Proyecto FIbRA

Informe Final

Gustavo Armagno
garmagno@fing.edu.uy

Facundo Benavides
fbenavid@fing.edu.uy

Claudia Rostagnol
crostagnol@gmail.com

Tutores
Gonzalo Tejera, Ernesto Copello

www.fing.edu.uy/inco/grupos/mina/pGrado/FIRA2005.html

26 de noviembre de 2006

Instituto de Computación
Facultad de Ingeniería - Universidad de la República
Montevideo - Uruguay

Agradecimientos

Después de un año y muchos meses, es un poco difícil recordar con nitidez cada detalle de los tantos que hubo a lo largo del camino recorrido. Atrás quedaron largas horas de tertulias -improvisadas al comienzo- para intentar entender de cosas que nunca habíamos oído, las n reuniones, las discusiones interminables, los raptos de delirio, las soluciones hiper-mega-complejas, los recortes para poder llegar, las arquitecturas y los patrones de diseño, los viajes de estudio y las amistades cosechadas y otras tantas experiencias que no olvidaremos. ¡Cuántos cruces de caminos! ¡Cuántas decisiones tomadas! ¡Cuántas equivocadas y otras que creemos acertadas! Dificultades no faltaron, tampoco oportunidades para sortearlas. En fin, un largo periplo que no hemos recorrido solos. Por eso, ahora que hemos llegado al final, queremos agradecer.

En primer término a nuestros tutores, sin cuyos aportes al trabajo y apoyo permanente no hubiéramos concluido esta tarea a satisfacción.

En segundo lugar, a la organización del *Brazil Agent School*, realizado en la Universidad Federal de Río Grande del Sur (UFRGS), por otorgarnos tres becas de estudio que nos permitieron participar del evento. En particular, expresar nuestro especial agradecimiento a Ana Bazzan, Daniel y Eduardo Basso y a Jomi Fred Hübner por su colaboración durante el evento y luego, a la distancia.

En el mismo sentido, por la notable atención que recibimos en Buenos Aires durante la realización del IV Campeonato Argentino de Fútbol de Robots (excepto por la goleada 0x20 de bienvenida), a Gonzalo Zabala, docente de la Universidad Abierta Interamericana (UAI), quien además resultó ser una verdadera "ficha".

Por el apoyo que nunca se hizo rogar y los innumerables y totalmente valorables aportes, vaya nuestro muy cálido agradecimiento a, nuestros antes amigos que ingenieros, Diego, Martín, Natalia y "Charo".

Para finalizar, primero agradecer a los amigos del equipo Forrest ("Serry", "Chonchi" y "Negro") que han recorrido un camino muy similar al nuestro y con los que compartimos e intercambiamos experiencias muy ricas; y por último, pero no menos importante, a todos aquellos amigos (familia incluida) que, comprendiendo nuestras ausencias en momentos complicados, nos alentaron y apoyaron siempre, durante éste que ha sido un largo pero muy disfrutable proyecto de grado.

¡Gracias!

Resumen

Con el advenimiento de nuevas y poderosas tecnologías, el desarrollo de sistemas robóticos y los avances en Inteligencia Artificial se han potenciado fuertemente en las últimas décadas. A su vez, son varias las iniciativas que se han valido del componente lúdico para captar un mayor interés, sobre todo de los estudiantes e investigadores más jóvenes. Actualmente, el fútbol de robots, que explota ese recurso, representa un desafío interesante para desarrollar sistemas sobre él, integrando diferentes propuestas como: arquitecturas cooperativas, teoría de control, navegación, filtrado de información, procesamiento de imágenes y razonamiento en tiempo real, entre otras temáticas.

En el presente trabajo, se describe una solución al problema de construir un equipo de fútbol de robots para la liga simulada, *Middle League SimuroSot*, de la FIRA. En él se presenta parte sustancial de la investigación realizada, así como una descripción de la estructura global del equipo y de los módulos más importantes que lo componen.

El equipo FIbRA se construye en base al trabajo realizado en tres direcciones concretas que atacan los siguientes problemas: reconocimiento y predicción del juego oponente, planificación de movimientos y toma de decisiones. Está desarrollado en *Java* y se utiliza una arquitectura en capas donde se encapsulan: el modelo del mundo -que incluye predicción, detección y monitoreo de distintos aspectos del entorno-, la planificación de los movimientos, el sistema de toma de decisiones y la interacción con otras aplicaciones.

Para la resolución de la predicción se utilizan ideas inspiradas en el comportamiento de sistemas biológicos, particularmente colonias de hormigas, combinadas con algoritmos de *clustering* utilizados en la resolución de problemas relacionados con el procesamiento de imágenes.

La planificación de movimientos se caracteriza por ser heterogénea, integrando una diversidad de controles de movimientos, para lograr una mayor flexibilidad al momento de resolver cómo se mueven los robots en diferentes escenarios. Otra particularidad, es que ninguno de los algoritmos utilizados considera objetivos móviles, resolviéndose este problema mediante su combinación con el uso de predicción de trayectorias. La implementación de algunos controles se basa en funciones de Lyapunov para garantizar estabilidad.

En la resolución de la toma de decisiones se utiliza un enfoque difuso. Los principales objetivos de ésta son facilitar la adaptación del equipo a los cambiantes estados de juego y la asignación dinámica de roles, como formas de potenciar el juego cooperativo. Es un sistema de tipo Mamdani, estructurado en capas, donde se toman decisiones en diferentes niveles: elección de la estrategia de juego, asignación de roles y, finalmente, asignación de acciones. El ciclo de procesamiento presenta una variante frente al de los sistemas difusos clásicos, para mejorar el desempeño a través de optimización de la cantidad de cálculos realizados.

Finalmente, en cuanto a la interacción con otras aplicaciones se definen dos mecanismos: punto de entrada a través de una interfaz que puede ser utilizado por otras aplicaciones *Java* que se ejecuten localmente, e intercambio de mensajes a través de un canal *UDP*, que particularmente se utiliza en la interacción con el simulador oficial de la FIRA.

Palabras clave Fútbol de robots, FIRA, SimuroSot, Inteligencia Artificial, Robótica, Lógica difusa, Reconocimiento de patrones, Sistemas biológicos, *Ants*.

Índice general

1. Introducción **13**
- 1.1. Antecedentes . . . 14
- 1.2. Objetivos del proyecto . . . 14
- 1.3. Organización del documento . . . 14

2. Marco teórico **17**
- 2.1. Reconocimiento de Patrones . . . 17
  - 2.1.1. Modelo . . . 17
  - 2.1.2. Diseño . . . 18
  - 2.1.3. Aprendizaje . . . 18
  - 2.1.4. Reconocimiento de patrones en el fútbol de robots . . . 19
- 2.2. Inspiraciones biológicas . . . 19
  - 2.2.1. Sociedades de hormigas . . . 19
- 2.3. Toma de decisiones . . . 20
  - 2.3.1. Paradigmas . . . 20
  - 2.3.2. Lógica difusa . . . 22
- 2.4. Planificación de movimientos . . . 23
  - 2.4.1. Planificación de caminos . . . 24
  - 2.4.2. Planificación dinámica de trayectorias . . . 25

3. Descripción de la solución **27**
- 3.1. Planteo del problema . . . 27
- 3.2. Taxonomía de la solución . . . 28
- 3.3. Estructura de la solución . . . 28
- 3.4. Interacción con otras aplicaciones . . . 30
- 3.5. Modelo del mundo . . . 32
  - 3.5.1. Características del entorno . . . 32
  - 3.5.2. Modelado del entorno . . . 33
    - 3.5.2.1. Información percibida . . . 34
    - 3.5.2.2. Predicción . . . 34
    - 3.5.2.3. Predicados difusos . . . 36
- 3.6. Predicción del ataque oponente . . . 38
  - 3.6.1. Formación . . . 38
    - 3.6.1.1. Robots como hormigas . . . 39
    - 3.6.1.2. Clusterización . . . 44
    - 3.6.1.3. Resultado de la calibración . . . 47
  - 3.6.2. Juego posicional . . . 48
    - 3.6.2.1. Colisiones entre los robots y la pelota . . . 48
    - 3.6.2.2. Goles . . . 49
    - 3.6.2.3. Pases entre robots oponentes . . . 50
  - 3.6.3. Comportamiento . . . 52
  - 3.6.4. Características técnicas . . . 54
  - 3.6.5. Evaluación . . . 55
    - 3.6.5.1. Formación . . . 55

3.6.5.2. Juego posicional . . . 55
3.6.5.3. Comportamiento . . . 56
3.7. Toma de decisiones . . . 56
3.7.1. Variables lingüísticas, particiones y reglas . . . 56
3.7.2. Estructura . . . 58
3.7.3. Modelado y Ajuste . . . 61
3.7.4. Proceso de toma de decisiones . . . 64
3.7.4.1. Optimización del proceso . . . 64
3.7.5. Evaluación . . . 65
3.7.5.1. Adaptabilidad . . . 65
3.7.5.2. Cooperación . . . 67
3.8. Planificación de Movimientos . . . 69
3.8.1. Problemática . . . 69
3.8.2. Solución . . . 71
3.8.2.1. Características del algoritmo reutilizado . . . 71
3.8.2.2. Adecuación del algoritmo . . . 73
3.8.2.3. Estructura de la solución . . . 74
3.8.3. Evaluación . . . 75
3.8.4. Resultados . . . 76

4. Evaluación global **83**
4.1. Introducción . . . 83
4.2. Experimentos . . . 83
4.2.1. Limitaciones tecnológicas . . . 83
4.2.2. Criterios de evaluación . . . 84
4.3. Ambiente de prueba . . . 85
4.3.1. Porcentaje de tiempo que la pelota se juega en campo propio u oponente . . . 86
4.3.2. Distribución de jugadores en la cancha . . . 87
4.3.3. Cantidad de goles . . . 88
4.3.4. Efectividad del ataque oponente . . . 88
4.3.5. Efectividad de la defensa propia . . . 90
4.4. Ambiente de competición . . . 91
4.4.1. Porcentaje de tiempo que la pelota se juega en campo propio u oponente . . . 91
4.4.2. Distribución de jugadores en la cancha . . . 91
4.4.3. Cantidad de goles . . . 92
4.4.4. Efectividad del ataque oponente . . . 93
4.4.5. Efectividad de la defensa propia . . . 95
4.5. Otros resultados . . . 95
4.6. Análisis de resultados obtenidos . . . 96

5. Conclusiones **99**
5.1. Trabajo futuro . . . 100

A. Glosario **103**

Bibliografía **105**

Índice de figuras

2.1. Etapas de un sistema de reconocimiento de patrones. 18
2.2. Tipos de funciones de pertenencia. 23

3.1. Ejemplo de intercambio de estrategias que posibilita la capa Equipo. 29
3.2. Arquitectura simplificada de la solución. 30
3.3. Flujo de información entre las capas. 31
3.4. Estructura de los mensajes que se intercambian con aplicaciones externas. 32
3.5. Niveles de información del modelo del mundo. 33
3.6. Ejemplo de una red de *pipes* y *filters*. 36
3.7. Comparación entre conjuntos clásicos y difusos. 37
3.8. Posiciones donde se coloca feromona para un robot. 41
3.9. Pruebas para determinar la cantidad de feromona depositada y evaporada. 41
3.10. Secuencias de partidos jugados contra dos equipos que presentan un patrón de comportamiento estable. 42
3.11. Comparación de resultados de las hormigas para el equipo Australia cada 300 iteraciones. 43
3.12. Análisis de diferentes umbrales de concentración de feromona a las 6.000 iteraciones de un partido con el equipo Australia. 45
3.13. Evolución de la concentración de feromona durante el primer tiempo de juego. . . 46
3.14. Comparación de resultados de *clusterización* para el equipo Australia cada 300 iteraciones. 47
3.15. Gráfico del movimiento de la pelota. Movimiento practicamente nulo. 49
3.16. Gráfico del movimiento de la pelota. Movimiento y colisión verticales. 49
3.17. Situaciones a considerar para detectar un gol. 50
3.18. Ejemplo del proceso de detección de pases. 51
3.19. Ejemplo de detección de un pase *oponente-arco* con rebote en el golero. 51
3.20. Ejemplo de detección de pases. 52
3.21. Resultado de una prueba de detección de colisiones y pases. 53
3.22. Ejemplo de formación y pases del oponente. 54
3.23. Ejemplo de generación del grafo que modela el comportamiento. 54
3.24. Red de *pipes* y *filters* para generar el grafo que modela el comportamiento. 54
3.25. Evolución de la detección de colisiones y pases en el primer tiempo de juego contra el equipo Australia. 55
3.26. Generación del grafo que modela el comportamiento del equipo Australia. 56
3.27. Particiones de un conjunto difuso. 57
3.28. Función de pertenencia para variables que representan distancias. 57
3.29. Regla que define la función de utilidad de la acción *Tirar al arco*. 58
3.30. Función de pertenencia *Tirar al arco*. 59
3.31. Estructura del sistema de toma de decisiones. 59
3.32. Variante de la regla asociada a la acción *Tirar al arco*. 63
3.33. Evolución del tanteador entre Pollito Five II y FIbRA. 66
3.34. Asignación de estrategias. 66
3.35. Distribución porcentual de roles. 67
3.36. Roles asignados al robot 0 en el partido frente a PFII. 68

3.37. Roles asignados al robot 1 en el partido frente a PFII. 68
3.38. Roles asignados al robot 3 en el partido frente a PFII. 68
3.39. Asignación del rol golero. 69
3.40. Desplazamiento del eje de las ruedas respecto al centro del robot. 70
3.41. Elección de la configuración objetivo para la acción *Tiro al arco*. 73
3.42. Interacción entre los distintos componentes encargados de resolver la planificación de movimientos. 74
3.43. Dependencia entre los componentes involucrados en la planificación de movimientos. 75
3.44. Robot simétrico respecto al plano vertical determinado por el eje de las ruedas. . 76
3.45. Asimetrías en el movimiento de los robots. 78
3.46. Trayectorias generadas tomando como configuración objetivo la posición de la pelota y una rotación de 0 radianes, sin utilizar predicción. 79
3.47. Convergencia entre la configuración del robot y la configuración objetivo para las trayectorias de la figura 3.46. 80
3.48. Trayectorias generadas tomando como configuración objetivo la posición de la pelota y una rotación de 0 radianes, utilizando predicción. 80
3.49. Convergencia entre la configuración del robot y la configuración objetivo para las trayectorias de la figura 3.48. 81

4.1. División de la cancha para evaluar el movimiento de la pelota y los robots. 84
4.2. Porcentaje de iteraciones que la pelota se encuentra en campo propio u oponente. . 86
4.3. Porcentaje de iteraciones que la pelota se encuentra en campo propio u oponente. . 86
4.4. Distribución de los robots en la cancha antes de generada la predicción. 87
4.5. Distribución de los robots en la cancha después de generada la predicción. 87
4.6. Distribución de los robots en la cancha sin predicción. 88
4.7. Goles convertidos en cada arco para cada equipo. 89
4.8. Porcentaje y efectividad de las llegadas al arco opuesto. 89
4.9. Efectividad del ataque oponente. 90
4.10. Efectividad de la defensa del equipo FIbRA. 90
4.11. Porcentaje de iteraciones que la pelota se encuentra en campo propio u oponente. . 91
4.12. Distribución de los robots en la cancha antes de generada la predicción. 92
4.13. Distribución de los robots en la cancha después de generada la predicción. 92
4.14. Distribución de los robots en la cancha durante todo el primer tiempo. 92
4.15. Goles convertidos en cada arco para cada equipo. 93
4.16. Porcentaje y efectividad de las llegadas al arco opuesto. 94
4.17. Efectividad del ataque oponente. 94
4.18. Efectividad de la defensa del equipo FIbRA. 95
4.19. Tiempo de respuesta de la estrategia. 96

Índice de cuadros

2.1. Clasificación de *Russell* y *Norvig*. 21

3.1. Declaración de una red de *pipes* y *filters*. 36
3.2. Valores parciales de la calibración del modelo de formación. 43
3.3. Algoritmo de *clusterización* utilizado para generar *clusters*. 45
3.4. Valores finales de la calibración del modelo de formación. 47
3.5. Estrategias. 60
3.6. Roles. 61
3.7. Acciones. 62

4.1. Resultado de los partidos de prueba. 88
4.2. Cantidad de llegadas de cada equipo al arco opuesto. 89
4.3. Resultado parcial de los partidos jugados con los distintos equipos en el CAFR2006. 93
4.4. Resultado final de los partidos jugados en el CAFR2006. 93
4.5. Cantidad de llegadas de cada equipo al arco opuesto. 94
4.6. Tiempo promedio de respuesta de la estrategia y sus componentes. 96

Capítulo 1

Introducción

[1]

¿Es posible construir máquinas que realicen tareas, que si fueran realizadas por humanos, requerirían de inteligencia? Si un perro pudiera realizar operaciones aritméticas simples ¿diríamos que es inteligente? En ese caso ¿podríamos aceptar que un programa que realiza la liquidación de toda la plantilla de una fábrica, es también inteligente? ¿Puede un programa pensar como un humano?[2] ¿Existe una cota por encima de la cual se pueda decir que algo es inteligente? Hace no muchos años se pensaba que si una máquina pudiera vencer al campeón de ajedrez sería, irremediablemente, inteligente...

La Inteligencia Artificial es un concepto que suele considerarse moderno y relacionado con las nuevas tecnologías. Sin embargo, es tan antiguo que existen registros de su uso desde cientos de años antes de Cristo. A través de las épocas, han sido innumerables los intentos del hombre por construir máquinas que actúen como seres vivos e incluso con inteligencia. Desde estatuas movidas por vapor o caídas de agua, de las culturas griega, etíope y china, hasta los sofisticados sistemas expertos utilizados en la actualidad.

Filosofía, matemática, economía, neurociencia, psicología, cibernética y lingüística han sido algunas de las disciplinas que han contribuido con ideas, enfoques y técnicas al desarrollo de la Inteligencia Artificial. A pesar de tantas contribuciones, no menos han sido las críticas o polémicas que han rodeado este desarrollo: desde la definición misma de inteligencia[3], para la cual aún hoy no se concibe una redacción que contemple todos los aspectos de la actividad humana que implican inteligencia; hasta el propio término Inteligencia Artificial, criticado por quienes sostienen que es una contradicción terminológica porque la inteligencia es una característica distintiva de cierto tipo de sistemas biológicos; así como también los cuestionamientos a los pequeños avances por quienes sostienen que una máquina nunca será capaz de sentir emociones, discernir entre lo bueno y lo malo, arrepentirse o ser creativa.

De todos modos, para bien o para mal, la Inteligencia Artificial se ha desarrollado fuertemente y es una realidad de nuestros días. Ya no se puede negar la existencia de máquinas o programas "inteligentes". El hombre a diario se vale de soluciones o tareas realizadas por máquinas o programas. Hoy son de uso frecuente los sistemas que resuelven teoremas matemáticos, traducen textos en diferentes idiomas, asisten a los médicos en la elaboración de diagnósticos, permiten el reconocimiento de lugares inaccesibles para el hombre, realizan planificación logística, dan soporte en educación y potencian los juegos de computadora, entre otras tareas que requieren, al menos, cierta inteligencia.

[1] Esta introducción se realizó utilizando como fuente de inspiración las interrogantes, conceptos, fundamentos y relatos históricos vertidos en los siguientes textos [Sim87, RN02]

[2] Esta interrogante permanece sin respuesta hasta nuestros días debido a que los avances científicos (neurociencia, psicología) aún no conciben modelos completos de todas las actividades de la mente que permitan reconocer, en sentido completo, cómo piensa un humano.

[3] El término Inteligencia Artificial fue propuesto por McCarthy en 1956 y acuñado por la comunidad de investigadores desde entonces.

Una definición muy citada en la literatura técnica es la propuesta por Marvin Minsky. "Inteligencia Artificial es el arte de construir máquinas capaces de hacer cosas que requerirían inteligencia en caso de que fuesen hechas por los seres humanos".

Un gran número de investigadores, ante los resultados concretos y alentadores, han sentido la motivación de repensar soluciones tradicionales a problemas complejos para lograr mejores resultados aplicando Inteligencia Artificial. De esta forma también han surgido áreas nuevas de investigación que promueven la resolución de problemas frecuentes o de interés para las actividades del hombre, en ambientes reducidos y controlados.

El fútbol de robots, que surgió en 1992 como resultado de esfuerzos por desarrollar entornos donde realizar experimentos con múltiples robots, es un ejemplo de ello. En 1993, se realizó el primer torneo de fútbol de robots, motivando la creación del proyecto internacional RoboCup (*Robot Soccer World Cup*), creado para promover la investigación en Inteligencia Artificial y Robótica. En 1997, se creó oficialmente la FIRA (*Federation of International Robot-soccer Association*) para ofrecer a las generaciones más nuevas el desafío de trabajar sobre sistemas de robots autónomos móviles. De esta forma, utilizando el componente lúdico, estos emprendimientos han logrado el acercamiento de miles de estudiantes al mundo de la Robótica y la Inteligencia Artificial.

Actualmente, el fútbol de robots ofrece un entorno adverso, dinámico, continuo y multiagente, que representa un desafío interesante para desarrollar sistemas sobre él, a la vez que es utilizado como banco de pruebas de propuestas sobre arquitecturas cooperativas, comunicación, teoría de control, navegación, procesamiento de imágenes y razonamiento en tiempo real, entre otras temáticas.

1.1. Antecedentes

En Uruguay, el Instituto de Computación de la Facultad de Ingeniería de la UdelaR ha promovido varios emprendimientos en las áreas de Inteligencia Artificial y Robótica. En el año 2003, se realizaron dos proyectos de grado en el contexto de la categoría real MiroSot de la FIRA: visión artificial y construcción de robots; y uno en el de la categoría simulada SimuroSot sobre controles de movimiento, acciones y estrategia de un equipo de fútbol.

Un poco al margen del fútbol de robots, pero en el área de Robótica, en el 2004 se realizó un proyecto que impulsó la construcción de un robot bípedo.

Con estos antecedentes, en el año 2005 se propuso este proyecto de grado para dar continuidad a los trabajos realizados sobre comportamiento de alto nivel de equipos de fútbol de robots de la categoría SimuroSot de la FIRA.

1.2. Objetivos del proyecto

Los objetivos planificados desde el comienzo del proyecto fueron:

- construir programas inteligentes que permitan controlar un equipo de fútbol de robots en un entorno controlado y simulado[4].
- analizar la factibilidad de reutilizar los artefactos producidos por los proyectos realizados en el año 2003.
- lograr una mejora comparativa en los aspectos de control de movimientos, acciones y estrategia, haciendo mayor hincapié en los dos últimos.

1.3. Organización del documento

En el capítulo 2 se describen las ideas y técnicas estudiadas que conformaron el marco teórico dentro del cual se desarrolló la solución. Reconocimiento de patrones, toma de decisiones, lógica difusa y planificación de movimientos son algunos de los temas abordados.

La solución propuesta se describe en el capítulo 3. Primeramente, se presenta en líneas generales cuál es el problema a resolver y los aspectos más relevantes que lo caracterizan. A continuación se describe la representación o modelo del mundo que utiliza el sistema, un módulo que permite

[4] Generado por *Robot Soccer Simulator*, el simulador oficial que se utiliza en la categoría *Simurosot Middle League* de la FIRA.

realizar predicción o reconocimiento de la formación del equipo oponente, el sistema de toma de decisiones y finalmente, las estrategias y controles utilizados en el movimiento de los robots.

En el capítulo 4 se muestran los experimentos y pruebas realizadas sobre el desempeño del equipo y se realiza una evaluación a partir de los resultados obtenidos. En este sentido, se consideran resultados de pruebas de laboratorio realizadas con sistemas construidos hace dos o tres años y también los resultados obtenidos en el último campeonato argentino de la categoría[5].

Finalmente, las conclusiones generales sobre todo el trabajo, las líneas de investigación consideradas y el trabajo futuro propuesto se presentan en el capítulo 5.

Glosario y Referencias conforman el apéndice y contienen la terminología, acrónimos y definiciones utilizadas en el documento y los textos, publicaciones y artículos consultados, respectivamente.

[5] El sistema FIbRA compitió en el IV Campeonato Argentino de Fútbol de Robots (CAFR2006) de la categoría SimuroSot de la FIRA realizado en la Universidad Abierta Interamericana (UAI) de la ciudad de Buenos Aires, sin obtener galardones en lo deportivo pero recibiendo el premio al mejor trabajo de investigación por la presentación de un *paper* [ABR06c] en el II *Workshop* en Inteligencia Artificial aplicada a la Robótica Móvil (WCAFR2006) que se realizara como parte del evento.

Capítulo 2

Marco teórico

En este capítulo se presenta un resumen del estudio realizado en las diferentes áreas temáticas que aportaron ideas para la construcción de la solución final. Un análisis más detallado de éstas y otras áreas investigadas, que tienen aplicación en el fútbol de robots, puede ser consultado en [ABR05a].

Para la resolución del problema que ocupa a este proyecto de grado -la construcción de un equipo de fútbol de robots- se investigan las ideas, técnicas y metodologías de tres grandes líneas de trabajo:

- *Predicción* : Se estudian las técnicas propuestas para reconocimiento de patrones (sección 2.1) y su aplicación tanto en la FIRA como en RoboCup, combinadas con enfoques inspirados en sistemas biológicos (sección 2.2).
- *Toma de decisiones* : Se analizan diferentes enfoques para la construcción de sistemas de toma de decisiones, al tiempo que se estudia la utilización de lógica difusa y su aplicación para resolver problemas en equipos de fútbol de robots (sección 2.3).
- *Planificación de movimientos* : Se evalúan distintas estrategias de resolución del problema de planificación de movimientos en base a las características del entorno (sección 2.4).

2.1. Reconocimiento de Patrones

El ser humano es capaz de percibir datos sensoriales a través de sus sentidos y reconocer objetos, conceptos, características, comportamientos, fenómenos, etc. En otras palabras, a través del procesamiento e interpretación de los datos sensoriales se obtiene una descripción concisa y representativa del universo observado[1].

Los elementos del universo se perciben como patrones y los procesos que llevan a su comprensión son llamados *procesos perceptuales*. El etiquetado de esos elementos es lo que se conoce como *reconocimiento de patrones*. Por lo tanto, el reconocimiento de patrones es una herramienta esencial para la interpretación automática de datos sensoriales.

Por ejemplo, el sistema nervioso humano recibe aproximadamente 10^9 bits de datos sensoriales por segundo y la mayoría de esta información es adquirida y procesada por el sistema visual [dTdI06].

2.1.1. Modelo

En lo que refiere al sistema de reconocimiento de patrones del ser humano, se encuentra que los procesos perceptuales pueden ser modelados como un sistema de reconocimiento automático de patrones de tres etapas [dTdI06] esquematizadas en la figura 2.1 y descritas a continuación:

- *Adquisición de datos sensoriales (Sensor)* : Proporciona una representación de los elementos del universo a ser clasificados.

[1] Por observable se entiende todo aquello que sea perceptible, sin restringirse únicamente al sentido de la vista.

- *Extracción de características (Extractor)* : Elimina información redundante e irrelevante a partir del patrón de representación.
- *Toma de decisiones (Clasificador)* : Comprende la toma de decisiones del sistema de reconocimiento. Su rol es decidir la categoría apropiada de los patrones.

Figura 2.1: Etapas de un sistema de reconocimiento de patrones.

Por otro lado, según [PP02], para modelar el reconocimiento automático de patrones son suficientes dos etapas: aprender las propiedades comunes e invariantes de un conjunto de muestras que caracterizan una clase, y clasificar los nuevos miembros de una clase en base a las propiedades en común con las demás muestras del conjunto. Por consiguiente, según este enfoque el reconocimiento de patrones por medio de una computadora puede describirse como una doble transformación: de un espacio de muestras a un conjunto de características, y finalmente de éste a un espacio de decisiones.

2.1.2. Diseño

En lo que refiere al diseño de sistemas de reconocimiento de patrones, se presentan algunos problemas que alejan la posibilidad de encontrar en la práctica una solución óptima. Teóricamente, se podría diseñar un sistema de reconocimiento de patrones óptimo, por ejemplo, mediante la aplicación directa de la teoría de Bayes[2], si se conocieran en su totalidad las características estadísticas del modelo en el proceso de generación de los patrones. Sin embargo en la práctica ésto no es posible. En [dTdI06] se hace referencia a este impedimento, y se identifican como causas: el desconocimiento del modelo completo y las restricciones económicas (*hardware*, tiempo) que surgen a raíz la complejidad del sistema.

2.1.3. Aprendizaje

En general, la base de conocimiento disponible para el diseño de un sistema de estas características es un conjunto de entrenamiento[3] constituido por observaciones, ya sean etiquetadas o no.

Si las observaciones se encuentran etiquetadas, se asume que para cada patrón o vector de observaciones en el conjunto de entrenamiento, un experto asigna una etiqueta con la clase correcta. El diseño de un sistema basado en un conjunto de datos clasificados *a priori* se conoce como *aprendizaje supervisado*.

Si no se dispone de conocimiento experto sobre el conjunto de datos, o si el etiquetado de los patrones de entrenamiento es impracticable, entonces el problema de diseño implica la necesidad de una primera etapa de análisis de los mismos. Este proceso primario de análisis se conoce como *aprendizaje no supervisado*.

En general, dado un conjunto de entrenamiento, el diseño del sistema de reconocimiento de patrones implica [dTdI06]:

1. Inferencia del modelo a partir de los datos (aprendizaje).

[2] El teorema de Bayes relaciona las probabilidades condicional y marginal de eventos estocásticos [Lar02].
[3] Conjunto de datos reales que se utilizan para entrenar y ajustar el sistema.

2. Desarrollo de reglas prácticas de decisión.

3. Simulación y evaluación del rendimiento del sistema.

2.1.4. Reconocimiento de patrones en el fútbol de robots

La predicción del comportamiento futuro de un oponente en entornos competitivos suele ser dificultosa. Una de estas dificultades radica en que se desconocen los planes del oponente. El fútbol de robots es un ejemplo de entorno competitivo, altamente dinámico, multiagente y de tiempo real[4], donde las decisiones deben ser tomadas de forma rápida y precisa. En este sentido, la habilidad de predecir el comportamiento del oponente puede potenciar la precisión de las decisiones tomadas.

Aplicación en FIRA

En [LF04] se presenta una aplicación de reconocimiento de patrones en el fútbol de robots. Más concretamente, se propone un ensayo de solución al problema de identificar algunas facetas del comportamiento oponente para la categoría SimuroSot de la FIRA.

La predicción del comportamiento se realiza en dos instancias. Una relacionada con la estimación del área de movimiento de los distintos robots en una ventana de tiempo, clasificándolos según su modo de juego (defensa, atacante, izquierdo, derecho, etc.). La otra consiste en extraer comportamientos grupales del equipo, determinando las formaciones más comunes. Esto se realiza en base a las posiciones, tomadas como puntos de un espacio multi-dimensional, agrupando las formaciones similares y con mayor tendencia a aparecer, mediante un algoritmo de agrupamiento o *clusterización*.

2.2. Inspiraciones biológicas

Algunas de las conclusiones y estudios realizados sobre sociedades de agentes se basan en observaciones realizadas al comportamiento cooperativo en comunidades de animales como hormigas, abejas y pájaros. En el fútbol de robots incluso se ha llegado a estudiar la compatibilidad con sociedades de predadores, intentando reflejar ciertas conductas en el comportamiento competitivo de los robots. También se ha comenzado a estudiar el comportamiento de los insectos respecto a la interconectividad física que permite una navegación colectiva sobre terrenos grandes y complejos [ABR05a].

2.2.1. Sociedades de hormigas

Las sociedades de insectos, y en particular las colonias de hormigas, son sistemas distribuidos que, independientemente de las particularidades de sus individuos, presentan una organización social muy estructurada. Como resultado de esta organización, las colonias de hormigas pueden realizar tareas que en muchos casos exceden las capacidades de un único individuo [DS04].

Comunicación entre hormigas

Las hormigas utilizan ciertas sustancias químicas para comunicarse entre sí, haciendo uso del ambiente (principalmente el suelo) como intermediario en la comunicación. En determinadas especies de hormigas se ha observado el siguiente comportamiento: cuando una recolectora encuentra una fuente de alimento, retorna al nido dejando un rastro de feromona[5] en su camino. Esta feromona atrae a otras hormigas, las que buscarán la fuente de alimento y regresarán al nido reforzando el rastro químico del mismo camino, atrayendo cada vez a más individuos.

Dado que la feromona es un componente químico, posee la particularidad de evaporarse con el tiempo. Esta característica determina que los caminos marcados por una hormiga, que no son

[4] Sistema de tiempo real refiere a las categorías físicas, sin embargo es deseable considerar esta restricción en categorías simuladas para permitir la migración de los sistemas entre los entornos.

[5] Sustancia o conjunto de sustancias químicas producidas por algunos organismos vivos que las utilizan para transmitir mensajes a otros miembros de la misma especie. Hay varios tipos de feromona, como por ejemplo de alarma, de rastros al alimento, sexuales, y otras que afectan el comportamiento de los individuos de la especie.

seguidos por otras o por ella misma, desaparezcan con el tiempo. Así, los caminos "óptimos" o más cortos se van reforzando y remarcando hasta tanto no se agote la fuente de alimento, mientras que los caminos más largos o menos recorridos se desvanecen con el tiempo.

Modelos y algoritmos basados en hormigas

El área de "algoritmos de hormigas" o *ant algorithms* estudia modelos derivados de la observación del comportamiento real de las hormigas, y los utiliza como fuente de inspiración para el diseño de nuevos algoritmos aplicados principalmente a la resolución de problemas de optimización y control distribuido.

La idea principal es utilizar los principios de organización que permiten un comportamiento coordinado en las sociedades reales de hormigas, para potenciar la coordinación de agentes artificiales que colaboran para resolver problemas computacionales. Muchos de los aspectos del comportamiento de las colonias de hormigas han inspirado diferentes tipos de algoritmos: recolección de alimento, división de tareas, cuidado de las crías y transporte cooperativo. Uno de los ejemplos más exitosos de *ant algorithms* es el *ACO (Ant Colony Optimization)*[6] [DS04].

2.3. Toma de decisiones

Con el advenimiento de nuevas y poderosas tecnologías el desarrollo de sistemas robóticos se ha potenciado fuertemente en las últimas décadas. Se ha incrementado su uso y también la complejidad de los problemas que resuelven. A su vez, los sistemas robóticos han comenzado a utilizarse en áreas de aplicación donde se requiere la participación de más de un robot para cumplir con los objetivos.

En Robótica, Mecánica y Visión Artificial, por citar algunas áreas de aplicación, se han logrado muchos avances; sin embargo, la cuestión de cómo decidir qué debe hacer un robot, aún no tiene una respuesta precisa. Los robots de hoy pueden resolver problemas que implican motricidad fina, pueden reconstruir casi totalmente el entorno donde están ubicados y orientarse, pero ¿cuándo deben hacerlo? ¿por qué deben hacerlo? o ¿en qué medida la acción individual aporta al objetivo global? Estas razones han provocado un mayor acercamiento entre Robótica e Inteligencia Artificial. Mientras la Robótica brinda a los robots diferentes vías para interactuar con el mundo, la Inteligencia Artificial les permite decidir teniendo en cuenta ciertos aspectos del mundo.

2.3.1. Paradigmas

Los sistemas multi-robots pueden clasificarse según la forma en que realizan la toma de decisiones. Como se muestra en el análisis realizado en [ABR05a], existen sistemas que basan sus decisiones exclusivamente en la última información del mundo captada a través de sensores, llamados *reactivos*; existen sistemas que además utilizan información del pasado y pueden manejar la noción de metas y utilidades, llamados *deliberativos*; y también los hay *híbridos* que intentan explotar los beneficios de ambos enfoques minimizando sus debilidades.

El cuadro 2.1 muestra otra clasificación de agentes inteligentes, presentada en [RN02], que toma en cuenta el tipo de información que utiliza el agente y la complejidad que implica su análisis.

