Technical Report
Shepherding with Temporal Difference Learning

Alysson Ribeiro da Silva
Computer Science Graduate Program
Federal University of Minas Gerais
Belo Horizonte, Brazil

Abstract—Neste projeto foi desenvolvido e avaliado um simulador para controle de múltiplos agentes, assim como um modelo de pastoreio baseado em regras e também por aprendizado. Os resultados sugerem que a recompensa e o aprendizado temporal foram suficientes para o aprendizado de uma política de pastoreio de um único agente.

Index Terms—Shepherding, q-learning, temporal difference learning, reinforcement learning

1 Introduction

Colisões de pássaros com aviões em aeroportos e multidões de pessoas depredando propriedades públicas e privadas são problemas difíceis de serem mitigados no contexto do pastoreio, porque a solução dos mesmos envolve liderar ou coibir os pássaros ou humanos para um local seguro escolhido a priori. Uma das possíveis abordagens para sanar tal problema é considerar que os pássaros ou pessoas são agentes que devem ser coagidos a se deslocar para uma área alvo por um robô líder ou robô pastor.

A criação de robôs pastores pode seguir duas abordagem. A primeira abordagem visa a criação de um conjunto de regras por um especialista humano que são utilizadas para guiar o pastor. Já a segunda abordagem, visa aprender uma política que irá gerar ou auxiliará na escolha de comportamentos que o pastor deve executar para concluir a tarefa. Ambas as abordagens visam minimizar o tempo de deslocamento dos agentes até a área alvo. Por exemplo, Anil et al. [1] propôs em 2017 um conjunto de regras para guiar pastores de forma a auxiliar a realocação de agentes para uma área alvo. Em contrapartida, Zhi et al. [2] propõe que o pastor aprenda a guiar agentes através da utilização de redes neurais profunda e aprendizado por reforço.

Tipicamente, o problema de pastoreio pode ser modelado como um Processo de Decisão de Markov Parcialmente Observável (PDMPO), onde deseja-se aprender uma política que permita escolha de ações do pastor de forma a minimizar o tempo de deslocamento dos agentes até a área objetivo. Para lidar com um PDMPO é necessário um modelo de duas camadas. A primeira camada é responsável por servir como uma memória que armazena os estados do PDMPO. Já a segunda camada é utilizada para auxiliar no processo de otimização e obtenção de um caminho que maximize a recompensa total.

Os maiores problemas ligados a utilização de um PDMPO são a falta da capacidade de convergência por consequência da quantidade de estados e a dificuldade de otimização do mesmo sem conhecimento especialista do domínio. Consequentemente, muitos autores, como Guillaume et al. [3] e Binyu et al. [4], modelam a primeira camada com auxílio de uma rede neural, profunda ou não e de diversas naturezas, e a segunda camada através do aprendizado temporal (TD-Learning). Dessa forma, é possível generalizar os estados, reduzindo sua quantidade e complexidade do problema, e ainda eliminar a necessidade de um especialista através da utilização do aprendizado temporal.

Dentre as técnicas mais comuns para a incorporação do TD-Learning, existe o Q-Learning. Por exemplo, G. A. Gardona et al. [5] e X. Qiu et al. [6] mostram como diferentes arquiteturas de redes neurais em conjunto ao Q-Learning podem ser utilizadas para permitir que agentes aprendam uma política que permita com que os mesmos naveguem em um ambiente desconhecido.

Com isso em mente, o principal objetivo deste projeto é explorar, avaliar propor, implementar, e avaliar agentes pastores utilizando uma técnica de aprendizado temporal. O mesmo possui os seguintes sub-objetivos:

1) Desenvolvimento de um ambiente de simulação multi-agentes para realizar simulações de pastoreio com foco em aprendizado por reforço e técnicas reativas.
2) Validação do simulador com técnica reativa.
3) Proposta, implementação, e avaliação do desempenho de um agente pastor que utiliza o q-learning em conjunto a uma memória dinâmica para aprender uma política de pastoreio.

