ALAN-P: Dynamic Action Pruning for Efficient Navigation in Complex Environments

Julio Godoy, Joaquin Soto and Fernando Gutierrez

***Abstract*—Multi-agent navigation consists on efficiently moving a set of agents from start to goal locations. This is a challenging task since most environments contain static and dynamic obstacles that can significantly restrict the movement of an agent. While existing methods, such as *ALAN*, can overcome some of these limitations by increasing the action space of the agents, their behavior can be suboptimal in many of the situations that agents can find themselves into. In this work, we propose *ALAN-P*, a multi agent local navigation method based on the *ALAN* framework, which improves the agent's behavior by dynamically adapting the action space to the agent's local conditions. The results of our experiments show that the proposed *ALAN-P* can lead to significant performance improvements over *ALAN* in a variety of challenging environments.**

***Index Terms*—multi agent systems, navigation, multi agent coordination.**

I. Introducción

El problema de navegación de múltiples agentes consiste en mover, de forma eficiente y sin colisiones, a un conjunto de agentes desde una posición inicial a una posición objetivo. Esta tarea puede ser realizada bajo un enfoque *centralizado*, donde una entidad determina y coordina los movimientos o acciones de cada uno de los agentes del conjunto. Sin embargo, debido al alto costo computacional del cálculo de las rutas óptimas para los agentes (*P-SPACE hard* [1]), en las últimas décadas se han propuesto múltiples métodos basados en un enfoque *distribuido* [2], [3], [4]. En este enfoque, cada agente determina sus movimientos de forma independiente y utilizando información local, para llegar a su objetivo. Los métodos distribuidos pueden escalar en número de agentes sin dificultades [2], [3]. No obstante, estos métodos también pueden producir situaciones de congestión o *deadlocks* en ambientes restringidos, cuando el desplazamiento de algunos agentes interfiere con otros [5]. Para evitar estas demoras, métodos como *C-Nav* [6], [7] buscan predecir las acciones futuras de agentes cercanos para evitar aglomeración, suponiendo que los agentes pueden comunicarse entre ellos. Por otra parte, *ALAN* [8], utiliza aprendizaje por refuerzo para permitir a los agentes escoger acciones de manera independiente, sin requerir algún tipo de comunicación entre ellos. Sin embargo, dada la naturaleza probabilística de *ALAN*, los agentes son propensos a tener un comportamiento sub-óptimo y errático, lo que se traduce en demoras innecesarias en la navegación.

Julio Godoy, Joaquin Soto Department of Computer Science, Universidad de Concepcion, Chile {jonarodriguez, juliogodoy}@udec.cl
Fernando Gutierrez AIDA SpA, Chile fernando.a.gutierrez.gomez@gmail.com

En este trabajo, proponemos un método de navegación local para múltiples agentes llamado *ALAN-P*. Basado en la mecánica de selección de acciones detrás de *ALAN*, nuestra propuesta permite a cada agente escoger de manera inteligente sus movimientos a través de la eliminación de acciones *garantizadas* en ser sub-óptimas, dentro del conjunto de acciones disponibles. La *poda* de estas acciones se obtiene al considerar la última acción realizada y su recompensa. Si una nueva acción no puede retornar al agente una recompensa mayor a la acción previa, es eliminada. Esto permite a los agentes usando *ALAN-P* formular el proceso de toma de acciones como una cadena de Markov, donde la acción a escoger depende sólo de la última acción escogida. *ALAN-P* fue evaluado en ocho escenarios simulados, que enfatizan la restricción de movimiento para los agentes en distintas formas. Nuestra propuesta ofrece claras mejoras sobre *ALAN*, con distintos niveles de eficiencia según cada escenario.

Este trabajo está organizado de la siguiente forma: en la sección II se describen trabajos relacionados a navegación multi-agentre. En la sección III se describe *ALAN*, el método base sobre el cual se desarrolló este trabajo. En la sección IV se presenta y describe el método propuesto. La sección V describe los experimentos para evaluar, mientras que en la sección VI se presentan y analizan su resultados. Finalmente, se presentan las conclusiones del trabajo en la sección VII.

II. Trabajos relacionados

El problema de la navegación multi-agente consiste en un conjunto de agentes $\mathcal{A}_1, \ldots, \mathcal{A}_n$, donde cada agente $\mathcal{A}_i$ tiene una posición inicial y una posición final diferentes a las de $\mathcal{A}_j$ $(i \neq j)$, en un ambiente 2D. Cada agente debe moverse entre estas dos posiciones en el menor tiempo posible y sin colisiones. En cada instante de tiempo, el agente $\mathcal{A}_i$ tiene una posición p_i, una velocidad $\mathbf{v}_i$, y una velocidad máxima $v_i^{\text{máx}}$.

En nuestro problema, cada agente $\mathcal{A}_i$ tiene a su disposición un conjunto de *acciones* o *velocidades preferidas* $\mathbf{v}_i^{\text{pref}}$ (Fig. 2(b)), que utiliza para movilizarse hacia su objetivo [8]. La $\mathbf{v}_i^{\text{pref}}$ ejecutada en un tiempo t corresponde a la *intención de movimiento* de $\mathcal{A}_i$ en ese instante de tiempo. Al mismo tiempo, $\mathcal{A}_i$ también posee una *velocidad preferida hacia el objetivo* $\mathbf{v}_i^{\text{goal}}$, dirigida en todo momento al objetivo del agente. En la ausencia de otros agentes y obstáculos, $\mathbf{v}_i^{\text{pref}} = \mathbf{v}_i^{\text{goal}}$.

Para resolver el problema de la navegación multi-agente se han propuesto distintos paradigmas, desde el comportamiento emergente a partir de reglas sencillas (*boids*) [2], a modelos que consideran reglas cognitivas [9], factores psicológicos o sociológicos [10], [11], y fuerzas sociales [3]. Durante la

última década, los métodos basados en el concepto de *velocity obstacles* [12] adquirieron gran popularidad, mayoritariamente por su variante *ORCA* (Optimal Reciprocal Collision Avoidance) [13]. *ORCA* es un método predictivo de evasión de colisiones para la navegación de múltiples agentes, utilizado para la simulación de peatones [14], [15] y la navegación de robots [16] [17], [18], entre otras aplicaciones [19]. Un agente que usa *ORCA* intenta navegar hacia su objetivo con $\mathbf{v}^{\text{pref}}$ y proyecta las posiciones de los agentes cercanos en un horizonte temporal, en base a su velocidad observada. Si se detecta una posible colisión futura entre agentes, *ORCA* opera bajo el concepto de *reciprocidad*, es decir, que los agentes potencialmente involucrados en una colisión comparten la tarea de evadirla. En particular, *ORCA* calcula una *velocidad libre de colisión* $\mathbf{v}^{\text{new}}$ para cada uno de los agentes involucrados, que está garantizada de evitar colisiones en el horizonte temporal considerado.

