
Escuela de Ingeniería y Gestión

# Aprendizaje por Refuerzo con Opciones y Función de refuerzo Universal

**Autores:** Diego Bruno Cilla (57028)
Matías Heimann (57503)
Giuliano Scaglioni (57244)

**Tutor:** Juan Miguel Santos

Trabajo final para la obtención del título de ingeniero en informática

Buenos Aires, Diciembre de 2020

**Resumen**

El aprendizaje por refuerzo permite a un agente (por ejemplo, un robot) aprender distintos comportamientos. Un comportamiento o política es una asociación entre estados y acciones. Durante el aprendizaje, el agente explora los estados posibles. En cada estado selecciona una acción recibiendo un valor numérico que representa una recompensa o castigo por haber elegido esa acción en ese estado específico. Este valor numérico es devuelto por una función de refuerzo que recibe 3 parámetros: el estado previo a la acción, la acción tomada y el nuevo estado.

El objetivo de los algoritmos de aprendizaje por refuerzo es maximizar las recompensas acumuladas a lo largo del tiempo para hallar un comportamiento objetivo. De esta forma, para aprender distintos comportamientos, la variable a cambiar sería la función de refuerzo dada para ese problema.

El objetivo de este trabajo es explorar una alternativa en la cual se puedan adquirir distintos comportamientos, manteniendo siempre la misma función de refuerzo, siendo que la variable sea los distintos entornos en los que se realiza el aprendizaje.

Para lograr este objetivo, se nos introdujo inicialmente en al algoritmo de Q-Learning, un algoritmo de aprendizaje por refuerzo el cual no requiere de un modelo de entorno y puede manejar problemas de transiciones estocásticas y recompensas sin requerir adaptaciones.

Una vez ya introducido Q-Learning, se buscaron métodos para optimizar el aprendizaje del agente, la solución elegida fue el uso de Opciones. Las opciones brindan un nivel de abstracción mayor al problema al poder subdividirlo en problemas más pequeños los cuales son más fácil de resolver con el algoritmo de Q-Learning.

# Índice

# 1. Introducción

El Aprendizaje Automático es una rama del campo de la Inteligencia Artificial. Dentro de este campo se tiene como objetivo la investigación y aplicación de algoritmos con el fin de que las computadoras aprendan. Estos algoritmos van mejorando mediante la experiencia. Dentro del Aprendizaje Automático podemos encontrar tres categorías.

- Aprendizaje Supervisado: el algoritmo establece una correspondencia entre las entradas y las salidas deseadas del sistema.
- Aprendizaje No Supervisado: solamente se utilizan las entradas para el proceso de modelado.
- Aprendizaje por Refuerzo: hay un agente el cual interactúa con un ambiente dinámico. A medida que el agente interactúa con el ambiente este recibirá una recompensa o castigo en función de lo realizado. Se puede decir que el agente irá aprendiendo con prueba y error.

Este informe estará centrado en el tercer tipo de aprendizaje mencionado, el Aprendizaje por Refuerzo, específicamente orientado a un algoritmo en particular llamado Q-Learning.

El trabajo está estructurado de una forma tal que inicialmente se hará una introducción al aprendizaje por refuerzo donde se explicarán algunos fundamentos importantes los cuales se deben tener en cuenta para poder comprender en mayor profundidad el informe. Los fundamentos serán matemáticos, como conceptos específicos del aprendizaje por refuerzo y se ampliará sobre el algoritmo en cuestión.

Luego, se desarrollará sobre el encuadre de opciones. El encuadre de opciones hará énfasis en la mejora principal aplicadas al algoritmo previamente explicado de Q-Learning. Se explicará qué son y cómo mejoran el algoritmo cambiando lo que antes era un Proceso de Markov a un Semi Proceso de Markov.

En la siguiente sección se explicarán los distintos conceptos y componentes que forman parte de la implementación. También se detallarán los problemas y dificultades encontradas durante el desarrollo de la misma así como las soluciones propuestas.

# 2. Introducción al aprendizaje por refuerzo

Aprendizaje por refuerzo es aprender qué hacer, es decir, cómo asociar acciones a estados, con el objetivo de maximizar una señal de refuerzo numérica. Por otro lado, al agente no se le indica qué acciones realizar, sino que debe explorar probando qué acciones llevan a obtener más recompensa (Sutton, 2018).

Las acciones tomadas por el agente afectan no solo la recompensa inmediata, sino también el próximo estado, y por consiguiente, todas las recompensas futuras. El aprendizaje por refuerzo es, en simultáneo, un problema, una clase de soluciones que funcionan bien con el problema y el campo que estudia este problema y los métodos de solución.

El problema del aprendizaje por refuerzo, formalmente, se plantea usando ideas de la teoría de sistemas dinámicos, específicamente, como el control óptimo de procesos de decisión de Markov (MDP). La idea básica es capturar los aspectos más importantes del problema real con un agente interactuando con su ambiente a lo largo del tiempo para llegar a un objetivo. El agente debe poder sensar el estado de su ambiente hasta cierto nivel, ya que no posee conocimiento del estado completo, y debe poder tomar acciones que afecten al mismo. El agente también debe tener uno o más objetivos relacionados con el estado del ambiente. La intención de los MDPs es incluir estos tres aspectos: sensado, acción y objetivo, de la forma más simple posible pero sin trivializar ninguno de ellos.

## 2.1. Exploración vs. Explotación

Uno de los desafíos que aparecen en el aprendizaje por refuerzo es la relación de compromiso entre exploración y explotación. Se puede notar que, para obtener mucha recompensa, el agente debe preferir realizar acciones que ya probó en el pasado y que fueron efectivas en producir recompensa. Pero, por otro lado, para descubrir estas acciones, debe probar acciones que no había seleccionado con anterioridad. De esta forma, el agente debe explotar lo que ya experimentó anteriormente para obtener recompensa, pero también tiene que explorar para poder mejorar las acciones que seleccionará en el futuro. El problema es que ni la exploración, ni la explotación se pueden realizar de forma exclusiva sin fallar en la tarea. El agente debe probar distintas acciones y progresivamente favorecer aquellas que parecen ser las mejores. En una

tarea estocástica, cada acción debe ser probada varias veces para poder obtener una estimación confiable de la recompensa esperada.

## 2.2. Elementos del Aprendizaje por Refuerzo

Existen distintos elementos presentes en el aprendizaje por refuerzo. Anteriormente se mencionó el ambiente y el agente. Existen además otros, la política, la señal de recompensa (o refuerzo), una función de valor y, opcionalmente, un modelo del ambiente.

### 2.2.1. Política

Una política define cómo se comportará un agente en un momento determinado. En líneas generales, una política es una asociación entre los estados que el agente percibe del ambiente y las acciones que debe tomar cuando se encuentra en esos estados. La política es lo más importante de un agente de aprendizaje por refuerzo ya que, por sí sola, es suficiente para determinar el comportamiento. Otro aspecto de las políticas es que estas pueden ser estocásticas. En estos casos, en vez de ser una asociación entre estados y acciones, se asocian estados con probabilidades para cada acción.

### 2.2.2. Señal de recompensa

La señal de recompensa define el objetivo del problema de aprendizaje por refuerzo. En cada instante de tiempo, el ambiente envía al agente de aprendizaje por refuerzo un solo número llamado recompensa. El único objetivo del agente es maximizar la recompensa total recibida a largo plazo.

La señal de recompensa define qué eventos son buenos para el agente y cuáles no. Además produce un efecto directo en la modificación de la política. Si una acción es seleccionada por la política y lleva a una baja recompensa, entonces, la política podría ser cambiada para que seleccione otra acción ante la misma situación en el futuro.

En general, las señales de refuerzo pueden ser funciones estocásticas dependientes del estado del ambiente y de las acciones tomadas.

### 2.2.3. Función de valor

Mientras que la señal de recompensa nos indica que es lo que está bien de forma inmediata, la función de valor específica que es lo que está bien a largo plazo.

En líneas generales, el valor de un estado, nos indica cuánta recompensa puede llegar a acumular el agente en el futuro, empezando desde ese estado. En otras palabras, el valor de un estado tiene en consideración los estados que probablemente pueda llegar a visitar el agente y las recompensas para cada uno de esos estados. Entonces, una acción llevada a cabo desde un estado puede darnos una recompensa baja pero de igual forma tener un valor alto porque los próximos estados que visitará el agente siguiendo la política actual le permitirá acumular recompensas altas.

### 2.2.4. Valor vs. Recompensa

Las recompensas son consideradas primarias, mientras que la función de valor, como predicciones de la recompensas, secundaria. Esto es porque sin recompensas no podría haber valores, y el único propósito de estimarlos es obtener mayores recompensas. Más allá de esto, es la función de valor la que nos importa cuando hacemos o evaluamos decisiones. Elegimos acciones que llevan a estados de mayor valor, y no recompensa, ya que son estas acciones las que nos dan mayor recompensa a largo plazo.

Desafortunadamente, es más difícil determinar los valores de los estados que determinar recompensas. Las recompensas son dadas directamente por el ambiente, mientras que los valores deben ser estimados a partir de la secuencia de observaciones que el agente hace a lo largo de su ciclo de vida. De hecho, el componente más importante de casi todos los algoritmos de aprendizaje por refuerzo que se considera es un método para estimar valores eficientemente.

### 2.2.5. Modelo del ambiente

El modelo es un componente opcional de algunos sistemas de aprendizaje por refuerzo que permite simular el comportamiento del ambiente.

Los modelos se utilizan en la planificación para saber qué curso de acción tomar en base a cómo se va a comportar el ambiente en el futuro sin tener que experimentarlo antes.

## 2.3. Métodos basados en modelo

Los métodos basados en modelos son métodos que utilizan un modelo del ambiente para realizar la planificación. Se contraponen a los métodos libres de modelo en los cuales no se utiliza un modelo y el aprendizaje se realiza básicamente en forma de prueba y error.

## 2.4. Proceso de decisión de Markov finito

Un proceso de decisión de Markov (MDP) es la formalización clásica de decisión secuencial donde las acciones no solo afectan la recompensa inmediata sino que también afectan los estados futuros y por consiguiente las recompensas a largo plazo. Esto hace que tengamos que tener en cuenta el balance entre recompensa inmediata y futura.

## 2.5. Elementos de un MDP

### 2.5.1. Interacción Agente - Entorno

El agente es la entidad que realiza el aprendizaje y toma las decisiones. Por otro lado, el agente interactúa con el ambiente, esto es, todo lo que lo rodea. Esta interacción se da de forma continua, el agente realiza acciones y el ambiente evoluciona a nuevos estados en base a las acciones seleccionadas. Adicionalmente, el ambiente entrega recompensas al agente, las cuales se buscan maximizar a lo largo de su curso de acción.

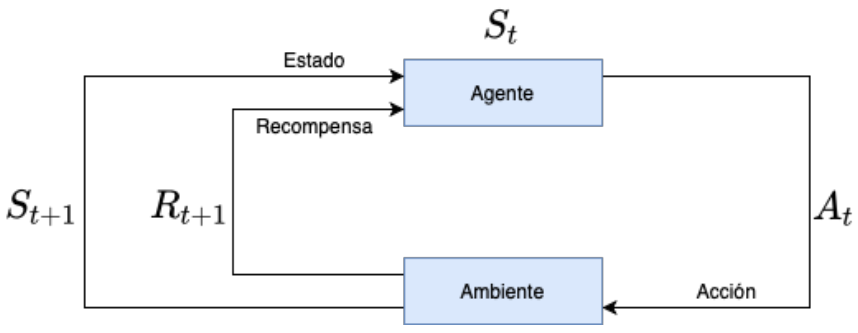



Figura 1: Diagrama de interacción entre agente y ambiente.

Formalmente, el agente y el ambiente interactúan siguiendo una secuencia de pasos discreta, $t = 0, 1, 2, ...$ En cada paso $t$, el agente observa una representación del estado $S_t$ y realiza un

acción $A_t$ en el ambiente. El ambiente, como resultado de la acción del agente, evoluciona a un nuevo estdo $S_{t+1}$ y entrega al agente una recompensa $R_{t+1}$.

En un MDP finito, tanto el conjunto de estados, como el conjunto de acciones son finitos. En este caso, las variables aleatorias $S_{t+1}$ y $R_{t+1}$ son distribuciones de probabilidad discretas solo dependientes del estado y acción inmediatamente anterior. Esto es, para valores particulares de estas variables aleatorias, $s'$ y $r$, existe una probabilidad de que ocurran estos valores en el tiempo $t$, dados valores particulares para el estado $s$ y acción $a$ inmediatamente anterior: $p(s', r|s, a) \equiv Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\}$ para todo $s$, $s'$ en el espacio de estados, para todo $r$ en el espacio de recompensas, para todo $a$ en el espacio de acciones. Esta función de probabilidad $p$ define la dinámica del MDP.

