Tesis de Licenciatura

Departamento de Computación

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Aprendizaje por Refuerzo en Robots Modulares Autoreconfigurables

El aprendizaje como metodología para resolver tareas de locomoción y reconfiguración

Tesista: Ricardo Martín Kirkner (L.U. 479/98)

Director: Dr. Juan Miguel Santos

1 de marzo de 2007

Resumen

La metodología del Aprendizaje por Refuerzo se ha visto reiteradamente ignorada en la literatura que abarca el estudio de las tareas de locomoción y reconfiguración en Robots Modulares Autoreconfigurables.

Sin embargo, esta metodología ha demostrado poseer beneficios importantes en los casos en los que se la utilizó para resolver problemas asociados a entornos dinámicos e impredecibles. Esta característica resulta muy adecuada a los problemas habitualmente encontrados en el campo de los Robots Modulares Autoreconfigurables.

En el presente trabajo se informa acerca del estado del arte en el campo de los Robots Modulares Autoreconfigurables, para determinar los problemas aún no resueltos y las posibles vetas de investigación presentes en dicho campo. Luego se implementa un simulador con el fin de estudiar la aplicación de la metodología de Aprendizaje por Refuerzo para obtener comportamientos que resuelvan eficientemente las tareas de locomoción y reconfiguración en un robot autoreconfigurable M-TRAN. Finalmente se plantean diversos experimentos que involucran las mencionadas tareas, para los cuales se estudia la representación más adecuada del espacio de estados y acciones, la forma de discretizar el mismo de manera que los experimentos sean computacionalmente factibles, y el impacto de distintos aspectos involucrados en un problema de aprendizaje (definición de episodio, velocidad de aprendizaje, evaluación de la política aprendida).

Como este tipo de estudio tiene un componente experimental importante, luego de diseñar los experimentos y llevarlos a cabo, se evalúan los resultados obtenidos y se extraen conclusiones acerca de la efectividad y posibilidades que tiene el Aprendizaje por Refuerzo en este tipo de robots.

Abstract

Reinforcement Learning has been found repeatedly missing in the literature about reconfiguration and locomotion tasks in Self-Reconfigurable Modular Robots.

This methodology, however, has been proved to produce significant benefits in those cases where it was used to solve problems related to dynamic and unpredictable environments. This kind of environments is very common when dealing with Self-Reconfigurable Modular Robots.

In this work, we first survey the state of the art in the field of Self-Reconfigurable Modular Robots, in order to determine what the yet unsolved problems and the possible niches of research are. We then develop a simulator in order to study the Reinforcement Learning methodology applied to the generation of behaviours that efficiently solve reconfiguration and locomotion tasks using the M-TRAN Self-Reconfigurable Modular Robot. Finally, several experiments are suggested in order to test this methodology. For those experiments it is necessary to define the best representation for the state-action space, and the way to discretize it in order to make those experiments computationally feasible. Another issue analyzed is the impact of several aspects involved in Reinforcement Learning problems, such as the definition of an episode, the learning velocity and policy evaluation.

Since this kind of study involves a significant experimental component, after designing and performing the experiments, the results are analyzed and conclusions are then drawn about the effectiveness and possibilities of Reinforcement Learning when applied to these kind of robots.

Índice general

1. Introducción **1**

2. Estado del arte **3**
2.1. Introducción 3
2.2. Características generales acerca de los Robots Modulares Autoreconfigurables 4
2.2.1. ¿Qué son Robots Modulares Autoreconfigurables? . . . 4
2.2.2. ¿Qué hace tan interesantes a los Robots Modulares Autoreconfigurables? 4
2.2.3. ¿Cuáles son los escenarios más aptos para los Robots Modulares Autoreconfigurables? 7
2.3. Una Clasificación para Robots Modulares Reconfigurables . . 9
2.3.1. Robots Modulares 10
2.3.2. Robots Reconfigurables 12
2.3.3. Robots Autoreconfigurables 12
2.4. Problemas habituales 14
2.4.1. Reconfiguración 14
2.4.2. Locomoción 16
2.4.3. Metodología 17
2.5. Los competidores 20
2.5.1. Robots Tipo Cadena 20
2.5.2. Robots Tipo Malla 21
2.5.3. Comparación 29
2.6. Las soluciones 29
2.6.1. Reconfiguración 30
2.6.2. Locomoción 32
2.7. Los destacados 33
2.7.1. M-TRAN 33
2.7.2. Polybot 34
2.7.3. CONRO 34
2.8. Discusión 34

2.8.1. Metodología 35
2.8.2. Diseño de Hardware 35
2.9. Conclusiones 35

3. Hipótesis, Materiales y Métodos **37**
3.1. Objetivo 37
3.2. M-TRAN 37
3.2.1. Mecanismos de conexión 39
3.2.2. Ilustraciones 42
3.3. Simulador 42
3.3.1. Detalles de implementación 43
3.3.2. Problemas encontrados y sus soluciones 45
3.3.3. Limitaciones 47
3.4. Aprendizaje 49
3.4.1. Aprendizaje por Refuerzo 49

4. Experimentación **55**
4.1. Locomoción 55
4.1.1. Experimento 1: Locomoción en un robot de configuración mínima 56
4.2. Reconfiguración 69
4.2.1. Experimento 2: Reconfiguración básica 70
4.2.2. Experimento 3: De estructura lineal a circular, acciones simples 73
4.2.3. Experimento 4: De estructura lineal a circular, acciones complejas 78
4.2.4. Experimento 5: De estructura lineal a circular con acoplamiento/desacoplamiento, acciones complejas 83
4.2.5. Experimento 6: De estructura lineal a circular con acoplamiento, sin desacoplamiento, acciones simples 85
4.2.6. Experimento 7: De estructura circular a lineal con acoplamiento/desacoplamiento, acciones complejas 90
4.2.7. Experimento 8: De estructura circular a lineal con acoplamiento/desacoplamiento, acciones simples 95
4.2.8. Conclusiones 97

5. Demostración conceptual **99**
5.1. Combinación de políticas 101
5.2. Resultados 103
5.2.1. Política 1: realizar locomoción mediante una configuración circular 103

5.2.2. Política a: pasar de los estados finales de la política 1 al estado inicial de la política 2 105
5.2.3. Política 2: reconfigurarse para pasar de una configuración circular a una configuración lineal 105
5.2.4. Política 3: realizar locomoción mediante una configuración lineal 109
5.2.5. Política b: pasar de los estados finales de la política 3 al estado inicial de la política 4 109
5.2.6. Política 4: reconfigurarse para pasar de una configuración lineal a una configuración circular 112
5.3. Conclusiones 112

6. Conclusiones **115**

A. Comparación de Robots Modulares Reconfigurables **117**

B. Resultados experimentales **118**

C. Imágenes **121**
C.1. M-TRAN II 121
C.2. M-TRAN III 129

Bibliografía **130**

Capítulo 1

Introducción

Estamos inmersos en una realidad en la que los robots constituyen un elemento cada vez más habitual y necesario para resolver tareas con mayor eficiencia y menor riesgo para los humanos. Los robots están presentes en todos lados, desde la industria hasta el entretenimiento y la medicina. Sin embargo, en la mayoría de los casos, los robots utilizados están restringidos a un entorno bien determinado y poco flexible. Si el mismo llegara a modificarse sustancialmente, el robot podría quedar parcial o completamente inutilizado, puesto que dejaría de operar dentro de los parámetros para los que fue diseñado.

A su vez, existen muchos tipos de robots que exponen características diferentes. Los robots son utilizados como plataformas móviles, como manipuladores e incluso como laboratorios de análisis móviles. Debido a que deben operar en ambientes que cumplan con rígidas condiciones, no es posible aprovecharlos en condiciones para las que no fueron diseñados.

Es aquí donde los *Robots Modulares Autoreconfigurables* cobran importancia. Estos robots están compuestos por módulos independientes que pueden ser combinados de múltiples maneras. De esta forma, es posible construir varios robots especializados diferentes a partir de los mismos módulos. A su vez, estos robots pueden modificar su estructura de forma de adaptarse lo mejor posible a cada tarea y a cada entorno en el que deben operar.

Es por ello que este tipo de robots resuelve las limitaciones inherentes a los robots no reconfigurables. Al ser modulares, pueden adquirir diferentes capacidades en distintos momentos y, por ejemplo, combinar las capacidades de una plataforma móvil con las de un manipulador, de forma de obtener los beneficios de ambos tipos de robots.

Por otro lado, en el campo del Aprendizaje Automático, existe una metodología denominada *Aprendizaje por Refuerzo* [Sut98, Wat89] que resulta eficiente para permitir encontrar soluciones a problemas en entornos dinámi-

cos o poco estructurados, así como realizar tareas complejas que dependen de factores imprevisibles del entorno.

En este trabajo se informa acerca del estado del arte en el campo de los Robots Modulares Autoreconfigurables, para determinar los problemas aún no resueltos y las posibles vetas de investigación presentes en dicho campo (cap. 2). A continuación se presenta la hipótesis que se desea verificar, y se introducen los materiales que se utilizarán para ello (cap. 3). Entre éstos se encuentra un simulador que se desarrolló con el fin de estudiar la aplicación de la metodología de Aprendizaje por Refuerzo para obtener comportamientos que resuelvan eficientemente las tareas de locomoción y reconfiguración en un robot autoreconfigurable M-TRAN. Finalmente (caps. 4 y 5) se plantean diversos experimentos que involucran las mencionadas tareas, para los cuales se estudia la representación más adecuada del espacio de estados y acciones, la forma de discretizar el mismo de manera que los experimentos sean computacionalmente factibles, y el impacto de distintos aspectos involucrados en un problema de aprendizaje (definición de episodio, velocidad de aprendizaje, evaluación de la política aprendida). Luego de diseñar los experimentos y llevarlos a cabo, se evalúan los resultados obtenidos y se extraen conclusiones acerca de la efectividad y posibilidades que tiene el Aprendizaje por Refuerzo en este tipo de robots (cap. 6).

Capítulo 2

Estado del arte

2.1. Introducción

En este capítulo se analiza el estado del arte en el campo de los Robots Modulares Autoreconfigurables, al momento de realizar el presente trabajo.

Se verá por qué este tipo de robots son necesarios, los beneficios que se pueden obtener mediante su utilización, y cuales son las situaciones en las que pueden ser utilizados. Además, se verá por qué a veces un robot "clásico" no puede ser utilizado, y cuáles son las tareas en las que mejor se desempeñan los robots autoreconfigurables.

El resto de este capítulo se organiza de la siguiente manera: en la sección 2.2 se enuncian las principales definiciones y se establecen las capacidades requeridas por un Robot Modular Autoreconfigurable. En la sección 2.3 se describe una posible taxonomía para este tipo de robots. La sección 2.4 trata acerca de los problemas más comunes que se encuentran al trabajar con este tipo de robots. En la sección 2.5 se describen los Robots Modulares Autoreconfigurables más representativos encontrados en la bibliografía. La sección 2.6 explica cómo alguno de los problemas presentados fueron solucionados por las diferentes implementaciones de este tipo de robots. En la sección 2.7 se analiza más detalladamente los ejemplares más citados, y finalmente, en las secciones 2.8 y 2.9 se discuten algunas consideraciones y conclusiones sobre este tipo de robots.

2.2. Características generales acerca de los Robots Modulares Autoreconfigurables

En esta sección se discuten las principales características que describen a los Robots Modulares Autoreconfigurables.

2.2.1. ¿Qué son Robots Modulares Autoreconfigurables?

- Son *modulares*

  Robots *modulares* se construyen a partir de múltiples módulos interconectados.

- Son *reconfigurables*

  Un robot se dice *reconfigurable*, cuando es posible modificar su estructura con el fin de alcanzar diferentes configuraciones que le permitan realizar diversas tareas.

- Son *autoreconfigurables*

  Un robot es *autoreconfigurable*, cuando puede reconfigurarse por sus propios medios, sin intervención externa.

2.2.2. ¿Qué hace tan interesantes a los Robots Modulares Autoreconfigurables?

Se pueden identificar varios aspectos que hacen de los Robots Modulares Autoreconfigurables un tema interesante para estudiar. En este apartado se analizan los aspectos más relevantes.

Robustez

> "Los Robots Modulares Autoreconfigurables están compuestos por muchos módulos idénticos, y por lo tanto, si un módulo falla, es posible remplazarlo por otro." [1]

Dado que estos robots están construidos a partir de módulos individuales, es posible remplazar un módulo defectuoso (o incluso separarlo completamente del robot), de forma que no afecte el comportamiento del mismo. Esta

[1] "Self-reconfigurable robots are made from many identical modules and therefore if a module fails it can be replaced by another." [Sto04]

característica permite que el robot degrade suavemente su rendimiento cuando uno o más módulos fallan, en vez de quedar completamente inutilizado por el mal funcionamiento de un módulo.

Versatilidad

> "Los módulos pueden ser combinados de diferentes maneras, permitiendo al mismo sistema robótico realizar una amplia gama de tareas." [2]

Un Robot Modular Autoreconfigurable es capaz de adquirir diferentes configuraciones y utilizar diferentes tipos de módulos para alcanzar diversos objetivos. De esta forma, es posible equipar al robot con las mejores herramientas para cada tarea a desempeñar en cada momento.

Si bien distintas tareas requieren habilidades diferentes, el robot siempre será el mismo (dado que los módulos son reutilizados, aún cuando la estructura externa del robot varíe, estará formado por los mismos módulos), pero podrá ser preparado de la mejor manera posible para cada nueva tarea a resolver. Esto quiere decir que un único Robot Modular Autoreconfigurable puede realizar las mismas tareas para las que se requerirían varios robots especializados (no reconfigurables).

Naturalmente, esto no quiere decir que los Robots Modulares Autoreconfigurables sean la mejor solución a todos los problemas, dado que es factible que el costo de producción de un robot de este tipo sea mayor que el costo de varios robots especializados.

Sin embargo, existen casos en los que la utilidad de un Robot Modular Autoreconfigurable supera ampliamente la diferencia de costos. Los escenarios más adecuados para este tipo de robots son aquellos en los que el ambiente en el que deben desempeñar sus tareas son dinámicos y poco estructurados, y cuando se dispone de muy poca (o ninguna) información acerca del entorno en el cual deberá desenvolverse el robot (como por ejemplo, tareas de exploración planetaria o de búsqueda y rescate en derrumbes o construcciones debilitadas).

En este tipo de escenarios, la habilidad de estos robots de adquirir nuevas configuraciones es crítica. Dado que no es posible predecir con exactitud el entorno, es necesario que el robot pueda adaptarse al mismo dinámicamente. La mayoría de los robots especializados requieren que el entorno esté determinado de antemano y se rija por estrictas especificaciones, y por lo tanto no son adecuados en estas circunstancias.

[2] "The modules can be combined in different ways making the same robotic system able to perform a wide range of tasks." [Sto04]

Costo

> "Una vez obtenido el diseño final de un módulo, es posible producirlo en masa, y por consiguiente mantener bajos los costos de producción de un módulo, en relación con su complejidad." [3]

Si bien los Robots Modulares Autoreconfigurables pueden resultar costosos, debido a la complejidad del hardware involucrado para incorporar todos los componentes necesarios en módulos independientes, y debido a la complejidad del software necesario para resolver la tarea de reconfiguración, en muchos casos puede resultar más efectivo (desde el punto de vista del costo) utilizar este tipo de robots en vez de una colección de robots especializados para alcanzar los mismos objetivos.

Además, la robustez de estos sistemas robóticos hace que el Costo Total de Propiedad (TCO) disminuya, dado que el mismo robot puede ser utilizado en diversas circunstancias, evitando tener que disponer de un robot especializado para cada una de ellas, siendo el mismo luego descartado (dado que su utilidad queda limitada a la situación para la cuál se lo diseñó).

La robustez, versatilidad y adaptabilidad de estos sistemas robóticos le proporcionan una vida útil mucho mayor que a su contrapartida no reconfigurable, dado que en los sistemas tradicionales, en donde muchos robots especializados interactúan para lograr un objetivo, cuando uno solo de éstos falla, puede forzar a todo el conjunto a tener que abandonar la tarea. Si esto ocurriera en una situación en la que no es posible recuperar los robots, entonces la falla del más pequeño componente puede resultar tan costosa como el sistema completo. En estos casos es cuando los Robots Modulares Autoreconfigurables son claramente menos costosos.

Adaptabilidad

> "Mientras que un robot autoreconfigurable está realizando una tarea, puede modificar su estructura física para adaptarse a cambios en el entorno." [4]

La adaptabilidad es realmente crucial cuando el robot debe operar en un medio ambiente poco estructurado y dinámico, en el que las condiciones operativas no pueden ser predichas (ni se mantienen estables).

---

[3] "When the final design for the basic module has been obtained, it can be mass produced, thereby keeping the cost of an individual module low, compared to its complexity." [Sto04]

[4] "While the self-reconfigurable robot performs its task, it can change its physical shape to adapt to changes in the environment." [Sto04]

En estas situaciones, si el robot no puede adaptarse a las condiciones locales, puede quedar impedimentado por agentes externos, sin lograr completar su tarea.

Los Robots Modulares Autoreconfigurables se desenvuelven particularmente bien en situaciones como éstas, dado que su habilidad para cambiar su estructura les permite superar obstáculos imprevistos.

En un caso extremo, incluso es posible desprender parte de la estructura (que haya quedado inutilizada por alguna razón), y continuar operando (aunque sea de manera restringida).

Escalabilidad

> "El tamaño del sistema robótico puede ser incrementado o disminuido mediante la adición o substracción de módulos." [5]

Dado que los Robots Modulares Autoreconfigurables están compuestos por módulos interconectados, es posible agregar mayor cantidad de éstos al robot para permitirle conseguir configuraciones que requieran de un "mayor tamaño".

También es posible para el robot dividirse en robots más pequeños que puedan operar en paralelo (y posiblemente realizar diferentes tareas), para luego volver a unirse en un único robot. Esto es especialmente útil para robots que realizan tareas de exploración o vigilancia, dado que al operar en paralelo, les permite cubrir mayor superficie.

En general, siempre que un único robot tenga que realizar tareas que puedan ser divididas en subtareas más pequeñas y donde exista cierto grado de paralelización, un Robot Modular Autoreconfigurable puede producir una diferencia significativa en el rendimiento, al dividirse en robots más pequeños que puedan ser asignados a cada subtarea.

2.2.3. ¿Cuáles son los escenarios más aptos para los Robots Modulares Autoreconfigurables?

Búsqueda y Rescate

Un escenario en el que el robot requiera explorar cierta ubicación, por ejemplo un derrumbe, en la que las condiciones en las que tuviera que operar no pueden ser predichas puede ser un caso ideal para este tipo de robots. Podría haber terreno irregular que el robot tuviera que superar; podría haber

[5] "The size of the robot system can be increased or decreased by adding or removing modules." [Sto04]

aberturas y pasajes que condujeran a áreas de otra forma inaccesibles, o incluso podría haber secciones del suelo que no soportaran demasiado peso. En este escenario, el robot podría tener que interactuar con personas (en el caso de encontrar sobrevivientes), objetos (recuperar objetos dentro del edificio) y el entorno en general (manipular objetos o desplazar obstáculos).

Una tarea como ésta normalmente requeriría varios tipos de robots: un robot con buenas propiedades de locomoción, un robot manipulador, un robot que pudiera interactuar con personas, etc. El problema reside en que muchos de estos robots específicos requieren de entornos controlados para operar correctamente, situación que muy probablemente no se cumpla en un escenario como el descripto.

En cambio, un Robot Modular Autoreconfigurable podría operar de forma efectiva en este tipo de entorno. Si el suelo fuera lo suficientemente llano, podría incurrir primero en locomoción circular (similar a una rueda); luego al encontrar escombros podría adquirir una configuración más adecuada para superar ese tipo de obstáculos (como por ejemplo una configuración cuadrúpeda). Es posible que encontrara un pasaje que no fuera lo suficientemente amplio para que el robot pudiera caber, pero si se reconfigurara en forma lineal (como un gusano), podría pasar a través del pasaje. Una vez superado el obstáculo, podría reconfigurarse de la manera más adecuada, según las necesidades que surgieran.

Ya evaluada la situación, el robot podría retirarse y dejar que otro equipo hiciera el rescate; podría determinar los requerimientos necesarios para otro robot más especializado, que dispondría de un entorno predecible sobre el cuál desenvolverse más eficientemente (el entorno habiendo sido evaluado y sus características establecidas por el Robot Modular Autoreconfigurable). También es posible volver a enviar el mismo robot autoreconfigurable, pero equipado con otro tipo de módulos que le permitieran realizar mejor las nuevas tareas.

Por lo mencionado aquí, se puede ver que *Búsqueda y Rescate* es un caso perfecto para este tipo de robot, donde el entorno no se puede determinar previamente (e inclusive puede variar de un momento a otro).

Exploración

La exploración es otro campo apto para este tipo de robot. Desde la exploración planetaria hasta la exploración submarina, cualquier entorno desconocido, o que no puede ser predicho, constituye una buena oportunidad para explotar la capacidad de estos robots. En este tipo de entornos, pueden demostrar sus habilidades y lograr exitosamente un objetivo que para otro tipo de robots sería imposible cumplir.

Nuevamente, es la capacidad de operar en entornos impredecibles y dinámicos, y poder adaptarse a cambios en dichos entornos, lo que hace que los Robots Modulares Autoreconfigurables estén excepcionalmente bien preparados para tareas de exploración. Esta capacidad (adaptarse a los cambios ambientales) les da una ventaja competitiva sobre sus pares no reconfigurables.

No importa cuan bien se desenvuelvan estos robots en los escenarios recién descriptos, su habilidad para adaptarse y operar en una variedad de circunstancias, les permite ser utilizados para prácticamente cualquier tarea. Naturalmente no serán igual de eficientes que robots específicamente diseñados para una tarea puntual, pero gracias a su versatilidad, este factor no detrimenta su competitibilidad. Por el contrario, son una opción perfecta en muchos casos, especialmente cuando no se dispone de gran capital, pero es necesario utilizar robots para resolver varias tareas, puesto que puede resultar más económico adquirir un robot de este tipo que varios robots especializados.

2.3. Una Clasificación para Robots Modulares Reconfigurables

La clasificación propuesta se basa en clasificaciones existentes, que distinguen los robots según distintas propiedades: la *clasificación según la homogeneidad de los módulos* [Rus00, Dit04, Cas02] y la *clasificación según la estructura espacial de los módulos* [Dit04, Yos01b, Cas02]. Se pretende integrar dichas clasificaciones en una única clasificación que tenga en cuenta: homogeneidad de los módulos, estructura espacial de los mismos y reconfigurabilidad del robot. De esta manera, es posible clasificar los robots con un mayor nivel de granularidad que al utilizar solamente una de las clasificaciones existentes.

La clasificación propuesta se estructura según las distintas propiedades consideradas:

- Homogeneidad
  - Homogéneos
  - Heterogéneos
  - Híbridos (N-Homogéneos)
- Estructura espacial

  - Tipo cadena (Chain-type)
  - Tipo malla (Lattice-type)
  - Híbridos

- Reconfigurabilidad
  - Reconfigurables
  - Autoreconfigurables

Cabe aclarar que dado que esta clasificación abarca únicamente a Robots Modulares Reconfigurables, no se consideran en ella los casos de robots no modulares o no reconfigurables.

2.3.1. Robots Modulares

La característica principal de los Robots Modulares es que están compuestos por un conjunto de módulos independientes interconectados. Dichos módulos pueden estar especializados, construidos específicamente para un robot particular, o bien pueden ser módulos estandarizados, diseñados para uso general.

Robots Homogéneos

En este tipo de robot, todos los módulos que lo componen son idénticos.

- **Ventajas**

  Como todos los módulos son idénticos, pueden ser producidos en masa. Esto permite disminuir el costo total del robot.

  Dado que existe un único tipo de módulo, el problema de reconfiguración se simplifica, ya que no es necesario contemplar el problema de seleccionar el módulo adecuado (todos los módulos son adecuados, pues son idénticos).

- **Desventajas**

  Cada módulo debe estar autocontenido, proveyendo su propia energía, actuadores y sensores. Por esta razón, este tipo de módulos tienden a ser relativamente grandes, lo que generalmente se asocia con un mayor peso y los hace más costosos y engorrosos.

  Además, cada módulo debe proveer todo tipo de actuadores y sensores que pueda llegar a ser necesario para el robot en su totalidad, lo

que claramente no resulta en una estrategia de utilización de recursos óptima, ya que no todos los módulos están involucrados en el mismo tipo de interacción (usando los mismos sensores y actuadores) al mismo tiempo, y por lo tanto podrían compartir sus recursos.

Robots Heterogéneos

Este tipo de robots puede estar construido a partir de diferentes tipos de módulos que posean características diferentes entre sí.

- **Ventajas**

  Es posible especializar cada módulo de acuerdo a su función, simplificando su producción y reduciendo su costo.

  Es posible agregar módulos específicos paulatinamente, y permitir que el robot los asimile como si hubiera sido diseñado desde un principio con dichos módulos, dándole mayor funcionalidad a medida que se requiera y permitiendo un proceso de mejora incremental a lo largo de la vida útil del robot. De esta forma es posible utilizar ciertos módulos como base para el robot, y luego agregar módulos especializados de acuerdo con la tarea a desarrollar.

- **Desventajas**

  La reconfiguración se complica, dado que al haber diferentes tipos de módulos, es necesario tener en cuenta la cantidad de módulos de cada tipo, y cómo interactúan dichos módulos entre sí.

  La diversidad modular restringe la cantidad de módulos de un mismo tipo, por lo que es posible que un módulo se torne crítico en la estructura del robot, en el caso de no contar con otros módulos del mismo tipo para remplazarlo.

Robots N-Homogéneos

Los robots N-homogéneos resultan un caso intermedio entre un robot homogéneo y un robot heterogéneo. Este tipo de robot puede estar formado por módulos de como máximo N distintos tipos.

- **Ventajas**

  La cantidad de módulos diferentes que deben ser tomados en cuenta es pequeña (normalmente, entre 2 y 3). Esto permite que el problema de reconfiguración quede acotado y no se complejize tanto como en el caso heterogéneo.

Dado que la cantidad de módulos diferentes está acotada, la construcción del robot no es tan compleja como en el caso heterogéneo y como existen diferentes tipos de módulos, cada uno puede especializarse en una tarea particular, pudiendo ser más pequeños, más eficientes y más económicos.

- **Desventajas**

  No siempre es posible remplazar un módulo por otro (dado que existen diferentes tipos de módulos).

  Es necesario mantener una baja heterogeneidad para que la complejidad quede acotada (de lo contrario se transforma en un caso de robot heterogéneo).

2.3.2. Robots Reconfigurables

Un robot se dice reconfigurable, cuando es posible modificar su estructura con el fin de alcanzar diferentes configuraciones que le permitan realizar diversas tareas.

- **Ventajas**

  La habilidad de reconfigurarse de un robot lo hace más versátil, dado que puede operar en una variedad de situaciones.

- **Desventajas**

  Los robots que poseen la capacidad de reconfiguración, son más complejos de construir, y por lo tanto su costo es mayor. Asimismo, los robots reconfigurables necesitan disponer de un software más complejo, lo que a su vez incrementa el costo de desarrollo.

2.3.3. Robots Autoreconfigurables

Un robot se dice autoreconfigurable, cuando puede reconfigurarse por sus propios medios, sin intervención externa.

- **Ventajas**

  Un robot autoreconfigurable posee todas las ventajas de un robot reconfigurable, pero además puede operar en entornos dinámicos e impredecibles, gracias a su naturaleza autónoma dada por su independencia de intervención externa para reconfigurarse. Esta autonomía le permite ser utilizado en ambientes donde la intervención externa (mayormente humana) no es posible.

- **Desventajas**

  Una mayor autonomía implica una mayor complejidad en el software y en el hardware, que resulta en una manufactura más compleja y costosa.

Robots Tipo Cadena

En los robots de tipo cadena, los módulos se categorizan en distintos roles, según su ubicación.

- Módulos *divisores* (*branching* modules)

  Estos módulos establecen conexiones a por lo menos tres módulos.

- Módulos *extensores* (*chain* modules)

  Estos módulos establecen conexiones a exactamente dos módulos.

- Módulos *terminales* (*end* modules)

  Estos módulos establecen conexiones a exactamente un módulo.

La reconfiguración se produce al transformar un módulo de un tipo en un módulo de otro tipo (conectando o liberando otros módulos), hasta alcanzar la configuración deseada.

Robots Tipo Malla

En este tipo de robots, los módulos se interconectan formando una grilla o malla. La reconfiguración se logra reubicando los módulos dentro de la estructura, hasta llegar a la configuración deseada.

Robots Híbridos

En el caso del robot M-TRAN (ver sección 3.2), se trata de un robot que dispone de las cualidades de un robot de tipo cadena, así como de las pertenecientes a los robots tipo malla. Por esta razón, es necesario incluir la categoría de robots híbridos dentro de esta clasificación.

Robots Tipo Cadena versus Robots Tipo Malla

En comparación con los robots de tipo cadena, los robots de tipo malla resuelven mejor la tarea de reconfiguración. En los primeros, una hilera de módulos tiene que torcerse y acoplarse con el resto del robot. El proceso de acoplamiento involucra varios módulos y es difícil de controlar.

La ventaja de los sistemas tipo malla es que en muchos casos solamente unos pocos módulos están involucrados en el proceso de reconfiguración. Sin embargo, aún cuando los sistemas de tipo malla pueden reconfigurarse mejor que los de tipo cadena, las implementaciones actuales no son realmente eficientes. Esto se debe básicamente a la dificultad de conseguir un mecanismo de acoplamiento eficiente. Uno de los intentos más exitosos se encuentra en la implementación provista por el robot M-TRAN [Kur02, Mur02].

Otros sistemas interesantes incluyen Telecube [Suh02], Molecule [Kot98], y I-Cubes [Üns99], pero aún no han sido construidos suficiente cantidad de módulos para poder evaluar estos sistemas a gran escala [Sto04]. Los demás robots evaluados (ver sección 2.5) son bidimensionales, lo que simplifica notablemente la tarea de reconfiguración.

Los robots autoreconfigurables de tipo cadena tienen un mayor grado de movilidad que los sistemas de tipo malla. La razón para ello reside en que generalmente los grados de libertad de los robots de tipo cadena están menos restringidos que en los sistemas de tipo malla. Si se analizan las diferencias entre estos sistemas, se puede observar que los sistemas de tipo cadena están diseñados para lograr locomoción utilizando una forma definida y ocasionalmente reconfigurarse para cambiar de forma. Los sistemas de tipo malla, por el contrario, están diseñados principalmente para la reconfiguración. Una excepción es el robot M-TRAN, que resulta un híbrido entre un sistema de tipo cadena y un sistema de tipo malla.

2.4. Problemas habituales

En esta sección se discuten algunos de los problemas habitualmente encontrados en Robots Modulares Autoreconfigurables. Dado que este tipo de robots se encuentra aún en una temprana etapa de investigación, la mayoría de los problemas se relacionan con temas básicos, como la locomoción, y con la reconfiguración propiamente dicha.

2.4.1. Reconfiguración

El primer problema que debe resolver un Robot Modular Autoreconfigurable es el que le proporciona su característica fundamental: la reconfiguración (propiamente dicho, la autoreconfiguración).

La primer pregunta que surge al tratar de resolver el problema de reconfiguración es: ¿cómo lograrlo?

A fin de cambiar de forma, el robot debe tener "noción" de la forma (estructura) actual en la que se encuentra, la configuración que desea alcanzar,

y los pasos que debe realizar para lograr la transición de una configuración a otra.

Por lo tanto, un problema fundamental que debe ser resuelto a fin de lograr la reconfiguración es cómo transicionar entre dos configuraciones determinadas.

Otros problemas específicos a la tarea de reconfiguración son:

- Acoplamiento
- Mecanismo de conexión entre dos módulos

Acoplamiento

Este problema afecta principalmente a los robots de tipo cadena. Este tipo de robots deben acoplarse consigo mismos para lograr cambiar un módulo de un tipo a otro. Dado que el acoplamiento debe ocurrir de forma automática e independiente (sin intervención externa), este problema resulta fundamental.

Para lograr el acoplamiento, debe existir una forma para que dos módulos puedan "percibirse", y "reconocerse", a fin de de conectarse correctamente entre ellos (es necesario monitorear el proceso para poder controlar el delicado movimiento necesario para alinear los módulos de manera que el mecanismo de conexión pueda accionar).

Esto requiere algún método para sensarse entre ellos, y alguna manera de realizar movimientos dirigidos. Este requerimiento no es trivial, pues impone restricciones tanto sobre el software como sobre el hardware (complicando todo el desarrollo del robot).

Mecanismo de conexión

Otro factor que debe ser contemplado es el mecanismo de conexión utilizado entre módulos.

Un buen mecanismo de conexión debe ser lo más simple posible, y al mismo tiempo ser lo suficientemente robusto para garantizar que los módulos no se separarán a menos que sea intencionalmente. [Nil02]

Otro aspecto importante a tener en cuenta al diseñar un mecanismo de conexión es la carga que deberá soportar el mismo. Esto puede tener un importante impacto en el rendimiento global del robot, dado que un mecanismo que no le permita al robot levantar su propio peso, ciertamente limitará las posibilidades de reconfiguración.

Existen propuestas para varios tipos de mecanismos de conexión, algunas utilizando conexiones mecánicas [Rus00, Nil02, Yim00, Cas02], otras utilizando propiedades magnéticas [Yos01a, Pat04, Mur94, Suh02, Jor04, Kur02].

Muchos de los mecanismos propuestos no son a prueba de fallo, en el sentido de que si un módulo falla, no existe la posibilidad de liberarlo. Sin embargo, existen algunas soluciones que prometen algún tipo de tolerancia a fallas [Jan01].

2.4.2. Locomoción

Una vez resuelto el problema de reconfiguración, el primer problema de aplicación al mundo real que surge es el problema de la locomoción. Esto se debe a que la locomoción es la primer tarea que un Robot Modular Autoreconfigurable debe resolver para poder ser utilizado en escenarios reales.

Para lograr locomoción, la tarea de reconfiguración debe estar resuelta, dado que en el caso de los robots de tipo cadena el robot debe adquirir una estructura determinada que le permita realizar la locomoción deseada, y en el caso de los robots de tipo malla, deben reconfigurarse constantemente para poder desplazarse.

Este problema puede dividirse en dos casos, según el tipo de robot.

Robots Tipo Cadena

En los robots de tipo cadena, la locomoción normalmente se logra siguiendo un programa establecido para la configuración dada, utilizando una tabla (*gait control table*) que registra la secuencia de movimientos que deben ser ejecutados por cada módulo para obtener el movimiento deseado.

El primer problema que se encuentra al utilizar este enfoque es cómo construir dicha tabla, y cómo almacenarla, dado que dependiendo de la configuración del robot, un simple "paso" (en el caso de los robots que disponen de extremidades) puede involucrar varios módulos. Esto, sumado al hecho de que un Robot Modular Autoreconfigurable puede adquirir muchas configuraciones genera que esta tabla aumente de tamaño rápidamente.

A su vez, esto impone una restricción sobre el hardware que debe disponer un robot, a fin de poder adquirir capacidades de locomoción. Estos requerimientos de hardware hacen que los movimientos necesarios para desplazar el robot sean más complejos, dado que es necesario involucrar más hardware para realizar dichos movimientos.