Otros enfoques que han sido utilizados por un número significativo de métodos de navegación derivan del *aprendizaje por refuerzo* (RL, por sus siglas en inglés) [20], [21]. Bajo una estrategia de RL, los agentes pueden encontrarse en uno de varios posibles estados, y aprenden el valor de las acciones que realizan en cada estado en base a *recompensas*.

En el caso de sistemas de múltiples agentes, los métodos basados en RL deben considerar el ambiente no estacionario para cada agente. Esto puede incrementar significativamente el número de posibles estados en los que se puede encontrar cada agente y que deben ser explorados lo suficiente para poder aprender la mejor acción en cada caso [21], [22]. Para reducir la complejidad, el problema de aprendizaje puede ser formulado sin estados, como un *problema de bandido multi-brazos* [23]. En esta formulación, cada agente debe escoger su movimiento o acción en base a las recompensas obtenidas anteriormente con cada acción. Bajo esta formulación, el balance entre la explotación de la mejor acción actual y la exploración de otras acciones potencialmente mejores es crítica [24], [25]. Una forma de considerar este balance es a través de la política de selección de acciones *Softmax* [26], un método probabilístico en el que la probabilidad de escoger una acción está directamente relacionada con la relación entre los valores estimados para cada una.

III. *ALAN*: Adaptive Learning for Multi-Agent Navigation

Nuestra propuesta usa elementos de *ALAN* [8], un método de aprendizaje de movimientos eficientes para la navegación de múltiples agentes. *ALAN* formula el problema de selección de acciones para cada agente como un problema de RL sin estado, basado en el *bandido multi-brazo* [23], y utiliza *ORCA* para la evasión de colisiones.

El método *ALAN* (Algoritmo 1) está compuesto de un *método de selección de acciones*, una *función de recompensa* y una *ventana temporal* de recompensas. Estos elementos son ejecutados por cada agente en cada uno de los ciclos del método, hasta que llega a su posición objetivo. Primero (línea 4), cada agente selecciona una acción a del conjunto de acciones disponibles mediante la función $ChosenAct$ en el tiempo t. Segundo, a partir de la acción escogida, se calcula una velocidad nueva sin colisiones $\mathbf{v}^{\text{new}}$ (mediante ORCA) que el agente usará para moverse durante su siguiente ciclo (líneas 6-8) a su nueva posición $\mathbf{p}$ en el tiempo $t+1$ (línea 9). Por último, luego de moverse, cada agente evalúa la acción ejecutada mediante una función de recompensa (líneas 11-13, explicada con detalle en la sección B), y calcula el valor estimado de cada acción para escoger aquella a ejecutar durante el siguiente ciclo. Los componentes de *ALAN* se describen a continuación.

Algorithm 1: The ALAN algorithm for an agent

1: initialize simulation
2: **while** not at the goal **do**
3: **if** $UpdateAction(t)$ **then**
4: $a^t \leftarrow ChosenAct(t)$
5: **end if**
6: $\mathbf{v}^{\text{pref}} \leftarrow getVpref(a^t)$
7: $\mathbf{v}^{\text{new}} \leftarrow ORCA(\mathbf{v}^{\text{pref}})$
8: $Execute(\mathbf{v}^{\text{new}})$
9: $\mathbf{p}^{t+1} \leftarrow \mathbf{p}^t + \mathbf{v}^{\text{new}} \cdot \Delta t$
10: $t \leftarrow t+1$
11: $\mathcal{R}_a^{goal} \leftarrow GoalReward(a^{t-1})$ (cf. Eq. 3)
12: $\mathcal{R}_a^{polite} \leftarrow PoliteReward(a^{t-1})$ (cf. Eq. **??**)
13: $\mathcal{R}_a \leftarrow (1-\gamma) \cdot \mathcal{R}_a^{goal} + \gamma \cdot \mathcal{R}_a^{polite}$
14: **end while**

A. Selección de Acciones

En *ALAN*, cada agente escoge su próxima acción mediante el método $Softmax$. Este método usa la distribución de Boltzmann para asignar una probabilidad de selección a cada acción. En concreto, la probabilidad de escoger la acción a, del conjunto $Actions$ de acciones, esta dada por la siguiente ecuación [26]:

$$Softmax(a, Actions, \tau) = \frac{\exp\left(\frac{\mathcal{R}_a}{\tau}\right)}{\sum_{a=1}^{|Actions|} \exp\left(\frac{\mathcal{R}_a}{\tau}\right)} \quad (1)$$

donde $\mathcal{R}_a$ es el valor de la recompensa de la acción a. La variable de *exploración* τ fue fijada, en *ALAN*, en $0,2$ [8].

B. Función de Recompensa

Un agente *ALAN* calcula la recompensa de la última acción ejecutada en base a dos criterios: cuánto acerca al agente a su objetivo, y su influencia en el movimiento de los agentes cercanos. El primer criterio motiva a los agentes a llegar a sus objetivos, mientras que el segundo les motiva a ejecutar acciones que no dificulten el progreso de los otros agentes. Para este último criterio los agentes utilizan la propiedad de reciprocidad de *ORCA*: cuando una colisión es posible entre dos agentes, ambos se desviarán para evitarla. Por lo tanto, si *ORCA* determina para un agente una velocidad libre de colisión muy diferente a su velocidad preferida, esto implica que también otro agente deberá desviarse para evadir la colisión. Para minimizar el impacto de las decisiones de cada agente en sus agentes cercanos, se motiva la selección de acciones

cuya $\mathbf{v}^{\text{new}}$ es similar a su $\mathbf{v}^{\text{pref}}$, es decir, aquellas acciones que son *permitidas* por *ORCA*. Se define la recompensa $\mathcal{R}_a$ de una acción a como sigue:

$$\mathcal{R}_a = (1-\gamma)\cdot\mathcal{R}_a^{goal} + \gamma\cdot\mathcal{R}_a^{polite} \tag{2}$$

