DESARROLLO DE UN ENTORNO DE SIMULACIÓN PARA EL APRENDIZAJE DE UN ALGORITMO DE NAVEGACIÓN AUTÓNOMA DE UN VELERO DE 2 METROS DE ESLORA

Esteban Chacón Mosquera
Universitat Politècnica de Catalunya (UPC), Jordi Girona 31, 08034 Barcelona, estebanfex@gmail.com

Josep Eudald Mesegué Basallo
Facultat de Nàutica de Barcelona (UPC), Pla de Palau 18, 08003 Barcelona, eudald.mesegue@gmail.com

Rosa M. Fernández-Cantí
Facultat de Nàutica de Barcelona (UPC), Pla de Palau 18, 08003 Barcelona, rosa.mari.fernandez@upc.edu

José A. Lázaro Villa
Universitat Politècnica de Catalunya (UPC), Jordi Girona 31, 08034 Barcelona, jose.lazaro@tsc.upc.edu

Resumen

Se presenta el desarrollo de un entorno virtual de simulación basado en Python Turtle para el entrenamiento de un algoritmo de aprendizaje por refuerzo destinado a la navegación autónoma de un velero de 2 metros de eslora. Este entorno de simulación permite entrenar el pilotaje autónomo en diferentes condiciones de viento y datos de navegación de la embarcación, en ausencia de obstáculos, por medio de la observación causa-efecto y una estrategia de recompensas que permiten al agente decidir las mejores acciones. La generación virtual de situaciones de navegación reduce las horas de pruebas de mar.

Palabras clave: Navegación autónoma, velero de 2m de eslora, entorno de simulación, aprendizaje por refuerzo, Python Turtle

1 INTRODUCCIÓN

Dentro del ámbito de la navegación autónoma, uno de los puntos críticos es el algoritmo de navegación, encargado de tomar las decisiones de pilotaje en función de la información que le llega de los diferentes sensores instalados en la embarcación [2].

Un buen algoritmo de navegación autónoma debe ser capaz de llevar la embarcación a su punto de destino tomando la ruta más adecuada, optimizando el consumo energético y tiempo de travesía, y garantizando en todo momento la seguridad de la navegación. Todo ello debe hacerse, además, sin intervención humana externa.

En los últimos años se han producido numerosos avances en el campo de los vehículos autónomos, tanto en el ámbito terrestre como en los ámbitos aéreo y náutico [3], [4]. El entorno marino presenta sus propias particularidades y, en concreto, la maniobra autónoma de veleros presenta dificultades y retos añadidos como son: la interacción entre la posición de la vela y los timones, la influencia de la escora, las maniobras específicas como pueden ser las ceñidas con viento de proa y la menor capacidad de maniobra en comparación con otro tipo de embarcaciones, lo que puede dificultar la evitación de obstáculos, especialmente en situación de vientos débiles. La Figura 1 muestra el velero autónomo Sensailor, el cual ha sido diseñado y construido para estudiar estos temas [1], [6].

Figura 1: Velero Sensailor

En relación a los algoritmos en sí, recientemente se están ensayando muchos enfoques basados en la inteligencia artificial, tanto supervisados como no supervisados [7]. Este trabajo se centra en el algoritmo de aprendizaje por refuerzo, no supervisado, implementado en el velero autónomo Sensailor.

Como todos los algoritmos de este tipo, la fase de entrenamiento es crítica ya que el correcto funcionamiento del algoritmo, una vez puesto en servicio, depende fuertemente del entrenamiento y de la calidad y cantidad de los datos utilizados en este. El entrenamiento debe realizarse con un número suficiente de datos y debe cubrir satisfactoriamente el máximo de situaciones posibles. Ello implica someter repetidamente a la embarcación a diferentes condiciones de navegación. Puesto que en la mayoría de situaciones ello no es viable, porque las pruebas de mar tienen una duración limitada y porque tampoco tenemos control sobre las condiciones de navegación, es necesario contar con herramientas de simulación.

En este artículo se describe el desarrollo de un entorno de simulación basado en Python Turtle [5] para el entrenamiento de algoritmos de aprendizaje no supervisado por refuerzo. Su aplicación al algoritmo de navegación del Sensailor demuestra que este entorno es una herramienta útil para la fase de aprendizaje ya que reduce el en tiempo de pruebas y puede presentar variedad de situaciones.