Reactivos

Estos sistemas se basan en el modelo condición-acción[7]. Se limitan a percibir el entorno y en base al estado del mismo definen una acción. En entornos totalmente observables pueden desempeñarse correctamente pero están expuestos a *deadlocks* o *livelocks* en entornos parcialmente observables. Los mejores resultados se han obtenido en entornos dinámicos que sufren modificaciones considerables en forma repentina [Sic05].

[6] El algoritmo *Ant Colony Optimization* fue desarrollado por Moyson y Manderick en 1988 y luego continuado por Marco Dorigo. Consiste en una técnica probabilística para la resolución de problemas computacionales, que en ocasiones puede reducirse a encontrar un camino óptimo en un grafo [DS04].

[7] En el modelo condición-acción las acciones son elegidas exclusivamente a partir de reglas simples que relacionan las percepciones con el conjunto de acciones posibles.

Reactivos simples	Se basan exclusivamente en las percepciones actuales ignorando cualquier otro tipo de información. La función que lleva valores del dominio de las percepciones al de acciones se basa en la regla condición-acción.
Reactivo basado en modelo	Cuentan con un modelo de mundo que les permite tomar en cuenta estados pasados. No es capaz de encadenar reglas por lo que no puede planificar acciones futuras. Se desempeñan mejor en entornos pequeños y deterministas.
Basado en metas	Cuentan con un modelo interno del entorno y reglas que utilizan para tomar decisiones como los sistemas basados en modelo. Además, pueden establecer objetivos o metas de largo plazo ya que internamente pueden proyectar el efecto de las acciones adoptadas en el presente. Suelen desempeñarse bien en entornos grandes y complejos.
Basado en utilidades	Son más adecuados para entornos no deterministas. Su representación interna del mundo les permite, en cada momento, elegir la acción que más utilidad brinde al sistema. Para medir la utilidad utilizan *funciones de utilidad* que miden el beneficio que otorga cada estado futuro a los que pueden llegar dependiendo de la acción que elijan. Esta característica es muy utilizada en sistemas donde coexisten metas de mediano plazo que por momentos pueden entrar en conflicto. En general se les considera lo suficientemente sofisticados como para desempeñarse bien en cualquier entorno.
Aprendiz	Poseen la capacidad de aprender de sus propias decisiones. Esta capacidad les permite revisar sus creencias y adaptarse al entorno durante la ejecución. Son los más complejos y pueden desempeñarse correctamente en cualquier entorno.

Cuadro 2.1: Clasificación de *Russell* y *Norvig*.

Se destacan por la rapidez que le imprime a la toma de decisiones, siendo ésta su principal fortaleza. Por el contrario, logran comportamientos simples y, al no tener memoria y por lo tanto capacidad de retener estados anteriores, son incapaces de adaptar las decisiones a los cambios.

Habitualmente son construidos en base a *máquinas de estado* o *árboles de decisión*. Son relativamente simples de construir pero presentan problemas prácticos de inflexibilidad que condicionan la introducción de cambios en el sistema a medida que éste se hace más complejo. En los sistemas que adoptan este enfoque todas las decisiones deben estar definidas de antemano, incluyendo la coordinación entre agentes. Estas restricciones afectan sobre todo a los sistemas multi-robots donde es importante que los robots puedan anticipar las consecuencias de sus actos y la de los demás. El resultado es un cúmulo de robots persiguiendo objetivos individuales, que en ocasiones pueden parecerse a un equipo con objetivos globales (*Swarm robotics*[8]). En conclusión, se verifican comportamientos bastante primitivos con los cuales es difícil lograr objetivos no triviales en entornos complejos, dinámicos e incompletos [PAC04].

Deliberativos

Los sistemas deliberativos, a diferencia de los reactivos, basan sus decisiones en el modelo condición-cognición-acción. Este ciclo agrega una etapa donde, clásicamente, se realiza reconocimiento de la información del entorno, planificación o búsqueda. Suelen considerar información del pasado reciente además de la que reciben a través de los sensores. Pueden considerar información predictiva para anticipar el efecto de acciones pasadas y también el uso de motores

[8] *Swarm robotics* es una aproximación que intenta resolver el problema de coordinación en sistemas multi-robots. El objetivo es construir/diseñar agentes simples a partir de los cuales emerjan comportamientos colectivos. Una fuente de inspiración son las sociedades de insectos, aunque no la única.

lógicos que permitan mejorar el manejo de reglas y proposiciones que se utilizan para modelar los diferentes escenarios [PAC04].

Entre las ventajas que presentan sobre los sistemas reactivos se destaca el poder de expresividad, entendido éste como la capacidad de modelar aspectos del mundo que los sistemas reactivos no tienen.

Una debilidad importante de este enfoque es el tiempo de cómputo que puede requerir adoptar una única decisión. Esta característica no debe subestimarse en general, y particularmente en sistemas de tiempo real o con fuertes restricciones en el tiempo de respuesta. Dependiendo de la cantidad de aspectos del mundo tomados en cuenta y cuán complejo sea el análisis realizado, una sola decisión podría tomar desde una fracción de segundo hasta minutos, tornando este enfoque prohibitivo en algunos entornos como el fútbol de robots.

Híbrido

Los enfoques *reactivo* y *deliberativo* muestran fortalezas y debilidades. Este enfoque intenta ser una solución que adopta lo mejor de ambos, minimizando sus debilidades. La propuesta del enfoque híbrido es tomar la expresividad del enfoque deliberativo y el tiempo de procesamiento del enfoque reactivo, intentando tomar decisiones inteligentes, garantizando el tiempo de respuesta. Para ello, los sistemas *híbridos* se conforman, básicamente, de dos componentes de toma de decisiones, uno deliberativo y otro reactivo. Estos componentes ejecutan en paralelo todo el tiempo. Normalmente, el sistema utiliza las decisiones adoptadas por el componente deliberativo, pero si la respuesta toma demasiado tiempo, utiliza la componente reactiva. La premisa es: la decisión reactiva no es tan buena como la deliberativa pero es mejor que no hacer nada [PAC04].

2.3.2. Lógica difusa

Aristóteles y otros filósofos que lo precedieron se preocuparon y dedicaron tiempo a lo que en esos tiempos denominaban el estado de precisión de las matemáticas. En sus esfuerzos por definir una teoría lógica concisa se basó lo que hoy conocemos como lógica clásica (Booleana).

Sus conclusiones condujeron al siguiente enunciado: "Toda sentencia es o bien verdadera o bien falsa". Esta conclusión es conocida como *Ley de bivalencia,* siendo aceptada como filosóficamente correcta por más de dos mil años. A pesar de la aceptación generalizada, no ha existido sin objeciones puntuales: ¿por qué no concebir que algo pueda ser verdadero y falso simultáneamente? Por ejemplo, según la lógica clásica un vaso con agua puede estar *lleno* o *no lleno*. Sin embargo, un vaso lleno hasta la mitad puede estar medio lleno y no medio lleno al mismo tiempo. Del ejemplo surge que las sentencias *medio lleno* y *no medio lleno* representan simultáneamente la misma situación y por lo tanto violan la ley de bivalencia de Aristóteles [Cob02].

Estas ideas fueron la base de la lógica multivaluada que concibe la existencia de otros valores entre verdadero y falso. En esta lógica, pueden existir una cantidad infinita de valores permitiendo que las sentencias tengan diferentes grados de verdad representados por números reales pertenecientes al intervalo [0..1]. Este concepto fue fundamental para que *Lotfi A. Zadeh*, en el año 1964, introdujera la teoría de conjuntos difusos, lógica difusa y otras extensiones aplicadas al campo matemático [Cob02].

Las principales características de la lógica difusa definidas por *Zadeh* son:

- Los razonamientos exactos son un caso límite de razonamientos inexactos o aproximados.
- Todo es asunto de grados.
- Cualquier sistema lógico puede hacerse difuso.
- El conocimiento es interpretado como una colección de restricciones elásticas o difusas sobre un conjunto de variables.
- La inferencia es vista como un proceso de propagación de las restricciones difusas.

A partir de estas características puede afirmarse que la lógica difusa es una generalización o un superconjunto de la lógica clásica o convencional que se extiende para manejar el concepto de *verdades parciales* entre los casos extremos de verdad y falsedad absoluta.

El concepto de conjunto difuso se define como una generalización del concepto clásico de conjunto. Para definir conjuntos difusos, en lugar de definirlos por extensión enumerando cada uno de sus elementos, o por comprensión especificando condiciones que sus elementos deben cumplir, la pertenencia se define por medio de una función. Esta función de pertenencia, nombrada en general con la letra μ, asocia a cada elemento del universo de discurso un grado de pertenencia en el intervalo [0..1].

Los tipos de funciones más utilizadas se muestran en la figura 2.2.

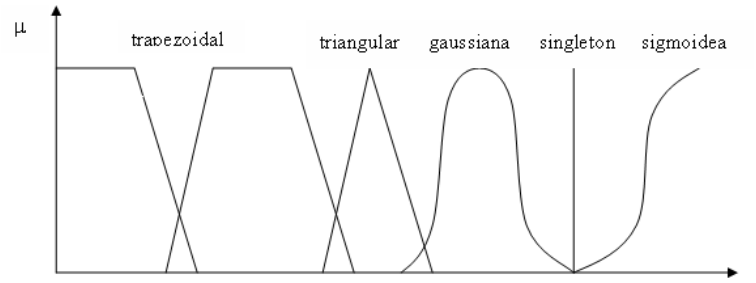


Figura 2.2: Tipos de funciones de pertenencia.

A pesar de que los conceptos de conjunto difuso y lógica difusa se utilizan desde hace años, desde su concepción en 1964, aún hoy siguen siendo objeto de críticas por parte de la comunidad matemática internacional [Cob02]. Sin embargo, la presencia de lógica difusa en el lenguaje natural y el razonamiento humano, dominios difíciles de traducir a 0's y 1's, hacen que se haya difundido ampliamente.

Actualmente, es muy utilizada en sistemas expertos difusos. En esos casos, el uso de conceptos difusos permite definir reglas que formalizan conocimientos imprecisos de forma natural. En áreas como ingeniería y control industrial, se utiliza la gradación de los valores de verdad para dotar a los procesos de controles más suavizados [ECA+03]. El área de fútbol de robots tampoco estuvo ajena a los conceptos difusos. Así, desde hace algunos años a la fecha, varios equipos han adoptado la lógica difusa con diferentes propósitos. Algunos de ellos son:

- Filtrado de datos imprecisos y clasificación de actividades de juego[MKK04].
- Navegación, determinar cuándo evadir obstáculos y cuándo avanzar hacia el objetivo [BKC94].
- Navegación inteligente, autonomía en la navegación permitiendo a los robots cambiar el objetivo local para maximizar el beneficio global del equipo [TLJ97].
- Robots autónomos, toma de decisiones basada en reglas difusas que contemplan la clasificación del escenario, asignación dinámica de roles y de acciones individuales [ENW05].
- Director técnico virtual, control de formación del equipo y tácticas de juego[MMM05].

2.4. Planificación de movimientos

En el desarrollo de sistemas robóticos autónomos, la planificación de movimientos juega un papel clave. Su objetivo es, en primera instancia, especificar en un lenguaje de alto nivel[9] qué tareas deben realizar los robots; y en segunda instancia, traducir esta especificación a un conjunto de comandos de bajo nivel que puedan ser interpretados por los robots para ser transformados en movimientos primitivos que les permitan cumplir con las tareas asignadas.

El Problema de la Mudanza del Piano (*Piano Mover's Problem*) es una versión clásica de problema de planificación de movimientos, el cual consiste en encontrar un camino libre de obstáculos

[9] Esto es, un lenguaje abstraído de los detalles de la arquitectura de los robots y de la forma en que deben de coordinarse sus componentes.

que permita mover un piano desde una habitación a otra (en [CLH+05] puede encontrarse una síntesis del mismo). Aplicado a la Robótica, se pretende encontrar un camino que permita *mover* un robot desde una *configuración inicial*[10] hacia una *configuración objetivo*. A partir del planteo clásico, la planificación de movimientos ha evolucionado para atender una enorme cantidad de variaciones del problema, permitiendo aplicaciones en la industria (p. ej.: diseño automático del plano de distribución de una fábrica, automatización de la línea de ensamblaje), el entretenimiento (p. ej.: animación de personajes digitales), la medicina (p. ej.: planificación quirúrgica, diseño de medicamentos), la realización de tareas de rescate (p. ej.: cartografía de entornos inexplorados), entre otros.

Cada variación tiene su propia particularidad, ya que depende de aspectos inherentes al dominio de la aplicación, como pueden ser:

- las restricciones impuestas por el entorno (p. ej.: leyes físicas que actúan sobre los cuerpos, dimensionalidad);
- la geometría de los robots;
- la cantidad, el tipo y la forma de las articulaciones, que acotan la libertad de movimiento;
- si el entorno es abierto (admite nuevos miembros) o cerrado;
- si los obstáculos son móviles o estacionarios.

Sin embargo, a pesar de su especialización a cada dominio de aplicación, el problema puede clasificarse de acuerdo a qué aspectos del entorno serán tenidos en cuenta para la planificación: si se considera que el entorno es invariante a lo largo del tiempo (entorno estático), el problema se denomina Planificación de Caminos (*path planning*); si se considera que el entorno cambia a lo largo del tiempo (entorno dinámico), el problema se denomina Planificación Dinámica de Trayectorias (*dynamic trajectory planning*)[11].

La solución al primer tipo de problema se reduce a calcular la geometría del camino que debe seguir el robot. Para lograrlo, considera todos los escenarios posibles -teniendo en cuenta las restricciones del entorno- *antes* de que el robot comience la recorrida. Dentro de las restricciones posibles están, por ejemplo, la forma y la ubicación de los obstáculos, el nivel de carga de las baterías y la dinámica de los cuerpos. En el caso estático, todas pueden ser tenidas en cuenta de antemano porque el entorno garantiza que no habrá variaciones con el paso del tiempo.

La solución al segundo tipo debe tener en cuenta que pueden ocurrir eventos no previstos *mientras* el robot completa la trayectoria. Si bien las restricciones pueden ser idénticas a las del caso estático, el quid radica en que no hay garantía de *cuándo* pueden actuar. Por ejemplo, si el robot es propenso a pinchar una rueda, la planificación debe periódicamente verificar el nivel de aire para cambiar de trayectoria en caso de ser necesario. Otro ejemplo es el de la presencia de obstáculos móviles para los cuales se desconoce qué configuración tendrán en cada instante futuro. En este caso, la planificación debe adecuarse a las nuevas configuraciones para esquivarlos[12].

2.4.1. Planificación de caminos

Tradicionalmente, el trabajo en el área de planificación de movimientos se ha centrado en buscar algoritmos que calculen la geometría del camino, bajo la hipótesis de que los robots se mueven en un entorno estático [FS94]. En términos generales, se trata de un problema bien estudiado que admite métodos conocidos para resolverlo. Entre ellos, se encuentran los siguientes:

[10] La configuración de un robot es un conjunto de parámetros independientes que definen de manera única la posición y la orientación de cada punto del robot [Fra99].

[11] Esta clasificación fue realizada por Th. Fraichard *et al.* en [FS94]. Utilizando el mismo criterio, en [Fra99] el autor clasifica el problema de planificación de movimientos en *path planning* y *trajectory planning*.

[12] Esto no quita que existan métodos que permitan predecir, con cierto grado de confianza, cuál será la próxima configuración de los obstáculos. El resultado de la predicción permite trazar una trayectoria tantos instantes hacia el futuro como sea conveniente. Luego de transcurrido un período de tiempo, se vuelve a evaluar el entorno para corregir el movimiento.

- *Mapa de Ruta (Road Map planning)* : A partir de todas las configuraciones iniciales y finales posibles, se construye un grafo topológico que intenta maximizar la eficiencia de los movimientos. Intuitivamente, los caminos deberían ser fáciles de alcanzar desde cada configuración inicial hasta cada configuración objetivo, y el grafo debería poder ser consultado rápidamente.
- *Descomposición en Celdas (Cell Decomposition planning)* : Se descompone el espacio libre de obstáculos en un conjunto de regiones finitas denominadas *celdas*, reduciendo el problema a la búsqueda de un camino entre ellas.
- *Campos de Fuerza (Potential Field planning)* : Se le asigna una fuerza de atracción al objetivo y fuerzas repulsivas a los obstáculos. La suma de todas estas fuerzas "imaginarias" determina la dirección subsiguiente y la velocidad del movimiento.

En [Val06] se describen los métodos basados en Mapas de Ruta y en Descomposición en Celdas, entre otros. En [KB91] se identifican cuatro problemas significativos propios al método de planificación basado en Campos de Fuerza: situaciones de estancamiento debido a mínimos locales (comportamiento cíclico), pasaje cerrado entre obstáculos próximos, oscilaciones en presencia de obstáculos y oscilaciones en pasajes angostos.

2.4.2. Planificación dinámica de trayectorias

Al introducir la hipótesis de que el entorno cambia con el tiempo, el algoritmo de planificación deberá contemplar la dinámica del robot (la interacción entre sus partes) y su interacción física con el entorno a medida que se describe la trayectoria, además de tener en cuenta las restricciones cinéticas y geométricas que impone el movimiento. A modo de ejemplo, una asignación equivocada de velocidades podría provocar que el robot pierda su trayectoria, o que desperdicie tiempo o energía.

Una de las aplicaciones que ha motivado el estudio de este tipo de problemas es el desarrollo de vehículos completamente autónomos, que atiendan a situaciones adversas como el aumento de congestionamientos, polución e inseguridad vehicular en las grandes urbes [FS94].

El problema de Planificación Dinámica de Trayectorias es considerablemente más complejo que el problema de Planificación de Caminos. Además, mientras que para el segundo la vigencia de una trayectoria no depende del tiempo, para el primero la trayectoria encontrada en un momento determinado podría ser inalcanzable en el futuro debido a la evolución del entorno [FS95]. Esta característica ha vuelto infructuosa la búsqueda de una solución general, obligando al desarrollo de soluciones que resuelvan problemas específicos.

En el marco particular del fútbol de robots simulado, el entorno dinámico en el que se mueven los robots se convierte en un atractivo para experimentar en el área de la planificación automática de movimientos. En la sección 3.8 se describe la problemática inherente a la categoría SimuroSot de la FIRA y la solución propuesta.

Capítulo 3

Descripción de la solución

En este capítulo se describe la solución propuesta al problema de construir un equipo de fútbol de robots. Este problema se plantea en términos generales en la sección 3.1. La solución se clasifica en 3.2 según los criterios presentados en 2.3. Su estructura se diagrama y explica de manera abstracta en 3.3. En 3.4 se analiza la forma en que fue resuelta la comunicación entre el sistema construido y el simulador. En 3.5 se describen las principales características del entorno y se define el modelo del mundo (información del entorno, predicción y predicados difusos) que se consideró apropiado para dar soporte a las capacidades deliberativas del sistema; en particular, en 3.6 se profundiza sobre un aspecto de este modelo, que se utiliza para intentar predecir el comportamiento del oponente. El mecanismo de toma de decisiones y la arquitectura propuesta se describen en la sección 3.7. Los algoritmos de planificación de trayectorias, descritos en 3.8, utilizan el modelo del mundo para mejorar el desempeño (rapidez y precisión) a través de la predicción de estados futuros.

3.1. Planteo del problema

Para controlar un equipo de fútbol de robots en el contexto de la categoría simulada de la FIRA (SimuroSot) es necesario tener en cuenta dos problemas que pertenecen a dominios completamente diferentes: por un lado, la comunicación con el simulador oficial; por otro, la construcción del software que permita resolver, en última instancia, qué tareas deben ejecutar los robots para alcanzar el objetivo global que es ganar el partido.

El primer problema está vinculado con los aspectos técnicos que introduce la comunicación entre el programa y la aplicación externa encargada de enviarle la información del entorno. Durante la ejecución del partido, esta aplicación es el simulador oficial. Durante la fase de construcción, esta aplicación podría ser alguna herramienta que ayude a mejorar las cualidades del equipo.

El segundo problema admite más de un planteo. Sin embargo, pueden identificarse tres grandes líneas de trabajo relacionadas entre sí:

- Resolver la toma de decisiones.
- Resolver el movimiento de los robots.
- Encontrar un modelo del mundo que dé soporte a la toma de decisiones y a la planificación de movimientos.

Debe tenerse en cuenta que el entorno[1] en el que están situados los robots es parcialmente observable: si bien el simulador proporciona los datos procesados de la localización de los robots y la pelota -de la misma forma en que un sistema de visión resolvería el procesamiento de imágenes de la cancha tomadas desde arriba-, no se cuenta con toda la información relevante para la toma de decisiones, como ser el estado del tanteador.

[1] Una descripción completa acerca de las características del entorno para el fútbol de robots se describe en la sección 3.5.

Otra característica del entorno es su gran dinamismo, lo cual además de determinar qué clase de problema de planificación de movimientos hay que resolver (ver sección 2.4), acota fuertemente toda capacidad de predicción acerca de cuál será su estado futuro.

Desde el punto de vista de la Ingeniería de Software, el producto desarrollado debe poseer ciertos atributos de calidad para atender algunas necesidades prácticas:

- *Adaptabilidad y configurabilidad*[2] : La competencia admite un breve período de tiempo para realizar cambios en las estrategias de los equipos para permitir adaptarlas a situaciones emergentes; una solución adaptable y configurable ayuda a reducir el riesgo de exceder el tiempo reglamentario.
- *Modularidad* : En iniciativas universitarias es común que los recursos humanos se vayan actualizando; una solución modular ayuda a reducir el tiempo de incorporación de nuevos investigadores al equipo de desarrollo, además de facilitar la transferencia de conocimiento y minimizar la probabilidad de introducir errores.
- *Mantenibilidad* y *extensibilidad* : Se busca que el proyecto tenga continuidad en el futuro; una solución extensible, mantenible y reusable ayuda a que la experiencia perpetúe.

3.2. Taxonomía de la solución

Estrictamente, el sistema es centralizado y pertenece a la categoría *basado en utilidades* descrita en el cuadro 2.1. La única meta de largo plazo considerada por el sistema es la de convertir más goles que el oponente, mientras que las metas de corto plazo implican el cumplimiento de las acciones asignadas a cada robot. No se consideran metas de mediano plazo. En tal sentido, el sistema no considera secuencias de acciones para lograr el objetivo global. Sin embargo, para que cada robot cumpla con la acción asignada debe, reiteradamente, avanzar hacia la concreción de la acción a través de la ejecución de movimientos que sí se planifican de antemano. Estos movimientos se ajustan para seguir trayectorias y éstas a su vez permiten al robot cumplir con su meta de corto plazo: la acción asignada.

En cuanto al uso de utilidades, el sistema basa todas sus decisiones en una discretización de los estados del mundo que realiza utilizando predicados lógicos. Estos predicados son funciones de pertenencia a conjuntos difusos que se definen para modelar los escenarios de interés. Así, en cada momento, la componente encargada de la toma de decisiones utiliza estos predicados para calcular utilidades que le permitan evitar conflictos en la asignación de acciones.

En lo referente al aprendizaje, el sistema no evalúa el resultado de las decisiones adoptadas por lo que es incapaz de aprender de sus propios errores o aciertos. Sin embargo, realiza un análisis o reconocimiento del comportamiento del oponente que le permite decidir presuponiendo la formación ofensiva del oponente.

3.3. Estructura de la solución

La solución está estructurada en capas. Cada una tiene la responsabilidad de resolver un conjunto específico y exclusivo de problemas que se relacionan por pertenecer a un mismo dominio de abstracción. A su vez, cada capa sólo depende funcionalmente de aquellas con menor nivel de abstracción. Las de mayor nivel tienen *visibilidad* sobre el resultado generado por las de nivel inmediatamente menor.

De mayor a menor nivel de abstracción se definen las siguientes capas:

Simulador Proporciona la interfaz que permite a otras aplicaciones comunicarse con el programa. Al aislar el resto del sistema de la contraparte con la que interactúa, pueden utilizarse otras

[2] En la jerga de la Ingeniería de Software, *configurabilidad*, *extensibilidad*, *mantenibilidad* y *modularidad*, son anglicismos utilizados para hacer referencia a distintas cualidades del software: ser fácilmente configurable, extensible, mantenible y cuyos componentes sean módulos altamente cohesivos y poco acoplados entre sí, respectivamente.

aplicaciones además del simulador oficial, que faciliten la construcción y puesta a punto del equipo[3]. Su función es recibir el estado del entorno proveniente del exterior, traducirlo y encapsularlo para enviarlo a la capa *Equipo* (ver sección 3.4).

Equipo Es responsable de enviar la información del entorno a las capas *Modelo del Mundo* y *Toma de Decisiones*, así como de administrar la información de predicción. Esta última responsabilidad implica manejar los tiempos de predicción y la oportunidad de utilizar sus resultados. Su diseño permite encapsular la parte del sistema encargada de resolver el comportamiento de los robots (toma de decisiones y planificación de movimientos), manteniendo intactas las características predictivas del modelo del mundo. Esto posibilita su modificación o el reemplazo íntegro de sus partes sin que esto afecte a los demás componentes. Como ejemplifica la figura 3.1, sería posible implementar varias estrategias esencialmente diferentes en su concepción e intercambiarlas a lo largo de un mismo partido sin desaprovechar la predicción proporcionada por la capa *Modelo del Mundo*.

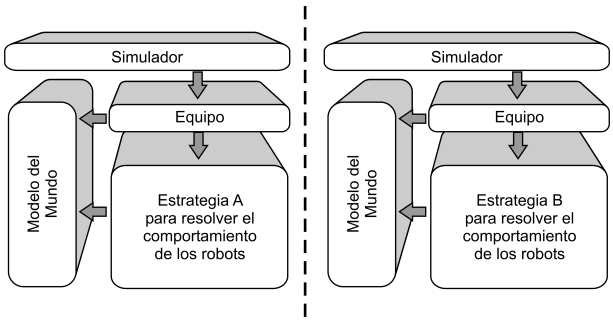


Figura 3.1: Ejemplo de intercambio de estrategias que posibilita la capa Equipo.

Toma de Decisiones En base a la información proveniente del entorno y de la capa *Modelo del Mundo*, decide en cada momento qué acción debe realizar cada robot. Una vez designadas, son comunicadas a la capa *Planificación de Movimientos* (ver sección 3.7).

Planificación de Movimientos Traduce las acciones en comandos de bajo nivel (velocidades de las ruedas) que permitan a los robots adecuarse a los comportamientos esperados. Para mejorar este proceso, se apoya en la predicción de trayectorias que proporciona la capa *Modelo del Mundo* (ver sección 3.8).

Modelo del Mundo Aumenta la potencia cognitiva sobre el entorno al proporcionar información con valor agregado que utilizan las capas *Toma de Decisiones* o *Planificación de Movimientos* (ver sección 3.5).

La figura 3.2 esquematiza esta estructura. El diagrama es una simplificación de la arquitectura real del sistema, que puede consultarse en [ABR06a]. Las flechas representan la relación de dependencia entre las capas.

[3] A modo de ejemplo, se utilizó un visor de *logs* para visualizar partidos grabados lo cual facilitó la realización de pruebas y la detección de errores.

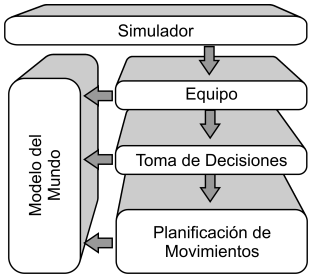


Figura 3.2: Arquitectura simplificada de la solución.

La figura 3.3 muestra el flujo de información durante un ciclo completo de ejecución del sistema cuyos pasos se explican a continuación:

1 — La capa *Simulador* recibe los datos de la localización de los robots y la pelota provenientes de la aplicación externa (p. ej.: el simulador oficial) y los transforma a la representación interna que utiliza el sistema.

2 — Los datos transformados son enviados a la capa *Equipo*.

3 y 4 — La capa *Equipo* envía estos datos a los componentes de predicción de la capa *Modelo de Mundo*, así como a la capa *Toma de Decisiones*. Si bien no espera respuesta de la primera, se queda esperando a que *Toma de Decisiones* retorne el conjunto de velocidades asignadas a las ruedas de los robots.

5 y 7 — Una vez que *Modelo del Mundo* generó la información adicional a partir de los datos de localización, ésta pasa a estar disponible para ser utilizada por las capas *Toma de Decisiones* y *Planificación de Movimientos*.

6 — Luego de definir qué acción deberá ser asignada a cada robot, la capa *Toma de Decisiones* envía a la capa *Planificación de Movimientos* los datos de localización, así como el conjunto de asignaciones; queda esperando las velocidades que la capa *Planificación de Movimientos* determinará para cada robot.

8 — La capa *Planificación de Movimientos* procesa el conjunto de acciones y las traduce a velocidades de las ruedas izquierda y derecha para ser ejecutadas por cada robot. El resultado es retornado a la capa *Toma de Decisiones*.

9 — La capa *Toma de Decisiones* retorna el resultado de las velocidades a la capa *Equipo*.

10 y 11 — La capa *Equipo* recibe las velocidades y se las envía a la capa *Simulador*, la cual transforma el conjunto de velocidades a la representación que maneja la aplicación externa.

3.4. Interacción con otras aplicaciones

Es importante que el sistema pueda interactuar con otras aplicaciones, incluyendo el simulador oficial de la FIRA. A su vez, estas aplicaciones pueden servir a diferentes propósitos, que se describen a continuación:

- *Visores de logs* : Que permiten reproducir partidos previamente grabados, interactuando con el sistema para poder repetir todo el proceso (toma de decisiones, planificación de movimientos, predicción) realizado durante el juego original. Son de gran utilidad para realizar ajustes manuales a la configuración del sistema, así como para la confirmación de resultados esperados.

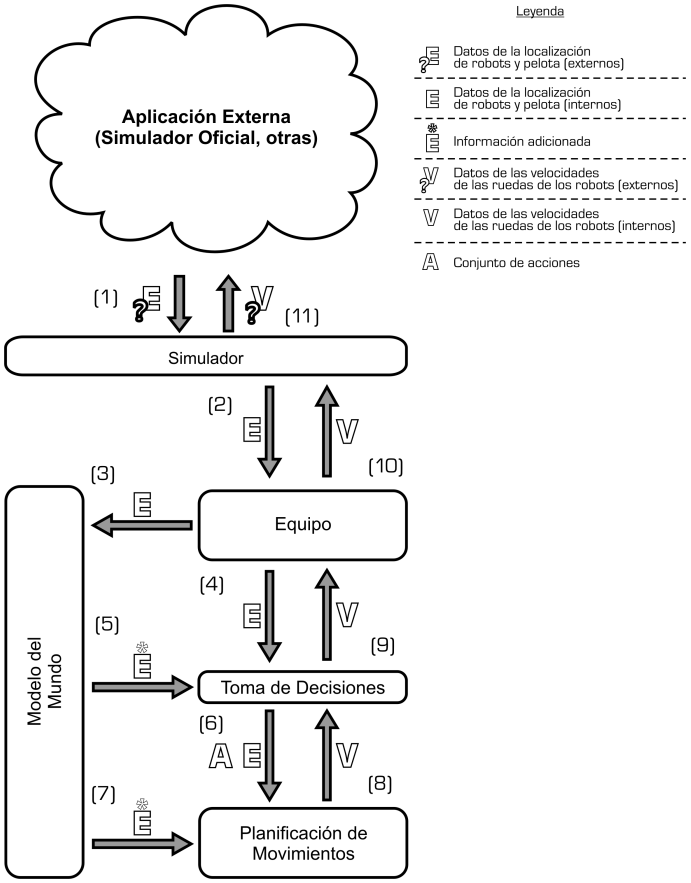


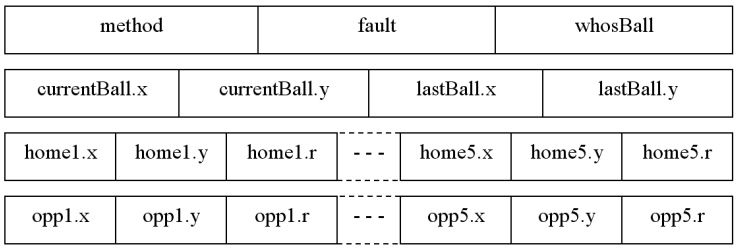
Figura 3.3: Flujo de información entre las capas.

- *Otros simuladores* : Que, a diferencia del simulador oficial, permitan un control programático de sus funcionalidades y, de esa forma, posibiliten la realización de entrenamiento automático mediante la ejecución de secuencias de partidos o jugadas preestablecidas, repetidas veces.
- *Monitores de rendimiento* : Que permitan visualizar y mantener controlado el consumo de cómputo realizado por cada uno de los módulos de la solución, controlando así el tiempo de respuesta de todo el sistema para que se adecue a las restricciones del entorno. Pueden ser de mucho valor a la hora de modificar la implementación o el cambio integral de uno o varios módulos, habilitando la comparación entre el desempeño de una solución y otra.

El sistema debería ser capaz de interactuar con otras aplicaciones sin importar el lenguaje de programación en el que fueron desarrolladas, la arquitectura de *hardware* que requieren y por derivación si el acceso es local o remoto. Para cumplir con esos requerimientos se definen dos interfaces[4]:

- *Software* : En la capa *Equipo* descrita en la sección 3.3, se define un punto de entrada al sistema que implementa la misma interfaz requerida por el simulador oficial. Esta interfaz puede ser utilizada por aplicaciones *Java* que se ejecuten en forma local.
- *Comunicación* : Se define un mecanismo de comunicación sobre el protocolo *UDP* para permitir el acceso remoto desde cualquier tipo de aplicación. En la figura 3.4.(a) se muestra el formato esperado de los datos de entrada, y en la figura 3.4.(b) lo retornado por el sistema.

[4] La especificación de estas interfaces así como los recursos de bajo nivel utilizados en la comunicación se describen en detalle en el documento Descripción de la arquitectura [ABR06a].



(a) Datos de entrada (Situación de juego y estado de los objetos).

(b) Datos de salida (Velocidad de las ruedas de cada robot).

Figura 3.4: Estructura de los mensajes que se intercambian con aplicaciones externas.

3.5. Modelo del mundo

Tomando como referencia la investigación realizada sobre la toma de decisiones en agentes, que se resume en la sección 2.3, se describe a continuación el modelo de mundo sobre el que se apoya la toma de decisiones del sistema construido. El propósito es caracterizar el entorno en el que se desarrollan los partidos de fútbol a la vez de mostrar la organización interna de las componentes que modelan el entorno, y cómo se representa la información que finalmente es consumida en la toma de decisiones.

3.5.1. Características del entorno

Los entornos tienen características que los distinguen. Estas características o dimensiones condicionan fuertemente las soluciones que sobre ellos pueden diseñarse, afectando la complejidad de las mismas. En [RN02] se destacan algunas de las más importantes desde este punto de vista. Para el caso del fútbol de robot simulado de la FIRA el entorno presenta las siguientes características:

- *Parcialmente observable* : Cada agente (robot) cuenta con toda la información del entorno brindada por el simulador. Sin embargo, la información proporcionada no contempla todos los aspectos relevantes para la toma de decisiones (p.ej.: tanteador).
- *Estocástico* : Los estados futuros no dependen solamente de las acciones del agente y del estado actual, por lo tanto, no se pueden determinar exclusivamente a partir de esta información. En este aspecto incide fundamentalmente el hecho de no conocer el proceso de toma de decisiones del equipo oponente ni los algoritmos de planificación de movimientos que utiliza.
- *Secuencial* : Cada tiempo reglamentario de juego podría verse como un episodio que divide la experiencia de los agentes. Sin embargo, las decisiones a corto plazo inciden, definitivamente, en las de largo plazo. Sucede que la calidad del comportamiento presente de los agentes afecta directamente el comportamiento futuro o subsiguiente. Este efecto es notorio durante el transcurso de un período pero también se puede verificar entre períodos, en la incidencia que tiene el desempeño logrado durante la primer etapa sobre el de la segunda. Un ejemplo de ello es un equipo que se ve obligado a cambiar su estrategia de juego para remontar un resultado adverso cosechado durante el primer tiempo de juego.
- *Dinámico* : En la liga simulada el entorno no se modifica mientras el sistema delibera, por lo que, en sentido estricto, es estático. A pesar de ello, el hecho de que los robots oponentes se mueven es una restricción fuerte que debe tenerse en cuenta. También existen restricciones débiles de tiempo (ver [ABR06b]) que tienen por objetivo, básicamente, garantizar una simulación con cierta continuidad que permita una visualización de los partidos que no se entrecorte. Otro aspecto que hace interesante respetar esta restricción es que la liga simulada

sirva, efectivamente, como banco de prueba de sistemas desarrollados para la liga real, donde el entorno es verdaderamente dinámico y, por tanto, minimizar el tiempo de respuesta es crucial para no condicionar la eficiencia del equipo.