### 2.5.2. Objetivos y recompensas

En el aprendizaje por refuerzo, el objetivo del agente se formaliza en términos de una señal especial llamada recompensa, la cual es pasada desde el ambiente al agente. En cada instante de tiempo, la recompensa es un número $R_t \in \mathbb{R}$. Informalmente, el objetivo del agente es maximizar la suma de recompensas que recibe. Esto no significa maximizar la recompensa inmediata sino la acumulada en el largo plazo.

El uso de la señal de recompensa o refuerzo para formalizar la idea de un objetivo es una de las características más distintivas del aprendizaje por refuerzo. Por ejemplo, consideremos el caso de un agente en un ambiente grilla en el cual hay un punto de inicio, un punto de finalización y varios objetos dispersos por la grilla. Supongamos que la representación del estado que percibe el agente es la distancia polar al punto de finalización y al objeto más cercano. Si el objetivo es que el agente llegue en la menor cantidad de pasos, la función de recompensa va a premiar al agente siempre que disminuya la distancia al punto de finalización. Por otro lado, si el objetivo es que capture la mayor cantidad de objetos, la función de recompensa va a reforzar los casos en que el agente pasó sobre un objeto. Como se puede observar, cambiando la función de recompensa (o refuerzo) logramos cambiar el objetivo del agente.

### 2.5.3. Retorno y episodios

Anteriormente se explicó que lo que se desea maximizar es la recompensa acumulada a lo largo del tiempo. Definiendo esto formalmente, si consideramos una secuencia de recompensas $R_{t+1}, R_{t+2}, R_{t+3}...$, recibida luego del instante de tiempo $t$, entonces lo que queremos maximizar es el retorno esperado, esto es $G_t$, una función de la secuencia de recompensas. El caso más simple del retorno es la suma de la recompensas:

$$G_t \equiv R_{t+1} + R_{t+2} + R_{t+3} + ... + R_T,$$

donde $T$ es el tiempo en que finaliza la secuencia.

Esta definición del retorno tiene sentido en casos donde hay una noción de tiempo final, esto es, cuando la interacción entre el agente y el ambiente se divide en subsecuencias las cuales llamamos episodios. Cada episodio termina en un estado especial llamado estado terminal, a lo que sigue un restablecimiento a un estado de iniciación o una muestra de una distribución de estados de iniciación. Los episodios empiezan independientemente de cómo finalizó el episodio anterior, por ejemplo, en un juego, si se finaliza ganando o perdiendo, esto no afecta al juego siguiente. Las tareas que se pueden subdividir en episodios las llamamos tareas episódicas.

Por otro lado, hay casos en los que la interacción entre el agente y el ambiente no se puede dividir naturalmente en episodios identificables sino que continúan por siempre. Un ejemplo de esto es el caso de evitar obstáculos, no hay un estado final sino que la tarea continua sin límite. A estas tareas las llamamos tareas no episódicas o continuas. En estos casos, la definición del retorno dada anteriormente, se hace problemática porque el tiempo final sería $T = \infty$, y el retorno, que es lo que tratamos de maximizar, podría también ser infinito. Por esta razón, se suele utilizar una definición un poco más compleja pero simple matemáticamente, considerando el concepto de descuento. Esto nos permite ver el problema de una forma diferente. En este caso, el agente trata de elegir acciones mediante las cuales la suma de las recompensas descontadas que recibe a lo largo del futuro sean maximizadas. En particular, elige $A_t$ para maximizar el retorno descontado:

$$G_t \equiv R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$

donde $\gamma$ es un parámetro, $0 \leq \gamma < 1$, llamado factor de descuento.

El factor de descuento permite determinar el valor presente de recompensas futuras. Si el valor de $\gamma$ es 0, entonces el agente sólo buscará maximizar las recompensas inmediatas. A medida que $\gamma$ se acerca a 1, se da mayor importancia a las recompensas futuras. Notar que el caso particular en que $\gamma = 1$ es el caso en donde el retorno es la suma de las recompensas recibidas.

### 2.5.4. Políticas y funciones de valor

Casi todos los algoritmos de aprendizaje por refuerzo involucran estimar funciones de valor, como se explicó anteriormente, una noción de que tan bueno es un estado.

Formalmente una política es una asociación de estados a probabilidades de seleccionar cada acción. Si el agente está siguiendo una política $\pi$ en el tiempo $t$, entonces $\pi(a|s)$ es la probabilidad de que $A_t = a$ si $S_t = s$. Entonces, los algoritmos de aprendizaje por refuerzo indican cómo es que se debe cambiar la política a partir de la experiencia del agente.

La función de valor de un estado $s$ siguiendo una política $\pi$, denotado como $v_\pi(s)$, es el retorno esperado cuando se empieza en $s$ y se sigue la política $\pi$. Para MDPs, podemos definir formalmente a $v_\pi$ como:

$$v_\pi(s) \equiv E_\pi[G_t|S_t = s], \text{ para todo estado } s$$

, donde $E_\pi$ denota el valor esperado de una variable aleatoria dado que se sigue la política $\pi$. Llamamos a esta función $v_\pi$, la función estado-valor para la política $\pi$.

Podemos definir también de forma similar, el valor de tomar la acción $a$ en el estado $s$ siguiendo la política $\pi$, denotado $q_\pi(s, a)$, como el retorno esperado partiendo de $s$, tomando la acción $a$ y luego siguiendo la política $\pi$:

$$q_\pi(s, a) \equiv E_\pi[G_t|S_t = s, A_t = a].$$

Llamamos a esta función $q_\pi$ la función acción-valor para la política $\pi$.

### 2.5.5. Políticas optimales y funciones de valor optimales

Resolver una tareas de aprendizaje por refuerzo es, básicamente, encontrar una política que reciba la mayor cantidad de recompensa en el largo plazo. Para MDPs finitos, podemos

definir una política optimal de la siguiente manera. Las funciones de valor definen un orden parcial sobre las políticas. Una política $\pi$ se define como mejor o igual a una política $\pi'$ si el retorno esperado es mayor o igual que el de $\pi'$ para todos los estados. Siempre hay una política que es mejor o igual que el resto de las políticas, esta es una política optimal. A pesar de que puede haber más de una, denotamos a todas las políticas optimales como $\pi*$. Todas comparten la misma función de estado-valor, llamada la función de estado-valor óptima, definida como:

$$v_*(s) \equiv max_\pi v_\pi(s), \text{ para todo estado } s.$$

Las políticas optimales también comparten la misma función de accion-valor óptima denotada como $q_*$, y definida como:

$$q_*(s,a) \equiv max_\pi q_\pi(s,a).$$

## 2.6. Ecuaciones de Bellman

Una propiedad importante de las funciones de valor es que satisfacen una relación recursiva. Para cualquier política $\pi$ y estado $s$ se cumple la siguiente condición de consistencia entre el valor de $s$ y el valor de los posibles estados que le suceden:

$$\begin{aligned} v_\pi(s) &\equiv E_\pi[G_t|S_t = s] \\ &= E_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \end{aligned}$$

La última ecuación, es la ecuación de Bellman para $v_\pi$.

## 2.7. Q-Learning

El algoritmo Q-Learning es un algoritmo de aprendizaje por refuerzo que permite encontrar una aproximación de la función de valor óptima $q_*$. Este método es de tipo off-policy, es decir, aproxima la función de valor óptima independientemente de la política que se siga. La razón de esto es que, si bien la política determina que pares de estado-acción son los que se actualizan, para que $Q$ converga a $q_*$ solo es necesario que se visiten todos los pares. Por esta última razón es que es importante el balance entre exploración y explotación.

El algoritmo aprende una función $Q$ que, como se mencionó anteriormente, aproxima $q_*$. La regla de actualización de $Q$ es:

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \mathbf{max}_a Q(S_{t+1}, a) - Q(S_t, A_t)],$$

donde $\alpha$ es un parámetro que determina el factor de aprendizaje. Analizando la regla de actualización, se ve que lo que se está haciendo es desplazar el valor de $Q(S_t, A_t)$ al valor del mejor par estado-acción sucesivo recibiendo $R_{t+1}$ de recompensa. El parámetro $\alpha$, o factor de aprendizaje, justamente determina con qué magnitud se realiza esta corrección.

## 2.8. Pseudocódigo del algoritmo

A continuación se detalla el algoritmo de aprendizaje por Q-Learning en pseudocódigo.

**Algoritmo 1** Q-Learning

```
while e < max_episodes do
    s ← InicializarEstado
    while s no es terminal do
        π ← DerivarPolítica(Q)
        a ← ObtenerAcción(π, s)
        s', r ← RealizarAcción(a)
        Q(s, a) ← Q(s, a) + α[r + γmax_a Q(s', a) − Q(s, a)]
    end while
    e ← e + 1
end while
```

donde maxEpisodes es la cantidad de episodios de aprendizaje, Q es la función a aprender, $\alpha$ es el factor de aprendizaje y $\gamma$ es el factor de descuento.

## 2.9. Problema del aliasing perceptual

Existe una dificultad o problema al momento de realizar el aprendizaje. Como el agente no percibe el estado completo del ambiente, sino que observa parcialmente al mismo, existe lo que se denomina *aliasing perceptual*. Lo que sucede es que puede existir más de un estado del

ambiente diferente, que, para la percepción del agente, son idénticos. El problema surge cuando el agente debería aprender cursos de acción diferentes entre los estados que confunde lo cual no es posible porque no los puede distinguir.

Una solución a esta problemática es la modificación del sensado del agente. Si bien va seguir existiendo aliasing perceptual, es importante eliminar la confusión entre los estados que pueden traer dificultad al momento del aprendizaje.

Un ejemplo de esta situación podría ser un agente que debe llegar a un objetivo en una habitación con obstáculos. Si el mismo solo tiene información de la dirección en que se encuentra el objetivo, los estados en donde hay un obstáculo delante del agente son confundidos con los estados en donde el camino está libre. Esto hace que el agente aprenda a avanzar en dirección al objetivo, pero habría casos indistinguibles, donde esa acción hace que el agente no llegue nunca ya que tiene un obstáculo delante. En este caso, se podría solucionar este problema agregando un sensor de proximidad que permita distinguir cuando hay un obstáculo delante y cuando no. De este modo, para un estado se aprende que avanzar es bueno y para el otro, cuando hay un obstáculo, rodearlo es mejor.

# 3. Encuadre de Opciones

. En esta sección se explicará sobre el encuadre de Opciones (Sutton, Precup, y Singh, 1999), se va a describir qué son, cómo funcionan y cómo se aplican en el algoritmo de Q-Learning.

El término de opciones es una generalización de acciones primitivas para incluir temporalmente cursos de acción extendidos. Las opciones tienen 3 componentes básicos, la política $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$, una condición de terminación $\beta : \mathcal{S} \mapsto [0, 1]$, y un conjunto de iniciación $\mathcal{I}$, tal que $\mathcal{I} \subseteq \mathcal{S}$. Una opción está disponible en un estado específico si y sólo si el estado pertenece al conjunto de iniciación de esta. Si la opción fue elegida, entonces las acciones elegidas irán acorde a la política de la opción hasta que esta finalice estocásticamente según su condición de finalización. En particular, si la opción elegida en el estado $s_t$ es de Markov, luego la acción $a_t$ es elegida en función de la política. Luego de ejecutarse la acción el ambiente realiza una transición al estado $s_{t+1}$ donde la opción puede terminar o realizar otra acción $a_{t+1}$ la cual será definida por la política, y continuará así hasta que la opción termine. Cuando la opción termine, el agente tiene la posibilidad de elegir otra opción.

El conjunto de iniciación y la condición de terminación de una opción restringen conjuntamente su rango de aplicación de una manera potencialmente útil. En particular, limitan el rango sobre el cual la política de la opción se debe tener en cuenta. Para opciones de Markov es común asumir que todos los estados donde la opción puede continuar son estados donde la opción puede ser elegida. En este caso la política de la opción solamente precisa ser definida en $\mathcal{I}$ y no en todos los estados de $\mathcal{S}$.

