Universidade de Brasília
Faculdade de Tecnologia

Aprendizado por Reforço na Geração de Marcha em Robô Quadrúpede

Aluno: Rubens Saito Mira Braz

Pesquisador Responsável: Alexandre Ricardo Soares Romariz

Resumo

Este projeto de pesquisa explora o Aprendizado por Reforço para melhoria de marchas de um robô quadrúpede. Tais marchas são geradas por parâmetros e, com um algoritmo em Python que interpreta notas, foi possível obter uma melhoria de pouco mais de 30% na distância percorrida.

Introdução

Contextualização

Robôs quadrúpedes são estudados pelas suas diversas aplicações. A disposição das patas permite que essa classe de robôs consiga acessar áreas acidentadas e assim ajudar no resgate de vítimas. Além disso, o transporte de cargas também é bastante explorado já que a distribuição do peso se dá entre as patas, proporcionando um alívio sobre os motores.

Histórico da Plataforma e Objetivos

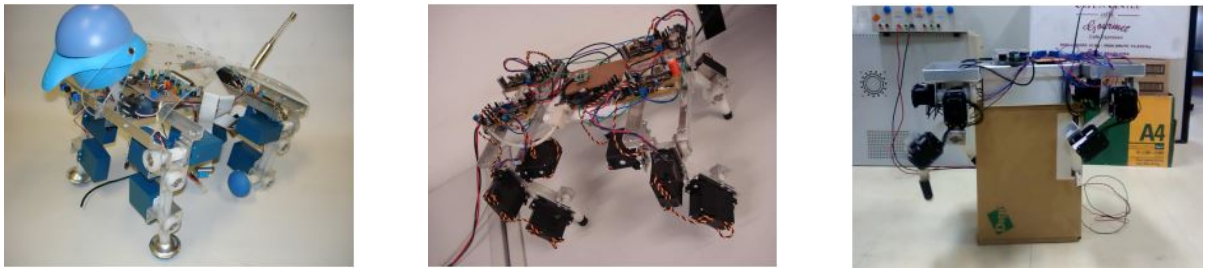
Diversos trabalhos foram realizados desde 2006 para a implementação e aperfeiçoamento de um robô quadrúpede no Laboratório de Automação e Robótica (LARA) da

Universidade de Brasília (UnB). Na tabela 1 são mostrados os últimos projetos feitos utilizando esse robô. Alguns dos principais objetos de estudo eram a robótica comportamental e a robótica terrestre.

Autores	Ano	Título
PORPHIRIO, C. de F.; SANTANA, P. H. M.	2017	Geração de marchas para a plataforma quadrúpede utilizando algoritmo genético.
FLORIANO, B. R de O.	2017	Desenvolvimento e interação de um sistema de controle de equilíbrio para um robô quadrúpede.
DOS SANTOS, P. M .F; DUARTE, V. A.	2018	Aprendizado de marchas factíveis para um robô quadrúpede utilizando algoritmo genético multiobjetivo.
BRAZ, JÚLIA CABRAL DINIZ	2018	Melhoramento de marcha para robô quadrúpede utilizando estratégia de aprendizagem por reforço

Tabela 1: Os mais recentes projetos desenvolvidos utilizando a plataforma quadrúpede do LARA.

Na figura 1, podemos ver as várias versões da plataforma utilizada. Durante o desenvolvimento desse projeto, o robô usado é o da figura 1 c).



(a) Primeira versão desenvolvida [4]

(b) Quadrúpede após o trabalho de Batista e Cardoso [5]

(c) Plataforma reestruturada por Santos [6]

Figura 1: Histórico do robô quadrúpede do LARA. [7]

Aprendizado de Máquina

Como Arthur Samuel definiu [1], aprendizado de máquina é o "campo de estudo que dá aos computadores a habilidade de aprender sem serem explicitamente programados". Isto é, utilizando algoritmos para coletar e aprender com os dados, podemos fazer previsões e melhorar comportamentos.

Definição do problema e Objetivo do Projeto

Dos Santos e Duarte [2] criaram em 2018 um modelo de marcha adequado ao funcionamento da plataforma quadrúpede utilizada nesse projeto. Essa marcha possui quatro poses, que são descritas por um conjunto de parâmetros.