- *Continuo* : La cantidad de percepciones y acciones posibles es ilimitada. Las posiciones de los objetos móviles -robots y pelota- recorren un rango de valores reales de forma suave a lo largo del tiempo. Por otro lado, el conjunto de acciones también es continuo, ya que, siendo las ruedas los únicos actuadores[5] que poseen los robots, las acciones posibles quedan determinadas por las velocidades asignadas a cada una de las ruedas en un rango de valores reales [-125 .. 125].

- *Multiagente* : Existen entornos de fútbol donde participan equipos formados por un solo robot. En esos casos se trata de entornos multiagente netamente competitivos, donde un agente intenta maximizar su medida de rendimiento o desempeño, la cual según las reglas, minimiza la medida de rendimiento del agente oponente. En la categoría simulada de la FIRA, donde cada equipo tiene más de un jugador, coexisten agentes competitivos y cooperativos. Compiten entre sí los robots de equipos distintos, individual y colectivamente, y cooperan entre sí los miembros de un mismo equipo.

3.5.2. Modelado del entorno

El modelo del mundo se construye en base a tres tipos de información:

- *Información percibida* : Corresponde a la información proporcionada por el simulador de la FIRA y constituye la información básica a partir de la cual se infieren los otros dos niveles de información que conforman el modelo.

- *Predicción* : Corresponde tanto a la detección de eventos relevantes para la toma de decisiones como a la predicción propiamente dicha de estados futuros y comportamiento oponente.

- *Predicados difusos* : Corresponde a una discretización del continuo de estados de juego mediante la definición de conjuntos difusos.

En la figura 3.5 se muestra la dependencia entre los tres niveles de información. A su vez, la disposición de los niveles está directamente relacionada con el nivel de abstracción del tipo de información que representan. De esta forma, se puede entender que la información proporcionada por el simulador es información en bruto, que la información que proporciona el nivel de predicción tiene cierto valor agregado que permite al sistema reconocer eventos de interés y realizar planificaciones a futuro, y finalmente, el nivel de mayor abstracción discretiza el espacio de estados proporcionando predicados lógicos que permiten reconocer escenarios sobre los que se basa la toma de decisiones.

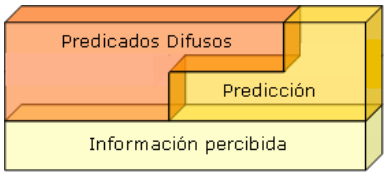


Figura 3.5: Niveles de información del modelo del mundo.

[5] Los actuadores o efectores de un robot son las partes que le permiten modificar el entorno en el que están situados.

3.5.2.1. Información percibida

La información del simulador oficial representa la situación de juego y la localización -o configuración- de los objetos móviles (robots y pelota). La situación de juego corresponde al juego normal o a aquellas jugadas especiales de pelota quieta como ser penales, piques, saques de arco, entre otras. La información recibida corresponde al momento actual del partido, salvo para la pelota, en cuyo caso también se proporciona la localización inmediatamente anterior.

3.5.2.2. Predicción

Para poder predecir el comportamiento esperado de algún objeto o aspecto del entorno es necesario observar y detectar el comportamiento actual que está experimentando. Esto conlleva a la sub-división del proceso de predicción en dos niveles de abstracción:

1. detección y extracción de la información actual;
2. refinamiento de la información actual y predicción a futuro.

Para resolver el problema de extracción de información, se definen *detectores* o filtros de información. Cada detector se especializa en determinado aspecto observable del entorno, filtrando y generando información adicional. Ejemplo de esto lo constituye el detector de goles, responsable de determinar cuándo se ha convertido un gol a partir de los datos del entorno.

Por su parte, el refinamiento de esta información se realiza por medio de *predictores* y *monitores*[6]. Éstos toman la información derivada de los detectores, combinándola o refinándola, para generar nueva información que permita predecir. Los predictores se caracterizan por procesar información durante un período determinado, finalizando su proceso al generarse el resultado. En contraposición, los monitores procesan información sin tiempo límite, generando resultados parciales, sin necesidad de finalizar su ejecución. Por ejemplo, asumiendo la existencia de un detector de goles, ¿cómo puede conocerse el tanteador del partido? La respuesta queda determinada luego de identificar si esta información referida al tanteador debe ser actualizada y consultada durante todo el partido o sólo en un momento determinado. Para el caso del tanteador es imprescindible que la información se actualice y se encuentre disponible durante todo el partido, lo cual implica la definición de un monitor.

Se definen los siguientes detectores que enriquecen la información percibida:

- *Detector de obstáculos* : Identifica los obstáculos que interfieren en una trayectoria determinada por una posición de origen y una de destino.
- *Detector de colisiones* : Detecta las colisiones que se producen entre la pelota y los robots.
- *Detector de pases* : Identifica la realización de pases entre robots oponentes a partir de las colisiones detectadas.
- *Detector de zonas de actividad oponente* : Determina las zonas por las que se mueven los robots oponentes con mayor frecuencia.
- *Detector de goles* : Detecta los goles convertidos durante el transcurso del partido.
- *Detector de atascamientos* : Detecta si algún robot propio se encuentra atascado, inhabilitándolo para cumplir con las acciones asignadas.

En lo que refiere al monitoreo y predicción a futuro, se definen los siguientes componentes:

- *Predictor de la evolución de las zonas libres* : Predice cuáles serán las zonas libres de obstáculos en las próximas iteraciones.
- *Predictor de trayectorias* : Genera la predicción del movimiento esperado de los elementos del entorno. Concretamente, se predicen las posiciones de todos los robots y de la pelota, n iteraciones a futuro.

[6] En [ABR06a] se describen en detalle las características más relevantes de detectores, predictores y monitores.

- *Monitor del estado del juego* : Determina el estado en el que se encuentra el entorno y por cuánto tiempo se ha mantenido en ese estado[7].
- *Monitor del tanteador del partido* : Monitorea el tanteador del partido ofreciendo información de la cantidad de goles que cada equipo ha convertido. Se utiliza la información generada por el detector de goles.
- *Monitor de atascamientos* : Observa el entorno para determinar si hay robots atascados. Se utiliza la información generada por el detector de atascamientos.
- *Predictor y Monitor de comportamiento oponente* : Observa el comportamiento de los robots oponentes, permitiendo predecir su comportamiento a futuro y monitoreando la evolución de dicha predicción. Se basa en la información derivada de los detectores de colisiones, pases, zonas de actividad oponente y goles. Esta predicción es descrita con mayor nivel de detalle en la sección 3.6.

Las restricciones de tiempo derivadas de las características del entorno y de las condiciones de desempeño presentadas en [ABR06b] implican una rápida respuesta del sistema en cada momento. Esto condiciona la predicción, pues en algunos casos puede ser necesario un tiempo de procesamiento mayor al aceptable. Para realizar la predicción sin deteriorar el desempeño del sistema es necesario incorporar paralelismo en la ejecución. Es imprescindible entonces que la toma de decisiones se realice a tiempo aunque no se cuente con la información más actualizada de la predicción: es peor no tomar una decisión a tiempo que tomarla en base a información no actualizada.

A la luz de estas consideraciones, se construye un *framework* de predicción que se ejecuta en paralelo con el resto del sistema. Su implementación se basa fuertemente en la utilización de hilos para lograr paralelismo y en el patrón de arquitectura *Pipes & Filters* que permite combinar detectores para crear predictores o monitores.

Framework de predicción

Cada predictor o monitor puede estar compuesto por uno o más detectores. El paralelismo se logra a través de la creación de un hilo de ejecución para cada detector que compone un predictor o un monitor, mientras que la combinación de la información adicional de cada uno se modela utilizando redes de *pipes* y *filters*[8].

Cada red es alimentada periódicamente con la información percibida desde el entorno. Esta información es procesada por los detectores que componen la red, los cuales están encapsulados dentro de los *filters*. Una vez finalizado el proceso, generan información adicional que puede ser refinada por otros detectores para obtener nueva información, y así sucesivamente hasta obtener el resultado final, colocado en la bandeja de salida por el último detector de la red. Este resultado constituye el producto generado por el monitor o predictor modelado por la red en cuestión. La figura 3.6 muestra un ejemplo de red de *pipes* y *filters* que puede ser instanciada con el *framework*.

Las redes se definen en un archivo de configuración, el cual es cargado por el *framework* al iniciar el sistema. Cada *filter* definido en el archivo es creado como un nuevo hilo de ejecución, que encapsula el procesamiento de un detector. Los *pipes* son modelados como colas bloqueantes, transportando los datos de un *filter* a otro. El cuadro 3.1 muestra el formato de un archivo utilizado para cargar la red de la figura 3.6.

[7] El estado del juego se relaciona con las jugadas especiales de pelota quieta que reconoce el simulador [ABR05c], por ejemplo tiro penal, juego normal, saque de arco, entre otras.

[8] Una red de *pipes* y *filters* se compone de una bandeja de entrada o *source*; varias unidades lógicas de procesamiento o *filters*, conectadas entre sí a través de conectores o *pipes*, que transportan los datos de una unidad lógica a otra; y una bandeja de salida o *sink*. La información inicial es depositada en el *source*. Todos los *filters* esperan la llegada de información procedente del *source* o de los *pipes* de entrada. Al recibir la información, la procesan y colocan el resultado en los *pipes* de salida o en el *sink* si están al final de la red.

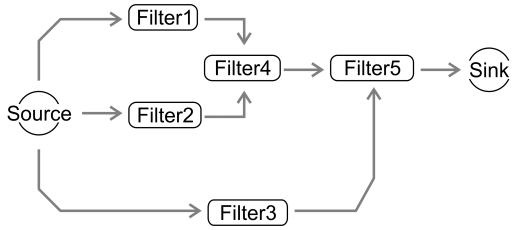


Figura 3.6: Ejemplo de una red de *pipes* y *filters*.

```
<Filters>
      <title> Example Network </title>
      <Filter id="1">
            <class> package.filterClass1 </class>
            <source> true </source>
            <outPipes> <pipe> 4 </pipe> </outPipes>
            <sink> false </sink>
      </Filter>
      <Filter id="2">
            <class> package.filterClass2 </class>
            <source> true </source>
            <outPipes> <pipe> 4 </pipe> </outPipes>
            <sink> false </sink>
      </Filter>
      <Filter id="3">
            <class> package.filterClass3 </class>
            <source> true </source>
            <outPipes> <pipe> 5 </pipe> </outPipes>
            <sink> false </sink>
      </Filter>
      <Filter id="4">
            <class> package.filterClass4 </class>
            <source> false </source>
            <inPipes> <pipe> 1 </pipe>
                      <pipe> 2 </pipe>
            </inPipes>
            <outPipes> <pipe> 3 </pipe> </outPipes>
            <sink> false </sink>
      </Filter>
      <Filter id="5">
            <class> package.filterClass5 </class>
            <source> false </source>
            <inPipes> <pipe> 3 </pipe>
                      <pipe> 4 </pipe>
            </inPipes>
            <sink> true </sink>
      </Filter>
</Filters>
```

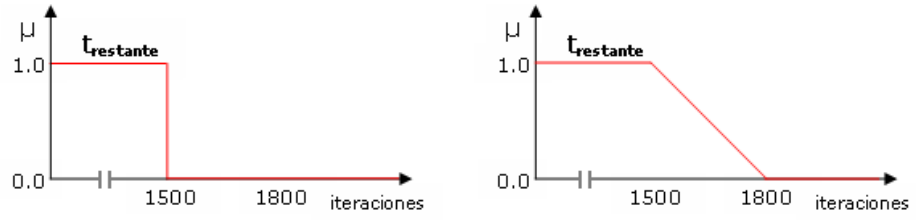
Cuadro 3.1: Declaración de una red de *pipes* y *filters*.

3.5.2.3. Predicados difusos

Los predicados lógicos utilizados son la información de mayor nivel de abstracción del modelo del mundo y se construyen en base a las de menor nivel como lo muestra la figura 3.5. El objetivo es dotar a la toma de decisiones de herramientas que le permitan deliberar sobre un conjunto de estados discretos que representan los escenarios de mayor interés. Estos escenarios pueden verse como situaciones de juego donde el equipo debe, a través de la cooperación de sus miembros, adoptar comportamientos competitivos que lo acerquen al objetivo global. Ejemplos de escenarios interesantes son: situaciones de ataque o defensa en general, tanteadores muy adversos o muy favorables, tiempo de juego escaso, cercanía al arco propio u oponente, cercanía a la pelota, y otros que representan o bien oportunidades o bien peligros que el equipo debe controlar.

La utilización de un enfoque difuso para definir los predicados permite a la toma de decisiones formalizar conocimientos imprecisos de forma natural. Para el caso de un juego de fútbol de robots, es interesante poder basar las decisiones en proposiciones tales como "el robot x está cerca de la pelota" o "la pelota está cerca del arco propio" o "el tiempo de juego se está terminando".

En la figura 3.7 se muestra la diferencia entre modelar estos conceptos empleando conjuntos lógicos clásicos y conjuntos difusos. En estos casos, el enfoque clásico no es apropiado ya que no parece muy sensato que una vez transcurridas 1495 iteraciones el sistema considere que aún resta suficiente tiempo de juego y 10 iteraciones más tarde (160 ms después) considere que ya no hay tiempo. Aún siendo arbitrarias, las funciones de pertenencia μ permiten expresar una gradación de valores que modela mejor los conceptos de este tipo.



(a) Conjunto clásico.

(b) Conjunto difuso con μ trapezoidal.

Figura 3.7: Comparación entre conjuntos clásicos y difusos.

Los conjuntos difusos se definen a partir de la determinación de variables lingüísticas. Para el ejemplo de la figura 3.7 el tiempo de juego transcurrido puede verse como una variable lingüística. En un sistema difuso las variables lingüísticas pueden ser de entrada o salida. En la toma de decisiones se utilizan las siguientes variables de entrada:

- *Tiempo* : Permite modelar el tiempo de juego restante. Se usa en combinación con la variable tanteador para modificar la formación del equipo tornándola más o menos agresiva. Básicamente, se intenta modelar aquellos escenarios donde el tanteador es ajustado (favorable o adverso) y el tiempo de juego se agota. El objetivo es que el equipo pueda considerar una estrategia más defensiva para cuidar un resultado parcial apenas favorable (p.ej.: 1-0) cuando el tiempo restante es escaso (p.ej.: 30 segundos) como para sobreponerse a recibir un gol.

- *Distancia* : Permite modelar la noción de cercanía entre diferentes objetos que intervienen en el juego. Lo más utilizado es la cercanía entre robots propios y oponentes a la pelota, entre robots y arcos, entre la pelota y los arcos. También se utiliza para modelar la cercanía de los robots propios a zonas libres de la cancha que son tenidas en cuenta en jugadas como *despejar*, *pasar*, *recibir pase*, entre otras.

- *Ángulo* : Permite modelar ángulos de tiros al arco, pases y despejes.

- *Tanteador* : Permite modelar tanteadores muy adversos o muy favorables y conjuntamente con la variable tiempo adaptar la formación del equipo.

- *Posición relativa* : Permite modelar las relaciones posicionales existentes entre robots propios y oponentes y la pelota. Estas relaciones pueden describir situaciones como "el robot x está detrás de la pelota" o "la pelota está entre un robot propio y uno oponente".

- *Área* : Permite modelar zonas particularmente interesantes como los arcos. En ocasiones, los robots son empujados hacia adentro de los arcos por otros robots en medio de una jugada. La forma y área de los arcos hace que sea más eficiente utilizar acciones específicas para retirar al robot hacia afuera por lo que es necesario considerar conjuntos que permitan modelar estas situaciones.

- *Estado de juego* (*freekicks*) : Representa la colección de variables que permiten modelar las jugadas de pelota quieta determinadas por el árbitro [ABR05c]. Se diferencian de las demás porque definen conjuntos *singleton*, cuyos únicos elementos representan a cada una de las jugadas (ver figura 2.2).

- *Obstaculización en trayectorias* : Permite modelar los niveles de obstaculización de las trayectorias de navegación. Esta variable tiene incidencia directa en la elección de los robots que deben ir a ciertos destinos. Junto con distancia y orientación suelen ser los factores determinantes que permiten decidir cuál es el mejor candidato para realizar la acción que involucra ese desplazamiento.

Las variables de salida son *estrategia*, *rol* y *acción*, las cuales permiten modelar estrategias, roles y acciones ajustadas a los escenarios de juego, respectivamente.

Las estrategias determinan la adecuación de los roles disponibles a la situación de juego. Los roles a su vez inciden sobre las acciones que un robot está en condiciones de desempeñar, o de otro modo, el beneficio que desempeñar una acción dada en un rol dado puede traer al equipo (utilidad). Estas ideas, así como la arquitectura sobre la que se sustentan, se describen en detalle en la sección 3.7.

3.6. Predicción del ataque oponente

La capacidad de observar el comportamiento del oponente y predecir sus movimientos posibilita la anticipación de sus jugadas, con lo cual es posible el mejoramiento del comportamiento propio. Si el equipo es capaz de detectar las zonas de la cancha que el oponente utiliza con mayor frecuencia, así como identificar un patrón en las secuencias de pases realizados, podría utilizar esta información para rearmar sus jugadas y obtener mayor provecho de éstas. Entre otras cosas podría potenciar su defensa, colocando jugadores en posiciones estratégicas que impidan el libre movimiento de los oponentes en la cancha, e incluso anticipar pases, intentando interrumpir las jugadas ofensivas.

La tarea de detectar un patrón de comportamiento entre los robots es un problema de reconocimiento de patrones, como se ha descrito en la sección 2.1. Tomando como referencia el trabajo realizado por Laurenzo y Facciolo [LF04] sobre este problema aplicado a la categoría SimuroSot, se propone una solución al reconocimiento del comportamiento del equipo oponente. Para ello se considera la resolución de tres problemas:

1. *Formación* : Detectar las zonas de la cancha más utilizadas para el ataque.
2. *Juego posicional* : Detectar las secuencias de pases entre los robots.
3. *Comportamiento* : Consolidar la información de los problemas 1 y 2 en un único resultado que represente el comportamiento ofensivo del oponente.

A continuación se presentan las soluciones propuestas a cada uno de los problemas planteados y, finalmente, en la sección 3.6.5, los experimentos realizados así como los resultados obtenidos.

3.6.1. Formación

Para la resolución del problema de reconocimiento de la formación del equipo oponente se define un proceso de dos etapas:

- *Generación de información* : Se utilizan ideas basadas en teorías sobre el comportamiento de insectos, particularmente el de las hormigas (ver sección 2.2).

- *Procesamiento de la información* : Se utiliza un algoritmo de clusterización.

3.6.1.1. Robots como hormigas

Las observaciones sobre el uso de feromona que realizan las hormigas para marcar trayectorias han inspirado la búsqueda de un mecanismo que permita determinar el rastro de los robots oponentes sobre el campo de juego, de forma de identificar sus trayectorias. Para ello, se modela cada robot como una hormiga, la cual se mueve por toda la cancha dejando a su paso un rastro sobre el terreno. Si se observa el campo de juego luego de un tiempo determinado, se podrán identificar las zonas donde se ha acumulado más feromona, lo que puede ser asociado a un nivel mayor de actividad en la zona.

Para intentar que el resultado refleje la formación de los robots, se identifican tres parámetros de ajuste:

- *Política de evaporación* : Determinar la forma en que la feromona es evaporada.
- *Tasa de depósito y de evaporación* : Determinar la concentración de feromona que cada robot deposita sobre el campo en cada instante de tiempo y la cantidad evaporada según la política de evaporación.
- *Tiempo de procesamiento* : Determinar durante cuánto tiempo se marca el rastro de los robots oponentes.

La resolución de estos problemas implica la realización de pruebas de ensayo y error que ayuden a calibrar los parámetros. Así, el diseñador ajusta subjetivamente el sistema, de acuerdo a sus criterios, para lograr que los resultados colmen sus expectativas. Antes de describir estas pruebas y los resultados obtenidos, se presenta el modelo del entorno con el cual se trabaja.

Modelado del entorno

El campo de juego corresponde a parte de un plano que contiene infinitas posiciones, por lo que debe buscarse una representación que modele una superficie bidimensional y permita discretizar las posiciones por donde transitan los robots. En base a ésto, se modela el campo de juego como una matriz de NxM, donde N y M se relacionan con el tamaño de la cancha y de los robots[9]. Más específicamente, cada celda de la matriz corresponde aproximadamente a 1 cm^2de superficie, y las hormigas se modelan como cuerpos que ocupan una superficie cuadrada de 64 celdas.

El modelado de la feromona determina el tipo de datos de las celdas. Por simplicidad se determina que la mínima cantidad depositable es 1 unidad de feromona y que en ningún caso puede haber celdas con concentración negativa, resultando apropiado utilizar números naturales para su representación.

A partir del análisis de varios partidos, se observa una alta frecuencia de atascamientos de robots en algunas posiciones, principalmente en las esquinas de la cancha. Estas situaciones pueden generar falsos positivos[10] en el modelo planteado, por lo que se incorpora una característica adicional para evitar que la concentración de feromona se incremente considerablemente en estas situaciones. Cada celda de la matriz, además de contener la feromona, contiene un contador de tiempo, que indica cuántas iteraciones han transcurrido desde la última vez que se depositó feromona en ella. Si el tiempo es menor que un determinado umbral, el depósito de feromona es descartado. De esta forma el contador establece un umbral máximo para la frecuencia en que se deposita feromona, incidiendo más en las situaciones de atascamiento que en las restantes, debido a que la frecuencia de depósito de las hormigas no atascadas es menor que cuando están atascadas.

En [ABR06a] se profundiza la descripción de la solución desde el punto de vista técnico.

Política de evaporación

La política de evaporación implica resolver cómo y cada cuánto se evapora la feromona. Una vez depositada la feromona sobre el campo de juego, ésta comienza a evaporarse lentamente hasta

[9] Según las reglas de la categoría *SimuroSot* [ABR05c], los robots son cuerpos cúbicos cuya superficie en el campo de juego corresponde a un cuadrado de 7,5 cm de lado, mientras que la cancha es un rectángulo de 220 cm x 180 cm.

[10] Falsos positivos son aquellos resultados producidos en el proceso de detección de zonas, que reflejan cierto nivel de regularidad pero que no se condice con el comportamiento voluntario del oponente.

desaparecer completamente, a menos que el rastro sea reforzado con un nuevo depósito. Dado que efectuar la evaporación en forma continua, por iteración, recarga los tiempos de procesamiento, se opta por realizarla cada n iteraciones, siendo n un parámetro a determinar. El tiempo transcurrido entre una evaporación y la siguiente se denomina *ciclo de evaporación*.

Este modelo implica computar un ciclo de evaporación cada determinada cantidad de iteraciones, posibilitando dos alternativas que determinan resultados diferentes:

1. *Evaporar durante el proceso* : En el momento en que se completa el ciclo de evaporación.
2. *Evaporar al final del proceso* : Evaporando todos los ciclos computados al mismo tiempo una vez finalizado el proceso.

La principal diferencia entre una u otra política radica en la noción de temporalidad, la cual se muestra en el siguiente ejemplo:

- *Escenario inicial* : Se cuenta con un único robot en la cancha y no se realiza evaporación de feromona. Inicialmente el robot se mueve libremente en una esquina de la cancha durante un período de tiempo determinado. Luego el robot se traslada a otra esquina, pasando a moverse libremente allí durante un período de tiempo idéntico al anterior. Finalizado el segundo período de movimiento se han generado dos zonas con igual concentración promedio.

Definido el escenario inicial, se muestran los diferentes resultados que pueden obtenerse según la política de evaporación utilizada.

- *Escenario A* : Se introduce la política de evaporación 1 al escenario inicial. La primera zona en ser generada tendrá un mayor decremento del nivel de feromona por el efecto de una mayor cantidad de evaporaciones.
- *Escenario B* : Se introduce la política de evaporación 2 al escenario inicial. En ambas zonas se realiza la misma cantidad de evaporaciones, siendo imposible distinguir qué zona se generó primero una vez finalizado el proceso.

Así, podría surgir la controversia entre: ¿la zona con menor concentración de feromona fue la primera en ser generada o simplemente tenía menor concentración? ¿Cómo interpretar estos resultados una vez finalizado el proceso si no se conociera la hipótesis de trabajo planteada en el ejemplo: ambas zonas tenían igual cantidad de feromona al final del período de depósito?

Ambas políticas son igualmente válidas, pero dependiendo del resultado que se desee obtener deberá elegirse la más apropiada. Subjetivamente se analizan los resultados obtenidos en el período de calibración, eligiéndose la segunda política debido a que se ajusta mejor al problema que se quiere resolver. Con ella es posible registrar tanto los comportamientos regulares más antiguos como los más recientes sin que el tiempo le reste importancia a los primeros (como ocurriría con la primera política).

Tasa de depósito y de evaporación

Para determinar la cantidad de feromona que una hormiga deposita sobre el campo de juego en cada instante de tiempo, se debe responder las siguientes preguntas: ¿sobre qué superficie (celdas) del campo deposita la feromona cada hormiga?, y ¿qué concentración de feromona deposita por iteración? Lo primero se relaciona con el tamaño del robot y las dimensiones de la cancha. Lo segundo depende de otras variables e implica un análisis más profundo.

En base a la cantidad de celdas que ocupa el robot sobre la matriz se determina dónde colocar la feromona. Se ubica en la matriz la celda correspondiente a la posición del centro del robot, depositando feromona sobre todas aquellas celdas que corresponden a la superficie del mismo. Como simplificación se asume que el robot siempre se encuentra con rotación 0 radianes. La figura 3.8 muestra el modelado del robot sobre la matriz. Se indican la celda correspondiente a la posición del centro y el perímetro del robot. Las celdas coloreadas indican dónde se deposita la feromona.

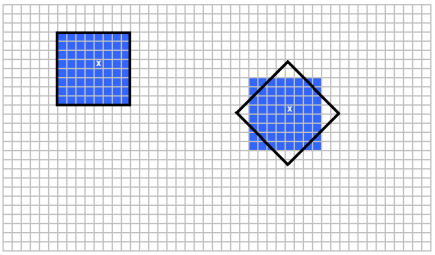
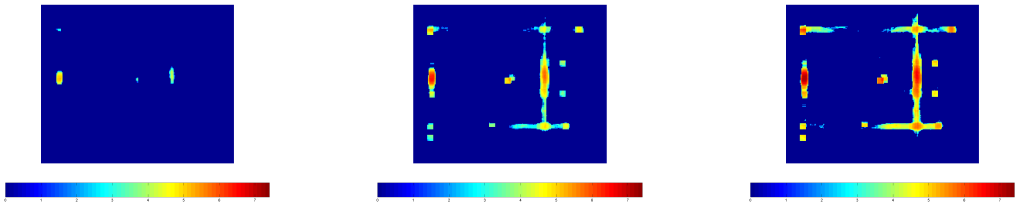


Figura 3.8: Posiciones donde se coloca feromona para un robot.

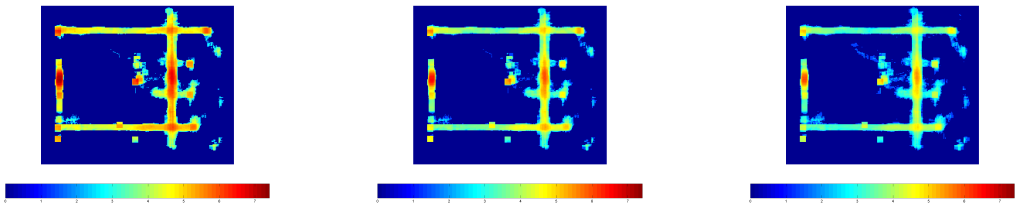
La determinación de la cantidad de feromona depositada por iteración se encuentra estrechamente vinculada con la cantidad que se evapora, la política de evaporación adoptada y las características del entorno, por lo que la elección de un valor resulta compleja. Es necesario realizar una serie de pruebas para diferentes valores, seleccionando aquellos que se ajusten al comportamiento esperado. La figura 3.9 muestra algunos de los resultados obtenidos para diferentes valores[11].



(a) 2 unidades de depósito y 5 unidades de evaporación.

(b) 5 unidades de depósito y 5 unidades de evaporación.

(c) 10 unidades de depósito y 8 unidades de evaporación.



(d) 10 unidades de depósito y 4 unidades de evaporación.

(e) 5 unidades de depósito y 2 unidades de evaporación.

(f) 3 unidades de depósito y 1 unidad de evaporación.

Figura 3.9: Pruebas para determinar la cantidad de feromona depositada y evaporada.

Estos resultados corresponden al análisis de un partido[12] con el equipo Xihua durante 7.000 iteraciones (aproximadamente 2 minutos de juego) y diferentes valores para la cantidad de feromona depositada y evaporada. En este caso se realiza la evaporación cada 300 iteraciones aplicando la política elegida.

Luego del análisis de los resultados obtenidos y la observación del movimiento de los robots durante el partido, se determina subjetivamente que la figura 3.9.(e) se ajusta al comportamiento esperado, con lo cual se seleccionan dichos valores para la cantidad de feromona depositada en cada iteración (5 unidades) y la cantidad de feromona evaporada en cada ciclo de evaporación

[11] Las pruebas han sido realizadas para equipos oponentes que juegan del lado izquierdo (equipo amarillo). La escala de colores determina la concentración de feromona en cada posición. El color azul oscuro implica muy baja concentración, mientras que el color rojo indica muy alta concentración.

[12] El partido con Xihua se toma como ejemplo para mostrar los resultados obtenidos durante el período de calibración, por tratarse del equipo con mayor regularidad.

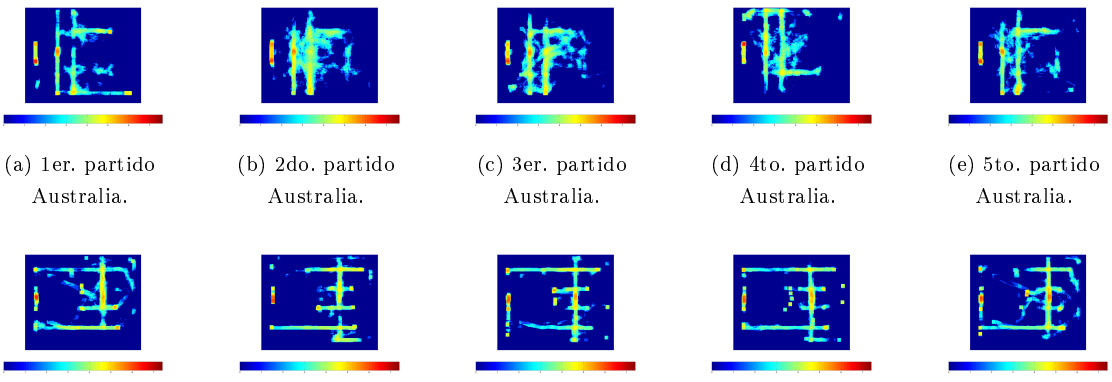
(2 unidades). Se ha observado también que manteniendo cierta proporción entre ambos valores los resultados obtenidos son similares en cuanto a las zonas de actividad generadas, encontrando diferencias en la concentración final obtenida (ver figuras 3.9.(d), (e) y (f)).

Tiempo de procesamiento

El tiempo máximo posible corresponde a un tiempo completo de juego (5 minutos). Sin embargo, tratándose de una técnica *online*, el resultado de la predicción debería estar disponible con antelación suficiente como para ser aplicado. Por otra parte, si el tiempo invertido fuera muy escaso, los resultados podrían no ser suficientemente representativos del comportamiento futuro del oponente. Se busca entonces un valor intermedio que maximice la relación costo/beneficio existente entre el tiempo que se le dedica al procesamiento y la representatividad de la información obtenida.

Otro aspecto a tener en cuenta es que el ajuste basado en los resultados obtenidos para un único equipo puede no ser útil o aplicable para otros. Debe tenerse en cuenta que esta técnica puede no aplicar en algunos casos, por ejemplo para equipos que adaptan su comportamiento al oponente o que cambian aleatoriamente sus movimientos. Para comprobar el determinismo en el comportamiento de algunos equipos, se juegan sucesivos partidos para generar las matrices con los rastros de feromona. Aquellos que realmente presentan un patrón estable en su comportamiento generan matrices similares de un partido a otro.

La figura 3.10 presenta un ejemplo del análisis realizado, donde las secuencias muestran las matrices correspondientes a diferentes partidos jugados contra los equipos Australia y Xihua. En ambas series, a la izquierda puede observarse una alta concentración de feromona relacionada con la regularidad del comportamiento del golero. La primera serie (equipo Australia) pone de manifiesto un comportamiento defensivo regular, con los defensas desplazándose en paralelo hacia ambos lados de la cancha. La segunda serie (equipo Xihua) muestra una defensa que se desplaza horizontalmente y se concentra a ambos lados del arco. A diferencia del ataque del equipo Australia para el cual no se ha detectado un patrón de formación claro (rastros tenues y que no se repiten a lo largo de la serie), el ataque del equipo Xihua muestra una marcada regularidad, utilizando cuatro zonas de aproximación al arco oponente a través de la horizontal y una zona paralela y cercana al arco, donde el atacante se localiza en torno al centro. Es importante remarcar que la ausencia de feromona (en el ataque del equipo Australia o en la defensa del equipo Xihua) no necesariamente indica que no haya habido actividad, sino que el comportamiento no presentó suficiente regularidad como para ser detectado.



(a) 1er. partido Australia. (b) 2do. partido Australia. (c) 3er. partido Australia. (d) 4to. partido Australia. (e) 5to. partido Australia.

(f) 1er. partido Xihua. (g) 2do. partido Xihua (h) 3er. partido Xihua. (i) 4to. partido Xihua. (j) 5to. partido Xihua.

Figura 3.10: Secuencias de partidos jugados contra dos equipos que presentan un patrón de comportamiento estable.

Al reproducir estos partidos, utilizando los valores del cuadro 3.2, generando resultados (matri-

ces) cada 300 iteraciones[13] y comparándolos dos a dos (para calcular las diferencias entre ambos) se puede determinar cómo varían los rastros de feromona en cuanto a forma y concentración. Para ello, se crea una matriz de diferencias que permite medir la distancia entre dos resultados consecutivos, en base a la cantidad de celdas cuyo valor difiere de cero. La ecuación 3.1 presenta el cálculo realizado, donde M^{dk} es la matriz de diferencias, M^{k} es una de las matrices resultado y M^{k+1} es la siguiente.

$$M_{ij}^{dk} = abs(M_{ij}^{k} - M_{ij}^{k+1}) \tag{3.1}$$

Parámetro	Valor
Tasa de depósito	5
Tasa de evaporación	2
Ciclo de evaporación	300
Política de evaporación	Política 2

Cuadro 3.2: Valores parciales de la calibración del modelo de formación.

La figura 3.11 muestra una gráfica con la evaluación de diferencias para un partido con el equipo Australia. En todos los casos se generan matrices de 205 filas por 250 columnas, lo cual corresponde a un total de 51.250 celdas por matriz. Se grafican las evoluciones de:

- la cantidad de celdas positivas de las matrices de diferencia;
- el promedio de celdas con feromona: $\frac{\#celdasFeromonaM_k + \#celdasFeromonaM_{k+1}}{2}$;
- el cociente entre la primera evolución y la segunda (diferencia relativa al promedio).