2 Method

Dois tipos de agentes foram utilizados nesta proposta. O primeiro tipo de agente é o pastor e o mesmo tem a finalidade de guiar. O segundo tipo de agente é o agente ovelha, e o mesmo representa um conjunto de pessoas, animais, ou outros agentes. O objetivo do pastor é guiar as ovelhas para

zonas de segurança através da influência de seu campo de repulsão nas ovelhas.

2.1 Ovelha

As ovelhas foram da proposta utilizam o modelo de boids proposto por Reynolds [7], pois o mesmo proporciona um comportamento mais realista para sua movimentação. Esse modelo propõe que cada agente seja guiado por forças de repulsão, alinhamento, e coesão. Um exemplo das forças presentes no modelo de boids de Reynols é ilustrado pela Figura 1.

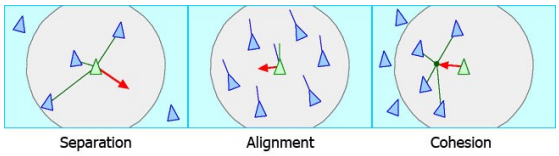

Fig. 1. Forças presentes no modelo de boids de Reynolds [7].

A força de repulsão é responsável por não deixar que as ovelhas se juntem e causem colisões. Já a força de alinhamento é calculada com a finalidade de que um agente se alinhe com seu vizinho próximo e dentro de seu campo de visão. Por fim, a coesão tem a finalidade de direcionar os agentes para um centro de massa local perto de sua vizinhança.

2.2 Pastor

Os robô pastor possui dois estados, sendo o primeiro responsável por selecionar ações e executar o aprendizado temporal, e o segundo tem a finalidade de executar um movimento no espaço. Enquanto o pastor executa um movimento, o mesmo não pode executar o aprendizado temporal e vise-versa. A seguir são apresentados a representação do estado do mesmo e o modelo de recompensa utilizado para o POMD e Q-Learning.

2.3 Estado

Um estado do modelo abordado é representado como uma matriz I de tamanho NxM, onde N e M são a altura e largura de um retângulo cujo centroide é a posição do robô pastor. Cada célula dessa matriz representa um quadrante de um retângulo de observação, como ilustrado pela Figura 2. Agentes que estão dentro do retângulo de observação são representados como uma célula preenchida. Já os agentes que estão fora do retângulo de observação, são representados na borda da matriz I. A direção do objetivo também é representada por uma célula preenchida na borda da matriz I.

Para saber qual célula será preenchida dentro da matriz I, é utilizado o algoritmo de Clipping de Liang Barsky. O intuito por trás do mesmo é transformar um ponto fora do retângulo de observação em uma ponto na borda da matriz I, assim representando a direção do objeto. Caso o ponto observado esteja dentro do retângulo de observação, então o mesmo é considerado como dentro de I. Uma vez que a resolução de I é menor que do retângulo de observação, normalizações e transformações são aplicadas para representação em I.

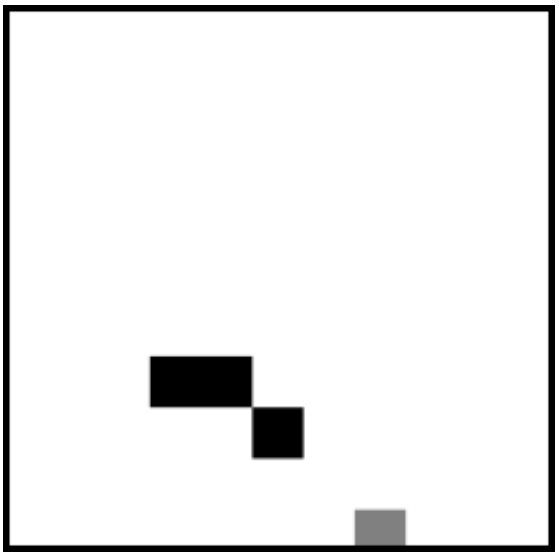

Fig. 2. Representação do estado do agente dentro da matriz I. É importante notar que o tamanho de I é menor que o tamanho da área de visibilidade e portanto o mesmo representa uma redução do espaço de busca. As células pretas representam ovelhas e a célula cinza representa a direção do objetivo.

2.4 Memória

Para ser capaz de armazenar os estados do PDMPO com certo grau de generalização, Zhi et al. [2] utiliza uma rede neural convolucional. Porém, para a validação deste projeto foi utilizado uma Q-Table dinâmica.