Devido ao grande número de testes feitos no robô para a obtenção dos parâmetros ideais, esse projeto foca em utilizar formas mais eficiente de aprendizado, utilizando-se do mesmo modelo de marcha.

Fundamentos Teóricos

Aprendizado por reforço

Aprendizado por reforço (RL, do inglês *Reinforcement Learning*) é diferente de aprendizado supervisionado (AS). No AS, o agente aprende a partir de um conjunto de dados pré-determinados por um supervisor externo. Ou seja, as ações tomadas foram definidas anteriormente. O objetivo do RL é o agente generalizar sua resposta para situações que ainda não tenham sido programadas.

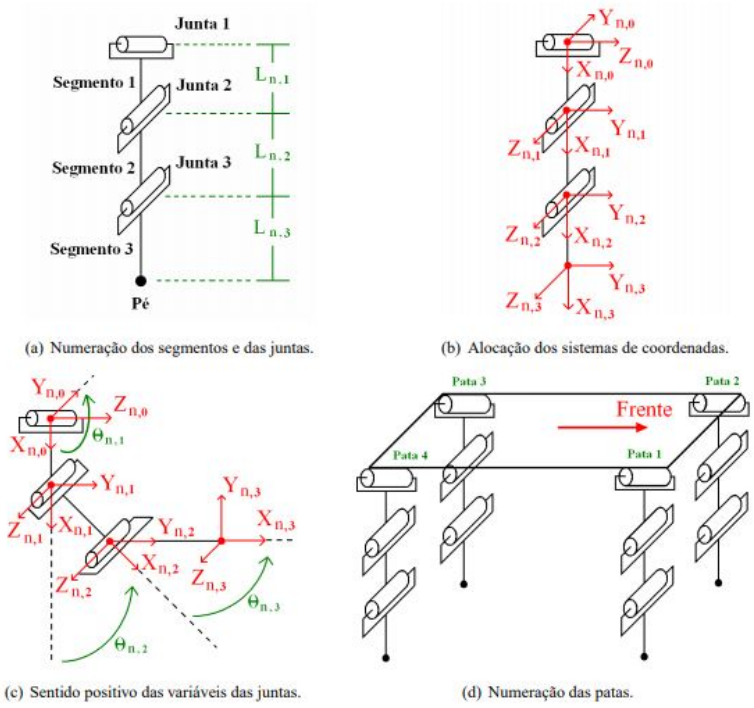
Pode-se imaginar que o RL é um aprendizado não-supervisionado pois ele não precisa de exemplos de comportamentos corretos, no entanto, esse tipo de algoritmo tenta maximizar um sinal de recompensa em vez de encontrar padrões.

Na maioria das vezes, projetos de pesquisa envolvendo RL usam o processo de decisão de Markov. Isto é, um agente - tomador de decisões - encontra uma situação para a qual ele ainda não foi testado. Logo, seu objetivo passa a ser encontrar uma sequencia de ações para operar o sistema de forma mais eficiente, com base em critérios de funcionamento previamente estabelecidos e levando em conta que o estado do sistema futuro à tomada de decisão depende do estado atual [3].

Estrutura do robô quadrúpede

O robô possui uma estrutura de alumínio e patas de nylon. Sua estrutura é composta por quatro patas. Cada pata possui 3 servos motores, totalizando assim doze graus de liberdade de movimento.

A Figura 2 representa a localização dos servos motores na plataforma e a convenção adotada para nomenclatura das patas. Os servos utilizados são do modelo *Dynamixel* RX-28 da *Robotis*, a tensão de alimentação é de no mínimo 12 Volts, sendo 14,8 Volts a tensão recomendada pelo fabricante [8]. O torque máximo de servo é de 3,7 N.m.



(a) Numeração dos segmentos e das juntas. (b) Alocação dos sistemas de coordenadas.

(c) Sentido positivo das variáveis das juntas. (d) Numeração das patas.

Figura 2: Estrutura do quadrúpede [9].

Sensoriamento

O sistema de sensoriamento do robô foi feito por Porphiro e Santana [10]. Atualmente, o sistema do quadrúpede oferece medidas sobre a aceleração - utilizando-se um acelerômetro (modelo Digital ADXL345) - e medidas das forças nas extremidades das patas - utilizando-se um sensor resistivo de força (modelo FSR 400, possui resolução de 0,2N e atua entre 0,2 e 20N).