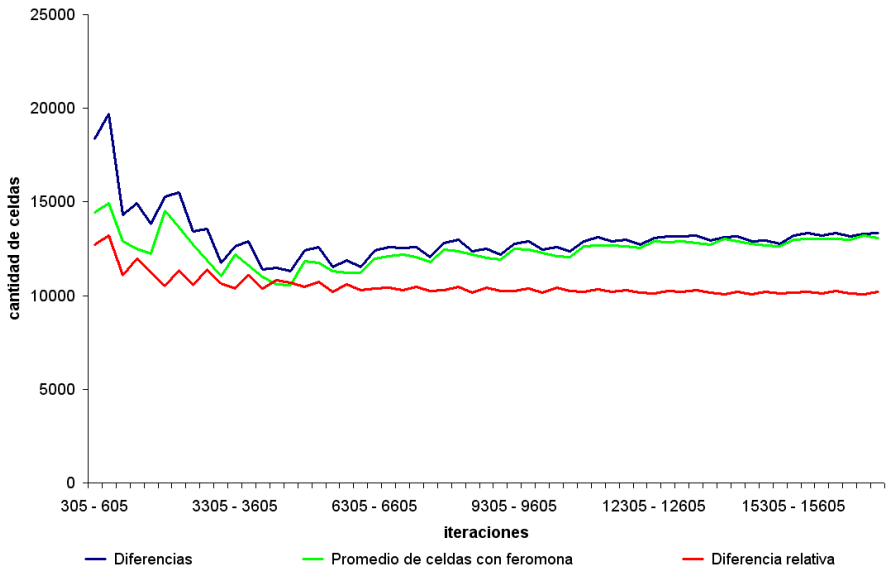


Figura 3.11: Comparación de resultados de las hormigas para el equipo Australia cada 300 iteraciones.

[13] Los resultados se generan cada 300 iteraciones para asegurar la existencia de solamente un ciclo de evaporación en cada uno.

Puede observarse una tendencia a la estabilidad en la evolución de las diferencias a partir de las 5.000 iteraciones aproximadamente. Resultados similares se obtuvieron con otros equipos.

Para determinar un valor adecuado es necesario considerar la segunda etapa de este proceso, la cual se presenta a continuación.

3.6.1.2. Clusterización

Una vez realizado el proceso descrito anteriormente, se obtiene una matriz con los valores de actividad de los robots. Esta información debe ser filtrada (eliminación de ruido) para determinar exactamente las zonas que serán utilizadas por la toma de decisiones, y descartar las celdas con valores de poca o nula actividad. Es necesario entonces aplicar algún algoritmo que realice el recorrido de toda la matriz para generar agrupamientos o cúmulos (*clusters*) de celdas útiles al sistema.

Algoritmo de *clusterización*

Tomando como base el algoritmo de clusterización propuesto en [FST04], se define un algoritmo de *clusterización* de tres fases que toma la matriz con rastros de feromona como entrada y genera una lista de *clusters* convexos, descartando aquellas celdas de la matriz cuya concentración de feromona no representa un nivel de actividad relevante para el sistema.

Se establece un umbral mínimo de concentración aceptable para considerar una celda dentro de un *cluster*. Todas las celdas de la matriz que posean un valor mayor a ese umbral son candidatas a formar parte de algún *cluster*, mientras que las demás celdas serán descartadas.

- *Primera fase*: Selección de celdas con nivel de feromona apropiado, generando rachas de celdas para cada fila de la matriz.
- *Segunda fase*: Tomar las rachas de celdas de la primer fase y formar parches. Cada parche se compone de varias rachas adyacentes (pertenecientes a filas consecutivas y que poseen al menos una celda en la misma columna). Los parches que no poseen una forma convexa son corregidos agregando las celdas necesarias para lograr convexidad. Una vez obtenido el conjunto de parches de toda la matriz se descartan aquellos que son muy pequeños[14].
- *Tercera fase*: Recorrer todos los parches resultantes de la segunda fase y transformarlos en *clusters*, pasando así de la representación matricial a la representación interna de un *cluster* utilizada por el sistema.

El pseudocódigo del algoritmo de tres fases que se acaba de describir se muestra en el cuadro 3.3.

Parámetros de la *clusterización*

Definido el algoritmo a utilizar para filtrar la matriz, se debe determinar el valor del umbral de aceptación de celdas, estableciendo la concentración mínima de feromona de las celdas que pueden formar parte de un *cluster*.

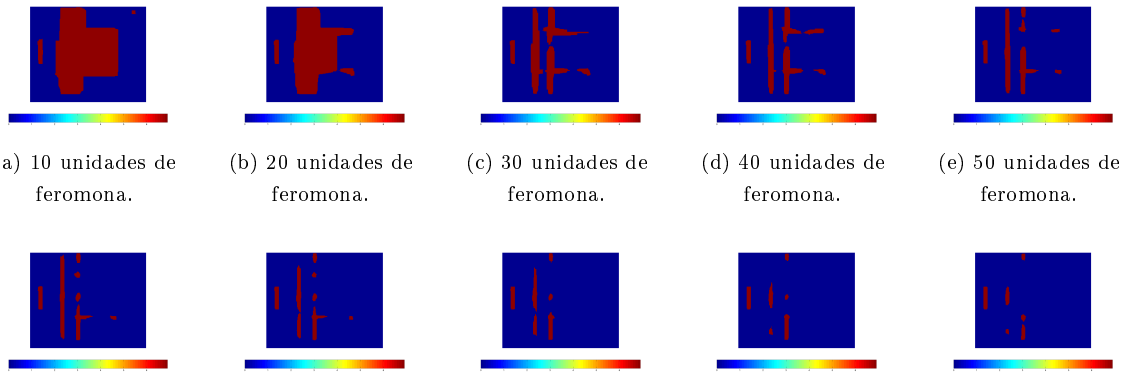
Para encontrar un valor adecuado a dicho umbral, se analizan los resultados obtenidos con distintos valores del mismo a las 6.000 iteraciones[15] con varios equipos. La figura 3.12 muestra un ejemplo de este análisis, donde se obtienen pocos *clusters* y no muy definidos con umbrales muy pequeños, y *clusters* de tamaño muy reducido con umbrales muy altos. A partir de la observación subjetiva de estos resultados y su relación con las matrices de rastros de feromona obtenidas para los mismos equipos, se selecciona el valor de 50 unidades de feromona para el umbral. Con este nivel de concentración se logra una correspondencia visual entre los rastros registrados en las matrices y el resultado final obtenido luego de la *clusterización*. Todas las celdas con una concentración de feromona inferior son descartadas por el algoritmo de *clusterización*.

[14] El tamaño mínimo del parche es configurable y se ha establecido en 60 celdas, correspondiente al tamaño aproximado de un robot. Este valor se establece para una matriz de 205 filas y 250 columnas.

[15] Se selecciona este valor debido a que se observa una tendencia a la estabilidad en la evolución de los resultados de las hormigas a partir de las 5.000 iteraciones (ver sección 3.6.1.1).

Entrada: matriz de valores
Salida: lista de *clusters*
Pseudo-código:
fase 1: generar rachas.
para cada fila de la matriz
para cada celda de la fila
si el valor de la celda es mayor que el umbral mínimo y menor que el umbral máximo
si la celda anterior pertenece a una racha, agregar la celda a la misma racha.
sino crear una nueva racha y agregar la celda.
fase2: generar parches a partir de las rachas
se comienza a procesar a partir de la segunda fila
para cada racha de cada fila
comprobar si la racha tiene columnas en común con alguna racha de la fila anterior.
si comparten alguna columna, se une la racha actual a la racha de la fila anterior.
para cada conjunto de rachas agrupadas
se crea un parche por cada grupo de rachas.
si la forma del parche no es convexa se completan las celdas correspondientes
si el tamaño del parche resultante es menor que el tamaño aceptable se descarta
fase3: generar *clusters* a partir de los parches
para cada parche
crear un *cluster*
asignar al *cluster* el ancho, largo, área y centro

Cuadro 3.3: Algoritmo de *clusterización* utilizado para generar *clusters*.



(a) 10 unidades de feromona. (b) 20 unidades de feromona. (c) 30 unidades de feromona. (d) 40 unidades de feromona. (e) 50 unidades de feromona.

(f) 60 unidades de feromona. (g) 70 unidades de feromona. (h) 80 unidades de feromona. (i) 90 unidades de feromona. (j) 100 unidades de feromona.

Figura 3.12: Análisis de diferentes umbrales de concentración de feromona a las 6.000 iteraciones de un partido con el equipo Australia.

Por otra parte, se realiza un análisis de la evolución de la concentración de feromona sobre el campo de juego a lo largo del partido para diferentes equipos. La figura 3.13 muestra los resultados obtenidos para tres de ellos, a partir de los cuales se obtiene una aproximación lineal ($y = 0,423x + 0,7731$) para la evolución de la concentración de feromona durante el primer tiempo de juego. Esta aproximación lineal permite, una vez establecido el umbral para un tiempo determinado, ajustar su valor para cualquier otro período de procesamiento.

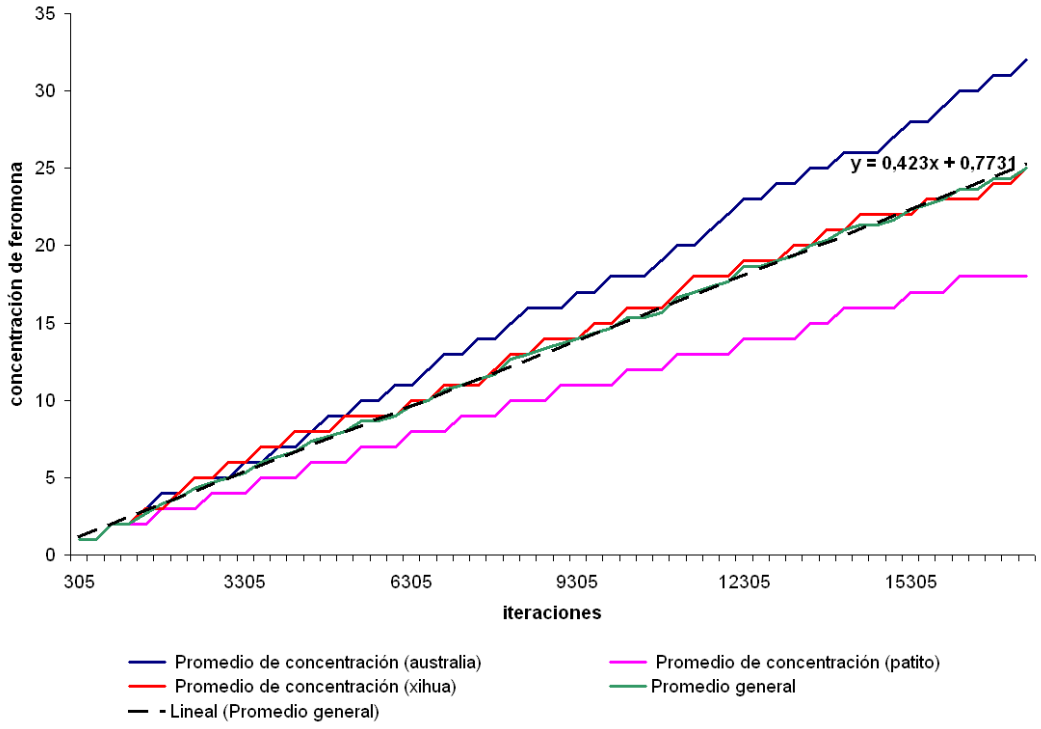


Figura 3.13: Evolución de la concentración de feromona durante el primer tiempo de juego.

A modo de ejemplo, para determinar el valor apropiado del umbral cuando se desea generar el resultado a las 4.000 iteraciones, se realiza el siguiente cálculo:

$y = 0,423 * 6000 + 0,7731 \cong 2538$

$y = 0,423 * 4000 + 0,7731 \cong 1692$

$50 \longrightarrow 2538$

$x \longrightarrow 1692 \qquad x \cong \mathbf{33}$

Tiempo de procesamiento

Al igual que con la detección de rastros de feromona, se realiza un análisis de la evolución de los *clusters* para observar su comportamiento en lo referente a la estabilidad de las diferencias y determinar el tiempo necesario para obtener un resultado adecuado.

Se reproducen los partidos jugados con cada equipo generando los *clusters* cada 300 iteraciones al igual que antes. Cada resultado es comparado con el inmediatamente siguiente para calcular su diferencia. Se genera una matriz con las mismas dimensiones que la matriz de rastros de feromona correspondiente, marcando las celdas que pertenecen a algún *cluster*. Estas celdas contienen un valor positivo único, mientras que las demás contienen el valor nulo. A partir de estas matrices se crea la correspondiente matriz de diferencias. A continuación se cuenta la cantidad de celdas con valor positivo, representando la distancia o diferencia entre ambos resultados. La ecuación 3.2 presenta el cálculo realizado, donde M^{dk} es la matriz de diferencias, M^{k} es una de las matrices resultado y M^{k+1} es la siguiente.

$$M_{ij}^{dk} = abs(M_{ij}^{k} - M_{ij}^{k+1}) \tag{3.2}$$

La figura 3.14 muestra una gráfica con la evaluación de diferencias para un partido con el equipo Australia. En todos los casos se generan matrices de 205 filas por 250 columnas, lo cual corresponde a un total de 51.250 celdas por matriz. Se grafican las evoluciones de:

- la cantidad de celdas positivas de las matrices de diferencia;

- el promedio de celdas contenidas en *clusters:* $\frac{\#celdasClustersM_k + \#celdasClustersM_{k+1}}{2}$;
- el cociente entre la primera evolución y la segunda (diferencia relativa al promedio).

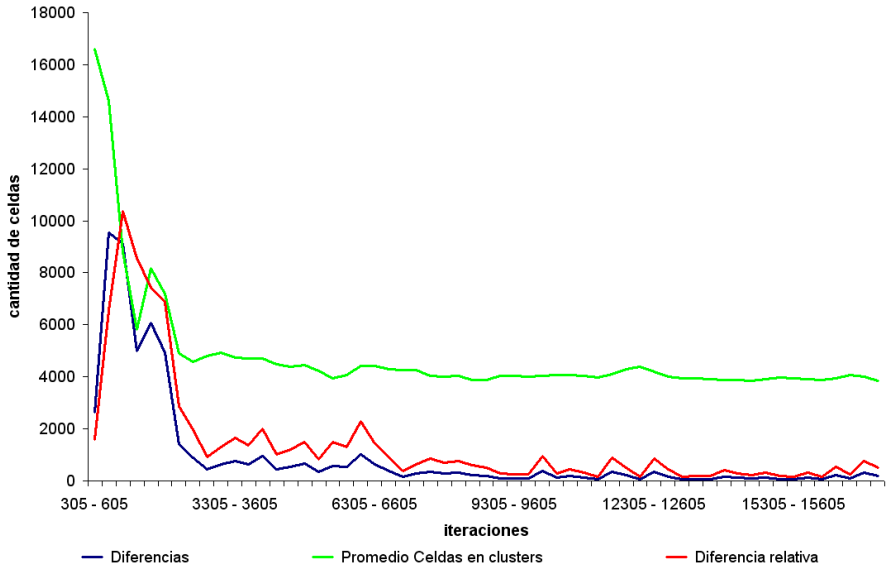


Figura 3.14: Comparación de resultados de *clusterización* para el equipo Australia cada 300 iteraciones.

En general, los resultados muestran una tendencia a la estabilidad a partir de las 3.000 iteraciones, observándose mayor estabilidad a partir de 7.000 iteraciones. Como se planteó anteriormente, es importante minimizar el tiempo de procesamiento para poder utilizar el resultado en la toma de decisiones durante el mayor tiempo posible. En base a ésto y a los resultados observados, se admite como aceptable un período de procesamiento de entre 3.000 y 4.000 iteraciones. En particular, 3.500 iteraciones corresponden a 1/5 de cada tiempo de un partido, permitiendo utilizar los resultados obtenidos durante los 4/5 restantes (4 minutos aproximadamente).

3.6.1.3. Resultado de la calibración

A partir del análisis y la calibración del modelo realizados en las secciones 3.6.1.1 y 3.6.1.2, se presentan los valores finales de los parámetros en el cuadro 3.4.

Parámetro	Valor
Tasa de depósito	5 unidades de feromona
Tasa de evaporación	2 unidades de feromona
Ciclo de evaporación	300 iteraciones
Política de evaporación	Política 2
Umbral de clusterización	33 unidades de feromona
Tiempo de procesamiento	4.000 iteraciones (aprox. 1 minuto)

Cuadro 3.4: Valores finales de la calibración del modelo de formación.

3.6.2. Juego posicional

La resolución del problema de detectar el juego posicional del oponente, planteado al inicio de esta sección (3.6), implica la detección de pases entre los robots del equipo oponente.

Dada la morfología que los robots poseen en la categoría simulada, y la ausencia de agarraderas que les permitan transportar la pelota, se considera que la única forma de realizar pases es colisionando la pelota. De esta forma, un pase es detectado cuando se producen dos golpes o colisiones consecutivas entre dos robots oponentes y la pelota. Resulta relevante, además, considerar los tiros al arco propio, y principalmente aquellos a través de los cuales se concretan goles, ya que ésto ofrece información significativa respecto al comportamiento ofensivo del oponente.

A partir de estas puntualizaciones, la resolución al problema de detección de pases entre los robots oponentes se subdivide en tres etapas:

- detectar los golpes o colisiones entre los robots oponentes y la pelota;
- determinar cuándo se ha convertido un gol oponente;
- consolidar los dos resultados anteriores para identificar pases.

Se analiza a continuación cada una de ellas.

3.6.2.1. Colisiones entre los robots y la pelota

Debido a que el entorno sólo ofrece información sobre las posiciones de los objetos, se modela la solución considerando la dinámica y cinemática de los cuerpos. Cuando un robot colisiona con la pelota, ésta sufre una aceleración positiva durante el tiempo que tiene efecto el golpe. La aceleración promedio puede ser calculada a partir de la variación de velocidad promedio de la pelota entre una iteración y la siguiente, como muestra la ecuación 3.3.

$$acel = \frac{(vel_{final} - vel_{inicial})}{(tiempo_{final} - tiempo_{inicial})} = \frac{\triangle vel}{\triangle tiempo} \tag{3.3}$$

Asimismo, la velocidad puede ser calculada a partir de la distancia recorrida y el tiempo empleado en dicho recorrido, como muestra la ecuación 3.4.

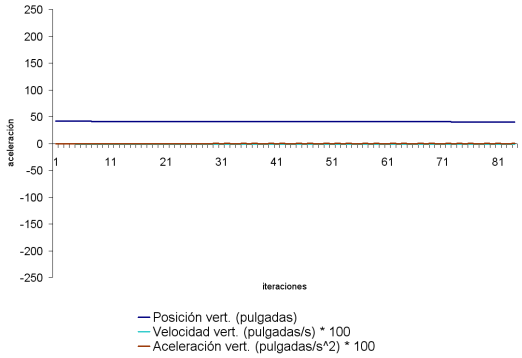
$$vel = \frac{pos_{final} - pos_{inicial}}{tiempo_{final} - tiempo_{inicial}} = \frac{\triangle pos}{\triangle tiempo} \tag{3.4}$$

El manejo del tiempo se simplifica asumiendo que el período entre cada iteración es constante: $\triangle tiempo = K$. A partir de este análisis se deriva que es necesario considerar tres posiciones consecutivas para calcular la aceleración en un momento determinado. Por ejemplo, si se desea determinar la aceleración de la pelota en el momento $t = t_n$, es necesario conocer las posiciones de la pelota en los instantes $t = t_{n-2}, t = t_{n-1}, t = t_n$. Con estos datos se obtiene una nueva fórmula para la aceleración como muestra la ecuación 3.5.

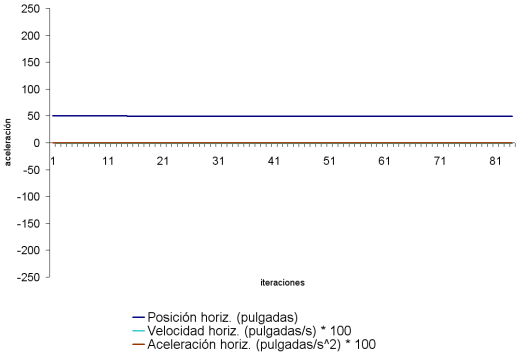
$$acel = \frac{pos_n - pos_{n-1}}{K^2} - \frac{pos_{n-1} - pos_{n-2}}{K^2} \tag{3.5}$$

En condiciones ideales, el valor de la aceleración de la pelota se mantiene en cero mientras no se producen colisiones. Al producirse una colisión, la aceleración adquiere un valor positivo, hasta que el contacto desaparece. Como no se conocen las particularidades del modelo físico utilizado por el simulador oficial y, además, se realizan simplificaciones al manejo del tiempo, es necesario comprobar la factibilidad del modelo propuesto. Las figuras 3.15 y 3.16 muestran parte del análisis realizado, graficando la evolución en los valores de la aceleración, la velocidad y la posición de la pelota durante 80 iteraciones consecutivas. La figura 3.15 representa la pelota en reposo y sin colisiones mientras que la figura 3.16 muestra la pelota con un movimiento vertical y una colisión sobre ésta a partir de la iteración 25. Los resultados del análisis permiten concluir que el modelo se adapta a las necesidades del sistema y permite resolver satisfactoriamente el problema de la detección de colisiones entre los robots y la pelota.

Las posiciones de los objetos en el entorno son expresadas a partir de dos coordenadas, por lo que la aceleración de la pelota se calcula en dos componentes: aceleración en el eje *0x* y aceleración en el eje *0y*. Para la detección de colisiones se toma el máximo de ambas componentes.

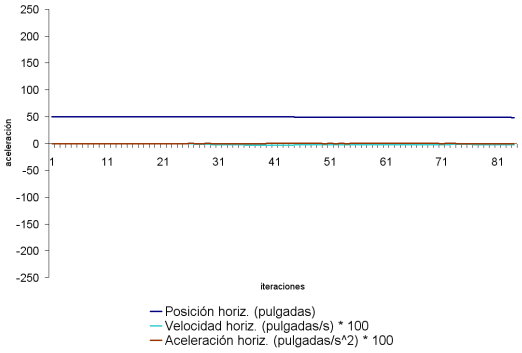


(a) Componentes horizontales de la posición, la velocidad y la aceleración de la pelota.

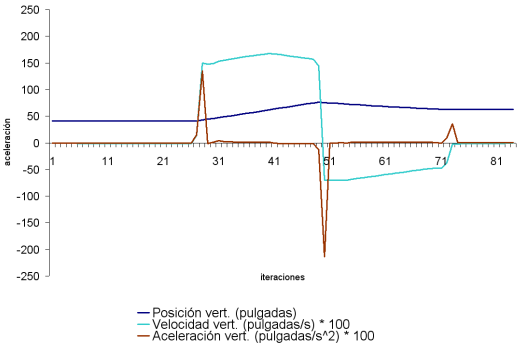


(b) Componentes verticales de la posición, la velocidad y la aceleración de la pelota.

Figura 3.15: Gráfico del movimiento de la pelota. Movimiento practicamente nulo.



(a) Componentes horizontales de la posición, la velocidad y la aceleración de la pelota.



(b) Componentes verticales de la posición, la velocidad y la aceleración de la pelota.

Figura 3.16: Gráfico del movimiento de la pelota. Movimiento y colisión verticales.

Las pruebas realizadas han permitido observar situaciones especiales que deben ser tenidas en cuenta para mejorar los resultados de la detección de colisiones. El cálculo de la aceleración a partir de las posiciones de la pelota es sensible a las imperfecciones del modelo o al movimiento manual de la pelota. Cuando la pelota se mueve manualmente su posición suele variar considerablemente (mucho más de lo que puede variar a partir de una colisión normal), provocando que el cálculo de la aceleración arroje valores muy elevados. La corrección de estas situaciones particulares se resuelve incorporando un umbral o límite máximo para los valores de aceleración permitidos. En lo que refiere a las imperfecciones del modelo, es frecuente que la aceleración así calculada tome valores que no son exactamente cero. Esto conlleva a la incorporación de un umbral o límite mínimo para los valores de la aceleración. Los valores de la aceleración que no se encuentren entre ambos límites son descartados.

3.6.2.2. Goles

Debido a que el simulador no envía información sobre la conversión de goles, es necesario que el sistema logre detectarlos por sus propios medios. Esta detección se realiza a partir del conocimiento de las dimensiones de la cancha y la posición de la pelota, computando un gol al equipo correspondiente cuando la pelota atraviesa completamente la línea de un arco. En la figura

3.17 se muestran dos situaciones diferentes donde se debe determinar si se ha producido un gol. La figura 3.17.(a) muestra una situación que no se considera como gol, ya que la pelota no se encuentra totalmente dentro del arco. Sin embargo, la figura 3.17.(b) muestra un caso donde sí se genera la detección de un gol.

(a) Situación que no se considera gol. (b) Situación que se considera gol.

Figura 3.17: Situaciones a considerar para detectar un gol.

3.6.2.3. Pases entre robots oponentes

Una vez resueltos los problemas de la detección de colisiones y de la detección de goles se debe determinar cómo se modela un pase a partir de las soluciones anteriores. En este sentido, se definen dos tipos de pases:

- *Oponente-oponente* : Son aquellos determinados por la intervención de dos robots. Uno iniciando el pase (colisión origen) y otro recibiéndolo (colisión destino).
- *Oponente-arco* : Son aquellos determinados por la intervención de un solo robot. El pase se inicia con una colisión y finaliza en gol.

Los pases *oponente-arco* resultan convenientes para poder completar las jugadas ofensivas del oponente, distinguiéndolas de aquellas que no concretan un gol.

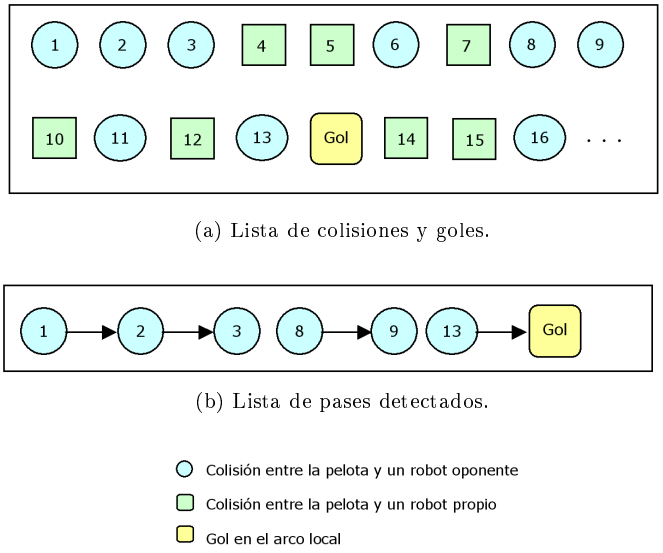
La información de los resultados previos (colisiones y goles) se combina y filtra para obtener sólo aquellos datos relacionados con los robots oponentes. Como resultado se genera una lista de secuencias de pases oponentes, construidos a partir de la lista de colisiones y de los goles detectados previamente.

Cada pase debe cumplir las siguientes restricciones:

- El origen de todo pase debe ser siempre una colisión entre la pelota y un robot oponente.
- El destino de un pase puede ser otra colisión entre la pelota y un robot oponente o puede ser un gol en el arco propio.

El proceso de detección de pases implica recorrer las listas de colisiones y de goles. Si dos colisiones consecutivas corresponden a robots oponentes se genera un pase a partir de ellas. Asimismo, la colisión que representa el destino de un pase puede ser origen del siguiente, generándose de esta forma una secuencia de pases. Las colisiones correspondientes a robots propios u oponentes que no conforman un pase son descartadas. Una secuencia de pases finaliza al encontrarse un gol. Ese gol puede ser destino de un pase si se convirtió en el arco propio y la colisión inmediatamente anterior cumple las restricciones para ser origen del pase. Si no se cumplen ambas condiciones, el gol es descartado para la detección de pases. La figura 3.18 muestra un ejemplo de detección de pases a partir de una colección de colisiones y goles.

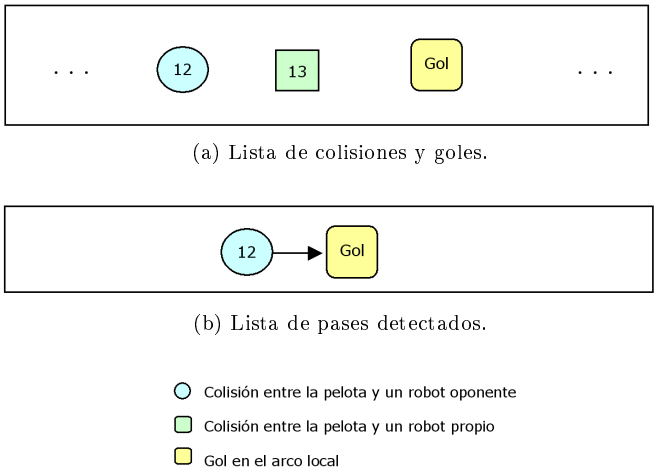
Con la ejecución de muchos partidos se ha observado una gran cantidad de casos en los que el equipo oponente convierte el gol, pero la pelota golpea al arquero antes de entrar en el arco. Con la solución planteada anteriormente estos casos son descartados. Ésto se resuelve incorporando la posibilidad de generar un pase en el caso de encontrar una colisión de un robot oponente, seguida de una colisión de un robot propio dentro del área chica del arco propio, seguida de un gol en el arco propio. La figura 3.19 muestra un ejemplo de esta situación.



(a) Lista de colisiones y goles.

(b) Lista de pases detectados.

Figura 3.18: Ejemplo del proceso de detección de pases.



(a) Lista de colisiones y goles.

(b) Lista de pases detectados.

Figura 3.19: Ejemplo de detección de un pase *oponente-arco* con rebote en el golero.

La figura 3.20 muestra un ejemplo de lo que se espera detectar sobre el campo de juego durante un partido. Las flechas amarillas indican los pases *oponente-oponente* detectados y el sentido de los mismos. Dentro del área chica se observa el rebote de la pelota con el golero propio, que no logra impedir el gol, generándose de esta manera el pase *oponente-arco* correspondiente.

Por último, se realizan pruebas para visualizar los resultados obtenidos a partir del modelo planteado. Se ejecuta una serie de partidos donde se analizan el recorrido de la pelota y las colisiones detectadas durante su trayectoria. La figura 3.21 muestra el resultado de la detección obtenido durante un fragmento (600 iteraciones) de partido contra el equipo Australia. La figura 3.21.(a) presenta la trayectoria de la pelota (puntos negros) y todas las colisiones detectadas contra los robots (puntos rojos). La figura 3.21.(b) presenta las colisiones y los pases oponentes generados (puntos verdes y trayectorias azules). En ambas se puede observar que las posiciones de las colisiones pertenecen a la trayectoria y coinciden con sus puntos de cambio.

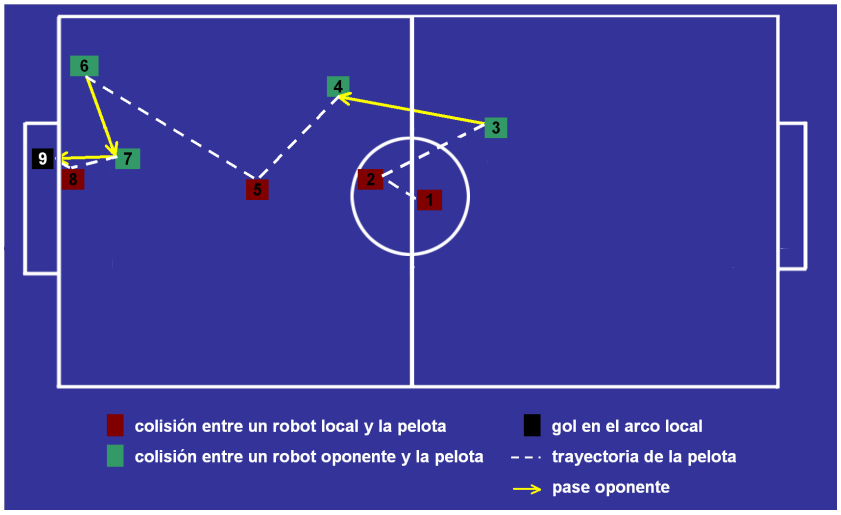


Figura 3.20: Ejemplo de detección de pases.

3.6.3. Comportamiento

Una vez resueltos los problemas de detectar las zonas de mayor actividad y las secuencias de pases de los robots oponentes, se cuenta con la información requerida para resolver el problema de determinar el comportamiento aproximado del oponente. Se deben consolidar los resultados de la formación y el juego posicional para generar información que represente el comportamiento del equipo oponente.

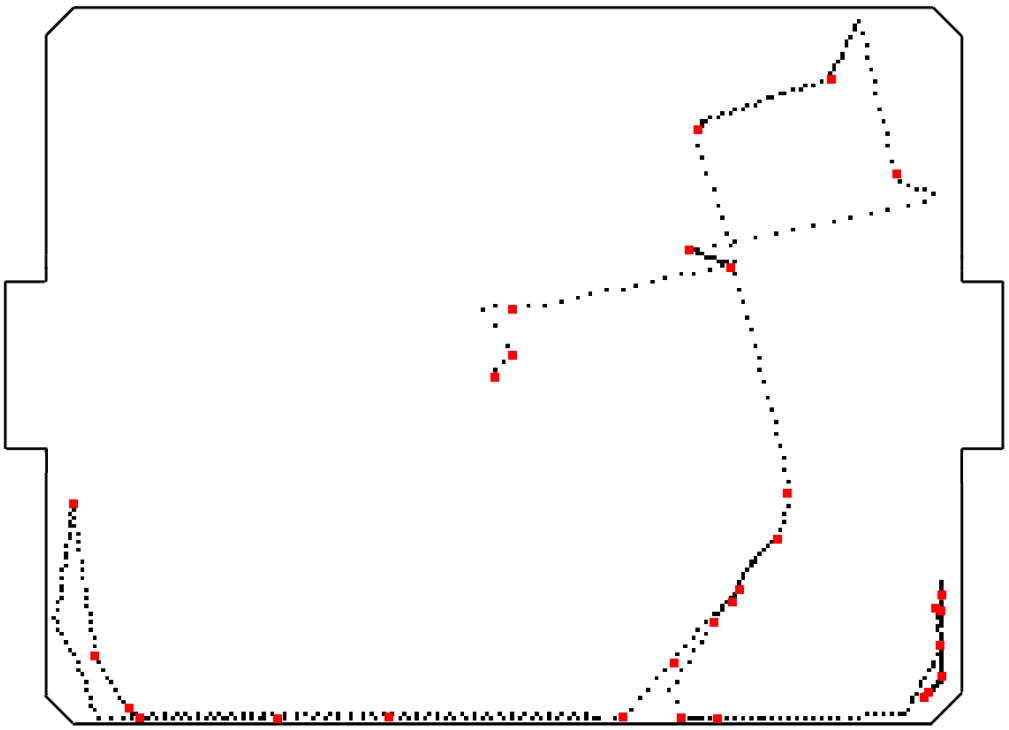
Se genera un grafo para modelar el comportamiento, que posee las siguientes características:

- *Nodos* : Los nodos quedan determinados por las zonas (*clusters*) de la cancha donde se ha detectado mayor presencia de los robots del equipo oponente. Por otra parte, las jugadas pueden terminar en gol, lo cual se desea registrar en el grafo, de manera que se incluye un nodo para representar el arco propio.
- *Aristas* : El peso de las aristas queda definido a partir de los pases realizados entre dos zonas. Existe una única arista entre cada par de nodos.
- *Grafo* : Se modela como un grafo completo. Inicialmente se crea un nodo por cada zona detectada y se agregan todas las aristas con peso nulo. Al procesarse las secuencias de pases se actualizan los pesos de las aristas correspondientes.