Por esta razón, es necesario encontrar un equilibrio entre las capacidades de locomoción del robot, y el hardware utilizado para construirlo (y con ello el costo del mismo).

Robots Tipo Malla

Este tipo de robots se desplaza mediante la reconfiguración constante (cambiando de forma todo el tiempo). Al desplazar cada módulo, el robot en su totalidad es transportado a una nueva ubicación, efectivamente realizando locomoción.

En este tipo de robots, el problema es que la locomoción y la reconfiguración se vuelven una unidad indivisible, que debe ser resuelta al mismo tiempo (un robot de tipo malla que no puede mover sus módulos no podría ser nunca un robot autoreconfigurable).

Entonces, si bien la locomoción es una habilidad "natural" para este tipo de robots, en general serán más complejos que su contraparte de tipo cadena, y por ende más complicados de construir.

La mayoría de los robots de tipo malla son tratados como si fueran autómatas celulares, en los que cada célula representa un módulo, de forma que mayoritariamente son programados utilizando la teoría de autómatas celulares (especificando reglas que indican el movimiento de cada módulo para cada instante).

Naturalmente, para almacenar dichas reglas dentro del robot, es necesario disponer de memoria, y dado que estos conjuntos de reglas rara vez son pequeños, y se vuelven más grandes a medida que los movimientos se hacen más complejos, la cantidad de memoria necesaria se transforma en un inconveniente.

A medida que la memoria utilizada aumenta, los tiempos de acceso necesarios para determinar la regla correcta y ejecutarla, se vuelven más largos, y por lo tanto afectan el rendimiento y las capacidades de operación en tiempo real que estos robots podrían tener. Según se puede ver, este problema se aplica también a los robots de tipo cadena (en donde el aumento de tamaño de la tabla de movimientos produce el mismo efecto).

2.4.3. Metodología

Si bien no se trata realmente de un problema, en este apartado se verán las distintas metodologías utilizadas por los investigadores al desarrollar controladores para la tarea de reconfiguración (y locomoción).

Las metodologías se pueden clasificar en:

- Planificación (Planning)
- Evolución
- Aprendizaje

Planificación

La metodología más ampliamente utilizada resulta ser la planificación (es decir, determinar de forma previa las respuestas del robot ante toda posible situación). La mayoría de los investigadores utilizaron el enfoque de las tablas de movimiento (gait control tables), para establecer los movimientos necesarios para realizar locomoción [But02, Jor04, Kur02, Yim00]. Este enfoque consiste en asociar un conjunto de acciones (posiblemente secuenciales) a un estado, de forma que al encontrarse el robot en dicho estado, realice las acciones indicadas. Habitualmente se utilizan estructuras asociativas como tablas para almacenar las relaciones, de lo que proviene el nombre de este enfoque.

En muchos trabajos se describe cómo utilizar esta técnica para varios aspectos involucrados en la tarea de reconfiguración. Algunos estudian el problema de la planificación de movimientos para la reconfiguración inmediata [Yos01a, Yos01b, But02], mientras otros utilizan planificación para determinar el movimiento a nivel global, como por ejemplo, planificación de trayectoria [Yos01a, Yos01b].

La planificación parece ser una técnica exitosa para afrontar los problemas involucrados en esta tarea, pero de alguna manera fuerza a los desarrolladores a "preparar" las reacciones del robot (al utilizar una tabla de movimientos previamente desarrollada, por ejemplo), e imponen fuertes restricciones de hardware sobre el robot (dado que la mayoría de los algoritmos de planificación requiere de gran capacidad de cómputo por parte del procesador), que pueden a veces incluso atentar en contra de los principios básicos de los Robots Modulares Autoreconfigurables (es decir, ser autónomos y autocontenidos).

Evolución

Otro enfoque utilizado es el de la evolución (utilizando algoritmos genéticos). Algunos autores decidieron dejar evolucionar los controladores, a fin de obtener mejores controladores que los obtenidos de formas alternativas (como la generación de controladores mediante algoritmos de planificación, o simplemente codificados manualmente) [Kam03].

El uso del enfoque evolucionario permite a los investigadores disminuir el sesgo que introducen en los controladores, y al dejar que los algoritmos genéticos determinen los controladores óptimos, pueden incrementar el rendimiento del robot, lo que a su vez reduce la relación costo-rendimiento.

Sin embargo, esta metodología tiene el inconveniente de que para lograr conseguir un controlador suficientemente bueno, el algoritmo genético debe

ser evaluado durante un lapso de tiempo considerable (lo que disminuye las posibilidades de este tipo de robots de operar en tiempo real). Además, dado que la *función de fitness* utilizada debe ser adaptada para cada robot, cada vez que se modifique la estructura física del mismo, se deberá regenerar el controlador. Nuevamente, esto afecta la capacidad de adaptarse al medio ambiente de forma *veloz*.

Aprendizaje

Sorprendentemente, esta metodología no parece haber sido adoptada. El enfoque del aprendizaje tiene la ventaja de permitir al robot adaptarse continuamente al entorno y aprender de la interacción.

Desde el punto de vista del robot, una alteración en la percepción del entorno se puede deber tanto a una modificación real del medio, como a una modificación de las capacidades de percepción del robot. Por lo tanto, si el robot es capaz de resolver situaciones en un entorno altamente dinámico, también será capaz de modificar su comportamiento para ajustarse a distintas capacidades de percepción (por ejemplo cuando disminuyen, en el caso de dañarse algún sensor).

Esta característica le permitiría al robot ser inmerso en el entorno real en una temprana etapa del desarrollo, evolucionando el software y el hardware simultáneamente.

Al aprender desde tempranas etapas de la vida del robot, otro aspecto que se gana es poder verificar la robustez de la implementación física del robot, así como de su controlador. Sería posible realizar cambios en el entorno, y evaluar cómo reacciona el robot, y al mismo tiempo, el robot estaría aprendiendo a adaptarse a ese tipo de cambios. Por lo tanto, el desarrollo y la verificación estarían sucediendo simultáneamente, efectivamente reduciendo el tiempo de desarrollo, y con ello su costo.

Naturalmente, ninguna metodología es perfecta, y el aprendizaje tiene sus propios inconvenientes. El mayor de ellos es que es necesario invertir gran cantidad de tiempo para lograr aprender un comportamiento que resuelva el problema planteado de forma eficiente (pero dado que el aprendizaje sucede desde una temprana etapa, parte del tiempo invertido sería en realidad tiempo ganado). Otro inconveniente es que para el aprendizaje se requiere de más poder de cómputo que al utilizar tablas de movimiento, por ejemplo, que permitieran al robot realizar un movimiento prefijado y limitado; pero al mismo tiempo le permitirían al robot adquirir comportamientos competentes, e incluso refinarlos continuamente (sin mencionar el hecho de poder adaptar el comportamiento a cambios en el entorno).

Según lo visto, como un cambio en el entorno es indistinguible de un cam-

bio en la percepción que el robot tiene del mismo, permitiéndole adaptarse continuamente al entorno, la técnica de aprendizaje le proporciona al robot la habilidad de desenvolverse frente a eventos inesperados y a lidiar con situaciones imprevistas o a reaccionar ante eventos como, por ejemplo, la falla de algún componente de hardware.

2.5. Los competidores

En esta sección se destacan las principales características de los Robots Modulares Reconfigurables (y autoreconfigurables) existentes.

2.5.1. Robots Tipo Cadena

CONRO

Este robot fue desarrollado por Will y Shen en el Polymorphic Robotics Laboratory del Information Sciences Institute de la University of Southern California (USC/ISI), en el año 2000. Se trata de un robot homogéneo cuyos módulos miden aproximadamente 10cm de largo y pesan 115g. Cada módulo, según se puede observar en la figura 2.1, está compuesto por dos motores (servomotores estándar, para equipos de radiocontrol), dos baterías, CPU (microcontrolador stamp II) y sensores IR. El mecanismo de conexión es bipartito (1 conector hembra y 3 macho por cada módulo). Los módulos tienen dos grados de libertad, que le permiten realizar movimientos verticales (pitch) y horizontales (yaw). Los sensores IR son utilizados para la transferencia de datos y comunicación entre módulos.

Polybot

Este robot fue desarrollado por Yim et al. en el año 2000, en el Xerox Palo Alto Research Center. Hasta el momento se diseñaron tres generaciones de este robot (G1, G2 y G3). Se trata de un robot homogéneo cuyos módulos constan de un motor (en el robot G3, es una versión modificada de un motor Maxon de 32mm), batería, CPU (Motorola PowerPC 555, en el robot G3), y sensores IR (utilizados para la comunicación entre módulos). El mecanismo de conexión utilizado se basa en conectores hermafroditas. En la figura 2.2 se puede ver una imagen de la versión G2.

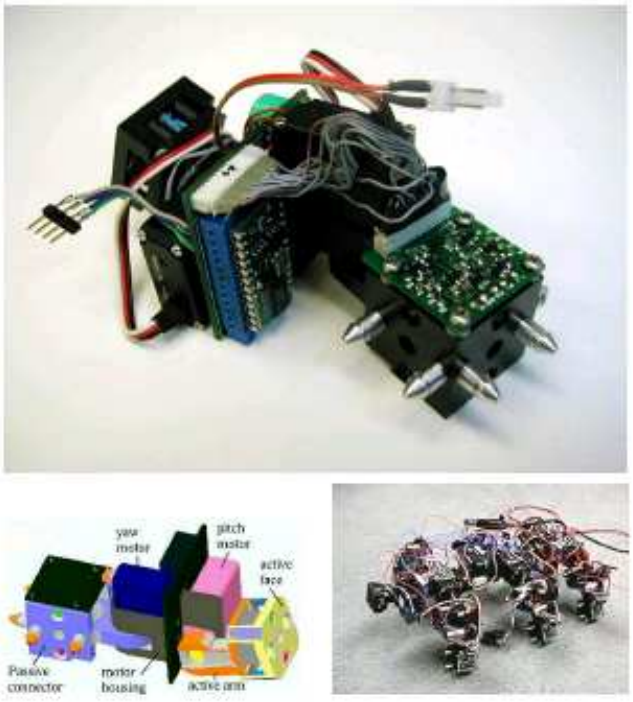

Figura 2.1: Módulo CONRO (arriba). Esquema CAD de un módulo CONRO (izquierda). Un robot hexápodo construido a partir de módulos CONRO (derecha). (USC Information Sciences Institute)

ACM

El ACM (Active Cord Mechanism) es un robot homogéneo, que fue desarrollado por Hirose et al. en 1993 (Tokyo Institute of Technology). Se lo utilizó principalmente para tratar de simular el movimiento de las serpientes. Si bien puede ser utilizado para manipulación y locomoción, y opera en tres dimensiones, este robot no tiene la capacidad para autoreconfigurarse.

2.5.2. Robots Tipo Malla

Metamorphic Robotic System

Este robot fue desarrollado en Johns Hopkins University, por Chirikjian durante 1994, y luego continuado por Chirikjian y Pamecha en 1996. Se trata de un robot homogéneo compuesto por módulos hexagonales (según se ve en la figura 2.4), los cuales disponen de tres grados de libertad (en realidad los módulos están compuestos por seis varillas interconectadas, con actuadores

Figura 2.2: Esquema CAD de un módulo Polybot G2 (izquierda). Robot Polybot G2 en una estructura circular (derecha).

Figura 2.3: Robot ACM-R1.

ubicados cada dos juntas). Si bien cada módulo puede conectarse, desconectarse y rotar alrededor de otros módulos, las posibilidades de reconfiguración de este robot están limitadas al plano (ya que los actuadores operan en un nivel bidimensional). Cada módulo posee conectores de ambos géneros, por lo que se trata de un mecanismo de conexión bipartito, aunque dada la ubicación de los mismos, un conector de un género nunca podrá aparearse con otro conector del mismo género (lo que simplifica la tarea de conexión). Si bien un robot autoreconfigurable debe disponer de su propio CPU, en el trabajo publicado en 1996, aún no estaba integrado, y se utilizaba un controlador externo (Motorola 68HC11).

Crystalline

El robot Crystalline, desarrollado en el Dartmouth College por Rus y Vona en el año 2000, es otro ejemplo de un robot homogéneo. Sus módulos

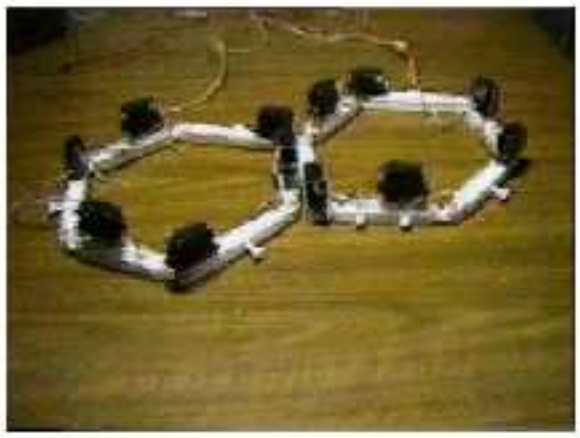

Figura 2.4: Dos módulos del robot Metamorphic.

están compuestos por un CPU (Atmel AT89C2051), un sistema de comunicación IR y una batería. Utiliza un mecanismo de conexión bipartito de tipo mecánico. La reconfiguración se logra al expandir y contraer los módulos y modificar las conexiones entre distintos módulos. Si bien el diseño es planar, es posible extenderlo a tres dimensiones. Un módulo Crystalline puede verse en la figura 2.5.

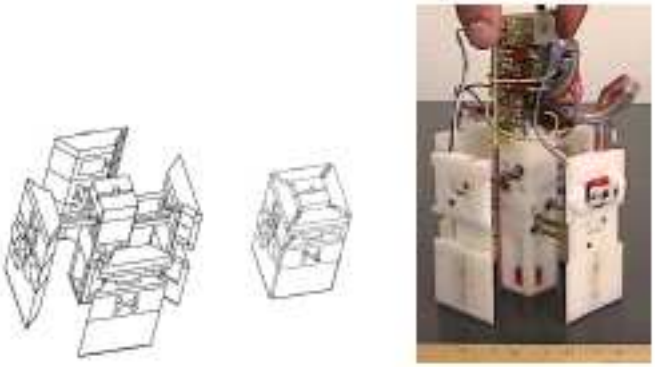

Figura 2.5: Esquema de módulo Crystalline expandido y contraído (izquierda). Módulo Crystalline real (derecha).

Fractum

Desarrollado por Tomita y Murata en 1999 (National Institute of Advanced Industrial Science And Technology – AIST), este robot homogéneo consta de módulos que disponen de CPU (Zilog Z80), un sistema de comunicación óptico, baterías y tres ruedas esféricas. Su mecanismo de conexión bipartito consta de 6 conectores (3 de cada género), utilizando conectores electromagnéticos así como imanes pasivos. Este robot, que puede observarse en la figura 2.6 solamente puede adoptar configuraciones planares.

Figura 2.6: Tres robots de tipo Fractum.

Micro Unit

Micro Unit, desarrollado en 2002 por Yoshida et al. en el National Institute of Advanced Industrial Science and Technology (AIST), se centra en la miniaturización mediante la utilización de materiales inteligentes como el SMA (Shape Memory Alloy), tanto para los conectores como para los actuadores. Su implementación se puede observar en la figura 2.7.

RIKEN Vertical

El robot RIKEN Vertical, desarrollado en The Institute for Physical and Chemical Research (RIKEN) por Hosokawa et al. en 1998, es un ejemplo interesante, dado que si bien este robot pueda reconfigurarse únicamente en un plano, a diferencia del resto de los robots similares, éste lo hace en el plano vertical. Sus módulos tienen dos grados de libertad, y utiliza conectores magnéticos. En la figura 2.8 se puede ver ilustrado este ejemplar.

Figura 2.7: Módulo Micro Unit (izquierda). Estructura formada por módulos Micro Unit (derecha).

Telecube

Este robot desarrollado en PARC (Suh, 2002), según se puede apreciar en la figura 2.9 es una versión tridimensional del robot Crystalline. En este robot homogéneo, se utilizan imanes móviles para el acoplamiento.

MEL 3D Unit

El robot MEL 3D fue desarrollado por Murata et al. en el Mechanical Engineering Laboratory del AIST, en el año 2000. Este robot homogéneo es capaz de realizar reconfiguración en tres dimensiones.

Molecule

Este robot fue desarrollado por Kotay, Rus y McGray en 1998 (Dartmouth College). Un módulo consiste de dos "átomos" con dos grados de libertad interconectados. Este robot homogéneo es capaz de realizar reconfiguración tridimensional. Una fotografía de estos módulos puede verse en la figura 2.10.

M-TRAN

En el caso del M-TRAN (Modular Transformer), desarrollado por Murata et al. durante el año 2000 en el Intelligent Systems Research Institute del National Institute of Advanced Industrial Science and Technology, se trata de un robot homogéneo cuyos módulos tienen 3 conectores en cada extremo

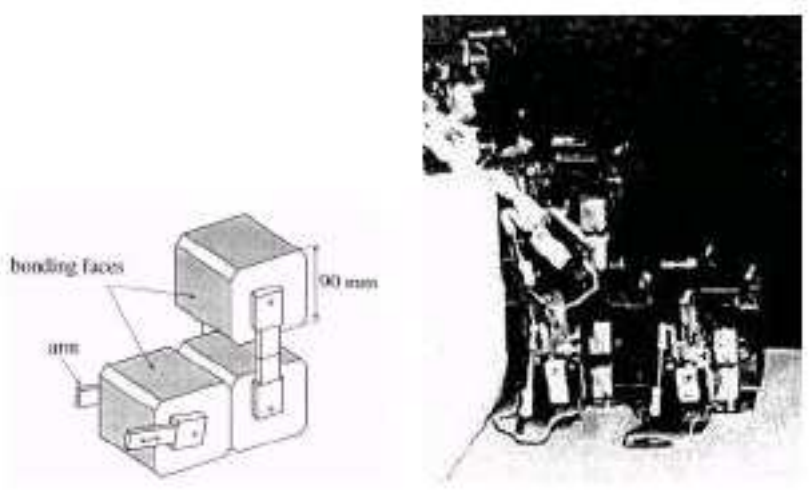

Figura 2.8: Esquema de módulos RIKEN Vertical (izquierda). Módulos RIKEN Vertical en una configuración apta para subir escalones (derecha).

y disponen de dos grados de libertad. El mecanismo de conexión es bipartito y puede realizar autoreconfiguración en tres dimensiones. Este robot se estudiará con mayor detalle en el capítulo 3.

I(ces)-Cubes

I(ces)-Cubes fue desarrollado por Ünsal y Khosla en el año 2000, en el Institute for Complex Engineered Systems de la Carnegie Mellon University. Este robot se caracteriza por ser uno de los pocos ejemplares heterogéneos. Existen dos tipos de módulos: el módulo "link", que mide 80mm de lado y pesa alrededor de 370g, posee tres grados de libertad así como dos conectores "macho" que pueden manipular los módulos "cube" (que son completamente pasivos), y contienen exclusivamente conectores "hembra" en cada una de sus caras. Este robot es capaz de realizar autoreconfiguración. Un par de módulos, "cube" y "link" pueden verse en la figura 2.11.

Fracta

Este robot (ilustrado en la figura 2.12) es la versión en tres dimensiones del robot Fractum. Fue desarrollado por Murata et al. hacia 1998, en el Mechanical Engineering Laboratory del AIST. Este robot homogéneo fue el primer sistema capaz de realizar autoreconfiguración. Con sus módulos similares a cubos de aproximadamente 25cm de lado y 7Kg de peso, este

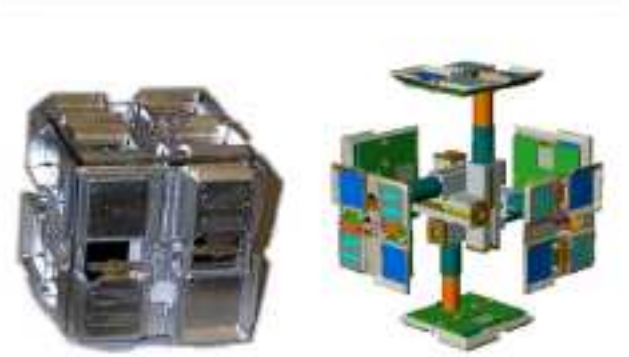

Figura 2.9: Un robot Telecube (izquierda). Esquema CAD de un módulo Telecube en estado expandido (derecha).

robot con 12 grados de libertad por módulo, ilustra claramente los problemas típicos que sufren los sistemas homogéneos: los módulos tienden a ser grandes y poco manipulables.

Proteo

Proteo fue desarrollado por Bojinov et al. en el año 2000 (Xerox Palo Alto Research Center). Este robot homogéneo, consta de módulos en forma de dodecaedros rómbicos con 12 caras de conexión idénticas. Para el mecanismo de conexión, utiliza conectores electromagnéticos que le permiten realizar movimientos mediante una combinación de rotaciones de los módulos alrededor de las aristas de sus caras. Disponer de 12 caras de conexión genera una alta complejidad y gran costo. Aún no existen implementaciones reales de este robot, y todo el trabajo sobre el mismo se ha realizado en simulaciones.

Miniaturized Self-Reconfigurable Robot

El robot Miniaturized Self-Reconfigurable Robot, desarrollado en el Mechanical Engineering Laboratory del AIST, fue presentado por Yoshida et al. en 1999. Se trata de un robot homogéneo de pequeñas dimensiones (los módulos miden 40mm de alto, 50mm de ancho y pesan 80g). El sistema fue diseñado para utilizar actuadores SMA que le permiten reducir su tamaño. El mecanismo de conexión que utiliza es del tipo bipartito. Su reducido tamaño le confiere un torque limitado y lo restringe en el rango de movimientos que puede realizar.

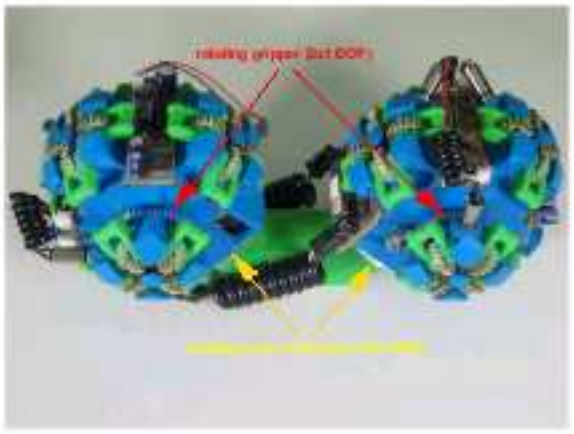

Figura 2.10: El robot Molecule.

Semi-Cylindrical Reconfigurable Robot

Otro robot homogéneo capaz de operar en tres dimensiones es Semi-Cylindrical Reconfigurable Robot. Este sistema, presentado por Kurokawa en el año 2000 (desarrollado en el AIST), consta de módulos formados por dos cubos semicilíndricos dotados de un servomotor cada uno. El mecanismo de conexión de este robot utiliza conectores magnéticos, así como resortes SMA, para lograr el desacoplamiento. Los pequeños imanes le confieren una fuerza limitada, y su capacidad de reconfiguración se ve acotada debido a la naturaleza bipartita de sus conectores. Sin embargo, este robot tiene la capacidad de autoreconfigurarse en tres dimensiones.

TETROBOT

Desarrollado por Hamlin y Sanderson en 1996 (Rensselaer Polytechnic Institute), TETROBOT es un robot homogéneo, que introdujo un diseño novel al utilizar juntas esféricas. Este robot, que puede observarse en la figura 2.13 requiere que la reconfiguración sea realizada manualmente.

CEBOT

El sistema robótico celular (CEBOT) fue desarrollado por Fukuda (Nagoya University) y Kawauchi (Science University of Tokyo) en 1990. Se trata de un robot homogéneo de diseño celular en el que cada célula tiene capacidades de sensado y cómputo limitadas. Este robot no es capaz de autoreconfigurarse.

Figura 2.11: Módulos de tipo “cube” y “link” del robot I-Cubes.

2.5.3. Comparación

Como se puede observar en la tabla A.1, incluida en el apéndice A, la mayor parte de la investigación en robots reconfigurables se realizó en sistemas homogéneos, con el objetivo de operar en tres dimensiones.

En la mayoría de los casos, los módulos propuestos pueden desplazarse por encima de sus módulos vecinos, y el robot compuesto por este tipo de módulos es capaz de autoreconfigurarse.

La cantidad de grados de libertad varía entre 0 y 12, siendo 2 el valor predominante.

No parece haber sido desarrollado un mecanismo de conexión tolerante a fallas aún. La mayoría de los mecanismos de conexión dependen de que el módulo esté en condiciones de desprenderse por sus propios medios [Jan01].

El tamaño de los módulos varía bastante (desde los más grandes, como Fracta, hasta los más pequeños como Micro Unit o Miniaturized Self-Reconfigurable Robot), pero el término medio se encuentra en el rango de los 40mm - 80mm de ancho.

2.6. Las soluciones

En esta sección se discuten algunas de las soluciones propuestas a los problemas descriptos en la sección 2.4.

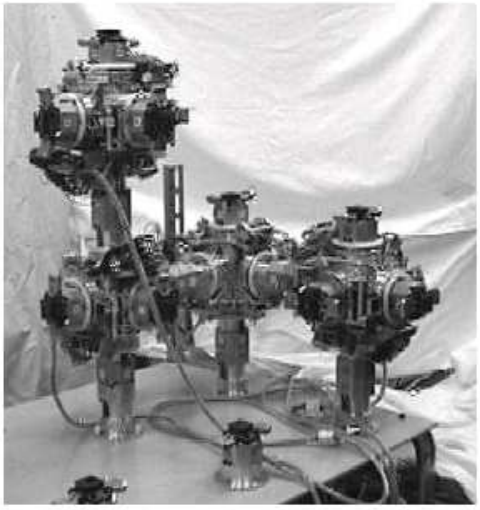

Figura 2.12: Módulos del robot Fracta.

Figura 2.13: TETROBOT.

2.6.1. Reconfiguración

En la mayoría de los Robots Modulares Autoreconfigurables se intentó resolver el problema de la reconfiguración utilizando tablas de movimientos predefinidas.

Según este enfoque, el robot solamente debe determinar la entrada de la tabla que corresponde a su estado actual, y efectuar los movimientos descriptos por la misma.

Esta solución es simple y eficiente, pero tiene un inconveniente bastante importante: las tablas deben ser construidas previamente, y transferidas de alguna manera al robot.

Dado que en este caso la reconfiguración resulta ser básicamente una

búsqueda en la tabla, el costo de este proceso suele ser relativamente bajo (en tiempo de cómputo).

Esta forma de resolver el problema de reconfiguración parece ser adecuada para los robots de tipo cadena, en donde, como se vio anteriormente, la reconfiguración es independiente de la locomoción, y por lo tanto no debería ser realizada frecuentemente (solamente se reconfigura el robot cuando es necesario adquirir una forma de locomoción diferente, y por lo tanto, la cantidad de estados involucrados en la reconfiguración es baja, por lo que el tamaño de la tabla queda acotado).

Otro enfoque utilizado es evolucionar dicha tabla [Kam03]. Este enfoque es menos popular, pero no deja de ser interesante, dado que intenta evitar algunas de las limitaciones del enfoque anterior. Mediante la evolución, el robot es capaz de operar en una variedad de escenarios, que pueden no haber sido anticipados, dado que es capaz de construir su propia tabla de movimientos a medida que surjan circunstancias para las que no tenga soluciones predeterminadas.

Con éste último método será necesario disponer de cierto tiempo antes de que el robot pueda desenvolverse tan bien como al utilizar directamente las tablas pregeneradas, pero dado que el robot puede ser utilizado antes de conseguir una tabla de movimientos completa, el tiempo invertido puede ser equivalente al tiempo que le lleva a una persona a desarrollar una tabla de movimientos correcta y completa.

Acoplamiento

Un ejemplo interesante se da en el robot Polybot, en donde los investigadores desarrollaron un mecanismo que le permite al robot acoplarse por sus propios medios, utilizando solamente un par de sensores IR [Rou00].

Este ejemplo muestra que si bien el problema de acoplamiento no es sencillo de resolver, dado que la tarea involucra movimiento controlado, sensado y "reconocimiento" del módulo "destino", entre otras cosas, existen implementaciones que logran resolver este problema razonablemente bien.

Otra solución para este problema consiste en utilizar conectores magnéticos [Yos01a, Pat04, Mur94, Suh02, Jor04, Kur02], de forma que una vez que los módulos se encuentran lo suficientemente cerca, las fuerzas magnéticas los acoplan automáticamente. Esta es una solución inteligente, pero requiere que los módulos sean acercados lo suficiente como para permitir que las fuerzas magnéticas puedan atraer los módulos entre sí.

Mecanismo de conexión

Los mecanismos de conexión son un aspecto indispensable en un Robot Modular Autoreconfigurable, y por lo tanto éste resulta ser un problema central en su diseño.

La mayoría de las soluciones propuestas se basan en alguna forma de acoplamiento mecánico, utilizando conectores bipartitos (macho/hembra) [Jor04, Sto02, Kur02, Cas02]. Si bien este enfoque es simple y sencillo de implementar, tiene algunos inconvenientes. El paradigma bipartito reduce las posibilidades de reconfiguración, dado que un módulo solamente puede conectarse con otro módulo mediante un conector del género opuesto. Si esto no fuera posible (debido a la configuración de los módulos), el robot no podrá cambiar su configuración. Por otro lado, la mayoría de los conectores mecánicos tienen que ser controlados por el módulo, por lo que si el mismo falla, no podrá ser desconectado del robot (lo que puede degradar el rendimiento del mismo).

Otro intento de resolver este problema, consiste en utilizar conectores magnéticos [Yos01a, Pat04, Mur94, Suh02, Jor04, Kur02]. En este caso, los conectores son más robustos (los imanes no pueden fallar), y la restricción bipartita puede evitarse utilizando electroimanes (que pueden modificar su polaridad). Sin embargo, los electroimanes requieren que el módulo funcione correctamente, dado que necesitan corriente para generar el campo magnético. En el caso de pérdida de corriente, es posible desacoplar el módulo, pero nada impide que el módulo sea incapaz de desacoplarse aún teniendo corriente (por ejemplo si falla el circuito encargado de la conexión/desconexión, puede ser imposible invertir la polaridad del electroimán).

Además de utilizar imanes pasivos, algunos diseños utilizan resortes SMA (*Shape Memory Alloy*), que pueden producir suficiente fuerza para separar los imanes de forma de desacoplar los módulos. Un beneficio de estos resortes es que son pequeños y requieren muy poca cantidad de energía para operar.

2.6.2. Locomoción

Dado que la locomoción está tan intrincadamente ligada al problema de reconfiguración, las soluciones propuestas para dicha tarea resuelven los problemas que puedan aparecer durante la locomoción.

Además, generalmente, el mayor problema a resolver durante la tarea de locomoción es disponer de la cantidad de hardware necesario para almacenar las grandes tablas de movimientos, o disponer de suficiente memoria para almacenar los conjuntos de reglas para los autómatas celulares que controlan a los robots. Dado que estos problemas de hardware son comunes a todos los

tipos de robots, no se los estudia en esta oportunidad.

2.7. Los destacados

En esta sección se examinan con mayor detalle los robots que se consideran los más exitosos entre los propuestos en la sección 2.5.

La breve reseña sobre cada uno de estos robots solamente tiene la intención de incentivar al lector a consultar bibliografía adicional al respecto de cada uno. No se pretende proveer un análisis detallado acerca de las capacidades o limitaciones de cada robot.

2.7.1. M-TRAN [Mur02, Yos01a, Yos01b, Kam03, Kur02, But02]

Este es uno de los casos más exitosos de un Robot Modular Autoreconfigurable. Una de las razones para ello es que este robot resulta un híbrido entre los robots de tipo cadena y los robots de tipo malla. Esta característica le proporciona las ventajas típicas de ambos tipos de robots, y reduce el impacto de sus limitaciones respectivas.

Un robot M-TRAN puede operar (desde el punto de vista de su morfología) como un robot de tipo cadena y realizar locomoción sin reconfigurarse, o como un robot de tipo malla que se reconfigura continuamente para lograr desplazarse.

Su hardware es muy sencillo, y por tal motivo puede ser producido de forma relativamente económica. Debido a su tamaño, no tiene demasiadas limitaciones físicas, exceptuando que no puede efectuar demasiada fuerza sobre sus módulos (restringiendo su capacidad de levantar grandes pesos).

Su mecanismo de conexión, utilizando imanes pasivos para el acoplamiento, y resortes SMA para el desacoplamiento, le provee una gran flexibilidad al momento de reconfigurarse.

Si bien este robot no es tan popular como los otros dos candidatos presentados en esta sección, tiene ciertas ventajas sobre los mismos que lo posiciona en primer lugar. Estas ventajas incluyen el hecho de que gracias a su hardware simple y relativamente económico, puede ser utilizado en ocasiones en las que debido al factor económico, las otras opciones quedan excluidas. Otro punto a su favor consiste en que al ser híbrido, puede operar como un robot de tipo cadena y como un robot de tipo malla, permitiendo la experimentación en ambos campos de estudio, sin requerir dos robots diferentes.

Sin embargo, este robot ha sido considerado principalmente un robot de tipo malla, y por ende, gran parte de la investigación sobre el mismo ha sido

centralizada en la teoría de autómatas celulares para proveerle capacidades de locomoción [But02].

Aún así, se puede ver con facilidad que su diseño le permite realizar locomoción sin reconfigurarse, por lo que ambos aspectos pueden ser explorados simultáneamente, permitiendo a los investigadores encontrar el equilibrio entre ambos diseños.

2.7.2. Polybot [Yim00, Yim02, Rou00]

Este robot puede considerarse el más popular, y está siendo continuamente mejorado. Si bien las simulaciones y las demostraciones con robots reales han mostrado un potencial altamente prometedor, este robot sigue teniendo un aspecto desfavorable: su hardware no es tan simple como en el caso del M-TRAN, y al ser más complejo, su costo aumenta.

Sus investigadores en PARC (Palto Alto Research Center), están trabajando hace tiempo sobre este robot, y han demostrado su potencial en varias ocasiones.

Aún cuando este robot no logre cumplir con los requerimientos de un grupo de investigación, Polybot no debe ser ignorado, pues sus creadores muestran continuamente soluciones noveles a problemas reales, que pueden generar avances importantes dentro del campo de los Robots Modulares Autoreconfigurables.

2.7.3. CONRO [Cas02, Sto02, Sto03, She]

CONRO es un competidor directo de Polybot, pero debido a una limitación particular le corresponde el tercer puesto: su hardware es el más complejo entre los tres robots destacados.

Es posible que no sea más costoso de producir que los otros modelos, pero dado que sus módulos son de mayor tamaño y peso, las oportunidades para su utilización también se ven restringidas.

2.8. Discusión

A continuación se presentan algunas ideas sobre cómo mejorar los robots propuestos, y sobre algunos requisitos que un robot debería cumplir a fin de ser más eficiente.

2.8.1. Metodología

Uno de los principales argumentos de esta sección es que de acuerdo al conocimiento adquirido en base a la bibliografía disponible, un aspecto se ha visto ignorado repetidamente: la utilización de aprendizaje en las tareas de reconfiguración y locomoción.

Esta metodología tiene tantas ventajas que resulta sorprendente no haber encontrado referencias acerca de su utilización en el campo de los Robots Modulares Autoreconfigurables.