El resto del documento está organizado como sigue: en la Sección 2 se presentan las principales características del velero autónomo Sensailor incluyendo la propulsión y la instrumentación. La Sección 3 presenta brevemente el algoritmo de aprendizaje por refuerzo implementado en el velero. La Sección 4 presenta el entorno de simulación desarrollado y se comentan sus principales características. Finalmente, en la Sección 5 se presentan las conclusiones y líneas futuras de investigación.

2 AUTOMATIZACIÓN DEL VELERO

2.1 CONSTRUCCIÓN

El casco del velero Sensailor, así como sus apéndices, se han diseñado y construido para satisfacer las especificaciones propias de una embarcación autónoma destinada a la recogida de datos oceanográficos, destacando la autonomía, la capacidad de recuperar la verticalidad en caso de vuelco y la protección de los circuitos y componentes eléctricos y electrónicos.

Para la construcción del casco se ha creado un molde con espuma de poliuretano, a partir del cual se ha laminado el casco. Para el laminado se ha usado resina de poliéster y fibra de vidrio. Los detalles del diseño y la construcción, incluyendo los cálculos y las simulaciones de estabilidad, se pueden consultar en [6].

La Tabla 1 muestra las principales dimensiones del velero.

Tabla 1: Dimensiones del Sensailor

Nombre	**Descripción**
Eslora L_H	2 m
Eslora flotación L_{WL}	1.873 m
Manga B_H	0.71 m
Desplazamiento Δ	50 kg

2.1.1 Propulsión: vela y alerón

Con el objetivo de mejorar la autonomía y aprovechamiento energético, el tipo de propulsión escogido para esta embarcación es una vela. Para tal propósito, se ha implementado una configuración de vela rígida, orientada mediante un alerón. Dicha vela puede girar libremente sobre el mástil, sin ningún elemento de fijación mecánica, pero sin lograr giros de 360º en su configuración original [6]. La sección transversal de la vela y el alerón corresponden a un perfil aerodinámico de la base de datos NACA, en concreto el perfil simétrico NACA 0015.

Este tipo de configuración logra orientar la vela para conseguir los vientos más propicios para el avance, minimizando el consumo energético para mantener la vela en esta posición. El ángulo del alerón varía mediante un servomotor, y el propio alerón modifica el ángulo de la vela, ahorrando un servomotor de mayores dimensiones que mantenga fijo el ángulo de la vela actuando directamente sobre esta.

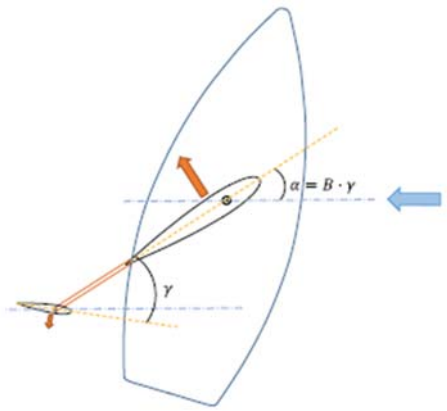

Figura 2: Fuerzas generadas en la vela y el alerón [6]

2.2 INSTRUMENTACIÓN

La embarcación dispone de dos tipos de instrumentación: la destinada a permitir la navegación y la que recopila datos oceanográficos para estudios científicos.

Una placa Arduino UNO recoge la información de los sensores que operan a 5V: tanto los destinados a la navegación autónoma (GPS, anemómetro, veleta, magnetómetro), como los de recogida de datos (temperatura, rayos UV). También recoge los datos de un sensor LDR (luminosidad) y enciende la luz de posición cuando la medida del sensor LDR es inferior a un valor umbral.

La información de los sensores conectados al Arduino se transmite a una placa Raspberry Pi mediante una conexión USB. Adicionalmente, esta placa también recibe información de una cámara NoIR, para visión nocturna, y de un sensor LIDAR, para la detección de obstáculos.

La placa Raspberry Pi, además, es la encargada de controlar el ángulo de los timones y del servo del alerón de la vela. Para el control del alerón, se ha añadido un microcontrolador ESP32 en el interior de la vela, que se comunica con la Raspberry Pi mediante Bluetooth, ahorrando una conexión física del interior del casco a la vela.

La Figura 3 muestra la estructura de la instrumentación.