A veces es útil que las opciones tengan un tiempo de expiración, es decir, que finalicen luego de un periodo de tiempo determinado incluso aunque no hayan llegado a alguna condición de terminación específica. Esto no es posible con opciones de Markov debido a que la finalización de estas depende solamente del estado actual del agente, y no desde hace cuanto la opción se está ejecutando. Para tratar con estos casos se considera una generalización a opciones de Semi-Markov, en las cuales, sus políticas y condiciones de terminación pueden depender de eventos previos desde que se inició la opción. En general, una opción es iniciada en un tiempo $t$, determina las acciones seleccionadas por un número de pasos $k$, y después termina en un estado $s_{t+k}$. Para cada tiempo intermedio $T$, tal que $t < T < t + k$, las decisiones de la opción

de Markov dependen solamente de $s_T$, donde las decisiones de Semi-Markov pueden depender de toda la secuencia $s_t, s_{t+1}, ..., s_{t+k}$ , pero no de los eventos previos a $s_t$. A esto se le llamará secuencia de historia desde $t$ hasta $T$ y se lo denotará como $h_{tT}$. Se denota al conjunto de historias como $\Omega$. En las opciones de Semi-Markov, la política y la condición de terminación son funciones de posibles historias, en este caso serían, $\pi : \Omega \times \mathcal{A} \mapsto [0, 1]$ y $\beta : \Omega \mapsto [0, 1]$. Las opciones de Semi-Markov resultan útiles cuando la representación del estado es mucho más detallada que la que está disponible para la política que selecciona las opciones.

Dado un conjunto de opciones, su conjunto de iniciación implícitamente define un conjunto de opciones disponibles $\mathcal{O}_s$ para cada estado $s \in \mathcal{S}$. Este $\mathcal{O}_s$ es equivalente al conjunto de acciones disponibles, $\mathcal{A}_s$, en el enfoque del aprendizaje por refuerzo. Se pueden unificar estos dos conjuntos considerando que las acciones pueden ser consideradas como opciones. Cada acción $a$ corresponde a una opción que está disponible siempre y cuando se pueda ejecutar $a$ y esta opción siempre durará exactamente un paso y seleccionará siempre $a$ como la acción a ejecutar, es decir, $\pi(s, a) = 1$, siendo $\pi$ la política de la opción. Por lo tanto, podemos considerar que la decisión del agente siempre estará ligada a opciones, de las cuales algunas tendrán una duración de únicamente un paso y otras se extenderán a lo largo del tiempo. A las opciones previamente mencionadas, nos vamos a referir como opciones primitivas u opciones de un único paso. Al igual que en las acciones, es conveniente anotar las diferencias en las opciones disponibles entre los estados. $\mathcal{O} = \bigcup_{s \in \mathcal{S}} \mathcal{O}_s$, representa al conjunto de todas las opciones disponibles.

La definición de opciones está armada de forma tal que sea lo más similar posible a la de acciones. Al estar bien establecida la forma en que una opción termina, se puede definir una secuencia de opciones al igual que se hace con las acciones. También se pueden considerar políticas que tomen en cuenta opciones primitivas en vez de acciones, y también se puede modelar las consecuencias de elegir una opción al igual que se hacía cuando se elegía una acción.

Dadas dos opciones $a$ y $b$, considerémoslas que tomando en forma secuencial, tomando primero $a$ hasta que finalice y luego tomando $b$ hasta que termine. Podemos decir que ambas opciones en bloque forman otra opción, la cual podemos llamar $ab$, correspondiente al comportamiento de elegir primero $a$ y luego $b$. La composición de dos opciones de Markov generalmente suele ser una opción de Semi-Markov, no de Markov, esto se debe a que las acciones elegidas

no dependen completamente del estado actual, sino de la composición de estas. La composición de dos opciones de Semi-Markov siempre es una opción de Semi-Markov. Debido a que las acciones son casos particulares de las opciones, estas también pueden ser compuestas entre sí, para producir una secuencia de acciones deterministas.

Algo más interesante es la política sobre opciones. Cuando se inicia en un estado $s_t$, las políticas de Markov sobre opciones $\mu : \mathcal{S} \times \mathcal{O} \mapsto [0, 1]$, elige una opción $o \in \mathcal{O}_{s_t}$ respecto a la distribución de probabilidades $\mu(s_t, \cdot)$. La opción $o$ es elegida en el estado $s_t$, determinando acciones hasta que la opción termine en el estado $s_{t+k}$, momento en el cual una nueva opción es seleccionada acorde a $\mu(s_{t+k}, \cdot)$ y así sucesivamente. De esta forma, una política de opciones, $\mu$, determina una política convencional sobre acciones o *flat policy*, $\pi = \text{flat}(\mu)$. De ahora en adelante, usaremos el término de políticas para políticas sobre opciones, donde las flat policies están incluidas y son un caso específico de estas. Cabe aclarar que a pesar de que la política sea de Markov y que todas las opciones elegidas sean de Markov, es muy improbable que la flat policy sea de Markov si alguna opción no es primitiva (se extiende a lo largo del tiempo). La acción seleccionada por la flat policy en el estado $s_T$ no solamente depende de $s_T$ sino también de la opción en la que está y la opción depende estocásticamente en la historia $h_{tT}$ desde que la política fue inicializada en el instante $t$. De la misma forma que llamamos opciones de Semi-Markov, llamamos a las políticas que dependen de estas historias, políticas de Semi-Markov.

La definición de estado-valor y de acción-valor puede ser generalizada para ser aplicada a las opciones y las políticas. Dado un estado $s \in \mathcal{S}$ y siguiendo una política de Semi-Markov $\pi$, se define como el valor esperado de retorno $V_\pi(s)$ si la política empezara en $s$:

$$V^\pi(s) = \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... \mid \mathcal{E}(\pi, s, t)\right]$$

donde $\mathcal{E}(\pi, s, t)$ representa el evento de iniciar $\pi$ en el estado $s$ en el tiempo $t$.

Es natural generalizar funciones acción-valor a funciones opción-valor. Para esto se define $Q^\pi(s, o)$, el valor de elegir la opción $o$ en el estado $s$ con la política $\pi$ como:

$$Q^\pi(s) = \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... \mid \mathcal{E}(o\pi, s, t)\right]$$

donde $o\pi$, es decir, la composición de $o$ y $\pi$, hace referencia a la política de Semi-Markov que empieza en $o$ y la sigue hasta que ésta termina y a continuación se elige otra opción acorde a $\pi$. Para opciones de Semi-Markov es útil definir $\mathcal{E}(o, s, t)$, el evento de continuar $o$ en $h$ en

el tiempo $t$ donde $h$ es una historia que termina en el estado $s_t$. A continuación, las acciones son seleccionadas como si la historia hubiera precedido a $s_t$. Eso es que $a_t$ es elegida en base a $o(h, \cdot)$, y $o$ termina en $t+1$ con una probabilidad de $\beta(ha_t r_{t+1} s_{t+1})$; si no termina entonces $a_{t+1}$ es elegida de acuerdo a $o(ha_t r_{t+1} s_{t+1}, \cdot)$ y así sucesivamente.

## 3.1. Metodos de Semi-Markov (opción a opción)

Las opciones están muy relacionadas con las acciones en un proceso de decisión semi-markoviano o PDSM. De hecho, cualquier conjunto fijo de procesos de decisión markoviano o PDM es un PDSM.

La relación entre un PDMs, opciones y PDSMs proveen una base para la teoría para los métodos de planning y de aprendizaje con opciones.

El planning con opciones requiere un modelo con sus consecuencias. Para cada estado en el cual una opción puede haber empezado, este tipo de modelo predice el estado en el cual la opción terminará y la recompensa total recibida en el trayecto.

$$r_s^o = \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... + \gamma^{k-1} r_{t+k} \mid \mathcal{E}(0, s, t)\right]$$

donde $t+k$ es el tiempo en el cual $o$ termina. La parte de estado-predicción del modelo de $o$ es:

$$p_{ss'}^o = \sum_{k=1}^{\infty} p(s', k)\gamma^{k-1}$$

para todo $s' \in \mathcal{S}$, donde $p(s', k)$ es la probabilidad de que $o$ termine luego de $k$ pasos. En consecuencia, $p_{ss'}^o$ es una combinación entre la probabilidad que $s'$ es el estado en el cual $o$ terminará y la medida en que tanto demorará. A estos tipos de modelos se los llama modelo de tiempos múltiples porque describe el resultado potencial de la opción en distintos tiempos combinados de una forma apropiada.

Usando modelos de tiempos múltiples se pueden escribir las ecuaciones de Bellman para políticas y opciones. Para cualquier política de Markov $\pi$ la función de estado-valor puede ser

escrita como:

$$V^{\pi}(s) = \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... + \gamma^{k-1} r_{t+k} + \gamma^k V^{\pi}(s_{t+k}) \mid \mathcal{E}(\pi, s, t)\right]$$

(donde $k$ es la duración de la opción elegida en $s$)

$$= \sum_{o \in \mathcal{O}_s} \pi(s, o) \left[r_s^o + \sum_{s'} p_{ss'}^o V^{\pi}(s')\right]$$

Luego, la ecuación de la función de valor en un estado $s$ para una opción $o$ sería:

$$Q^{\pi}(s) = r_s^o + \sum_{s'} p_{ss'}^o \sum_{o \in \mathcal{O}_s} \pi(s', o) Q^{\pi}(s', o').$$

Finalmente, estas son las generalizaciones de la funciones de valor óptimas y las ecuaciones óptimas de Bellman para opciones y políticas con opciones. Claramente las funciones óptimas de valor convencionales $V^*$ y $Q^*$ no se ven afectadas por el uso de opciones. Sin embargo, es interesante saber que tan bien puede funcionar un conjunto restringido de opciones la cual no tiene todas las acciones. Suponiendo que se representa a un conjunto restringido de opciones como $\mathcal{O}$ y el conjunto de todas las políticas eligiendo solamente las opciones en $\mathcal{O}$ por $\Pi(\mathcal{O})$. Entonces, la función óptima de valor suponiendo que solo podemos elegir opciones de $\mathcal{O}$ es:

$$V_{\mathcal{O}}^*(s) = \max_{o \in \mathcal{O}_s} \mathbb{E}\left[r + \gamma^k V_{\mathcal{O}}^*(s') \mid \mathcal{E}(o, s)\right],$$

donde $r$ hace referencia al refuerzo acumulado descontado dentro de la opción, $k$ a la cantidad de pasos entre $s$ y $s'$ y $s'$ el estado donde termine $o$. La función de valor y la ecuación de Bellman para funciones opción-valor óptimas es:

$$Q_{\mathcal{O}}^*(s, o) = \mathbb{E}\left[r + \gamma^k \max_{o' \in \mathcal{O}_{s'}} Q_{\mathcal{O}}^*(s', o') \mid \mathcal{E}(o, s)\right],$$

donde $r$, $k$ y $s'$ tienen el mismo significado que en la anterior función pero esta vez para $o \in \mathcal{O}$.

Dado un conjunto de opciones $\mathcal{O}$, la correspondiente política óptima, $\pi_{\mathcal{O}}^*$, es aquella que logra $V_{\mathcal{O}}^*$ para todo $s \in \mathcal{S}$.

### 3.1.1. Métodos de Semi-Markov - Planning

Con las definiciones anteriores, un PDM junto a su conjunto de opciones $\mathcal{O}$ comprende formalmente un PDSM, y los métodos y resultados estándar estándar aplican a estos. Cada una

de las ecuaciones de Bellman para opciones conforma un sistema de ecuaciones cuya solución es única y corresponde a la función de valor. Estas ecuaciones de Bellman pueden ser utilizadas con el fin de actualizar las reglas del sistema y encontrar la función de valor. La solución de estos métodos para este problema es una aproximación de $V^*_{\mathcal{O}}(s)$ o de $Q^*_{\mathcal{O}}(s,o)$ para todos los estados y todas las opciones. Un ejemplo de este tipo de algoritmos es el de *synchronous value iteration* (SVI) cuyas opciones comienzan con una aproximación arbitraria $V_0$ y $V^*_{\mathcal{O}}$ y luego se computa una secuencia de nuevas aproximaciones.