Imbuir al robot con capacidades de aprendizaje, indudablemente lo haría más robusto y versátil (cualidades altamente estimadas en un Robot Modular Autoreconfigurable).

2.8.2. Diseño de Hardware

En relación al diseño del hardware, un robot exitoso debe disponer de un hardware que sea económico de producir y fácil de manipular (para permitir la reconfiguración).

Para que un diseño logre cumplir estos objetivos, varios aspectos deben ser tomados en cuenta. Por un lado, se debe garantizar una óptima utilización y ubicación de los recursos (para maximizar la efectividad de los sensores, actuadores y energía disponibles). Por otro lado, debe contemplar la posibilidad de que un módulo falle, y cómo actuar ante tales circunstancias.

2.9. Conclusiones

En este capítulo se ha presentado brevemente el estado del arte en el campo de la investigación sobre Robots Modulares Autoreconfigurables. Se han visto los mayores puntos de interés en el mismo, y se han presentado algunos de los principales problemas y cómo fueron solucionados por las diferentes propuestas.

Se han presentado varios robots, analizando sus características principales, cuando presentaban una solución novel o interesante a algún problema inherente a este campo.

Luego se han propuesto tres robots destacados como los más exitosos en el área, comentando brevemente sus mayores virtudes. También se presentaron algunas reflexiones acerca de las condiciones que debe cumplir un robot de este tipo para ser exitoso, y se han discutido algunos de los problemas existentes en los diseños actuales.

Resulta claro que este campo aún es joven dentro de la disciplina de la robótica. Aún así, ya se han visto avances importantes en el mismo. Toda-

vía queda un largo camino por recorrer antes de que los Robots Modulares Autoreconfigurables sean capaces de operar exitosamente en el mundo real, realizando las tareas para las que mejor están capacitados.

La aparición de los Robots Modulares Autoreconfigurables marcó un hito en la robótica. Sus posibilidades de acción son muchas, y su utilización puede extenderse a diversas áreas de aplicación. Es simplemente una cuestión de tiempo antes de que se vea a este tipo de robots rescatar a personas de áreas de desastre o verlos explorar planetas distantes autónomamente.

Capítulo 3

Hipótesis, Materiales y Métodos

3.1. Objetivo

Según se mencionó en la sección 2.8, no se ha encontrado bibliografía acerca de la utilización de la metodología de aprendizaje para la obtención de comportamientos que resuelvan las tareas de locomoción y reconfiguración en Robots Modulares Autoreconfigurables.

De acuerdo al conocimiento acerca del *Aprendizaje por Refuerzo* y su campo de aplicación, la hipótesis que se desea verificar es que los Robots Modulares Autoreconfigurables pueden ser dotados de recursos para que aprendan comportamientos adecuados y eficientes para resolver tareas de locomoción y reconfiguración sin supervisión humana.

Se determinó que el robot más apropiado para utilizar en el marco de este trabajo era el robot M-TRAN, y que la experimentación se llevaría a cabo mediante simulaciones, para lo cual fue necesario desarrollar un simulador que contemple todas las características del problema: la simulación de la estructura de un robot M-TRAN así como la implementación del algoritmo de aprendizaje.

En este capítulo se introducen los elementos utilizados a lo largo de este trabajo. A continuación se describen las características y propiedades del robot utilizado, así como del simulador desarrollado y del algoritmo de aprendizaje implementado.

3.2. M-TRAN

M-TRAN (Modular Transformer) es un Robot Modular Autoreconfigurable, desarrollado conjuntamente por AIST y Tokyo-Tech desde 1998.

El diseño de este robot presenta las ventajas de ambos tipos de robots

reconfigurables (tipo cadena y tipo malla, descriptos en el capítulo 2). El diseño híbrido, y la forma particular de los bloques que componen los módulos de este robot, son aspectos claves que hacen del mismo un sistema robótico muy flexible.

Un módulo M-TRAN está compuesto por dos bloques, interconectados mediante un eje (ver figura 3.1). Cada una de las tres superficies planas de estos bloques es capaz de conectarse y acoplarse con la superficie de otro módulo. Dado que los bloques tienen un género definido, las superficies de un bloque activo solamente pueden acoplarse con las superficies de un bloque pasivo, y viceversa. Sin embargo, el hecho de disponer de tres superficies conectores por bloque, permite que los bloques pertenecientes a diferentes módulos puedan acoplarse en múltiples posiciones (ver figura 3.2).

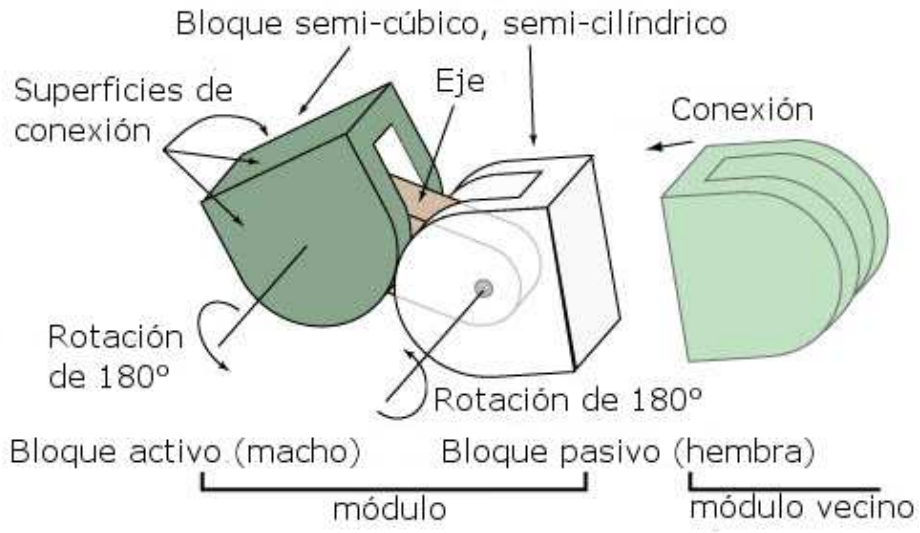

Figura 3.1: Representación esquemática de un módulo M-TRAN.

Figura 3.2: Acoplamiento entre dos módulos, en diferentes posiciones relativas.

Este diseño, como ya fue mencionado, le permite al robot desenvolverse como si fuera un robot de tipo malla. Si cada bloque se orienta respecto de

los otros de manera que sus motores se encuentren en ángulos de -90, 0 ó 90 grados, todos los módulos quedarán alineados en una estructura similar a una grilla tridimensional.

Por otro lado, este robot también es capaz de utilizarse como un robot de tipo cadena. Si todos los motores (o gran parte de ellos) son activados sincrónicamente, el robot puede realizar movimientos con gran flexibilidad.

Cada módulo M-TRAN dispone de su propio controlador, lo que le provee de cierta cuota de "inteligencia" y autonomía. Los distintos módulos interactúan mediante sus controladores, formando en conjunto, un Sistema Autónomo Distribuido.

Actualmente, el robot M-TRAN se encuentra en su tercer implementación. La primera versión del mismo, fue desarrollada hacia 1998. Esta implementación se caracterizaba por utilizar conectores magnéticos (ver apartado 3.2.1), y era controlado de forma remota a través de un cable. En su segunda implementación (2002), las mejoras más significativas incluyeron una mayor autonomía del robot, mediante la utilización de baterías, un control descentralizado y distribuido y comunicación inalámbrica hacia los módulos (unidireccional). Esta versión también era un poco más pequeña que su antecesora. Finalmente, la tercera versión de este robot, realizada hacia 2005, eliminó los conectores magnéticos, remplazándolos por conectores mecánicos (ver apartado 3.2.1), y proveyó a los módulos de la capacidad de comunicación bidireccional inalámbrica, así como de sensores IR de proximidad.

Figura 3.3: Distintas implementaciones del robot M-TRAN: M-TRAN I (izquierda), M-TRAN II (centro) y M-TRAN III (derecha).

3.2.1. Mecanismos de conexión

A lo largo de la evolución de este robot, se utilizaron conectores magnéticos, así como conectores mecánicos, para el mecanismo de conexión. A

continuación se detallarán ambos tipos de conectores, según fueron implementados.

Conectores Magnéticos

Este tipo de conectores fue utilizado en las primeras dos implementaciones del robot M-TRAN. Cada una de las superficies de conexión de los bloques que componen un módulo está dotado de magnetos permanentes. En el caso de los bloques pasivos, los magnetos se encuentran directamente sobre las superficies de conexión, mientras que en el caso de los bloques activos, se encuentran sobre una estructura móvil, denominada placa conectora.

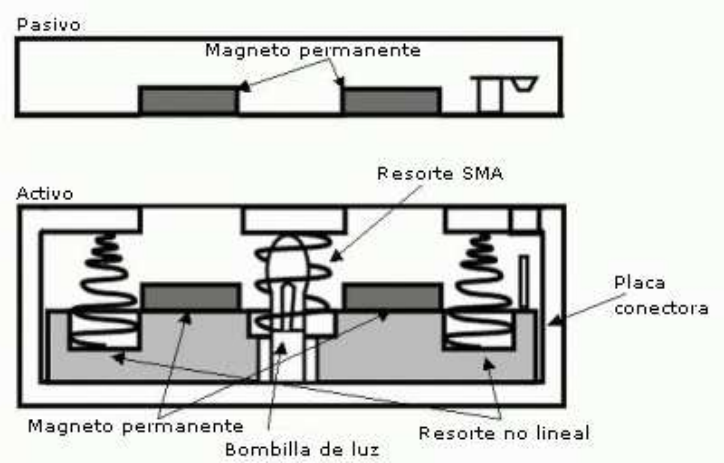

Figura 3.4: Mecanismo de conexión magnético.

El proceso de acoplamiento y desacoplamiento se desarrolla de la siguiente manera:

- Acoplamiento

  1. Las superficies de conexión entran en contacto.
  2. La placa conectora es atraída hacia la superficie de conexión gracias a la fuerza magnética. Los resortes no lineales no disponen de la suficiente fuerza para contrarrestar la fuerza magnética.

- Desacoplamiento

  1. Se enciende la bombilla de luz. La misma genera calor y calienta el resorte SMA que se encuentra a su alrededor.

2. El resorte se expande debido al calor, y separa los bloques lo suficiente como para que la fuerza magnética sea inferior a la fuerza de repulsión ejercida por los resortes no lineales.
3. La placa conectora se separa de la superficie de conexión, debido a la fuerza ejercida por los resortes no lineales.
4. La bombilla de luz se apaga, y el resorte SMA comienza a enfriarse, permitiendo un posterior acoplamiento.

Conectores Mecánicos

Debido a que el mecanismo de conexión magnético, utilizado en los robots M-TRAN I y II, resultaba lento y consumía demasiada energía para controlar la conexión entre módulos, se decidió implementar un mecanismo de conexión mecánico en el robot M-TRAN III. Este mecanismo debía permitir al robot realizar la conexión y desconexión de manera más eficiente y más veloz que sus antecesores.

Lamentablemente, no se encontró información detallada sobre la estructura de este conector mecánico, ni sobre el mecanismo de control que habilita la conexión y desconexión.

La única información que fue posible encontrar acerca del mismo son las figuras que se muestran a continuación.

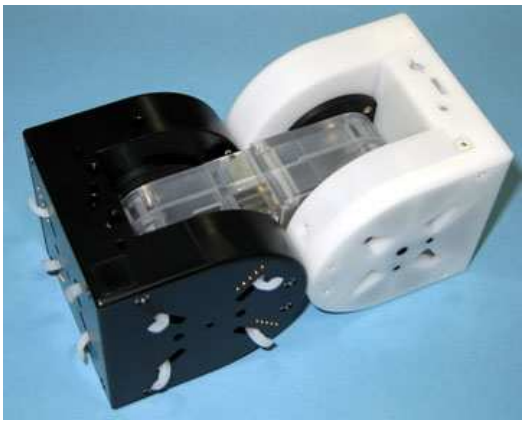

Figura 3.5: Módulo M-TRAN III.

Según lo que se puede observar en estas figuras, el mecanismo de conexión utiliza grampas en los bloques activos que son insertadas en aberturas presentes en los bloques pasivos. De acuerdo a la forma de algunas de las piezas que se pueden observar en la figura 3.6, es posible sugerir que una vez insertadas las grampas en los orificios de los bloques pasivos, las mismas

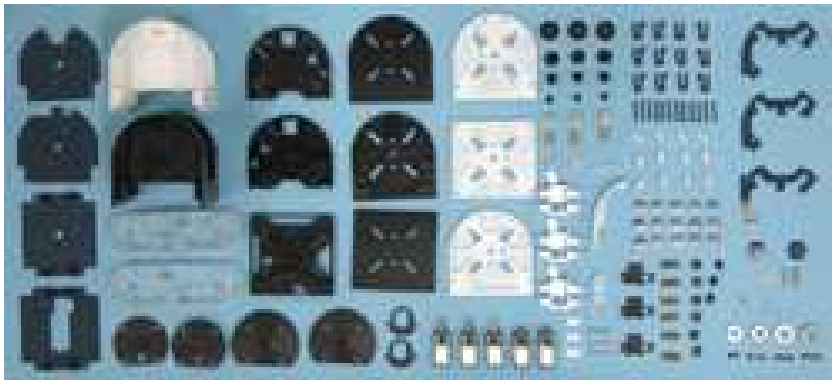

Figura 3.6: Partes mecánicas que componen un módulo M-TRAN III.

son sujetadas mediante trabas. Sin embargo, esto es todo lo que se puede especular acerca del funcionamiento de este mecanismo de conexión.

3.2.2. Ilustraciones

En el Apéndice C se incluyen imágenes que ilustran situaciones en las que un robot M-TRAN realiza tareas de reconfiguración y locomoción.

3.3. Simulador

Para realizar este trabajo, se desarrolló un simulador que describa la estructura y los movimientos de un robot M-TRAN. El mismo fue implementado utilizando una librería de simulación física para cuerpos rígidos [ODE]. Esto quiere decir, que si bien los estudios se realizaron sobre simulaciones, las mismas son relativamente realistas, gracias a que contemplan aspectos como la gravedad, masa de los cuerpos en movimiento, colisiones elásticas, fuerzas que interactúan entre los cuerpos al moverse, torque de los motores, etc.

El simulador implementado fue diseñado con la intención de presentar los resultados de la manera más realista posible. Por lo tanto, el ciclo de visualización forma parte integral del ciclo de simulación, y la misma transcurre en tiempo "normal" (bajo tiempo "normal" se entiende que el tiempo de simulación se rige aproximadamente por un factor de compresión temporal de valor 1).

El simulador fue desarrollado de forma de permitir trabajar con robots compuestos por una cantidad arbitraria de módulos y configuraciones, así como utilizar diferentes políticas de selección de acciones.

3.3.1. Detalles de implementación

Estructura

El simulador fue diseñado en base a componentes modulares. De esta forma, puede ser fácilmente modificable y extensible. El mismo se estructura básicamente en los siguientes componentes:

- Motor de simulación
- Robot
- Cerebro
- Configuración
- Descripción de la escena

El *Motor de simulación* compone el núcleo del simulador. Este componente implementa el algoritmo de simulación, e integra el manejo de eventos y la visualización con el ciclo de simulación.

El componente *Robot* describe la estructura de un robot simulado. Un robot está compuesto por *Módulos*, que a su vez están compuestos por dos *Bloques* y un *Eje*. El componente *Eje* contiene dos *Motores*, que controlan el movimiento de los bloques. Esta estructuración se corresponde con la implementación física de los módulos M-TRAN.

El *Cerebro* provee toda la funcionalidad relacionada con la "inteligencia" que dispone el robot. En este componente se implementa la política utilizada por el robot para determinar las acciones a seguir, así cómo el algoritmo de aprendizaje utilizado.

Los componentes de *Configuración* y *Descripción de la escena* son dos archivos de texto, utilizados para la configuración de los experimentos. El archivo correspondiente a la *Configuración* define los valores de las unidades utilizadas para describir al robot, el entorno, las fuerzas que interactúan, etc., mientras que el archivo correspondiente a la *Descripción de la escena* describe los componentes que conforman un experimento: los objetos que se encuentran en el entorno simulado, como las paredes y el suelo, así como los objetos que forman parte activa de la simulación, es decir, los robots (puede haber más de uno). En este archivo también se define el tipo de *Cerebro* que será implementado (y con ello la política que utilizará), así como la cantidad de módulos que componen un robot y su configuración inicial.

Políticas de aprendizaje

Para implementar las políticas utilizadas durante la fase de aprendizaje en los experimentos realizados, se utilizó una política de tipo ϵ-greedy. Este tipo de políticas permite combinar etapas de exploración con etapas de explotación. Para disminuir la cantidad de exploración realizada, a medida que transcurre un episodio, se modificó el valor de ϵ de acuerdo a una función lineal decreciente. Este hecho produce que el impacto de la exploración sea alto inicialmente, pero a medida que transcurre el tiempo, cada vez sea menor, y la mayor parte del desempeño se deba a las acciones de explotación.

Por otro lado, para poder comparar la efectividad de las políticas obtenidas mediante el aprendizaje, también se implementó una política completamente azarosa, y una política que es capaz de reproducir una secuencia de acciones preestablecidas.

Para simplificar la tarea de obtener una política de forma manual (a fin de poder comparar con las políticas aprendidas), se implementó un *Cerebro* capaz de aprender a partir de una secuencia de acciones predeterminadas. Mediante el mismo, es posible enseñarle activamente al robot, el comportamiento deseado. Esto constituye un caso de aprendizaje supervisado, que permite, a partir de unos pocos ejemplos, aprender una política capaz de reproducir determinadas acciones. Este mecanismo se puede utilizar, por ejemplo, para derivar una política de locomoción circular, a partir de los movimientos que comprenden una vuelta completa del robot.

Mecanismo de conexión

A fin de poder simular los acoplamientos, se estudiaron las condiciones de alineación y cercanía entre módulos. De esta forma, si los bloques que deben acoplarse se encuentran alineados y posicionados de forma que el acoplamiento sea posible, el simulador establece el acoplamiento entre los mismos, sin la necesidad de simular un mecanismo de conexión completo.

La condiciones de acoplamiento utilizadas se asemejan más a las condiciones requeridas por el robot M-TRAN III, con su mecanismo de conexión mecánico, que a las condiciones requeridas por el mecanismo magnético de sus predecesores. Esto se debe a que la librería de simulación física no permite simular propiedades magnéticas (ver 3.3.3).

A fin de determinar si el acoplamiento es factible, se establecen las distancias mínimas entre 4 puntos ubicados simétricamente alrededor del centro geométrico de cada superficie de conexión. Estos puntos se corresponden con las posiciones de los hoyos y de las grampas en el mecanismo del robot M-TRAN III. Si la distancia entre cada uno de los puntos de una superficie de

conexión y los puntos de la otra superficie de conexión participante del acoplamiento no superan cierto umbral, entonces se considera que las superficies se encuentran alineadas correctamente, y a una distancia lo suficientemente pequeña, como para poder acoplarse.

Una vez acoplados los módulos entre sí, las fuerzas que interactúan para mantener dicha conexión permiten que en el caso de existir un pequeño desaliñamiento, el mismo sea corregido (gracias a la propia dinámica de la simulación).

3.3.2. Problemas encontrados y sus soluciones

Durante los trabajos realizados con el simulador se encontraron diversos inconvenientes. A continuación se detallan los mismos, junto con las soluciones utilizadas para cada uno.

Regulación (tuning) de las unidades utilizadas

Para lograr una simulación lo más realista posible, se utilizaron valores de masa y tamaño de los módulos y torque de los motores, según se encontraron en distintos artículos acerca del robot M-TRAN [Mur02, Kur02]. Para confirmar estos valores, se solicitó información (mediante correo electrónico) acerca de los mismos al equipo encargado de desarrollar el robot M-TRAN, en el AIST.

Los valores utilizados inicialmente fueron:

Medida	Valor	Unidad	Observaciones
masa	400	g	módulo completo
distancia	60	mm	lado de un bloque
torque	1.9	Nm	
gravedad	9.81	m/s	
tiempo	1	seg.	

Tabla 3.1: Valores iniciales para las medidas utilizadas en las simulaciones.

Sin embargo, estos valores no pudieron ser utilizados directamente en el simulador, ya que debido a una limitación propia de la librería de simulación física utilizada, para que la simulación sea lo más estable posible, la documentación de dicha librería sugería que se ajustaran las unidades de forma de estar en el rango $0{,}1 - 10{,}0$. Luego de escalar todas las unidades (masa, distancias, tiempo, gravedad, torque) de forma consistente para cumplir este último requisito, las simulaciones obtenidas no resultaron satisfactorias, ya

que las mismas no se veían realistas. Por lo tanto, fue necesario ajustar algunos valores de forma manual, a fin de que las simulaciones se vieran más realistas.

Método de resolución del sistema de ecuaciones asociado al sistema físico

La librería de simulación física utilizada provee dos métodos para resolver el sistema de ecuaciones asociado al sistema físico. Uno de estos métodos, denominado WorldStep, utiliza un enfoque llamado "big matrix", que intenta resolver la matriz asociada al sistema de ecuaciones de forma tradicional (calculando la matriz inversa). El otro método, denominado QuickStep, utiliza un algoritmo iterativo para resolver dicha matriz. En el primer caso (WorldStep), se utiliza una variante del método de Dantzig, también conocido como SIMPLEX, mientras que el segundo método (QuickStep) utiliza una algoritmo iterativo para ir relajando paulatinamente las restricciones del sistema de ecuaciones hasta poder resolverlo.

Si bien el primer enfoque produce resultados más exactos y fieles a la realidad, es altamente sensible a situaciones en las que la matriz asociada al sistema físico se asemeja a una matriz singular. En dichos casos la simulación se vuelve inestable (en el sentido de inestabilidad numérica, es decir, la matriz asociada al sistema físico tiende a divergir a medida que pasa el tiempo, en vez de converger hacia un estado de reposo), produciendo comportamientos imprevisibles (en la simulación esto se puede observar como "explosiones", es decir colisiones muy fuertes entre los distintos componentes simulados que hacen que cada uno sea impulsado en una dirección diferente, a gran velocidad). El segundo método es más veloz que el primero y no es tan sensible al problema de las matrices singulares, pero no produce resultados con la misma precisión.

En este caso se optó por utilizar el método WorldStep, ya que la sensibilidad frente a las matrices singulares se puede modificar, alterando ciertos parámetros de la librería (en realidad, se relaja la condición de "singularidad" entendida por la librería). La librería permite modificar un parámetro denominado CFM (Constraint Force Mixing), que permite "relajar" las restricciones impuestas sobre el sistema físico, de manera de simular, entre otras cosas, juntas que actúen como resortes, o bien, materiales esponjosos. Incrementar el valor de CFM también tiene el efecto de alejar un sistema de un estado singular.

3.3.3. Limitaciones

En cuanto a las limitaciones de este simulador, se pueden dividir en dos categorías: las limitaciones derivadas del diseño e implementación misma del simulador, y las limitaciones heredadas de las librerías utilizadas.

En cuanto a las limitaciones propias del diseño

El simulador fue diseñado de manera que resultara simple realizar experimentos con el mismo, pero no con el objetivo que resultara de uso general. Por esta razón, para modificar ciertos comportamientos o aptitudes provistas por el simulador, resulta necesario modificar el código fuente del mismo. A pesar de ello, se hizo el esfuerzo para implementarlo de la manera más genérica posible, y así evitar el inconveniente de tener que modificar el código fuente del simulador en algunos casos.

La descripción de la escenas (el entorno sobre el que se realiza la experimentación), así como de la configuración del robot utilizado, que incluye la posición inicial de todos los bloques, debe ser especificada manualmente en un archivo de configuración. Si bien esto simplifica la realización de diferentes experimentos (un archivo de descripción por cada experimento), definir la posición inicial de cada bloque del robot resulta un proceso delicado, ya que un pequeño error en la posición de un bloque puede prevenir que el mismo quede acoplado, y de esta forma evitar que se obtenga la configuración inicial pretendida.

Dado que el simulador fue diseñado de manera que las simulaciones transcurrieran en tiempo "normal", la duración del aprendizaje puede requerir de una cantidad considerable de tiempo (es decir, no se aprovecha la potencial diferencia entre el tiempo simulado y tiempo de ejecución de la simulación).

En cuanto a las limitaciones heredadas

El simulador no es capaz de simular el mecanismo de conexión implementado por el robot M-TRAN (en sus versiones I y II), ya que la librería de simulación física utilizada, ODE, no provee de mecanismos de simulación de propiedades magnéticas. Sin embargo, para poder simular el acoplamiento entre módulos se implementó un mecanismo de conexión "virtual" que simplemente acopla dos módulos cuando se cumplen las condiciones necesarias para que esto sea factible en la realidad, sin necesidad de simular el proceso concreto de conexión.

Por otro lado, dicha librería, tampoco dispone de las primitivas necesarias para describir cuerpos como los bloques que conforman los módulos. Debido a esto, fue necesario simular dichos bloques como si fueran cubos. Este hecho

produce que la simulación de los módulos no sea completamente realista. Aún así, la aproximación utilizada es suficiente para obtener resultados indicativos del comportamiento real del robot.

El hecho de tener que implementar los bloques como cubos produce que para poder rotar los bloques que componen un módulo, haya sido necesario deshabilitar la detección de colisiones entre los mismos. En la figura 3.7 se puede apreciar cómo los cubos que representan los bloques colisionan al ser rotados. De no deshabilitar la detección de colisiones entre dichos cuerpos, no sería posible rotar los mismos.

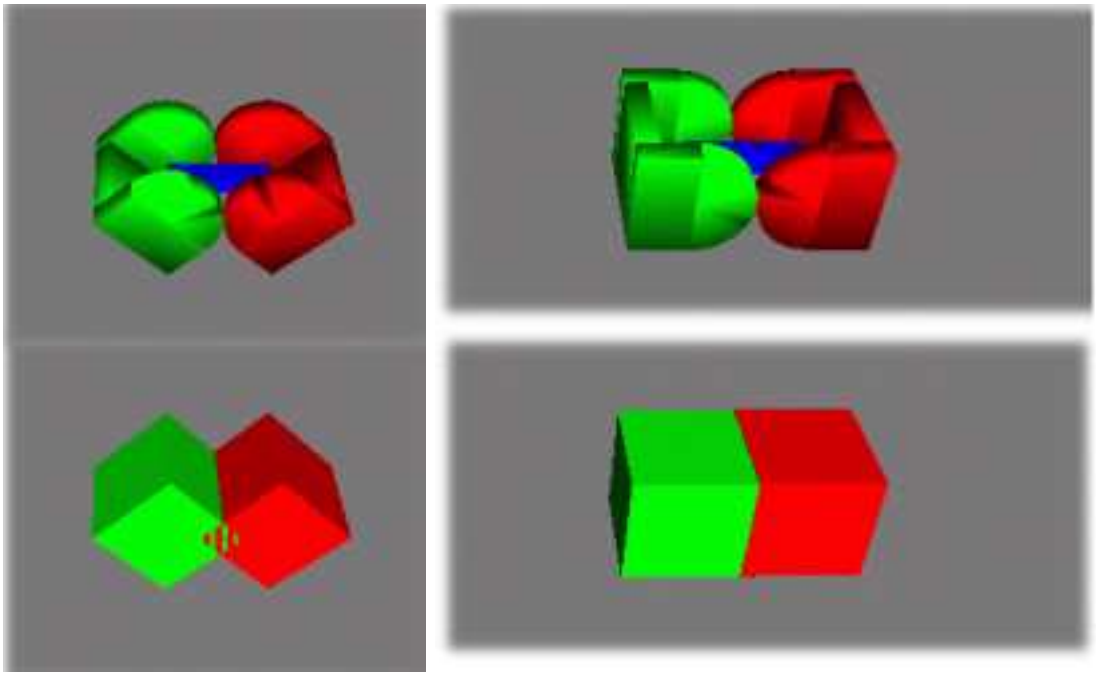

Figura 3.7: Visualización de la estructura real de un módulo (arriba) y de la estructura simulada (abajo).

Al deshabilitar la detección de colisiones fue necesario implementar otro método para limitar el rango de rotación de los bloques (de haber sido posible implementar los cuerpos según sus formas reales, el mismo mecanismo de detección de colisiones hubiera alcanzado para limitar el rango de rotación a 180 grados). Para tal fin, fue necesario restringir los ángulos impuestos sobre los motores en el momento de especificarlos. Es decir, al momento de determinar el nuevo ángulo deseado para un motor, se limitaba el mismo de forma que si $|\alpha| > 90$, entonces $\alpha = signo(\alpha) \times 90$.

3.4. Aprendizaje

Para lograr el aprendizaje de una política de locomoción o reconfiguración adecuada, se utilizó la técnica de Aprendizaje por Refuerzo, implementada mediante el algoritmo denominado Q-Learning.

3.4.1. Aprendizaje por Refuerzo

El Aprendizaje por Refuerzo abarca una clase de problemas dentro del espectro del Aprendizaje Automático en los que un agente debe explorar un entorno, dentro del cuál es capaz de percibir su propio estado, y realizar acciones que le permitan modificarlo.

Normalmente, es posible formular el entorno mediante un Proceso de Decisión de Markov (MDP) de estados finitos, y los algoritmos de Aprendizaje por Refuerzo utilizados en este contexto, están fuertemente relacionados con las técnicas de programación dinámica.

Elementos del Aprendizaje por Refuerzo

Además del agente y del entorno, es posible identificar tres elementos principales en un sistema de Aprendizaje por Refuerzo: una *política*, una *función de refuerzo*, y una *función de valor*.

La *política* define el comportamiento del agente para cada instante de tiempo (básicamente se trata de una relación entre los estados percibidos por el agente, y las acciones posibles de realizar en cada uno de ellos). Para representar una política es posible utilizar desde simples tablas de búsqueda, hasta procesos complejos que involucren gran cantidad de cómputo.

La *función de refuerzo* define el objetivo en un problema de Aprendizaje por Refuerzo. Se trata de una función que asigna un valor numérico a cada posible tupla sas', indicando la conveniencia intrínseca de llevar a cabo una acción a partiendo de un estado s y llegando a un estado s'. El objetivo de un agente de Aprendizaje por Refuerzo consiste en maximizar la suma de los refuerzos obtenida a largo plazo. La función de refuerzo no debe ser alterable por el agente, dado que representa las características intrínsecas del problema a resolver. Sin embargo, es posible utilizarla para modificar la política utilizada: en el caso de que una acción produzca una recompensa baja, o negativa, es posible modificar la política de forma de no volver a elegir dicha acción en el futuro.

Mientras que la función de refuerzo indica qué acciones y estados son "buenos" en el corto plazo, la *función de valor* especifica los estados óptimos

a largo plazo. El *valor* de un estado, es la recompensa total que un agente puede esperar acumular partiendo desde dicho estado.

Formalización del problema de Aprendizaje por Refuerzo

Dentro del paradigma del Aprendizaje por Refuerzo, el agente interactúa con el entorno en intervalos discretos de tiempo $t = 0, 1, 2, 3, \ldots$. En cada instante de tiempo t, el agente recibe una representación del estado del entorno $s_t \in S$, donde S es el conjunto de posibles estados, y selecciona una acción $a \in A(s_t)$, donde $A(s_t)$ representa el conjunto de acciones posibles de realizar en el estado s_t. En el siguiente instante de tiempo, como parte de las consecuencias de la acción realizada, el agente recibe una recompensa $r_{t+1} \in \mathbb{R}$, y percibe un nuevo estado s_{t+1}. Esta interacción se esquematiza en la figura 3.8.

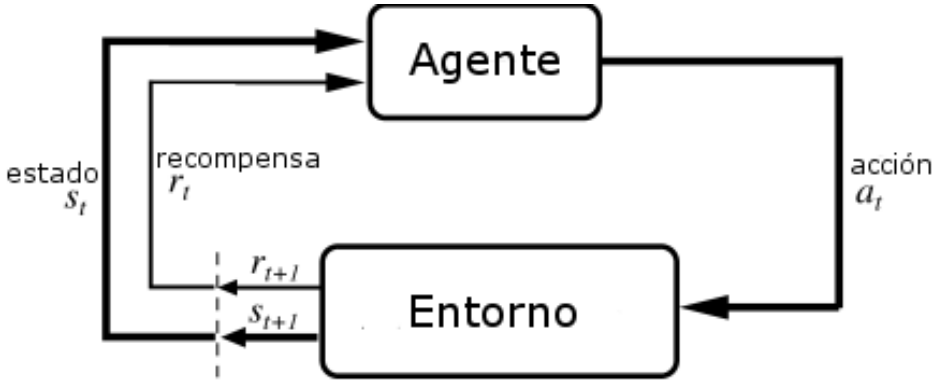

Figura 3.8: Diagrama de la interacción entre el agente y el entorno, en un problema de Aprendizaje por Refuerzo.

Para determinar las acciones a realizar, el agente utiliza una *política*, que relaciona cada estado con las mejores acciones posibles de realizar en dicho estado. Los algoritmos de Aprendizaje por Refuerzo permiten obtener la política óptima, a fin de maximizar las recompensas obtenidas.

Algoritmos de Aprendizaje por Refuerzo

La técnica de Aprendizaje por Refuerzo se diferencia de las técnicas de aprendizaje supervisado por el hecho de que en las primeras nunca se indican las soluciones "correctas", ni se corrigen explícitamente las acciones "erróneas". El foco central de esta técnica se localiza en el desempeño "on-line", que involucra encontrar un balance entre exploración (intentar acciones aún no realizadas, o poco utilizadas) y explotación (del conocimiento adquirido). La

dicotomía entre exploración y explotación ha sido estudiada ampliamente en este campo, mediante problemas como el del *bandido de n-brazos* [Sut98], entre otros.

Por lo tanto, el Aprendizaje por Refuerzo está particularmente bien adaptado a problemas en los que es necesario encontrar un balance entre recompensas inmediatas y recompensas a largo plazo. Esta técnica ha sido aplicada con éxito para la resolución de varios problemas, incluyendo control de robots, programación de elevadores [Cri98], telecomunicaciones [Kum98], backgammon [Tes95] y ajedrez [Bax97].

Una vez definida la función de recompensa, es necesario determinar el algoritmo utilizado para encontrar la política que maximice la misma. Existen principalmente dos enfoques para ello: el de la función de valor, y el método directo.

El método directo involucra dos pasos:

a) Para cada política posible, recolectar las recompensas obtenidas al utilizar la misma

b) Elegir la política que provea la mayor recompensa esperada

Este enfoque tiene básicamente dos problemas. Por un lado, la cantidad de políticas posibles puede ser muy grande, o incluso, infinitas. Por otro lado, las recompensas podrían ser estocásticas, en cuyo caso sería necesario recolectar una cantidad muy grande de muestras para estimar con precisión el valor esperado para cada política. A pesar de ello, este enfoque ha sido utilizado como base para los algoritmos habituales en el campo de la robótica evolutiva [Nol98].