2.5 Modelo de Recompensa

A recompensa que permite o aprendizado de uma política capaz de gerar caminhos eficientes para os agentes é calculada em 3 (três) etapas. A primeira etapa é responsável por calcular a recompensa projetiva. Já a segunda etapa, calcula a recompensa de deslocamento. Por fim, a última etapa calcula a recompensa de violação. A recompensa final é a soma das recompensas calculadas em cada etapa ponderadas por fatores de escala.

2.5.1 Recompensa Projetiva

A primeira etapa é responsável por calcular a projeção do vetor $\vec{v}$ para o vetor $\vec{u}$, onde $\vec{v}$ é o vetor formado pelo deslocamento do centroide dos agentes P entre os tempos t e $t-1$, e onde o vetor $\vec{u}$ representa o deslocamento de P até o próximo sub-objetivo do caminho representante C. O caminho representante por sua vez, pode ser um único ponto ou um caminho planejado com auxílio de um Roadmap do ponto P (centro de massa dos agentes) até o ponto objetivo O. A projeção é calculada como $\vec{v}.\vec{u}$ e ela representa o quão correta é a direção que os agentes estão com relação ao caminho representante que leva ao objetivo.

2.5.2 Recompensa de Deslocamento

O objetivo da recompensa de deslocamento é saber se os agentes (não pastores) estão se deslocando na direção correta do caminho representante (caminho que leva o ponto P até a região objetivo O). A mesma é calculada como a diferença $d_{final} = d_{t-1} - d_t$ entre as distâncias geodésicas,

d_{t-1} e d_t, do tempo t e tempo $t-1$, entre o pontos P (centroide dos agentes) e O (ponto objetivo). A distância geodésica é a quantidade de arestas ou sub-objetivos em C (caminho representante) que ainda devem ser percorridos pelo controlador dos agentes (não pastores) para que cheguem ao ponto objetivo O. A aproximação dos agentes do objetivo a cada iteração será refletida como $d_{final} > 0$ e o seu distanciamento como $d_{final} < 0$.

2.5.3 *Recompensa de Violação*

A recompensa de violação tem o intuito de auxiliar identificar condições de fracasso pelo robô pastor para penalizar o mesmo caso algum comportamento inadequado seja explorado. Para auxiliar o cálculo da recompensa de violação são utilizadas duas variáveis. A primeira variável é o raio r entre o agente (não pastor) mais distante do centroide P. Além disso, a segunda variável utilizada é a distância d_{pastor} entre a posição do robô pastor para o centroide P. A recompensa de violação r_v é calculada como,

$$r_v = \begin{cases} -\alpha & \text{caso } r > d_{agentes} \\ -\beta & \text{caso } d > d_{pastor} \\ 0 & \text{caso contrário} \end{cases} \quad (1)$$

onde α e β são fatores de penalização substanciais definidos a priori, e $d_{agentes}$ e d_{pastor} são limiares de tolerância para a dispersão dos agentes (não pastores) e distanciamento do robô pastor para os demais agentes, respectivamente. Além disso, é importante notar que se $r > d_{agentes}$ ou $d > d_{pastor}$, então a simulação do modelo é finalizada e reinicializada.

2.6 Experimental Configuration

Para validar a proposta, foi construído um simulador multi-agentes com auxílio do OpenGL. O mesmo consiste em três camadas. A primeira camada é responsável pela visualização. Já a segunda camada é responsável por permitir o controle e manutenção dos agentes e rotinas de simulação. Por fim, a última camada permite gerenciar o aprendizado por reforço, controlando os episódios, épocas, e rodadas. O mesmo pode ser visualizado na Figura 3.