donde $\mathcal{R}_a^{goal}$ evalúa el progreso hacia el objetivo del agente en el último paso. Este componente favorece movimientos que acercan al agente a su objetivo. Por otro lado, $\mathcal{R}_a^{polite}$ evalúa el impacto de la acción a en los agentes cercanos, y se calcula comparando la velocidad preferida $\mathbf{v}^{\text{pref}}$, correspondiente a la acción a escogida por el agente, y la velocidad libre de colisión, $\mathbf{v}^{\text{new}}$ calculada por ORCA. Más formalmente:

$$\mathcal{R}_a^{goal} = \mathbf{v}^{\text{new}}\cdot\frac{\mathbf{g}-\mathbf{p}}{\|\mathbf{g}-\mathbf{p}\|};\;\; \mathcal{R}_a^{polite} = \mathbf{v}^{\text{new}}\cdot\mathbf{v}^{\text{pref}} \tag{3}$$

donde $\mathbf{p}$ y $\mathbf{g}$ corresponden a la posición actual del agente y la de su objetivo, respectivamente. El *factor de coordinación* γ controla la influencia de cada uno de estos componentes en el valor final de la recompensa. En *ALAN*, γ fue experimentalmente fijado en 0,5 [8].

C. Ventana Temporal de las Recompensas

ALAN calcula un *valor estimado* por cada acción en base al promedio de las recompensas obtenidas en una ventana de tiempo limitada. Usar este enfoque evita que las recompensas obtenidas con mucha anterioridad impacten negativamente en la selección de acciones y, por el otro lado, ayuda a que la estimación del valor de cada acción no cambie de manera abrupta. Si una acción no es ejecutada durante el tiempo considerado en la ventana, su valor estimado es cero, representando una estimación no sesgada del mismo. En *ALAN*, el tamaño de la ventana fue experimentalmente fijado en 2 segundos.

Como resultado de lo anterior, los agentes que usan *ALAN* demuestran movimientos más eficientes en una variedad de escenarios, comparados con *ORCA* [8]. Sin embargo, dada la naturaleza probabilística de la selección de acciones, los movimientos efectuados por los agentes *ALAN* son sub-óptimos en muchos casos. Esto implica que, por ejemplo, en condiciones de nula congestión los agentes periódicamente escogen acciones que los alejan de sus objetivos. Si bien este comportamiento ayuda a que los agentes *ALAN* exploren acciones que pudiesen tener una mejor recompensa en situaciones de congestión, esto también se traduce en que, incluso cuando un agente esté libre para moverse hacia su objetivo, pueda escoger acciones como alejarse o moverse perpendicularmente con respecto a su objetivo. Además, este comportamiento irregular aumenta el ruido en el proceso de aprendizaje de los otros agentes.

IV. *ALAN-P*

En este trabajo proponemos *ALAN-P*, un método de navegación local para múltiples agentes, que no asume ningún tipo de comunicación entre ellos. Nuestro método permite un comportamiento adaptativo, dependiendo de las condiciones locales en que se encuentren los agentes. Para lograr esto, *ALAN-P* evita la elección de acciones que están garantizadas a no mejorar la navegación local de cada agente, es decir, son garantizadas en ser sub-óptimas. Esto se traduce en: (1) movimientos más eficientes, (2) reducción de la dependencia en las recompensas históricas y, finalmente, (3) una disminución del tiempo de viaje de los agentes, comparado con *ALAN* y *ORCA*.

Para lograr lo anterior, *ALAN-P* incorpora las técnicas de *poda dinámica de acciones* y *modelo Markoviano de selección de acciones*. Estas técnicas buscan limitar, de manera inteligente, la cantidad de acciones disponibles a los agentes y la forma en que estiman el valor de cada acción. Los detalles de las dos técnicas propuestas son presentados a continuación.

A. Poda Dinámica de Acciones

Como se mencionó anteriormente, una de las desventajas de *ALAN* es la naturaleza probabilística de la selección de acciones. Independiente del estado actual del agente, siempre habrá una probabilidad mayor a cero de escoger la acción con menor *valor estimado*. Como la estimación del valor de cada acción se basa en recompensas históricas, el agente no puede determinar con certeza cuál es la acción localmente óptima antes de ejecutarla en cada paso.

En *ALAN-P* se propone una mejora en el comportamiento en la navegación de los agentes. Esto se logra reduciendo el número de acciones disponibles para cada agente, descartando aquellas que están garantizadas a ser localmente sub-óptimas.

Para determinar qué acciones caben dentro de esta categoría, cada agente compara las últimas recompensas obtenidas con una estimación de la máxima recompensa $\mathcal{R}_a^{max}$, de cada acción a. Aquellas acciones cuyos valores máximos estimados sean menores a los obtenidos recientemente por el agente, son descartadas o podadas del conjunto de acciones disponibles. De esta forma, en la siguiente selección de acción, sólo serán consideradas aquellas acciones cuya máxima recompensa estimada es mayor a la recompensa obtenida por el agente con la última acción escogida.

Primero, cada agente calcula $\mathcal{R}_a^{free}$, el valor de recompensa obtenido bajo el supuesto de que la acción a pueda ejecutarse sin ninguna restricción. El valor de $\mathcal{R}_a^{free}$ se desglosa en $\mathcal{R}_a^{goal_{free}}$ y $\mathcal{R}_a^{polite_{free}}$:

$$\mathcal{R}_a^{goal_{free}} = \mathbf{v}^{\text{pref}}\cdot\frac{\mathbf{g}-\mathbf{p}}{\|\mathbf{g}-\mathbf{p}\|};\;\; \mathcal{R}_a^{polite_{free}} = \mathbf{v}^{\text{pref}}\cdot\mathbf{v}^{\text{pref}} \tag{4}$$

que se diferencia de la Ecuación 3 en que $\mathbf{v}^{\text{new}}$ se reemplaza por $\mathbf{v}^{\text{pref}}$, ya que se asume que la velocidad libre de colisión del agente, en una situación ideal, es igual a su velocidad preferida. Luego, $\mathcal{R}_a^{free}$ se calcula de la siguiente manera:

$$\mathcal{R}_a^{free} = (1-\gamma)\cdot\mathcal{R}_a^{goal_{free}} + \gamma\cdot\mathcal{R}_a^{polite_{free}} \tag{5}$$

Cuando $\mathcal{R}_a^{free} \geq 0$, este valor es también el máximo a obtener con la acción a. Sin embargo, cuando $\mathcal{R}_a^{free} < 0$ esto no es así, ya que la acción a puede resultar en una recompensa mayor a $\mathcal{R}_a^{free}$. Un ejemplo de lo anterior es cuando el agente no se puede mover, ya que su recompensa será $\sim$0. Por lo tanto, una vez calculado $\mathcal{R}_a^{free}$, podemos calcular el valor máximo estimado para cada acción a, $\mathcal{R}_a^{max} \leftarrow$

$\max\{0, \mathcal{R}_a^{free}\}$. Esto permite que en situaciones cuando el agente reciba recompensas muy bajas (negativas), cualquier acción esté disponible para ser escogida a continuación.