Si se asume cierta estructura en el problema que se intenta resolver, algunos de estos inconvenientes pueden ser salvados. El enfoque de la función de valor se basa en que los retornos obtenidos por una política pueden influenciar las estimaciones hechas para otra. En este enfoque se trata de estimar el valor esperado al utilizar una política π, partiendo de un estado s

$$V^{\pi}(s) = E_{\pi}\left\{R_t | s_t = s\right\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s\right\}$$

o bien, estimar el valor esperado al tomar una acción a en un estado s y luego continuar según la política π

$$Q^{\pi}(s,a) = E_{\pi}\left\{R_t | s_t = s, a_t = a\right\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s, a_t = a\right\}.$$

Dada una función Q para la política óptima, es posible seleccionar las acciones óptimas, simplemente eligiendo aquellas acciones que presenten el máximo valor para cada estado s. A fin de realizar lo mismo mediante la función V, es necesario disponer de un modelo del entorno, en la forma de las probabilidades $P(s'|s,a)$, que permite calcular Q según la ecuación

$$Q(s,a) = \sum_{s'} V(s')P(s'|s,a) \tag{3.1}$$

o bien, se puede utilizar alguno de los métodos denominados "Actor-Crítico" [Sut98], en los que el modelo es separado en dos partes: el crítico, que mantiene información sobre el valor estimado de la función V, y el actor, que es el responsable de elegir las acciones apropiadas para cada estado.

Expandiendo la ecuación que define la función V^π, se obtiene:

$$\begin{aligned} V^\pi(s) &= E_\pi\{R_t|s_t = s\} \\ &= E_\pi\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}\middle| s_t = s\right\} \\ &= E_\pi\left\{r_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^k r_{t+k+2}\middle| s_t = s\right\} \\ &= E_\pi\{r_{t+1} + \gamma V^\pi(s_{t+1})|s_t = s\} \end{aligned}$$

Dada una política determinada π, estimar $E_\pi\{R_t|s_t = s\}$ para $\gamma = 0$ resulta trivial, ya que solamente es necesario promediar las recompensas inmediatas. La forma más sencilla de realizar esto para $\gamma > 0$ es promediar la recompensa total, luego de cada estado. Sin embargo, este tipo de muestreo al estilo de los métodos Monte Carlo, requiere que el MDP asociado al entorno tenga estados terminales.

Por lo tanto, realizar esta estimación para $\gamma > 0$ en el caso general no resulta obvio. Sin embargo, es realmente sencillo, si se advierte que la esperanza de R forma una ecuación recursiva de Bellman:

$$E_\pi\{R_t|s_t\} = r_t + \gamma E_\pi\{R_{t+1}|s_{t+1}\} \tag{3.2}$$

Remplazando dichas esperanzas por $V^\pi(s_t)$ y $V^\pi(s_{t+1})$, es posible derivar el algoritmo de aprendizaje por diferencia temporal TD(0), cuya regla de actualización se describe por

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)].$$

En el caso más simple, el conjunto de estados y de acciones son ambos discretos, y es posible mantener un estimativo para cada par estado-acción. Otros métodos similares son SARSA y Q-Learning.

Los métodos mencionados convergen todos al estimativo correcto para una política dada, pero además pueden ser utilizados para determinar la política óptima. Habitualmente esto se realiza siguiendo una política π que es derivada de los valores estimados en cada momento (por ejemplo, eligiendo la mayor parte del tiempo, las acciones asociadas a un valor máximo de Q, pero ocasionalmente tomando acciones al azar, a fin de explorar otras acciones posibles).

Q-Learning

Q-Learning [Wat89], según se vio en el punto anterior, es un algoritmo de Aprendizaje por Refuerzo que utiliza la información asociada a un estado s' (i.e. $Q(s', a')$) para corregir la estimación de $Q(s, a)$, al tomar una acción a estando en un estado s, siguiendo una política determinada π. Un beneficio importante de este algoritmo reside en su capacidad de poder comparar la esperanza del valor de recompensa para cada una de las acciones posibles, sin la necesidad de disponer de un modelo del entorno (ver ecuación 3.1).

El mecanismo central del algoritmo se basa en la simplicidad de la regla de actualización de los estimativos. Para cada estado $s \in S$, y para cada acción $a \in A$, se calcula el valor esperado de recompensa de acuerdo a la ecuación

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \tag{3.3}$$

en donde, r_{t+1} es la recompensa obtenida luego de realizar la acción a_t, estando en el estado s_t; α es la taza de convergencia, tal que $0 < \alpha < 1$ y γ es la taza de descuento, tal que $0 < \gamma < 1$.

La función de estado-acción Q aprendida mediante este método, aproxima directamente a Q^*, la función de estado-acción óptima, independientemente de la política utilizada. Este hecho simplifica radicalmente el análisis del algoritmo. La política sigue siendo utilizada para determinar las acciones a tomar en cada estado, sin embargo, solamente es necesario un requisito para garantizar la correcta convergencia: todos los pares estado-acción deben continuar siendo actualizados. Esto quiere decir que, siempre que se realice un mínimo de exploración en la política utilizada, el algoritmo garantiza que la función Q convergerá hacia Q^*.

Este algoritmo se describe según el siguiente esquema:

1. Inicializar $Q(s, a)$ de manera arbitraria
2. **Repetir** para cada episodio
   a) Inicializar s
   b) **Repetir** para cada paso (de un episodio)
      1) Elegir a para s, utilizando una política derivada de Q (por ejemplo, ϵ-greedy)
      2) Realizar la acción a, y obtener la recompensa r y el nuevo estado s'
      3) $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma max_{a'} Q(s', a') - Q(s, a)]$
      4) $s \leftarrow s'$
   c) **hasta que** s sea un estado terminal

Para la implementación de este algoritmo se definieron algunas clases: *Policy*, *ActionValueFunction* y *RLBrain*. *RLBrain* encapsula todo el comportamiento del aprendizaje. En esta clase se define, mediante la política utilizada (implementada en la clase *Policy*), la mejor acción posible a dado un estado s. Para determinar esto, la política dispone de la información provista por la función Q, implementada en la clase *ActionValueFunction*. Es decir, *Policy* implementa una política derivada de Q. Una vez obtenida la acción a y realizada la misma, en *RLBrain* se obtiene la recompensa asociada con dicha acción (y estado), y se invoca un método que actualiza la información de la tabla Q.

La condición de terminación, determinada como "s sea un estado terminal" en el esquema anterior se define en *RLBrain* (o cualquiera de sus clases derivadas), permitiendo que dicha condición dependa del experimento realizado (esto también es válido para la función de recompensa, que depende de la clase derivada de *RLBrain* utilizada en el experimento en cuestión).

Capítulo 4

Experimentación

En el presente capítulo se analiza el impacto de la utilización de la técnica de Aprendizaje por Refuerzo para resolver las tareas de locomoción y reconfiguración en robots M-TRAN.

Para tal fin, los experimentos se realizan de forma episódica. En todos los casos en los que no se indique lo contrario, se considera un episodio como la ejecución de una simulación, según la definición dada a continuación:

Simulación La ejecución del programa simulador hasta cumplirse una condición de finalización. Las condiciones de finalización varían según el tipo de *Cerebro* (ver 3.3.1) utilizado. Algunas condiciones de finalización posibles son: la cantidad de acciones realizadas; la evaluación de una condición, como por ejemplo, "distancia a la fuente emisora menor que un valor crítico".

4.1. Locomoción

Se lleva a cabo un experimento, con el objetivo de:

1. validar el simulador desarrollado, mediante la aplicación de un caso de estudio real

2. consolidar la metodología y conocimientos necesarios para realizar una experiencia de aprendizaje automático en Robots Modulares Autoreconfigurables utilizando dicho simulador

4.1.1. Experimento 1: Locomoción en un robot de configuración mínima

Se intenta resolver una tarea de locomoción sencilla, en un ambiente simple. El entorno simulado consta de cuatro paredes y, naturalmente, un suelo. El mismo es uniformemente llano, y el recinto se encuentra libre de obstáculos (a excepción de las paredes). El robot utilizado para este experimento consiste de un único módulo.

Para la tarea del aprendizaje se utiliza el algoritmo *Q-Learning*, según fue presentado en la sección 3.4.1.

Análisis del problema

Dado que el experimento se realiza en un ambiente simulado, pero pretendiendo mantener la mayor correlación posible con la realidad, es necesario evitar ciertas "bondades" presentes en este tipo de ambientes; una de ellas es la capacidad de obtener la posición del robot de forma absoluta. Dado que un robot M-TRAN no dispone de dicha capacidad, resulta necesario encontrar una forma alternativa para determinar el desplazamiento del robot.

Según lo visto en la sección 3.4.1, el algoritmo utilizado involucra los conceptos:

- Estado
- Acción
- Recompensa

En el caso de *Estado* y *Acción*, es necesario tener en cuenta que se busca obtener una definición de ambos conceptos que mantenga la cantidad de posibilidades "estado-acción" acotada. Esto se debe a que la capacidad y velocidad de aprendizaje del robot dependen directamente del tamaño de los conjuntos determinados por las variables utilizadas.

Estado

En este punto es necesario diferenciar entre la noción de estado en tanto a las capacidades de percepción del robot y en tanto al conjunto de valores necesarios para el aprendizaje. Por ejemplo, aunque el robot sea capaz de determinar la fecha, probablemente este dato le sea de poca utilidad para aprender a desplazarse. Entonces, la fecha, aunque forme parte del estado sensorial del robot (porque el mismo es capaz de percibir este estímulo), no

pertenece al conjunto de variables que describen el estado utilizado para el aprendizaje, puesto que no contribuye al objetivo del mismo.

En general, el estado para el aprendizaje es un subconjunto del estado sensorial del robot.

Acción

En función del objetivo planteado, las acciones que debe considerar el algoritmo de aprendizaje están relacionadas estrictamente con el desplazamiento del robot en el entorno de experimentación. Dado que el robot está formado por un único módulo, las acciones disponibles están restringidas a las que puede realizar un módulo; estas acciones se limitan al movimiento de los motores de los bloques que lo conforman.

Se pretende simular motores tipo servo, que son controlados indicando el ángulo al que se desea llegar. Por lo tanto, las acciones consisten básicamente de los ángulos que se desean imponer a los motores en cada momento. Para posibilitar la tarea de aprendizaje resulta óptimo disponer de la menor cantidad de ángulos posibles que permitan el desplazamiento del robot en el entorno de experimentación.

Recompensa

Según el objetivo planteado, la función de recompensa debe favorecer los casos en los que el robot se desplace en la dirección deseada, y desfavorecer los casos en los que el robot no avance o se aleje. Es importante tratar de evitar situaciones en las que el robot pueda quedar inmovilizado.

Solución propuesta

Según visto anteriormente, es necesario evitar el posicionamiento absoluto. Por tal motivo, es posible introducir una fuente de luz o radio en el ambiente, y dotar al robot con sensores que le permitan establecer si se acerca o se aleja de la fuente emisora, sin necesidad de conocer su posición absoluta.

Dadas las limitaciones de movilidad que tiene un robot de este tipo (el hecho de estar formado por un único módulo lo limita a un desplazamiento unidimensional), el mismo deberá estar alineado inicialmente con la fuente emisora utilizada, a fin de maximizar las posibilidades de aprendizaje.

En estas circunstancias, el objetivo del robot consiste en desplazarse tratando de minimizar la distancia entre sí mismo y la fuente emisora.

Para poder implementar esta solución, resulta necesario definir:

- ¿Dónde ubicar los sensores?
- ¿Cuántos sensores utilizar por módulo?
- ¿Cómo codificar el valor de los sensores?

Luego de analizar las posibilidades, se decidió implementar la solución que se describe a continuación.

Sensores (estímulo)

Para determinar el avance del robot, se utilizan sensores posicionados en el centro de masa de cada bloque. Por simplicidad, los sensores son omnidireccionales (miden la distancia en línea recta hasta la fuente emisora).

Los sensores tienen un alcance máximo definido, y dividen el espectro de medición en una cantidad finita de intervalos idénticos y equidistantes. El valor del sensor se determina en función del intervalo registrado por el mismo. La función utilizada para calcular este valor está dada por la ecuación 4.1.

$$v = discretizar(|p_s - p_f|) \tag{4.1}$$

con

$$\begin{aligned} discretizar(x) &= \begin{cases} 0 & \text{si } x \geq rango_s \\ \lfloor (1 - \frac{x}{rango_s}) \times niveles_s \rfloor & \text{sino} \end{cases} , \\ p_s &= \text{posición del sensor} , \\ p_f &= \text{posición de la fuente emisora} , \\ rango_s &= \text{distancia máxima de sensado} , \\ niveles_s &= \text{cantidad de niveles de discretización del sensor} . \end{aligned}$$

Cabe aclarar que si bien el cálculo del valor del sensor utiliza las posiciones del robot y de la fuente emisora, este detalle está encapsulado dentro del sensor y por lo tanto constituye un "detalle de implementación" del mismo. El robot nunca dispone de información acerca de su posición, sino que trabaja únicamente con el valor provisto por el sensor.

Robot (estado)

El estado del robot (en función de su capacidad de percepción) está conformado por 4 componentes. El robot es capaz de determinar:

- el ángulo en el que se encuentra cada uno de los bloques que lo conforman (información provista por los encoders de los motores)
- el valor percibido por cada uno de los sensores ubicados en los bloques que lo conforman

A su vez, el robot dispone de una serie de "sensores virtuales", mediante los cuales puede determinar:

- "pseudorotación" de cada módulo del robot
- dirección hacia la fuente emisora

Bajo "pseudorotación" se entiende un valor que se obtiene de la posición relativa entre los bloques que conforman un módulo, y que da una noción acerca de la rotación del módulo en el espacio.

La forma de calcular este valor está dada por la ecuación 4.2.

$$v = pseudorotación(v_{activo}, v_{pasivo}) \tag{4.2}$$

donde

$$\begin{aligned} v_{activo} &= \text{valor del sensor ubicado en el bloque activo ,} \\ v_{pasivo} &= \text{valor del sensor ubicado en el bloque pasivo ,} \\ pseudorotación(x,y) &= \begin{cases} -1 & \text{si } (x-y) > 0 \\ 1 & \text{si } (x-y) < 0 \\ 0 & \text{en otro caso} \end{cases} . \end{aligned}$$

Por otro lado, cada módulo puede determinar si la fuente emisora se encuentra más cerca de su bloque activo o pasivo, lo que determina la dirección hacia la que el mismo se deberá mover.

Para calcular el valor de este sensor, se utiliza la ecuación 4.3.

$$v = normalizar((d_{activo} + d_{pasivo}) \times eje_s) \tag{4.3}$$

donde

$$\begin{aligned} d_{activo} &= \text{dirección del bloque activo ,} \\ d_{pasivo} &= \text{dirección del bloque pasivo ,} \\ eje_s &= \text{alineación inicial de los bloques ,} \\ normalizar(x) &= \begin{cases} 1 & \text{si } x > 1 \\ -1 & \text{si } x < -1 \\ 0 & \text{en otro caso} \end{cases} . \end{aligned}$$

Estos valores se pueden obtener según las ecuaciones indicadas a continuación:

$$
\begin{aligned}
d_{bloque} &= map(dirección, p_{bloque} - p_f) \ , \\
map(f, v) &= concatenar(f(cabeza(v)), map(f, resto(v))) \ , \\
dirección(x) &= \begin{cases} -1 & \text{si } x > 0 \\ 1 & \text{si } x < 0 \\ 0 & \text{en otro caso} \end{cases} \ , \\
p_{bloque} &= \text{posición del bloque} \ , \\
p_f &= \text{posición de la fuente emisora} \ , \\
eje_s &= normalizar(p_{i_{pasivo}} - p_{i_{activo}}) \ , \\
p_{i_{bloque}} &= \text{posición inicial del bloque} \ .
\end{aligned}
$$

donde *concatenar*, *cabeza* y *resto* son funciones primitivas del manejo de listas.

Nuevamente, en este caso, si bien el cálculo del valor del sensor involucra las posiciones del robot y de la fuente emisora, el robot no dispone de información acerca de las posiciones, sino que trabaja con el valor indicado por este sensor. De encontrar una manera alternativa de calcular el valor de este sensor, será posible remplazar el cálculo sin afectar el comportamiento del robot.

Aprendizaje (estado)

Como se vio anteriormente, el estado del robot y el estado para el aprendizaje (estado utilizado por la política para determinar la acción a seguir) no son necesariamente iguales. Para este experimento es necesario hacer la distinción, y establecer el estado para el aprendizaje como:

- $e_{robot} = e_{módulo}$
- $e_{módulo} = [\psi, \delta, \alpha, \beta]$
- $\psi =$ "pseudorotación"

  Valores posibles: $\begin{cases} -1 & \text{bloque activo más cerca de la fuente emisora} \\ 0 & \text{bloques equidistantes a la fuente emisora} \\ 1 & \text{bloque pasivo más cerca de la fuente emisora} \end{cases}$

- δ = dirección hacia la fuente emisora

  Valores posibles: $\begin{cases} -1 & \text{hacia la "derecha" (coordenadas positivas)} \\ 0 & \text{paralelo a la fuente emisora} \\ 1 & \text{hacia la "izquierda" (coordenadas negativas)} \end{cases}$

- α = ángulo del bloque activo

  Valores posibles: -90, 0, +90

- β = ángulo del bloque pasivo

  Valores posibles: -90, 0, +90

Esta definición de estado permite diferenciar cada configuración posible del robot, mediante los valores de los ángulos de cada bloque. A su vez, el valor ψ provee indicios acerca de la rotación del módulo en el espacio, lo que permite determinar la dirección asociada a "ir hacia adelante", mientras que δ permite decidir si se debe ir "hacia adelante" o "hacia atrás".

En la figura 4.1 se muestran ejemplos de distintos estados, utilizando gráficos para su mejor comprensión.

Cabe destacar que el valor de los sensores no forma parte del estado para el aprendizaje. Esto se debe a que la posición del robot (en función de la distancia hasta la fuente emisora) no influye en la elección de la acción a tomar. La posición del robot (derivada a partir de los sensores) se utiliza para determinar el avance del robot y calcular la recompensa obtenida para cada acción realizada.

Aprendizaje (acciones)

Las acciones que puede realizar el robot se definen como una lista de movimientos que serán ejecutados simultáneamente. Cada movimiento se corresponde con el accionar de un motor. Una acción puede involucrar uno o dos movimientos (dado que al estar conformado el robot por un único módulo, la cantidad total de motores es 2).

En función de esta definición, una acción se describe formalmente cómo:

- $a = m_1 \vee a = m_1, m_2$
- $m_i = módulo|bloque|ángulo$
- *módulo* es el número de módulo al que pertenece el motor a mover (comenzando desde 0). En este caso, al haber un único módulo, este valor será siempre 0. Se define de forma genérica, puesto que experimentos posteriores (que involucran más de un módulo) se utiliza esta misma definición de estado.

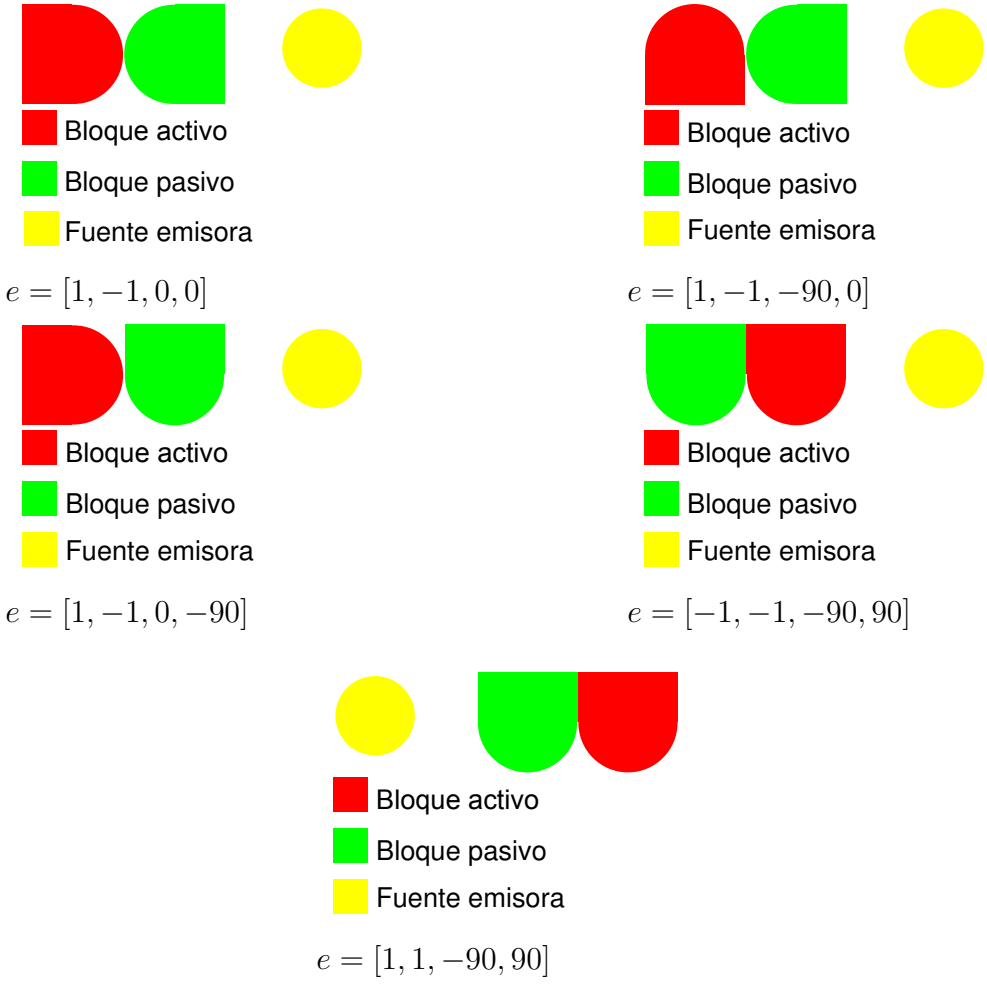

Figura 4.1: Estados posibles de un robot

- *bloque* es A si el motor pertenece al bloque activo y P si pertenece al pasivo.
- *ángulo* es -90 o +90. En la definición de ángulo para las acciones, el mismo es considerado como relativo, a diferencia del ángulo que describe el estado del módulo, que es considerado como un valor absoluto. Por ello, en esta ocasión, 0 no es un valor válido.

Dado que las acciones que puede realizar un robot formado por un único módulo son pocas, pueden describirse explícitamente. En función de los valores indicados anteriormente, las acciones que puede realizar un módulo son:

- 0|A|-90
- 0|A|+90
- 0|P|-90
- 0|P|+90
- 0|A|-90,0|P|-90
- 0|A|-90,0|P|+90
- 0|A|+90,0|P|-90
- 0|A|+90,0|P|+90

La elección de esta definición para las acciones se debe a que corresponde a los datos mínimos indispensables para poder modificar la configuración del robot (que es el método que tiene éste para desplazarse) y por ende su estado. La elección de los ángulos utilizados se debe a que estos ángulos (-90, +90) son suficientes para permitirle al robot realizar locomoción (aunque sea rudimentaria), y por lo tanto, agregar mayor cantidad de ángulos incrementaría innecesariamente el cardinal del espacio de estado-acción.

Aprendizaje (recompensa)

La función de recompensa utilizada se define como:

$$recompensa(s_t, a_t) = \begin{cases} -5 & \text{si el robot no cambió de estado}^6 \\ distancia(s_t, a_t) & \text{en otro caso} \end{cases}$$

donde $distancia(s_t, a_t)$ es la distancia recorrida durante la última acción. Este valor depende del desplazamiento que haya realizado el robot durante la última acción, pero puede considerarse que normalmente se encuentra en el rango $[-1, 1]$.

La penalización en caso de no cambiar de estado se utiliza para desfavorecer acciones que potencialmente pueden llevar al robot hacia estados de inmovilización.

Utilizando esta función de recompensa, las acciones que provocan un avance en la dirección de la fuente emisora se ven recompensadas en forma proporcional al avance realizado y las acciones que provocan un retroceso, se ven igualmente penalizadas.

Resultados

Inicialmente se analizó el impacto de las variables involucradas en el algoritmo de aprendizaje. Se detectaron tres variables a ser estudiadas:

- *alpha* y *gamma* son parámetros del algoritmo *Q-Learning* que definen el impacto de las acciones realizadas sobre las futuras decisiones a tomar
- *épsilon* es un parámetro de la política utilizada para determinar el curso de acción. En una política ϵ-*greedy*, *épsilon* define la proporción de acciones tomadas al azar.

Se realizaron tres series de 90 simulaciones variando los valores de *alpha*, *épsilon* y *gamma*, para determinar cuál de ellos generaba el mayor impacto en el resultado obtenido.

Las variables observadas en cada simulación fueron:

- **distancia avanzada**

  Distancia que recorrió el robot en dirección hacia la fuente emisora.

- **distancia recorrida**

  Distancia total recorrida (se cuentan todos los movimientos del robot, incluso los que lo hacen retroceder).

[6] en este caso el estado analizado es el estado real del robot, no el estado utilizado para el aprendizaje. Para determinar este hecho se compara el estado del robot antes de ejecutar la acción y luego de realizar la misma.

- **acciones realizadas**

  Cantidad de acciones que necesitó realizar el robot para alcanzar el objetivo (con un máximo de 200 acciones posibles).

Para minimizar el impacto de la varianza de los resultados, se realizaron 10 ejecuciones con cada conjunto de parámetros, y se evaluaron los resultados en función del promedio de los valores obtenidos.

Cabe aclarar que en estas experimentaciones no se buscó encontrar los valores óptimos de las variables sino presentar y analizar un panorama acerca de los factores que influencian el problema tratado. Debido a esto, el análisis realizado no es exhaustivo y sólo fue utilizado para descubrir aspectos interesantes acerca de la naturaleza del problema.

Examinando los resultados obtenidos (que se incluyen en el Apéndice B), se puede observar que el aprendizaje no fue exitoso en los casos en que $\epsilon \geq 0{,}6$, mientras que los casos en los que el aprendizaje fue veloz ocurrieron cuando dicha variable tomaba valores inferiores a 0,3. Estos resultados, además, sugieren que los valores convenientes para α y γ está comprendidos entre 0,6 y 0,9.

A continuación se realizaron otras tres series de 90 simulaciones para comparar las políticas obtenidas en cada caso. Estas simulaciones se realizaron simplemente evaluando las políticas obtenidas durante el aprendizaje.

Además de los datos analizados durante el aprendizaje, para comparar las distintas políticas, se definió una medida de *performance*. Este valor se calcula en base a la ecuación 4.4.

$$performance = \frac{d_{avanzada}}{d_{recorrida}} \times \frac{k}{n} \tag{4.4}$$

con

$$\begin{aligned} d_{avanzada} &= \text{distancia avanzada ,} \\ d_{recorrida} &= \text{distancia recorrida total ,} \\ k &= \text{cantidad de acciones requeridas}^{7} \text{ ,} \\ n &= \text{cantidad de acciones realizadas .} \end{aligned}$$

Analizando esta ecuación, se puede ver que el cociente

[7] se considera este valor como la mínima cantidad de acciones requeridas para alcanzar el objetivo, es decir, la cantidad de acciones realizadas por la política óptima

$$\frac{d_{avanzada}}{d_{recorrida}}$$

da una métrica acerca de la velocidad de avance del robot (es decir, si las acciones que realizó resultaban mayoritariamente en un avance hacia el objetivo), mientras que el cociente

$$\frac{k}{n}$$

indica la velocidad promedio de avance que generan las decisiones tomadas por el robot.

Por lo tanto, las mejores políticas deben reflejar un valor alto de performance (cercano a 1). De ello se desprende que para encontrar políticas óptimas es necesario maximizar el valor de la variable performance.

Los resultados obtenidos se pueden observar en la tabla B (que se encuentra en el Apéndice B).

De estos resultados se pueden extraer algunos gráficos (figuras 4.2, 4.3, 4.4) que explican la varianza de la performance en función de cada parámetro.

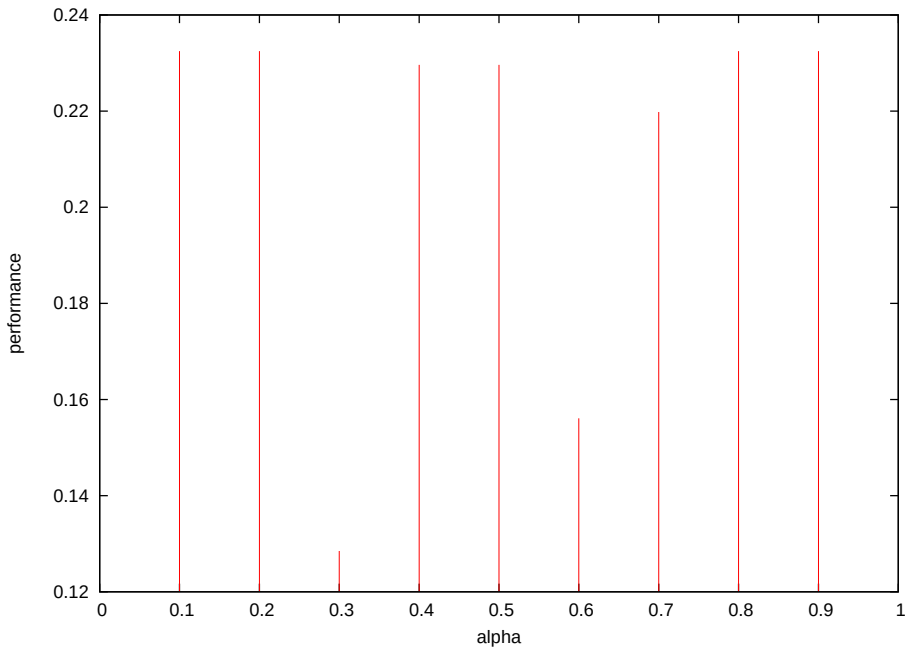

Figura 4.2: Performance de las políticas obtenidas en función del parámetro α, para $\epsilon = \gamma = 0{,}5$.

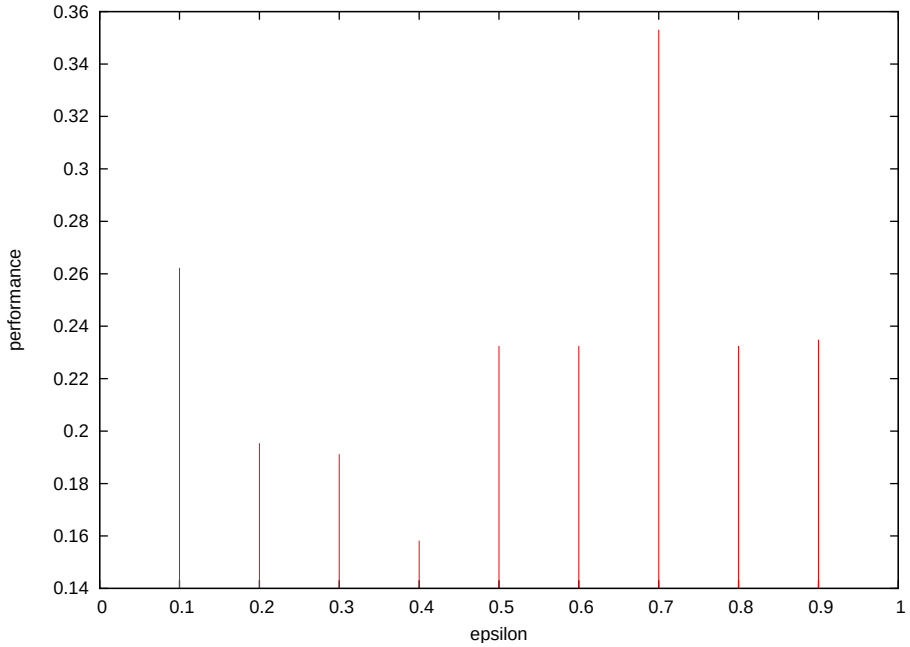

Figura 4.3: Performance de las políticas obtenidas en función del parámetro ϵ, para $\alpha = \gamma = 0{,}5$.

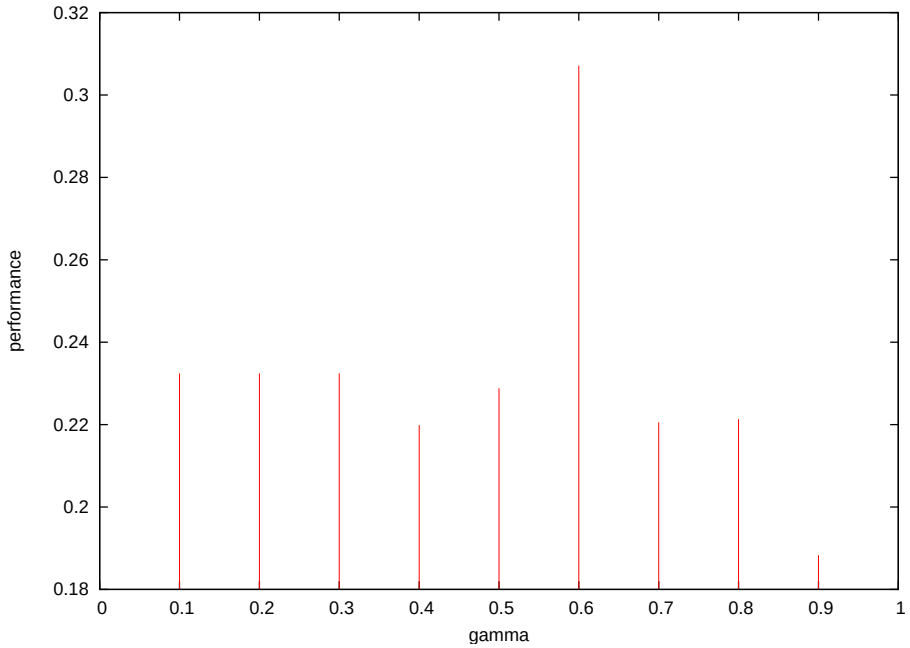

Figura 4.4: Performance de las políticas obtenidas en función del parámetro γ, para $\alpha = \epsilon = 0{,}5$.

Finalmente, en la tabla 4.1.1, se comparan estos resultados con los valores obtenidos al ejecutar una política arbitraria (manualmente generada) y una política completamente al azar.

Política	Distancia Avanzada	Distancia Recorrida	Acciones Realizadas	Performance
Aprendizaje[a]	29.50	62.00	56	0.35310
Aprendizaje[b]	22.50	65.50	148	0.12850
Al azar	3.45	153.30	201	0.00391
Arbitraria	29.50	56.70	56	0.35599

[a] Se indican los valores de la mejor política aprendida
[b] Se indican los valores de la peor política aprendida

Tabla 4.1: Comparación de performance de las políticas aprendidas respecto de las políticas de control

En dicha tabla se puede apreciar que todas las políticas obtenidas fueron mejores que la política al azar, y si bien quedaron por debajo de la política construida manualmente, esto se debe a que por la simplicidad (y por ende su tamaño reducido) del espacio de estados y acciones, es fácil determinar una política óptima manualmente. Sin embargo, en el caso de que el robot consistiera de una mayor cantidad de módulos y/o pudiera realizar una mayor cantidad de acciones, encontrar una política óptima de forma manual no sería tan sencillo. En dicha circunstancia, contar con un algoritmo de aprendizaje que provea políticas cercanas a óptimas sería altamente deseable.

En los resultados obtenidos, es posible observar que los valores óptimos para α se encuentran en los extremos del espectro. En el caso de utilizar valores del rango inferior, se estaría minimizando el impacto de las acciones tomadas sobre el aprendizaje (llegando al extremo, $\alpha = 0$ equivaldría a no aprender), mientras que utilizar valores del rango superior implicaría basarse casi exclusivamente en la estimación del par estado-acción sucesor (i.e. $Q(s', a')$) para determinar la próxima acción; en algún sentido también equivaldría a no aprender, dado que un valor elevado para α induciría al robot a "olvidar lo aprendido en cada instante".