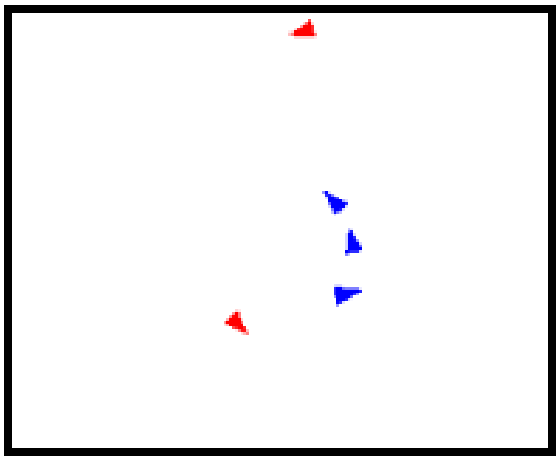

Fig. 3. Simulador desenvolvido para a proposta. Os triângulos vermelhos representam pastores e os azuis são as ovelhas.

3 Configuração Experimental

Dois tipos de sistemas foram utilizados para avaliar a proposta. O primeiro sistema, utilizado para avaliar o simulador, é um sistema multi-pastores (reativos) e multi-ovelhas. Já o segundo, utilizado para avaliar o aprendizado temporal, é composto por um único agente e ovelha. O desempenho de todos os sistemas foi avaliado em termos de taxa de sucesso em 120 épocas compostas por 10 episódios cada. O desvio padrão de cada época foi calculado em 10 rodadas. Entre cada episodio, as ovelhas e pastores são posicionados em locais aleatórios para validar as propostas estatisticamente. Entre as rodadas, a memória utilizada para o aprendizado temporal é zerada, isso permite o cálculo do desvio para cada época.

3.1 Parâmetros das Ovelhas

As ovelhas foram configuradas com força de atração, repulsão, e alinhamento em 0.1. A repulsão do pastor é considerada a partir de 50 pixels de distância. A velocidade linear máxima das ovelhas é 20 pixels por segundo. A aceleração das mesmas ao encontrar um pastor é de 10 pixels por segundo. É aplicada uma força de freio constante a cada iteração de -2.0 pixels por segundo para estabilização do movimento e uma força de -5.0 pixels por segundo para estabilização após ser repelida por um pastor. A velocidade angular das ovelhas é limitada a 0.01 pixels por segundo. Por fim, o ângulo de visibilidade das mesmas é de 45 graus. Demais parâmetros, referentes a cada teste, são explicitados em suas respectivas seções. Para os testes de múltiplos agentes, foram utilizadas 4 ovelhas.

3.2 Parâmetros dos pastores reativos

Os pastores reativos, propostos por [1], baseiam-se em 4 regras.

1) Nada na visão
2) Se uma ovelha está na visão
3) Se um pastor está na visão ou proximidade
4) Se o objetivo está na visão

Para a regra 1, foi utilizada uma aceleração de 10 pixels por segundo e uma velocidade angular de 3 graus por segundo. Já para a regra 2, foi utilizada apenas uma aceleração de 30 pixels por segundo. Para a regra 3, foi utilizada uma aceleração de -10 pixels por segundo. E por fim, para a regra 4 foi utilizada uma aceleração de 10 pixels por segundo e uma velocidade linear de 1 grau por segundo. A velocidade linear máxima foi configura em 15 pixels por segundo e a mínima em 1 pixel por segundo. Por fim, a velocidade angular foi limitada a 1 grau por segundo. Quatro pastores foram utilizados para esta técnica uma vez que a mesma demanda cooperação. Não foram avaliadas performance com diferentes quantidade de pastores.

3.3 Parâmetros do q-learning

O Q-learning foi configurado com $\gamma = 0.9$, $\alpha = 0.1$, total de ações igual a 8, $\epsilon = 1$. O deslocamento do agente para cada ação foi configurado em 20 pixels, e velocidade linear máxima igual 5 pixels por segundo. A recompensa total é normalizada, onde a soma de todas as mesmas é igual a

1. O retângulo de visibilidade do agente foi configura para uma área de 150x150 pixels e a matriz I de representação do estado foi configurada em três tamanhos diferentes descritos na Seção 4.2.

4 Resultados

Nesta seção são apresentados os resultados referentes a validação do simulador e também do pastor que utiliza o aprendizado temporal.