A informação sobre a força aplicada nas patas é relevante para se conhecer a carga que cada pata está submetida, além de mostrar o exato momento do toque da pata no chão.

O acelerômetro informa a orientação do centro de massa do quadrúpede, por conseguinte é possível saber se o mesmo está na trajetória definida.

Sistemas embarcados

O quadrúpede é controlado por uma integração entre *Raspberry PI* e o *Arduíno*, em uma relação mestre-escravo. O Arduíno coleta dados dos sensores de força e do acelerômetro e envia ao *Raspberry PI*, via serial. O *Raspberry PI* é responsável por controlar

os servo motores a partir de um código em C++, com uma biblioteca da *Dynamixel*, fabricante dos motores.

Definições da marcha e simulações

O trabalho realizado de Dos Santos e Duarte [2], que utiliza algoritmo genético, encontrou 4 poses para as patas do quadrúpede utilizando cinemática inversa. Essas poses podem ser vistas na figura 3 e foram inspiradas em caminhadas de cavalos.

Cada marcha para o quadrúpede é uma composição dessas 4 poses. E, além disso, cada pose é alterada por parâmetros (são 5 ao todo) - eles são: ângulos de aberturas das poses 2 ($angle - front$) e 3 ($angle - back$) e posição no espaço da extremidade durante pose de transição, pose 1 (x_c, y_c, z_c). Logo, os parâmetros são: $P = \{angle - front, angle - back, x_c, y_c, z_c\}$.

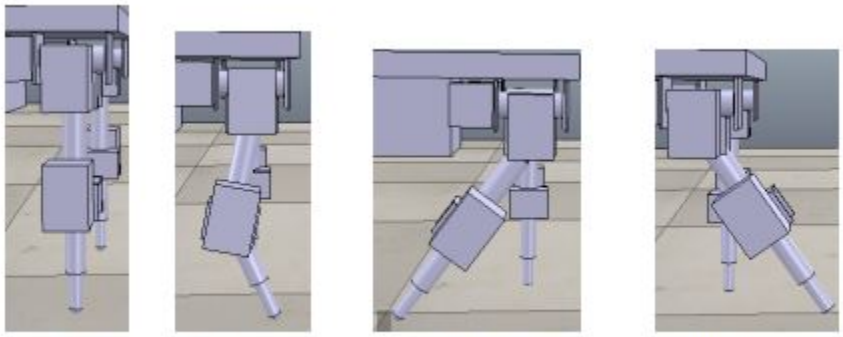


Figura 3: Poses das patas geradas por cinemática inversa [2].

Cada pata executa a seguinte sequência de poses durante uma marcha (a ordem das poses é chamada de ciclo e ele é repetido duas vezes):

- **Pata 1:** Pose 3 → Pose 4 → Pose 1 → Pose 2;
- **Pata 2:** Pose 1 → Pose 2 → Pose 3 → Pose 4;
- **Pata 3:** Pose 3 → Pose 4 → Pose 1 → Pose 2;
- **Pata 4:** Pose 1 → Pose 2 → Pose 3 → Pose 4;

* A localização de cada pata na estrutura do robô pode ser vista na figura 2.

O software *V-REP* foi escolhido para simular o comportamento do robô pois ele se destaca por oferecer uma grande gama de técnicas de simulação e pela portabilidade de

seus modelos e controladores [10]. Além disso, ele também já havia sido adotado por Porphirio e Santana [10] para simulação da plataforma quadrúpede e esse trabalho usou esse modelo já desenvolvido anteriormente.

Metodologia

Introdução

Para a tentativa de melhora das marchas desenvolvidas por Dos Santos e Duarte [2], foram utilizados dois métodos diferentes de aprendizado por reforço. Na primeira tentativa, foi usado um algoritmo de diferença temporal com traço de elegibilidade, mas ao se mostrar demorado e não muito efetivo, outro algoritmo foi utilizado. Esta segunda tentativa é baseada no gradiente de política, que se mostrou mais rápido e eficaz. Sendo inclusive factível para aplicação direta no robô, sem antes passar por simulações.