El valor sugerido por los resultados para γ es esperable, dado que representa un balance entre utilizar experiencia reciente y "olvidar" experiencia adquirida hace mayor tiempo (de no ser así, o bien no se tendría en cuenta la experiencia – γ muy pequeño –, o no se "olvidarían los errores" – γ muy alto).

El valor obtenido para el ϵ inicial sin embargo, es interesante. Los mejores resultados fueron obtenidos cuando ϵ presentaba un valor relativamente

alto ($\epsilon = 0{,}7$). Sin embargo, en muchos casos en los que ϵ tenía un valor en su rango superior ($\epsilon > 0{,}6$), el robot no lograba llegar hasta el objetivo durante su etapa de aprendizaje. Esto quiere decir que si bien las mejores políticas involucran un alto grado de exploración, lo cual tiene sentido, ya que es *necesario* cierto grado de exploración para poder aprender una política óptima, una proporción elevada de exploración también puede conducir a un desaprovechamiento de la experiencia adquirida, produciendo un desempeño subóptimo. El hecho de que las mejores políticas presentaran un valor elevado en esta variable, resulta del tamaño acotado del espacio de estados, que habilita que una alta exploración *inicial* (en las simulaciones realizadas, la exploración es fuerte al principio, y va decayendo a medida que pasa el tiempo) permita encontrar velozmente las acciones óptimas para cada estado.

Conclusiones

A la luz de los resultados obtenidos, se puede verificar que es *factible* realizar tareas de aprendizaje automático (en particular, utilizando la técnica de Aprendizaje por Refuerzo) sobre Robots Modulares Autoreconfigurables de tipo M-TRAN.

Queda visto que el simulador desarrollado cumple con los requisitos necesarios para poder realizar este tipo de experimentos y que así como en otros ejercicios de este tipo es necesario definir cuidadosamente los aspectos asociados al algoritmo de aprendizaje (estado, acción y función de recompensa).

Los resultados obtenidos son consistentes con los conocimientos existentes acerca del Aprendizaje por Refuerzo. Es decir, en función del problema tratado, los valores de las variables analizadas que producían los mejores resultados, son coherentes.

Finalmente, si bien la tarea de locomoción propuesta era sencilla, el éxito obtenido de las simulaciones realizadas permite sugerir que la utilidad de la técnica de Aprendizaje por Refuerzo para la tarea de locomoción sobre Robots Modulares Autoreconfigurables de tipo M-TRAN es real e importante.

4.2. Reconfiguración

En este caso se pretende verificar la capacidad de un robot autoreconfigurable de tipo M-TRAN para alcanzar comportamientos de reconfiguración mediante el Aprendizaje por Refuerzo.

A continuación se describen los experimentos realizados en el marco de este trabajo.

4.2.1. Experimento 2: Reconfiguración básica

En este experimento se trabaja con un robot compuesto por tres módulos. La configuración inicial del robot es del tipo lineal, es decir, los módulos se encuentran alineados según se muestra en la figura 4.5.

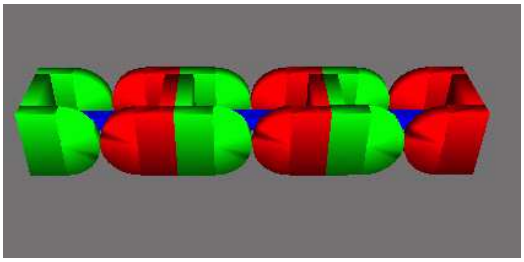

Figura 4.5: Configuración del tipo lineal para 3 módulos M-TRAN, correspondiente al estado inicial del experimento 2

El objetivo de este experimento consiste en que el robot adopte la configuración indicada por la figura 4.6.

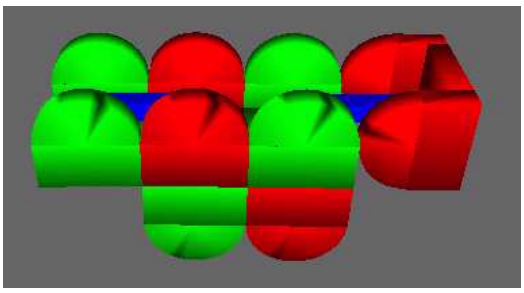

Figura 4.6: Configuración final (3 módulos) correspondiente al experimento 2

El estado del robot se representa por medio de una lista que indica el estado de cada módulo. A su vez, el estado de un módulo es representado por un par de ángulos, correspondientes a los motores presentes en el mismo, es decir

$$\begin{aligned} estado &= [estado_{\text{módulo}_1}, estado_{\text{módulo}_2}, estado_{\text{módulo}_3}] \\ estado_{\text{módulo}_i} &= \text{'}\alpha_i::\beta_i\text{'} \ . \end{aligned} \quad (4.5)$$

Según esta representación, los estados inicial y final del experimento son

$$\begin{aligned} estado_{inicial} &= [\text{'0.0::0.0'}, \text{'0.0::0.0'}, \text{'0.0::0.0'}] \\ estado_{final} &= [\text{'0.0::-90.0'}, \text{'-90.0::90.0'}, \text{'90.0::-90.0'}] \end{aligned}$$

Las acciones que puede realizar el robot consisten únicamente en la modificación de los ángulos de los distintos motores. Dichas acciones están definidas según

$$\begin{aligned} acción &= \text{'}módulo|bloque|ángulo\text{'} , \\ módulo &= 0 \vee 1 \vee 2 , \\ bloque &= A \vee P , \\ ángulo &= \text{-}90 \vee +90 . \end{aligned} \tag{4.6}$$

Las acciones definidas según esta ecuación serán denominadas "acciones simples", dado que cada acción permite modificar el ángulo de un único motor. En contraposición, en experimentos futuros se utilizarán acciones "complejas" que permiten modificar más de un motor simultáneamente.

La función de recompensa utilizada se define como:

$$recompensa = \begin{cases} 1 & \text{si se alcanzó el estado objetivo} \\ 0 & \text{en otro caso} \end{cases} \tag{4.7}$$

Resultados

Luego de realizar 13 episodios de aprendizaje, se obtuvo una política que cumplía con el objetivo propuesto. Según se puede ver en la figura 4.7, durante la fase de aprendizaje, la cantidad de acciones realizadas en cada episodio, antes de alcanzar el objetivo propuesto, tiende a disminuir, debido a que el robot utiliza la experiencia acumulada (es decir, aprende). Sin embargo, en la figura mencionada se puede observar que en ciertos episodios, el robot realiza una mayor cantidad de acciones que en episodios anteriores. Si bien esto pudiera parecer un indicio de que el robot no está utilizando la experiencia pasada, en realidad se debe a que la política de aprendizaje alterna períodos de exploración con períodos de explotación del conocimiento adquirido. Gracias a esta exploración el robot es capaz de determinar los estados que lo llevarán a alcanzar el objetivo. A su vez, este hecho le hace tener que realizar mayor cantidad de acciones a fin de determinar dichos estados.

Otro aspecto que se puede observar en esta figura es una recta denominada "Tendencia global", que se obtiene mediante una aproximación lineal de los datos, utilizando el método de los mínimos cuadrados.

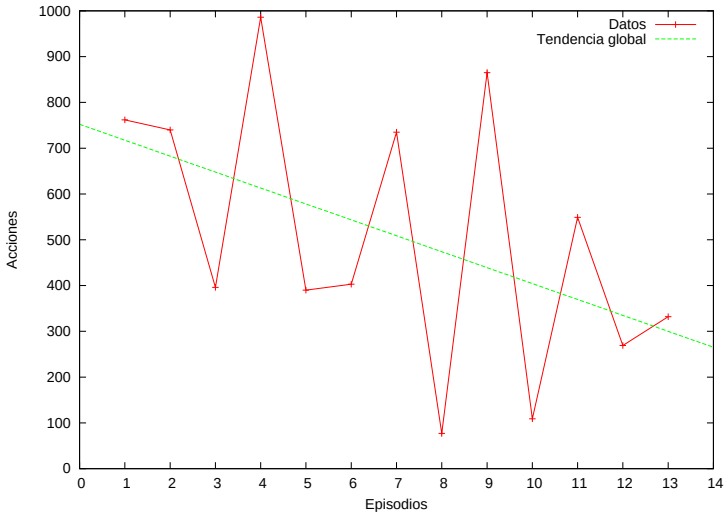

Figura 4.7: Cantidad de acciones realizadas antes de alcanzar el objetivo, durante la fase de aprendizaje de una política de reconfiguración, para el experimento 2

En la figura 4.8 se puede apreciar que utilizando la política obtenida se realizan siempre la misma cantidad de acciones. Esto indica que la política es estable, en el sentido de que en todas las instancias de validación, el robot se comporta de manera idéntica. Otro aspecto llamativo resulta de la diferencia en la cantidad de acciones realizadas durante la fase de aprendizaje y durante la evaluación de la política aprendida. Es notable que si bien es necesaria una cantidad importante de acciones durante el aprendizaje (entre 100 y 1000 acciones fueron necesarias durante los distintos episodios de aprendizaje), al utilizar la política aprendida se necesitan únicamente 5 acciones para alcanzar el objetivo. La magnitud de la diferencia entre dichos valores indica que luego de realizar suficiente cantidad de episodios de aprendizaje para lograr determinar una política estable, la misma resulta altamente eficiente.

Discusión

Según muestran los resultados obtenidos, el primer intento de aprender una política para resolver una tarea de reconfiguración fue exitoso. Si bien la tarea propuesta era sencilla (la configuración objetivo era fácilmente alcanzable), este primer éxito incentiva la realización de experimentos más complejos.

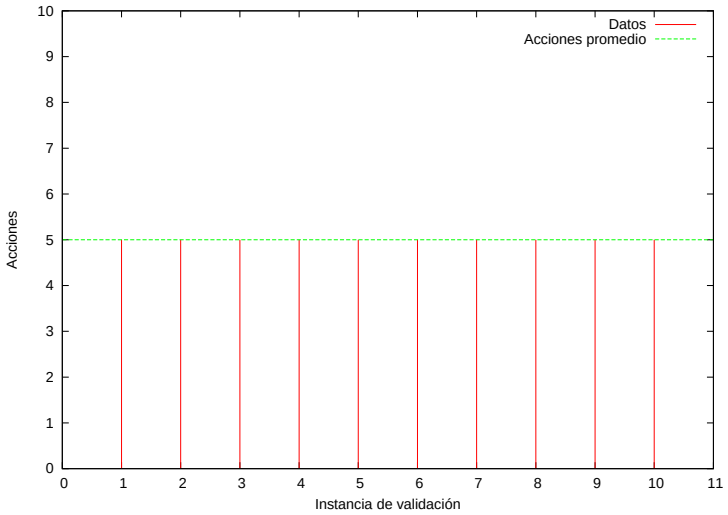

Figura 4.8: Cantidad de acciones realizadas antes de alcanzar el objetivo, utilizando la política aprendida, para el experimento 2

4.2.2. Experimento 3: De estructura lineal a circular, acciones simples

Una vez verificado que una tarea de reconfiguración sencilla era posible de resolver mediante Aprendizaje por Refuerzo, se modificó la configuración objetivo con el fin de verificar esta técnica utilizando una configuración final diferente. En este caso, partiendo de la misma configuración inicial que en el experimento 2, se intentó llegar a una configuración "circular", según se muestra en la figura 4.9.

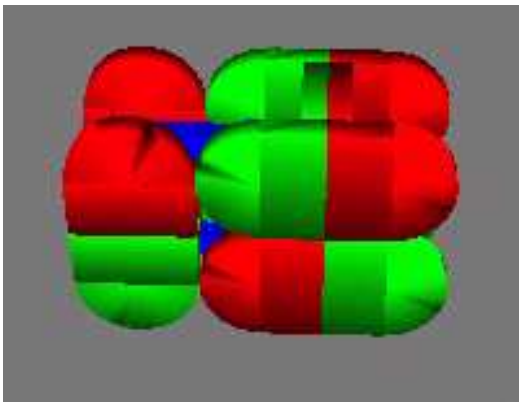

Figura 4.9: Configuración final de 3 módulos, formando una estructura circular, correspondiente al experimento 3

Según la especificación de estado dada en el experimento 2 (que se mantiene idéntica en este experimento), la configuración objetivo, mostrada por la figura 4.9 se define como

$$estado_{final} = \text{['-90.0::90.0', '0.0::90.0', '-90.0::0.0']}$$

En este experimento se utilizó la misma definición para las acciones y la misma función de recompensa que en el experimento 2.

Resultados

Se realizaron 13 episodios de aprendizaje, al final de los cuales se evaluó la política aprendida hasta ese momento. Los resultados de estos primeros 13 episodios se muestran en las figuras 4.10 y 4.11.

En el caso de la figura 4.10, se muestra la cantidad de acciones realizadas durante cada episodio, antes de alcanzar la configuración objetivo. Es notable que la tendencia global de los datos obtenidos resulta en un comportamiento descendente, lo que igualmente al caso anterior, indica que se está aprendiendo. Los picos de crecimiento de la cantidad de acciones que se pueden observar, también se deben, como en el caso anterior, a la exploración realizada durante el aprendizaje. Cabe destacar que la mayor complejidad de la configuración final utilizada en este experimento se refleja en la mayor cantidad de acciones realizadas (en promedio), en relación al experimento 2.

En la figura 4.11 se muestra la cantidad de acciones necesarias para alcanzar la configuración objetivo, según la política aprendida. En este caso, se puede apreciar que la política obtenida todavía resulta inestable (en el sentido de que no determina siempre las mismas acciones). Este hecho se debe a que aún no fue explorada suficiente cantidad del espacio de estados para poder determinar fehacientemente la secuencia de acciones óptima. De esto resulta que existen estados en los que más de una acción tiene un valor de Q máximo (hasta ese momento), lo que produce que no sea posible elegir de forma determinística la mejor acción posible para dichos estados.

A raíz de ello, se extendió la fase de aprendizaje. Se intercalaron series de episodios de aprendizaje con episodios de evaluación de la política aprendida, hasta que se obtuvo una política estable (es decir, que siempre realiza las mismas acciones).

Los resultados de estos episodios se muestran en las figuras 4.12 - 4.15.

En las figuras 4.12 y 4.14 se puede observar cómo a medida que se realizan más cantidad de episodios de aprendizaje, la cantidad de acciones requeridas disminuye, confluyendo hacia un valor estable. Esto indica que el robot está aprendiendo, efectivamente utilizando la experiencia pasada.

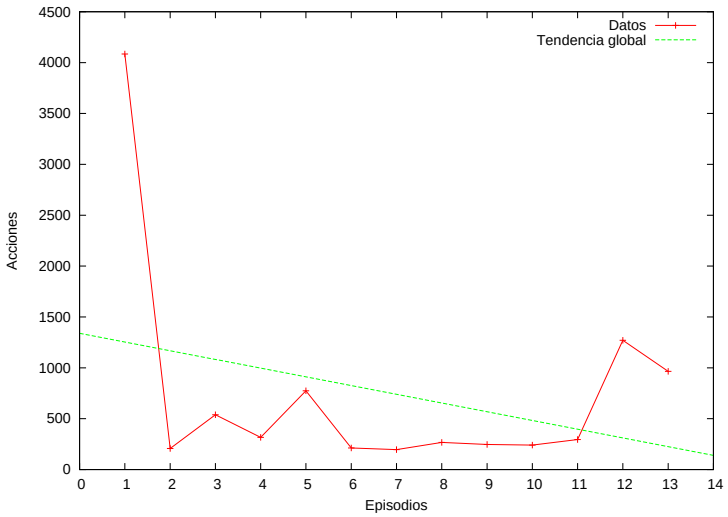

Figura 4.10: Cantidad de acciones realizadas antes de alcanzar el objetivo, durante la fase de aprendizaje, de 13 episodios, de una política de reconfiguración para el experimento 3

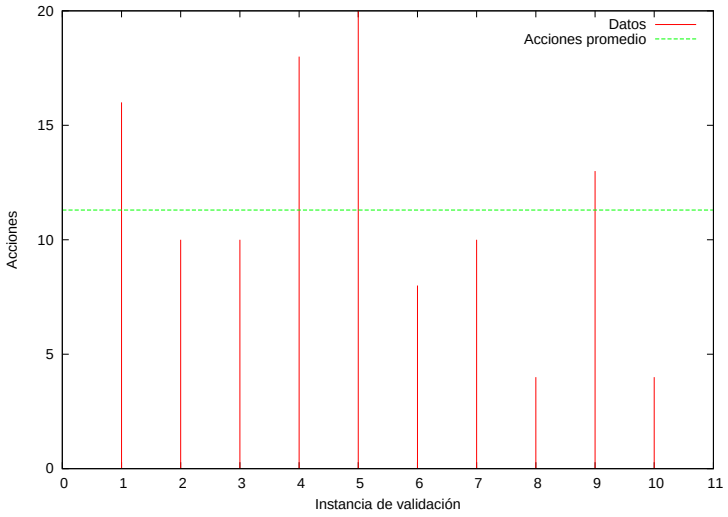

Figura 4.11: Cantidad de acciones realizadas antes de alcanzar el objetivo, utilizando la política aprendida mediante 13 episodios, para el experimento 3

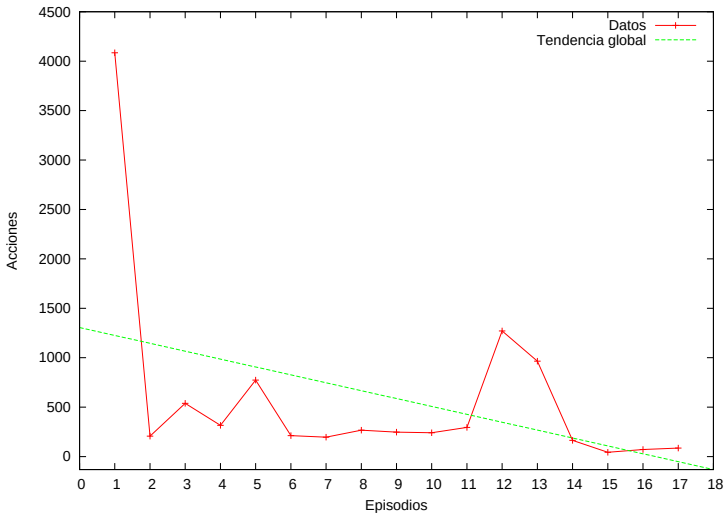

Figura 4.12: Cantidad de acciones realizadas antes de alcanzar el objetivo, durante la fase de aprendizaje, de 17 episodios, de una política de reconfiguración para el experimento 3

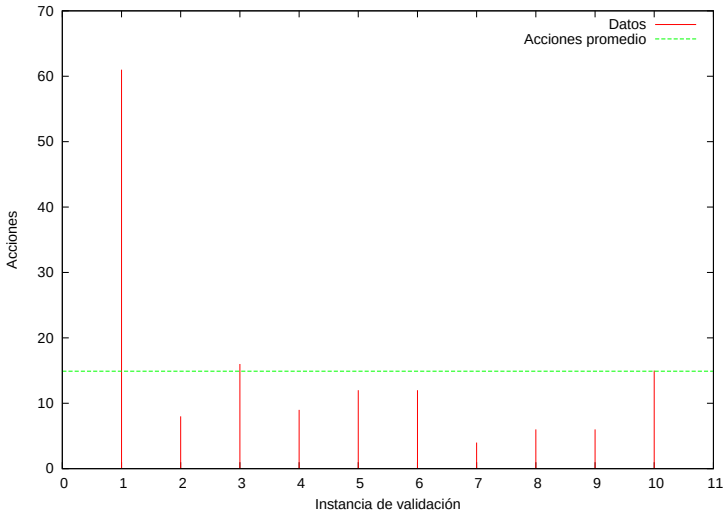

Figura 4.13: Cantidad de acciones realizadas antes de alcanzar el objetivo, utilizando la política aprendida mediante 17 episodios, para el experimento 3

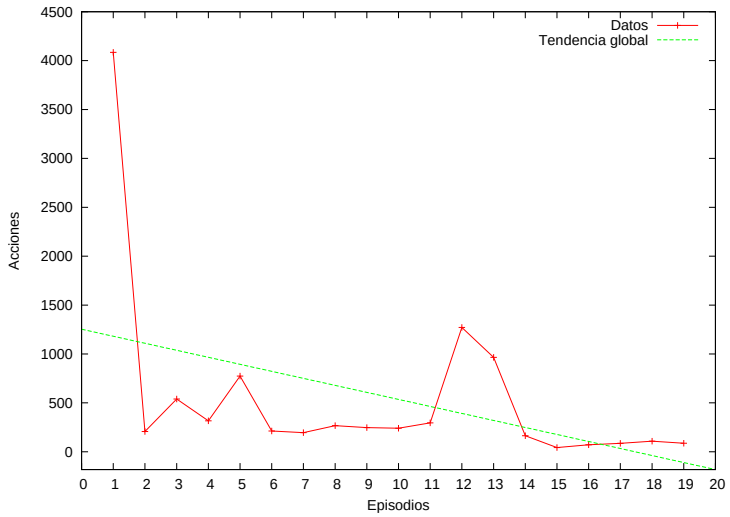

Figura 4.14: Cantidad de acciones realizadas antes de alcanzar el objetivo, durante la fase de aprendizaje, de 19 episodios, de una política de reconfiguración para el experimento 3

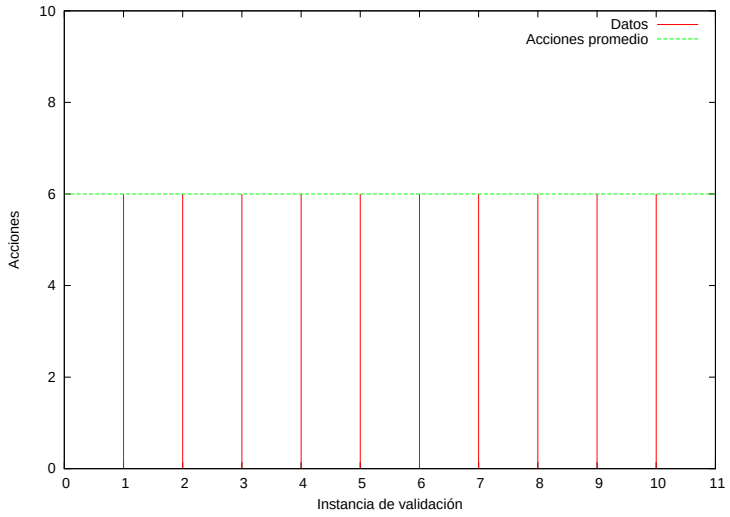

Figura 4.15: Cantidad de acciones realizadas antes de alcanzar el objetivo, utilizando la política aprendida mediante 19 episodios, para el experimento 3

Por otra parte, en la figura 4.13 se puede apreciar que luego de 17 episodios de aprendizaje aún no se consiguió obtener una política estable, pero que, según se ve en la figura 4.15, luego de 19 episodios efectivamente sí se alcanza una política estable y altamente eficiente (en función de la diferencia entre la cantidad de acciones realizadas durante el aprendizaje, y la cantidad de acciones requeridas por la política).

Discusión

Como se puede observar, se sigue cumpliendo el hecho de que a medida que se progresa con los episodios de aprendizaje, la cantidad de acciones realizadas antes de llegar a la configuración objetivo decrece a causa del aprendizaje realizado. Asimismo, se puede observar cómo luego de cierta cantidad de episodios, la política aprendida se estabiliza en una cantidad mínima de acciones necesarias.

4.2.3. Experimento 4: De estructura lineal a circular, acciones complejas

En este caso, se introduce una variación respecto del experimento anterior. En vez de utilizar acciones "simples", se redefinen las acciones posibles, de forma que se puedan activar múltiples motores simultáneamente. La principal razón para este cambio consiste en evaluar si al poder realizar más movimientos en simultáneo, la política final resulta más eficiente.

Las acciones "complejas" quedan entonces definidas según

$$\begin{aligned} \textit{acción} &= \text{'}\textit{movimiento}\text{'} \vee \text{'}\textit{movimiento}; \textit{acción}\text{'}^{8} \ , \\ \textit{movimiento} &= \text{acción simple, según ecuación 4.6 .} \end{aligned} \tag{4.8}$$

Esta definición se completa con dos restricciones:

1. En una misma acción no pueden haber dos movimientos que actúen sobre el mismo motor
2. Como máximo puede haber 6 movimientos en una acción compleja (es decir, la cantidad de motores existentes en el robot). Esta restricción se desprende de la restricción anterior.

[8] cabe aclarar que si bien en la definición una acción está representada como una secuencia de movimientos, no importa el orden en el que aparezcan los mismos; es decir todas las acciones que puedan ser consideradas permutaciones de los mismos movimientos, serán consideradas como la misma acción

Resultados

En este experimento, se realizó la evaluación de la política luego de unos pocos episodios (dado que también se pretendía verificar si al utilizar acciones complejas se aceleraba el proceso de aprendizaje).

Los resultados se pueden ver en las figuras 4.16 - 4.21.

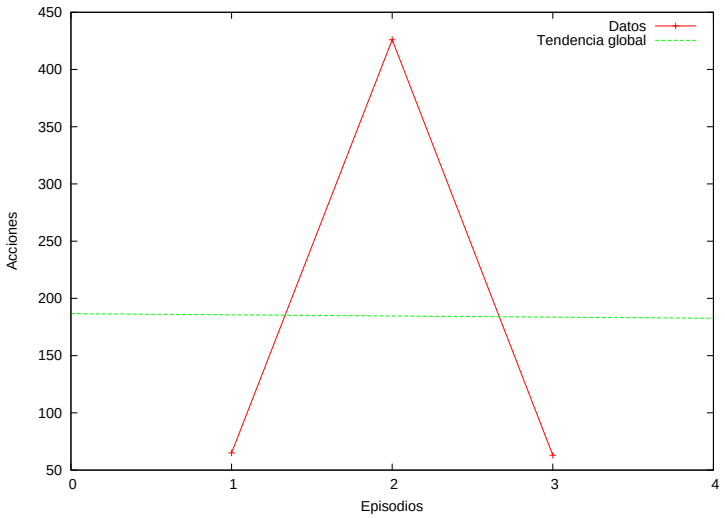

Figura 4.16: Cantidad de acciones realizadas antes de alcanzar el objetivo, durante la fase de aprendizaje, de 3 episodios, de una política de reconfiguración para el experimento 4

Discusión

En el caso de la primer figura, se puede observar que la cantidad de acciones realizadas durante la fase de aprendizaje fluctúa fuertemente. Si bien esto sólo no es indicio de que la política aún no haya sido refinada suficientemente, en conjunción con la figura siguiente (4.17) se puede verificar que efectivamente así sucede (ya que la política obtenida resulta inestable). La razón de ello es que la cantidad de episodios de aprendizaje realizados es aún escasa. En la siguiente figura (correspondiente al aprendizaje realizado durante 7 episodios), se evidencia nuevamente una tendencia global creciente en la cantidad de acciones realizadas durante el aprendizaje, incluso presentando una pendiente considerable. Sin embargo, es notable que la cantidad de acciones realizadas es, en termino medio, bastante inferior a las realizadas durante los primeros episodios de aprendizaje en los experimentos anteriores. Este hecho indica que el aprendizaje efectivamente se ve acelerado por el hecho de utilizar acciones complejas. La política obtenida, aún es inestable

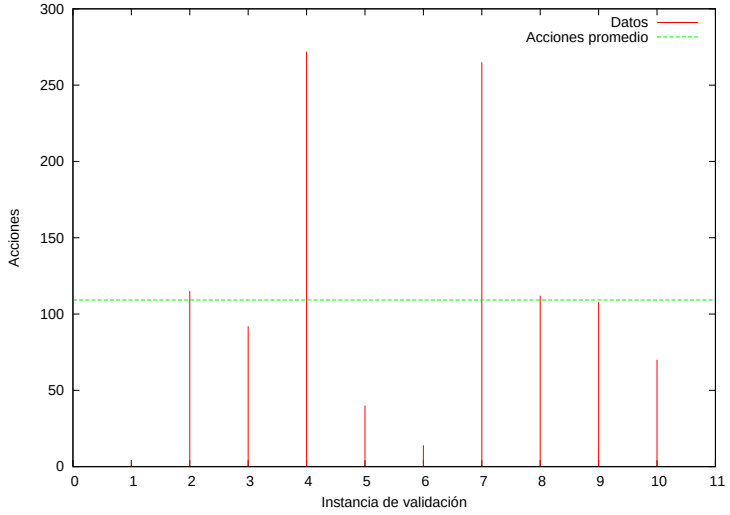

Figura 4.17: Cantidad de acciones realizadas antes de alcanzar el objetivo, utilizando la política aprendida mediante 3 episodios, para el experimento 4

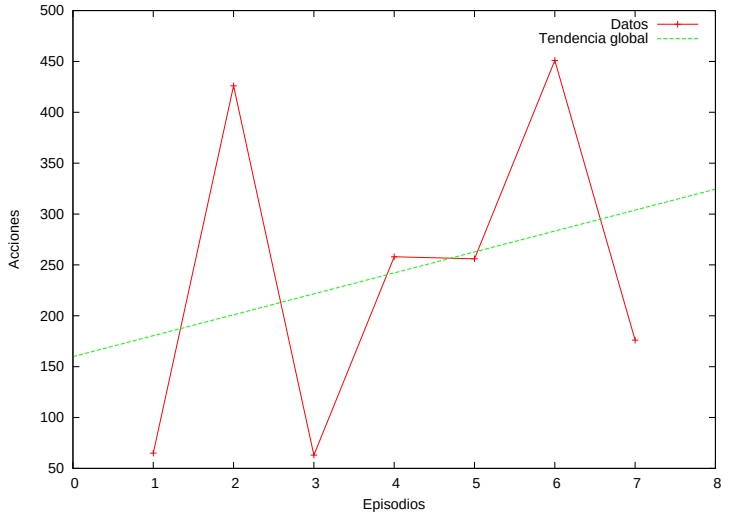

Figura 4.18: Cantidad de acciones realizadas antes de alcanzar el objetivo, durante la fase de aprendizaje, de 7 episodios, de una política de reconfiguración para el experimento 4

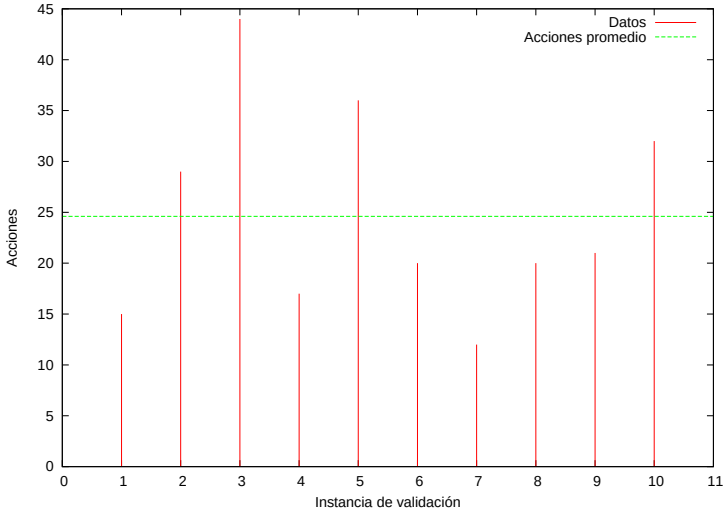

Figura 4.19: Cantidad de acciones realizadas antes de alcanzar el objetivo, utilizando la política aprendida mediante 7 episodios, para el experimento 4

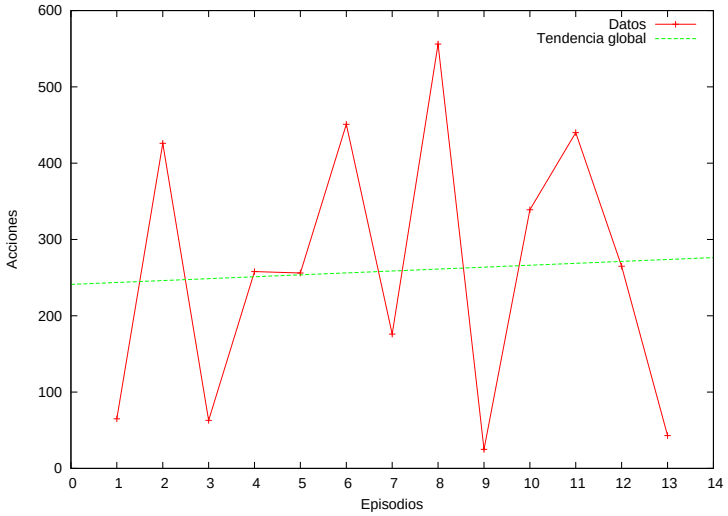

Figura 4.20: Cantidad de acciones realizadas antes de alcanzar el objetivo, durante la fase de aprendizaje, de 13 episodios, de una política de reconfiguración para el experimento 4

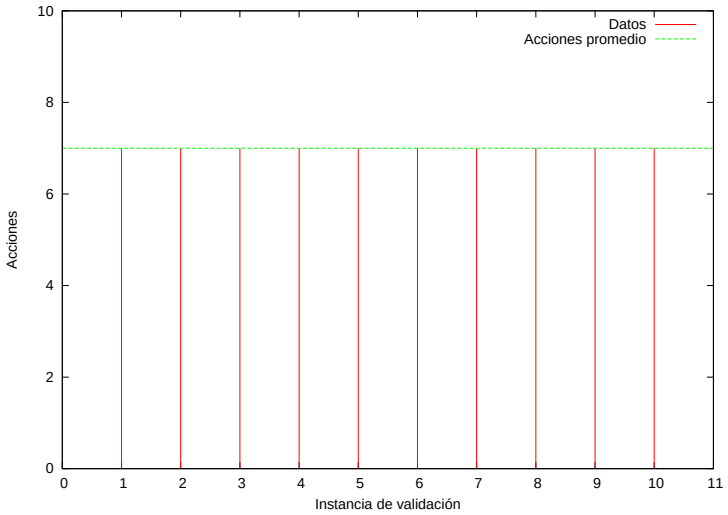

Figura 4.21: Cantidad de acciones realizadas antes de alcanzar el objetivo, utilizando la política aprendida mediante 13 episodios, para el experimento 4

(según se observa en la figura 4.19), lo que indica que es necesario realizar mayor cantidad de episodios de aprendizaje.

Finalmente, luego de 13 episodios de aprendizaje, se obtiene una política estable (ver figura 4.21). Aún cuando durante la fase de aprendizaje, la tendencia global sea creciente, la pendiente es notablemente inferior a la de la figura 4.18, lo que evidencia que en término medio la cantidad de acciones realizadas durante los subsiguiente episodios efectivamente disminuye.

Otro factor a destacar es que la cantidad de acciones realizadas por la política va decreciendo (en término medio) a medida que la misma se refina, hasta obtener una política estable cuya cantidad de acciones se mantiene constante.

Si bien la cantidad de acciones requeridas por la política obtenida no es estrictamente menor que las acciones requeridas por la política obtenida durante el experimento 3, y aún cuando no se puede afirmar que la primera sea mejor (o más eficiente) que la segunda, el resultado obtenido es suficientemente bueno, ya que la cantidad de episodios requeridos para alcanzar una política estable es menor a la cantidad de episodios requeridos durante el experimento 3, y la diferencia entre las acciones que realizan ambas políticas es mínima.