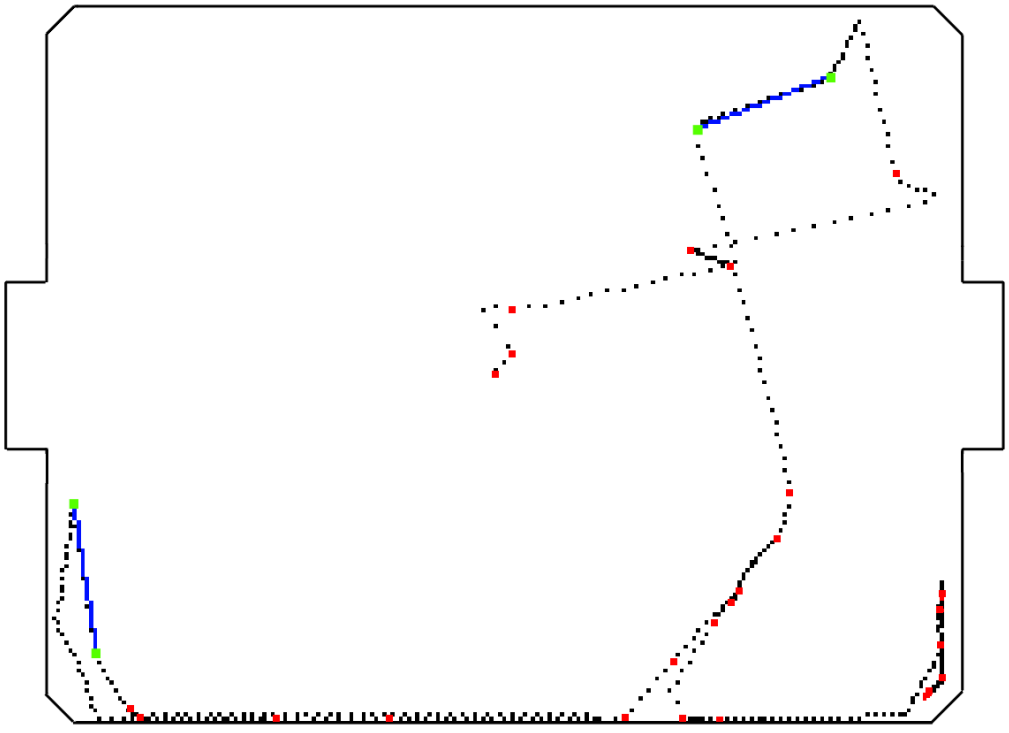
Ejemplo de creación de un grafo

Las figuras 3.22 y 3.23 muestran gráficamente un ejemplo del proceso completo. La figura 3.22 contiene los resultados derivados de la resolución de la detección de la formación y detección de los pases (correspondientes con el ejemplo de la figura 3.20).

La figura 3.23 muestra la creación del grafo a partir de los dos resultados intermedios de la figura 3.22. Sobre la izquierda se muestra la generación de los nodos del grafo a partir de los *clusters* y el arco propio. En la imagen de la derecha se han incorporado las aristas del grafo y los pesos de las mismas. Sólo los pesos de dos aristas han sido actualizados con los pases detectados. El pase que corta la mitad de la cancha ha sido descartado debido a que la posición de la colisión origen no pertenece a ningún nodo del grafo.

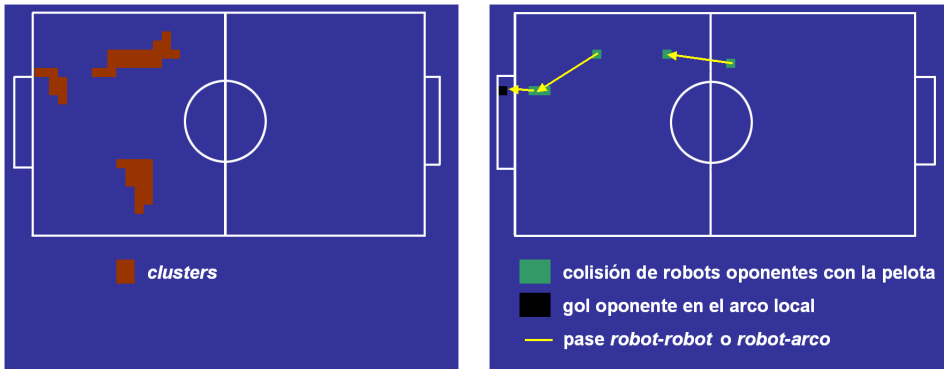


(a) Trayectoria de la pelota y colisiones detectadas.



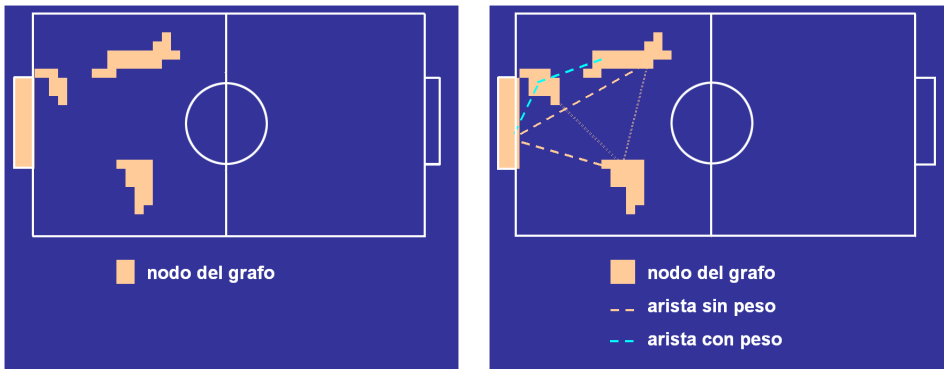
(b) Trayectoria de la pelota, colisiones y pases detectados.

Figura 3.21: Resultado de una prueba de detección de colisiones y pases.



(a) Zonas (*clusters*) de tránsito del oponente. (b) Pases del oponente.

Figura 3.22: Ejemplo de formación y pases del oponente.



(a) Nodos del grafo. (b) Aristas del grafo

Figura 3.23: Ejemplo de generación del grafo que modela el comportamiento.

3.6.4. Características técnicas

La forma en que se ha modelado la solución general se corresponde con un proceso realizado en etapas, una por cada problema planteado, pudiendo realizarse en paralelo las dos primeras (detectar *clusters* y detectar pases). A su vez, cada una de esas etapas puede dividirse en partes, determinando que el proceso completo esté compuesto de varias tareas, que pueden requerir gran cantidad de cálculos y, por tanto, insumir un tiempo mayor al tiempo de respuesta esperado para el sistema en su conjunto.

Como consecuencia de la división de tareas realizada (formación y juego posicional) y el paralelismo inherente de las mismas, resulta apropiado modelar todo el proceso como una red de *pipes* y *filters*, donde cada paso del procesamiento es realizado por un *filter*, permitiendo el paralelismo necesario. Se utiliza el *framework* de predicción presentado en la sección 3.5.2.2, mostrándose en la figura 3.24 la red definida para este proceso.

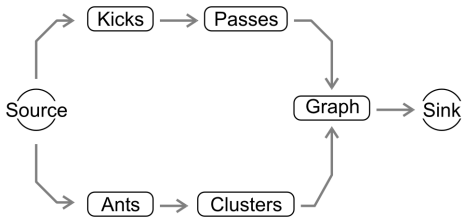


Figura 3.24: Red de *pipes* y *filters* para generar el grafo que modela el comportamiento.

3.6.5. Evaluación

Para comprobar la factibilidad de la solución propuesta a la predicción del comportamiento oponente se realiza una evaluación de los resultados obtenidos.

3.6.5.1. Formación

En la mayoría de las pruebas realizadas los resultados fueron positivos, comprobándose, a través de la observación subjetiva de los partidos, una correspondencia entre los resultados y el comportamiento ofensivo de los equipos. Sin embargo, existen casos para los que la solución no se ajusta adecuadamente. Algunos de los aspectos que pueden presentar los equipos y que inciden negativamente son:

- *Frecuencia* : Comportamientos ofensivos que gocen de cierta regularidad, pero que su frecuencia sea demasiado baja como para ser detectados por el mecanismo.
- *Aleatoriedad* : Comportamientos con alto grado de aleatoriedad que dificulten la obtención de información o provoque que los *clusters* generados no sean representativos del comportamiento ofensivo, y por lo tanto útiles en la toma de decisiones.
- *Adaptabilidad* : Comportamientos que se modifican en base a la capacidad de aprendizaje pueden invalidar los resultados obtenidos.

3.6.5.2. Juego posicional

En lo que refiere a la detección de pases, las pruebas realizadas con diferentes equipos, durante distintos períodos de tiempo, muestran que la cantidad de pases detectados entre oponentes es muy baja y poco frecuente. Las colisiones suelen intercalarse entre propios y oponentes. Se atribuye esta deficiencia al alto grado de dinamismo del entorno, la morfología de los robots, su gran capacidad de reacción y la relación entre su superficie de apoyo y el área de la cancha. La figura 3.25 presenta la evolución en la detección de colisiones y pases en un tiempo completo de partido contra el equipo Australia, donde se verifica la baja tasa de detección de pases entre robots oponentes.

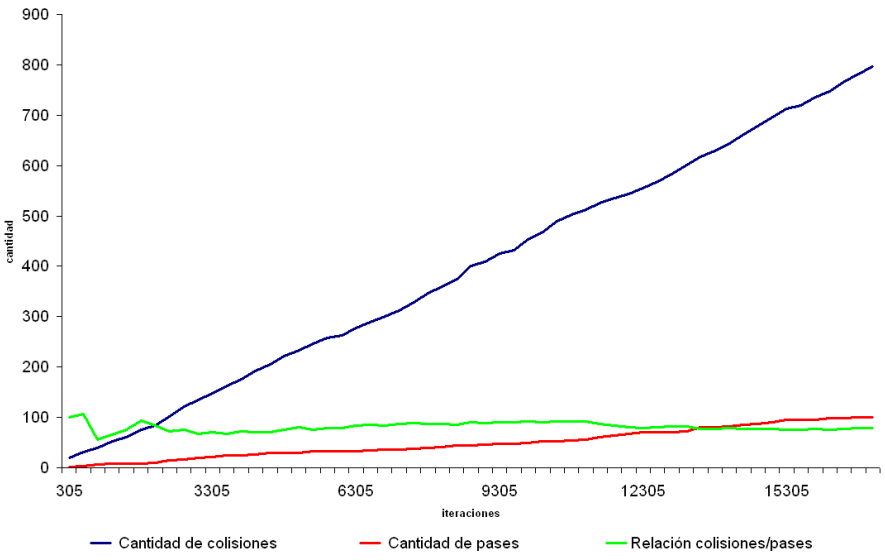
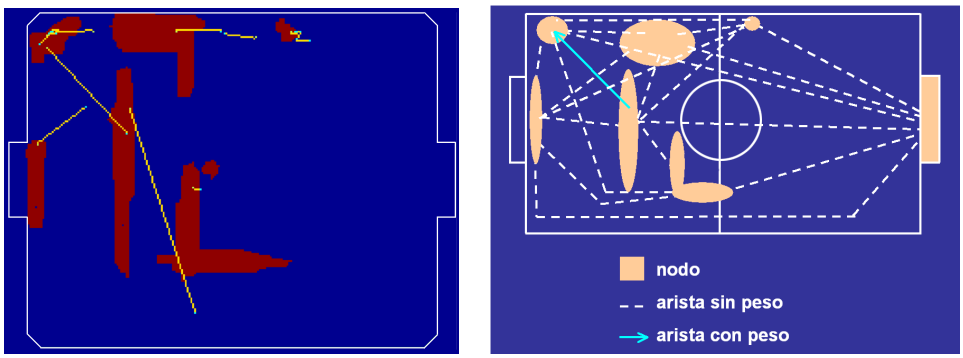


Figura 3.25: Evolución de la detección de colisiones y pases en el primer tiempo de juego contra el equipo Australia.

3.6.5.3. Comportamiento

Finalmente, debido a la baja cantidad de pases que se detectan, son muy pocas las aristas del grafo que modela el comportamiento que poseen un peso positivo. Esta problemática se ve potenciada por los escasos pases que parten de un *cluster* y llegan a otro, decrementándose aún más la probabilidad de tener aristas con peso no nulo.

La figura 3.26 muestra un ejemplo de la detección realizada durante 3.000 iteraciones de juego contra el equipo Australia, donde se ha actualizado el peso de una única arista (indicada con una flecha). Las demás aristas del grafo quedan con peso nulo.



(a) *Clusters* y pases detectados. (b) Representación del grafo generado.

Figura 3.26: Generación del grafo que modela el comportamiento del equipo Australia.

3.7. Toma de decisiones

En términos generales, el objetivo del sistema de toma de decisiones es elegir las acciones que los robots deben realizar de forma tal que se logre alcanzar el objetivo global del equipo: ganar el partido. Para cumplir con ese objetivo, se diseñó un sistema difuso que modela la toma de decisiones en base a reglas que formalizan más fácilmente conocimientos subjetivos e imprecisos.

En ese sentido, el sistema permite al diseñador modelar el comportamiento individual y colectivo a través de reglas que puede construir y que le permiten expresar subjetivamente sus apreciaciones sobre los diferentes aspectos del entorno -que están representados en el modelo del mundo-, y particularmente sobre la información generada durante el reconocimiento del patrón de formación del equipo oponente, para mejorar el desempeño defensivo del equipo FIbRA.

3.7.1. Variables lingüísticas, particiones y reglas

Variables lingüísticas Como se describió en la sección 3.5.2.3, las variables de entrada son utilizadas para manejar los siguientes universos: tiempo, distancias, ángulos, direcciones, áreas, tanteador, entre otros. Estas variables definen el universo de discurso de los conjuntos difusos del sistema, y por tanto, establecen el dominio conceptual sobre el que se realiza la toma de decisiones. Por otro lado, se definen tres variables de salida para representar los siguientes universos: estrategias, roles y acciones. Las acciones son el resultado que el sistema de toma de decisiones retorna cada vez que es invocado, y las estrategias y roles se utilizan como variables de salida de procesos internos a la toma de decisiones que se describen a continuación.

Particiones Clásicamente, en los sistemas difusos se definen particiones que dividen el dominio de los conjuntos según diferentes funciones de pertenencia, permitiendo un análisis más rico de los

valores de las variables lingüísticas. En la figura 3.27 se ilustran tres particiones sobre un universo de distancias. Siguiendo la figura, el sistema podría evaluar una variable según tres criterios (funciones de pertenencia): muy cerca, cerca y lejos.

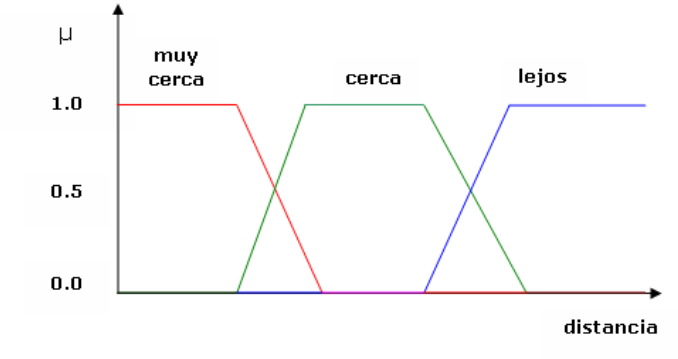


Figura 3.27: Particiones de un conjunto difuso.

A pesar del potencial y flexibilidad que introducen las particiones en los sistemas difusos, para la toma de decisiones del sistema FIbRA se definieron conjuntos mono-particionados. El motivo obedece a dos razones principalmente. La primera se vincula directamente con la cantidad de variables definidas y la complejidad asociada al ajuste de los parámetros de las funciones de pertenencia. La segunda, vincula la complejidad mencionada con el tiempo necesario para realizar dicho ajuste y el alcance del proyecto. Básicamente, las alternativas consideradas fueron tres: definir múltiples particiones y realizar el ajuste manualmente asumiendo el costo de tiempo que eso pudiera requerir; definir múltiples particiones y utilizar alguna técnica de ajuste automático (aprendizaje), también asumiendo el costo en tiempo para seleccionar una técnica y adaptarla al sistema[16]; o definir una única partición.

De esta forma, y continuando con el ejemplo de la figura 3.27, con la simplificación adoptada los conjuntos cuyo universo de discurso son distancias quedan particionados como lo muestra la figura 3.28. Así, distancias que pertenezcan al intervalo [0 .. umbral] serán consideradas como muy cercanas, las que pertenezcan al intervalo (umbral .. minDist] serán consideradas como cercanas y por último todas las que sean mayores que minDist serán consideradas como lejanas. Un criterio similar es utilizado con las demás variables definidas en el sistema.

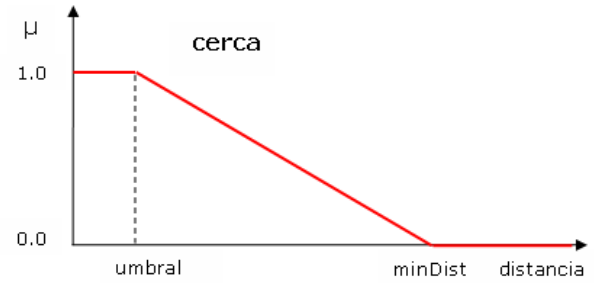


Figura 3.28: Función de pertenencia para variables que representan distancias.

[16] La utilización de aprendizaje automático requiere de un entorno de simulación que permita la ejecución programática de secuencias de partidos o jugadas para posibilitar el entrenamiento. En [ABR05b] se evalúan en detalle las características necesarias o recomendadas que debería poseer un simulador para permitir el entrenamiento automático de un equipo, al tiempo que se explica por qué el simulador oficial de la FIRA no es útil para este propósito.

Reglas Las reglas permiten formalizar razonamientos como "si el tanteador es adverso y el tiempo se está agotando, entonces debe adoptarse una estrategia más agresiva" o "si el robot x está muy cerca del arco, entonces debe asumir el rol de golero" o "si el delantero x está detrás de la pelota y de frente al arco, entonces debe tirar al arco". Para lograr este nivel de expresividad, las reglas se construyen en base a los predicados difusos que conforman el modelo de mundo y los operadores lógicos *And*, *Or* y *Not*. Los operadores *And* y *Or* se implementan como operadores n-arios basados en las funciones *mín(...)* y *máx(...)*, respectivamente. El operador *Not* se implementa como un operador un-ario y se basa en la función 1-μ (donde μ es la función de pertenencia -o predicado- que se aplica a la variable). El operador de implicación lógica se resuelve directamente -sin operadores lógicos- ya que se establece una relación biunívoca entre el conjunto de acciones y el conjunto de reglas que evalúan las utilidades. Así, cada acción que puede ser cumplida por un robot, tiene asociada una regla que define las consideraciones sobre el estado del entorno (predicados) que el sistema de toma de decisiones debe tener en cuenta para evaluar la utilidad de asignarla a un robot en particular.

En la figura 3.29 se muestra como ejemplo una regla que permite considerar el beneficio o la utilidad de asignarle a un robot la acción *Tirar al arco*. En ese caso la regla se construye a partir de los operadores lógicos *And* y *Not* y los predicados *Atascado*, *Atrás de la pelota* y *Sin obstáculos* (que brindan al sistema nociones difusas sobre el nivel de atascamiento, cuán atrás de la pelota -tomando como referencia el arco rival- está un robot y qué tan libre de obstáculos está el camino desde un robot dado a la pelota, respectivamente).

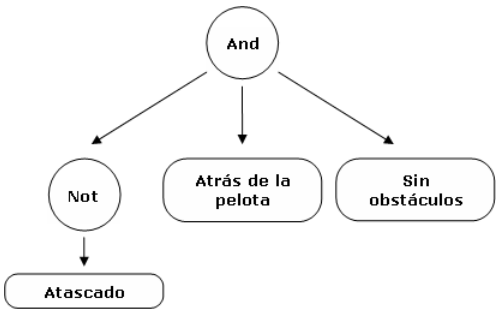


Figura 3.29: Regla que define la función de utilidad de la acción *Tirar al arco*.

Continuando con el ejemplo de la figura, si el predicado *Atascado* evaluara en 0.2, entonces (al aplicar la función que tiene asociada el operador *Not*) se tendría que *Not Atascado* evaluaría en 0.8. Por otro lado, si los predicados *Atrás de la pelota* y *Sin obstáculos* evaluaran en 0.6 y 0.9 respectivamente, la utilidad de la acción *Tirar al arco* (aplicando la función asociada al operador *And*) sería 0.6.

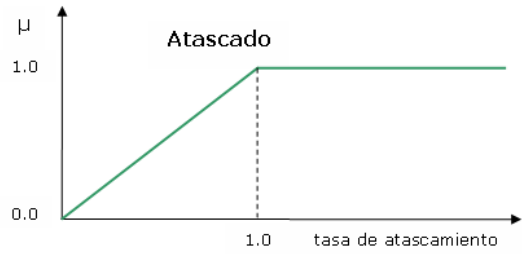
La función de pertenencia que representa la regla asociada a la acción *Tirar al arco* se muestra en la figura 3.30.

En suma, el sistema de toma de decisiones tiene en cuenta -en la evaluación de reglas- cuarenta y nueve predicados difusos (descritos en [ABR06a]) que se utilizan para modelar las utilidades de estrategias de juego, roles y acciones que los robots deben ejecutar.

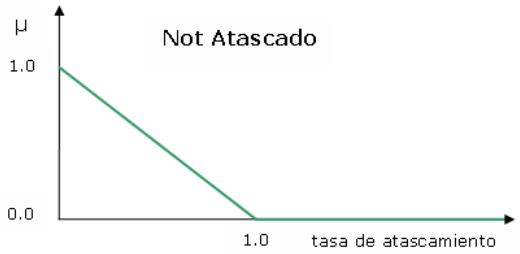
3.7.2. Estructura

Para lograr cumplir con el objetivo principal de ganar el partido, se definen dos objetivos de corto y mediano plazo:

- *Formación dinámica*: Lograr adaptar el comportamiento general del equipo, básicamente a través de su formación, en función de resultados parciales. Es decir, tomando en cuenta el tanteador y el tiempo transcurrido, modificar la formación del equipo tornándolo más ofensivo o defensivo.

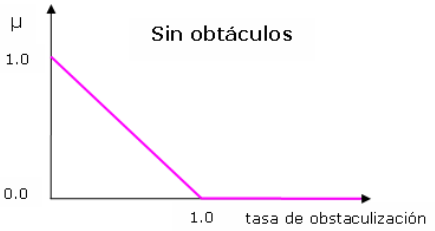


(a) μ-*Atascado*.

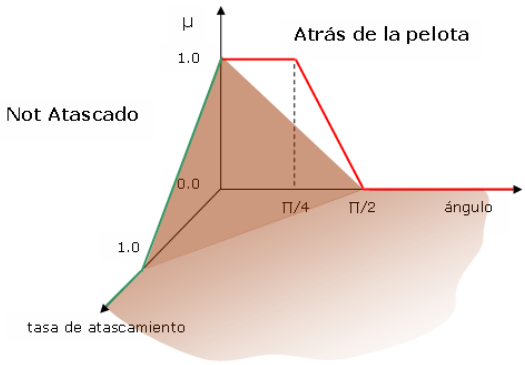


(b) μ-Not *Atascado*.

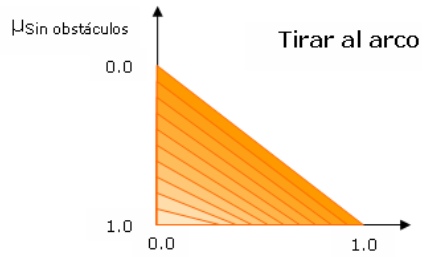
(c) μ-*Atrás de la pelota*.



(d) μ-*Sin obstáculos*.



(e) μ-Not *Atascado* And *Atrás de la pelota*.



(f) Curvas de nivel de μ-*Tirar al arco*.

Figura 3.30: Función de pertenencia *Tirar al arco*.

- *Asignación dinámica de roles*: Lograr que los roles definidos, *a priori*, sean desempeñados en cada momento por los robots que estén en mejores condiciones de hacerlo.

Con estos objetivos se pretende dotar al equipo de un mayor poder de adaptación a las diferentes situaciones de juego, al tiempo que se promueve la cooperación entre sus integrantes, no solo a través de jugadas conjuntas sino también promoviendo el relevo dinámico de posiciones en la formación.

Para cumplir con estos objetivos se define una estructura en capas que muestra en la figura 3.31 y se describe a continuación.

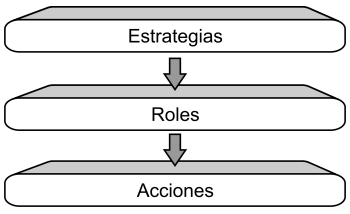


Figura 3.31: Estructura del sistema de toma de decisiones.

Estrategias

Esta capa es responsable de elegir la mejor estrategia para el equipo en todo momento. Básicamente, es responsable de definir el comportamiento global del equipo. Sus decisiones se basan principalmente en nociones difusas relacionadas con la posición de la pelota en el campo de juego (*cancha propia* u *oponente*), el tanteador (*muy malo*, *malo*, *bueno* y *muy bueno*) y el tiempo de juego restante (*suficiente*, *casi insuficiente* e *insuficiente*).

Las estrategias definidas se describen en el cuadro 3.5.

Defensa Activa	Se utiliza cuando la pelota está en el *campo propio* y hay algún jugador propio en condiciones de tomar contacto con la pelota antes que un oponente. De esta forma el equipo maneja la idea de *posesión de la pelota* que le permite decidir acciones para comenzar un nuevo ataque.
Defensa Agresiva	Se utiliza cuando el tanteador no es ni *malo* ni *muy bueno,* el tiempo restante es *casi insuficiente,* y además, o bien el oponente tiene la pelota o bien la pelota está en el campo propio. De esta forma se maneja la noción de *recuperación de pelota* para revertir un resultado apenas desfavorable o cuidar uno apenas favorable.
Defensa Pasiva	Se utiliza cuando la pelota está en el campo propio y ningún jugador propio está en condiciones de tomar contacto con la pelota antes que un oponente. En comparación con la estrategia *Defensa Activa*, en este caso el equipo primero debe recuperar la posesión de la pelota antes de asignar acciones de ataque.
Ofensa Activa	Se utiliza cuando la pelota está en el campo contrario y hay algún jugador propio en condiciones de tomar contacto con la pelota antes que un oponente. Es análoga a la estrategia *Defensa Activa* y permite dar continuidad a las acciones de ataque.
Ofensa Agresiva	Se utiliza cuando el tanteador no es bueno y el tiempo restante es insuficiente como para absorber la diferencia de goles con una estrategia más conservadora. Con esta estrategia el equipo maneja la noción de que con ninguna de las demás estrategias ha logrado superar al oponente y necesita cambiar su comportamiento radicalmente.
Ofensa Pasiva	Se utiliza cuando la pelota está en el campo contrario y ningún jugador propio está en condiciones de tomar contacto con la pelota ya que hay oponentes mejor ubicados para ello. Es análoga a la estrategia *Defensa Pasiva* y permite al equipo reposicionarse en el campo oponente para reanudar un ataque.

Cuadro 3.5: Estrategias.

Roles

Esta capa es responsable de la asignación de roles. Para realizar tal asignación se contemplan los siguientes aspectos:

- *Estrategia de juego*: La estrategia de juego, que se determina previamente en la capa *Estrategias*, debe considerarse muy especialmente en la asignación de roles, ya que, para intentar plasmar la estrategia elegida, se debe contar con una formación adecuada de los robots en la cancha. De este modo, una estrategia ofensiva contará con una cantidad mayor de atacantes que otras más defensivas, y viceversa. Esta influencia de las estrategias sobre la elección de los roles es modelada mediante la utilización de ponderadores que pertenecen al intervalo real [0.0 .. 1.0], que ponderan ciertos roles frente a otros ante la elección de una estrategia en particular. Así, cada rol tiene asociado tantos ponderadores como estrategias se hayan definido y estos representan la utilidad de elegir el rol para implementar cada una de ellas.
- *Adecuación al rol*: A través de las reglas lógicas asociadas a cada rol, se evalúa y elige, para cada robot, el rol que está en mejores condiciones de desempeñar.

Una vez evaluada la adecuación al rol del robot y teniendo en cuenta la utilidad que ese rol tiene en la estrategia previamente definida, se está en condiciones de realizar la elección.

Los roles definidos se describen en el cuadro 3.6.

Atacante	Representa a los robots que realizan acciones de ataque o que por su ubicación en la cancha tienen mejores posibilidades de realizarlas.
Defensa	Representa a los robots que realizan acciones defensivas o que por su ubicación en la cancha tienen mejores posibilidades de realizarlas.
Golero	Representa al robot que defiende el arco.

Cuadro 3.6: Roles.

Acciones

Esta capa es responsable de la asignación de acciones. Una vez definida la estrategia de juego a seguir y los roles que desempeñarán los robots, es necesario asignarle a cada uno una acción concreta. Estas acciones serán a partir de ese momento las metas a corto plazo de cada robot.

Para realizar esta asignación se consideran los siguientes aspectos:

- *Rol del robot*: La idea intuitiva, subyacente, que se toma de la vida real, es que los roles están asociados con las capacidades. Así, en el fútbol de humanos, los jugadores tienen diferentes capacidades que los hacen más aptos para realizar ciertas tareas. A diferencia de éste, en la categoría simulada todos los robots tienen las mismas capacidades físicas y, por lo tanto, lo único que los diferencia entre sí es su posición relativa al entorno. No obstante, una vez que el robot tiene un rol asignado, es útil razonar en términos de que ese robot está en mejores condiciones de desempeñar unas tareas y no otras. Ejemplo de esto sería: el robot que tiene el rol de golero está en mejores condiciones de defender el arco que un atacante (asumiendo que el rol fue asignado correctamente). La forma que se utiliza para representar estos razonamientos es similar a la que se emplea para modelar la influencia de las estrategias en la asignación de roles. En este caso, cada acción tiene asociada tantos ponderadores como roles se hayan definido. Estos ponderadores también representan la utilidad de asignar cierta acción a un robot, dado el rol que deba desempeñar.
- *Utilidad de la acción*: Al igual que los roles, las acciones son evaluadas a partir de la regla que cada una tiene asociada y que modela las condiciones bajo las cuales es deseable ejecutarla.

Por lo tanto, para asignarle una acción a un robot, primero se evalúa la utilidad de todas las acciones y luego, ponderando esas utilidades según el rol, se elige una.

Las acciones definidas se describen en el cuadro 3.7.

3.7.3. Modelado y Ajuste

El objetivo de esta etapa es lograr un modelo que permita maximizar el rendimiento del equipo. El ajuste del modelo se realiza en forma manual y considera tanto las reglas lógicas como los ponderadores.

Reglas Las reglas son ajustadas mediante la observación del comportamiento grupal o individual y el conocimiento que el diseñador posee sobre qué comportamientos son esperados y cuáles no, en un determinado momento o frente a ciertas situaciones. En este caso lo que se intenta determinar es si la regla modela correctamente el comportamiento esperado. En caso contrario, identificar si la diferencia se debe a la especificación de la regla (cómo utiliza los operadores lógicos para combinar los predicados), la falta de predicados para considerar aspectos que no fueron tenidos en cuenta o el sobrante de predicados que evalúan aspectos irrelevantes para el comportamiento, pero que afectan su evaluación.

Despejar	Despeja la pelota desde la zona defensiva hacia una región de la cancha menos peligrosa. Esta acción se apoya en la información generada por el predictor de la evolución de las zonas libres descrito en 3.5.2.2.
Despejar desde el arco	Despeja la pelota en sentido paralelo a la linea de fondo de la cancha aprovechando la forma de las esquinas. Normalmente, es utilizada por el golero aunque en ocasiones también es utilizada por defensas.
Formación defensiva	Implica mantenerse alerta -girando sobre su propio eje- en una posición defensiva cercana al arco. Es utilizada por los defensas, cuando la jugada se está desarrollando "lejos" del arco propio, para monitorear la trayectoria de la pelota y reaccionar lo más rápidamente posible cuando comience el ataque oponente. Se utiliza sólo hasta que finaliza el reconocimiento de la formación del equipo oponente, momento a partir del cual, es sustituida por la *Formación defensiva inteligente*.
Atajar	Ataja la pelota cuando ésta se acerca en dirección al arco. Es utilizada por el golero, que avanzando y retrocediendo en línea recta, cubre el arco desde un extremo al otro.
Formación ofensiva	Es similar a la *Formación defensiva* en cuanto a la estrategia pero es utilizada por atacantes que esperan posicionados cerca del arco oponente el momento para tirar al arco después de recibir un pase o un rebote.
Salir de adentro del arco	En ocasiones los robots son empujados hacia adentro de los arcos. Esta acción presenta una forma especializada de movimiento -que tiene en cuenta la forma geométrica de los arcos- que permite a los robots salir más rápidamente de esas zonas.
Interceptar	Es una acción de propósito general y se utiliza mayoritariamente como forma de aproximación a la pelota, para disputarle la pelota al oponente o en situaciones donde otras acciones más específicas no serían adecuadas.
Pasar	Pasa la pelota a la posición donde se encuentra otro robot del equipo o a zonas de la cancha adonde el receptor pueda llegar fácilmente.
Recibir	Es la contrapartida de *Pasar*. Posiciona un robot de tal forma que reciba la pelota enviada por otro de su equipo o proveniente de un rebote.
Ir al arco	Moviliza un robot hacia el centro de la línea de su propio arco.
Tirar al arco	Tira al arco teniendo en cuenta qué zona está más desprotegida.
Formación defensiva inteligente	Es similar a la *Formación defensiva* en cuanto a los movimientos de monitoreo de la trayectoria de la pelota que realiza. Sin embargo, las posiciones donde se ubican los robots no se determinan en forma estática sino a partir de la información generada por el proceso de reconocimiento de la formación del equipo oponente (ver sección 3.6).
Desatascar	Implica que el robot se movilice procurando liberarse de un atascamiento. Esta acción se basa fuertemente en la información generada por el detector de obstáculos que se describe en la sección 3.5.2.2.
Jugadas especiales	*Saque de arco*, *Penal*, *Tiro libre*, *Pique*, *Comienzo*: Se utilizan durante muy pocas iteraciones luego de un fallo arbitral. Su único propósito es que los robots se comporten de acuerdo a las reglas en este tipo especial de jugadas.

Cuadro 3.7: Acciones.

En la figura 3.32 se muestra una variante de la regla asociada a la acción *Tirar al arco* que refleja esta situación. En ella el predicado *Cerca del arco propio* es irrelevante para la decisión de elegir la acción, sin embargo puede incidir negativamente en situaciones donde sería deseable elegirla, o positivamente en situaciones donde otras acciones fueran más aconsegables. Por otro lado, esta variante no toma en consideración los obstáculos que pueden existir entre el robot y la pelota (empobrecimiento del modelo de la acción), lo cual puede ocasionar que se elija la acción en situaciones donde el robot no podría completar satisfactoriamente su meta.

Ponderadores Durante el proceso de toma de decisiones, que se realiza en tiempo de ejecución, los ponderadores son considerados individualmente por su valor. Sin embargo, en la etapa de ajuste

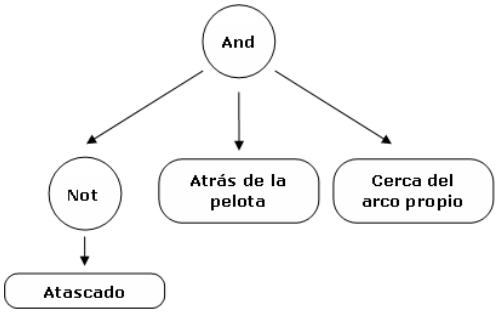


Figura 3.32: Variante de la regla asociada a la acción *Tirar al arco.*

de dichos valores, la atención se enfoca en la diferencia entre ellos. Es decir, lo que se pretende es modelar la utilidad relativa que tiene usar un rol con relación a otros en la aplicación de una estrategia concreta (también vale el razonamiento para acciones y roles). A tales efectos, además de determinarse si un rol dado es más o menos importante que otros, debe determinarse cuánto más o menos importante es. De esta forma, las utilidades relativas quedan representadas en las diferencias entre ponderadores y no en sus propios valores.