Diferença temporal com traço de elegibilidade

O modelo de robô quadrúpede usado nesse trabalho precisa aprender apenas os melhores valores para os parâmetros que geram as marchas. Considerando que o estado é um conjunto de parâmetros o importante é determinar qual o melhor estado (valores ótimos dos parâmetros), a ação a ser tomada em cada estado não é importante.

Para um teste de eficiência desse método, foi tentada a variação de apenas 3 dos 5 parâmetros que compões as marchas. Quando mais parâmetros são variados, maior é o espaço de estados e a análise se torna muito complexa e o tempo de simulação muito extenso. Cada parâmetro usado nesse algoritmo foi discretizado em um intervalo específico - isso é necessário para que o espaço de estados seja minimizado e também evitar valores que impeçam o correto funcionamento do robô.

Os valores de mínimo e máximo de cada parâmetro foi definido no trabalho de Dos Santos e Duarte [2] - e os mesmos estão definidos abaixo:

- $angle - back$ e $angle - front$: Estes parâmetros são os ângulos em que a pata do robô faz nas poses 2 e 3, que podem ser vistas na Figura 3. Possuem valor de 30°, pois nas marchas obtidas por Dos Santos e Duarte [2] eles se aproximavam deste valor;

- x_c: Representa a posição x que a extremidade da pata do robô deve alcançar durante a pose 1. Limitada no intervalo [17,21] cm, com um passo de 0,4.

- y_c: Representa a posição y que a extremidade da pata do robô deve alcançar durante a pose 1. Limitada no intervalo [-4,0] cm, com um passo de 0,4.

- z_c: Representa a posição z que a extremidade da pata do robô deve alcançar durante a pose 1. Limitada no intervalo [0,4] cm, com um passo de 0,4.

Na aplicação deste algoritmo, o problema de aprendizado é definido como um Processo de Decisão de Markov, que é composto de:

- S: Estados s que são um conjunto de parâmetros, $s(x_c, y_c, z_c)$;

- A: Ações $a(s)$: pode incrementar o valor de um parâmetro, decrementar o valor de um parâmetro, e continuar no mesmo estado.

- $T(s, a, s')$: Função de transição determinística que permite o incremento ou decremento de apenas um parâmetro e ainda permite o sistema ficar no mesmo estado.

- r: Uma recompensa negativa é dada ao robô quando ele falha em andar certa distância mínima, ou quando cai, e uma recompensa positiva é determinada como uma função da distância percorrida.

Foi utilizado um algoritmo no MATLAB [11] para otimizar esses parâmetros.

Gradiente de Política

Utilizando o método de Diferença Temporal com Traço de Elegibilidade, foi constatado que o mesmo demorava exageradamente para encontrar os valores dos estados. Por isso, foi buscada uma alternativa mais eficaz. Foi adotado o método de Gradiente de Política, que além de mais eficaz, também não necessita da discretização dos parâmetros - ele também é capaz de trabalhar em grande espaço de estados.

Foi utilizado um algoritmo em *MATLAB* baseado em um desenvolvido no trabalho de Kohl e Stone [12]. Este método busca um máximo global, em contrapartida aos métodos de gradiente tradicionais, que buscam máximos locais.

O método utilizado pode ser considerado uma derivação dos métodos de gradiente de política padrões. Ao utilizarmos métodos de gradiente de política a convergência em busca da política ótima apenas se aproximará do máximo local, em contraste com a técnica utilizada no trabalho citado acima em que converge para um máximo global.

Nesta técnica, consideramos um vetor de parâmetros iniciais $P_{ini} = \{angle{-}back, angle{-}front, x_c, y_c, z_c\}$ e é estimada a derivada parcial da função objetivo de P_{ini} em relação a cada parâmetro. Isso é feito primeiramente avaliando n políticas próximas de $P_{ini}(R_1, R_2, \cdots, R_n)$, que são geradas aleatoriamente. Essas políticas são definidas como:

$$R_i = \{angle - back + \Delta_1, angle - front + \Delta_2, x_c + \Delta_3, y_c + \Delta_4, z_c + \Delta_5\}$$

onde Δ_n é escolhido de modo aleatório entre três opções $\{+\epsilon_n, 0, -\epsilon_n\}$. Cada ϵ_n é um valor fixado de acordo com o parâmetro a qual ele está ligado.