El proceso de poda completo se presenta en el Algoritmo 2, enmarcado en la función $ChosenAct$. Una nueva lista $candAct(t)$ es creada para almacenar las acciones que no serán podadas (línea 2). Luego, la recompensa $\mathcal{R}_a^{max}$ de cada acción a es calculada y comparada, en cada ciclo, con el valor de recompensa obtenido por la última acción escogida, $\mathcal{R}_{\mathbf{a^{t-1}}}$ (línea 4). Las acciones a considerar luego de realizar esta selección, en el tiempo t, se agrupan en $candAct(t)$ (línea 5), $candAct(t) \subseteq Actions$. Una recompensa de 0 es asignada a aquellas acciones en $candAct(t)$, con excepción de la última acción escogida (línea 10, explicado en la sección B). Finalmente, mediante el método de selección de acciones basado en $Softmax$, la acción a ser ejecutada en el siguiente paso es escogida (líneas 13-16).

Con lo anterior, solo resultan elegibles aquellas acciones que tienen una probabilidad mayor a cero de mejorar la recompensa que recibe el agente, mejorando la eficiencia en la navegación local de los agentes.

Algorithm 2: ALAN-P $ChosenAct$ for time t

1: input: $\mathcal{R}_{\mathbf{a^{t-1}}}$, reward for action chosen at $t-1$, $\mathbf{a^{t-1}} = ChosenAct(t-1)$
2: $candAct(t) \leftarrow$ empty list
3: **for** $a' \in Actions$, $a' \neq \mathbf{a^{t-1}}$ **do**
4: **if** $\mathcal{R}_{a'}^{max} > \mathcal{R}_{\mathbf{a^{t-1}}}$ **then**
5: add action a' to $candAct(t)$
6: **end if**
7: **end for**
8: **for** $a' \in candAct(t)$ **do**
9: **if** $a' \neq \mathbf{a^{t-1}}$ **then**
10: $\mathcal{R}_{a'} = 0$
11: **end if**
12: **end for**
13: **for** $a \in candAct(t)$ **do**
14: compute $Softmax(a, candAct(t), 0)$
15: **end for**
16: $\mathbf{a^t} \leftarrow$ sample from $Boltzmann$ distribution
17: return $\mathbf{a^t}$

B. Modelo Markoviano de Selección de Acciones

Como se explicó en la Sección III, una de las características de *ALAN* es que mantiene una ventana de tiempo de dos segundos de las últimas recompensas obtenidas para evaluar cada acción. Por un lado, utilizar una menor cantidad de información temporal (una ventana de menor tamaño) aumenta la incertidumbre del agente respecto del valor real de la acción y del valor de otras acciones, al estar muy sesgado por la poca información disponible. Por otro lado, utilizar una mayor cantidad de información temporal (una ventana de mayor tamaño) impide a los agentes adaptarse rápidamente a los cambios en sus condiciones locales de navegación, los que invalidarían las recompensas obtenidas con anterioridad.

El uso de la *poda dinámica de acciones* en *ALAN-P* funciona como un filtro temporal que reduce la incertidumbre del agente respecto de las acciones más convenientes en cada paso, reduciendo a cero la probabilidad de escoger acciones garantizadamente sub-óptimas. Por otro lado, para la acción que actualmente se ejecuta, utilizar sólo la última recompensa obtenida en su evaluación permite mantener una estimación actualizada no sólo de su valor, sino también como una *observación* del estado local del agente al momento de ejecutar la acción. De esta manera, podemos formular el proceso de selección de acciones a través del tiempo como una cadena de Markov, en la cual la mejor acción a tomar para cada paso se calcula en base solamente a la recompensa obtenida en el paso anterior, mientras que el agente *olvida* inmediatamente las recompensas obtenidas por las otras acciones, reflejando la incertidumbre sobre su validez en las condiciones locales actuales (Algoritmo 2, línea 10).

La Figura 1 muestra un diagrama de los 6 estados en que puede estar un agente en el modelo Markoviano de selección de acciones, junto con las transiciones posibles. Las probabilidades de transición entre estados se determinan de manera dinámica en base a (1) la última recompensa obtenida, y (2) los parámetros de la función $Softmax$ que utiliza esta recompensa para determinar las probabilidades de transición. La Figura 1(b) muestra a 3 estados con valores de ejemplo para las probabilidades de transición entre uno y otro estado.

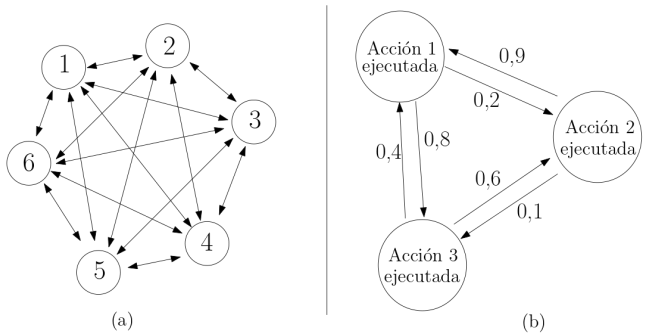
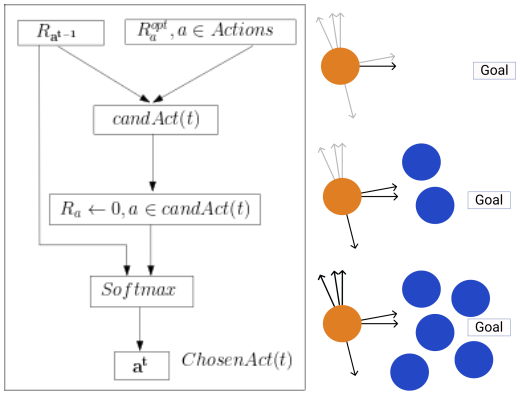


Fig. 1. (a) Diagrama de estados de la cadena de Markov utilizada en *ALAN-P*. (b) Parte del diagrama de estados y transiciones de ejemplo del modelo Markoviano de selección de acciones.

