×15 . A Figura exhibe as curvas de aprendizagem com as médias de recompensa acumulada em cada rodada, as linhas exibem a avaliação das melhores abstrações em cada rodada.

Fonte: Cleiton Alves da Silva, 2017

Observa-se na Figuras 16 e 19 que apesar do resultado negativo para o ambiente com transferência, os agentes com 32 abstrações não foram consistentemente melhores do que o agente com 64 abstrações. A diferença final foi pouca, porém metade dos recursos computacionais, como por exemplo memória, foram necessários para realizar o processo de aprendizado com um agente utilizando a metade das abstrações, i.e., subconjuntos com 32 abstrações ao invés das 64 abstrações.

6.3.3 Transferência de conhecimento: agente comportamental e atômico

O resultado apresentado na Figura 22 é referente a dois processo de transferências, o primeiro está relacionado a um ambiente 11 e utiliza um agente atômico, curva verde, e o outro a um ambiente 23×15 , utilizando um agente comportamental, curva azul.

Figura 22 – Transferência de abstrações entre ambientes diferentes.

Fonte: Cleiton Alves da Silva, 2017

Apos utilizar a abordagem *Hard Selection* com um agente utilizando 16 abstrações, obtêm-se o ranqueamento das 64 abstrações. As próximas etapas seguem removendo a quantidade de 4 abstrações do conjunto com o total de abstrações. Os subconjuntos com uma quantidade menor abstrações são novamente avaliados. Ao final do processo, compara-se o comportamento quando os subconjunto de abstrações foram transferidos de um ambiente origem para outro destino.

6.3.4 Análise qualitativa das abstrações

O algoritmo REPO é responsável por selecionar e avaliar subconjuntos de abstrações. Nos experimentos com uso da abordagem *Hard Selection* e um agente comportamental, utilizando subconjuntos com 16 abstrações. Em suas 10 repetições, observa-se que algumas abstrações aparecem com maior frequência nos subconjuntos selecionados, mas as seguintes abstrações aparecem em todas as 10 execuções: $F_{5-1,0}^C$, $F_{5,1}^D$, $F_{6,2}^D$ e $F_{7,1}^D$.

A abstração $F_{5-1,0}^C$ informa a velocidade da bola e como na maior parte do jogo a bola está parada, essa abstração tem um valor constante. A abstração $F_{5,1}^D$ é a diferença entre a posição x do agente aprendiz com a posição x da bola, enquanto a abstração $M_{6,2}^D$ é a diferença entre a posição y do agente aprendiz com a posição y da bola. Essas abstrações permitem ao agente aprendiz tomar decisões que o fazem se aproximar da bola, um comportamento fundamental para realizar gols. A abstração $F_{7,1}^D$ é a diferença entre a

posição x do agente de planejamento com a posição x da bola, essa abstração permite ao agente aprender a saber se o agente de planejamento está próximo a bola.

Esse experimento tem característica semelhantes ao anterior, porém utiliza 32 abstrações. Novamente as abstrações $F_{5-1,0}^C$, $F_{5,1}^D$, $F_{6,2}^D$ e $F_{7,1}^D$, apareceram e continuam com a mesma importância. Outras abstrações como $F_{8,2}^D$, representando a diferença entre a y do agente de planejamento com a posição y da bola, essa abstração permite ao agente aprender a saber se o agente de planejamento está próximo a bola. A combinação entre a abstração $F_{8,2}^D$ e a $F_{7,1}^D$ é interessante, visto que, indica ao agente aprender exatamente onde o adversário está em campo com a bola. A e $F_{3,1}^D$, é a diferença entre a direção da bola com a posição x da bola, como na maior parte do jogo a bola está parada, o agente aprender observa a posição da bola, a fim de saber se está próxima dela.

O experimento que conta com um conjunto com 666 abstrações, em sua última rodada selecionou um subconjunto com as seguintes abstrações. $F_{5,1}^D \times F_{6,2}^D$ e $F_{5,1}^D \times F_{7,1}^D$. A abstração Joint $F_{5,1}^D \times F_{6,2}^D$ como observado anteriormente, a relação entre essas abstrações é muito interessante. Novamente, essa única abstração permite ao agente aprender tomar decisões que o fazem se aproximar da bola, ou seja, qual a proximidade do agente aprender em relação ao eixo x da bola e o eixo y da bola. A abstração $F_{5,1}^D \times F_{7,1}^D$, permite saber quem está mais próximo a bola, se é o agente aprender ou agente de planejamento.