Uma nota (*score*) da marcha é atribuída para cada política. Essa nota pode ser tanto a distância percorrida por y, a variação no eixo x, a variação no eixo z, ou até mesmo uma média ponderada de todos estas medidas. A função objetivo pode ser maximizar, ou até mesmo minimizar, estes valores.

Depois que todas as políticas são avaliadas são estimadas as derivadas parciais em relação a cada um dos 5 parâmetros. Isso é feito agrupando cada R_i em um dos 3 conjuntos para cada umas das dimensões:

$$R_i \in \begin{cases} S_{+\epsilon,n}, & \text{se o enésimo parâmetro de } R_i \text{ é } p_n + \epsilon_n; \\ S_{0,n}, & \text{se o enésimo parâmetro de } R_i \text{ é } p_n + 0; \\ S_{-\epsilon,n}, & \text{se o enésimo parâmetro de } R_i \text{ é } p_n - \epsilon_n. \end{cases}$$

É computada então um nota média $Avg_{+\epsilon,n}$, $Avg_{0,n}$ e $Avg_{-\epsilon,n}$ para $S_{+\epsilon,n}$, $S_{0,n}$ e $S_{-\epsilon,n}$ respectivamente. Essas três médias representam qual é o beneficio de se alterar o enésimo parâmetro por $+\epsilon_n$, 0 ou $-\epsilon_n$.

Essas médias são usadas para ajustar um vetor A com 5 variáveis, onde:

$$A_n = \begin{cases} 0, & \text{se } Avg_{+0,n} > Avg_{+\epsilon,n} \text{ e } Avg_{+0,n} > Avg_{-\epsilon,n}; \\ Avg_{+\epsilon,n} - Avg_{-\epsilon,n}, & \text{caso contrário.} \end{cases}$$

O vetor A é normalizado e multiplicado por um escalar de passo η, para que o ajuste seja constante (fixo) a cada iteração. Ao final adicionamos A ao vetor P_{ini}, e uma nova iteração é iniciada.

O algoritmo desenvolvido no *MATLAB*, faz a cada iteração 5 avaliações de diferentes marchas. Como as simulações no *V-REP* possuem muito ruído, cada marcha era simulada 3 vezes. A cada vez que o algoritmo era rodado eram feitas 10 iterações. O algoritmo foi rodado pelo menos duas vezes, e, se em uma nova rodada, pelo menos uma política teve *score* maior que todas as políticas da rodada anterior, o algoritmo era rodado mais uma vez, caso contrário, ele não era mais iniciado e a política que obteve maior *score* da vez anterior era considerada a política ótima.

Vários objetivos diferentes foram testados nesse método de aprendizado [7]:

- Maximização da distância no eixo y. Para isso, o *score* é considerado a distância total percorrida neste eixo.
- Minimização da variação no eixo z, ou seja, minimizar o balanço da altura do robô na tentativa de penalizar políticas em que o robô acaba por cair. Para isso o *score* é o inverso da diferença entre o maior e menor valor lido pelo GPS no eixo z de coordenadas. Se utiliza do inverso pois o algoritmo visa maximizar o *score*.

- Minimização da variação no eixo x, pois se deseja que o robô ande apenas no eixo y, se há uma diferença grande no eixo x que dizer que a marcha gerada pende para um algum lado. Utiliza-se novamente o inverso da diferença entre os valores máximo e mínimos lidos pelo GPS.

O *MATLAB*, que inicia a simulação no *V-REP*, foi utilizado num primeiro momento para testar o algoritmo desenvolvido. Dado os bons resultados das simulações, o algoritmo foi reescrito em *Python 3.6* para que ele fosse testado na plataforma real. O código em *Python* gerava as marchas que o código em C++ executava.

Resultados - Gradiente de Política:

Maximização da distância percorrida em y

A princípio, a função objetivo foi apenas maximizar a distância percorrida no eixo y. Para começar a execução do algoritmo de gradiente de política, é necessário a escolha de uma política inicial, foi escolhida a marcha final obtida no trabalho de Dos Santos e Duarte (marcha que gera $61cm$ de distância percorrida), pois a marcha achada por eles obtém resultados satisfatórios e a convergência desse algoritmo depende de um bom chute inicial.