4.2.4. Experimento 5: De estructura lineal a circular con acoplamiento/desacoplamiento, acciones complejas

Ya habiendo logrado una reconfiguración básica exitosamente, en este experimento se pretende complicar la situación al introducir un aspecto de las capacidades del robot aún no explorado: el acoplamiento entre módulos. El objetivo de este experimento es entonces lograr una configuración similar a la de los últimos dos casos (es decir, partiendo de una configuración lineal llegar a una configuración circular), con la diferencia de que el robot deberá terminar con todos sus módulos acoplados entre sí, formando un anillo.

Para lograr esto, es necesario:

1. Modificar la definición de estado, para que incluya los acoplamientos entre módulos
2. Modificar la definición de acción, para permitir acoplar y desacoplar módulos

Para poder realizar cambios que afecten los acoplamientos entre los distintos módulos, y lograr aprender de ello, es necesario que el estado utilizado para el aprendizaje describa los acoplamientos entre los módulos. Se define el estado entonces como

$$\begin{aligned} estado &= (\acute{a}ngulos, acoplamientos)\ , \\ \acute{a}ngulos &= \text{estado del robot según ecuación 4.5}\ , \\ acoplamientos &= [acoplamiento_0, acoplamiento_1, \ldots]^{9}\ , \\ acoplamiento_i &= (m\acute{o}dulo_a, m\acute{o}dulo_b, cara^{10})\ , \\ cara &= \text{'inferior'} \vee \text{'delantera'} \vee \text{'trasera'}\ . \end{aligned} \tag{4.9}$$

En función de esta nueva definición de estado, los estados inicial y final para este experimento quedan expresados como:

$$\begin{aligned} estado_{inicial} &= (\text{['0.0::0.0', '0.0::0.0', '0.0::0.0'],} \\ &\quad \text{[(1, 0, 'inferior'), (2, 1, 'inferior')])} \\ estado_{final} &= (\text{['-90.0::90.0', '0.0::90.0', '-90.0::0.0'],} \\ &\quad \text{[(1, 0, 'inferior'), (2, 1, 'inferior'), (0, 2, 'inferior')])} \end{aligned}$$

[9] todas las permutaciones de acoplamientos serán consideradas como la misma. Véase la aclaración correspondiente a la definición de las acciones complejas (ecuación 4.8)

[10] del módulo a

Para permitir realizar acoplamientos y desacoplamientos entre módulos, las acciones disponibles se definen como

$$\begin{aligned} \textit{acción}' &= \textit{acción-compleja} \vee \textit{acoplamiento} \ , && (4.10) \\ \textit{acoplamiento} &= \text{'dock}|\textit{módulo}_a|\textit{módulo}_b|\textit{cara}^7\text{'} \vee \\ &\quad \text{'undock}|\textit{módulo}_a||\textit{cara}^7\text{'} \ , && (4.11) \\ \textit{acción-compleja} &= \text{acción según definida por la ecuación 4.8} \ . \end{aligned}$$

Al utilizar estas nuevas definiciones para estado y acción, es posible que durante un episodio el robot quede en un estado desde el cuál le será prácticamente imposible completar el objetivo. Por ejemplo, si se desacopla un módulo, si bien teóricamente podría volver a acoplarse, en la práctica, las condiciones que se deben cumplir para ello son tan estrictas que la probabilidad de que esto suceda es baja.

Por esta razón, si el robot se encontrara en una situación de este tipo, en vez de permitirle continuar, se optó por abortar el episodio, retornando una recompensa negativa. La función de recompensa utilizada para este experimento está dada por la siguiente ecuación

$$\textit{recompensa} = \begin{cases} 1 & \text{si se alcanzó el estado objetivo} \\ -1 & \text{si se desacopló algún módulo} \\ 0 & \text{en otro caso} \end{cases} \qquad (4.12)$$

Resultados

Se realizó una etapa de aprendizaje compuesta por 43 episodios, los cuales fueron todos abortados, sin llegar a alcanzar la configuración objetivo.

En el gráfico 4.22 se puede apreciar la cantidad de acciones realizadas durante cada episodio de aprendizaje.

Discusión

El hecho de haber introducido la posibilidad de abortar un episodio sin que el robot llegara a cumplir su objetivo, hace que sea necesario realizar una cantidad mucho mayor de episodios de aprendizaje, o bien reevaluar la definición de episodio.

Por lo que se puede apreciar en la figura 4.22, existe una tendencia decreciente en la cantidad de acciones realizadas por cada episodio, antes de ser abortado. Si bien esta tendencia podría indicar que efectivamente hace falta

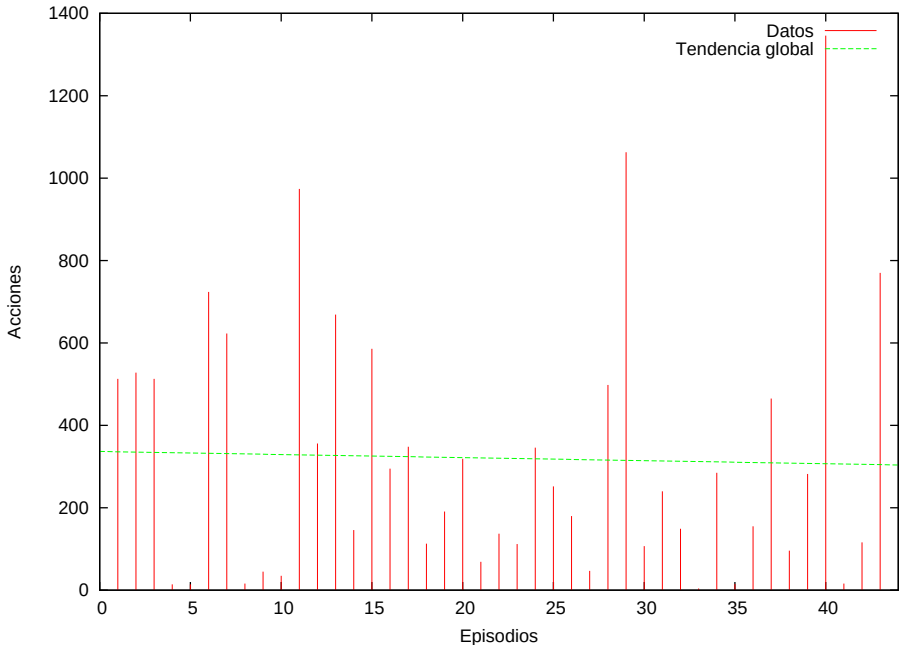

Figura 4.22: Cantidad de acciones realizadas durante cada episodio de aprendizaje, antes de ser abortado.

realizar mayor cantidad de episodios de aprendizaje a fin de lograr determinar una política estable, debido a que todos los episodios fueron abortados, esta correlación podría estar infundada. Esto se desprende del hecho de que la función de recompensa utilizada asigna recompensas de forma puramente demorada. Por lo tanto, la única información que es posible extraer de cada episodio abortado es que la serie de acciones realizadas no son efectivas para alcanzar el objetivo.

4.2.5. Experimento 6: De estructura lineal a circular con acoplamiento, sin desacoplamiento, acciones simples

Debido a los resultados obtenidos durante el experimento 5, se modificaron las definiciones de acción y función de recompensa, para evitar los problemas ocurridos.

En el caso de las acciones permitidas, se utilizó una definición similar a la dada por la ecuación 4.10, pero en vez de utilizar acciones complejas, se utilizaron acciones simples, y se restringió la posibilidad de acoplamiento únicamente a la acción *dock*. Es decir, básicamente se simplificaron las ac-

ciones y se evitó la acción de desacoplamiento, para evitar tener que abortar episodios. Esta decisión se basó en el hecho de que no era necesario la acción de desacoplamiento para poder llegar desde la configuración inicial hasta la configuración objetivo.

La función de recompensa quedó definida según

$$recompensa = \begin{cases} -1 & \text{en caso de abortar el episodio } [11] \\ recompensaParcial & \text{en otro caso} \end{cases} \tag{4.13}$$

donde *recompensaParcial* es la función definida en la tabla 4.2.

```
def recompensaParcial(estado, objetivo):
    ángulos, acoplamientos = estado
    recompensa = 0
    if ángulos == objetivo[0]:
        recompensa += 1
    if acoplamientos == objetivo[1]:
        recompensa += 1
    return recompensa
```

Tabla 4.2: Definición de la función *recompensaParcial*.

Según esta definición, la función de recompensa asigna recompensa por llegar a la configuración de ángulos correcta y por llegar a la configuración de acoplamientos correcta, siendo el máximo valor cuando ambas partes del estado son correctas (es decir, cuando se llegó a la configuración objetivo).

Resultados

De las simulaciones realizadas, se obtuvieron los resultados que se muestran en las figuras 4.23 - 4.28.

Discusión

En las figuras 4.23, 4.25 y 4.27 se puede observar la evolución de la cantidad de acciones realizadas durante la fase de aprendizaje, a medida que se realizaban mayor cantidad de episodios. La pendiente negativa indica nuevamente el efecto del aprendizaje en base a la experiencia obtenida en episodios

[11] Si bien anteriormente se mencionó que la modificación de las acciones debería evitar tener que abortar episodios, disponer de este caso en la función de recompensa no produce efectos negativos, y previene casos no previstos en los que el episodio debe ser abortado.

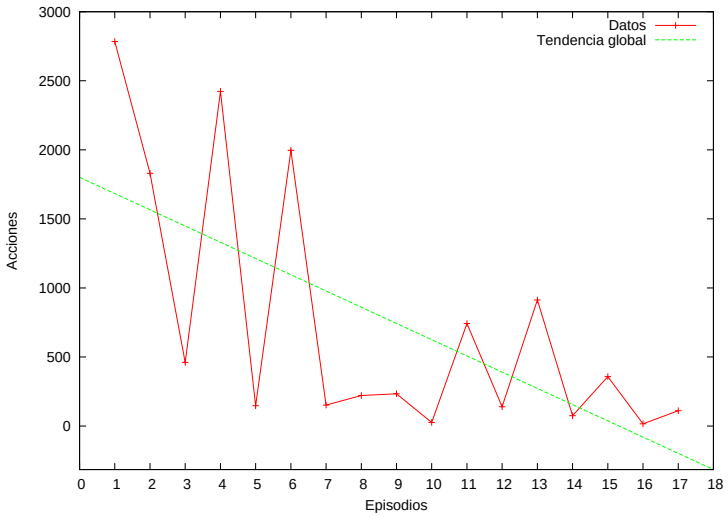

Figura 4.23: Cantidad de acciones realizadas antes de alcanzar el objetivo, durante la fase de aprendizaje, de 17 episodios, de una política de reconfiguración para el experimento 6

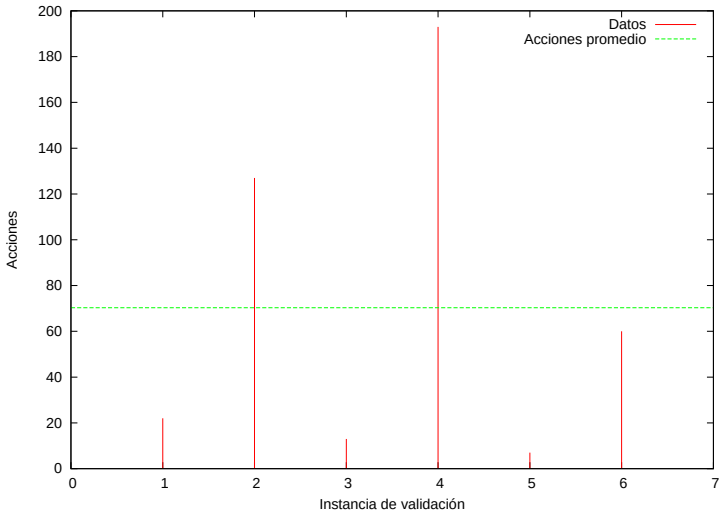

Figura 4.24: Cantidad de acciones realizadas antes de alcanzar el objetivo, utilizando la política aprendida mediante 17 episodios, para el experimento 6

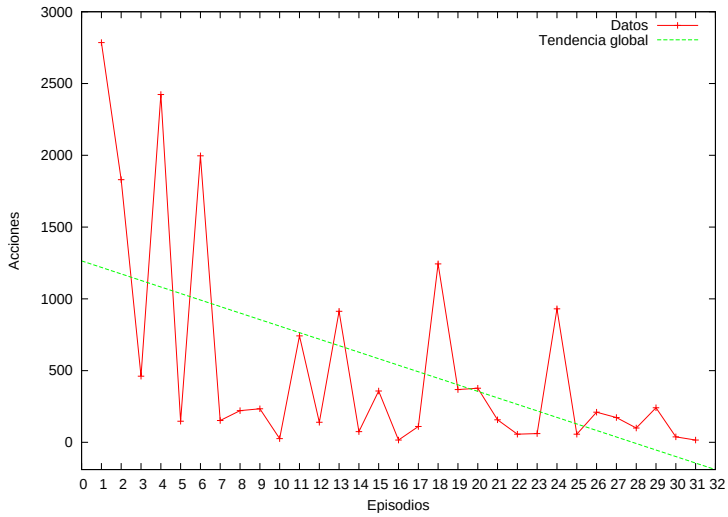

Figura 4.25: Cantidad de acciones realizadas antes de alcanzar el objetivo, durante la fase de aprendizaje, de 31 episodios, de una política de reconfiguración para el experimento 6

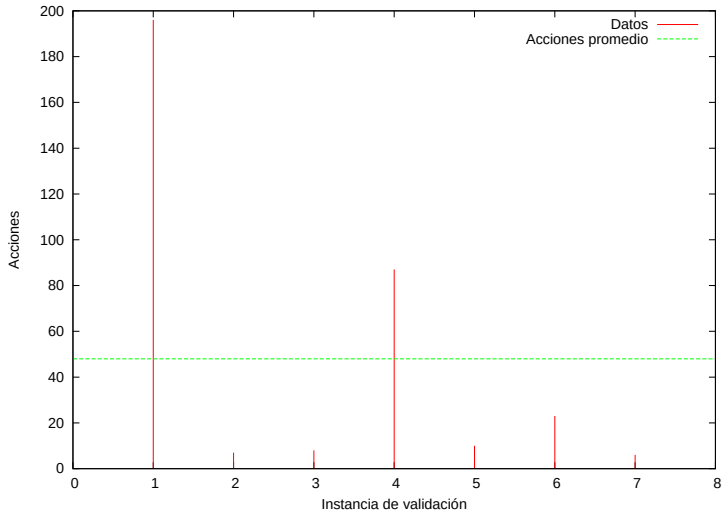

Figura 4.26: Cantidad de acciones realizadas antes de alcanzar el objetivo, utilizando la política aprendida mediante 31 episodios, para el experimento 6

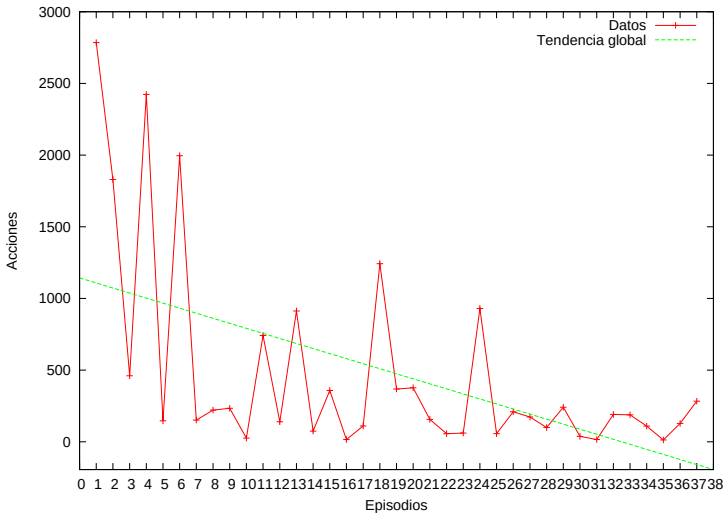

Figura 4.27: Cantidad de acciones realizadas antes de alcanzar el objetivo, durante la fase de aprendizaje, de 37 episodios, de una política de reconfiguración para el experimento 6

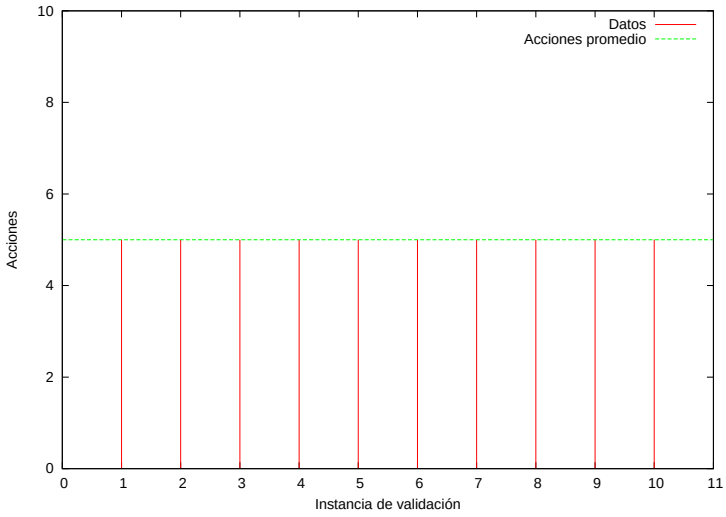

Figura 4.28: Cantidad de acciones realizadas antes de alcanzar el objetivo, utilizando la política aprendida mediante 37 episodios, para el experimento 6

pasados. Sin embargo, dicha pendiente se suaviza a medida que se realizan más episodios de aprendizaje. Esto indica que la cantidad de acciones realizadas durante dichos episodios, tiende a estabilizarse, lo que sugiere una mayor utilización de la experiencia, representada por la política obtenida. Esto quiere decir que si la cantidad de acciones realizadas durante la fase de aprendizaje se estabiliza, es altamente probable que se esté llegando a una política estable. La diferencia entre la cantidad de acciones realizadas en los primeros episodios de aprendizaje en relación con la cantidad de acciones realizadas durante los últimos, indica una importante mejora, atribuible indudablemente al aprendizaje realizado.

En las figuras 4.24, 4.26 y 4.28 se puede observar la evolución del rendimiento obtenido por la política derivada de la fase de aprendizaje, a medida que se realizaban mayor cantidad de episodios de aprendizaje. Como se puede ver, inicialmente la política resultaba altamente inestable, mientras que en las últimas etapas de aprendizaje, si bien aún no se había alcanzado la estabilidad, la cantidad de acciones promedio realizadas por la política en ese momento resultaban notablemente inferiores que en estadios primitivos del aprendizaje.

Finalmente, en la figura 4.28 se puede ver que se alcanza una política estable, que involucra una cantidad muy pequeña de acciones para lograr alcanzar la configuración objetivo. Claramente, este problema es más complejo que el planteado en los experimentos 3 y 4, lo que se refleja en la mayor cantidad de episodios necesarios para llevar a cabo la tarea (se necesitaron 37 episodios de aprendizaje para obtener una política estable).

4.2.6. Experimento 7: De estructura circular a lineal con acoplamiento/desacoplamiento, acciones complejas

Completada entonces la primera fase de experimentación (en donde se estudiaron distintas variantes sobre un mismo problema – pasar de una configuración lineal a una configuración circular), en los siguientes dos experimentos se intenta realizar la tarea opuesta, es decir, pasar de una configuración circular completamente acoplada a una configuración lineal.

Las acciones y el estado definidos para este experimento son los mismos que los utilizados para el experimento 5.

La función de recompensa utilizada, sin embargo, debió ser modificada. Dado que en este experimento sí era necesario desacoplar módulos, hubiera sido posible que algún módulo quedara completamente desacoplado del resto, y si bien hubiera podido volver a acoplarse (y así ser capaz de alcanzar el

estado objetivo), de acuerdo a las condiciones que se deben cumplir para que tal acoplamiento suceda, es tan poco probable, que resultó más práctico asignar una recompensa negativa a este hecho (para evitar que suceda en futuras ocasiones), y abortar la simulación (esta decisión se basa en el mismo razonamiento realizado para el experimento 5).

Por esta razón, la función de recompensa se definió como

$$recompensa = \begin{cases} 1 & \text{si se alcanzó la configuración objetivo} \\ -1 & \text{si se abortó la simulación} \\ 0 & \text{en otro caso} \end{cases} \tag{4.14}$$

Dado que es posible abortar las simulaciones, se redefinió un episodio como la secuencia de simulaciones necesarias hasta llegar a la configuración objetivo (es decir, la secuencia de simulaciones que son abortadas, más la simulación que produce recompensa positiva). En función de esta definición, la cantidad de acciones realizadas durante un episodio es la suma de las acciones realizadas durante las simulaciones que lo componen.

Resultados

En este experimento se realizaron 13 episodios que representan un total de 70 simulaciones. Al final de los 13 episodios no se obtuvo sin embargo una política que cumpliera el objetivo (la política obtenida luego de 13 episodios cicla entre 3 configuraciones las cuales están todas completamente acopladas).

El comportamiento cíclico se puede apreciar en la figura 4.29, que representa un ciclo completo producido por las acciones seleccionadas por la política aprendida.

Los resultados de las simulaciones se presentan en las figuras 4.30 - 4.33.

Discusión

Según se puede observar en las figuras 4.30, 4.31, 4.32 y 4.33, la tendencia de la cantidad de acciones realizadas durante los episodios de aprendizaje varía a medida que se realizan más episodios. Esto se debe a que cuando la cantidad de episodios de aprendizaje realizados es insuficiente, la aproximación realizada en base a los datos existentes, no se condice con la tendencia global a largo plazo. Se observa también, que inicialmente, la pendiente de dicha aproximación es más pronunciada que cuando se cuenta con mayor cantidad de datos (es decir, luego de haber realizado mayor cantidad de episodios de aprendizaje). Esto es así ya que a medida que se realizan más episodios, la aproximación realizada se acerca más a la tendencia real. Por lo tanto,

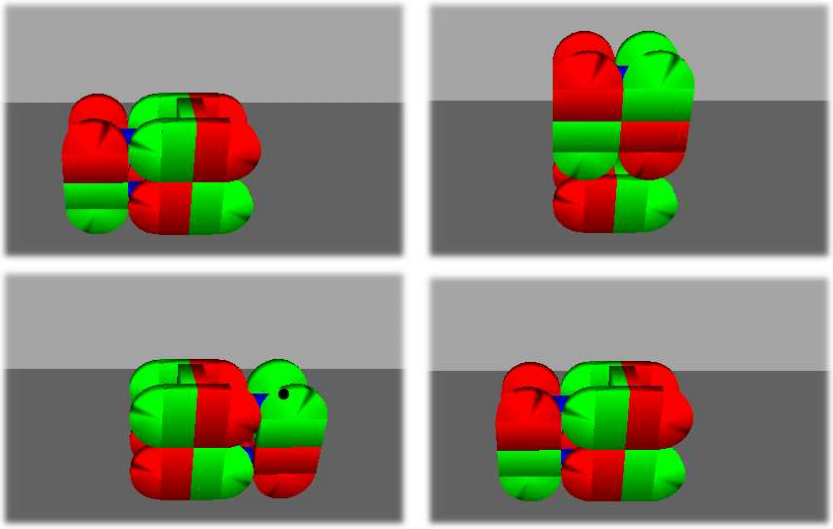

Figura 4.29: Estados correspondientes a un ciclo en las acciones producidas por la política aprendida en el experimento 7. Arriba izquierda: estado 1. Arriba derecha: estado 2. Abajo izquierda: estado 3. Abajo derecha: estado 4 (igual al estado 1).

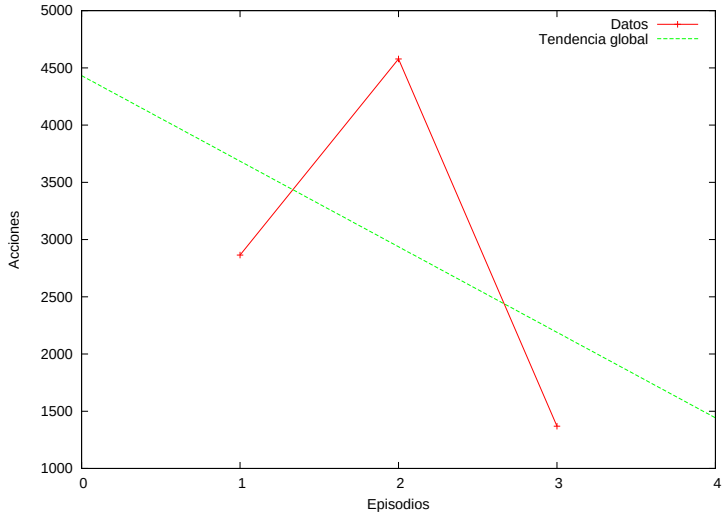

Figura 4.30: Cantidad de acciones realizadas antes de alcanzar el objetivo, durante la fase de aprendizaje, de 3 episodios, de una política de reconfiguración para el experimento 7

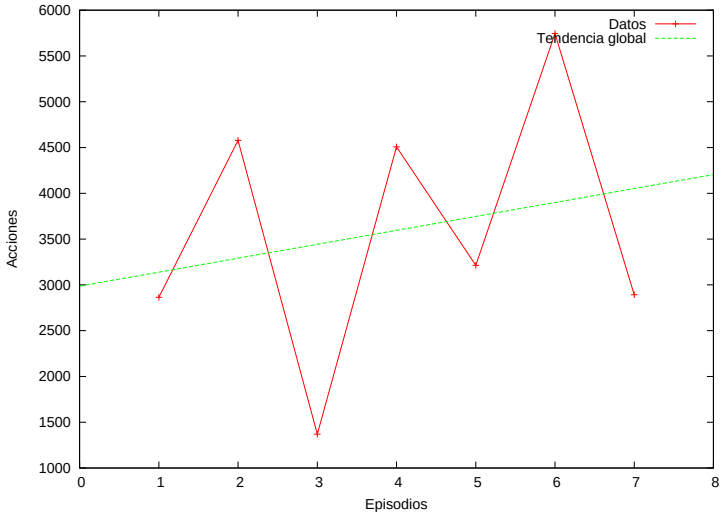

Figura 4.31: Cantidad de acciones realizadas antes de alcanzar el objetivo, durante la fase de aprendizaje, de 7 episodios, de una política de reconfiguración para el experimento 7

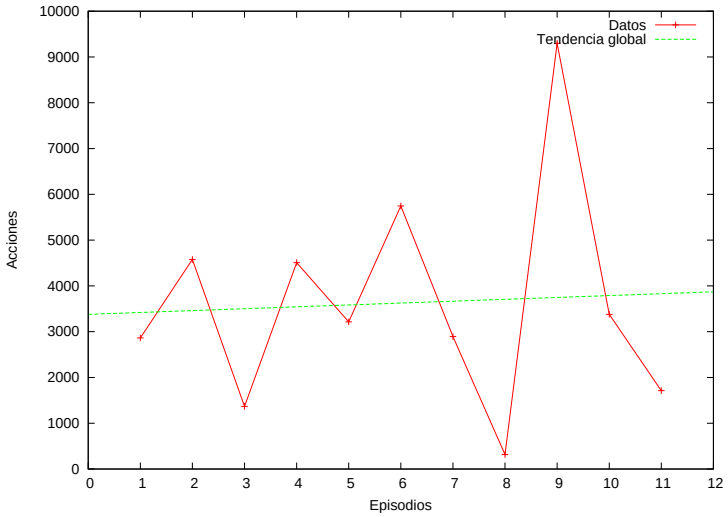

Figura 4.32: Cantidad de acciones realizadas antes de alcanzar el objetivo, durante la fase de aprendizaje, de 11 episodios, de una política de reconfiguración para el experimento 7

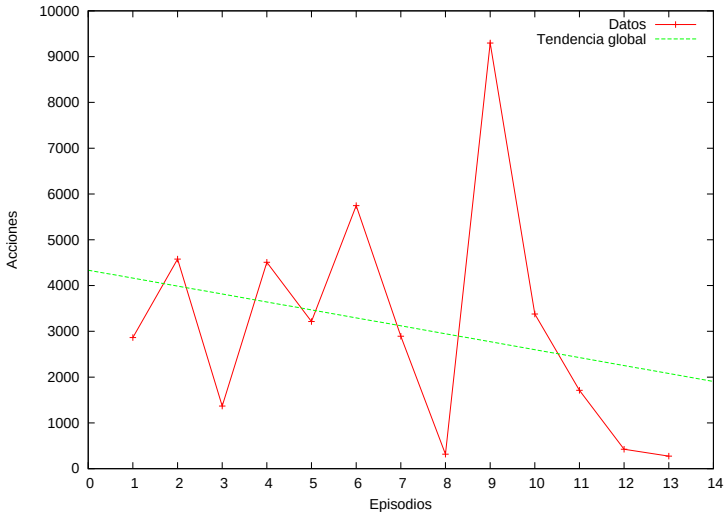

Figura 4.33: Cantidad de acciones realizadas antes de alcanzar el objetivo, durante la fase de aprendizaje, de 13 episodios, de una política de reconfiguración para el experimento 7

una fluctuación en dicha curva, podría indicar la necesidad de realizar mayor cantidad de episodios de aprendizaje.

Durante la observación del comportamiento del robot al utilizar la política obtenida luego de 13 episodios, se destaca el hecho de que el mismo cicla entre tres configuraciones, y por tal motivo no logra alcanzar la configuración objetivo. La política derivada, si bien realiza siempre las mismas acciones, no es satisfactoria, pues no permite alcanzar la configuración objetivo. Este inconveniente probablemente pueda ser superado realizando mayor cantidad de episodios de aprendizaje, a fin de proveer mayor cantidad de acciones exploratorias que permitan determinar mejores formas de alcanzar la configuración objetivo.

Por lo tanto, según se ve en los gráficos presentados, y según se pudo observar durante las simulaciones realizadas, el comportamiento presentado por este robot, con gran probabilidad se deba a la insuficiente cantidad de episodios de aprendizaje realizados.

En esta ocasión, se decidió no continuar realizando episodios de aprendizaje, puesto que se consideró más provechoso modificar el experimento para utilizar acciones simples en vez de acciones complejas. Esto se debe a que la utilización de acciones simples reduce el tamaño del espacio de estado-acción, permitiendo encontrar con mayor probabilidad una política óptima.

4.2.7. Experimento 8: De estructura circular a lineal con acoplamiento/desacoplamiento, acciones simples

Según lo mencionado anteriormente, este experimento es básicamente una simplificación del experimento anterior, modificando la definición de acción para utilizar acciones simples. Se pretende evaluar que el hecho de no haber alcanzado una política óptima en el experimento anterior se debe a que la cantidad de episodios de aprendizaje realizados resulta insuficiente para el tamaño del espacio de estado-acción. Por lo tanto, la simplificación de las acciones debe producir una reducción en el tamaño del espacio de estado-acción, lo que permite encontrar una política óptima con relativamente pocos episodios de aprendizaje.

Resultados

En esta ocasión, se realizaron 7 episodios sumando un total de 526 simulaciones. Los resultados extraídos de los mismos se detallan mediante las figuras 4.34 - 4.37.

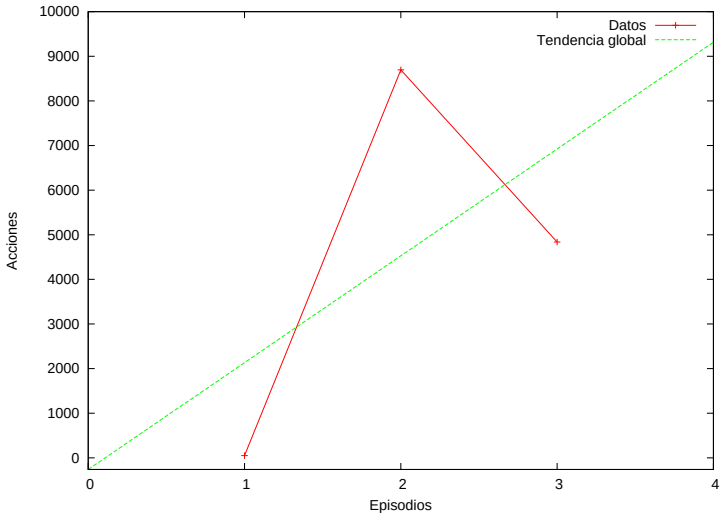

Figura 4.34: Cantidad de acciones realizadas antes de alcanzar el objetivo, durante la fase de aprendizaje, de 3 episodios, de una política de reconfiguración para el experimento 8

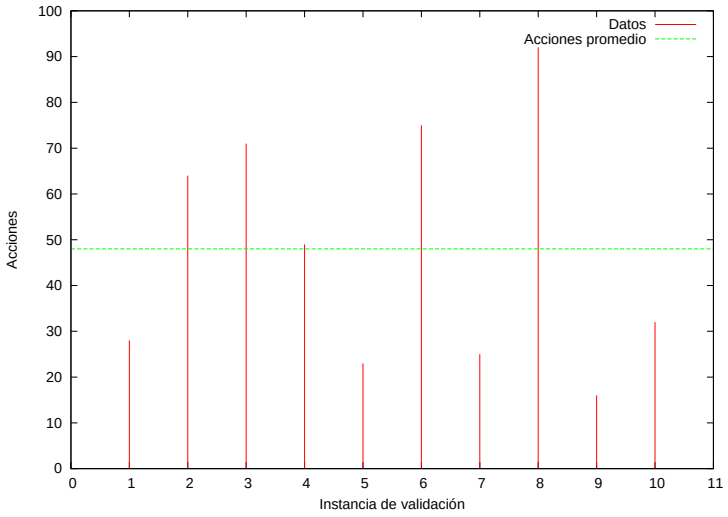

Figura 4.35: Cantidad de acciones realizadas antes de alcanzar el objetivo, utilizando la política aprendida mediante 3 episodios, para el experimento 8

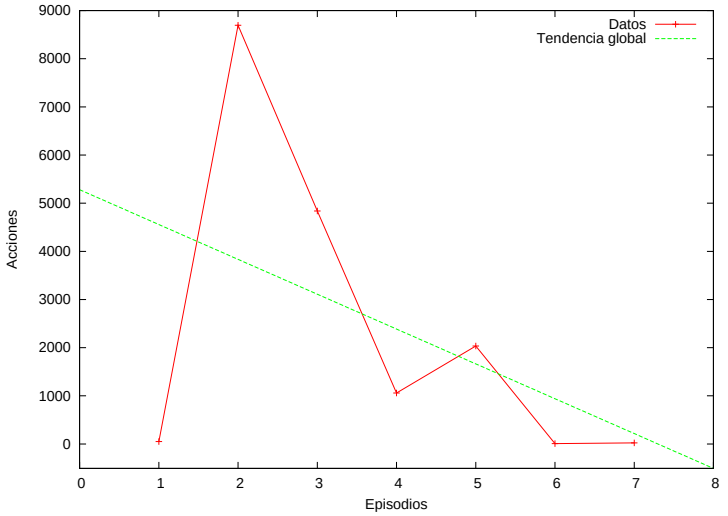

Figura 4.36: Cantidad de acciones realizadas antes de alcanzar el objetivo, durante la fase de aprendizaje, de 7 episodios, de una política de reconfiguración para el experimento 8

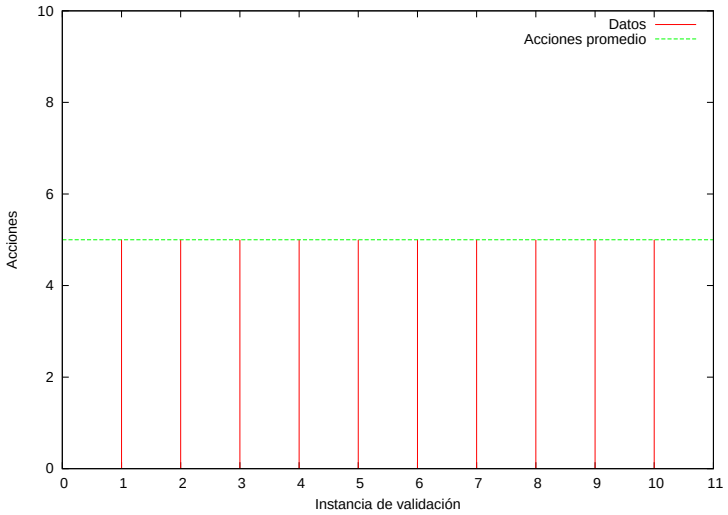

Figura 4.37: Cantidad de acciones realizadas antes de alcanzar el objetivo, utilizando la política aprendida mediante 7 episodios, para el experimento 8

Discusión

Como se puede observar en la figura 4.37, a diferencia del experimento anterior, en este caso se llegó a una política estable. Si bien se realizaron menor cantidad de episodios que en el experimento anterior, la cantidad de simulaciones realizadas fue mayor. La razón de ello se debe a que, al haber menor cantidad de acciones disponibles (recordar que en este experimento, a diferencia del anterior se están utilizando acciones simples), la probabilidad de realizar acciones de desacoplamiento (que conllevan a abortar simulaciones) es mayor. Al haber muchas simulaciones abortadas, es necesario realizar una mayor cantidad de simulaciones para cada episodio, lo que a su vez repercute en la cantidad de veces que se visitan los distintos estados, mejorando el aprendizaje del robot.