7 Conclusão

Nesta dissertação propôs-se um método para seleção de um subconjunto de abstrações que considera a avaliação do processo de aprendizagem no AR. A solução utilizada para o problema de selecionar um subconjunto de abstrações foi implementada no algoritmo REPO, responsável por avaliar e selecionar um subconjunto com melhores abstrações. Considerou-se que o processo de aprendizagem fosse repetido inúmeras vezes, visando os cenários que envolvem transferência de conhecimento, especificamente, transferência de subconjuntos com melhores abstrações.

O algoritmo foi avaliado em um simulador de Futebol 2D sob diferentes aspectos. Embora o algoritmo REPO necessite de muitos parâmetros em suas configurações, mostra-se competitivo contra uma abordagem bem conhecida na literatura, mesmo sem ajustar esses parâmetros. Na verdade, sob várias avaliações obteve-se uma boa melhora de desempenho, infelizmente as avaliações são dispendiosas.

Embora considerando que as avaliações utilizam custo computacionais elevados, o algoritmo REPO permite obter um subconjunto de abstrações que é útil em todo o processo de aprendizagem e pode ser útil em uma análise qualitativa da influência das abstrações no processo de aprendizagem. Nesta dissertação utilizou-se recompensas acumuladas como avaliação do processo de aprendizagem, mas outras avaliações podem ser consideradas, por exemplo, o tempo até um limite ou desempenho assintótico.

O algoritmo REPO encontrou subconjuntos com uma menor quantidade de melhores abstrações e como estas abstrações representam o problema original e melhoram substancialmente o desempenho do agente, diminuiu-se os custos computacionais na transferência de conhecimento.

7.0.1 Trabalhos futuros

Aponta-se três direções para continuar este trabalho e em primeiro lugar, a hipótese de que a avaliação do processo de aprendizagem é melhor do que uma avaliação off-line é um caminho, mas não houve testes. Em segundo lugar, é construído um ranqueamento de abstrações, mas o resultado é sensível ao tamanho dos subconjuntos avaliados, ou seja, ao parâmetro N_F , então, algum método deve ser projetado para selecioná-lo automaticamente. Finalmente, embora o algoritmo REPO retorne um ranqueamento de abstrações, como

selecionar o melhor tamanho de subconjunto para o subconjunto de abstrações sob esse ranqueamento continua a ser definido, visto que os experimentos na seção 6.2.3 podem indicar um caminho.

Além dessas direções, ainda é necessário a definir se a escolha da distribuição de probabilidade por rodadas influencia no resultado e avaliar o processo de transferência de abstração para dois ou mais agentes.

7.0.2 Trabalhos publicados

SILVA, C. A.; FREIRE, V. . Seleção de Atributos no Aprendizado por Reforço avaliando o processo de aprendizagem.. In: XIII Encontro Nacional de Inteligência Artificial e Computacional, 2016, Recife, PE. Anais do XIII Encontro Nacional de Inteligência Artificial e Computacional, 2016. p. 133-143.

SILVA, C. A.; FREIRE, V. . Feature Selection for Reinforcement Learning by Learning Process Evaluation. In: 1st Workshop on Transfer in Reinforcement Learning (TiRL), 2017, São Paulo. Proceedings of the 1st Workshop on Transfer in Reinforcement Learning (TiRL) at the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), 2017., 2017. p. p. 1-8.

SILVA, C. A.; FREIRE, V. . Learning-Process Evaluation to select Spatial Abstractions in Reinforcement Learning.. In: XIV Encontro Nacional de Inteligência Artificial e Computacional (ENIAC), 2017, Uberlândia. 2017 Brazilian Conference on Intelligent Systems., 2017. p. p. 1-6.