4.1 Validação Simulador

O simulador foi validado através do modelo multi-agentes reativo composto por regras e proposto por [1], onde um conjunto de robôs pastores deve colaborar para liderar as ovelhas até a região de destino. O comportamento do pastor é guiado por um conjunto de quatro regras que servem para decidir qual velocidade angular ou linear os agentes devem possuir. O comportamento descrito pelas mesmas visa fazer com que os múltiplos agentes pastores formem um círculo que englobe todas as ovelhas e levem as mesmas para a direção do objetivo. O mesmo é ilustrado pela Figura 4.

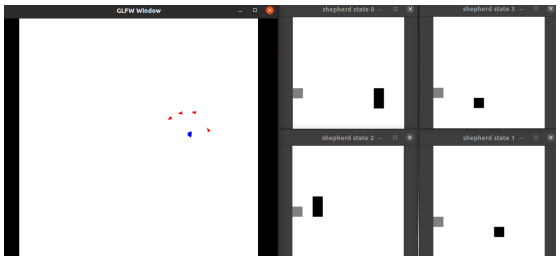

Fig. 4. Comportamento circular dos agentes propostos por [1] no simulador desenvolvido em conjunto ao estado de cada agente.

Como ilustrado pela Figura 5, os agentes cooperativos e reativos obtiveram performance em 100% para todas as épocas. Não foram observados desvios inferiores e superiores. Esse comportamento pode indicar a robustez da técnica ao utilizar 4 pastores para os parâmetros utilizados para guiar o comportamento de bando das ovelhas. Além disso, a implementação desta técnica mostra a viabilidade do simulador desenvolvido para controlar diversos agentes, simular episódios, épocas, e passes, e permitir não somente a visualização em tempo real dos pastores e ovelhas mas também do estado observado para cada um deles simultaneamente.

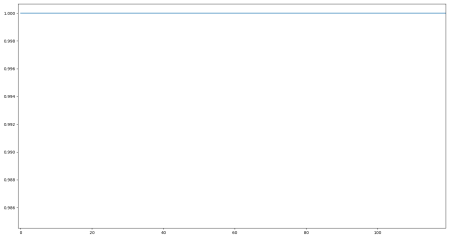

Fig. 5. Resultados para .

4.2 Aprendizado por Reforço

O aprendizado por reforço foi avaliado em três configurações de visibilidade. A primeira configuração utiliza uma visibilidade I de 3x3 células. Já a segunda, utiliza uma visibilidade de 5x5 células. A última, utiliza uma viabilidade de 11x11 células. O valor de ϵ é decrementado um fator de 0.01 entre cada época. Após um passe, o mesmo é configurado para o valor 1

Foi observada uma média de taxa de sucesso ao redor de 0.1 até a época 60 com um desvio máximo superior em 0.25% e inferior em 0.1. Porém, observou-se que após a época 60 houve um aumento linear na taxa de sucesso até uma média próxima a 0.8. Os desvios após a convergência foram de 0.58 e 0.9. Esse comportamento indica que para a quantidade de estados gerados pela representação 3x3 e para essa taxa de aprendizado, o aprendizado por reforço conseguiu aprender uma política que proporciona uma taxa de sucesso de cerca de 80% para o tempo de treinamento disponível.

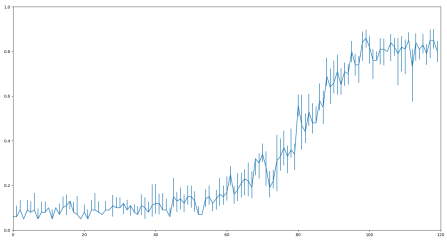

Fig. 6. Evolução do agente utilizando visibilidade I de 9 células e retângulo de observação de 150 pixels.

Em contrapartida, como ilustrado pela Figura 7, utilizando uma visibilidade de 5x5 o agente convergiu apenas após a época 100 e sua taxa de sucesso foi cerca de 0.7 com um desvio superior e inferior próximos de 0.4 e 0.8, respectivamente. A convergência e piora no resultado indica que a quantidade de estados gerados pela representação impactou negativamente na capacidade de reaproveitamento de estados. Com uma generalização prejudicada, tende-se a super especializar e não convergir a um ótimo local ou global.