La Figura 2 (a), correspondiente a la función $ChosenAct(t)$, muestra las relaciones entre distintas variables que conlleva al cálculo de la acción escogida para ser ejecutada en el paso t, $\mathbf{a^t}$. Esta elección depende de la recompensa obtenida por la acción ejecutada en el último paso, es decir, de $R_{\mathbf{a^{t-1}}}$ (resultado de $ChosenAct(t-1)$), como así también de las acciones consideradas luego de la poda de acciones ($candAct$). La Figura 2 (b) muestra las acciones podadas por *ALAN-P* (en gris), y aquellas disponibles para ser escogidas (en negro), en diferentes condiciones de congestión del agente.

V. Experimentos

En la presente sección, entregamos detalles de la evaluación del desempeño de *ALAN-P*. Estos detalles consideran condiciones de ejecución de las evaluaciones, la métrica de evaluación y los métodos comparados.



(a) (b)

Fig. 2. (a) Diagrama del flujo en la función $ChosenAct(t)$ en *ALAN-P*. (b) Conjunto de velocidades preferidas o *acciones* disponibles en *ALAN-P*, para un agente (naranjo) en tres situaciones locales: libre de congestión (arriba), congestión entre el agente y su objetivo (medio) y congestión al frente y a los lados (abajo). Las flechas en gris indican las acciones podadas en cada situación local del agente, y los círculos azules representan a otros agentes.

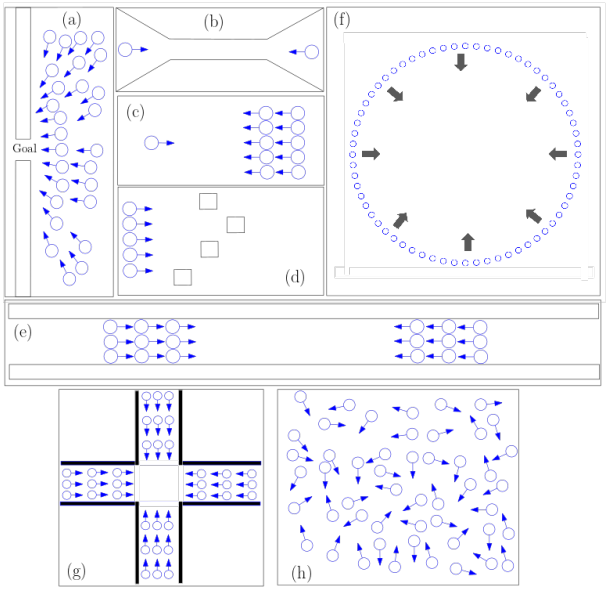


Fig. 3. Escenarios utilizados para la experimentación de *ALAN-P* : (a) CONGESTED, (b) DEADLOCK, (c) INCOMING, (d) BLOCKS, (e) BIDIRECTIONAL, (f) CIRCLE, (g) INTERSECTION y (h) CROWD.

Escenarios de evaluación. Para poder determinar de manera empírica la efectividad de el método propuesto, hemos ejecutado *ALAN-P* en ocho escenarios simulados. Estas simulaciones fueron implementadas en $C++$ usando la biblioteca RVO2 [27]. Los escenarios de evaluación, propuestos por Godoy *et al* [8] (Figura 3) son descritos a continuación:

- CONGESTED: 32 agentes que deben salir de un área a través de una apertura en el medio, la que sólo permite el tránsito de un agente a la vez, representado en la Figura 3(a).
- DEADLOCK: Dos grupos de cinco agentes deben cruzar un área estrecha en dirección opuesta, representado en la Figura 3(b).
- INCOMING: Un agente se mueve en curso de colisión con un grupo de 15 agentes, y debe evitarlos antes de ser arrastrado por el grupo, representado en la Figura 3(c).
- BLOCKS: Cinco agentes que se mueven en dirección paralela deben sortear un grupo de obstáculos en su camino, representado en la Figura 3(d).
- BIDIRECTIONAL: dos grupos de 9 agentes se mueven en dirección opuesta, dentro de un área delimitada por dos paredes laterales, representado en la Figura 3(e).
- CIRCLE: 64 agentes deben moverse a sus posiciones antípodas en un círculo, representado en la Figura 3(f).
- INTERSECTION: 80 agentes son divididos en cuatro grupos que son ubicados en las extremidades del escenario. Los agentes son orientados, por su objetivo, hacia el grupo opuesto, representado en la Figura 3(g).
- CROWD: 300 agentes, con ubicación inicial y objetivo aleatoria, se mueven en un cuadrilátero delimitado por paredes, representado en la Figura 3(h).

Métrica de evaluación. En este trabajo, hemos utilizado *interaction overhead* (IO) [8], [6] como métrica de evaluación. Independiente de la estructura del escenario, IO captura las interacciones entre un conjunto de agentes $\mathcal{A}$ de la siguiente forma:

$$IO(\mathcal{A}) = TTime(\mathcal{A}) - MinTTime(\mathcal{A}) \tag{6}$$

donde $TTime$ corresponde al tiempo de viaje de $\mathcal{A}$ en el escenario. $MinTTime$, por otra parte, es el tiempo mínimo que $\mathcal{A}$ requiere para llegar a su objetivo, libres de interacciones y a maxima velocidad. Tanto $TTime$ como $MinTTime$ toman en consideracion el tiempo promedio de viaje μ de $\mathcal{A}$ y su dispersión σ:

$$TTime(\mathcal{A}) = \mu(GT(\mathcal{A})) + 3\sigma(GT(\mathcal{A})) \tag{7}$$

$$MinTTime(\mathcal{A}) = \mu(MinGT(\mathcal{A})) + 3\sigma(MinGT(\mathcal{A})) \tag{8}$$

donde GT corresponde al tiempo de navegación real de los agentes de $\mathcal{A}$, mientras que $MinGT$ es el tiempo mínimo de navegación para $\mathcal{A}$. Al considerar el tiempo mínimo necesario para que los agentes en $\mathcal{A}$ logren llegar a sus objetivos en un escenario, IO puede mostrar con mayor claridad cuánto tiempo los agentes dedican en resolver interacciones y obstáculos.

Métodos comparados. Para poder contextualizar adecuadamente los resultados de la evaluación de *ALAN-P*, hemos considerado dos tipos de evaluación. El primer tipo de evaluación se enfoca en determinar cómo se compara la efectividad del método propuesto frente a otros métodos establecidos en la literatura. Para esto, hemos considerado como métodos a comparar a *ORCA* y *ALAN*. En el segundo tipo de evaluación, se determinará el efecto que tiene cada uno de los componentes de *ALAN-P* en su desempeño.