Referências¹

- BISHOP, J.; MIIKKULAINEN, R. Evolutionary feature evaluation for online reinforcement learning. In: *Proceedings of 2013 IEEE Conference on Computational Intelligence and Games (CIG2013)*. [S.l.: s.n.], 2013. p. 267–275. Citado 2 vezes nas páginas 25 e 26.
- DIUK, C.; LI, L.; LEFFLER, B. R. The adaptive k-meteorologists problem and its application to structure learning and feature selection in reinforcement learning. In: *Proceedings of the 26th Annual International Conference on Machine Learning*. New York, NY, USA: ACM, 2009. (ICML '09), p. 249–256. ISBN 978-1-60558-516-1. Disponível em: <http://doi.acm.org/10.1145/1553374.1553406>. Citado 2 vezes nas páginas 23 e 25.
- DUDA, R.; HART, P.; STORK, D. *Pattern Classification*. [S.l.]: Wiley, 2001. Citado na página 37.
- FERNÁNDEZ, F.; VELOSO, M. Probabilistic policy reuse in a reinforcement learning agent. In: *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*. New York, NY, USA: ACM, 2006. (AAMAS '06), p. 720–727. ISBN 1-59593-303-4. Citado 2 vezes nas páginas 20 e 35.
- FROMMBERGER, L.; WOLTER, D. Structural knowledge transfer by spatial abstraction for reinforcement learning agents. *Adaptive Behavior*, v. 18, n. 6, p. 507–525, 2010. Citado na página 35.
- GAUDEL, R.; SEBAG, M. Feature selection as a one-player game. In: FURNKRANZ, J.; JOACHIMS, T. (Ed.). *ICML*. Omnipress, 2010. p. 359–366. Disponível em: <http://dblp.uni-trier.de/db/conf/icml/icml2010.html#GaudelS10>. Citado 2 vezes nas páginas 25 e 36.
- GERAMIFARD, A.; DANN, C.; HOW, J. P. Off-policy learning combined with automatic feature expansion for solving large MDPs. In: *1st Multidisciplinary Conference on Reinforcement Learning and Decision Making (RLDM)*. [S.l.: s.n.], 2013. Citado na página 24.
- GERAMIFARD, A. et al. Online discovery of feature dependencies. In: *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*. [S.l.: s.n.], 2011. p. 881–888. Citado na página 24.
- GUYON, I.; ELISSEEFF, A. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, JMLR.org, v. 3, p. 1157–1182, mar. 2003. ISSN 1532-4435. Disponível em: <http://dl.acm.org/citation.cfm?id=944919.944968>. Citado 2 vezes nas páginas 23 e 57.
- HALL, M. A. Correlation-based feature selection for discrete and numeric class machine learning. In: *Proceedings of the Seventeenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000. (ICML '00), p. 359–366. ISBN 1-55860-707-2. Disponível em: <http://dl.acm.org/citation.cfm?id=645529.657793>. Citado na página 37.
- JONG, N. K.; STONE, P. State abstraction discovery from irrelevant state variables. In: *Proceedings of the 19th International Joint Conference on Artificial Intelligence*. San

¹ De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.

Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005. (IJCAI'05), p. 752–757. Disponível em: <http://dl.acm.org/citation.cfm?id=1642293.1642414>. Citado na página 31.

KIRA, K.; RENDELL, L. A. A practical approach to feature selection. In: *Proceedings of the Ninth International Workshop on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1992. (ML92), p. 249–256. ISBN 1-5586-247-X. Disponível em: <http://dl.acm.org/citation.cfm?id=141975.142034>. Citado na página 37.

KOHAVI, R.; JOHN, G. H. Wrappers for feature subset selection. *Artif. Intell.*, Elsevier Science Publishers Ltd., Essex, UK, v. 97, n. 1-2, p. 273–324, dez. 1997. ISSN 0004-3702. Disponível em: [http://dx.doi.org/10.1016/S0004-3702\(97\)00043-X](http://dx.doi.org/10.1016/S0004-3702(97)00043-X). Citado na página 37.

KONIDARIS, G. A framework for transfer in reinforcement learning. In: *In Proceedings of the ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning*. [S.l.: s.n.], 2006. Citado na página 35.

KONIDARIS, G.; BARTO, A. Efficient skill learning using abstraction selection. In: *Proceedings of the 21st International Joint Conference on Artificial Intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009. (IJCAI'09), p. 1107–1112. Disponível em: <http://dl.acm.org/citation.cfm?id=1661445.1661622>. Citado 2 vezes nas páginas 23 e 24.

KUMAR, V.; MINZ, S. Feature selection: A literature review. *Smart CR*, v. 4, n. 3, p. 211–229, 2014. Disponível em: <http://dx.doi.org/10.6029/smartercr.2014.03.007>. Citado na página 36.

LAZARIC, A.; RESTELLI, M.; BONARINI, A. Transfer of samples in batch reinforcement learning. In: *ICML*. [S.l.]: ACM, 2008. (ACM International Conference Proceeding Series, v. 307), p. 544–551. Citado na página 35.

LI, L.; WALSH, T. J.; LITTMAN, M. L. Towards a unified theory of state abstraction for mdps. In: *ISAIM*. [s.n.], 2006. Disponível em: <http://dblp.uni-trier.de/db/conf/isaim/isaim2006.html#LiWL06>. Citado 2 vezes nas páginas 23 e 24.

PAINTER-WAKEFIELD, C.; PARR, R. Greedy algorithms for sparse reinforcement learning. In: *ICML*. [S.l.]: icml.cc / Omnipress, 2012. Citado na página 24.

PAN, S. J.; YANG, Q. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.*, v. 22, n. 10, p. 1345–1359, 2010. Disponível em: <http://dx.doi.org/10.1109/TKDE.2009.191>. Citado na página 35.