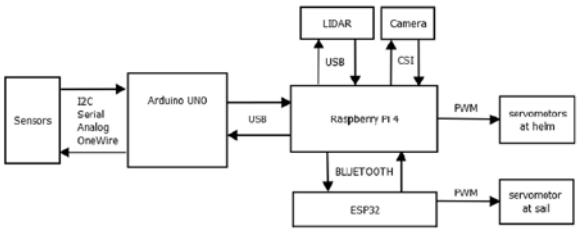

Figura 3: Organización de la instrumentación [1]

2.3 MANIOBRA

En la maniobra intervienen dos elementos: los timones y la vela. Los timones marcan el rumbo de la embarcación, y la vela es la encargada de la propulsión. Por lo que atañe a los timones, su funcionamiento es relativamente sencillo: su orientación respecto a la línea de crujía (longitudinal) marca el cambio en el rumbo. La vela, sin embargo, precisa de viento en unas direcciones concretas para generar propulsión.

Como en todas las embarcaciones a vela, la navegación se verá condicionada a que se reciba el viento en el rango de ángulos útiles para la propulsión (p.ej., no se puede navegar contra el viento). Cada embarcación tiene sus ángulos útiles de viento. La Figura 4 muestra los rangos útiles para la propulsión de nuestra embarcación:

Figura 4: Rango de ángulos útiles [6]

Por otro lado, en el caso concreto de nuestro velero, existe otra restricción: la vela no puede girar 360º [6]. Por lo tanto, la virada es siempre por proa.

La necesidad de recibir vientos que ayuden a la propulsión, junto con la limitación a la hora de hacer las viradas, son factores críticos que se deben tener en cuenta en la programación del algoritmo de navegación del velero. En la Figura 5 se muestra un esquema del recorrido navegado con 4 *waypoints* y una condición de viento determinada, donde se muestra la posición de la vela en cada situación.

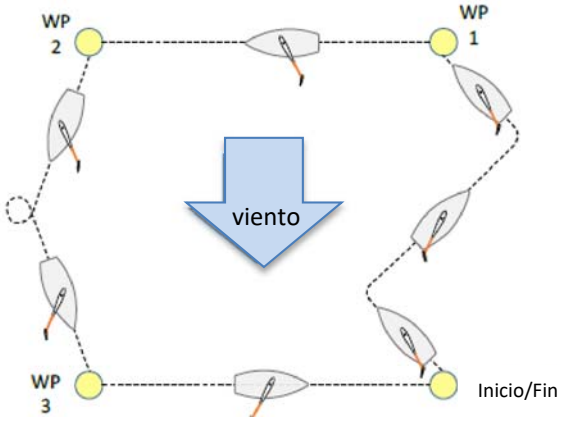

Figura 5: Posición de la vela según el rumbo deseado y dirección del viento

3 ALGORITMO DE APRENDIZAJE POR REFUERZO

El algoritmo escogido está basado en el DQN (Deep Q-Network) con descenso de gradiente estocástico desarrollado por el MIT [7].

En este algoritmo de aprendizaje por refuerzo se incluye un agente que interactúa con su ambiente y observa la respuesta del mismo a sus acciones. Según sea la respuesta, el agente recibe una recompensa o una penalización. De esta manera, el agente buscará elegir respuestas que maximicen la recompensa.

3.1 DISEÑO DEL AGENTE

El agente elegido es una red neuronal convolucional (CNN) consistente en 6 entradas, 2 capas ocultas de 64 neuronas cada una y 5 salidas. Todas las neuronas están interconectadas entre sí con lo que la red neuronal profunda cuenta con 4933 parámetros a entrenar.

En una primera aproximación al problema se supone que no hay obstáculos, por lo que el algoritmo solo se centrará en llevar la embarcación de un punto a otro en diversas condiciones de viento, de posición inicial de los actuadores de la embarcación, y de orientación de la embarcación.

Por lo tanto, las entradas a la red son la dirección del viento, los ángulos del alerón de la vela y de los timones y el rumbo de la embarcación, así como la distancia a las coordenadas de destino. La salida de la red son las señales de mando a los servomotores de los timones y alerón de vela (ver Figura 6).