Un ejemplo de ajuste podría ser: si se considera la estrategia *Ofensa Activa,* sería razonable pensar que el rol *Atacante* debe ser más importante o útil que los roles *Golero* y *Defensa*. Sin embargo, la diferencia de utilidades entre *Atacante* y *Golero* parece ser más notoria que entre *Atacante* y *Defensa*. La utilidad relativa del rol *Atacante* con respecto a *Golero* es mayor que con respecto a *Defensa* en esa estrategia. Una configuración que modele esta noción intuitiva podría ser: {*Ofensa Activa*: *Atacante*=1.0, *Defensa*=0.7, *Golero*=0.4} denotándose el peso relativo de la utilización de *Atacante*, *Defensa* y *Golero*, que será tenido en cuenta por la toma de decisiones cuando tenga que asignar roles durante la estrategia *Ofensa Activa*.

De este modo, el ajuste que se realiza sobre los valores de los ponderadores, además de reflejar todas las nociones intuitivas del diseñador del equipo, debe respetar ciertas restricciones que se describen a continuación:

- *Rango*: Todos los ponderadores deben tener el mismo signo y pertenecer a un único rango de valores[17] para garantizar que se pondera siempre en el mismo sentido y que la diferencia radica en el valor asignado y no en lo que se esté ponderando.
- *Ciclos o bucles*: Para garantizar la coherencia y evitar conflictos en la toma de decisiones, los valores de los ponderadores no pueden conducir a ciclos de utilidad relativa entre los objetos evaluados (roles o acciones)[18].

Con el objetivo de garantizar que la configuración de los ponderadores siempre respete las restricciones mencionadas, cada vez que se realiza un ajuste de una configuración, se verifican las mismas mediante un proceso de verificación/optimización *offline*. Este proceso implica que todas las restricciones asociadas a cada ponderador de roles y acciones estén representadas en forma matricial (conformando un sistema de ecuaciones) y se realiza mediante la aplicación del método *Simplex Revisto* [dIO06, Mur83], que toma como entrada dichas matrices. El resultado de este proceso de control, en primer lugar permite conocer si con la nueva configuración se sigue manteniendo la factibilidad del sistema de ecuaciones, y en segundo lugar puede ser una optimización de la configuración propuesta, para el caso de configuraciones factibles -desde el punto de vista de las restricciones- pero no óptimas[19].

[17] En el equipo FIbRA se utilizó el rango de valores reales [0.0 .. 1.0].

[18] Un ciclo puede entenderse como: $A<B<C<A$ (donde A, B y C son roles o acciones y $<$ el operador de utilidad relativa, con $x<y$ implicando que la utilidad de x es menor que la de y).

[19] Por no óptimas, en este contexto, deben entenderse aquellas configuraciones factibles que son pasibles de ser mejoradas a través del incremento de los valores de sus ponderadores.

3.7.4. Proceso de toma de decisiones

El proceso global de toma de decisiones se divide en tres etapas: *Emborronado*, *Evaluación de reglas (Toma de decisiones o Inferencia difusa)* y *Desemborronado*. Estas etapas son utilizadas en la mayoría de los sistemas difusos, formando un ciclo de ejecución {*Emborronado*▷*Toma de decisiones*▷*Desemborronado*} que se repite cada vez que el sistema es requerido. Este enfoque (clásico) aplicado en SimuroSot requeriría la ejecución de cinco ciclos completos, uno por cada robot para el que es necesario decidir una acción. A continuación se describen brevemente cada una de las etapas mencionadas.

Emborronado

Consiste en convertir los datos numéricos de entrada (información del entorno y de la predicción) en conjuntos difusos que representan los distintos universos de discurso que el sistema tiene en cuenta para decidir.

Este proceso se realiza mediante la utilización de los predicados difusos o borrosos que integran el modelo de mundo presentado en la sección 3.5.

Evaluación de reglas

Consiste en evaluar las reglas definidas en el sistema y es, en definitiva, la etapa donde se realiza la toma de decisiones propiamente dicha.

Este proceso tiene lugar para cada una de las capas que se presentaron en la sección 3.7.2, resolviéndose primero la estrategia que debe utilizar el equipo; luego, teniendo en cuenta esa decisión, se resuelve la asignación de roles; y finalmente, se asignan las acciones individuales teniendo en cuenta los roles asignados.

Desemborronado

Implica, generalmente, la conversión o pasaje de los resultados difusos o borrosos, obtenidos en la etapa de evaluación de reglas, en valores utilizables como salida del proceso de toma de decisiones.

En este caso, la salida del sistema son las tareas o acciones que cada robot debe ejecutar durante el siguiente paso de simulación. El criterio utilizado para elegir las acciones es el de *primer máximo* (utilizado en sistemas tipo *Mamdani*[20]) con lo cual se elige, para cada robot, la primer acción cuya evaluación represente un máximo global.

3.7.4.1. Optimización del proceso

Como se ha mencionado en secciones anteriores, la categoría simulada no presenta restricciones fuertes al tiempo de respuesta. También se ha dicho que ceñirse al tiempo reglamentario estipulado (16ms) es una forma de mantener próxima la posibilidad de reutilizar en entornos reales los sistemas desarrollados en entornos simulados. En tal sentido, se ataca el problema de la eficiencia en la toma de decisiones, fundamentalmente, a través del ahorro de cálculos. A continuación se describen los aspectos del proceso global de toma de decisiones sobre los que se adoptaron decisiones de diseño para lograr un manejo más austero de los recursos:

- *Emborronado a demanda*: Como cada valor de cada variable de salida (estrategia, rol o acción) tiene asociada una única regla de evaluación, es posible conocer de antemano qué predicados borrosos serán necesarios para evaluar dicha regla. De este modo, el emborronado se realiza paulatinamente, evaluando predicados a medida que son necesarios para realizar la evaluación de diferentes reglas. Este criterio puede ser útil en situaciones donde no se evalúan todas las reglas, evitándose por tanto, realizar más emborronados que los estrictamente necesarios.

[20] Los sistemas difusos o borrosos suelen clasificarse según cómo resuelven la etapa de inferencia difusa. Dos propuestas muy utilizadas son las realizadas por Ebrahim Mamdani y Michio Sugeno. Los sistemas que utilizan estas propuestas son clasificados como Sistemas tipo Mamdani o Sistemas tipo Sugeno, respectivamente.

- *Cache*: El proceso de emborronado contempla la memorización de resultados que, eventualmente, pueden reutilizarse en el mismo paso de simulación. A tales efectos, el resultado de la evaluación de los predicados es almacenada temporalmente mientras se realiza el proceso completo de toma de decisiones, y reutilizado cuando se evalúan otras reglas que requieren la misma información. Este criterio contempla el hecho de que el conjunto de acciones (conjunto de reglas) es el mismo para todos los robots y, por lo tanto, es muy factible que se puedan reaprovechar las evaluaciones realizadas. Otro caso que también queda contemplado, es el de predicados que se utilizan en más de una regla.

- *Evaluación perezosa*: Las reglas lógicas se evalúan utilizando el criterio de circuito corto o *lazzy* que permite cortar la evaluación frente a situaciones como: $Or(1.0, ...)$ y $And(0.0, ...)$, donde alcanza con evaluar el primer predicado para retornar el valor de toda la regla.

- *Sistema tipo Mamdani*: El desemborronado se realiza utilizando el criterio de selección del *primer máximo*. Esto, complementado con una evaluación *lazzy* -lo que resultaría en un desemborronado *lazzy*- permite acortar la secuencia de evaluación una vez que alguna de las variables de salida (estrategia, rol o acción) es evaluada en 1.0. Además, garantiza la no ambigüedad en la elección de reglas, pues si en el proceso se alcanza más de un máximo se selecciona la regla correspondiente al primero. En el peor caso, son evaluadas todas las acciones para cada robot, como ocurriría en el caso clásico.

Finalmente, para incorporar estas optimizaciones al proceso de toma de decisiones, se utiliza un ciclo de ejecución clásico (*Emborronado*▷*Toma de decisiones*▷*Desemborronado*) que genera resultados parciales. La idea es que en lugar de ejecutarse cinco veces, como en el caso clásico, evaluando todas las posibilidades para cada robot, se evalúe, de a una, cada acción para cada robot. De esta forma, se logra emborronar la mínima cantidad de información, así como evaluar la mínima cantidad de reglas, a costo de ejecutar más ciclos de toma de decisiones por cada paso de simulación.

3.7.5. Evaluación

Los resultados que se muestran a continuación fueron extraídos de un partido jugado contra el equipo Pollito Five II (PFII)[21]. El resultado final del partido fue 3x4 a favor de PFII y la evolución del tanteador se puede observar en la figura 3.33. Este partido fue seleccionado como referencia para la extracción de información, por tratarse de un partido que durante el transcurso del juego fue muy parejo (posesión de la pelota) con predominios alternados de uno y otro equipo, todo lo cual se reflejó en la evolución del tanteador, demandándole al equipo FIbRA la utilización de todas sus estrategias.

Los aspectos de la toma de decisiones que se intenta cubrir con los resultados que se muestran son: adaptación a las diferentes situaciones de juego por parte del equipo como un todo; y comportamientos cooperativos, fundamentalmente, a través de la colaboración entre robots que intercambian roles dinámicamente.

3.7.5.1. Adaptabilidad

Lograr que el equipo sea capaz de adaptarse dinámicamente a los diferentes escenarios que se plantean durante un partido es un objetivo muy codiciado. Detrás de ello está la idea intuitiva de que un equipo que juega siempre de la misma forma, sin importar si está perdiendo o ganando, debería ser menos competitivo que uno que es capaz de distinguir entre ambos escenarios y comportarse diferente en un caso que en otro. [ABR05a]

A continuación se muestran algunos de los resultados obtenidos a partir del relevamiento de información relacionada con el cambio dinámico de estrategias que realiza el sistema, persiguiendo el objetivo antes mencionado.

[21] Pollito Five II es uno de los equipos que participó en el CAFR2006.

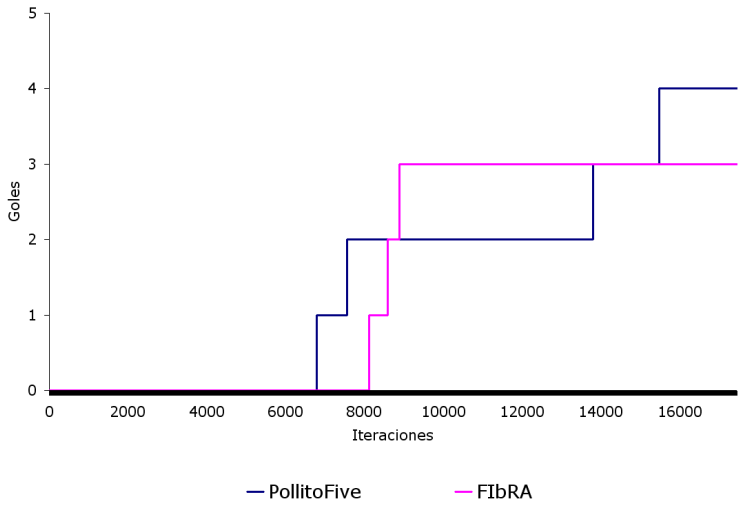
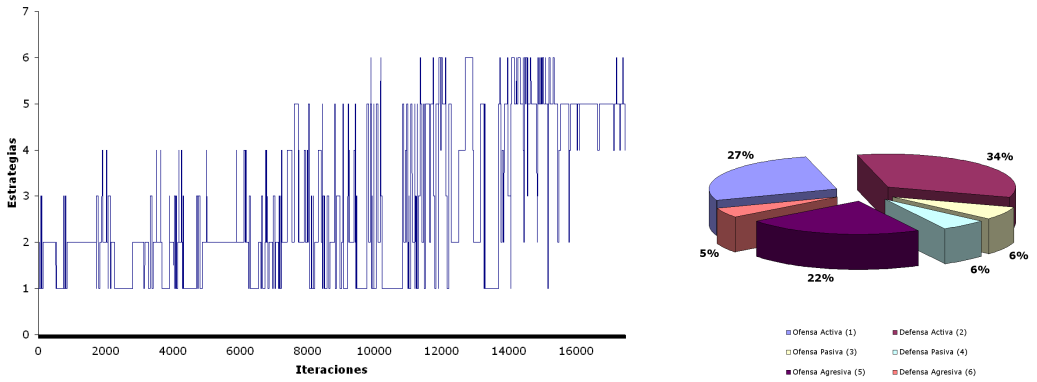


Figura 3.33: Evolución del tanteador entre Pollito Five II y FIbRA.

Cambio dinámico de estrategias La figura 3.34 muestra la secuencia de estrategias que fueron asignadas en cada iteración durante el transcurso del primer tiempo del partido, así como, en porcentaje, la distribución de las mismas según el tiempo que fueron utilizadas.



(a) Evolución de la asignación de estrategias.

(b) Distribución porcentual de la asignación de estrategias.

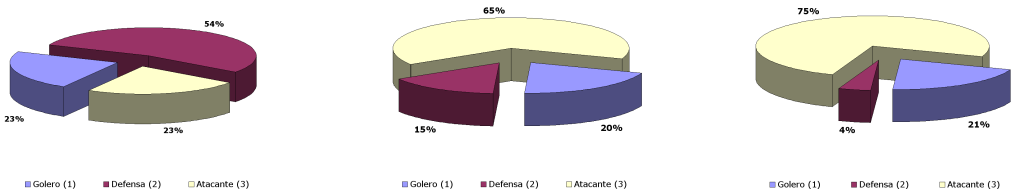
Figura 3.34: Asignación de estrategias.

En la gráfica de asignación de estrategias (figura 3.34.(a)) se puede observar que hasta aproximadamente la mitad del período, el equipo alternó entre las cuatro estrategias de uso general (*Ofensa Activa*, *Ofensa Pasiva*, *Defensa Activa* y *Defensa Pasiva*) que son utilizadas para disputar la pelota, tanto en campo propio como contrario. En cambio, la estrategia *Ofensa Agresiva* fue activada por primera vez unas pocas iteraciones después que el tanteador pasó a ser más adverso (0x2) y casi llegando a la mitad del período, posibilitando revertir rápidamente el resultado parcial. Por otro lado, se puede observar que la estrategia *Defensa Agresiva* no se utilizó hasta tanto el tanteador fue revertido (3x2), manteniéndose en uso incluso durante el período de empate (3x3) y volvió a dejarse de lado luego que el equipo PFII convirtió el cuarto gol sobre el final de la primera mitad.

En la gráfica de distribución porcentual de las estrategias (figura 3.34.(b)) se puede observar que las estrategias *Defensa Activa*, *Ofensa Activa* y *Ofensa Agresiva* fueron las más utilizadas durante el primer tiempo de juego, mostrando que existió una relación entre la agresividad del equipo oponente y la defensa propia, entre la agresividad propia y los goles convertidos y entre los resultados parciales y la intención de revertirlos adjudicando, circunstancialmente, una estrategia más agresiva.

Estos resultados muestran cómo el equipo -a pesar de no haber logrado alcanzar el objetivo global- comenzó el juego utilizando estrategias más conservadoras y luego modificó su comportamiento para intentar, primero, revertir el tanteador superando situaciones adversas, y segundo, cuidar resultados parciales favorables.

Influencia de la estrategia en la asignación de roles Tomando como referencia las tres estrategias predominantes (*Defensa Activa*, *Ofensa Activa* y *Ofensa Agresiva*), se muestra su incidencia en la asignación de roles. Así, en la figura 3.35 se puede ver cómo durante la estrategia *Defensa Activa*, en promedio, más de la mitad de los robots se desempeñaron como defensas. Durante la estrategia *Ofensa Activa*, más del 60 % de los robots adoptaron el rol de atacante y el 75 % durante la estrategia *Ofensa Agresiva*. Estos valores muestran que el cambio en la estrategia no es sólo una cuestión de nombres, sino que efectivamente se modifica la integración del equipo en el sentido que la estrategia propone -más ofensivo o defensivo-.



(a) Durante la estrategia *Defensa Activa*.

(b) Durante la estrategia *Ofensa Activa*.

(c) Durante la estrategia *Ofensa Agresiva*.

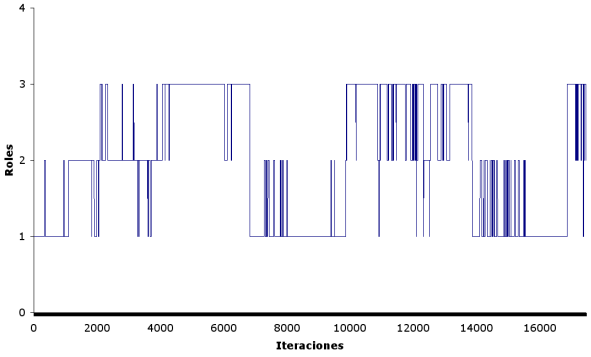
Figura 3.35: Distribución porcentual de roles.

3.7.5.2. Cooperación

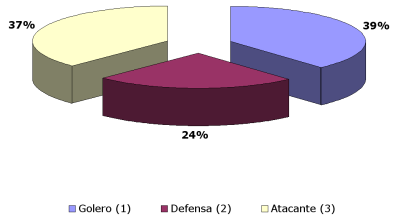
El aspecto cooperativo también es señalado como uno de los más influyentes en la obtención de buenos resultados. En los casos estudio descritos en [ABR05a], la asignación dinámica de roles es utilizada como forma de dotar a los sistemas de mayor adaptabilidad y de habilitar una forma más sutil de cooperación que trascienda la cooperación explícita presente cuando se realizan tareas entre dos robots (realizar un pase: pasar/recibir). En el caso de los equipos de fútbol de robots, la asignación dinámica de roles es un caso de cooperación robótica aplicada para incrementar el desempeño y la robustez de los sistemas. El desempeño es mejorado a través de la asignación que contempla las condiciones de los robots en un instante dado, y selecciona al más apto para desempeñar el rol. En tanto, la robustez se ve reforzada ya que se garantiza que la tareas que son imprescindibles siempre sean realizadas por algún robot (defender el arco).

Intercambio dinámico de roles En la figura 3.36, 3.37 y 3.38 se muestran los resultados de la asignación de roles de un robot por cada línea de la formación inicial[22]. El robot 0 ocupó casi el mismo tiempo en el rol *Golero* que en rol *Atacante*. El robot 1 ocupó casi el 60 % del tiempo en el rol de *Atacante* y *Defensa* casi en la totalidad del tiempo restante. Por otro lado, el robot 3 se desempeñó más como *Atacante*, aunque también como *Defensa* y *Golero*, en ese orden de dedicación.

[22] Al comienzo del primer tiempo el equipo formó con el robot 0 en la posición de golero, los robots 1 y 2 en las posiciones de defensa y los robots 3 y 4 en las de ataque.

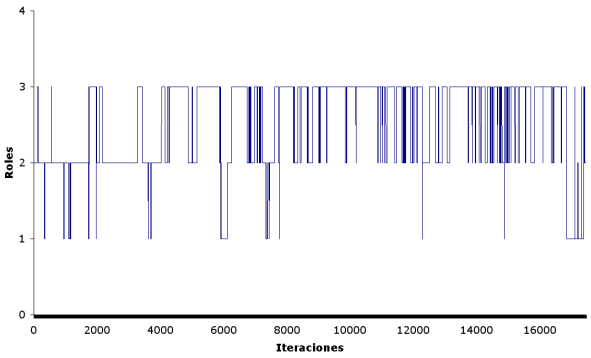


(a) Evolución de la asignación de roles.

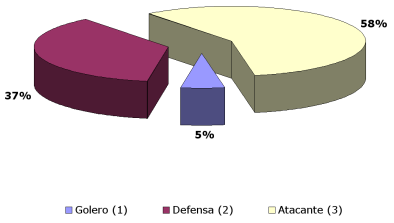


(b) Porcentaje de asignación de roles.

Figura 3.36: Roles asignados al robot 0 en el partido frente a PFII.

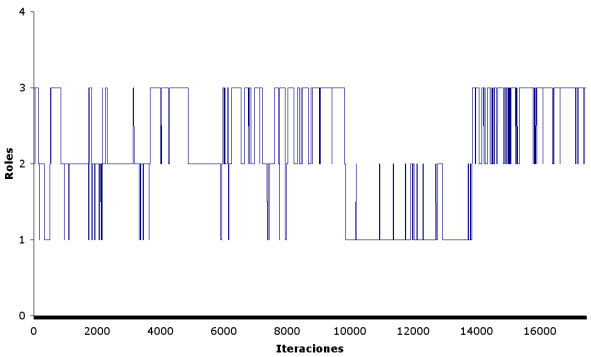


(a) Evolución de la asignación de roles.

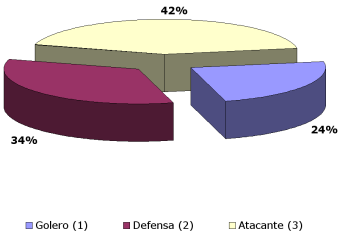


(b) Porcentaje de asignación de roles.

Figura 3.37: Roles asignados al robot 1 en el partido frente a PFII.



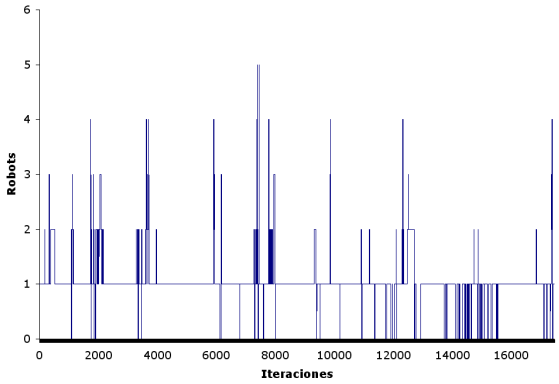
(a) Evolución de la asignación de roles.



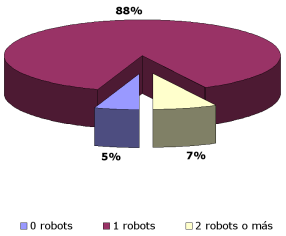
(b) Porcentaje de asignación de roles.

Figura 3.38: Roles asignados al robot 3 en el partido frente a PFII.

Robustez Este aspecto se muestra en la figura 3.39, donde se aprecia que, a pesar de la alta rotatividad de los robots desempeñándose en diferentes roles a lo largo del período, el rol *Golero* estuvo cubierto durante el 95 % del tiempo. Asimismo, se logró una asignación exclusiva (para un único robot) del rol durante el 88 % del tiempo y sólo en el 7 % restante hubo dos o más robots asignados.



(a) Robots asignados al rol.



(b) Porcentaje de no asignación o rol vacante.

Figura 3.39: Asignación del rol golero.

3.8. Planificación de Movimientos

En esta sección se presenta la solución a la planificación de movimientos propuesta y se evalúan sus fortalezas y limitaciones para determinar su impacto cualitativo en la solución global.

3.8.1. Problemática

Antes de describir la solución propuesta es importante conocer, por un lado, qué aspectos inciden en su elaboración y, por otro, qué características generales debería tener.

Características del entorno

Las características del entorno, presentadas en la sección 3.5, son determinantes en cuanto al tipo de problema de planificación de movimientos que se pretende resolver:

- Es dinámico: la planificación de movimientos se plantea como un problema de planificación dinámica de trayectorias (ver sección 2.4).
- Proporciona toda la información necesaria para localizar a cada robot y se conoce completamente el "mapa" donde se mueven (a través del reglamento; ver [ABR05c]): la solución -que bajo otras hipótesis podría comprender tareas que ayuden a descubrir el entorno[23]- debe resolver únicamente la *navegación* de cada robot.
- Es estocástico: debido a que se desconoce la forma en que el oponente resuelve su juego, cualquier tipo de predicción basada exclusivamente en la cinemática o la dinámica de los

[23] En [CLH+05] se realiza una clasificación del tipo de tareas que deben resolver los algoritmos de planificación.

cuerpos pierde su validez con el paso del tiempo. Si a esto se le suma el hecho de que los robots pueden cambiar rápidamente de dirección o sentido, la vigencia de la predicción se vuelve aún más al corto plazo.

- Es multiagente: la solución debe, por lo menos alcanzar un nivel de cooperación que evite, por ejemplo, que dos robots propios intenten realizar la misma tarea al mismo tiempo.

Características de los robots

La morfología de los robots simulados se ajusta a la de los robots de la categoría MiroSot [ABR05c]. Sin embargo, los detalles de su construcción (exceptuando sus dimensiones) se desconocen *a priori*, debido a que no están definidos en el reglamento, y deben ser determinados experimentalmente.

Existen tres características de los robots que restringen la geometría de las trayectorias:

- *Sistema no-holonómico*[24] : Los robots pueden moverse de manera autónoma hacia adelante y hacia atrás, pero no hacia los lados. Esta característica limita su libertad de movimiento y debe ser contemplada en la determinación de la geometría de las trayectorias.
- *Posición del eje* : A partir de la experimentación (ver sección 3.8.4) se puede determinar que el eje de rotación de las ruedas se encuentra desplazado horizontalmente respecto al centro del robot (ver figura 3.40). Esta asimetría determina que la geometría de las trayectorias sea distinta cuando se evalúa alcanzar el objetivo marcha adelante o marcha atrás.
- *Límite de velocidades* : La velocidad de cada rueda está limitada al intervalo [-125, 125] *unidades de velocidad*[25]. La geometría de la trayectoria debe permitir maximizar la velocidad sin comprometer la posibilidad de recorrerla.

Por otro lado existe una característica que impacta en la habilidad y en la capacidad de cooperación de los robots: la forma del chasis. Las caras planas del chasis, ausentes de agarraderas, dificultan el traslado de la pelota y la realización de pases[26] entre los robots.

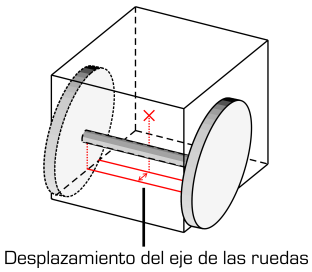


Figura 3.40: Desplazamiento del eje de las ruedas respecto al centro del robot.

Características de la solución

Las características del entorno y de los robots especifican qué problema debe ser resuelto. Cada solución particular determina cómo debe resolverse: a partir de alguna ecuación que permita

[24] En [CLH+05] se define en profundidad el concepto de sistema *no-holonómico*. Sin embargo, a los efectos de este documento alcanza con comprender la definición más básica: un sistema es *no-holonómico* si en un paso de tiempo no puede moverse de manera autónoma en todas las direcciones.

[25] El simulador oficial establece el rango [-125, 125] como dominio de velocidades que se pueden utilizar para mover los robots. Se utiliza el término *unidades de velocidad* debido a que no hay documentación que indique en qué unidades están expresados estos valores [ABR05c].

[26] La forma del chasis diferencia a SimuroSot de MiroSot, donde se permite que los robots posean una leve curvatura en su cara frontal para facilitar el traslado de la pelota. También la diferencia de otras ligas simuladas (p. ej.: *RoboCup Simulated League*), donde los robots sí poseen agarraderas que facilitan el manejo de la pelota.

maximizar el desempeño de los movimientos contribuyendo al logro del objetivo global que es ganar el partido. Esta ecuación deberá hacer un balance entre qué restricciones deben ser tenidas en cuenta, cuán aproximado debe ser el modelo del mundo y hasta qué punto es útil lograr cooperación entre los robots.

Por otro lado, el objetivo de aplicar la solución en una competencia exige la elección de un algoritmo de planificación de movimientos eficiente.

Por lo tanto, un algoritmo de planificación de movimientos óptimo, además de administrar correctamente las restricciones impuestas por el entorno, debería ser capaz de:

- maximizar la precisión en los movimientos;
- minimizar el tiempo de llegada al objetivo;
- maximizar la potencia de los movimientos;
- minimizar el tiempo de procesamiento[27].

3.8.2. Solución

Para la resolución del problema de planificación de movimientos se estableció como requerimiento no funcional (ver [ABR06b]) la reutilización de un algoritmo de planificación construido en otro proyecto de similares características[28]. Su incorporación al sistema FIbRA mantiene los principios arquitectónicos que rigen a otros componentes del sistema.

3.8.2.1. Características del algoritmo reutilizado

Una vez determinada la acción que deberá realizar un robot, el proceso que resuelve su movimiento se divide en dos etapas:

1. Se calculan las velocidades de las ruedas considerando únicamente las restricciones cinemáticas de los robots.
2. Se alteran las velocidades, en función de otras restricciones del entorno.

Primera etapa

El objetivo de la trayectoria puede ser algún punto fijo de la cancha o la pelota.

Cuando el objetivo es fijo, la configuración objetivo no varía a medida que el robot recorre la trayectoria para alcanzarla.

Cuando el objetivo es la pelota, se busca llegar con una configuración[29] apropiada que permita desviarla hacia alguna dirección, teniendo en cuenta que su posición puede cambiar con el tiempo. Para algunas acciones, es importante que esta dirección interseque un punto particular de la cancha, como ser el centro de una zona libre de obstáculos, algún punto dentro del arco oponente o un punto de la cancha donde otro robot podría recibir un pase. Para otras, alcanza con enviar la pelota en una dirección predeterminada.

El movimiento de los robots se resuelve, en última instancia, utilizando uno o varios algoritmos denominados *controles de movimiento*. Para cada acción, el diseñador define de antemano qué controles de movimiento deben utilizarse. Para flexibilizar la solución, los controles son independientes entre sí y cada uno posee un modelo único que le da soporte a los cálculos. A modo de ejemplo, algunas acciones se resuelven utilizando un control denominado *Direct LV*, basado en el método de Lyapunov[30], mientras que para otras se utiliza un control denominado *Direct Ang*, un algoritmo *ad hoc* que basa su cálculo en la diferencia entre la rotación del robot y el ángulo al punto donde se pretende enviar la pelota.

[27] El término *tiempo de procesamiento* refiere al tiempo que demora el algoritmo de navegación en analizar la información proveniente del entorno y determinar una trayectoria.

[28] Se trata de un proyecto de grado de la Universidad de la República, que consistió en la construcción de un equipo de fútbol de robots denominado "FRUTo". En [ABR05a] se presenta como caso de estudio.

[29] El concepto de configuración de un robot se define en la sección 2.4.

[30] El método de Lyapunov determina trayectorias suaves que convergen rápidamente hacia el objetivo. La implementación de este método se describe en [ABR05a]. En [Ind01] se hace referencia a sus ventajas y desventajas.

Los modelos detrás de los controles de movimientos tienen aspectos en común:

- Asumen que la configuración objetivo es fija.
- Sólo tienen en cuenta la configuración actual del robot y la configuración objetivo.
- Avanzan el robot hacia el objetivo calculando las velocidades de las ruedas para moverlo hacia la siguiente configuración.
- Se basan en la cinemática de los robots, considerándolos como sistemas *no-holonómicos*.
- Simplifican el cálculo de la trayectoria al no considerar las dimensiones del robot como una restricción: están concebidos para mover el punto central del robot hasta el punto objetivo. Para mejorar la convergencia se ajustan las ecuaciones para trasladar el punto de aplicación del movimiento hacia el frente del robot.
- Toman en cuenta la restricción que limita la velocidad de las ruedas, pero no integran la dimensión temporal en sus cálculos. Esto significa que en la determinación de la trayectoria no se busca explícitamente minimizar el tiempo de recorrida.

El primer aspecto es una limitante en caso de tomar como objetivo un punto móvil como ser, específicamente, el centro de la pelota. La determinación de la configuración para objetivos móviles depende de cada acción, pero en general se utiliza un mecanismo que puede ilustrarse tomando como ejemplo la resolución de la acción *Tirar al arco*:

1. Se predicen las futuras N posiciones de la pelota. (Ver punto 1. de la figura 3.41).
2. Para cada posición futura P:
   a) Se simula la aplicación del control de movimiento a partir de la configuración inicial del robot, tomando como configuración objetivo ficticia la posición P y la rotación determinada por la dirección entre P y el punto medio del arco.(Ver punto 2.(a) de la figura 3.41).
   b) Si esta simulación determina que, en el lapso que demoraría la pelota en alcanzar la posición P, la configuración del robot pudo acercarse a la configuración objetivo ficticia, entonces:
      1) Se elige a P como la posición a la cual se quiere enviar el robot. (Ver punto 2.(b) i. de la figura 3.41).
      2) Se predice la configuración futura que tendrán los robots propios y oponentes[31] cuando la pelota alcance la posición P, para intentar evitar los elementos que puedan bloquear la línea de tiro. A partir del resultado de esta predicción, se calcula el mejor ángulo de tiro y se determina la dirección hacia donde debería desviarse la pelota. Esta dirección se toma como la orientación objetivo del robot. (Ver punto 2.(b) ii. de la figura 3.41).
      3) La posición y orientación calculadas definen la configuración objetivo; por lo tanto, se da por terminada la búsqueda.
3. Si no se encontró una solución apropiada, se toma como configuración objetivo la última posición predicha de la pelota y la rotación determinada por la dirección entre esta posición y el centro del arco.

[31] De manera abstracta, esta predicción podría verse como la secuencia de configuraciones que tendrían los robots propios y oponentes si, de pronto (en el momento en que se realiza la predicción), los equipos se abstuvieran de enviar órdenes quedando los robots bajo la influencia de una aceleración nula. En esta predicción no entran en juego consideraciones respecto al posible comportamiento de los robots; ni siquiera de los propios. Esto se debe a que, por un lado, se desconoce la forma en que el oponente resuelve su juego y, por otro, también se desconoce la resolución de la trayectoria de los demás robots propios al momento de calcular la predicción.

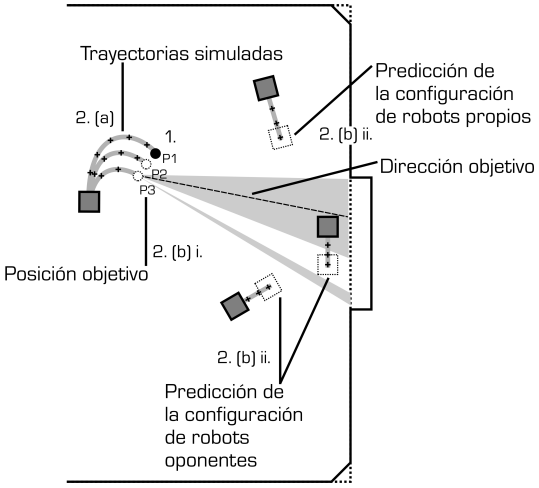


Figura 3.41: Elección de la configuración objetivo para la acción *Tiro al arco*.

Segunda etapa

El algoritmo considera dos restricciones para alterar las velocidades calculadas en la primera etapa:

- *Obstáculos* : Tanto los robots propios como los oponentes e incluso la pelota pueden ser considerados como obstáculos. El diseñador determina, para cada acción, el conjunto de obstáculos a tener en cuenta. En cada punto del recorrido, el algoritmo realiza una predicción sobre la trayectoria de la pelota y las configuraciones futuras de los robots para detectar si habrá colisión con alguno de los elementos del conjunto. En caso afirmativo, se altera la trayectoria mediante el cálculo de una configuración intermedia que permita evitar el obstáculo y a la vez avanzar hacia el objetivo.
- *Atascamientos* : El algoritmo considera que un atascamiento es una restricción que puede actuar sobre un robot durante cierto intervalo de tiempo. Se considera que un robot se encuentra atascado cuando se mantiene durante un intervalo de tiempo en una misma configuración por razones ajenas a él. La detección de un atascamiento depende de las últimas velocidades asignadas al robot, así como de su configuración reciente: de manera abstracta, si durante un lapso arbitrario la configuración del robot no condice con la configuración que debería tener a partir de la asignación de velocidades, entonces se considera que el robot está atascado. La resolución consiste en invertir el sentido de la trayectoria que condujo al robot a una situación de atascamiento.

Integración del proceso

La figura 3.42 muestra la integración de las etapas que componen el proceso que resuelve el movimiento de los robots.

3.8.2.2. Adecuación del algoritmo

Si bien en líneas generales la estructura se mantuvo, para incorporar y adecuar el algoritmo original al sistema se optó por un enfoque de reutilización "caja blanca"[32], debido a que fue necesario ajustar y corregir algunos aspectos:

- *Evasión de obstáculos* : Se corrigió un error en la detección, ya que algunos cuerpos eran considerados obstáculos cuando deberían de haber sido descartados como tales.