### 3.1.2. Métodos de Semi-Markov - Aprendizaje

El problema de encontrar una política óptima para un conjunto de opciones $\mathcal{O}$ puede ser resuelto mediante métodos de aprendizaje. Como ya sabemos, al componer opciones PDM, obtenemos un PDSM, por lo tanto podemos aplicar métodos de aprendizaje PSDM. Al igual que en los métodos de planning, cada opción es vista como una unidad indivisible. Cuando la opción $o$ es comenzada en el estado $s$, luego pasamos al estado $s'$ donde la opción $o$ finaliza. Basado en esta experiencia, una aproximación a la función opción-valor, $Q(s,o)$ es actualizada. Por ejemplo, la versión PSDM de un paso de Q-Learning, la cual llamamos PSDM Q-Learning se actualiza cuando cada opción termina:

$$Q(s,o) \leftarrow Q(s,o) + \alpha \left[ r + \gamma^k \max_{o' \in \mathcal{O}_s} (Q(s',o') - Q(s,o)) \right],$$

siendo $k$ el número de pasos luego desde que se llegó a $s'$ desde $s$, $r$ es la recompensa acumulada dentro de la opción. El estimado es que $Q(s,o)$ converge a $Q^*(s,o)$ para todo $s \in \mathcal{S}$ y todo $o \in \mathcal{O}$.

## 3.2. Mejora de estados de finalización (Termination improvement)

Los métodos PDSM aplican a opciones siempre y cuando estas sean tratadas como unidades indivisibles. Los métodos más interesantes y más efectivos son posibles alterando la estructura interna de la opción. Es posible analizar las opciones en términos de un PDSM y luego usar una interpretación PDM para modificar las opciones y generar un nuevo PDSM.

Podemos considerar analizar las condiciones de finalización. Una vez que una opción es elegida, los métodos requieren que la política sea seguida hasta que la opción termine. Supo-

niendo que tenemos definida una función action-valor $Q^\pi(s, o)$ para una política $\pi$ y para todo par $s$-$o$, estado-opción, que se pueda dar en $\pi$. Esta función nos indicará que tan bien nos irá mientras seguimos la política $\pi$. Suponiendo que en un tiempo $t$ estamos en la mitad de la ejecución de una opción $o$. Si $o$ es de Markov, entonces podemos comparar el valor de continuar $o$, el cual es $Q^\pi(s_t, o)$, con el valor de terminar y elegir una nueva opción acorde a $\pi$, la cual es $V^\pi(s, o) = \sum_q \pi(s, q)Q^\pi(s, q)$. Si la siguiente opción tiene un valor más alto, entonces ¿Por qué no terminar $o$ y permitir el cambio de opción? De hecho, demostraremos que esta forma de comportamiento es aún mejor.

Se puede caracterizar un nuevo comportamiento siguiendo una política $\pi'$, la cual es igual a $\pi$ pero con un nuevo conjunto de opciones $\mathcal{O}'$ tal que $\pi'(s, o') = \pi(s, o)$ para todo $s \in \mathcal{S}$. Para cada nueva opción $o'$ es exactamente lo mismo que la anterior opción $o$ excepto que termina cuando la finalización parece ser mejor que seguir con la misma opción de acuerdo a $Q^\pi$. En otras palabras, la condición de terminación $\beta'$ de $o'$ es la misma que en $o$, salvo que $\beta'(s) = 1$ si $Q^\pi(s, o) < V^\pi(s)$. Llamaremos a esta $\pi'$ una *termination improved policy* de $\pi$. Esto debilita el requerimiento de que $Q^\pi(s, o)$ tiene que ser completamente conocida. Y una mayor y más importante generalización es que esto se puede aplicar a opciones de Semi-Markov y a opciones de Markov.

La principal aplicación de la *termination improvement* es que considerando una política óptima $\pi$ con un conjunto de opciones $\mathcal{O}$. Ya hemos discutido cómo, mediante aprendizaje o planning, se puede obtener la función óptima de valor $V^*_\mathcal{O}$ y $Q^*_\mathcal{O}$ y desde estas funciones una política óptima $\pi^*_\mathcal{O}$ que se consigue en base a estas. Esto es lo mejor que se puede conseguir sin cambiar las opciones de $\mathcal{O}$, la cual es un PDSM definido por $\mathcal{O}$. El *termination improvement* nos da nos da una forma de mejorar $\pi^*_\mathcal{O}$, usando un poco más de tiempo de cómputo por paso. Esto se debe a que a cada paso nos frenamos para verificar si se debe terminar o no la opción. Estas verificaciones ocupan muy poco tiempo de cómputo por paso. Por lo tanto, *termination improvement* nos da una mejora casi gratis por sobre métodos PSDM de planning o aprendizaje que computan $Q^*_\mathcal{O}$ en pasos intermedios.

En el caso extremo, se podría interrumpir en todos los pasos y cambiar a la opción greedy en ese estado debido a que retorna el mayor valor de acuerdo a $Q^*_\mathcal{O}$. En ese caso, las opciones no son seguidas por más de un paso y pueden resultar innecesarias. Sin embargo, estas son importantes para generar $Q^*_\mathcal{O}$, la base para generar los cambios greedy y así poder encontrar

$Q^*_{\mathcal{O}}$ más rápido que lo que se tardaría en encontrar $Q^*$.

## 3.3. Aprendizaje del modelo dentro de la opción

El modelo de un ambiente para una opción $r^o_s$ y $p^o_{ss'}$, pueden ser aprendidos mediante la experiencia dado el conocimiento de la opción. Para una opción de Semi-Markov, el único enfoque general es ejecutar la opción hasta que termine múltiples veces en cada estado $s$, obteniendo el estado siguiente $s'$, el refuerzo acumulado $r$ y el tiempo necesario $k$. Estos resultados luego son promediados para obtener los valores esperados para $r^o_s$ y $p^o_{ss'}$.

$\hat{r}^o_s = \hat{r}^o_s + \alpha\,[r - \hat{r}^o_s]$, y

$\hat{p}^o_{sx} = \hat{p}^o_{sx} + \alpha\left[\gamma^k \delta_{s'x} - \hat{p}^o_{sx}\right]$,

para todo $x \in \mathcal{S}$, donde $\delta_{sx} = 1$ si $s = x$ y 0 en caso contrario y $\alpha$ es el parámetro del tamaño del paso, puede ser constante o depender del estado, de la opción y del tiempo. Sin embargo, cuando el promedio está hecho lo llamamos *métodos de aprendizaje del modelo* PDSM, porque como en los métodos de aprendizaje de la función de valor de PDSM están basados en saltar desde la iniciación hasta la terminación por cada opción ignorando lo que ocurre en el medio. En el caso especial que $o$ sea una opción primitiva, el aprendizaje del modelo se reduce a los métodos convencionales de acciones donde se usa solo un paso.

## 3.4. Aprendizaje de la función de valor dentro de la opción

Ahora vamos a describir el aprendizaje de la función opción-valor dentro de la opción. Si las opciones son de Semi-Markov, entonces podemos utilizar los métodos de aprendizaje explicados en la sección de aprendizaje para PDSM; una opción de Semi-Markov debe ser completada antes de ser evaluada de cualquier forma. En cambio, si las opciones son de Markov se pueden considerar algunos métodos para dentro de la opción.

Es conveniente introducir una nueva notación para la función de valor con el par estado-opción, dado que la opción es de Markov y es ejecutada desde la llegada al estado $s$:

$$U^*_{\mathcal{O}}(s,o) = (1-\beta(s))Q^*_{\mathcal{O}}(s,o) + \beta(s)\max_{o'\in\mathcal{O}} Q^*_{\mathcal{O}}(s,o').$$

Luego, podemos escribir algo parecido a las ecuaciones de Bellman que se relacionen con $Q^*_{\mathcal{O}}(s,o)$

para valores esperados de $U^*_{\mathcal{O}}(s', o)$ donde $s'$ es el sucesor inmediato de $s$ luego de iniciarse la opción de Markov $o$:

$$Q^*_{\mathcal{O}}(s, o) = \sum_{a \in \mathcal{A}_s} \pi(s, a) \left[ r^a_s + \sum_{s'} (p^a_{ss'} U^*_{\mathcal{O}}(s', o)) \right],$$

donde $r^a_s$ es la recompensa luego de que se llegó a $s'$. Ahora, considerando los métodos de aprendizaje basados en las ecuaciones de Bellman y suponiendo una acción $a_t$ que fue elegida en el estado $s_t$ y produce un siguiente estado $s_{t+1}$ con una recompensa $r_{t+1}$ y la acción $a_t$ es consistente con la política de Markov $\pi$ y una opción $o$, entonces, la ecuación de Bellman sugerida arriba se podría aplicar para una actualización de una off-policy de un paso de diferencia temporal:

$$Q(s, o) \leftarrow Q(s, o) + \alpha \left[ (r_{t+1} + \gamma U(s_{t+1}, o)) - Q(s_t, o) \right],$$

donde

$$U(s, o) = (1 - \beta(s)) Q_{\mathcal{O}}(s, o) + \beta(s) \max_{o' \in \mathcal{O}} (Q_{\mathcal{O}}(s, o')).$$

Este método se llama *one step intra-option Q-Learning* y se aplica para la actualización de la política para todas las opciones $o$ de forma consistente con la acción $a_t$.

## 3.5. Aprender la política de las opciones

El aspecto más importante de trabajar con PDM y con PDSM, es el hecho de que las opciones que conforman el PDSM pueden cambiar. Esto lo vimos mediante el cambio de las condiciones de finalización y tal vez, más importante que eso, cambiando las políticas. Es natural pensar que las opciones tienen como objetivo cumplir un subobjetivo y adaptan sus políticas con el fin de cumplirlo lo mejor posible. Dados los subobjetivos para las distintas opciones, se vuelve bastante directo poder diseñar un aprendizaje interno de la opción para que se ajuste a la política que resuelve el subobjetivo.

Por el otro lado, no existe una forma clara en la cual formular los subobjetivos con el fin de asociarlos a la opción o de al menos sentar las bases de qué evaluaciones hay que realizar. Una importante consideración a tener en cuenta es que el modelo de opción construido para cumplir un subobjetivo puede ser transferido para cumplir otro. El propósito sería poder plantear un agente el cual vaya aprendiendo distintas series de subtareas y se vuelva más y más capaz. La forma de poder realizar una transferencia completa entre subobjetivos probablemente resulte

en desarrollar ideas en donde se utilicen opciones jerárquicas generales (opciones que eligen otras opciones dentro de sí). La formalización de los subobjetivos que se presenta aquí alcanza para ilustrar algunas de las posibilidades y problemas que suelen aparecer. Un gran problema es que no se sabe de donde provienen los subobjetivos. Se asume que los subobjetivos son dados y se centran en cómo las opciones deben ser aprendidas para ser logradas y en cómo diferenciar sus distintas metas para luego ayudarse mutuamente.

Una forma simple de formular un subobjetivo es asignándole un valor a cada uno, $g(s)$, para cada estado $s$ en el conjunto $\mathcal{G} \subseteq \mathcal{S}$. Estos valores indican que tan deseable es terminar en ese estado en $\mathcal{G}$. Sea $\mathcal{O}_\mathcal{G}$ el conjunto de opciones tal que siempre terminan en estados de $\mathcal{G}$ en donde $g$ está definido. Dada la función subobjetivo-valor $g : \mathcal{G} \mapsto \mathbb{R}$, se puede definir una nueva función estado-valor representada como $V_g^o(s)$, para $o \in \mathcal{O}_\mathcal{G}$, siendo el valor esperado de la recompensa acumulada siendo si la opción $o$ fue iniciada en el estado $s$, más el valor del subobjetivo $g(s')$ donde $s'$ es el estado donde termina la opción. Asimismo, se puede definir una función acción-valor tal que $Q_g^o(s,a) = V_g^{ao}(s)$.

Finalmente, se puede definir una función de valor óptima para cada subobjetivo $g$:

$V_g^*(s) = \text{máx}_{o \in \mathcal{O}_g} V_g^o(s)$ y $Q_g^*(s,a) = \text{máx}_{o \in \mathcal{O}_g} Q_g^o(s,a)$.

Encontrar la opción que consigue estos máximos implica una subtarea bien definida. Para opciones de Markov, estas subtareas tienen ecuaciones de Bellman y métodos de aprendizaje y planning tal como las tareas originales. Por ejemplo, un paso del método de Q-Learning para obtener un estimado de $Q_g(s_t,a_t)$ a $Q_g^*(s_t,a_t)$ es:

$$Q_g(s_t,a_t) \leftarrow Q_g(s_t,a_t) + \alpha \left[ r_{t+1} + \gamma \max_a Q_g(s_{t+1},a_{t+1}) - Q_g(s_t,a_t) \right] \text{ si } s_{t+1} \notin \mathcal{G}$$

$$Q_g(s_t,a_t) \leftarrow Q_g(s_t,a_t) + \alpha \left[ r_{t+1} + \gamma g(s_{t+1}) - Q_g(s_t,a_t) \right] \text{ si } s_{t+1} \in \mathcal{G}.$$

Es interesante notar, que en general, todas las políticas aprendidas para aprender subobjetivos dependen en detalle de la precisión de los valores de $g$ para cada subobjetivo.