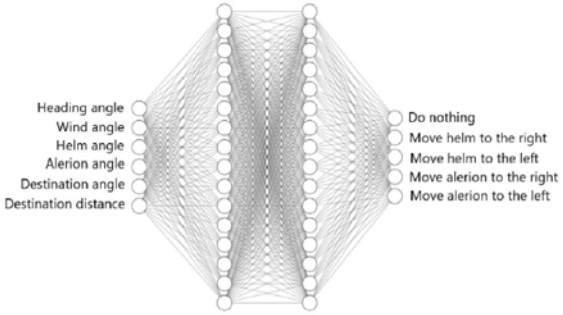

Figura 6: Entradas y salidas de la red neuronal

3.2 ENTRENAMIENTO

El agente se entrena interaccionando con su ambiente, del cual puede aprender a partir de las observaciones y recompensas/penalizaciones.

3.2.1 Ambiente de entrenamiento

En este trabajo, el ambiente es generado virtualmente y visualizado con ayuda de la herramienta Turtle codificada en Python [5] (ver Sección 4). Este entorno virtual genera aleatoriamente diferentes situaciones (de condiciones de viento y estado inicial de la embarcación) de manera que, al recibir las acciones del agente sobre los servomotores de timones y alerón, modifica y muestra el rumbo y la velocidad de la embarcación según un modelo interno.

El estado observado, tanto del viento como de la embarcación, se introduce de nuevo en el agente junto con la distancia al punto de destino. Una política decide si la acción previamente tomada recibe una recompensa o penalización, a fin de que el agente pueda aprender al buscar maximizar la recompensa después de cada interacción. Ver Figura 7.

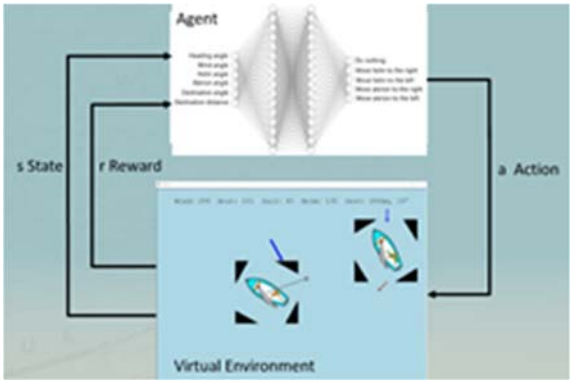

Figura 7: Entrenamiento del agente

El objetivo de cada episodio de entrenamiento es conseguir que la embarcación se oriente conforme al rumbo deseado para llegar al punto de destino y la vela tome un ángulo adecuado para hacer avanzar la embarcación en dicho rumbo, excepto en el caso en que el viento sea de proa, en cuyo caso habrá que realizar ceñidas.

Cada episodio de entreno se considera finalizado cuando la distancia al destino se ha reducido en, al menos, un metro. Durante un episodio de entreno se permite un máximo de 3000 pasos, donde cada paso es la acción de mover un timón cierto ángulo. Si el destino no se alcanza al llegar a este límite, el ambiente se resetea y el modelo no se actualiza. Adicionalmente, la cantidad de pasos (movimientos del timón) para llegar al destino se toma como penalización al resultado de cada episodio a fin de evitar que el modelo entrenado mueva los timones de manera frenética buscando la recompensa.

Para maximizar la recompensa cuando la respuesta es la esperada, la red neuronal busca resolver la ecuación de Bellman

$$Q^*(s,a) = E_{s'\sim\varepsilon}\left[r + \gamma \max Q^*(s',a') \middle| s,a\right] \quad (1)$$

donde $Q^*(s,a)$ es el valor máximo alcanzable siguiendo una determinada política después de las secuencias de estado s' y acciones a' para todas las posibles acciones a, r es la recompensa, y γ es un factor que reduce su valor en cada paso y permite

optimizar el número de acciones requeridas para alcanzar la meta deseada.

3.2.2 Políticas de entrenamiento

Para entrenar a la red neuronal se siguen las siguientes políticas:

- La recompensa se incrementa de manera inversamente proporcional a la diferencia entre el rumbo actual y el rumbo destino. Además, se da una recompensa adicional cuando la diferencia de ángulos es menor a 5%.

- La recompensa se incrementa de manera inversamente proporcional a la diferencia de distancia al destino entre un tiempo previo y el tiempo actual, penalizando cuando, por el contrario, la distancia al destino se incrementa. Y se da una recompensa adicional de alta ponderación al llegar al destino.