PONSEN, M.; TAYLOR, M.; TUYLS, K. Abstraction and Generalization in Reinforcement Learning: A Summary and Framework. In: TAYLOR, M.; TUYLS, K. (Ed.). *Adaptive and Learning Agents*. Springer Berlin Heidelberg, 2010, (Lecture Notes in Computer Science, v. 5924). p. 1–32. Disponível em: http://dx.doi.org/10.1007/978-3-642-11814-2_1. Citado na página 31.

PUTERMAN, M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. First. New York: Wiley, 1994. ISBN 0471619779. Disponível em: <http://portal.acm.org/citation.cfm?id=528623>. Citado na página 28.

- RUMMERY, G. A.; NIRANJAN, M. *On-Line Q-Learning Using Connectionist Systems*. Cambridge, England, 1994. Citado 3 vezes nas páginas 19, 29 e 30.
- SEIJEN, H. van; WHITESON, S.; KESTER, L. Efficient abstraction selection in reinforcement learning. In: *SARA 2013: Proceedings of the Tenth Symposium on Abstraction, Reformulation, and Approximation*. [S.l.: s.n.], 2013. p. 123–127. Extended Abstract. Citado 2 vezes nas páginas 24 e 26.
- STONE, P.; SUTTON, R. S.; KUHLMANN, G. Reinforcement learning for robocup-soccer keepaway. *Adaptive Behavior*, v. 13, n. 3, p. 165–188, 2005. Citado na página 34.
- SUTTON, R. S. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In: *Advances in Neural Information Processing Systems 8*. [S.l.]: MIT Press, 1996. p. 1038–1044. Citado na página 19.
- SUTTON, R. S.; BARTO, A. G. *Introduction to Reinforcement Learning*. 1st. ed. Cambridge, MA, USA: MIT Press, 1998. ISBN 0262193981. Citado 4 vezes nas páginas 19, 23, 33 e 34.
- TANG, J.; ALELYANI, S.; LIU, H. *Feature Selection for Classification: A Review*. [S.l.]: CRC Press, 2013. Citado 2 vezes nas páginas 20 e 36.
- TAYLOR, M. E.; STONE, P. Behavior transfer for value-function-based reinforcement learning. In: DIGNUM, F. et al. (Ed.). *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*. New York, NY: ACM Press, 2005. p. 53–59. Citado 2 vezes nas páginas 20 e 35.
- TAYLOR, M. E.; STONE, P. Transfer learning for reinforcement learning domains: A survey. *The Journal of Machine Learning Research*, v. 10, p. 1633–1685, 2009. Disponível em: <http://dl.acm.org/citation.cfm?id=1755839>. Citado 2 vezes nas páginas 20 e 35.
- TAYLOR, M. E.; WHITESON, S.; STONE, P. Transfer learning for policy search methods. In: *In ICML Workshop on Structural Knowledge Transfer for Machine Learning*. [S.l.: s.n.], 2006. Citado 2 vezes nas páginas 20 e 35.
- TORREY, L.; SHAVLIK, J. Transfer learning. *Handbook of Research on Machine Learning Applications*. IGI Global, CiteSeer, v. 3, p. 17–35, 2009. Citado na página 20.
- TORREY, L. et al. Skill acquisition via transfer learning and advice taking. In: *ECML*. [S.l.]: Springer, 2006. (Lecture Notes in Computer Science, v. 4212), p. 425–436. Citado 2 vezes nas páginas 20 e 35.
- WALSH, T. J.; LI, L.; LITTMAN, M. L. Transferring state abstractions between mdps. In: *In ICML Workshop on Structural Knowledge Transfer for Machine Learning*. [S.l.: s.n.], 2006. Citado 2 vezes nas páginas 20 e 35.
- WATKINS, C.; DAYAN, P. Technical Note. In: SUTTON, R. (Ed.). *Reinforcement Learning*. Springer US, 1992, (The Springer International Series in Engineering and Computer Science, v. 173). p. 55–68. ISBN 978-1-4613-6608-9. Disponível em: http://dx.doi.org/10.1007/978-1-4615-3618-5_4. Citado 8 vezes nas páginas 19, 27, 28, 29, 30, 31, 32 e 34.
- WHITESON, S.; TAYLOR, M. E.; STONE, P. *Adaptive Tile Coding for Value Function Approximation*. [S.l.], 2007. Citado na página 34.

ZHANG, T. Adaptive forward-backward greedy algorithm for sparse learning with linear models. In: NIPS. [S.l.], 2008. Citado na página 37.

ZHAO, F. et al. Greedy feature replacement for online value function approximation. *Journal of Zhejiang University - Science C*, v. 15, n. 3, p. 223–231, 2014. Disponível em: <http://dx.doi.org/10.1631/jzus.C1300246>. Citado na página 24.