En todas las evaluaciones, consideramos los valores de IO obtenidos con cada uno de los métodos, al resolver cada uno de los escenarios previamente descritos, luego de 100 iteraciones.

VI. RESULTADOS Y DISCUSIÓN

En esta sección se presentan y discuten los resultados de la evaluación de *ALAN-P*.

A. *Comparación con otros Métodos*

Como se puede apreciar en la Figura 4, *ALAN-P* presenta un rendimiento mucho mejor en comparación con *ORCA* y *ALAN* en todos los escenarios, con una diferencia estadísticamente significativa ($p < 0,001$). Esto indica que restringir las acciones disponibles a los agentes, eliminando acciones que afectarían negativamente el tiempo de viaje de los agentes, tiene un claro efecto positivo en la eficiencia de su navegación.

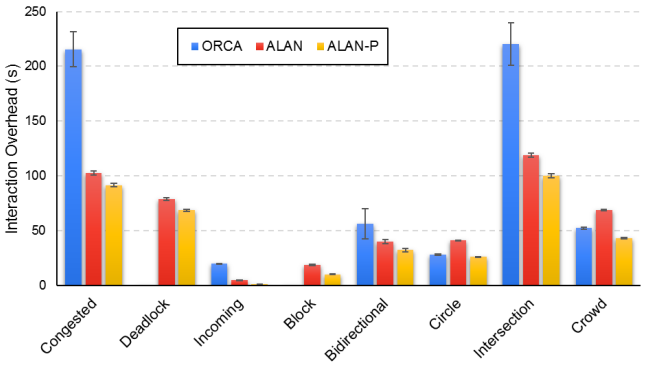


Fig. 4. Interaction Overhead de ORCA, *ALAN* y *ALAN-P*, en los escenarios presentados en la Figura 3.

Cabe destacar que *ALAN-P* obtuvo mejor desempeño que *ORCA* incluso en los escenarios *Crowd* y *Circle*, donde este último demostró mejores resultados que *ALAN*. En *Circle*, los agentes que usan *ALAN* demuestran movimientos poco eficientes una vez que se ha resuelto la congestión en el medio del escenario. En comparación, los agentes usando *ALAN-P* fueron capaces de resolver eficientemente la congestión, evitando la selección de acciones sub-óptimas. En el caso del escenario *Crowd*, los movimientos sub-óptimos de *ALAN* introducen ruido en las percepciones de los agentes. Mediante la *poda de acciones*, *ALAN-P* disminuye la variabilidad en los movimientos, que se traduce en menos ruido y mayor estabilidad en la trayectoria hacia el objetivo.

La Tabla I muestra la diferencia entre *ALAN* y *ALAN-P* en base al tiempo de CPU (segundos) que toma, en promedio, llevar a todos los agentes a su destino. Se puede observar que se mantiene la misma tendencia visible en la Figura 4.

B. *Análisis de Factor de Coordinación*

Un factor clave en el desempeño de *ALAN-P* es el *factor de coordinación* γ. Un valor muy bajo de este hiperparámetro se traduce en un comportamiento más *agresivo* y *egoísta* de parte de los agentes, mientras que un valor muy alto se traduce en un comportamiento muy conservador. En la Figura 5 se muestra, por escenario, el efecto que tiene la variación de γ en el IO de *ALAN-P*. En general, se puede observar que cuando este valor es muy alto ($\gamma > 0,7$) se observa un efecto negativo en el desempeño del método. Un γ alto provoca que los agentes posterguen su movimiento hacia el objetivo hasta que haya disminuido significativamente la interacción con otros agentes. Por otro lado, para valores de γ mas pequeños, se observa un mejor desempeño en varios escenarios. Sin embargo, a nivel general, los mejores resultados se obtuvieron con $\gamma = 0,5$, donde el desempeño fue eficiente en todos los escenarios.

Un caso particular se da en el escenario *Intersection*, donde la IO empeora significativamente con valores de γ entre 0,2 y 0,4. En dicho rango, la combinación de factor de coordinación, el espacio de acciones y los obstáculos del escenario se conjugan para demorar la llegada al objetivo de los últimos uno o dos agentes en el escenario. En concreto, los últimos agentes quedan detrás de una pared y alternan movimientos que los acercan y alejan de su objetivo, lo que se traduce en mayores tiempos de viaje. La Figura 6 muestra en línea contínua las acciones que el agente alterna en su ejecución, y en líneas punteadas las acciones podadas.

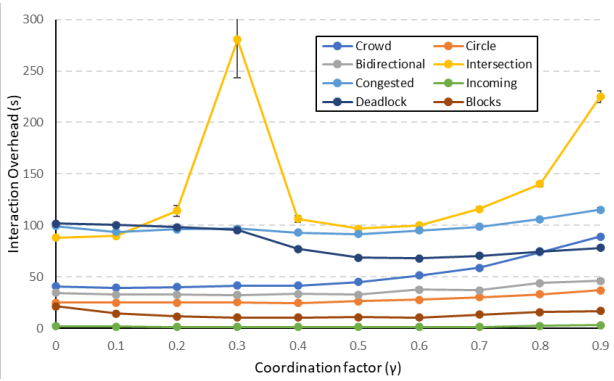


Fig. 5. Interaction overhead de *ALAN-P* para cada escenario variando el factor de coordinación (γ).

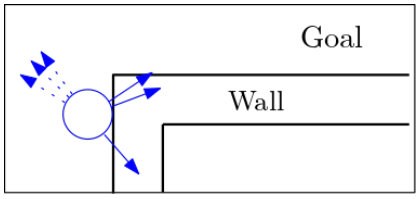


Fig. 6. Acciones disponibles (líneas contínuas) y podadas (líneas punteadas) para el agente detrás de la pared con un factor de coordinación (γ) de 0.3.

C. *Análisis de Componentes de ALAN-P*

Para poder determinar el impacto real que tiene cada uno de los componentes de *ALAN-P*, se evaluó el desempeño en IO(s) con y sin cada uno de estos.