[32] El enfoque de reutilización "caja blanca" implica la modificación del código fuente del componente para ajustarlo a las nuevas necesidades.

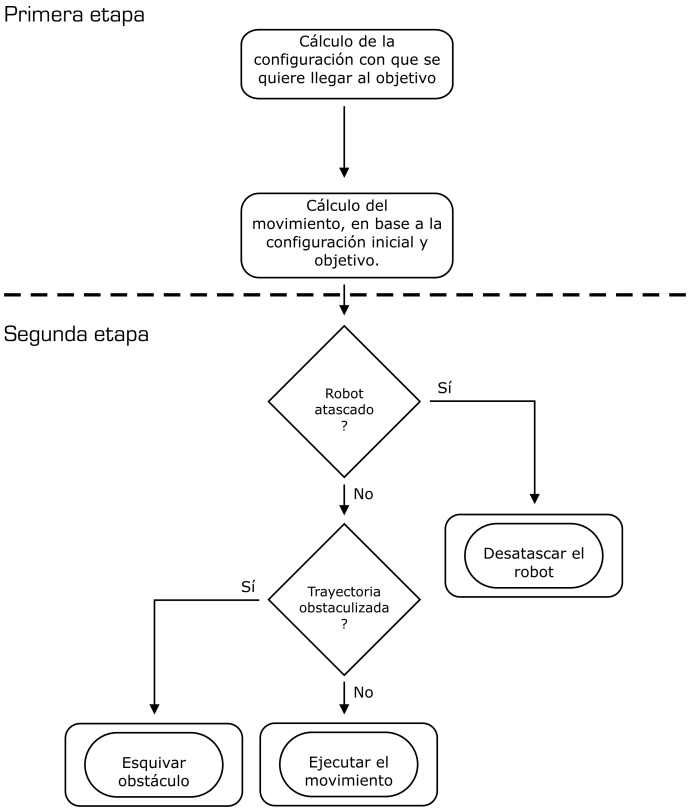


Figura 3.42: Interacción entre los distintos componentes encargados de resolver la planificación de movimientos.

- *Manejo de atascamientos* : Se anuló el manejo de atascamientos del algoritmo reutilizado para evitar interferencias, debido a que FIbRA delega este manejo a la toma de decisiones.
- *Incorporación y adaptación de acciones* : Se incorporaron nuevas acciones y se adaptaron algunas preexistentes, para compatibilizarlas con el conjunto de acciones definidas para FIbRA (en [ABR06a] se especifican los componentes que implementan las acciones del equipo FIbRA, mientras que en [ABR05a] se especifican las acciones del equipo FRUTo).

3.8.2.3. Estructura de la solución

Toda la funcionalidad reciclada fue encapsulada en la capa *Planificación de Movimientos* para independizarla del resto del sistema (ver sección 3.3). Los siguientes párrafos describen las funcionalidades reutilizadas.

Predicción Las funcionalidades de predicción de trayectorias fueron integradas a la capa *Modelo del Mundo* para mantener la coherencia con la filosofía de la solución. A estos efectos, se creó un predictor que reutiliza la predicción de la trayectoria de la pelota, de las configuraciones futuras de los robots propios y oponentes, y de la evolución de las zonas libres. Nuevamente, el sistema fue diseñado para encapsular los componentes involucrados con la resolución de estos cálculos. La figura 3.43 esquematiza este diseño, ilustrando las dependencias entre las acciones, el modelo predictivo del mundo y la planificación reutilizada.

Acciones La elección de las acciones a ser aplicadas sobre los robots es responsabilidad del mecanismo de toma de decisiones descrito en la sección 3.7. Cada acción se relaciona con un componente que la implementa, responsable de resolver el movimiento del robot. Para lograrlo,

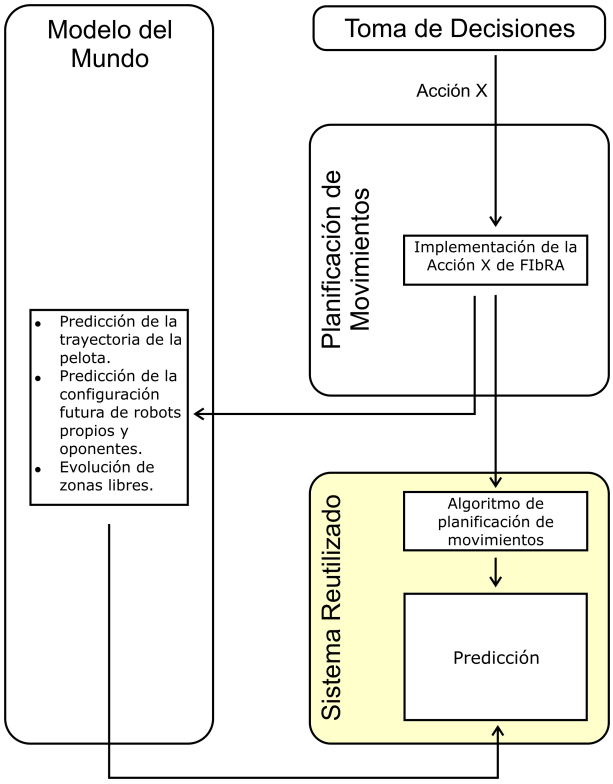


Figura 3.43: Dependencia entre los componentes involucrados en la planificación de movimientos.

puede delegar la totalidad de la resolución al algoritmo reutilizado, valerse del modelo predictivo de la capa *Modelo del Mundo* (definido en la sección 3.5) y resolverlo independientemente, u optar por una estrategia híbrida. Para la solución propuesta, la totalidad de acciones implementadas en la capa *Planificación de Movimientos* dependen del algoritmo de planificación reutilizado, como muestra la figura 3.43.

La solución admite la utilización de acciones compuestas. Éstas se aplican sobre varios robots, posibilitando la coordinación de movimientos.

3.8.3. Evaluación

Fortalezas

- *Arquitectura* : Encapsula la planificación de movimientos dentro de los componentes que implementan a las acciones. Esto permite, por ejemplo, reemplazar el algoritmo reutilizado por otro con un impacto reducido sobre el sistema.
- *Detección de obstáculos* : Para cada acción asignable, el diseñador tiene la opción de habilitar o deshabilitar el mecanismo de detección y evasión de obstáculos, facilitando su ajuste individual. La opción de deshabilitar este mecanismo no debe descartarse *a priori*, debido a que existen equipos referentes que -intencionalmente- no tienen en cuenta esta restricción del entorno.
- *Coordinación* : El diseño permite la implementación de acciones compuestas, que permiten coordinar el movimiento de dos o más robots.
- *Controles de movimiento* :
  - Cada acción se resuelve con la aplicación de uno o varios controles de movimiento independientes entre sí. Esta independencia permite utilizar el modelo subyacente de resolución de velocidades que mejor se adapte a las circunstancias.

- Para acciones que requieren de gran dinamismo se utiliza el método de Lyapunov.

Limitaciones

- *Trayectorias* : Para la resolución de las trayectorias no se tienen en cuenta los límites de la cancha. A raíz de ésto, en ocasiones los robots colisionan contra las paredes e incluso quedan atascados por planificar trayectorias que salen de dichos límites.
- *Predicción* : Tanto la predicción de la trayectoria de la pelota, como la predicción de las configuraciones futuras de los robots son a muy corto plazo (10 iteraciones: aproximadamente 160ms de simulación). Debido a que estas predicciones son utilizadas para calcular las trayectorias cuando el objetivo está en movimiento, el robot debe estar a 10 iteraciones de alcanzarlo para poder anticiparse y lograr una precisión aceptable. Por otro lado, si se aumenta el plazo, los resultados no mejoran debido a que la divergencia entre la predicción y el estado real del entorno es grande.
- *Controles de movimiento* :
  - Los modelos detrás de los controles de movimiento implementados no intentan explícitamente minimizar el tiempo de recorrida de la trayectoria.
  - Las dos alternativas que determinan trayectorias aceptables para trasladar un robot desde un punto de la cancha hacia otro (*Direct LV* y *Direct Ang*) no ofrecen buena estabilidad cerca del objetivo. Para resolver este problema, se implementa un nuevo control denominado *Direct Ang Mejorado* basado en *Direct Ang* que desacelera al robot en las proximidades del objetivo, tendiendo a conformar una maniobra de aparcamiento[33]. En [Ind01] se menciona la necesidad de contar con dichas maniobras para mejorar la estabilidad de los movimientos entorno al objetivo.

3.8.4. Resultados

Simetría de los robots

La figura 3.45 muestra el resultado de un experimento realizado para determinar si el movimiento de los robots es independiente de la dirección en que se aplican las velocidades, dato que no figura en la reglamentación oficial [ABR05c]. En otras palabras, se trata de determinar si es equivalente planificar una trayectoria tomando como frente la cara anterior de un robot o su cara posterior. Si así lo fuera, se podría suponer que los robots son simétricos respecto al plano vertical determinado por el eje de las ruedas (ver figura 3.44).

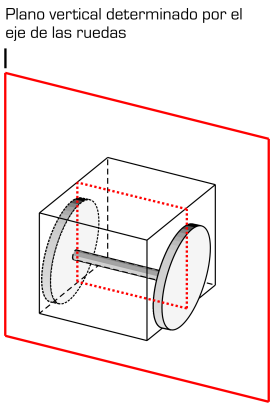


Figura 3.44: Robot simétrico respecto al plano vertical determinado por el eje de las ruedas.

[33] El término maniobra de aparcamiento es una traducción del término *"parking" maneuver* utilizado en la literatura técnica para describir la maniobra de frenado.

Para el experimento se definen dos instancias de planificación independientes. En la primera, se le asigna un determinado par de velocidades a las ruedas de un robot, de forma tal que la trayectoria resultante describa una circunferencia. En la segunda, se le asigna el mismo par de velocidades pero con el signo opuesto al mismo robot, partiendo de una configuración idéntica a la anterior. Si las trayectorias fueran diferentes, entonces se concluiría que el robot es asimétrico respecto al eje.

Las gráficas de la figura 3.45 comparan[34] las proyecciones horizontales (3.45.(a)) y verticales (3.45.(b)) del recorrido del robot para ambas trayectorias. La línea roja representa el recorrido con velocidades izq. = 60 y der. = 30, mientras que la línea negra representa el recorrido con velocidades opuestas.

El resultado del experimento pone en evidencia que la respuesta de un robot a una determinada asignación de velocidades difiere según éstas se orienten hacia su cara posterior o hacia su cara anterior. Esto confirma que el eje de las ruedas se encuentra desplazado horizontalmente respecto al centro del robot.

Precisión

Debido a que los modelos detrás de los controles de movimiento contemplan únicamente objetivos fijos, para enviar un robot hacia la pelota en movimiento es necesario anticipar su posición futura. Las gráficas de las figuras 3.46, 3.47, 3.48 y 3.49 permiten comparar cualitativamente la diferencia entre utilizar los controles *Direct Ang* y *Direct LV* sin predicción y con predicción sobre la trayectoria de la pelota.

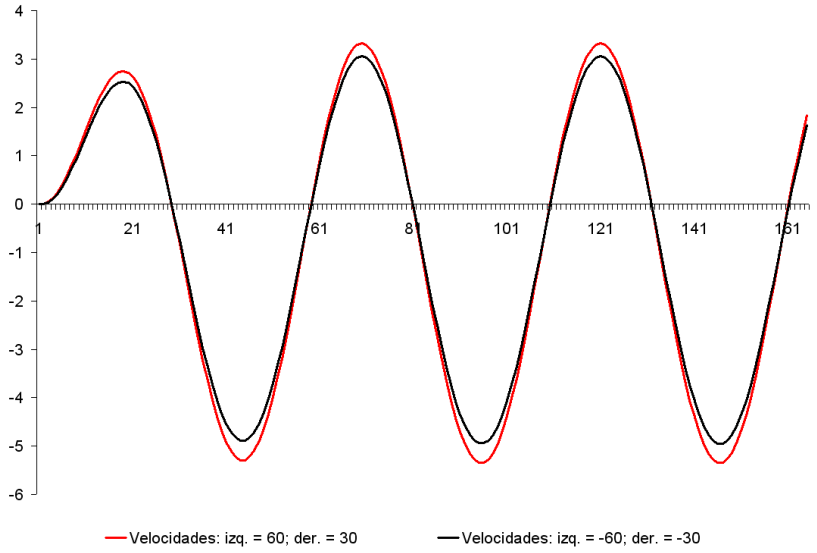
Para poder comparar los resultados, tanto el robot como la pelota parten de la misma configuración inicial. A la pelota, tomada como posición objetivo, se le imprime inicialmente una velocidad paralela al eje de las ordenadas. Como rotación objetivo se toma el ángulo 0 radianes.

La figura 3.46.(a) muestra la trayectoria generada por el control *Direct Ang* y la figura 3.46.(b) por el *Direct LV* sin utilizar predicción para calcular la configuración objetivo. La curva roja representa la trayectoria del robot; la curva negra representa la trayectoria de la pelota. Las circunferencias indican los puntos de mayor acercamiento entre la posición de la pelota (circunferencia negra) y la posición del robot (circunferencia roja).

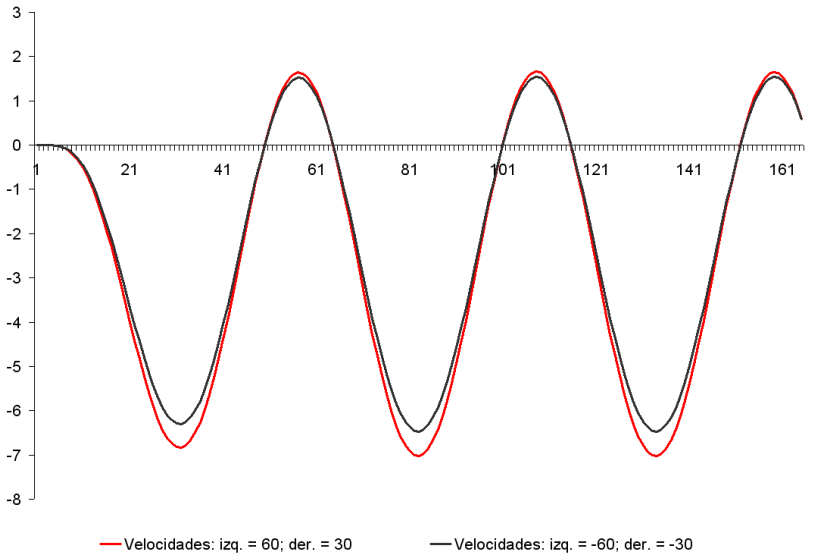
Las figuras 3.47.(a) y 3.47.(c) grafican la distancia entre el robot y la pelota, ilustrando la convergencia hacia la posición objetivo (la línea horizontal negra es la suma entre el radio del robot y el radio de la pelota) para los controles *Direct Ang* y *Direct LV*, respectivamente. Las figuras 3.47.(b) y 3.47.(d) grafican la rotación del robot[35], ilustrando la convergencia hacia la rotación objetivo.

[34] Los valores de las posiciones fueron transformados para que las gráficas sean visualmente comparables.

[35] Para graficar la rotación del robot se ajusta la fase de los ángulos a los efectos de que no haya saltos al superar los 2 π radianes.



(a) Proyección horizontal del movimiento.



(b) Proyección vertical del movimiento.

Figura 3.45: Asimetrías en el movimiento de los robots.

La descripción de las gráficas de las figuras 3.48 y 3.49 es análoga, salvo que para estos casos se utiliza predicción para calcular la configuración objetivo.

Los resultados se interpretan de la siguiente manera:

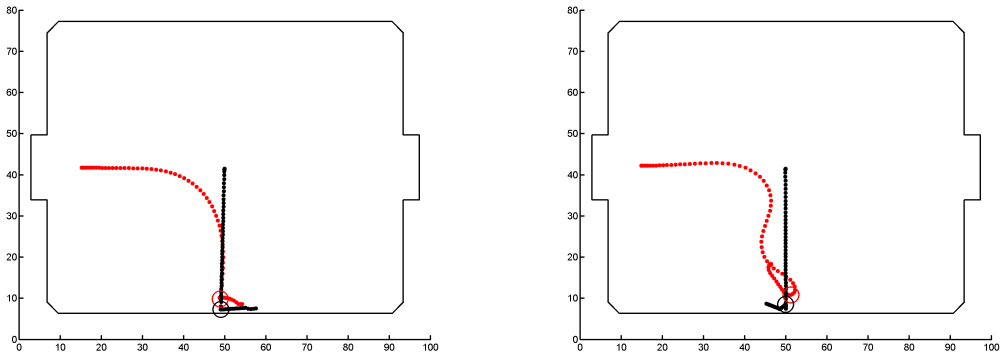
Control "Direct Ang" sin utilizar predicción Hacia el final del tramo, el robot alcanza la pelota siguiendo tangencialmente su trayectoria, debido a que es incapaz de anticipar su posición futura. Dado que la pelota se detiene en el lateral derecho, el control puede corregir la rotación (figura 3.46.(a)). Sin embargo, en el momento de mayor acercamiento entre el robot y la pelota, la rotación dista de alcanzar los 0 radianes. Por lo tanto, se concluye que el movimiento no converge hacia la configuración objetivo.

Control "Direct LV" sin utilizar predicción El robot alcanza la pelota casi al mismo tiempo

que en el caso de la figura 3.46.(a). Sin embargo, colisiona con la pelota con un ángulo cercano a los π radianes, con lo cual no converge hacia la configuración objetivo.

Control "Direct Ang" utilizando predicción A pesar de utilizarse la predicción de la trayectoria de la pelota, el movimiento del robot no anticipa apropiadamente su posición futura. Como en el caso de la figura 3.46.(a) en el momento de mayor acercamiento con la pelota, la rotación del robot también dista de los 0 radianes tomados como objetivo. El resultado confirma la observación realizada durante los partidos jugados: *Direct Ang* no converge adecuadamente para objetivos móviles.

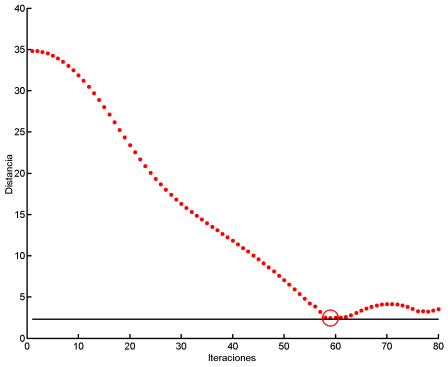
Control "Direct LV" utilizando predicción A diferencia del caso de la figura 3.46.(b), la utilización de la predicción para resolver la trayectoria con *Direct LV* permite una mejor reacción por parte del robot: cuando la pelota golpea el lateral derecho, el robot inmediatamente gira a 0 radianes (ver figuras 3.48.(b) y 3.49.(d)). Sin embargo, la trayectoria termina desviándose rápidamente, lográndose una convergencia pobre.



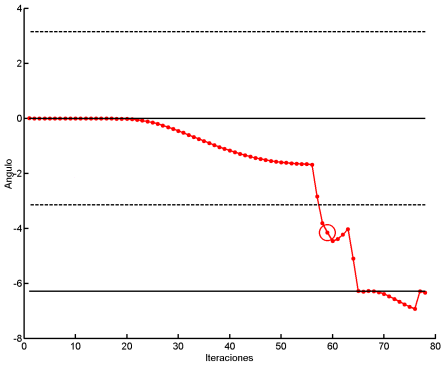
(a) *Direct Ang.* (b) *Direct LV.*

Figura 3.46: Trayectorias generadas tomando como configuración objetivo la posición de la pelota y una rotación de 0 radianes, sin utilizar predicción.

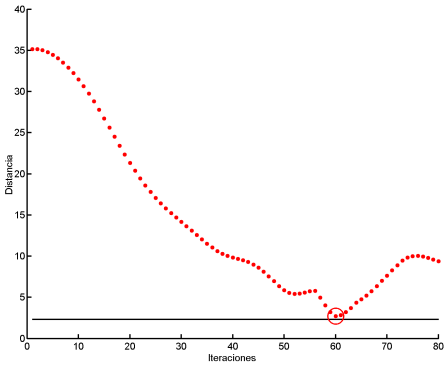
A pesar de que el comportamiento de estos controles es variable a lo largo de un partido, obteniéndose incluso rendimientos razonables, el conjunto de pruebas elegido plantea escenarios que pueden ocurrir durante el transcurso del juego y apuntan a mostrar que el algoritmo de planificación de movimientos admite mejoras. En un contexto competitivo, es vital atender a todas las situaciones que puedan ocurrir, debido a que las oportunidades desperdiciadas pueden convertirse en oportunidades aprovechables por el oponente.



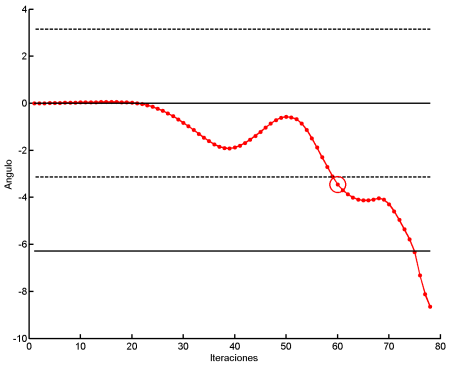
(a) Distancia entre la posición del robot y la pelota con el control *Direct Ang*.



(b) Rotación del robot con el control *Direct Ang*.

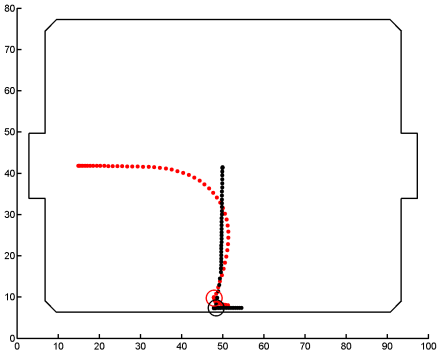


(c) Distancia entre la posición del robot y la pelota con el control *Direct LV*.

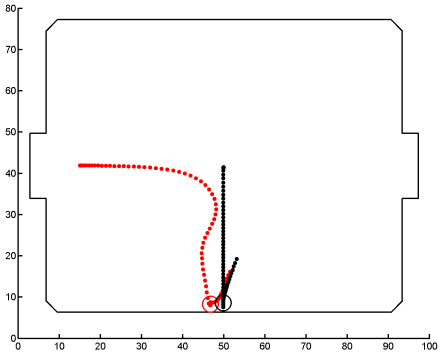


(d) Rotación del robot con el control *Direct LV*.

Figura 3.47: Convergencia entre la configuración del robot y la configuración objetivo para las trayectorias de la figura 3.46.

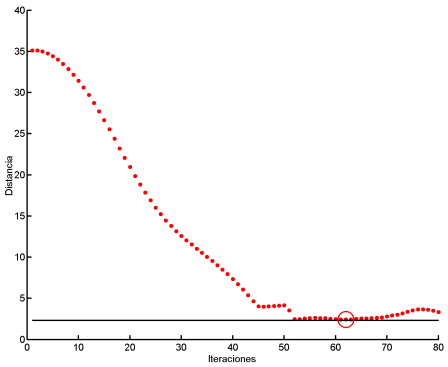


(a) *Direct Ang*.

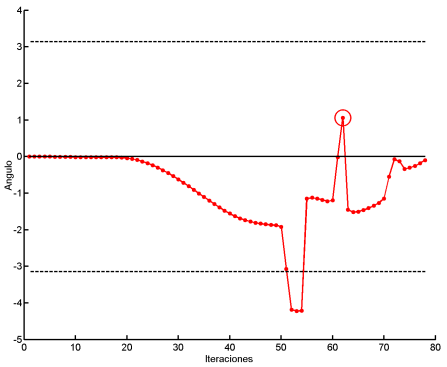


(b) *Direct LV*.

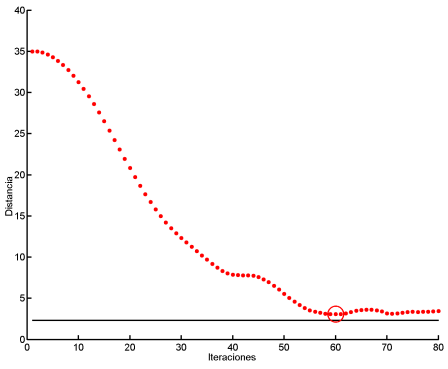
Figura 3.48: Trayectorias generadas tomando como configuración objetivo la posición de la pelota y una rotación de 0 radianes, utilizando predicción.



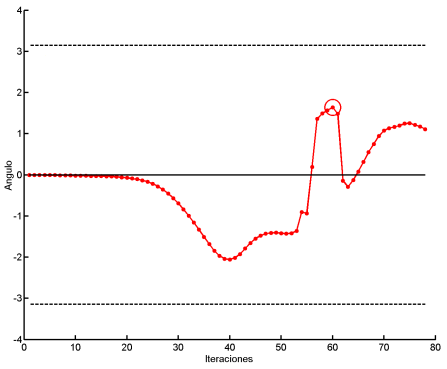
(a) Distancia entre la posición del robot y la pelota con el control *Direct Ang*.



(b) Rotación del robot con el control *Direct Ang*.



(c) Distancia entre la posición del robot y la pelota con el control Direct LV.



(d) Rotación del robot con el control *Direct LV*.

Figura 3.49: Convergencia entre la configuración del robot y la configuración objetivo para las trayectorias de la figura 3.48.

Capítulo 4

Evaluación global

4.1. Introducción

Este capítulo apunta a poner en evidencia el funcionamiento integrado de las distintas partes que componen el sistema (desde un punto de vista cualitativo) comportándose como un equipo de fútbol de robots para la categoría SimuroSot. No contempla la verificación desde un punto de vista de la Ingeniería del Software en busca de errores de programación o diseño. En ese sentido se analizan diferentes aspectos globales de su comportamiento que permiten conocer sus fortalezas y debilidades, y enfocar los futuros pasos a seguir en la mejora del equipo o las ideas propuestas. Es por esto que, más allá de las pruebas y resultados particulares presentados para los distintos componentes del sistema, se realiza una evaluación general que comprende a todos ellos y su ejecución en el entorno donde tiene lugar la competencia (simulador oficial).

En la sección 4.2 se plantean las principales características a evaluar y sus correspondientes experimentos. Luego, en las secciones 4.3 y 4.4, se presentan los resultados obtenidos en dos ambientes distintos:

- *Ambiente de prueba* : Utilizando equipos de los años 2003 y 2004, donde se privilegia la fluidez del juego por sobre la aplicación de la reglas.
- *Ambiente de competición* : Corresponde al CAFR2006.

La sección 4.5 presenta experimentos realizados para determinar los tiempos de respuesta de la estrategia y sus componentes. Por último, en la sección 4.6 se evalúan los resultados obtenidos.

4.2. Experimentos

4.2.1. Limitaciones tecnológicas

La cantidad y la calidad de los experimentos está condicionada por las herramientas que posibiliten y faciliten la grabación, reproducción y observación de los partidos, así como de la posibilidad de extraer información sintetizada a partir de los mismos.

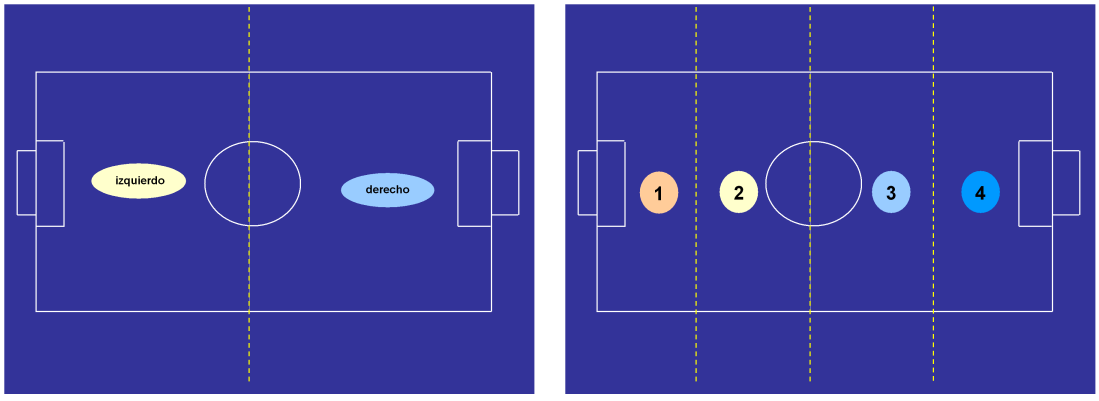
Para la construcción de equipos de la categoría SimuroSot, existen limitaciones tecnológicas que dificultan en gran medida la realización de estos experimentos. En particular, el simulador oficial requiere de la atención permanente de un operador humano (por ejemplo, para intervenir frente a la ocurrencia de goles o jugadas especiales que requieran la participación del árbitro, o para resolver manualmente un atascamiento), por lo que no es posible la reproducción automática de un juego (y por consiguiente tampoco series de partidos). Asimismo, carece de un mecanismo que permita grabar (o reproducir) partidos, por lo que se requiere de alguna herramienta alternativa que resuelva esta funcionalidad. Finalmente, no ofrece indicadores sobre el estado del juego, como el estado del tanteador, ni permite extraer estadísticos que proporcionen información resumida sobre el comportamiento de los equipos.

Con la herramienta *LogViewer*[1] es posible paliar al menos uno de estos problemas: la reproducción de partidos (grabados previamente por el sistema durante la ejecución del juego). Sin embargo esta herramienta no posee otras capacidades que permitan resolver el resto de los problemas mencionados. Como forma de avanzar en la resolución de esta problemática, se desarrolló un prototipo de motor de simulación con el objetivo de ser incorporado en la construcción de un simulador alternativo (los detalles de este desarrollo pueden encontrarse en [ABR06d]).

4.2.2. Criterios de evaluación

Los experimentos se basan en varios criterios que permiten evaluar el comportamiento global del equipo desde diferentes perspectivas. Algunos se extraen del documento [FTIAM00] (aplicados para medir similares características en equipos de la liga RoboCup), mientras que otros se proponen para evaluar aspectos concretos del equipo FIbRA. Estos criterios de evaluación se aplican durante el primer tiempo de juego (5 minutos de partido aproximadamente) y se describen a continuación:

- *Porcentaje de tiempo que la pelota se juega en campo propio u oponente* : Se toma la división de la cancha en dos campos, izquierdo y derecho (ver figura 4.1.(a)), indicando, en cada iteración, en cuál de ellos se encuentra la pelota ($*$). A partir de esta prueba se puede identificar el grado de presencia de la pelota en un lado u otro de la cancha. Una mayor presencia en el lado oponente sugiere una mayor oportunidad de gol para el equipo FIbRA, y viceversa.
  Los resultados son analizados en base a dos criterios:
  - considerando todo el partido;
  - considerando la disponibilidad de la predicción del comportamiento oponente.
- *Distribución de jugadores en la cancha*: Se divide la cancha en cuatro zonas, como muestra la figura 4.1.(b), indicando en cada iteración la cantidad de robots que se encuentran en cada una ($*$). A partir de esta prueba se puede indentificar el grado de presencia de ambos equipos en cada zona de la cancha. Una mayor presencia de los robots propios en las zonas 1 y 2 sugiere una mayor presión sobre el oponente, aumentando la oportunidad de gol para el equipo FIbRA, y viceversa para el oponente en las zonas 3 y 4.
  Los resultados son analizados en base a dos criterios:
  - considerando todo el partido;
  - considerando la disponibilidad de la predicción del comportamiento oponente.



(a) División en dos partes: campo izquierdo y campo derecho.

(b) División en cuatro partes: zona 1, zona 2, zona 3 y zona 4.

Figura 4.1: División de la cancha para evaluar el movimiento de la pelota y los robots.

[1] *LogViewer* es un visor de *logs* desarrollado por el proyecto FRUTo y reutilizado para el equipo FIbRA

- *Cantidad de goles* : Se cuenta la cantidad de goles convertidos en el partido (†) y se discrimina según los siguientes criterios:
  - total de goles convertidos por el oponente en el arco propio;
  - total de goles convertidos por el oponente en el arco oponente (goles en contra);
  - total de goles convertidos por el equipo FIbRA en el arco propio (goles en contra);
  - total de goles convertidos por el equipo FIbRA en el arco oponente.
- *Efectividad del ataque oponente* : Se cuenta la cantidad de llegadas con posibilidad de gol[2] de robots oponentes al arco propio (‡). Estos datos son analizados en base a los siguientes criterios:
  - considerando la disponibilidad de la predicción del comportamiento oponente;
  - relacionado con la cantidad de goles convertidos en cada arco.
- *Efectividad de la defensa propia* : Se cuenta la cantidad de pelotas que son atajadas por el golero en el arco propio ante una situación ofensiva del equipo oponente (‡). Estos datos son analizados en base a los siguientes criterios:
  - considerando la disponibilidad de la predicción del comportamiento oponente;
  - relacionado con la cantidad de llegadas al arco oponente y con la cantidad de goles convertidos.

Notas:

(∗) Los datos se obtienen exclusivamente de la información del entorno proporcionada por el simulador oficial. Se generan conjuntos de prueba que ofrecen datos concretos y precisos.

(†) Los datos son obtenidos a partir de la observación humana de los partidos y del detector de goles descrito en la sección 3.6.2.2. Los resultados quedan sujetos a la subjetividad en la observación de cada partido.

(‡) Los datos son obtenidos exclusivamente a partir de la observación humana de los partidos, sin apoyarse en herramientas o pruebas automáticas. Los resultados quedan sujetos a la subjetividad en la observación de cada partido.

4.3. Ambiente de prueba

Los resultados presentados en esta sección han sido obtenidos en el ambiente de desarrollo, ejecutando el primer tiempo de varios[3] partidos entre el equipo FIbRA y distintos equipos oponentes. Debido a que no se cuenta con un árbitro oficial para los partidos de prueba y se desconoce la forma en que los oponentes forman para cada una de las jugadas especiales de pelota quieta, no se cobran faltas ni se reposicionan robots, salvo atascamientos (10 segundos con la pelota detenida) para garantizar un mínimo de fluidez en el trascurso del juego. En caso de atascamiento, el juego es detenido y la pelota es colocada en el punto de pique más cercano a su posición (ver [ABR05c]) sin mover a los jugadores de lugar. El partido es reiniciado sin marcar ninguna jugada especial en el simulador. Para el caso de los goles no se contemplan situaciones particulares en las que el gol podría ser anulado, generando un saque de arco o un penal. Se considera que estas situaciones no se presentan con mayor frecuencia, por lo que no contemplarlas no afecta la evaluación que se realiza sobre los resultados obtenidos.

[2] Como criterio para considerar que una jugada es una llegada al arco con posibilidad de gol, se considera aquellas situaciones en que un robot entra en el área grande del arco dominando la pelota.

[3] En total se ejecutaron doce partidos con el cometido de formalizar los resultados obtenidos previamente en otras instancias de evaluación que permitieron la puesta a punto del equipo. Como consecuencia de las limitaciones mencionadas en la sección 4.2.1 y a los efectos de obtener los indicadores y estadísticos descritos en la sección 4.2.2, fue necesario reproducir cada partido varias veces debido a que en una ejecución no era posible centrar la atención en todos los aspectos a analizar, limitando la productividad del tiempo destinado a estos experimentos.

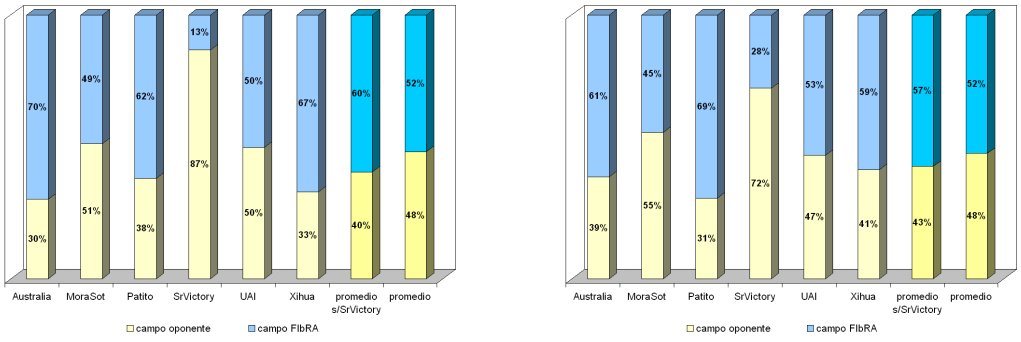
En todos los casos el equipo FIbRA juega con color azul sobre el campo derecho. Los equipos utilizados para este análisis son: Australia, MoraSot, Patito, SrVictory, UAI y Xihua.