Subobjetivos, opciones y modelos de opciones admiten nuevas posibilidades a agentes de aprendizaje por refuerzo. Por ejemplo, se puede presentar a un agente con un conjunto de tareas como subobjetivos y tal vez calificarlas en función de su dificultad. Para cada tarea, el agente irá aprendiendo nuevas opciones con las cuales conseguirá cada subobjetivo y aprenderá el modelo de la opción, es decir, irá creando su propia política. A pesar de que el modelo y la

opción son implementados en base a la tarea, estos se pueden transferir a cualquier otra tarea una vez aprendidos por el agente. La opción sólo indicará qué hacer; si comportarse de esta forma es útil para el siguiente paso o la siguiente tarea, entonces el agente lo hará.. De forma similar, el modelo solo predice las consecuencias de comportarse de esa forma; si comportarse de esta forma es útil para el siguiente paso o la siguiente tarea, entonces el agente lo hará. Siempre y cuando el modelo sea acorde a su opción, entonces puede resultar útil planear la siguiente tarea.

## 3.6. Busqueda de opciones

Si el conjunto de iniciación y la condición de terminación de la opción son especificadas, entonces, la política interna de la opción puede ser aprendida mediante algoritmos estándar de aprendizaje por refuerzo. Sin embargo, el problema principal está en encontrar los conjuntos de iniciación y la condición de terminación óptimos. Intuitivamente, es fácil reconocer que diferentes heurísticas podrían funcionar en diferentes ambientes. En esta sección se asumirá que el agente resolverá distintas tareas episódicas, es decir, con un objetivo definido, el cual al alcanzarse se dará por finalizada la tarea.

Se permitirá al agente explorar el ambiente previamente con el fin de aprender las opciones. El proceso de encontrar las opciones está compuesto por una serie de tareas al azar en el ambiente del experimento las cuales el agente resolverá. Durante este periodo, el agente obtendrá estadísticas respecto a la frecuencia de ocurrencia de los distintos estados. El algoritmo está basado en que la suposición de que si un estado ocurre frecuentemente en trayectorias que representan soluciones a tareas aleatorias, entonces estos estados deben ser importantes. Por lo tanto, se aprenderán opciones que tengan a estos estados como objetivos. El algoritmo consiste en:

1. Seleccionar un número aleatorio de estados de comienzo $S$ y de estados objetivo $T$, los cuales serán utilizados como tareas aleatorias para el agente.

2. Para cada par de estados $\langle S,\ T\rangle$:

   *a*) Realizar $N_{train}$ episodios de Q-Learning para aprender la política para ir de $S$ a $T$.

   *b*) Realizar $N_{test}$ episodios de greedy utilizando la política aprendida. Para cada estado

$s$, se contabilizarán la cantidad de veces, $n$(s), que se visitó el estado en las distintas trayectorias.

3. Repetir este ciclo hasta que se obtengan la cantidad de opciones deseadas.

   *a*) Elegir el estado que haya sido visitado más veces, $T_{max} = \text{argmax}_s\, n(s)$.

   *b*) Calcular $n(s, T_{max})$, siendo esto, la cantidad de veces que aparece $s$ en un camino que pasa por $T_{max}$.

   *c*) Calcular $\overline{n}(T_{max}) = \text{avg}_s\, n(s, T_{max})$

   *d*) Seleccionar todos los estados $s$ tal que $n(s, T_{max}) > \overline{n}(T_{max})$

4. Para cada opción se debe aprender la política interna; esto se consigue dando un refuerzo alto cada vez que se llega a $T_{max}$, y en caso contrario no darlo. El agente realizará episodios de Q-Learning los cuales comienzan en estados aleatorios de sus respectivos conjuntos de iniciación y finalizan cuando se llega a $T_{max}$.

Una vez que todas las opciones fueron encontradas, se utiliza Q-Learning con el fin de aprender la política por encima de las opciones. Es decir, la que relaciona las opciones previamente encontradas.

# 4. Hipótesis

En un problema de aprendizaje por refuerzo un elemento fundamental es la función de refuerzo o recompensa. La importancia de la misma, como se explicó en la sección de Fundamentos, yace en que la misma guía al agente indicando para cada estado qué acciones son buenas y cuáles no. En otras palabras, define cuál es el objetivo del agente.

Por otro lado, almacenar conocimiento adquirido para ser aprovechado en otras situaciones es de gran utilidad. Tener que aprender a caminar cada vez que se aprende a realizar una tarea que requiere de caminar no es eficiente. Nuestro trabajo se basa en comprobar dos hipótesis relacionadas con los temas mencionados anteriormente y proponer líneas de trabajo futuras en base a los mismos.

En primer lugar se buscará demostrar que es posible indicar el objetivo del agente en la definición del ambiente manteniendo siempre la misma función de refuerzo, la función de refuerzo universal.

En segundo lugar se plantea la hipótesis de que es posible adquirir conocimiento, incorporarlo y utilizarlo para resolver situaciones complejas que de otro modo no sería factible en un tiempo razonable.

Nuestro trabajo apunta a exponer la utilidad de los postulados anteriores. Por cuestiones relacionadas al alcance del trabajo, los experimentos pretenden proveer una demostración empírica de las hipótesis para determinadas situaciones. Se propone como líneas de trabajo a futuro la demostración formal de los mismos.

# 5. Implementación

En esta sección se explicarán los distintos componentes que forman parte de la implementación realizada. A su vez, se justificarán las decisiones tomadas en cuanto al diseño propuesto.

## 5.1. Ambiente

### 5.1.1. Descripción

El ambiente utilizado para realizar las experiencias es una grilla bidimensional de tamaño arbitrario, como se puede observar en la Figura 2. Cada celda de la grilla puede contener: al agente, una pared (representada en gris en la figura), una puerta (rojo), o la llave de alguna puerta. Por otro lado, la grilla tiene marcadores para las posiciones de inicio (verde) y finalización (amarillo). Estos marcadores no afectan la dinámica del ambiente, pero sí permiten que el componente encargado de realizar la simulación pueda situar al agente al iniciar, y determinar si se llegó al objetivo final.

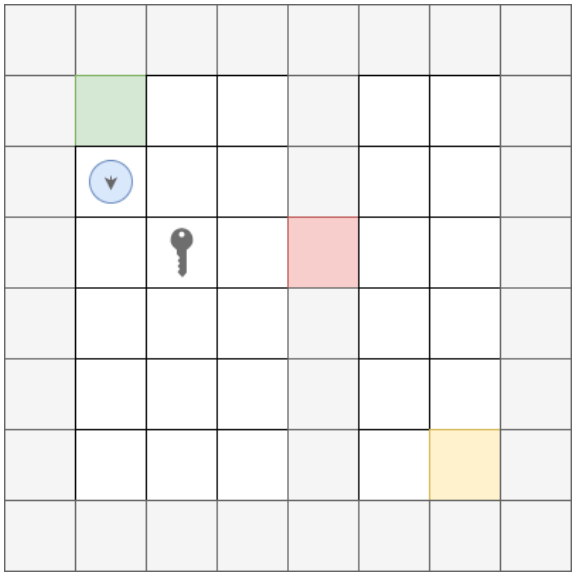


Figura 2: Representación gráfica del ambiente grilla

Se eligió este tipo de ambiente ya que si bien es sencillo, permite agregar distintas situaciones de dificultad incremental con facilidad, lo cual, agiliza la experimentación.

### 5.1.2. Acciones

El ambiente implementado admite tres acciones posibles: avanzar, rotar $90^{o}$ en sentido horario, y rotar $90^{o}$ en sentido anti horario. Cada una de estas acciones modifica directamente al ambiente. Las tres acciones mencionadas pueden ser realizadas en cualquier estado del mismo. Cabe destacar que dependiendo del estado del ambiente, el efecto que producen las acciones puede variar. Por ejemplo, al realizar la acción de avanzar, el agente no avanzará si la celda que tiene delante es una pared, mientras que, si está libre, sí se realizará el movimiento. Las acciones están implementadas de forma que proveen un efecto y un mecanismo para revertir ese efecto. Cuando un agente realiza una de estas acciones, el componente encargado de la simulación del ambiente realiza una serie de verificaciones para asegurarse que, ante ese efecto, el ambiente siga en un estado válido. De no ser así, se revierte el efecto.

### 5.1.3. Reglas del ambiente

El ambiente implementado tiene una serie de reglas básicas que definen la dinámica del mismo.

- Si el agente se sitúa sobre una celda con una llave, se elimina la llave de la grilla y se la disponibiliza al agente.
- Si el agente se sitúa sobre una celda con una puerta, sólo se admite el movimiento si el mismo tiene la llave de dicha puerta.
- Si el agente se sitúa sobre una celda con una pared, no hay efecto alguno, el estado permanece igual.
- Si el agente avanza a una celda libre contigua, cambiará su posición.
- Si el agente rota en cualquier dirección, cambiará su orientación, afectando movimientos futuros.

### 5.1.4. Mapa

La forma de almacenar una configuración inicial del ambiente es mediante el concepto de mapa. Se diferencia del concepto de ambiente en que el ambiente es una entidad dinámica,

esto es, cambia conforme el agente va realizando acciones sobre el mismo. Por otro lado, el mapa es estático. Define la ubicación de los elementos que conforman el ambiente, y se utiliza para inicializar al mismo. Los mapas son un archivo de texto en donde cada carácter tiene un significado y representa un elemento en la grilla. La gramática de los mismos es:

$$
\begin{aligned}
\langle \text{mapa} \rangle &\models \langle \text{fila} \rangle \langle \text{mapa} \rangle \mid \langle \text{fila} \rangle \\
\langle \text{fila} \rangle &\models \langle \text{celda} \rangle \langle \text{fila} \rangle \mid \langle \text{celda} \rangle \\
\langle \text{celda} \rangle &\models \langle \text{puerta} \rangle \mid \langle \text{llave} \rangle \mid \langle \text{vacío} \rangle \mid \langle \text{pared} \rangle \mid \langle \text{inicio} \rangle \mid \langle \text{fin} \rangle \\
\langle \text{vacío} \rangle &\models \textit{espacio} \\
\langle \text{puerta} \rangle &\models \textit{letra mayúscula} \\
\langle \text{llave} \rangle &\models \textit{letra minúscula} \\
\langle \text{pared} \rangle &\models \texttt{=} \\
\langle \text{inicio} \rangle &\models \texttt{0} \\
\langle \text{fin} \rangle &\models \texttt{1}
\end{aligned}
$$

Como se puede observar en la gramática, tanto las puertas como las llaves se representan con letras mayúsculas y minúsculas respectivamente. Esto establece una asociación entre las llaves y las puertas. Por ejemplo, una puerta representada por el carácter `A`, puede ser abierta solo por una llave representada por el carácter `a`. Es importante remarcar que pueden existir puertas y llaves con la misma letra en distintas posiciones del mapa. Esto permite situaciones en donde una llave puede abrir más de una puerta y casos en donde una misma puerta puede ser abierta por más de una llave.

## 5.2. Estado

El estado es una serie de valores que definen la percepción del agente sobre el ambiente. En otras palabras, el agente solo cuenta con información parcial acerca de las situación actual o estado del ambiente. Esta información parcial con la que cuenta es su percepción del ambiente y lo que en la implementación recibe el nombre de estado.

Esta representación es la que utiliza el agente para tomar decisiones tanto en su etapa de explotación como también durante el aprendizaje.

Un punto importante sobre el diseño de esta entidad es que, agregar demasiada información al estado, supone un aumento en el cardinal del espacio de estados. Esto trae como consecuencia, un aumento en la cantidad de entradas en la tabla de estados-acciones a valores $Q$ y por consiguiente una degradación en el proceso de aprendizaje debido a la gran cantidad de situaciones a considerar y aprender.

Por otro lado, se dificulta utilizar conocimiento previo en situaciones en las que el agente cuenta con más información. Por ejemplo, si el agente adquiere el conocimiento necesario, para llegar a un objetivo, contando solo con información de la distancia al mismo, luego se dificulta utilizar este conocimiento para aprender a llegar a un objetivo atravesando una puerta. Esto se debe a que en esta última situación, posiblemente, en el estado se cuente con información adicional relativa a la puerta y la llave, que en el conocimiento previo no se tiene.