4.2.8. Conclusiones

A la luz de los resultados obtenidos mediante los experimentos realizados, se puede ver que efectivamente el aprendizaje de reconfiguración en un robot de tipo M-TRAN es factible. Naturalmente, ciertos aspectos, como la definición de un episodio, la función de recompensa, estados y acciones dependen fuertemente del problema a tratar, y deben ser refinados a medida que surgen inconvenientes con las definiciones utilizadas, pero en última instancia, es posible realizar tareas de aprendizaje de forma exitosa.

Queda visto entonces, que los robots de tipo M-TRAN disponen de la

capacidades necesarias para realizar exitosamente tareas de aprendizaje en el ámbito de la reconfiguración (por lo menos las que son relativamente sencillas).

Capítulo 5

Demostración conceptual: combinación de políticas

Luego de haber realizado diversas series de experimentos que verificaban la hipótesis respecto de la capacidad de un robot autoreconfigurable M-TRAN de aprender comportamientos de reconfiguración y locomoción mediante Aprendizaje por Refuerzo, se decidió llevar a cabo una demostración conceptual para mostrar la utilidad real de este método.

Se planteó un experimento complejo que pone a prueba todas las capacidades analizadas en el marco de este trabajo. Se decidió disponer de un obstáculo en el entorno de experimentación, que no fuera superable por el robot a menos que éste cambiara de configuración para poder rodearlo.

En este caso, se decidió que la forma de superar el obstáculo fuera deslizándose por debajo del mismo, adoptando una configuración lineal "tipo oruga". Este obstáculo podría estar representando una abertura en una pared que permitiera el acceso a una habitación contigua a la que se encuentra el robot. Este tipo de escenarios sería muy esperable de encontrar en situaciones tales como la exploración de un edificio derruido, por ejemplo.

El experimento planteado comienza con el robot situado en un extremo de entorno de experimentación, y configurado de forma circular. El robot debe avanzar hasta encontrar el obstáculo. Una vez detectado el mismo, debe reconfigurarse de forma de adoptar una configuración lineal que le permita desplazarse por debajo del obstáculo. Ya superado el mismo, el robot debe nuevamente reconfigurarse para adoptar una configuración que le permita continuar avanzando del otro lado.

El problema recién descripto puede ser divido en varios subproblemas, cada uno de los cuales requiere que el robot aprenda un comportamiento distinto para resolverlo. Mediante la combinación de estos comportamientos individuales es posible resolver el problema global y superar el obstáculo.

Los subproblemas inicialmente identificados son:

1. Realizar locomoción mediante una configuración circular
2. Reconfigurarse para pasar de una configuración circular a una configuración lineal
3. Realizar locomoción mediante una configuración lineal
4. Reconfigurarse para pasar de una configuración lineal a una configuración circular

Se decidió que el robot utilizado en el marco de estos experimentos estuviera constituido por 5 módulos. Esta cantidad sería suficiente para que el problema adquiriese una complejidad significativamente superior a los problemas analizados en el capítulo 4.

Para cada subproblema identificado se obtuvo una política que permite al robot conseguir el objetivo planteado por la situación (avanzar, o reconfigurarse, dependiendo del caso).

Una vez obtenidas estas políticas, se procedió a determinar la forma de combinarlas de manera de obtener un comportamiento global más complejo que reflejara los comportamientos asociados a las mismas, de acuerdo a las distintas situaciones encontradas.

Al realizar esta combinación de políticas, se advirtió un problema: si bien cada política adquirida previamente lograba solucionar el problema planteado para ellas, todas funcionaban únicamente bajo las condiciones en las que fueron adquiridas. Esto eso, una política de reconfiguración solamente lograba alcanzar su objetivo si se la evaluaba comenzando a partir del mismo estado inicial utilizado durante el aprendizaje de la misma.

Dado que inicialmente no era factible determinar el estado en el que terminaría de ejecutarse una política y comenzaría a ejecutarse la siguiente, fue necesario conseguir nuevas políticas que permitieran al robot transcurrir entre cualquiera de los estados finales obtenidos al seguir una política y el estado inicial requerido por la siguiente política. Se requirieron nuevas políticas para:

a. Pasar de los estados finales de la política 1 al estado inicial de la política 2.
b. Pasar de los estados finales de la política 3 al estado inicial de la política 4.

Una vez obtenidas las nuevas políticas fue posible para el robot superar el obstáculo mediante la combinación de todas las políticas desarrolladas previamente.

5.1. Combinación de políticas

Al combinar las distintas políticas aprendidas independientemente, de forma de poder resolver el problema global, es necesario determinar qué política debe ser utilizada en cada momento. Para ello se implementaron sensores de proximidad que permiten detectar obstáculos. Se definió un robot que incluía un sensor de proximidad en cada módulo, además de los sensores descriptos en el capítulo 4.

El valor de los sensores de obstáculos se calcula en base a la ecuación siguiente:

$$v = \begin{cases} 1 & \text{si } distancia[i] < 0{,}5 * longitud[i] + rango_s \\ 0 & \text{en otro caso} \end{cases} \tag{5.1}$$

donde

$$\begin{aligned} distancia &= \text{vector de distancia entre el sensor y el obstáculo} \\ i &= \text{dimensión en la que la distancia es máxima} \\ longitud &= \text{vector de longitudes de los lados del obstáculo} \\ rango_s &= \text{rango máximo de sensado del sensor} \end{aligned}$$

Por medio de estos nuevos sensores es posible determinar si el robot tiene un obstáculo por delante, encima o detrás suyo. Para determinar esto se toman los valores de los sensores ubicados en el módulo más cercano a la fuente emisora y del módulo más lejano. El primer valor permite determinar si existe un obstáculo delante del robot, mientras que el segundo indica si el mismo se encuentra por detrás del robot. Si ambos sensores están activados (su valor es 1), el obstáculo se encuentra encima del robot.

Se definieron un conjunto de reglas en función de los valores de estos sensores, que permiten determinar la política vigente en cada momento. Entonces, luego de cada acción se evalúan estas reglas y se puede determinar si es necesario actualizar la política vigente (es decir, elegir una política diferente a la actual). En el caso de las políticas de reconfiguración, el cambio de política se produce al alcanzar la nueva configuración (o sea, al llegar al estado final asociado con dichas políticas).

De acuerdo a lo que se puede observar en las tablas 5.1 y 5.2, se puede apreciar cómo la política es determinada por los sensores de obstáculos, en el caso de las políticas de locomoción (las únicas reglas que producen cambios corresponden a las políticas 1 y 3, que son las políticas de locomoción), mientras que en los casos de reconfiguración la segunda tabla indica la política

Política actual	Valor de los sensores		Próxima política
	Delantero	Trasero	
1	0	0	1
1	0	1	1
1	1	0	a
1	1	1	a
a	0	0	a
a	0	1	a
a	1	0	a
a	1	1	a
2	0	0	2
2	0	1	2
2	1	0	2
2	1	1	2
3	0	0	b
3	0	1	3
3	1	0	3
3	1	1	3
b	0	0	b
b	0	1	b
b	1	0	b
b	1	1	b
4	0	0	4
4	0	1	4
4	1	0	4
4	1	1	4

Tabla 5.1: Conjunto de reglas que define la política a seguir, de acuerdo a los valores de los sensores de obstáculos

Política actual	Próxima política
1	1
a	2
2	3
3	3
b	4
4	1

Tabla 5.2: Conjunto de reglas que define la política a seguir, luego de alcanzar la condición de finalización

a ser utilizada una vez alcanzada la configuración objetivo (aquí las políticas 1 y 3 son las que no se ven alteradas).

5.2. Resultados

A continuación se presentan los resultados asociados con la obtención de las políticas involucradas en este experimento.

5.2.1. Política 1: realizar locomoción mediante una configuración circular

Esta política fue la única política que no fue obtenida mediante aprendizaje. Básicamente, la cantidad de estados posibles aún introduciendo conocimiento previo, excedía los tiempos que uno esperaría para el aprendizaje de un comportamiento. Sin duda, el tratamiento del espacio de estados para el uso de Aprendizaje por Refuerzo en este tipo de robots constituye una línea de investigación futura y necesaria. A continuación se provee un cálculo estimativo del tamaño del espacio de estado-acción, a fin de justificar esta decisión.

Las acciones que un módulo puede realizar se dividen en

1. Activar algún motor: 8 acciones posibles
2. Acoplarse con otro módulo: 3 acciones posibles (por cada modulo restante)
3. Desacoplarse de otro módulo: 3 acciones posibles

Entonces, si el robot está compuesto por 5 módulos, la cantidad total de acciones resulta ser

$$(8 + 3 * 4 + 3)^5 = 23^5 = 6436343$$

Naturalmente, este número es la cota superior, ya que en el problema particular planteado, no es necesario desacoplarse ni acoplarse. Por lo tanto, gracias al conocimiento previo sobre el problema, es posible eliminar dichas acciones, por lo que en la práctica este número queda reducido a

$$8^5 = 32768$$

Ahora bien, en este contexto, el estado de un módulo está formado por

- 2 motores, con 3 valores por motor
- 2 sensores, con 2 valores por sensor

Por lo tanto, la cantidad de estados en que se puede encontrar un módulo es

$$2 * 2 * 3 * 3 = 36$$

y con ello, la cantidad total de estados posibles para el robot es de

$$36^5 = 60466176$$

Hay que tener en cuenta que aquí se ha utilizado conocimiento previo para disminuir el tamaño del espacio de estados, dado que no se contemplan los acoplamientos posibles entre los distintos módulos, ya que se determinó que no es necesario acoplar o desacoplar módulos para cumplir con el objetivo. También es verdad que en realidad este último valor es una cota superior de la cantidad de estados efectivamente posibles, dada la configuración en la que se encuentra el robot. Puesto que el robot se encuentra la mayor parte del tiempo en la misma posición relativa a la fuente emisora, una cota más realista sería

$$(2 * 3 * 3)^5 = 1889568$$

Ahora bien, mediante estos valores calculados, podemos determinar que el tamaño aproximado del espacio de estado-acción es de

$$1889568 * 32768 = 6{,}19 * 10^{10}$$

valores posibles.

Asumiendo que se puede realizar una acción por segundo, entonces para recorrer el espacio de estado-acción completo harán falta

$$\begin{aligned} 6{,}19 * 10^{10} \text{ segundos} &\approx 1031956070 \text{ minutos} \\ &\approx 17199268 \text{ horas} \\ &\approx 716637 \text{ días} \\ &\approx 23888 \text{ meses} \\ &\approx 1990 \text{ años} \end{aligned}$$

Claramente el problema es complejo. Dado que para garantizar que el algoritmo de aprendizaje converja a la política óptima es necesario que todos

los estados sean visitados continuamente, este problema no puede ser tratado directamente. Para poder aprender una política que resuelva este problema será necesario acotarlo aún más, a fin de reducir el tamaño del espacio de estado-acción, o bien definir una representación más conveniente para estados y acciones. Como el objetivo de este experimento no consistía en encontrar la política óptima para el problema de locomoción en un robot constituido por 5 módulos en configuración circular, se optó por generar una política de forma manual, que si bien posiblemente no sea óptima, resuelve correctamente el problema.

5.2.2. Política a: pasar de los estados finales de la política 1 al estado inicial de la política 2

Para desarrollar esta política, se intentó un enfoque iterativo. Para simplificar el aprendizaje se decidió guiar la evolución de la política. Para ello se realizaron episodios de aprendizaje que consistían en aprender a revertir las acciones realizadas por la política anterior. Llamemos al estado inicial de la política 2 s_0, y llamemos a los posibles estados finales de la política 1 $s_1, s_2, \ldots$. Entonces, se comenzó aprendiendo una política para pasar del estado s_1 al estado s_0. Una vez aprendido esto, se modificó el objetivo del experimento para que se alterara la política aprendida de forma de aprender a pasar de s_2 a s_1. De la misma forma se hizo con el resto de los estados posibles (de s_3 a s_2, de s_4 a s_3, ...). Como resultado, se obtuvo una política capaz de alcanzar el estado s_0 desde cualquier estado final s_i.

De esta manera se pudo mantener acotada la complejidad del problema y obtener una política que resolvía el problema planteado de forma adecuada.

En esta ocasión, dado que cada episodio de aprendizaje constituía un problema distinto, y por ello independiente, no tiene sentido calcular la "Tendencia global" como en el resto de los experimentos realizados. Debido a esto, en las figuras 5.1 y 5.2 solamente se indican la cantidad de simulaciones y acciones requeridas por cada episodio de aprendizaje.

5.2.3. Política 2: reconfigurarse para pasar de una configuración circular a una configuración lineal

En este caso, se realizaron 7 episodios de aprendizaje, compuestos por un total de 332 simulaciones, en el transcurso de las cuales se realizaron 3038 acciones.

En este caso, el problema planteado resultó ser suficientemente simple como para no tener que reducir el espacio de estado-acción.

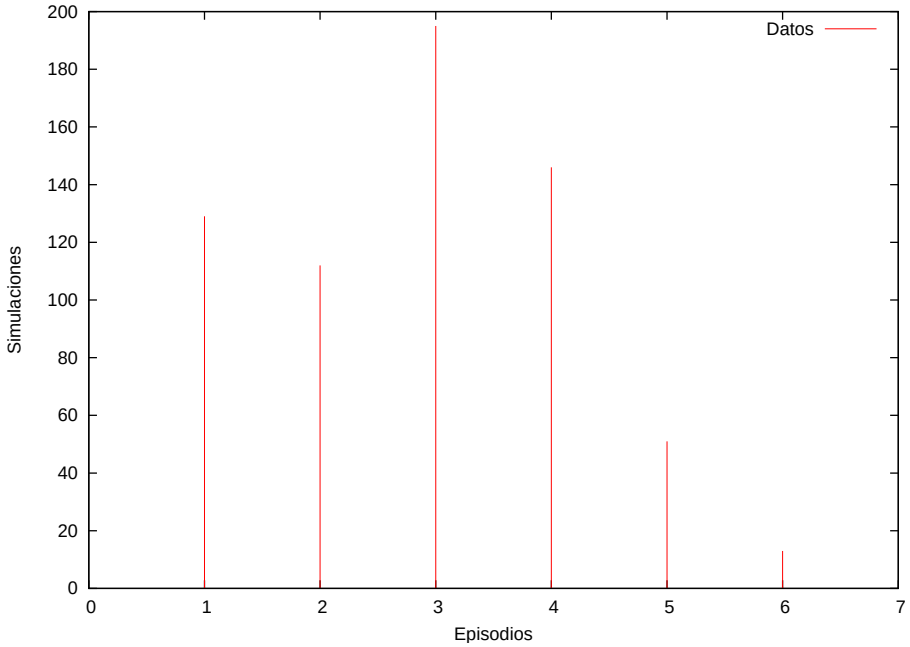

Figura 5.1: Cantidad de simulaciones que conforman cada episodio de aprendizaje de la política a.

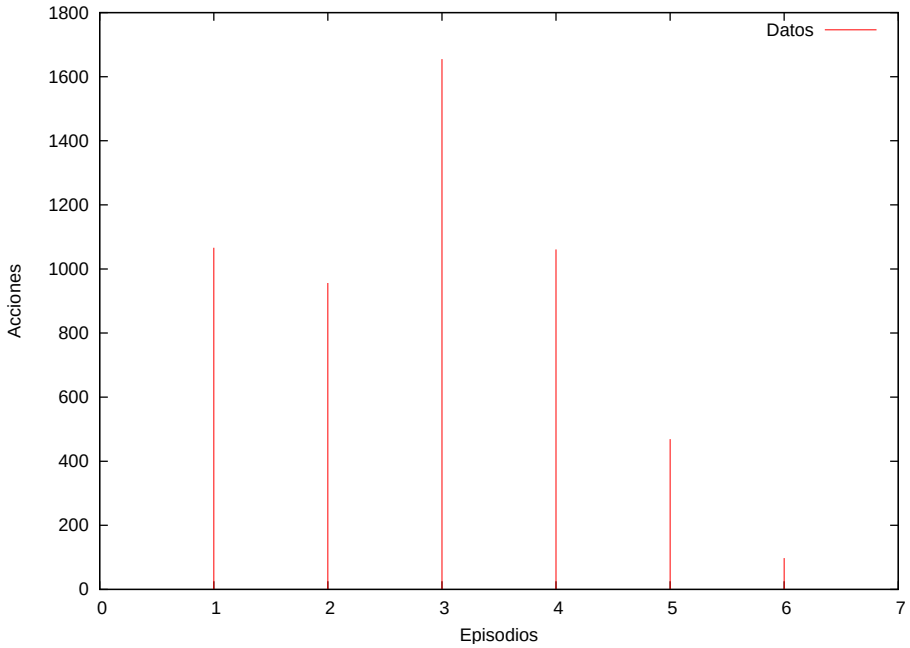

Figura 5.2: Cantidad de acciones realizadas durante cada episodio de aprendizaje de la política a.

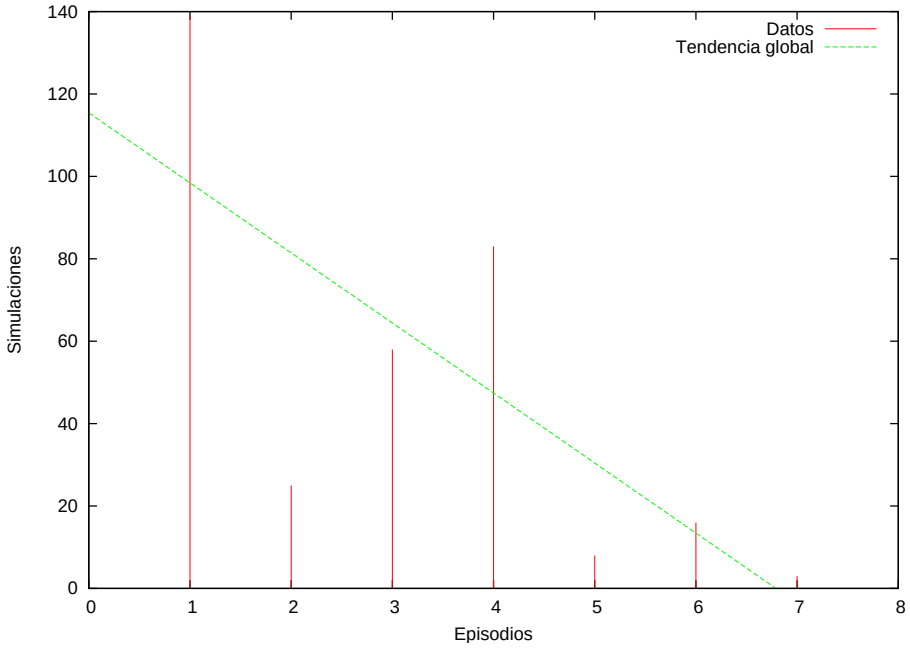

Figura 5.3: Cantidad de simulaciones requeridas por cada episodio de aprendizaje de la política *2*.

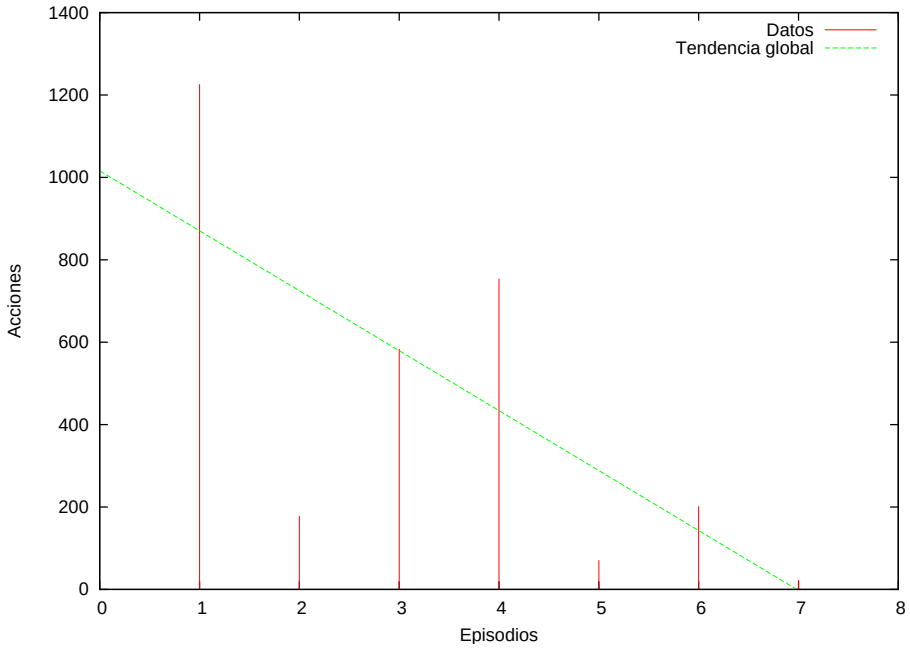

Figura 5.4: Cantidad de acciones realizadas durante cada episodio de aprendizaje de la política *2*.

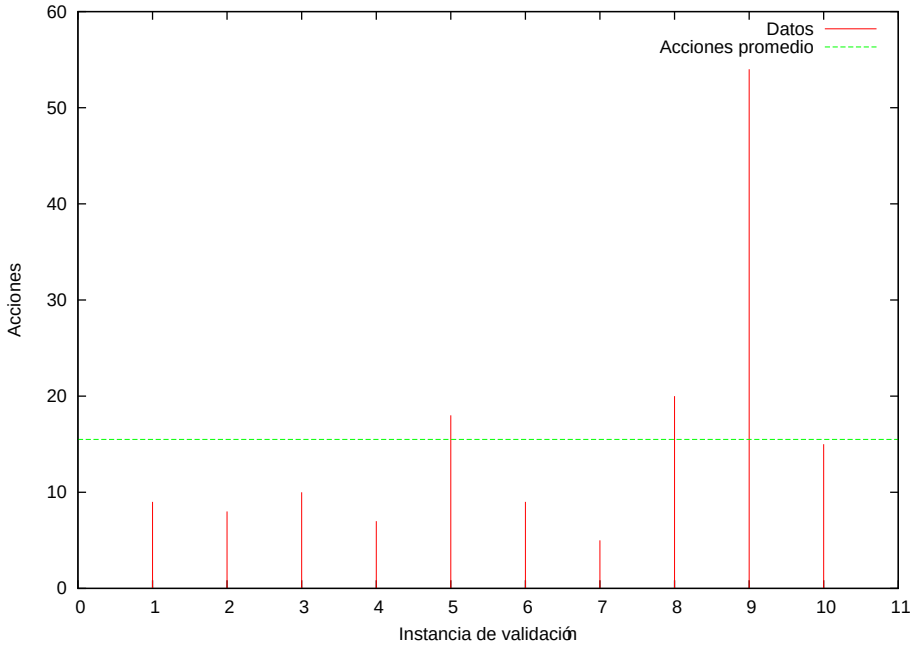

Figura 5.5: Cantidad de acciones realizadas por la política *2* antes de alcanzar el objetivo, durante la fase de validación, luego de 3 episodios de aprendizaje.

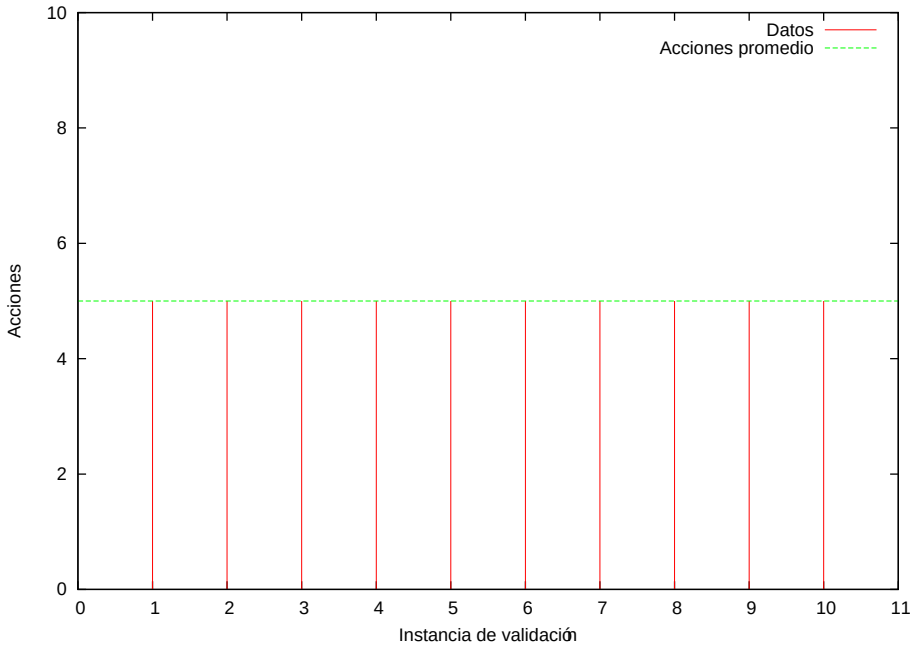

Figura 5.6: Cantidad de acciones realizadas por la política *2* antes de alcanzar el objetivo, durante la fase de validación, luego de 7 episodios de aprendizaje.

Como se puede observar en las figuras 5.3 - 5.6, al final de los episodios de aprendizaje, se obtuvo una política eficiente para la reconfiguración pretendida.

5.2.4. Política 3: realizar locomoción mediante una configuración lineal

Para la siguiente política se realizaron 13 episodios de aprendizaje, conformados por 16 simulaciones, que comprendieron un total de 2243 acciones. En este caso, debido al gran tamaño del espacio de estado-acción, fue necesario realizar una reducción del mismo, simplificando el problema. Las acciones se limitaron de manera que solamente fuera posible activar 2 motores: el motor del bloque pasivo del módulo más cercano a la fuente emisora, y el motor del bloque activo del módulo más lejano. Estos bloques corresponden a los extremos del robot.

Esta decisión se basó en que se pudo determinar de forma manual una política de locomoción que utilizara únicamente estos motores. Luego, se procedió a aprender una política que produjera un comportamiento similar al generado manualmente. De esta forma, se obtuvo una política por medio del aprendizaje, que lograba realizar locomoción en una configuración lineal. Es esperable que la política generada no sea la óptima en cuanto a las posibilidades de locomoción del robot en su totalidad (es decir, si no se hubieran restringido las acciones posibles), pero aún así, resultó suficientemente buena dentro del marco de este experimento.

5.2.5. Política b: pasar de los estados finales de la política 3 al estado inicial de la política 4

Para determinar una política correcta para el problema de adaptar la política 3 a la política 4 (caso descripto como b. anteriormente), fueron necesarios 36 episodios de aprendizaje. En esta ocasión se requirió únicamente una simulación por episodio y se realizaron un total de 170 acciones.

En esta ocasión se procedió de manera similar al caso de la política a. La política aprendida se construyó incrementalmente, pero en este caso, en vez de guiar la solución, se procedió a ampliar la política evaluando la reconfiguración entre cada posible estado final de la política 3 y el estado inicial de la política 4 directamente. Esta distinción se debe a que el problema planteado en este caso resulta ser mas simple que en el caso anterior.

Esto último se puede apreciar en los resultados obtenidos durante el aprendizaje, en donde en cada episodio de aprendizaje se realizaron 6 ac-

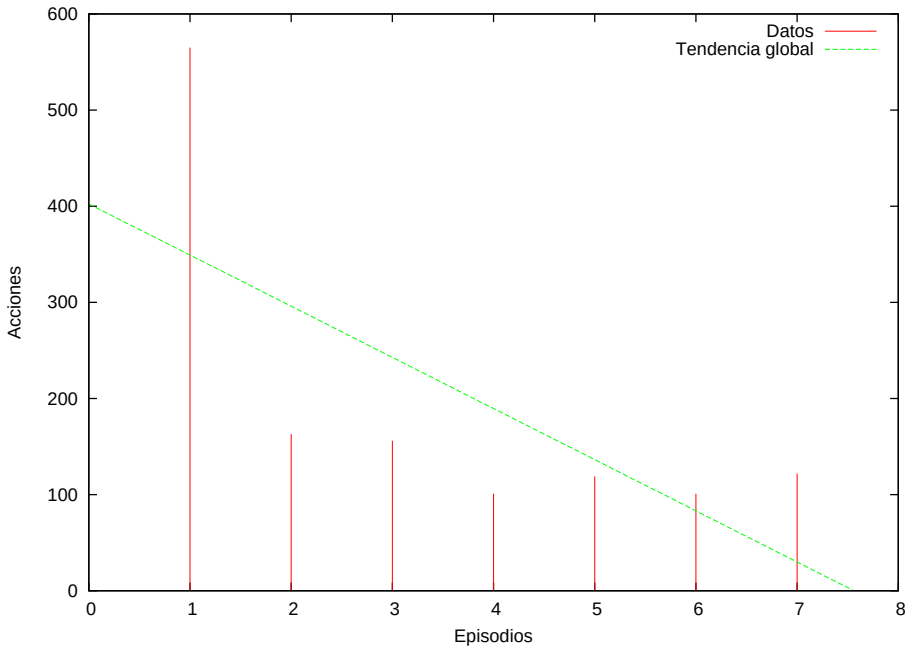

Figura 5.7: Cantidad de acciones realizadas antes de alcanzar el objetivo, durante la fase de aprendizaje de 7 episodios, de la política *3*.

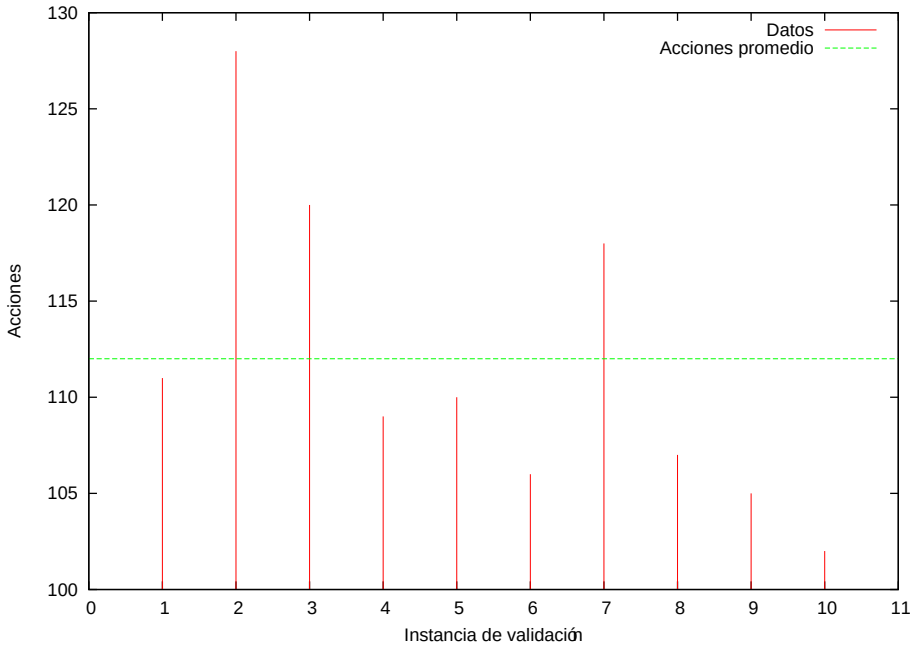

Figura 5.8: Cantidad de acciones realizadas antes de alcanzar el objetivo, durante la fase de validación de la política *3*, luego de 7 episodios de aprendizaje.

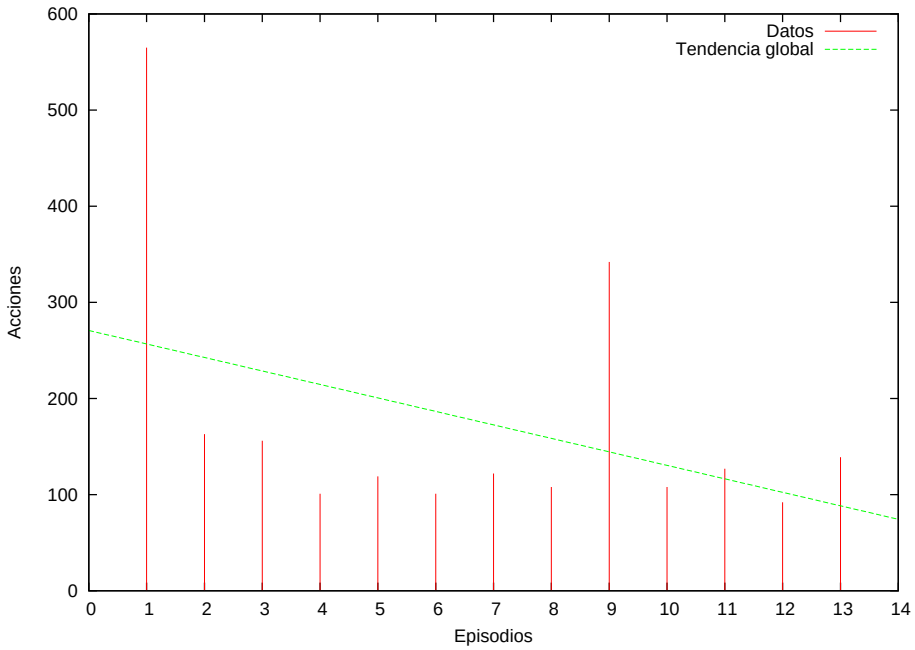

Figura 5.9: Cantidad de acciones realizadas antes de alcanzar el objetivo, durante la fase de aprendizaje de 13 episodios, de la política *3*.