A Figura 4 apresenta os resultados de duas execuções do algoritmo sendo iniciando com os parâmetros obtidos por Dos Santos e Duarte [2]. As iterações que possuem valor zero, revelam parâmetros incapazes de gerar marchas factíveis. Na Simulação 1, foram necessárias 20 iterações para se encontrar um valor máximo de $55cm$. Na Simulação 5, foram necessárias 30 iterações e foi encontrado um valor máximo de $65cm$.

Na terceira execução, foi adotada como política inicial os resultados obtidos na Simulação 5. Ao final, o resultado obtido foi uma distância total percorrida de $72cm$. Isso mostra que ao se iniciar as iterações em políticas boas, é possível a obtenção de melhores resultados. Também podemos verificar que começar em políticas diferentes geram resultados diferentes, pois esse método encontra a política ótima local.

Esses resultados foram advindos de simulações no *V-REP*, depois as marchas também foram testadas na plataforma real e os resultados foram bem próximos.

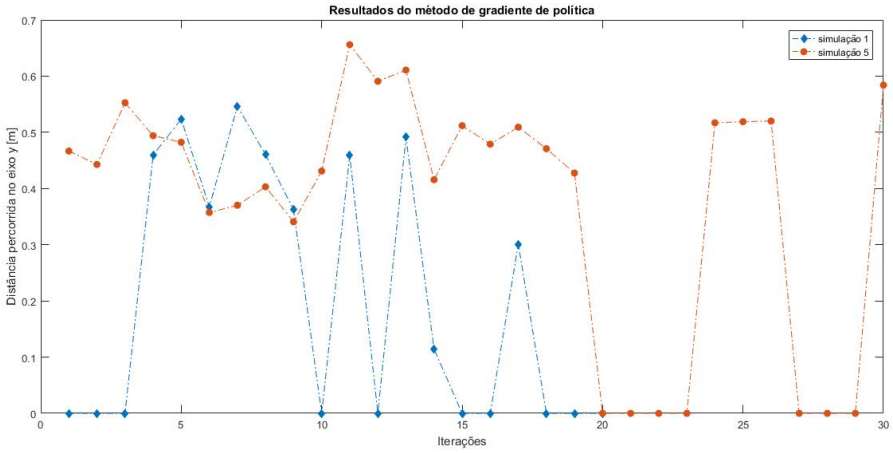


Figura 4: Distância percorrida ao final de cada iteração da primeira e segunda execução do algoritmo.

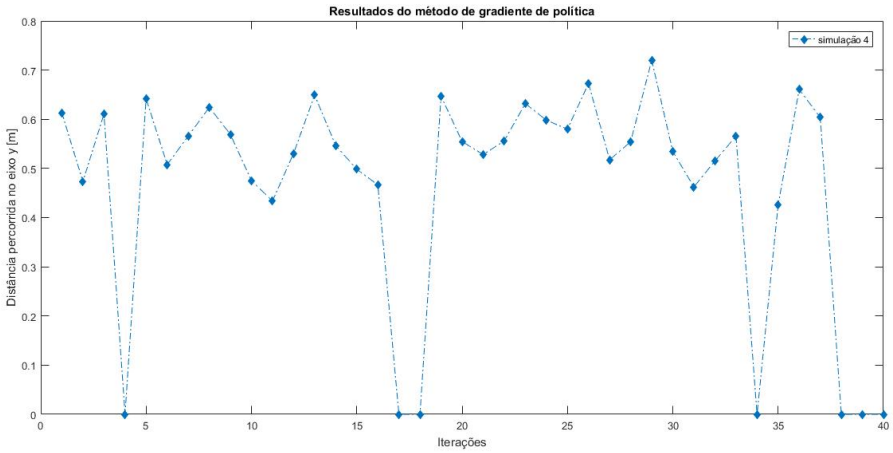


Figura 5: Distância percorrida ao final de cada iteração da terceira execução do algoritmo.