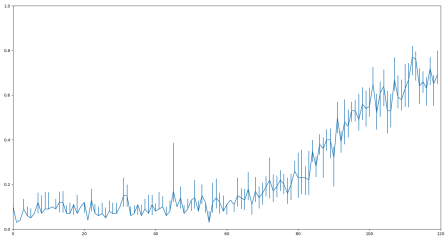

Fig. 7. Evolução do agente utilizando visibilidade I de 25 células e retângulo de observação de 150 pixels.

Por fim, a Figura 8 ilustra o desempenho do agente para uma visibilidade de 11x11 células. Não foi observada convergência para a quantidade de testes realizados, porém isso não indica que o algoritmo não seja capaz de convergir em

algum momento. Para esse teste, a quantidade de estados chega a ultrapassar 16000, o que torna muito difícil a capacidade de reaproveitar estados para situações semelhantes e consequentemente convergir a um ótimo local ou global.

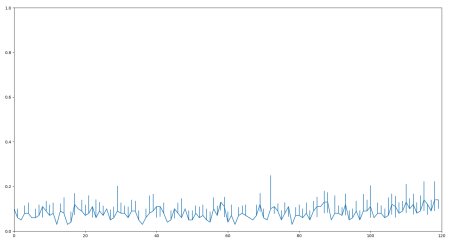

Fig. 8. Evolução do agente utilizando visibilidade I de 121 células e retângulo de observação de 150 pixels.

4.3 Conclusion

Neste projeto foi desenvolvido e avaliado um simulador para controle de múltiplos agentes, assim como um modelo de pastoreio baseado em regras e também por aprendizado. O simulador desenvolvido foi avaliado no contexto de múltiplos agentes. Já o agente que baseia-se em aprendizado foi avaliado utilizando um modelo de agente único e com auxílio do q-learning.

Através dos resultados foi possível observar que a implementação desta técnica mostra a viabilidade do simulador desenvolvido para controlar diversos agentes, simular episódios, épocas, e passes, e permitir não somente a visualização em tempo real dos pastores e ovelhas mas também do estado observado para cada um deles simultaneamente. Além disso, foi possível validar o funcionamento do q-learning aplicado ao contexto de pastoreio com diferentes representações para o ambiente. Os resultados sugerem a corretude da implementação, apesar de que a utilização de uma q-table dinâmica prejudicou a qualidade do aprendizado.

Com a realização deste projeto foram observadas as diversas dificuldades do pastoreio, principalmente o fato de que um agente que visa aprender uma política deve ser capaz de associar o impacto de suas ações em outros agentes. Como trabalhos futuros pretende-se melhorar o modelo de memória utilizado, incorporar cooperação entre diferentes tipos de pastores, e avaliar novas estratégias de pastoreio.

REFERENCES

[1] *Shepherding with robots that do not compute*, ser. ALIFE 2020: The 2020 Conference on Artificial Life, vol. ECAL 2017, the Fourteenth European Conference on Artificial Life, 09 2017. [Online]. Available: https://doi.org/10.1162/isal_a_056

[2] J. Zhi and J.-M. Lien, "Learning to herd agents amongst obstacles: Training robust shepherding behaviors using deep reinforcement learning," 2020.

[3] G. Sartoretti, Y. Wu, W. Paivine, T. Kumar, S. Koenig, H. Choset, Y. Wu, and T. Kumar, "Distributed reinforcement learning for multi-robot decentralized collective construction," 10 2018.

[4] B. Wang, Z. Liu, Q. Li, and A. Prorok, "Mobile robot path planning in dynamic environments through globally guided reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6932–6939, 2020.

[5] G. A. Cardona, C. Bravo, W. Quesada, D. Ruiz, M. Obeng, X. Wu, and J. M. Calderon, "Autonomous navigation for exploration of unknown environments and collision avoidance in mobile robots using reinforcement learning," in *2019 SoutheastCon*, 2019, pp. 1–7.

[6] X. Qiu, K. Wan, and F. Li, "Autonomous robot navigation in dynamic environment using deep reinforcement learning," in *2019 IEEE 2nd International Conference on Automation, Electronics and Electrical Engineering (AUTEEE)*, 2019, pp. 338–342.

[7] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '87. New York, NY, USA: Association for Computing Machinery, 1987, p. 25–34. [Online]. Available: https://doi.org/10.1145/37401.37406