Resultados de la estimación Markoviana de estado: Para evaluar este componente, se consideró el resultado en IO (s) con diferentes tamaños de ventana de tiempo de recompensas obtenidas. El mínimo valor a considerar corresponde a recordar sólo la última acción, es decir, corresponde a la versión propuesta de *ALAN-P*. Como muestra la Figura 7, al considerar un historial de recompensas de distintos tamaños para *ALAN-P*, este no presenta mejores resultados al aumentar el tamaño de la ventana de tiempo. Es más, su IO tiende a empeorar en varios escenarios, siendo *Intersection* el caso mas notorio. En estos casos, cuando los agentes llegan a una zona de congestión, solo tienen registros de ir hacia su objetivo (dentro de su ventana de tiempo hay mayoritariamente recompensas

de acciones que van en la dirección del objetivo del agente). Consecuentemente, los agentes continúan tratando de ir hacia su objetivo hasta que las recompensas en su ventana de tiempo bajan, aumentando la probabilidad de elegir acciones que antes no se habían escogido, entre ellas las que les permitirían evadir la congestión. Este comportamiento se traduce finalmente en un retraso en los tiempos de los agentes.

Por el otro lado, en varios escenarios los mejores resultados son obtenidos considerando sólo la última recompensa obtenida. Esto se debe a que los agentes utilizan la poda de acciones como un primer filtro en el potencial valor de las mismas. Con este filtro, la utilidad de contar con un mayor historial de recompensas por acción disminuye, lo que se traduce en que, a diferencia de lo que ocurre en *ALAN*, los agentes puedan escoger acciones que mejoran su eficiencia con un mínimo de información histórica.

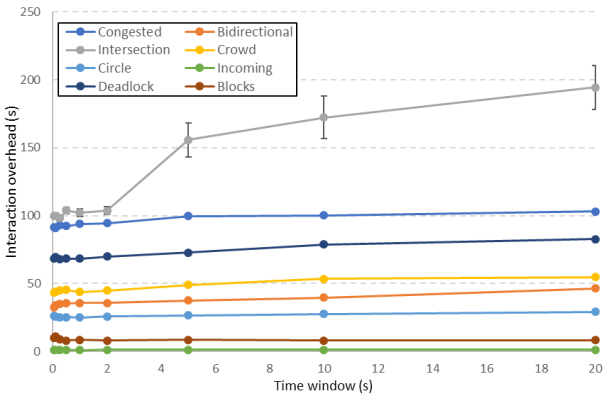


Fig. 7. Interaction Overhead de *ALAN-P* por escenario al variar el tamaño de la ventana de tiempo de cada agente.

Resultados de la poda de acciones: Como se puede observar en la Figura 8, la *poda de acciones* tiene un claro efecto positivo para *ALAN-P*, independiente del tipo de escenario. Esta diferencia es más marcada en escenarios donde los agentes deben interactuar con otros grupos desplazándose en direcciones opuestas, como es el caso en los escenarios CROWD, CIRCLE e INTERSECTION.

El enfocar la toma de decisiones en acciones que pueden mejorar la eficiencia en el movimiento de los agentes se traduce en una disminución significativa del tiempo de viaje de los agentes y, además, en una reducción del *ruido* que afecta al proceso de aprendizaje de los agentes, que ayuda a reducir la alta no-estacionaridad en su entorno. La Tabla II muestra que los agentes que utilizan *ALAN-P* realizan menos cambios de acciones comparado con *ALAN* (en porcentaje de todas las decisiones que toman durante su navegación). Este resultado indica que, efectivamente, *ALAN-P* reduce el ruido impuesto a otros agentes en la navegación.

VII. CONCLUSIONES

En este trabajo hemos presentado *ALAN-P*, un método de navegación para multiples agentes que busca mejorar la eficiencia en la navegación de un conjunto de agentes en ambientes restrictivos y sin comunicación. Nuestra propuesta reduce el tiempo de viaje de los agentes a través de la *poda*

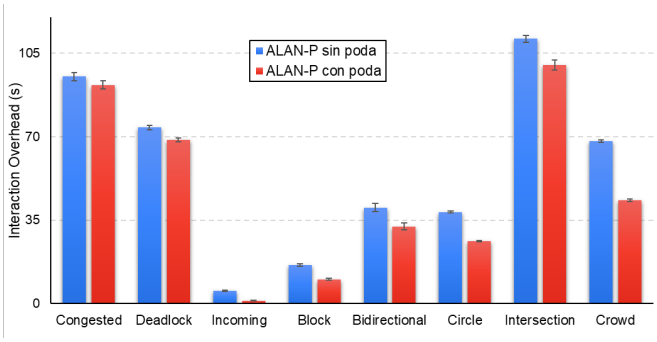


Fig. 8. Interaction Overhead para cada escenario comparando *ALAN-P* con y sin la poda de acciones.

de acciones subóptimas y de la *estimación Markoviana de estado*, que buscan producir evaluaciones más precisas sobre las acciones previas y eliminar acciones que lleven a peores resultados. La evaluación de nuestro método *ALAN-P* muestra una clara mejora sobre *ALAN* y *ORCA*, incluso en aquellos escenarios donde *ALAN* no logró superar a *ORCA*. Esto se debe a que en situaciones con pocas restricciones para la navegación, nuestro método se enfoca en aquellas acciones que muevan a cada agente hacia su objetivo, en lugar de explorar acciones que claramente no mejoran la eficiencia en la navegación.

Como trabajo futuro, hemos definido dos líneas de investigación. La primera línea involucra mejorar el conjunto de acciones a escoger por los agentes, utilizando percepciones locales de cada agente para determinar un conjunto a prior de acciones disponibles. La segunda línea de investigación considera mejorar el uso de información histórica para el aprendizaje, a traves del uso de ventanas de tiempo dinámicas.

TABLA I

TIEMPO PROMEDIO DE CPU (EN SEGUNDOS) PARA LLEVAR A LOS AGENTES A SUS OBJETIVOS, CON *ALAN* Y *ALAN-P*.

	Crowd	*Circle*	*Bidirectional*	*Intersection*	*Congested*	*Incoming*	*Deadlock*	*Blocks*
ALAN	1,82	0,395	0,032	0,427	0,074	0,018	0,027	0,008
ALAN-P	1,377	0,378	0,03	0,397	0,069	0,014	0,022	0,007

TABLA II

PORCENTAJE DE DECISIONES EN QUE LOS AGENTES CAMBIAN SUS ACCIONES, CON *ALAN* Y *ALAN-P*.