Para solucionar el problema anterior, se decidió enmascarar parte del estado con mayor información para que sea compatible con el estado utilizado al aprender el conocimiento previo. Al hacer esto, surgió un nuevo problema, se debe implementar una función máscara para cada conocimiento previo a utilizar.

Una forma de simplificar esta situación es dividir el estado en distintas partes etiquetadas. Por ejemplo, aprender a ir a un objetivo, puede realizarse con un estado con una sola parte etiquetada `distancia al objetivo` la cual puede ser las dos componentes de la distancia polar al objetivo. Por otro lado, la situación mencionada anteriormente con la puerta y la llave, podría realizarse con un estado con tres partes, etiquetadas `distancia a la llave`, `distancia a la puerta` y `distancia al objetivo`. Entonces, utilizando esta representación y agregando a cada conocimiento aprendido las etiquetas del estado que requiere, es fácil enmascarar solo la parte del estado necesaria. Esto permite, reutilizar conocimiento previo agregando nuevas partes y por otro lado, evita que la tabla relativa al comportamiento, por ejemplo, de ir a un objetivo, contenga información irrelevante a la tarea que resuelve, como son la distancia a la llave y a la puerta.

## 5.3. Sensores

En la sección anterior, se remarcó el problema enfrentado al reutilizar conocimiento previo y la solución propuesta de enmascarar partes del estado utilizando etiquetas. Una abstracción

útil que proponemos es la utilización de sensores.

Un sensor permite percibir una determinada característica del ambiente, como puede ser la distancia a un punto o la presencia de obstáculos. Al momento de realizar el aprendizaje, se agregan al agente distintos sensores mediante los cuales se construye el estado del mismo.

Las implementaciones de sensores provistas corresponden a los necesarios para los experimentos realizados. Estos son: sensor RSSI (*Received Signal Strength Indicator*) que simula el nivel de señal recibido desde un determinado tipo de celda; y sensor de proximidad que devuelve la distancia en celdas al obstáculo más cercano. Este último tipo de sensor se orienta en la dirección de sensado deseada y se pueden agregar al agente tantos como sean necesarios.

Utilizando esta abstracción, al definir el aprendizaje, se indican qué tipos de sensores son los que se van utilizar y los nombres de cada uno. Al finalizar el aprendizaje, se incluye esta información en el conocimiento adquirido de forma tal, que al reutilizarlo, se pueda reconstruir la función máscara del estado.

## 5.4. Políticas y Opciones

### 5.4.1. Opciones

Una opción, en la implementación, es una abstracción del concepto de opción explicado en la sección Encuadre de Opciones. La misma tiene algunas diferencias y agregados con respecto al concepto. Se agregó un campo para indicar el nombre de una opción así como una lista de sensores necesarios para ejecutarla. Por otro lado, las condiciones de iniciación y estados de terminación se definen como predicado del estado en ambos casos.

Existen dos implementaciones principales de opciones:

- `Option`: es una opción que encapsula una política y contiene una condición de inicio y terminación.
- `PrimitiveOption`: es una opción que realiza una acción primitiva y finaliza en un solo paso temporal.

Por otro lado, también se incluyen otras dos implementaciones de opciones que se basan en

`Option`, pero que agregan otros mecanismos de finalización además de la condición de terminación. Estos tipos son:

- `MaxStepsOption`: agrega un parámetro que permite indicar la cantidad máxima de pasos de la opción. La condición de terminación es que se cumpla la condición de la opción o que la cantidad de pasos sea mayor al parámetro indicado.
- `BetaOption`: permite indicar una función de la cantidad de pasos la cual es la probabilidad de que la opción termine. La condición de terminación de la opción es que ocurra el evento de finalización con probabilidad $\beta$, o que se cumpla la condición de terminación de la opción.

Anteriormente se mencionó la existencia de una lista de sensores necesarios para ejecutar la opción. Esta lista permite generar de forma automática la máscara del estado. Esta máscara es aplicada sobre el estado completo para obtener solo la porción relevante a la opción. Esta porción del estado se utiliza para obtener las acciones de la política subyacente y verificar las condiciones de inicio y terminación. Es importante aclarar que la máscara no es aplicada en las opciones primitivas. Notar que no es necesario ya que las mismas pueden comenzar siempre, finalizan en un solo paso temporal y no tienen una política subyacente para la cual haya que enmascarar el estado.

### 5.4.2. Políticas

Las políticas implementadas son sobre opciones (son incluídas las opciones primitivas). Se implementó el tipo de política greedy, es decir, una política en la cual se elige la opción que maximiza el valor de $Q$ para ese estado. La misma también admite que se indique un valor de $\varepsilon$, pasando a ser una política $\varepsilon$-greedy, mediante el cual se balancea exploración y explotación. Este valor es la probabilidad de que el agente elija una opción no óptima siguiendo esa política.

Las políticas internamente contienen una tabla de valores $Q$ la cual está indexada por estados y opciones. Para cada estado existe una lista de valores $Q$ vinculados a cada una de las opciones posibles en la política.

## 5.5. Proceso de aprendizaje

### 5.5.1. Pasos del proceso de aprendizaje

Para realizar el aprendizaje de un determinado comportamiento se debe seguir una serie de pasos. Se define qué estados, determinados por los mapas, son los que se van a presentar al agente durante el aprendizaje. Se indica de qué forma van a cambiar los parámetros $\alpha$ y $\epsilon$ en función del episodio. Se indican cuáles van a ser las opciones disponibles para el agente (incluyendo opciones primitivas). También se indica con qué sensores cuenta el agente, especificando nombre y propiedades del sensor (ver apartado Sensores). Luego se utiliza el método Q-Learning con opciones para realizar el aprendizaje, obteniendo como salida del mismo, una política. Finalmente se empaqueta la política en una opción. Este conocimiento obtenido, empaquetado en la opción se puede guardar y ser incorporado como una de las opciones en un nuevo aprendizaje. Es importante notar que *no se define ninguna función de refuerzo*. En la sección Experimentos se abordará en más detalle este tema.

### 5.5.2. Empaquetado de la opción

Se explicó cómo se realiza el aprendizaje de un comportamiento. Uno de esos pasos, el último, es el empaquetado del conocimiento obtenido, la política, dentro de una opción.

Se define cual es la política a empaquetar. Se indica un nombre para la opción a obtener. Se definen las condiciones de iniciación y terminación como funciones del estado. Además se provee la definición del agente utilizada de la cual se obtiene la lista de sensores. Esto permite generar la función máscara a partir de la lista de sensores obtenida desde el agente. Solo se consideran los sensores cuyo nombre esté en la lista de sensores indicada. Finalmente, se obtiene la opción que permite al agente realizar la tarea aprendida. En caso de que se desee utilizar la implementación de opción `MaxStepsOption`, se indica además la cantidad de pasos máxima que admite la opción. Para el caso de `BetaOption` se indica la función de probabilidad beta de que termine la opción dado un estado.

### 5.5.3. Pseudocódigo del proceso de aprendizaje

A continuación se detalla, a modo de ejemplo, los pasos seguidos para aprender a llegar a un objetivo contando con un sensor RSSI en pseudocódigo.

**Algoritmo 2** Proceso de aprendizaje

**procedure** APRENDEROBJETIVO

$M \leftarrow$ CARGARMAPAS

$A \leftarrow$ CREARAGENTE

$\mathcal{O} \leftarrow \{\texttt{Forward}, \texttt{RotateCW}, \texttt{RotateCCW}\}$ ▷ Definir opciones a usar

$\alpha \leftarrow f(e) = 0{,}3$ ▷ Factor de aprendizaje

$\varepsilon \leftarrow f(e) = 0{,}1$ ▷ Probabilidad de no seguir la política

AGREGARSENSOR(A, `RSSI`, `"goal_sensor"`)

$\pi \leftarrow$ QLEARNING(A, M, $\mathcal{O}$, $\alpha$, $\epsilon$) ▷ Iniciar aprendizaje

$p_0 \leftarrow f(s) =$ `TRUE` ▷ Condición de inicio

$p_f \leftarrow f(s) =$ `FALSE` ▷ Condición de terminación

**return** CREAROPCION(`"GoToGoal"`, $\pi$, $A$, $p_0$, $p_f$)

**end procedure**

# 6. Experimentos y resultados

En esta sección se describen los experimentos llevados a cabo para contrastar las hipótesis planteadas. Para cada experimento se define cual es la motivación, una descripción del mismo, los resultados obtenidos y un breve análisis de los mismos. Además se analizarán los resultados de esto y se confirmará si se pudo cumplir o no el objetivo de este.

## 6.1. Parámetros utilizados

Para realizar los aprendizajes de todos los experimentos, se utilizaron los mismos parámetros. En el Cuadro 1 se detallan los valores utilizados.

| Parámetro | Valor |
|---|---|
| Episodios | 1000 |
| Pasos por episodio | 500 |
| $\alpha$ | $0{,}8 \cdot e^{-t/episodes}$ |
| $\epsilon$ | $0{,}8 \cdot e^{-10t/episodes}$ |
| $\gamma$ | 0,9 |

Cuadro 1: Parámetros utilizados en el aprendizaje

## 6.2. Comportamiento complejo con opciones y sin opciones

El objetivo de este experimento es probar la hipótesis de que un comportamiento complejo, el cual es claramente subdivisible en tareas más pequeñas, es más eficiente resolverlo utilizando opciones que solo acciones. Donde la eficiencia del proceso la medimos en base a la cantidad de pasos temporales requeridos durante el aprendizaje para que el agente aprenda el comportamiento.

### 6.2.1. Descripción del experimento

El agente estará localizado en una habitación y tendrá como objetivo llegar a una determinada ubicación del mapa. Para poder llegar a esta ubicación el agente deberá salir de la habitación en la que está ubicado inicialmente, la única forma de salir es agarrando una llave ubicada en la habitación y abrir una puerta con la llave. Luego de realizar esto, el agente podrá ir a su objetivo y así terminar el experimento.

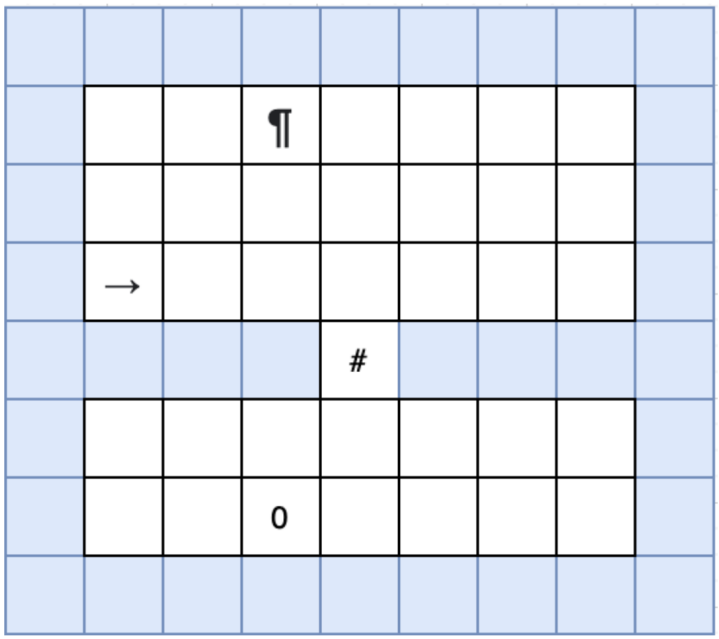



Figura 3: La flecha es la posición inicial del agente con su orientación, el # la puerta, el ¶ la llave y el 0 la posición objetivo. Las paredes se ven en azul.

En cuanto a la percepción del estado del agente contará con 3 sensores diferentes. Un sensor mide la distancia y el ángulo a la llave, otro la distancia y el ángulo a la puerta y el último mide la distancia y el ángulo a la celda objetivo. En cuanto a la distancia, el rango es de una celda con el fin de minimizar la cantidad de estados, es decir, puede tomar 0 si el objetivo está a más de una celda y 1 en caso contrario. El ángulo es discretizado de a $45^{o}$, eso quiere decir que los valores posibles son $0^{o}$, $45^{o}$, $90^{o}$, $135^{o}$, $180^{o}$, $225^{o}$, $270^{o}$ y $315^{o}$. El ángulo se ve modificado en función de la orientación del agente, eso quiere decir que $0^{o}$ hace referencia a lo que el agente tiene en frente y $180^{o}$ lo que tiene por detrás, el ángulo crece de forma antihoraria, es decir, que si por ejemplo la llave está a la izquierda del agente, el sensor del ángulo indicará que la llave está a $90^{o}$.