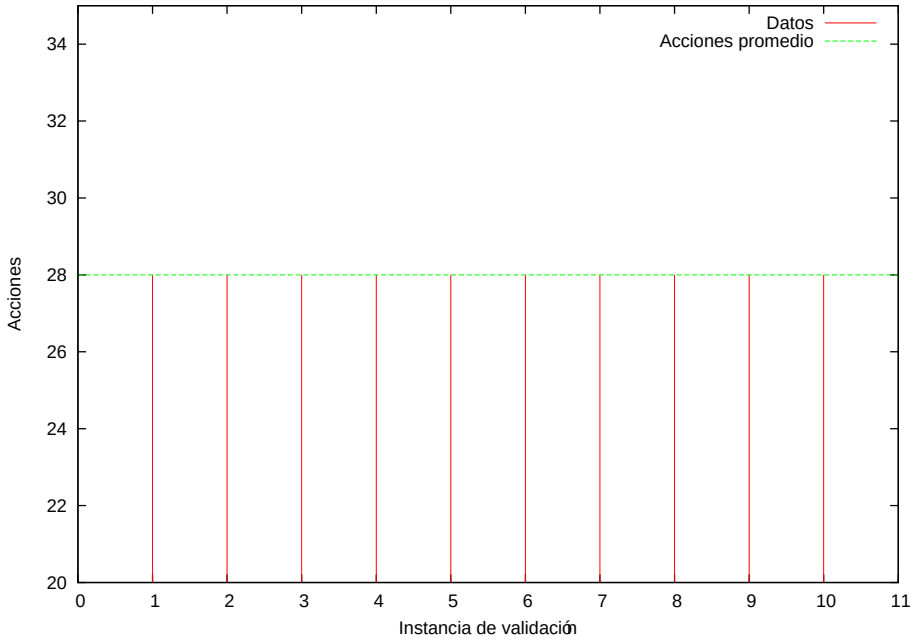

Figura 5.10: Cantidad de acciones realizadas antes de alcanzar el objetivo, durante la fase de validación de la política *3*, luego de 13 episodios de aprendizaje.

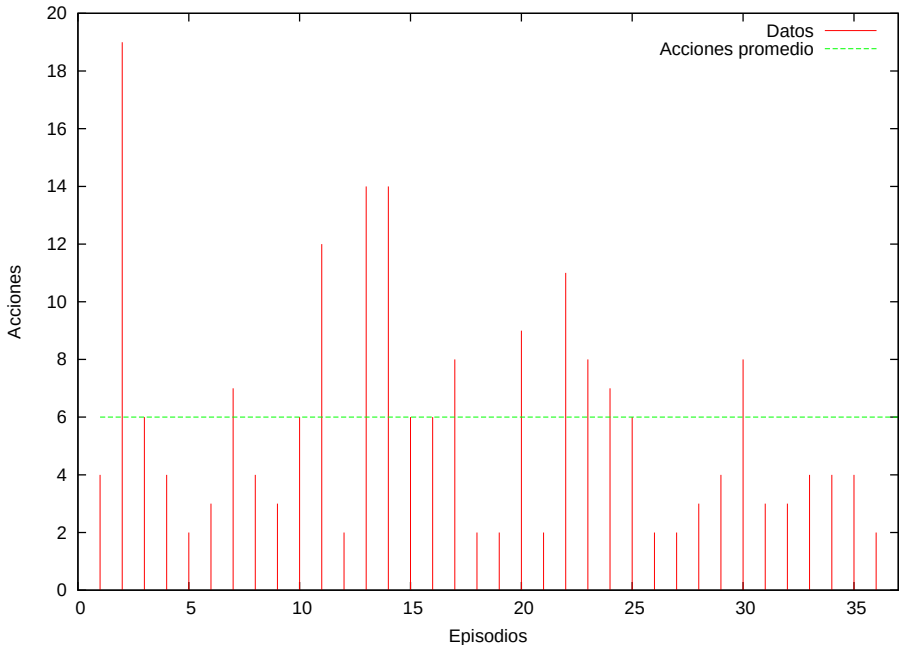

Figura 5.11: Cantidad de acciones realizadas durante cada episodio de aprendizaje de la política *b*.

ciones en promedio.

5.2.6. Política 4: reconfigurarse para pasar de una configuración lineal a una configuración circular

Finalmente, para la última política se realizaron 3 episodios de aprendizaje, sumando un total de 126 simulaciones que comprendieron la evaluación de 3207 acciones.

En este caso, la política fue obtenida velozmente y eficientemente, lo que se refleja en la poca cantidad de episodios necesarios para aprender la misma, y la poca cantidad de acciones requeridas por la política para alcanzar la configuración objetivo.

5.3. Conclusiones

A pesar de los inconvenientes en el aprendizaje de algunas políticas, se pudo ver que era posible combinar políticas simples aprendidas de forma independiente, y obtener como resultado un comportamiento complejo.

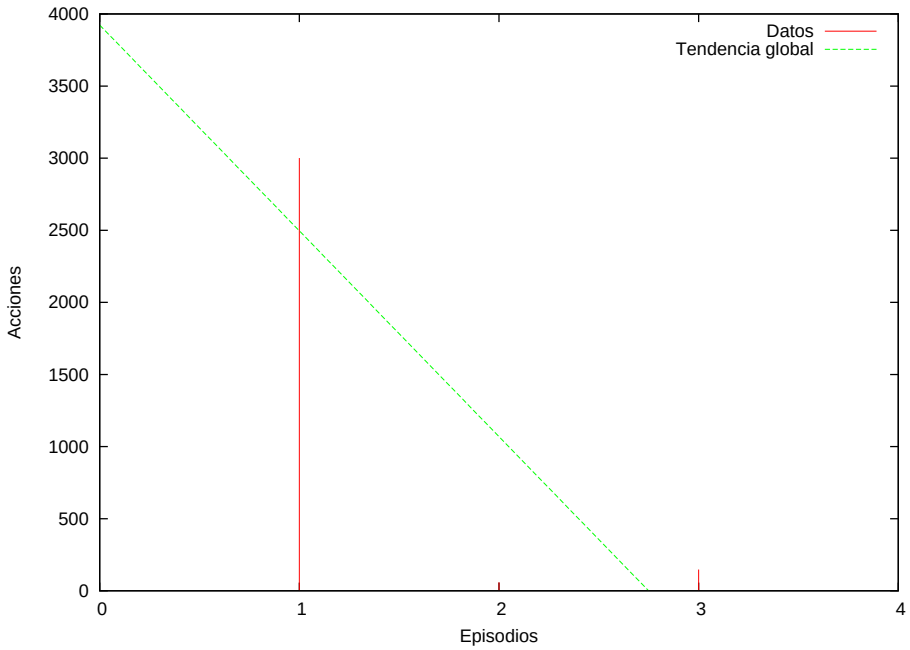

Figura 5.12: Cantidad de acciones realizadas antes de alcanzar el objetivo, durante la fase de aprendizaje de 3 episodios, de la política *4*.

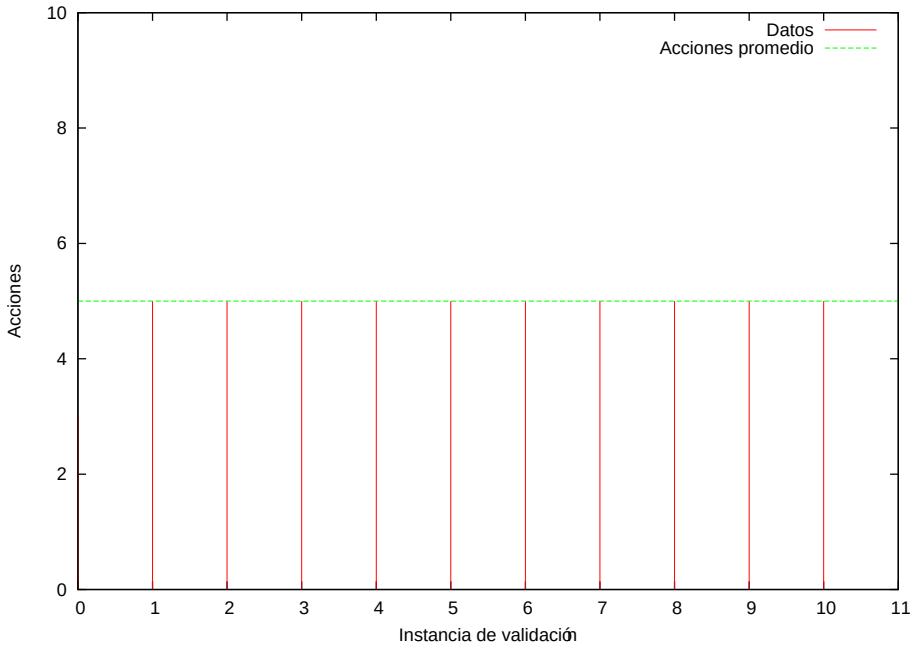

Figura 5.13: Cantidad de acciones realizadas antes de alcanzar el objetivo, durante la fase de validación de la política *4*, luego de 3 episodios de aprendizaje.

Se detectó un inconveniente debido a que las políticas inicialmente desarrolladas dependían fuertemente del estado inicial utilizado. Este inconveniente es fácilmente evitable si al desarrollar las políticas se tienen en cuenta todos los estados iniciales posibles (como se hizo luego con las políticas que fue necesario agregar para corregir el error), por lo que no representa un problema realmente.

Si bien no todas las políticas utilizadas fueron aprendidas, esto tampoco representa un problema, dado que

1. es posible aprender dichas políticas, restringiendo el tamaño del espacio estado-acción correctamente. En esta ocasión no se consideró relevante disponer de la extensa cantidad de tiempo requerido para lograr aprender estas políticas más complejas, dado que la forma de obtenerlas no alteran el resultado del experimento.

2. el mecanismo utilizado para combinar políticas no tiene en cuenta de qué manera fueron adquiridas las mismas, por lo que permite combinar políticas aprendidas, evolucionadas, generadas a mano, o adquiridas de cualquier otra forma.

Este último hecho puede considerarse como una característica positiva del método, dado que en ciertas ocasiones no es vital disponer de la política por medio del aprendizaje, y puede ser mejor obtener la política de otra forma; por ejemplo, es posible obtener una política óptima de forma manual, siempre y cuando los comportamientos que se desean reproducir sean lo suficientemente simples. En estos casos, puede resultar más beneficioso determinar la política de forma manual, y dedicar el tiempo de cómputo requerido para el aprendizaje de la misma a la generación de una política para un comportamiento más complejo.

Como conclusión final, queda claro que es perfectamente posible utilizar políticas asociadas a comportamientos simples y mediante su combinación, producir como resultado una política que describe un comportamiento más complejo. Este hecho resulta beneficioso, dado que obtener la política para el comportamiento global directamente puede resultar muy difícil (tanto si se intenta definirla manualmente como si se intenta un enfoque computacional, como el aprendizaje o la evolución).

Capítulo 6

Conclusiones

En el presente trabajo se ha presentado un panorama sobre el estado del arte en el campo de los Robots Modulares Autoreconfigurables y se ha desarrollado un simulador con el cual se han realizado experimentos relativos a la utilización de la metodología del Aprendizaje por Refuerzo aplicada al problema de obtener comportamientos que permitan realizar tareas de locomoción y reconfiguración en Robots Modulares Autoreconfigurables. Para cada tipo de comportamiento a ser aprendido, debió estudiarse la representación adecuada del espacio estado-acción y definir las condiciones para las cuáles el Aprendizaje por Refuerzo era factible. El análisis realizado constituyó un primer acercamiento al problema tratado en el marco de este trabajo, del cual fue posible extraer algunas de las características inherentes al problema tratado y determinar los factores a tener en cuenta para continuar la labor en este campo. Finalmente se ha propuesto un problema que combinaba tareas de locomoción así como de reconfiguración, y se lo ha resuelto mediante la metodología estudiada.

Durante el análisis del estado del arte en el campo de los Robots Modulares Autoreconfigurables se pudo ver que efectivamente, en la bibliografía no se ha visto tratada la utilización del Aprendizaje por Refuerzo como metodología para resolver tareas de locomoción y reconfiguración en este tipo de robots.

A través de los experimentos realizados se pudo verificar que el simulador desarrollado cumplía con los requisitos para poder ser utilizado en experimentos relacionados con la aplicación del Aprendizaje por Refuerzo a la obtención de comportamientos adecuados para las tareas de locomoción y reconfiguración. A su vez, se mostró que es posible aprender políticas eficientes para la realización de locomoción y reconfiguración en Robots Modulares Autoreconfigurables (por lo menos en el caso del robot M-TRAN).

En la mayoría de los experimentos realizados se obtuvieron resultados

exitosos. En el resto de los casos, se observó que el cardinal del espacio de estado-acción constituía un obstáculo importante para obtener sub-óptimos aceptables y sin duda un tratamiento más general constituye una línea de trabajo futuro.

Mediante los experimentos realizados se demostró que es factible utilizar la metodología del Aprendizaje por Refuerzo para aprender políticas eficientes. También quedó demostrado que es posible componer políticas simples aprendidas independientemente, y conseguir como resultado un comportamiento complejo. Este último hecho es de gran importancia, ya que gracias a esto no resulta indispensable resolver las tareas más complejas de forma directa, lo que puede resultar arduo debido al gran tamaño del espacio de estado-acción asociado con los problemas mas complejos; por el contrario, es posible dividir un problema complejo en subproblemas más sencillos de resolver, aprender una política adecuada para cada subproblema y luego combinar dichas políticas a fin de resolver el problema global.

Queda claro entonces, que la metodología propuesta puede ser útil en este área, y que efectivamente resulta necesario ampliar la investigación realizada en este contexto, a fin de determinar mejores maneras de permitir a estos robots adecuarse a cambios del entorno y con ello, explotar al máximo sus capacidades intrínsecas.

Algunas posibles líneas de trabajo futuras incluyen:

- Estudiar la manera de reducir el tamaño del espacio de estado-acción, para poder trabajar con robots de mayor tamaño, y/o con problemas más complejos.
- Generalizar la experimentación realizada, para poder trabajar con robots cuyas estructuras sean más complejas.
- Determinar formas de aprender la tabla de combinación de políticas, en vez de tener que especificarla manualmente.
- Estudiar las formas de aprender a adaptarse a condiciones cambiantes en el entorno.
- Estudiar las formas de aprender a adaptarse a capacidades de percepción diferentes (agregar/quitar sensores).

Apéndice A

Comparación de Robots Modulares Reconfigurables

	DOF	Homogéneo	3D	Tipo	Auto-R.	Acoplamiento
CONRO	2	✓	✓	Cadena	✓	Mecánico/Bipartito
Polybot	1	✓	✓	Cadena	✓	Mecánico/Bipartito
ACM	1-3	✓	✓	Cadena	×	Mecánico
Metamorphic	3	✓	×	Malla	✓	Mecánico/Bipartito
Crystalline	2	✓	✓	Malla	✓	Mecánico/Bipartito
Fractum	0	✓	×	Malla	✓	Magnético/Bipartito
Micro Unit	2	✓	×	Malla	✓	Mecánico/Bipartito
RIKEN Vertical	2	✓	×	Malla	✓	Magnético/Bipartito
Telecube	6	✓	✓	Malla	✓	Magnético/Bipartito
MEL 3D Unit	12	✓	✓	Malla	✓	Mecánico/Bipartito
Molecule	4	✓	✓	Malla	✓	Mecánico/Bipartito
M-TRAN (I y II)	2	✓	✓	Híbrido	✓	Magnético/Bipartito
M-TRAN III	2	✓	✓	Híbrido	✓	Mecánico/Bipartito
I(ces)-Cubes	3	×	✓	Malla	✓	Mecánico/Bipartito
Fracta	12	✓	✓	Malla	✓	Mecánico/Hermafrodita
Proteo	0	✓	✓	Malla	✓	Magnético/Bipartito
Miniaturized	2	✓	×	Malla	✓	Mecánico/Bipartito
Semi-Cylindrical	2	✓	✓	Malla	✓	Magnético/Bipartito
TETROBOT	3-5	✓	✓	Malla	×	Mecánico/Bipartito
CEBOT	1-3	✓	×	Malla	×	Bipartito

Tabla A.1: Comparación de Robots Modulares Reconfigurables

Apéndice B

Resultados experimentales

Parámetros			Distancia Avanzada	Distancia Recorrida	Acciones Realizadas	Completó el aprendizaje
alpha	épsilon	gamma				
0.1	0.5	0.5	22.55	168.65	172	✓
0.2	0.5	0.5	24.60	181.30	190	✓
0.3	0.5	0.5	23.40	180.70	181	✓
0.4	0.5	0.5	18.85	188.80	192	✓
0.5	0.5	0.5	22.20	191.40	199	✓
0.6	0.5	0.5	25.80	174.60	181	✓
0.7	0.5	0.5	23.25	172.75	178	✓
0.8	0.5	0.5	22.55	179.45	183	✓
0.9	0.5	0.5	19.40	186.10	189	✓

Tabla B.1: Resultados obtenidos durante la fase de aprendizaje de la tarea de locomoción propuesta en la sección 4.1, en función de distintos valores para el parámetro *alpha*

Parámetros			Distancia Avanzada	Distancia Recorrida	Acciones Realizadas	Completó el aprendizaje
alpha	épsilon	gamma				
0.5	0.1	0.5	29.80	73.50	78	✓
0.5	0.2	0.5	29.75	104.10	105	✓
0.5	0.3	0.5	29.55	122.05	125	✓
0.5	0.4	0.5	29.65	143.05	142	✓
0.5	0.5	0.5	22.35	179.45	181	✓
0.5	0.6	0.5	14.30	194.50	201	×
0.5	0.7	0.5	12.10	196.95	200	×
0.5	0.8	0.5	4.90	188.10	201	×
0.5	0.9	0.5	5.90	176.55	201	×

Tabla B.2: Resultados obtenidos durante la fase de aprendizaje de la tarea de locomoción propuesta en la sección 4.1, en función de distintos valores para el parámetro *épsilon*

Parámetros			Distancia Avanzada	Distancia Recorrida	Acciones Realizadas	Completó el aprendizaje
alpha	épsilon	gamma				
0.5	0.5	0.1	26.55	166.05	173	✓
0.5	0.5	0.2	24.55	169.10	179	✓
0.5	0.5	0.3	25.65	171.10	177	✓
0.5	0.5	0.4	22.85	183.15	187	✓
0.5	0.5	0.5	23.15	184.35	191	✓
0.5	0.5	0.6	29.35	162.75	162	✓
0.5	0.5	0.7	26.75	180.35	177	✓
0.5	0.5	0.8	23.85	168.30	170	✓
0.5	0.5	0.9	23.40	176.40	181	✓

Tabla B.3: Resultados obtenidos durante la fase de aprendizaje de la tarea de locomoción propuesta en la sección 4.1, en función de distintos valores para el parámetro *gamma*

Parámetros			Distancia	Distancia	Acciones	Performance
alpha	épsilon	gamma	Avanzada	Recorrida	Realizadas	
0.1	0.5	0.5	29.50	70.50	63	0.23247
0.2	0.5	0.5	29.50	70.50	63	0.23247
0.3	0.5	0.5	22.50	65.50	148	0.12850*
0.4	0.5	0.5	29.50	71.00	63	0.22962
0.5	0.5	0.5	29.50	71.50	63	0.22962
0.6	0.5	0.5	26.50	76.00	137	0.15608*
0.7	0.5	0.5	29.50	75.70	62	0.21981
0.8	0.5	0.5	29.50	70.50	63	0.23247
0.9	0.5	0.5	29.50	70.50	63	0.23247
0.5	0.1	0.5	29.50	63.50	62	0.26225
0.5	0.2	0.5	29.50	73.00	79	0.19539
0.5	0.3	0.5	29.50	78.00	102	0.19117*
0.5	0.4	0.5	27.00	90.00	114	0.15824*
0.5	0.5	0.5	29.50	70.50	63	0.23247
0.5	0.6	0.5	29.50	70.50	63	0.23247
0.5	0.7	0.5	29.50	62.00	56	0.35310
0.5	0.8	0.5	29.50	70.50	63	0.23246
0.5	0.9	0.5	29.50	74.50	59	0.23490
0.5	0.5	0.1	29.50	70.50	63	0.23247
0.5	0.5	0.2	29.50	70.50	63	0.23247
0.5	0.5	0.3	29.50	70.50	63	0.23247
0.5	0.5	0.4	29.50	75.50	62	0.21988
0.5	0.5	0.5	29.50	71.00	63	0.22884
0.5	0.5	0.6	29.50	57.00	98	0.30714*
0.5	0.5	0.7	29.50	75.50	62	0.22057
0.5	0.5	0.8	29.50	72.00	66	0.22132
0.5	0.5	0.9	29.50	83.50	88	0.18831*

* Al utilizar éstos parámetros, algunas simulaciones fueron abortadas debido al límite máximo de acciones permitidas.

Tabla B.4: Resultados obtenidos durante la fase de evaluación de las políticas aprendidas para la tarea de locomoción propuesta en la sección 4.1

Apéndice C

Imágenes del robot M-TRAN II y M-TRAN III

C.1. M-TRAN II

Figura C.1: Locomoción en un terreno sencillo.

Figura C.2: Reconfiguración a fin de superar un obstáculo en el terreno.

Figura C.3: Reconfiguración para adquirir altura (escalar).

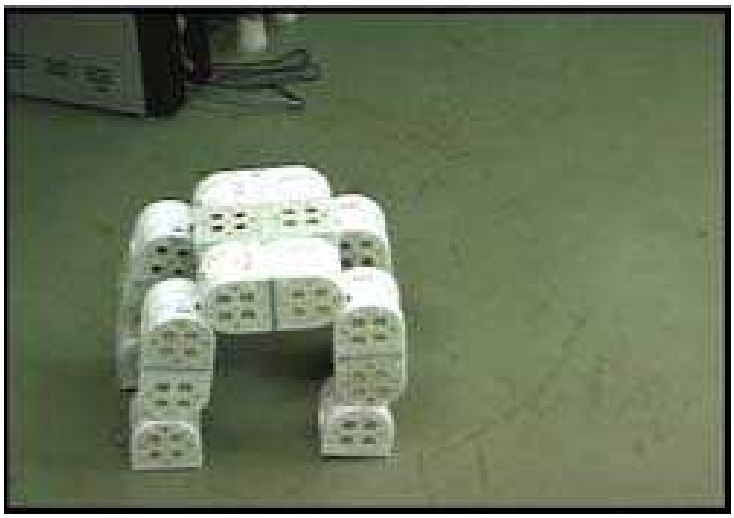

Figura C.4: Cuadrúpedo

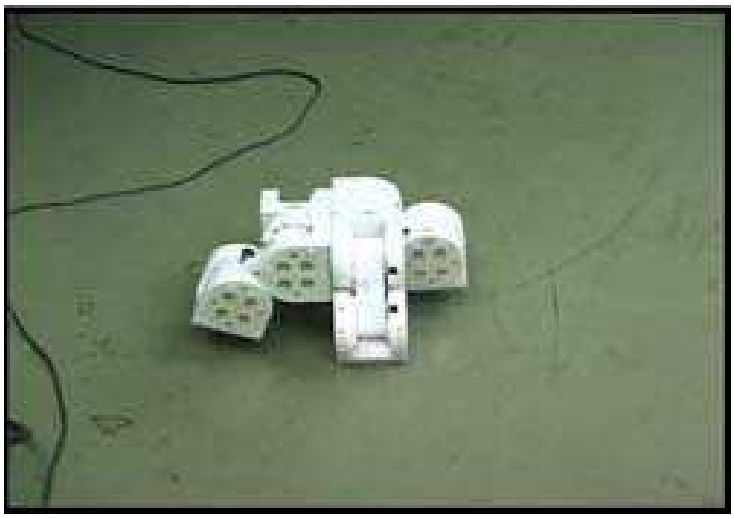

Figura C.5: Versión mínima de un robot cuadrúpedo.

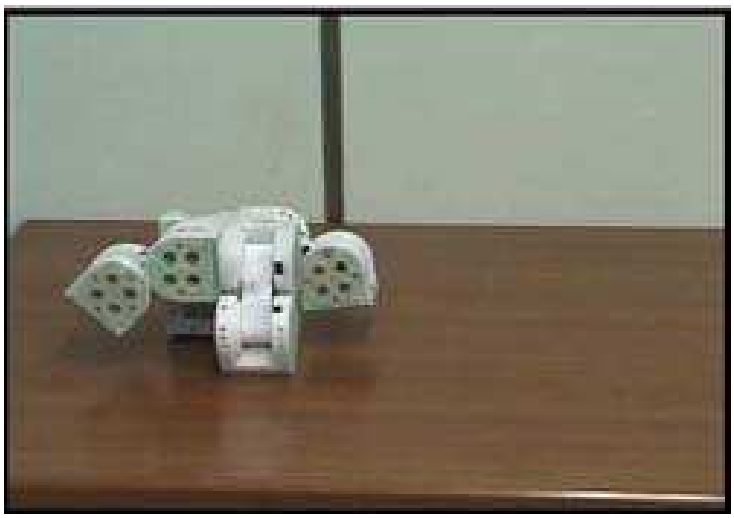

Figura C.6: Versión mínima de un robot cuadrúpedo.

Figura C.7: Configuración para poder realizar locomoción en forma de oruga.

Figura C.8: Configuración para poder realizar locomoción en forma de oruga.

Figura C.9: Configuración para locomoción hexápoda.

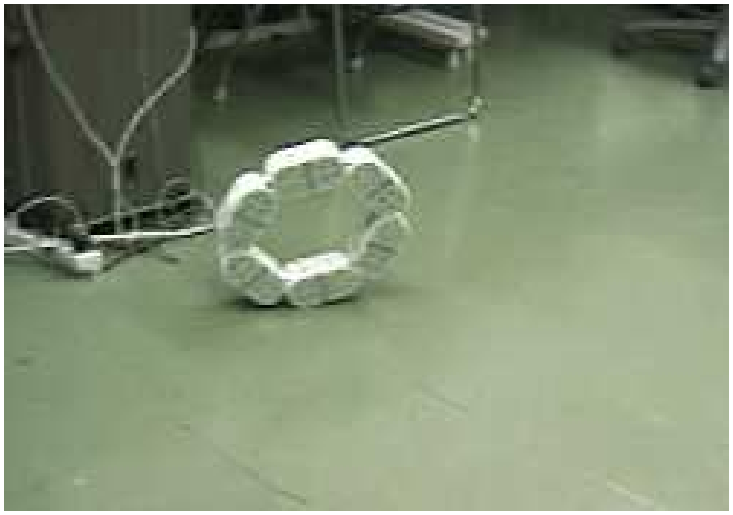

Figura C.10: Configuración para locomoción circular (rueda).

Figura C.11: Mediante la reconfiguración es posible superar obstáculos.

Figura C.12: Locomoción cuadrúpeda.

Figura C.13: Configuración para realizar locomoción similar a la de las serpientes.

Figura C.14: Configuración arácnida.

C.2. M-TRAN III

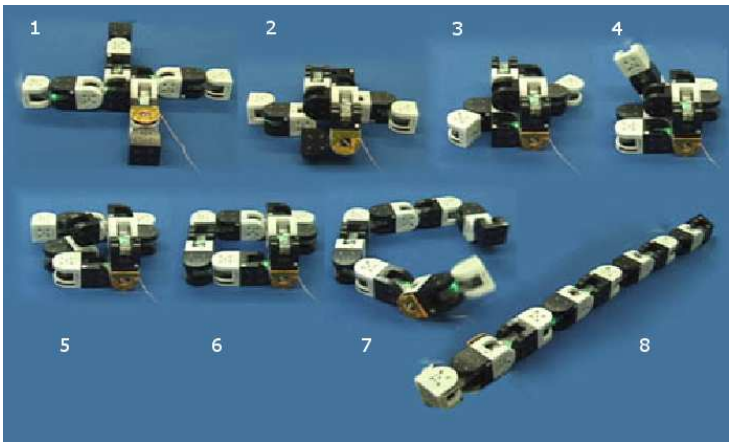

Figura C.15: Reconfiguración de forma cuadrúpeda a lineal (8 módulos).

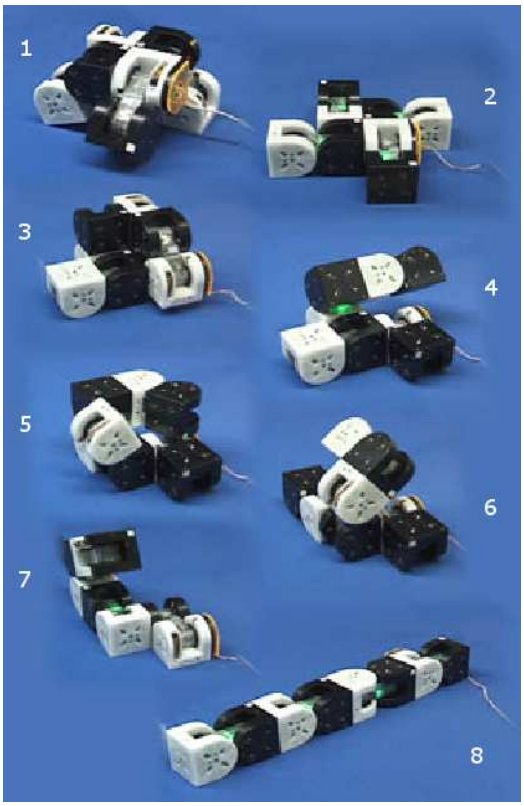

Figura C.16: Reconfiguración de forma cuadrúpeda a lineal (4 módulos).

Bibliografía

[Bax97] Baxter, J., Tridgell, A., y Weaver, L., "Knightcap: A chess program that learns by combining $td(\lambda)$ with minimax search", 1997.

[But02] Butler, Z., Kotay, K., Rus, D., y Tomita, K., "Generic decentralized control for a class of self-reconfigurable robots", *Proc. of IEEE IRCA*, págs. 809–815, 2002.

[Cas02] Castano, A., Behar, A., y Will, P., "The Conro Modules for Reconfigurable Robots", *IEEE/ASME TRANSACTIONS ON MECHATRONICS*, 7(4), 2002.

[Cri98] Crites, R. H. y Barto, A. G., "Elevator group control using multiple reinforcement learning agents", 1998.

[Dit04] Dittrich, E., "Modular Robot Unit - Characterisation, Design and Realisation", 2004.

[Jan01] Jantapremjit, P. y Austin, D., "Design of a Modular Self-Reconfigurable Robot", *Proceedings of the 2001 Australian Conference on Robots and Automation*, págs. 38–43, 2001.

[Jor04] Jorgensen, M. W., Ostergaard, E. H., y Lund, H. H., "Modular ATRON: Modules for a self-reconfigurable robot", *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[Kam03] Kamimura, A., Kurokawa, H., Yoshida, E., Tomita, K., Murata, S., y Kokaji, S., "Automatic Locomotion Pattern Generation for Modular Robots", *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, 2003.

[Kot98] Kotay, K., Rus, D., Vona, M., y McGray, C., "The Self-reconfiguring Robotic Molecule: Design and Control Algorithms", *International Conference on Robotics and Automation*, 1998.

[Kum98] Kumar, G. P. y Venkataram, P., "Network performance tuning using reinforcement learning", 1998.

[Kur02] Kurokawa, H., "Self-reconfigurable Modular Robot (M-TRAN) and its Motion Design", *Proc. ICARCV 2002*, págs. 51–56, 2002.

[Mur94] Murata, S., Kurokawa, H., y Kokaji, S., "Self-Assembling Machine", *Proc. IEEE Int. Conf. on Robotics and Automation*, págs. 441–448, 1994.

[Mur02] Murata, S., Yoshida, E., Kamimura, A., Kurokawa, H., Tomita, K., y Kokaji, S., "M-TRAN: Self-Reconfigurable Modular Robotic System", *IEEE/ASME Transactions on Mechatronics*, 7(4):431–441, 2002.

[Nil02] Nilsson, M., "Connectors for Self-Reconfiguring Robots", *IEEE/ASME Transactions on Mechatronics*, 7(4):473–474, 2002.

[Nol98] Nolfi, S. y Floreano, F., "How co-evolution can enhance the adaptive power of artificial evolution: implications for evolutionary robotics", *Proceedings of EvoRobot'98*, págs. 2–8, 1998.

[ODE] "Open Dynamics Engine", `http://www.ode.org/`.

[Pat04] Patterson, S. A., Knowles, K. A. J., y Bishop, B. E., "Towards Magnetically-Coupled Reconfigurable Modular Robots", *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, 2004.

[Rou00] Roufas, K., Zhang, Y., Duff, D., y Yim, M., "Six Degree of Freedom Sensing for Docking Using IR LED Emitters and Receivers", 2000.

[Rus00] Rus, D. y Vona, M., "A basis for self-reconfigurable robots using crystal modules", *Proc. IEEE IROS*, págs. 2194–2202, 2000.

[She] Shen, W.-M., Salemi, B., y Will, P., "Hormones for Self-Reconfigurable Robots", URL `http://www.isi.edu/conro/`.

[Sto02] Stoy, K., Shen, W.-M., y Will, P. M., "Using Role-Based Control to Produce Locomotion in Chain-Type Self-Reconfigurable Robots", *IEEE/ASME Transactions on Mechatronics*, 7(4):410–417, 2002.

[Sto03] Stoy, K., Shen, W.-M., y Will, P. M., "Implementing Configuration Dependent Gaits in a Self-Reconfigurable Robot", *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, págs. 3828–3833, 2003.

[Sto04] Stoy, K., "Emergent Control of Self-Reconfigurable robots", 2004.

[Suh02] Suh, J. W., Homans, S. B., y Yim, M., "Telecubes: Mechanical design of a module for self-reconfigurable robotics", *Proceedings on the IEEE International Conference on Robotics and Automation*, págs. 4095–4101, 2002.

[Sut98] Sutton, R. y Barto, A., *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, 1998.

[Tes95] Tesauro, G., "Temporal difference learning and td-gammon", *Communications of the ACM*, 38:58–68, 1995.

[Üns99] Ünsal, C., Kiliççöte, H., y Khosla, P. K., "I(CES)-cubes: a modular self-reconfigurable bipartite robotic system", *Proceedings of SPIE*, 3839:258–269, 1999.

[Wat89] Watkins, C., *Learning from Delayed Rewards*, Tesis Doctoral, King's College, Oxford, 1989.

[Yim00] Yim, M., Duff, D., y Roufas, K., "PolyBot: a Modular Reconfigurable Robot", *Proc. of the IEEE Int. Conf. on Robotics and Automation*, págs. 1734–1741, 2000.

[Yim02] Yim, M., Zhang, Y., y Duff, D., "Modular Robots", *IEEE Spectrum*, págs. 30–34, 2002.

[Yos01a] Yoshida, E., Murata, S., Kamimura, A., Tomita, K., Kurokawa, H., y Kokaji, S., "A motion planning method for a self-reconfigurable modular robot", *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, págs. 590–597, 2001.

[Yos01b] Yoshida, E., Murata, S., Kamimura, A., Tomita, K., Kurokawa, H., y Kokaji, S., "Reconfiguration Planning for a Self-Assembling Modular Robot", *Proceedings of the 4th IEEE International Symposium on Assembly and Task Planning*, 2001.