- Se da una recompensa adicional que es directamente proporcional a la velocidad de la embarcación.

Con las tres políticas mencionadas se busca enrumbar la embarcación y mover la vela para llegar a su destino en el menor tiempo posible.

4 DESARROLLO DEL ENTORNO DE SIMULACIÓN

La Figura 8 muestra la pantalla de visualización del ambiente virtual. Si se desea una mayor velocidad de entrenamiento, también es posible realizar el entrenamiento sin la visualización. Adicionalmente, este entorno de entreno puede usarse también en condiciones reales de navegación con lo que las medidas de rumbo, dirección del viento, etc., vendrán directamente de los instrumentos de navegación.

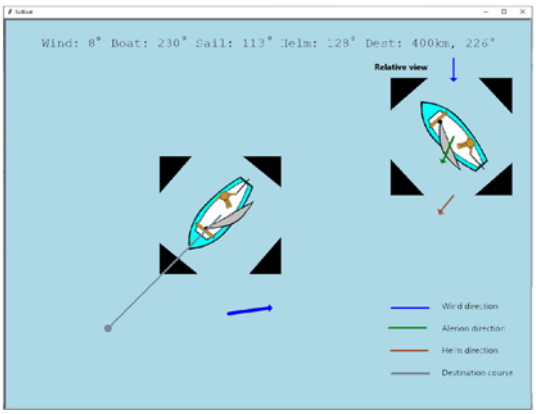

Figura 8: Entrenamiento con el entorno Turtle

La embarcación en el centro muestra el rumbo actual suministrado por el magnetómetro. La rotación de la imagen se realiza dentro de un cuadrado negro para que sea más fácil visualizar la orientación. La flecha azul muestra la dirección del viento aparente medida con la veleta. La línea gris muestra el rumbo a las coordenadas de destino. Puesto que la distancia al punto de destino puede exceder los límites de la pantalla, la travesía se divide en secciones de 10 metros (correspondientes a 40 píxeles).

La embarcación en el ángulo superior derecho muestra la posición relativa de manera que el viento (flecha azul) se sitúa en 270º. En esta vista, el ángulo del alerón (no de la vela) se muestra con la flecha verde y está físicamente limitado de 45º a 135º. Por su parte, el ángulo de los timones se representa con una flecha marrón, donde también se tiene en cuenta la limitación física de 45º a 135º, debida a los servomotores utilizados.

Finalmente, en la parte superior se muestran los valores numéricos de todos estos parámetros y la pantalla muestra el movimiento de todos los actuadores y rumbo. La embarcación, además, se mueve hacia delante cuando se reduce la distancia al destino.

Puesto que las redes neuronales profundas son cajas negras para el entorno, es difícil valorar cómo responde la embarcación a los comandos. Para ello se ha incluido también una interfaz con el usuario (ver Figura 9) que se puede usar para controlar tanto el velero virtual como el real. Esta interfaz ayuda a comprender cómo la recompensa o penalización de ciertas acciones puede agilizar el aprendizaje y se ha utilizado para definir las políticas de entrenamiento ya mencionadas.

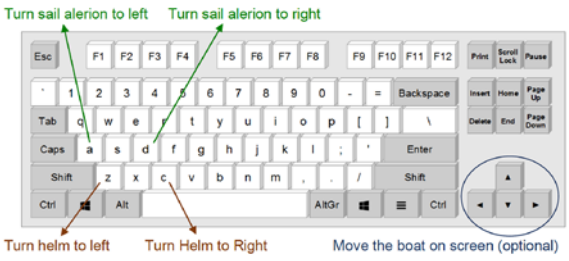

Figura 9: Interfaz con el usuario

En lo que respecta a los resultados, para este proyecto se han configurado 100 episodios de entrenamiento de 3000 pasos cada uno para que el velero logre enrumbarse al destino. La Figura 10 muestra la recompensa obtenida después de cada episodio de entrenamiento, donde los picos corresponden al caso en que ha llegado al destino y los valles corresponden al caso en que se completaron los 3000 pasos sin llegar

al destino. La reducción del número de valles a medida que se suceden los episodios, muestra cómo el modelo va aprendiendo la mejor manera de llegar al destino.