El agente tiene a su disposición 3 acciones posibles, `Walk`, avanza una posición en la orientación actual del agente, `RotateCW`, gira hacia la derecha o en sentido horario y `RotateCCW` rotar hacia la izquierda o en sentido anti-horario.

Luego, para el caso del experimento con opciones, se utilizó un máximo de 30 pasos por opción, y se decidió utilizar 3 opciones diferentes. `GoToKey`, para ir a llave, `GoToDoor`, para ir a la puerta y `GoToGoal` para ir al objetivo.

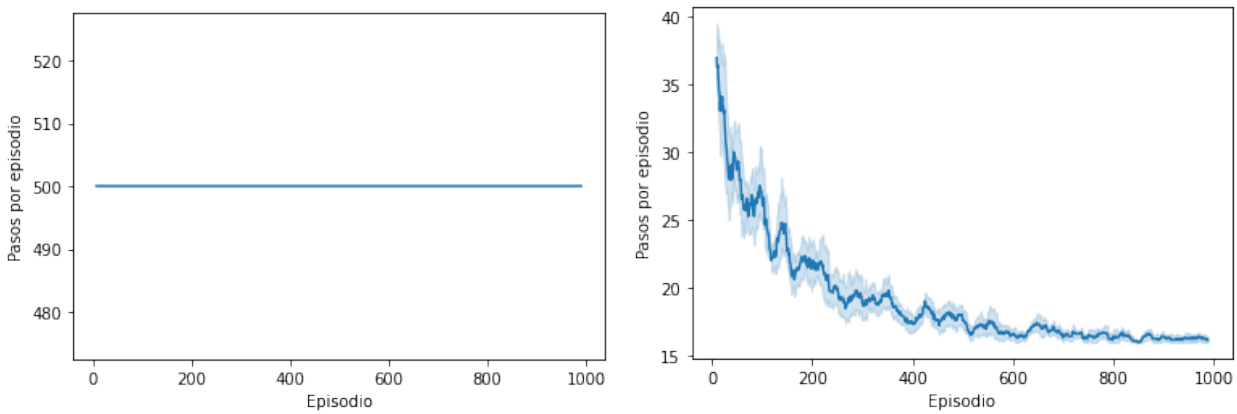
En la Figura 3 se puede observar el ambiente utilizado para el experimento.

### 6.2.2. Resultados

Para comparar el rendimiento del aprendizaje con opciones y sin opciones se utilizó como métrica la cantidad de pasos por episodio.

Para que el resultado sea estadísticamente significativo se hicieron 10 corridas por cada prueba. En los resultados se presenta un promedio de las mismas.

El resultado de la cantidad de pasos por episodio es muy ruidoso. Por esta razón se decidió hacer un promedio a lo largo del tiempo utilizando una ventana deslizante de 20 puntos de la serie.




(a) Aprendizaje sin opciones

(b) Aprendizaje con opciones

Figura 4: Curva de aprendizaje para el experimento descripto en la Sección 6.2.1

Como se puede observar en la Figura 4, el aprendizaje por refuerzo con opciones permite aprender un comportamiento complejo, como lo es agarrar una llave, cruzar una puerta y llegar a un objetivo, de forma exitosa en una cantidad de episodios razonable. Por otro lado, no se

logró aprender la política que permita realizar esta tarea sin utilizar opciones en la misma cantidad de episodios.

Con este experimento, además se puede comprobar cómo el agente logra aprender a resolver un problema cuya resolución implica dependencias temporales. Esto es, una opción sólo se puede llevar a cabo si otra opción determinada la precedió. Dicho comportamiento es conocido como aprendizaje instrumental y puede ser observado por ratones en cautiverio donde un pelet les es suministrado solamente cuando jalan una palanca.

## 6.3. Comportamiento no episódico con función de refuerzo universal

El objetivo de este experimento es contrastar la hipótesis de que es posible aprender un comportamiento no episódico para luego ser utilizado en una situación de mayor complejidad. Siempre utilizando la misma función de refuerzo universal.

### 6.3.1. Descripción del experimento

En primer lugar, se aprenderá a resolver un laberinto. Entendiéndose por laberinto, una serie de caminos, formados por pasillos, donde al menos uno parte desde el inicio y llega al objetivo. Los pasillos mencionados tienen un ancho de dos celdas en el mapa. Un ejemplo de un laberinto se puede visualizar en la Figura 5.

Es importante notar que, si bien el comportamiento de resolver un laberinto es no episódico, el aprendizaje que se realiza si lo es. El episodio arranca con el agente en un punto inicial y finaliza cuando este llega al objetivo. Esto permite exponer al agente a diferentes laberintos en cada episodio, con el fin de lograr que incorpore la mayor cantidad de casos posibles a su conocimiento.

En cuanto a la percepción del agente, el mismo cuenta con ocho sensores de proximidad que le permiten detectar la presencia de paredes en celdas cercanas. Estos sensores están dispuestos de forma uniforme alrededor del agente. Los direccionados a $0^{o}$, $90^{o}$, $180^{o}$ y $270^{o}$ con respecto a la dirección del agente, tienen un rango de sensado máximo de 1 celda (solo perciben si hay una pared en la celda adyacente). En cambio, los sensores direccionados a $45^{o}$, $135^{o}$, $225^{o}$ y $315^{o}$ con respecto a la dirección del agente tienen un rango máximo de 2 celdas. Todos los sensores

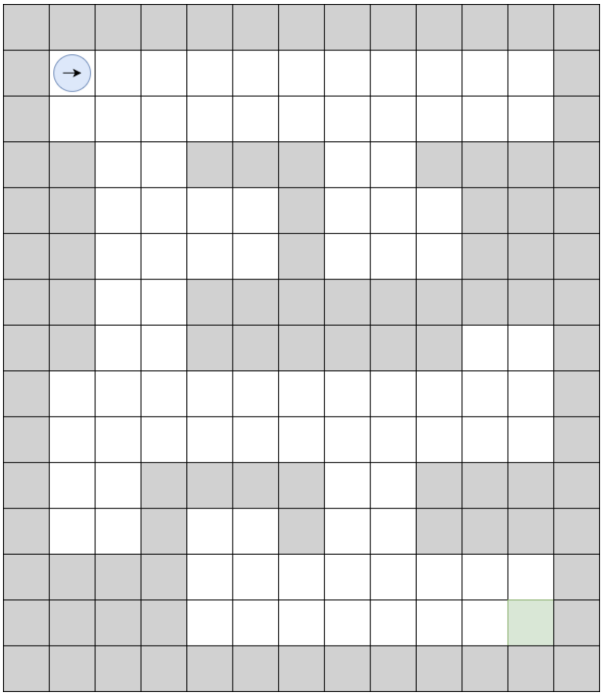


Figura 5: Ejemplo de laberinto (paredes en gris, objetivo en verde).

devuelven como valor de sensado la distancia en celdas a la pared más cercana o el valor del rango en caso de no detectar nada. Además el agente cuenta con un sensor RSSI que recibe la señal del objetivo. Este devuelve como valor la distancia, con un rango máximo de 0 (solo detecta cuando llegó al mismo), y el ángulo que forma la dirección del agente con la distancia al objetivo. Al conocer esta información, se puede finalizar el episodio cuando el agente llega al objetivo.

La opción para resolver laberintos se aprende en laberintos de 14x14 celdas como el de la Figura 6. Los mismos son generados de forma aleatoria, tanto la estructura como las posiciones iniciales y finales.

Por otro lado, este comportamiento será puesto a prueba en un ambiente de mayor complejidad donde el laberinto tiene mayor tamaño y el objetivo no se encuentra dentro del laberinto, sino que está situado en una habitación donde desemboca el laberinto.

Para poder obtener el objetivo una vez finalizado el laberinto, es de utilidad contar con una opción que permita guiar al agente al objetivo dentro de una habitación. Esta opción es aprendida en ambientes de 8x5 donde se disponen de forma aleatoria dentro del mismo, la posición inicial y el objetivo.

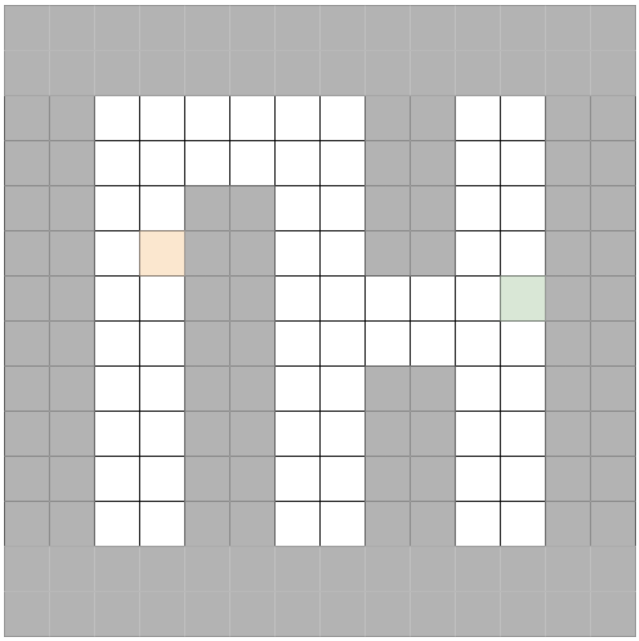


Figura 6: Ejemplo de laberinto generado (paredes en gris, objetivo en verde, inicio en amarillo).

La opción expuesta anteriormente, junto con la opción para resolver laberintos serán utilizadas luego en el ambiente de mayor tamaño descripto anteriormente. Se espera que el agente aprenda a elegir la opción de resolver el laberinto siempre que se encuentre dentro del mismo y que finalmente elija la opción de ir al objetivo cuando llegue a la habitación donde desemboca el laberinto.

Este último paso trae como desafío el hecho de que, en este caso, el laberinto no finaliza en el punto donde se encuentra el objetivo, sino antes. Para sobrellevar esta dificultad, la política que permite resolver laberintos es empaquetada en una opción `MaxStepsOption` con el fin de que finalice por cantidad de pasos, con la posibilidad de que el agente pueda volver a elegirla. Para realizar esta prueba, se modificaron algunos parámetros con respecto a los detallados en el Cuadro 1. Se incrementó la cantidad de episodios a 1500 y la cantidad de pasos máxima por episodio a 3000.

Es importante mencionar que como parte de experimentaciones auxiliares, se logró que el agente también aprenda a resolver un laberinto utilizando una función de refuerzo natural al problema. La misma recompensaba los estados en los cuales los sensores de proximidad a la derecha del agente sensaban una pared siempre que la acción sea avanzar. De esta manera, el agente aprendía a desplazarse por el laberinto manteniendo la pared por la derecha, llegando eventualmente al objetivo.

En este caso, como se mencionó en la motivación del experimento, se espera que todos los aprendizajes se realicen con la misma función de refuerzo universal definida a continuación:

$$r(s, s', o) = \begin{cases} 1 & \text{si } |s'_{goal}| = 0 \\ -1 & \text{sino} \end{cases},$$

donde $|s'_{goal}|$ es el módulo del valor del sensor de RSSI del objetivo. Es decir, la función de refuerzo solo da recompensa positiva cuando se llega al objetivo.

### 6.3.2. Resultados

Como se describió anteriormente, la primera parte del experimento consistió en aprender a resolver laberintos. Como métrica del desempeño de decidió utilizar la cantidad de pasos que le toma al agente resolver el laberinto. Esta métrica está directamente relacionada con la tarea en cuestión y permite analizar rápidamente el rendimiento del agente siguiendo la política aprendida. En la Figura 7 se puede observar un gráfico de los resultados del aprendizaje. Los mismos muestran una gran variación y cantidad de pasos elevada al comienzo del aprendizaje, lo cual es de esperar dado que no se tiene conocimiento aún. Luego del episodio 400 se nota una mejora considerable ya que se estabiliza la cantidad de pasos en un valor reducido con una variación despreciable. Esto indica que se aprendió correctamente a resolver laberintos.