Conclusão

Este trabalho foi capaz de usar a técnica de aprendizado por reforço para obter valores de parâmetros do gerador de marcha de um robô quadrúpede em um ambiente simulado.

Para a otimização foram utilizados dois métodos diferentes.

O primeiro foi o de diferença temporal, onde os parâmetros da marcha são considerados o estado do robô. Neste método, o valor dos estados são aproximados conforme os estados são visitados e uma implementação com parâmetros contínuos seria inviável.

Com as restrições impostas, para se obter uma estimativa da função valor eram necessárias muitas horas, cerca de 15. Os parâmetros foram discretizados a passos muito grandes e, sendo a marcha muito sensível a estes parâmetros, foi visto que outras algoritmos de aprendizado poderiam ser mais eficientes.

O segundo método utilizado foi o de gradiente de política. Nesse algoritmo, os parâmetros não são mais tidos como estados mas sim como uma política que o robô executa. Logo, o objetivo passa a ser encontrar a política ótima.

Com esse segundo método, o tempo de simulação diminuiu bastante, passando a ser de 1:30h, e com poucas iterações foi possível a obtenção de resultados bem melhores. A marcha usada para inciar o algoritmo conseguia percorrer uma distância de $61cm$ e a marcha final encontrada nesse trabalho chegou a andar $80cm$ (ambas distâncias medidas na plataforma real), um aumento de mais de 30%.

A desvantagem no uso deste segundo método está no fato de que, diferentemente do método utilizado anteriormente, a busca pelos melhores parâmetros converge apenas para um ótimo local.

A 'Marcha 0' é a que foi obtida no trabalho de Dos Santos e Duarte [2], e a 'Marcha 12' é a que obteve a maior distância percorrida.

Bibliografia

[1] SAMUEL, A. L. Some studies in machine learning using the game of checkers. IBM Journal ofresearch and development, IBM, v. 3, n. 3, p. 210–229, 1959.

[2] SANTOS, P. M. F. dos; DUARTE, V. A.Aprendizado de marcha para um robô quadrúpede utilizando algoritmo genético multiobjetivo. Monografia (Graduação) — Universidade de Brasília, Brasília, Julho 2018.

[3] PUTERMAN, M. L.Markov decision processes: discrete stochastic dynamic programming. 1. ed. New Jersey: Wiley-Interscience, 1994. (Wiley Series in Probability and Statistics). ISBN 0471619779,9780471619772.

[4] COTTA, G. H.; RAULINO NETO, L. Realização de uma plataforma para estudo de robótica comportamental baseada em quadrúpedes. Monografia (Graduação) — Universidade de Brasília, Brasília, 2006.

[5] BATISTA, G. F.; CARDOSO, I. F. Adequação de um sistema de locomoção de um robô quadrúpede para avaliação de algoritmos de aprendizagem. Monografia (Graduação) — Universidade de Brasília, Brasília, 2007.

[6] SANTOS, J. H. de S. Plataforma quadrúpede: Uma nova estrutura para robô quadrúpede do LARA. Monografia (Graduação) — Universidade de Brasília, Brasília, Julho 2016.

[7] BRAZ, JÚLIA CABRAL DINIZ. Melhoramento de marcha para robô quadrúpede utilizando estratégia de aprendizagem por reforço. Monografia (Graduação) — Universidade de Brasília, Brasília, Julho 2018.

[8] ROBOTICS.ROBOTIS E-Manual. Disponível em: `http://emanual.robotis.com/docs/en/software/rplus1/dynamixel\_wizard/`. Acesso em: 18 jun. 2019.

[9] SOUTO, R. F. Modelagem cinemática de um robô quadrúpede e geração de seus movimentos usando filtragem estocástica. Monografia (Graduação) — Universidade de Brasília, Brasília, Julho 2007.

[10] PORPHIRIO, C. de F.; SANTANA, P. H. M. Geração de marchas para a plataforma quadrúpede utilizando algoritmo genético. Monografia (Graduação) — Universidade de Brasília, Brasília, Junho 2017.

[11] MathWorks. MATLAB. Aplicativo para Windows 10.

[12] KOHL, N.; STONE, P. Policy gradient reinforcement learning for fast quadrupedal locomotion. In: IEEE.Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference. New Orleans, 2004.