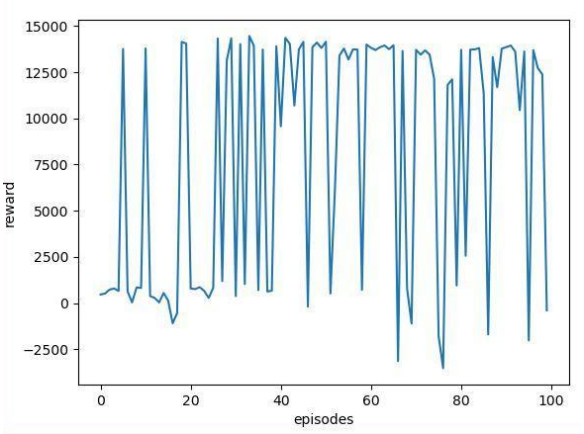

Figura 10: Resultados del entrenamiento

5 CONCLUSIONES

Los algoritmos de navegación autónoma basados en aprendizaje reforzado necesitan gran cantidad de datos en su fase de entrenamiento. Este entrenamiento es difícil de realizar únicamente por medio de pruebas de mar puesto que se requiere gran número de horas y diferentes condiciones de navegación para que el aprendizaje sea significativo.

El entorno de entrenamiento presentado en este trabajo permite entrenar al agente del algoritmo de aprendizaje por refuerzo tanto con datos reales (recogidos a partir de los sensores de la embarcación) como con datos virtuales (generados por ordenador).

En el caso virtual objeto de este trabajo, se ha comprobado que con las tres políticas de entrenamiento detalladas el modelo de aprendizaje es capaz de enrumbar el bote y tomar decisiones que lo acerquen a su destino.

Como líneas futuras, se consideran la incorporación de la esquiva de obstáculos en el algoritmo a partir de las lecturas del LIDAR y la comparación de las decisiones tomadas por el algoritmo con las decisiones que tomaría un piloto humano.

Agradecimientos

Los autores agradecen a la Facultat de Náutica de Barcelona la ayuda económica para la realización del proyecto Sensailor.

English summary

DEVELOPMENT OF A SIMULATION ENVIRONMENT FOR THE LEARNING OF AN AUTONOMOUS NAVIGATION ALGORITHM FOR A 2-METER LENGTH SAILBOAT

Abstract

The development of a virtual simulation environment based on Python Turtle is presented for the training of a reinforcement learning algorithm for the autonomous navigation of a 2 meters long sailboat. This simulation environment enables autonomous piloting to be trained in different wind conditions and navigation data from the vessel, in the absence of obstacles, through cause-effect observation and a reward strategy that allows the agent to decide the best actions. The virtual generation of navigation situations reduces the hours of sea trials.

Keywords: Autonomous navigation, 2m sailboat, simulation environment, reinforcement learning, Python Turtle.

Referencias

[1] Chacón, E. (2021) *Autonomous Sailboat Prototype Sensors and Electronics Implementation with Machine Learning for Navigation*, Master Thesis, Escola Tècnica Superior d'Enginyeria de Telecomunicacions de Barcelona, Universitat Politècnica de Catalunya.

[2] Kulbiej E., P. Wołejsza (2017) "Naval Artificial Intelligence", *The International Conference on Marine Navigation and Safety of Sea Transportation* (TRANSNAV 2017), Maritime University of Szczecin, Szczecin, Poland.

[3] Liu Z., Zhang Y., Yu, X., Yuan C. (2016) "Unmanned surface vehicles: An overview of developments and challenges", *Annual Reviews in Control*, vol. 41.

[4] Martin B., Tarraf, D.C., Whitmore, T.C., DeWeese, J., Kenney C., Schmid J., DeLuca, P. (2019) *Advancing Autonomous Systems. An Analysis of Current and Future Technology for Unmanned Maritime Vehicles*, Report RAND Corporation, Santa Monica, US.

[5] Python.org. Disponible en: https://docs.python.org/3/faq/general.html#general-information

[6] Sastre, J., Manich, C., (2020) *Disseny i construcció d'un dron de navegació automàtica a vela*. Trabajo Final de Grado, Facultat de Nàutica de Barcelona, Universitat Politècnica de Catalunya.

[7] Sutton R.S. and Andrew G. Barto (2017) *Reinforcement Learning: An Introduction* 1st ed. The MIT Press, Cambridge, MA, US.

.