	Crowd	*Circle*	*Bidirectional*	*Intersection*	*Congested*	*Incoming*	*Deadlock*	*Blocks*
ALAN	42,7 %	28,5 %	29,3 %	40 %	47,3 %	13,5 %	38,1 %	24,4 %
ALAN-P	27,5 %	6,6 %	12 %	26,3 %	36,8 %	0,8 %	26,7 %	11,1 %

AGRADECIMIENTOS

La investigacion presentada en este artículo fue apoyada por el Fondo Nacional De Ciencia Y Tecnologia (FONDECYT) de ANID, Chile, mediante los proyectos de Postdoctorado No. 3170971 y de Iniciación No. 11191197.

REFERENCIAS

[1] J. Yu and S. M. LaValle, "Structure and intractability of optimal multi-robot path planning on graphs," in *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[2] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 1987, pp. 25–34.

[3] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[4] I. Karamouzas, B. Skinner, and S. J. Guy, "Universal power law governing pedestrian interactions," *Physical review letters*, vol. 113, no. 23, p. 238701, 2014.

[5] A. Garcimartín, J. M. Pastor, C. Martín-Gómez, D. Parisi, and I. Zuriguel, "Pedestrian collective motion in competitive room evacuation," *Scientific reports*, vol. 7, no. 1, p. 10792, 2017.

[6] J. Godoy, S. J. Guy, M. Gini, and I. Karamouzas, "C-nav: Distributed coordination in crowded multi-agent navigation," *Robotics and Autonomous Systems*, p. 103631, 2020.

[7] J. Rodríguez, J. Godoy, and F. Gutierrez, "Multi-agent communication models for cooperative navigation in complex environments," *IEEE Latin America Transactions*, vol. 19, no. 9, pp. 1556–1563, 2021.

[8] J. Godoy, T. Chen, S. J. Guy, I. Karamouzas, and M. Gini, "ALAN: adaptive learning for multi-agent navigation," *Autonomous Robots*, vol. 42, no. 8, pp. 1543–1562, 2018.

[9] J. Funge, X. Tu, and D. Terzopoulos, "Cognitive modeling: knowledge, reasoning and planning for intelligent characters," in *26th Annual Conference on Computer Graphics and Interactive Techniques*, 1999, pp. 29–38.

[10] S. J. Guy, S. Kim, M. C. Lin, and D. Manocha, "Simulating heterogeneous crowd behaviors using personality trait theory," in *Proceedings of the 2011 ACM SIGGRAPH/Eurographics symposium on computer animation*, 2011, pp. 43–52.

[11] M. A. Ramos, V. Munoz, E. Castellanos, and F. Ramos, "Modeling workplace evacuation behaviors using intelligent agents," *IEEE Latin America Transactions*, vol. 14, no. 9, pp. 4150–4155, 2016.

[12] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using Velocity Obstacles," *The Int. J. of Robotics Research*, vol. 17, pp. 760–772, 1998.

[13] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Proc. International Symposium of Robotics Research*. Springer, 2011, pp. 3–19.

[14] S. J. Guy, J. Chhugani, S. Curtis, P. Dubey, M. C. Lin, and D. Manocha, "Pledestrians: A least-effort approach to crowd simulation." in *Symposium on computer animation*, 2010, pp. 119–128.

[15] F. Muhammad, S. Juniastuti, S. M. S. Nugroho, and M. Hariadi, "Crowds evacuation simulation on heterogeneous agent using agent-based reciprocal velocity obstacle," in *2018 International Seminar on Intelligent Technology and Its Applications (ISITIA)*. IEEE, 2018, pp. 275–280.

[16] D. Hennes, D. Claes, W. Meeussen, and K. Tuyls, "Multi-robot collision avoidance with localization uncertainty," in *Autonomous Agents and MultiAgent Systems*, 2012, pp. 147–154.

[17] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart, "Optimal reciprocal collision avoidance for multiple non-holonomic robots," in *Distributed Autonomous Robotic Systems*. Springer, 2013, pp. 203–216.

[18] A. Bojeri and G. Iacca, "Evolutionary optimization of drone trajectories based on optimal reciprocal collision avoidance," in *2020 27th Conference of Open Innovations Association (FRUCT)*. IEEE, 2020, pp. 18–26.

[19] S. Curtis, S. Guy, B. Zafar, and D. Manocha, "Virtual Tawaf: A case study in simulating the behavior of dense, heterogeneous crowds," in *2011 IEEE International Conference on Computer Vision Workshops, ICCV Workshops 2011*, Dec. 2011, pp. 128–135.

[20] H. Lehnert, M. Araya, R. Carrasco-Davis, and M.-J. Escobar, "Bio-inspired deep reinforcement learning for autonomous navigation of artificial agents," *IEEE Latin America Transactions*, vol. 17, no. 12, pp. 2037–2044, 2019.

[21] W. Uther and M. Veloso, "Adversarial reinforcement learning," Technical report, Carnegie Mellon University, 1997. Unpublished, Tech. Rep., 1997.

[22] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

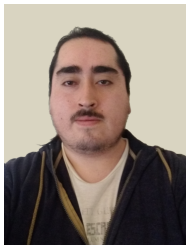
[23] H. Robbins, "Some aspects of the sequential design of experiments," *Bulletin of the American Mathematical Society*, vol. 58, no. 5, pp. 527–535, 1952.

[24] J.-Y. Audibert, R. Munos, and C. Szepesvári, "Exploration–exploitation tradeoff using variance estimates in multi-armed bandits," *Theoretical Computer Science*, vol. 410, no. 19, pp. 1876–1902, 2009.

[25] W. G. Macready and D. H. Wolpert, "Bandit problems and the exploration/exploitation tradeoff," *IEEE Trans. Evol. Comput.*, vol. 2, no. 1, pp. 2–22, 1998.

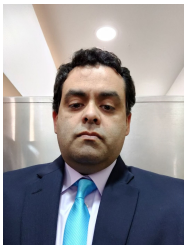
[26] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[27] "http://gamma.cs.unc.edu/rvo2/."



Joaquin Soto Obtuvo el título de Ingeniero civil informático en la Universidad de Concepción. Sus áreas de interés son la ingeniería de software y el desarrollo de proyectos en general.

Julio Godoy Obtuvo su doctorado en Computación en la Universidad de Minnesota, USA. Actualmente es profesor asistente en la Universidad de Concepción, Chile. Sus intereses de investigación están en la intersección de inteligencia artificial distribuida y robótica.



Fernando Gutierrez Obtuvo su doctorado en Computación en la Universidad de Oregon, USA. A través de su cargo en AIDA, trabaja en la integración de técnicas de Inteligencia Artificial a procesos industriales. Sus áreas de interés son integración de representación de conocimiento en sistemas multi-agentes y Web Semántica.