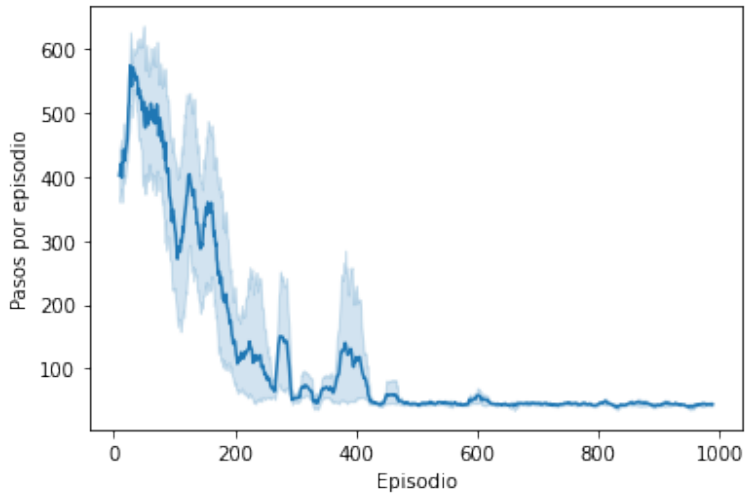



Figura 7: Pasos por episodio para aprendizaje de laberinto

El segundo paso del experimento es la evaluación del comportamiento no episódico aprendido anteriormente en un ambiente de mayores dimensiones. Como se puede observar en la

Figura 8 el agente efectivamente aprendió a resolver el ambiente de mayor complejidad propuesto, haciendo uso de conocimiento tanto episódico como no episódico.

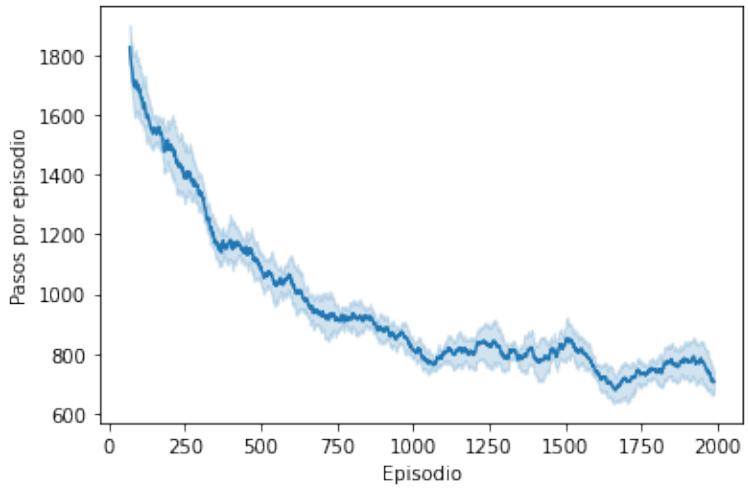



Figura 8: Pasos por episodio para aprendizaje de ambiente complejo

Como era esperable, para utilizar el comportamiento no episódico, el agente selecciona múltiples veces la misma opción hasta que finaliza el laberinto. Luego llega al objetivo utilizando la opción que resuelve esa tarea. Se puede consultar la lista de opciones elegida en el Apéndice A.

## 6.4. Refuerzo al finalizar cada opción y al finalizar cada acción

En el siguiente experimento se busca comparar qué ocurre en el caso de que se calcule el refuerzo al finalizar cada opción o se acumule luego de realizar una acción. Es importante remarcar que lo que varía es la forma en que se calcula el refuerzo ya que la actualización de Q se realiza siempre después de finalizada cada opción, tal como se explicó en la Sección 3 (Encuadre de Opciones).

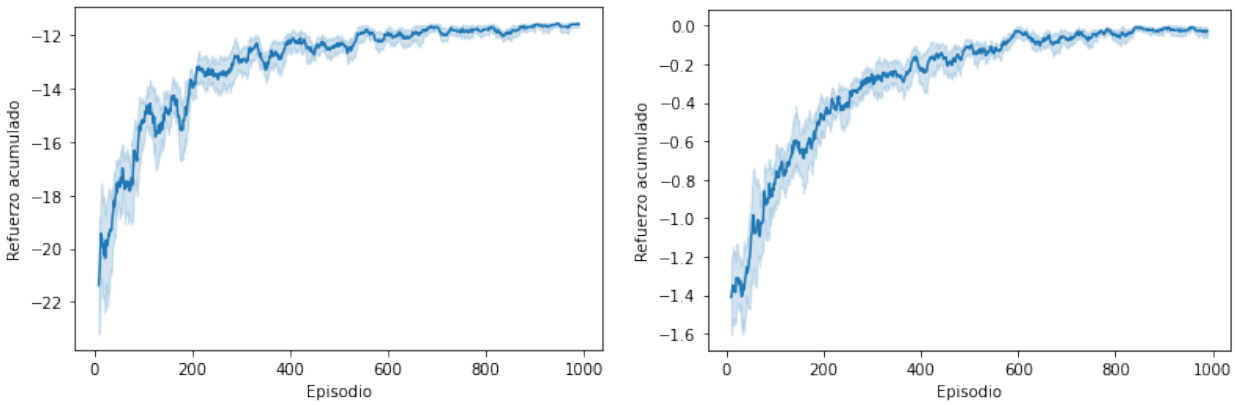
Para este experimento se utilizarán las mismas condiciones que para el experimento de la Sección 6.2, es decir, se realizará lo mismo que se realizó al comparar el uso de opciones con el uso de acciones. Claramente, se utilizará el método en el cual se utilizaron opciones. La hipótesis de este experimento es que no hay cambios mayores con respecto a la política en función de cuando se calcula el refuerzo. Es decir, no afecta el aprendizaje si el refuerzo es calculado luego de finalizar la opción o luego de cada acción.

### 6.4.1. Descripción del experimento

Al utilizar las mismas condiciones que en un experimento previo no se considera necesario explicar en mucho detalle el experimento realizado. Se utilizará el mismo ambiente descripto en la Sección 6.2. Se decidió repetir el mismo ambiente ya que en principio no es relevante el aprendizaje que se realiza sino el efecto en el mismo.

### 6.4.2. Resultados

Para analizar si el momento en que se calcula el refuerzo tiene un efecto significativo en el aprendizaje se analizará el refuerzo recibido por episodio ya que en principio se espera variación de esta métrica.




(a) Después de cada acción

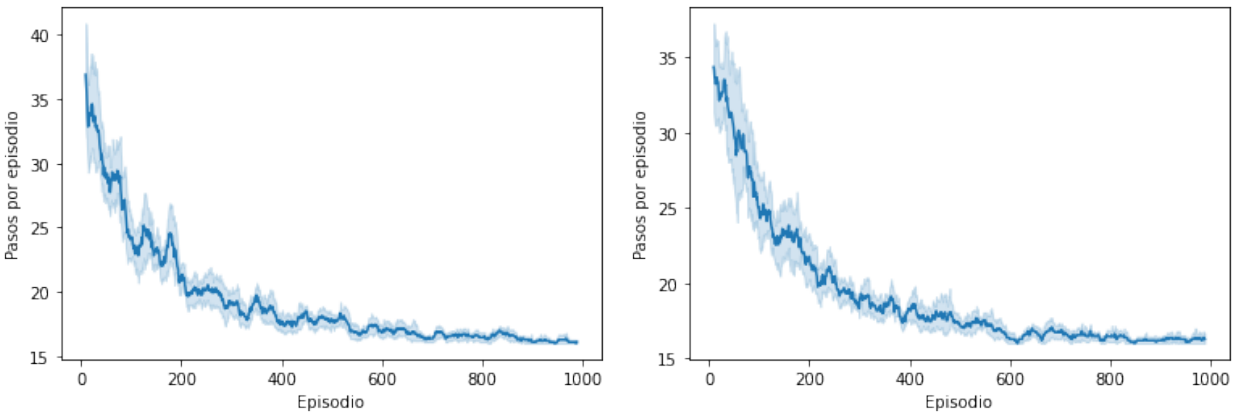
(b) Al finalizar opción

Figura 9: Comparación del refuerzo acumulado por episodio

Como se puede observar en la Figura 9 hay una diferencia entre los resultados para cada caso. De igual forma la diferencia radica en la escala de los refuerzos. Esto es de esperar ya que para el caso de opciones solo se calcula el refuerzo cuando la misma finaliza. Por otro lado, cuando se acumula el refuerzo después de cada acción el valor de refuerzo es mayor en módulo.

Si bien analizando el refuerzo acumulado ya permite observar que no hay grandes diferencias en cuanto al aprendizaje, se puede verificar esto en la cantidad de pasos por episodio. Esta métrica también permite ver el rendimiento del aprendizaje eliminando las posibles diferencias de escala.

Como se puede observar en la Figura 10 no hay diferencias notables en cuanto a la cantidad




(a) Después de cada acción

(b) Al finalizar opción

Figura 10: Comparación de la cantidad de pasos por episodio

de pasos que le toma al agente llegar a su objetivo en función del episodio del aprendizaje.

# 7. Conclusiones

En este informe se explicó desde el relevamiento hasta la implementación, experimentación y análisis de resultados que se realizó en el campo del aprendizaje por refuerzo, específicamente sobre el algoritmo de Q-Learning.

En este informe se pudo introducir al aprendizaje por refuerzo explicando algunos fundamentos importantes para la comprensión de este campo. Además, se pudo detallar sobre el encuadre de opciones y sus aplicaciones.

Se pudo concluir que utilizando Q-Learning como algoritmo para el aprendizaje por refuerzo se puede aprender tareas simples de forma realmente eficiente, como por ejemplo, llegar a un objetivo definido. Sin embargo, al querer realizar alguna tarea más compleja Q-Learning se torna ineficiente e incluso, desde el punto de vista práctico, ineficaz para los casos estudiados.

Debido a lo concluido previamente, se probó que la utilización de opciones ayuda a dar al algoritmo de Q-Learning un mayor alcance. En otras palabras, al utilizar Q-Learning con opciones se pueden realizar tareas más complejas de forma realmente simple y eficiente que con el algoritmo de Q-Learning sin estas tomaría mucho tiempo. Esto se debe a la posibilidad que brindan las opciones para subdividir tareas complejas en otras más simples. Con lo cual, se pudo comprobar nuestra primera hipótesis: es posible aprender comportamientos complejos eficientemente, de forma incremental con la ayuda de comportamientos previos.

Además, se pudo llegar a la conclusión que se puede utilizar una función de refuerzo única para el entrenamiento de un agente sin importar si la naturaleza de la tarea es episódica o no episódica. El hecho de utilizar una misma función ayuda a afrontar el problema de una forma más simple. Confirmando nuestra segunda hipótesis de que es posible indicar el objetivo del agente como parte de la definición del ambiente, manteniendo siempre la misma función de refuerzo.

Por último, se pueden proponer dos líneas de trabajo futuras diferentes basadas en lo investigado y experimentado en este informe. La primera es demostrar formalmente lo probado experimentalmente en el último experimento, es decir, que la política no varía si el refuerzo es calculado al finalizar una opción o luego de realizarse una acción. Y la segunda, probar formalmente que la función de refuerzo universal propuesta para los experimentos se puede

utilizar para cualquier entorno.

Esperamos que el trabajo sirva como una introducción al aprendizaje por refuerzo como práctica en general o para la introducción del encuadre de opciones a algún lector ya familiarizado con algunos conceptos del aprendizaje por refuerzo y ayude como disparador de nuevos experimentos e investigaciones.

# Referencias

Sutton, R. S. (2018). *Reinforcement learning: An introduction (adaptive computation and machine learning series)*. A Bradford Book. Descargado de https://www.xarg.org/ref/a/0262039249/

Sutton, R. S., Precup, D., y Singh, S. (1999). Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, *112*(1), 181 - 211. Descargado de http://www.sciencedirect.com/science/article/pii/S0004370299000521 doi: https://doi.org/10.1016/S0004-3702(99)00052-1

# A. Lista de opciones (experimento 2)

| # | Opción elegida |
|---|---|
| 1 | `GoToMaze` |
| 2 | `GoToMaze` |
| 3 | `GoToMaze` |
| 4 | `GoToMaze` |
| 5 | `GoToMaze` |
| 6 | `GoToGoal` |
| 7 | `GoToMaze` |
| 8 | `GoToMaze` |
| 9 | `GoToMaze` |
| 10 | `GoToGoal` |
| 11 | `GoToMaze` |
| 12 | `GoToMaze` |
| 13 | `GoToMaze` |
| 14 | `GoToMaze` |
| 15 | `GoToMaze` |
| 16 | `GoToGoal` |
| 17 | `GoToMaze` |
| 18 | `GoToGoal` |
| 19 | `GoToMaze` |
| 20 | `GoToMaze` |
| 21 | `GoToMaze` |
| 22 | `GoToMaze` |
| 23 | `GoToGoal` |
| 24 | `GoToGoal` |