Cabe aclarar que el equipo utilizado para estas pruebas coincide con el equipo presentado en el CAFR2006, de forma de poder comparar los resultados obtenidos en el ambiente de prueba con equipos del 2003 y 2004, con los resultados competitivos obtenidos con equipos del año 2006[4].

4.3.1. Porcentaje de tiempo que la pelota se juega en campo propio u oponente

A partir de la información generada para cada partido se calcula el porcentaje de tiempo que estuvo la pelota en cada lado de la cancha, distinguiéndose el caso en que se cuenta con la predicción del comportamiento oponente y el caso en que no.

La figura 4.2 muestra las gráficas obtenidas en estos partidos, pudiendo observarse que, en general, el porcentaje de juego en cancha propia supera al juego en cancha oponente.

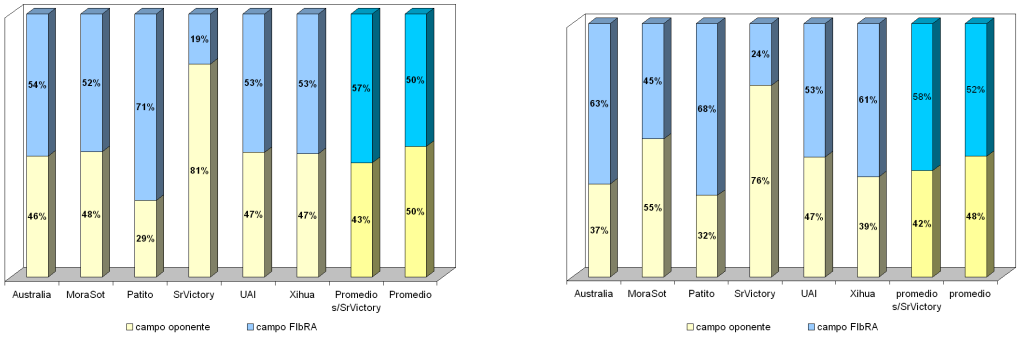


(a) Antes de la predicción.

(b) Después de la predicción.

Figura 4.2: Porcentaje de iteraciones que la pelota se encuentra en campo propio u oponente.

La leve diferencia en el porcentaje de permanencia de la pelota en cada lado de la cancha, antes y después de generado el resultado de la predicción, no es concluyente. Para tener un indicio de si la predicción verdaderamente influye se realiza una prueba similar sin generar predicción. La figura 4.3.(a) muestra el resultado de dicha prueba.



(a) Sin predicción.

(b) Con predicción.

Figura 4.3: Porcentaje de iteraciones que la pelota se encuentra en campo propio u oponente.

[4] El equipo presentado en el CAFR2006 no incluye las dos estrategias extremas presentadas en la sección 3.7 -*Ofensa Agresiva y Defensa Agresiva*- ya que el proceso de ajuste de éstas ha sido menos exhaustivo en relación al de las demás.

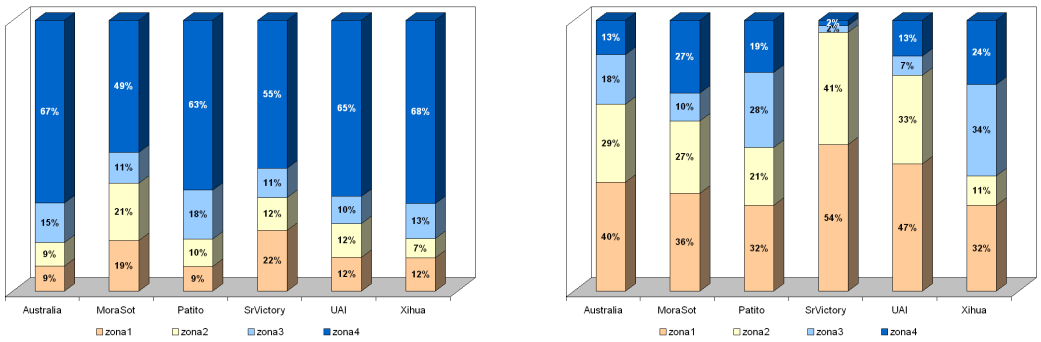
En términos generales, los resultados de ambas pruebas son similares, mostrando que el uso de la predicción no afecta sensiblemente el porcentaje de permanencia.

4.3.2. Distribución de jugadores en la cancha

En las figuras 4.4 y 4.5 se muestra la distribución de jugadores en la cancha (dividida en 4 zonas) a lo largo del tiempo de juego, antes y después de contar con la información de la predicción.

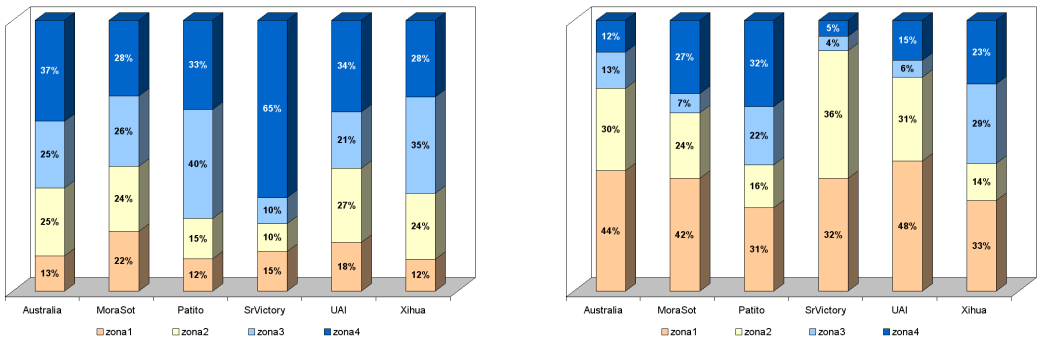
A partir de estos resultados, no se observa una variación importante en la cantidad promedio de robots oponentes en cada zona, los cuales mantienen un comportamiento muy similar tanto antes como después de la predicción. Sin embargo, se nota un cambio en el comportamiento de los robots propios, principalmente en la zona 4 (zona del arco propio), detectándose un desplazamiento del juego hacia el medio campo.

Para comprobar si realmente la predicción influye en el movimiento de los robots propios, se juega un tiempo de partido completo con los mismos equipos, pero sin generar predicción durante todo el período. La figura 4.6 muestra las gráficas obtenidas para los robots propios y oponentes.



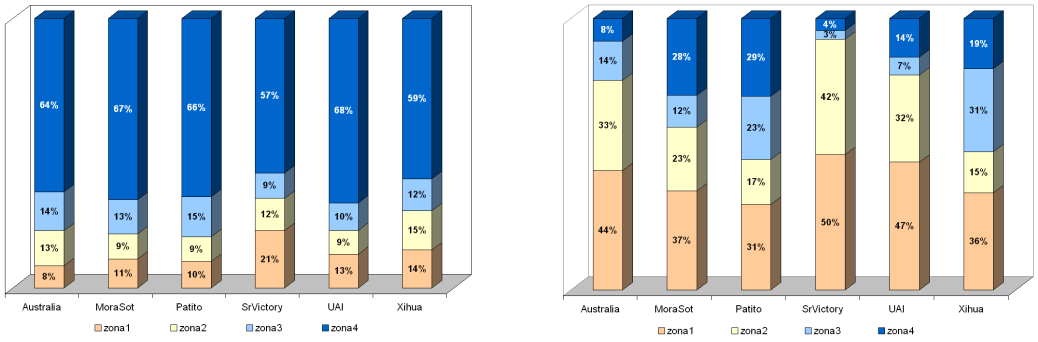
(a) Robots propios. (b) Robots oponentes.

Figura 4.4: Distribución de los robots en la cancha antes de generada la predicción.



(a) Robots propios. (b) Robots oponentes.

Figura 4.5: Distribución de los robots en la cancha después de generada la predicción.



(a) Robots propios.

(b) Robots oponentes.

Figura 4.6: Distribución de los robots en la cancha sin predicción.

Nuevamente, los resultados para los robots oponentes muestran valores similares a los anteriores. Los resultados para los robots propios confirman la influencia de la predicción en su comportamiento, pues los resultados obtenidos con predicción muestran un avance de los robots en la cancha respecto a los dos obtenidos sin predicción.

4.3.3. Cantidad de goles

Para cada partido se lleva cuenta de los goles realizados, discriminando entre los convertidos por cada equipo en el arco opuesto y los goles en contra.

El cuadro 4.1 muestra los resultados generales de los partidos y la figura 4.7 presenta las gráficas resultantes del análisis realizado, para el total general y también discriminando entre goles opuestos y goles en contra para cada equipo.

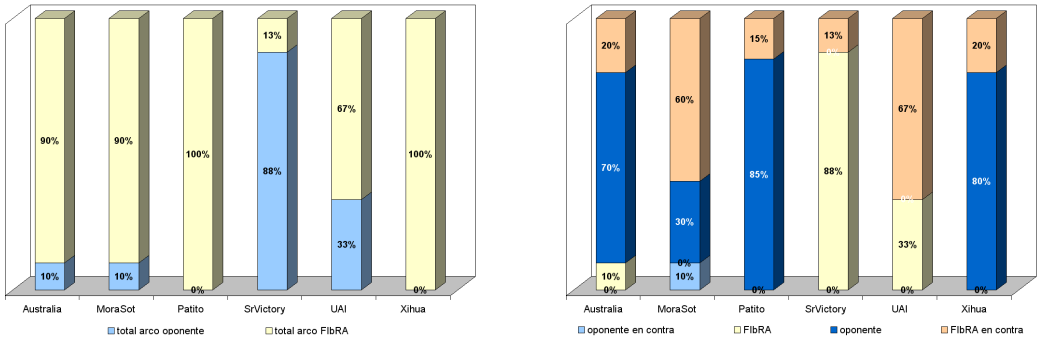
	Equipo FIbRA	**Equipo oponente**
Australia	1	9
MoraSot	1	9
Patito	0	20
SrVictory	7	1
UAI	2	4
Xihua	0	15

Cuadro 4.1: Resultado de los partidos de prueba.

Se observa una alta tasa de goles en contra para el caso del equipo FIbRA, perjudicando ampliamente los resultados globales de los partidos. En general los equipos oponentes presentan una tasa muy baja de conversión de goles en contra, además de la cantidad de goles que logran convertir en el arco opuesto. Por otro lado, son varios los goles en contra que pueden ser evitados mejorando la precisión en el movimiento de los robots.

4.3.4. Efectividad del ataque oponente

En lo referente a las llegadas al arco se obtienen los resultados globales detallados en el cuadro 4.2. Éstos son graficados y comparados con la cantidad de goles que se realiza en cada arco. En la figura 4.8.(a) se muestra la gráfica con porcentajes de llegadas de cada equipo al arco opuesto y en 4.8.(b) la gráfica que indica qué porcentaje de esas llegadas alcanzan el objetivo (convertir un gol), dando un indicador de la efectividad del equipo frente al arco opuesto.



(a) Total general de goles convertidos en cada arco.

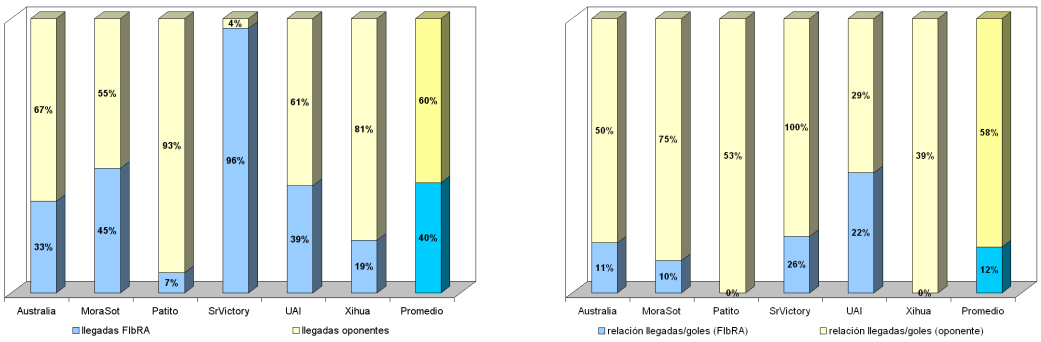
(b) Cantidad de goles convertidos en cada arco, discriminando goles en contra.

Figura 4.7: Goles convertidos en cada arco para cada equipo.

	Llegadas del equipo FIbRA al arco oponente	**Llegadas del equipo oponente al arco propio**
Australia	9	18
MoraSot	10	12
Patito	3	38
SrVictory	27	1
UAI	9	14
Xihua	9	38

Cuadro 4.2: Cantidad de llegadas de cada equipo al arco opuesto.

Como se puede apreciar en la figura 4.8, el porcentaje de efectividad es ampliamente mayor para todos los equipos oponentes. Esto, sumado a la falta de precisión de los robots atacantes del equipo FIbRA cuando logran tirar al arco, dificulta la obtención de buenos resultados en el tanteador.

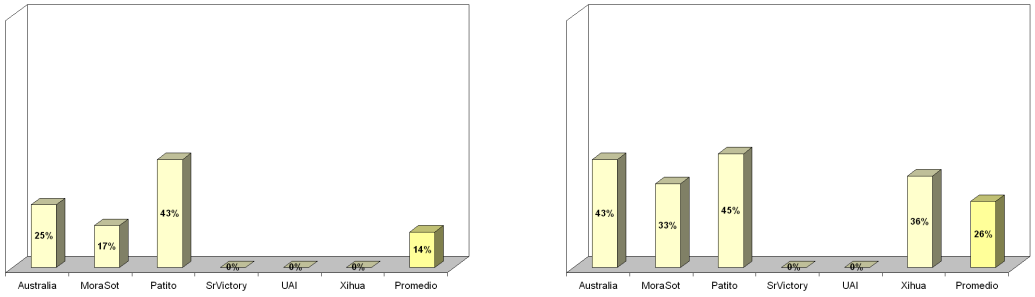


(a) Total general de goles convertidos en cada arco.

(b) Porcentaje de llegadas al arco opuesto.

Figura 4.8: Porcentaje y efectividad de las llegadas al arco opuesto.

Para comprobar si la predicción afecta la cantidad de llegadas de los robots al arco opuesto o la efectividad en cada llegada, se analizan estos resultados en la etapa previa a la generación de la información de predicción y posterior a la misma. Los resultados obtenidos en cada caso se muestran en la figura 4.9.



(a) Antes de generada la predicción. (b) Después de generada la predicción.

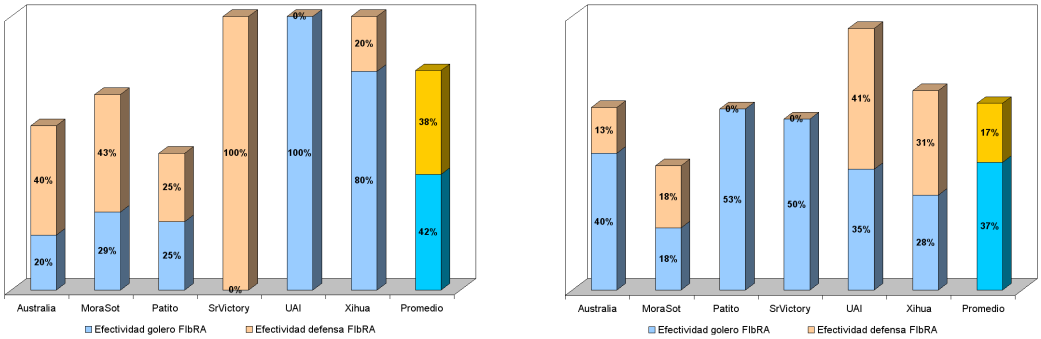
Figura 4.9: Efectividad del ataque oponente.

Las gráficas muestran que la efectividad de las llegadas oponentes al arco propio se incrementa considerablemente una vez generada la predicción.

4.3.5. Efectividad de la defensa propia

Resulta interesante, luego de conocida la cantidad de llegadas del oponente y los goles que convierte, determinar cuántas de esas llegadas son despejadas por la defensa, evitando el gol. Esto permite conocer la efectividad de la defensa del equipo FIbRA frente a las jugadas de ataque del oponente.

Las gráficas presentadas en la figura 4.10 muestran esta efectividad antes y después de contar con el resultado de la predicción.



(a) Antes de la predicción. (b) Después de la predicción.

Figura 4.10: Efectividad de la defensa del equipo FIbRA.

Estos resultados se relacionan con los obtenidos para la efectividad del ataque oponente, observándose cómo ésta aumenta cuando disminuye la efectividad del golero o la defensa propia. Esta disminución puede deberse a que los equipos oponentes muy ofensivos provocan la detección de muchos *clusters* cercanos al área donde se mueve el golero, entorpeciendo su accionar. Sin embargo,

en general, puede observarse una leve mejoría en el desempeño promedio del golero luego de la predicción.

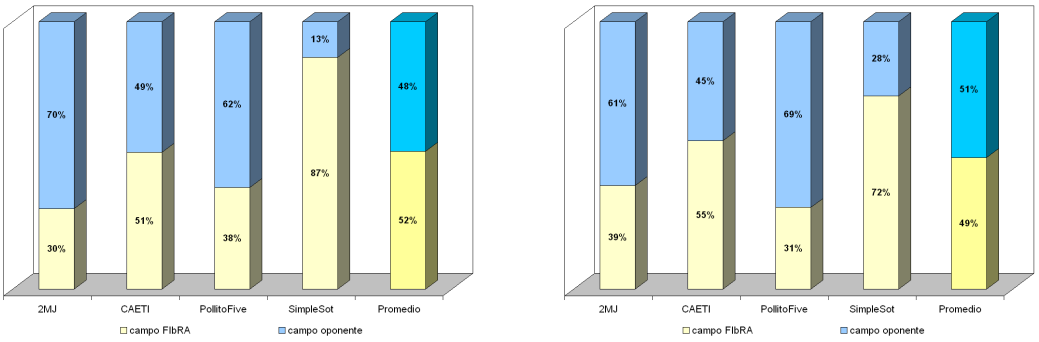
4.4. Ambiente de competición

Los resultados presentados en esta sección han sido obtenidos luego de reproducirse el primer tiempo de los partidos de la serie[5] que jugó el equipo FIbRA en el CAFR2006, enfrentándose a los siguientes equipos: 2MJ, CAETI, Pollito Five II y SimpleSot.

Estos partidos contaron con arbitraje oficial y, por tanto, todas las faltas fueron cobradas, ejecutando las jugadas especiales correspondientes.

4.4.1. Porcentaje de tiempo que la pelota se juega en campo propio u oponente

A partir de los partidos jugados se obtienen las estadísticas correspondientes al porcentaje de tiempo que la pelota se encuentra en campo propio u oponente. En la figura 4.11 se muestran las gráficas generadas a partir de estas estadísticas. Se puede observar la similitud de estos resultados con los presentados anteriormente para el ambiente de prueba. En general, la pelota se mantiene aproximadamente el mismo tiempo en campo propio u oponente, antes y después de la predicción.



(a) Antes de la predicción.

(b) Después de la predicción.

Figura 4.11: Porcentaje de iteraciones que la pelota se encuentra en campo propio u oponente.

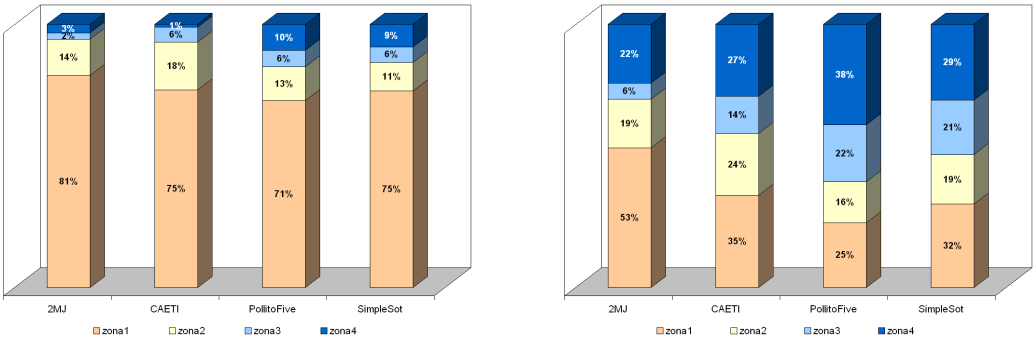
4.4.2. Distribución de jugadores en la cancha

Con los datos de los cuatro partidos jugados, se gráfica el porcentaje de utilización de cada zona de la cancha tanto para los robots propios como para los oponentes.

El análisis se realiza discriminando los resultados previos y posteriores a la generación de la predicción del comportamiento oponente, mostrados en las figuras 4.12 y 4.13 respectivamente.

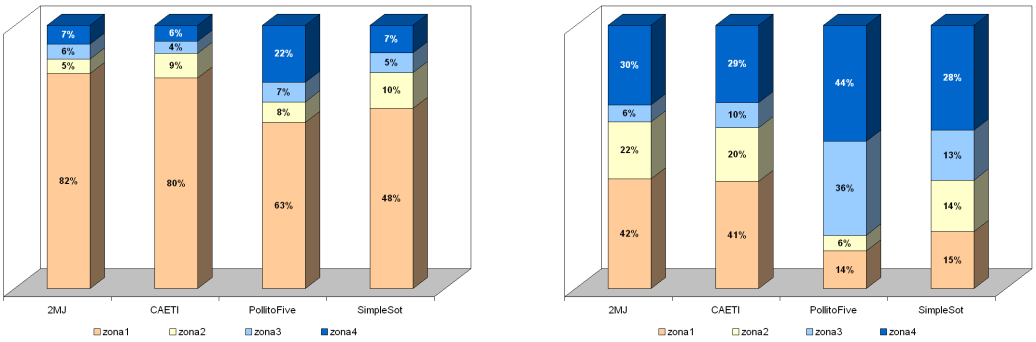
Estos resultados son contundentes en cuanto a la actitud defensiva y retrasada sobre el campo de juego del equipo FIbRA. La generación de la predicción parece contribuir a un leve desplazamiento del equipo hacia las zonas centrales de la cancha, buscando alejar las situaciones de riesgo del arco. Sin embargo, continúa siendo muy elevado el porcentaje de robots que se mueven cerca del arco propio. La figura 4.14 muestra el resultado general para cada partido.

[5] Todos los equipos de la serie accedieron a que se grabara información de los partidos, permitiendo su reproducción *a posteriori* para analizar los resultados con más detalle.



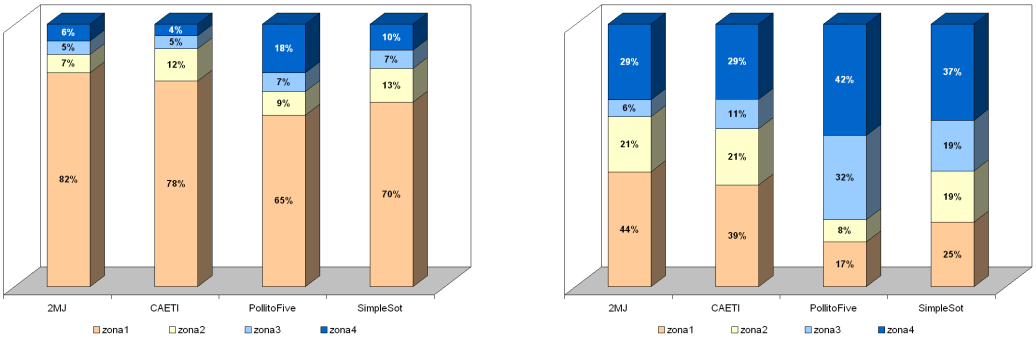
(a) Robots propios. (b) Robots oponentes.

Figura 4.12: Distribución de los robots en la cancha antes de generada la predicción.



(a) Robots propios. (b) Robots oponentes.

Figura 4.13: Distribución de los robots en la cancha después de generada la predicción.



(a) Robots propios. (b) Robots oponentes.

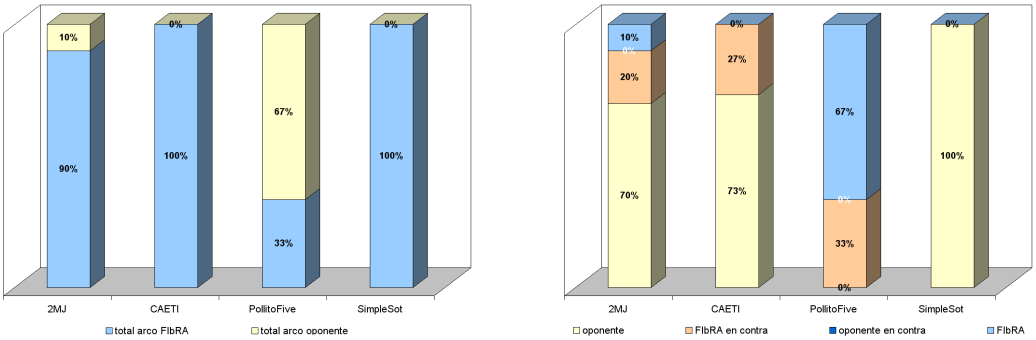
Figura 4.14: Distribución de los robots en la cancha durante todo el primer tiempo.

4.4.3. Cantidad de goles

El cuadro 4.3 presenta los tanteadores parciales (primer tiempo) de los partidos realizados con cada equipo y la figura 4.15 muestra las gráficas resultantes del análisis realizado, discriminando los goles en contra.

	Equipo FIbRA	Equipo oponente
2MJ	1	9
CAETI	0	15
Pollito Five II	2	1
SimpleSot	0	6

Cuadro 4.3: Resultado parcial de los partidos jugados con los distintos equipos en el CAFR2006.



(a) Sin discriminar goles en contra.

(b) Discriminando goles en contra.

Figura 4.15: Goles convertidos en cada arco para cada equipo.

Es importante resaltar el hecho de que ninguno de los equipos oponentes convirtió goles en contra. Por el contrario, el equipo FIbRA muestra una alta tasa de goles en contra, contribuyendo al incremento del tanteador en favor del oponente. En casos en los que el rival es netamente superior puede no notarse la diferencia en el resultado final, tal es el caso del equipo CAETI. Sin embargo, en partidos muy disputados, como el caso de PollitoFive, en que la diferencia en el tanteador es mínima, este tipo de errores puede inclinar la balanza en favor del adversario.

El cuadro 4.4 presenta los resultados finales de cada partido completo (2 tiempos de 5 minutos) de la serie disputada en el campeonato.

	Equipo FIbRA	Equipo oponente
2MJ	1	13
CAETI	0	20
Pollito Five II	3	4
SimpleSot	3	7

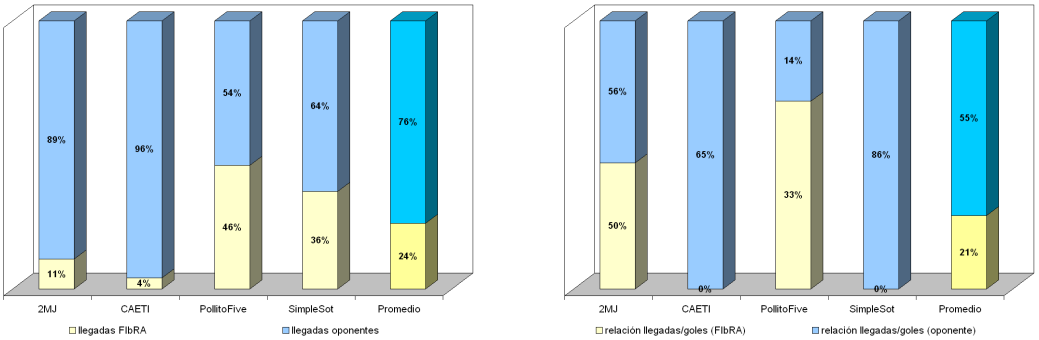
Cuadro 4.4: Resultado final de los partidos jugados en el CAFR2006.

4.4.4. Efectividad del ataque oponente

El cuadro 4.5 muestra la cantidad de llegadas de cada equipo a su arco opuesto. La cantidad de éstas que realmente culminan en gol es graficada en la figura 4.16.

	Llegadas del equipo FIbRA al arco oponente	Llegadas del equipo oponente al arco propio
2MJ	2	16
CAETI	1	23
Pollito Five II	6	7
SimpleSot	4	7

Cuadro 4.5: Cantidad de llegadas de cada equipo al arco opuesto.



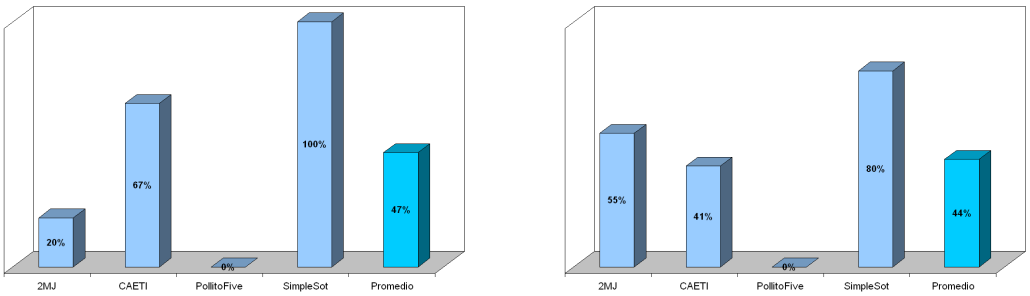
(a) Llegadas al arco opuesto.

(b) Porcentaje de llegadas que terminan en gol.

Figura 4.16: Porcentaje y efectividad de las llegadas al arco opuesto.

Los resultados para los dos primeros partidos muestran la falta de ataque del equipo FIbRA, reafirmando las conclusiones derivadas del análisis del movimiento de los robots en las distintas zonas de la cancha. La formación muy retraída es ampliamente notoria contra equipos fuertes y ofensivos, lo cual obliga a ser muy precisos al momento de ejecutar los tiros al arco, pues son mínimas las llegadas con posibilidad de conversión.

En la figura 4.17 se puede apreciar la efectividad del ataque oponente, presentando el resultado antes de la predicción y luego de contar con ella. Se puede visualizar, exceptuando el equipo 2MJ, una leve disminución en la efectividad del ataque oponente, lo cual es un buen indicio del uso de la predicción para frenar el ataque rival.



(a) Antes de la predicción.

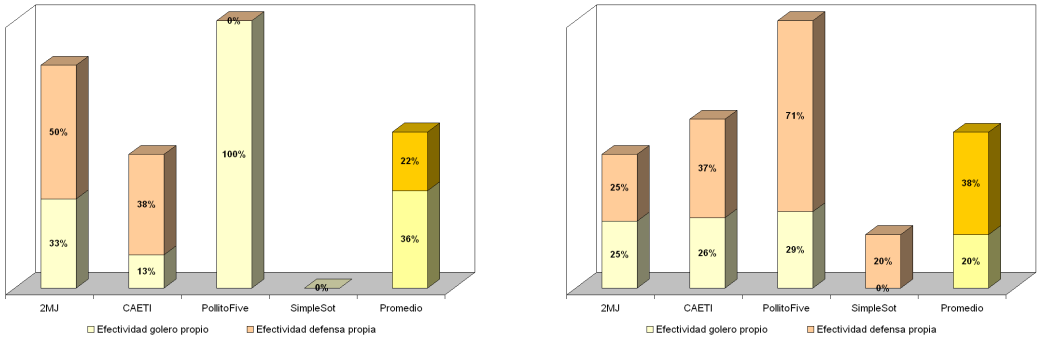
(b) Después de la predicción.

Figura 4.17: Efectividad del ataque oponente.

4.4.5. Efectividad de la defensa propia

A continuación se muestran los resultados obtenidos en relación a la efectividad de la defensa, llevando cuenta de cuántas llegadas oponentes son despejadas por el golero o los defensas.

Tomando los datos obtenidos para la cantidad de llegadas del oponente, la cantidad de goles convertidos por cada equipo y la cantidad de pelotas despejadas por la defensa del equipo FIbRA, se generan las gráficas de desempeño de la defensa, antes y después de la predicción. Los resultados son presentados en la figura 4.18.



(a) Antes de generada la predicción.

(b) Después de generada la predicción.

Figura 4.18: Efectividad de la defensa del equipo FIbRA.

Estos resultados muestran cómo cambia el porcentaje de partipación de los defensas y el golero en las jugadas de ataque del oponente, luego de generada la información de la predicción. Asimismo, contra algunos equipos se logra mantener o aumentar el porcentaje de efectividad de la defensa, a la vez que aumenta el protagonismo de los defensas, lo cual se interpreta como algo positivo pues estaría indicando que las jugadas de ataque del oponente son interceptadas más lejos del arco propio.

4.5. Otros resultados

Independientemente de los resultados respecto al comportamiento de los robots, es importante determinar el desempeño del sistema en lo que refiere a tiempos de respuesta. Para ello se analiza el tiempo que tarda la estrategia en resolver la asignación de acciones por iteración.

Debido a que la estrategia se compone de la toma de decisiones y de la planificación de movimientos, el análisis de tiempos de respuesta ofrece los datos discriminados para cada uno de estos componentes. Cabe aclarar que la predicción no ha sido tomada en cuenta en este análisis, pues la misma se ejecuta en paralelo, y por tanto, no afecta directamente y en forma sostenida los tiempos de respuesta de la estrategia.

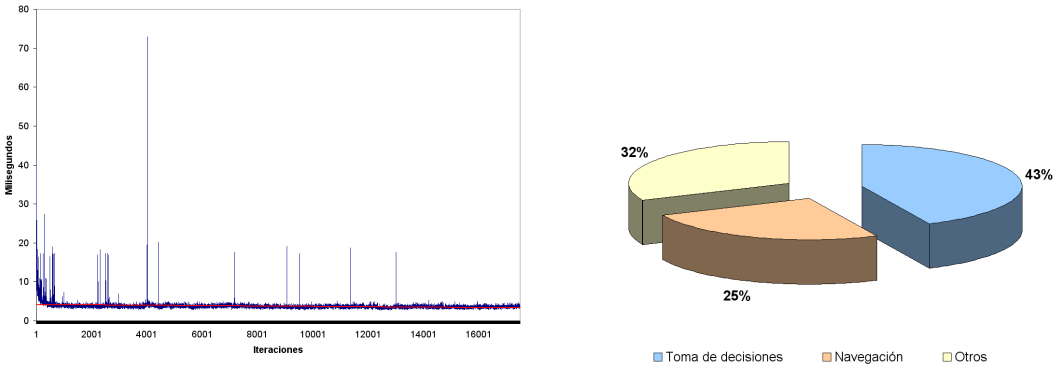
El cuadro 4.6 muestra los valores promedio (considerando las ejecución de los partidos utilizados en la sección 4.3) de tiempo de respuesta de la estrategia en su conjunto y de los componentes de navegación y toma de decisiones. El tiempo asignado a *Otros* corresponde al utilizado por la estrategia para combinar acciones, entre otros cálculos..

La figura 4.19.(a) muestra el promedio del tiempo de respuesta de la estrategia por iteración, calculado a partir de la ejecución de los partidos jugados en el ambiente de prueba. La figura 4.19.(b) muestra cómo incide cada componente en el tiempo total de procesamiento de la estrategia.

Los resultados obtenidos permiten extraer algunas conclusiones que se describen a continuación:

	Tiempo (ms)
Navegación	1,6
Toma de decisiones	1,0
Otros	1,2
Total de la estrategia	3,8

Cuadro 4.6: Tiempo promedio de respuesta de la estrategia y sus componentes.



(a) Evolución del tiempo de procesamiento de la estrategia.

(b) Procentaje de tiempo de procesamiento de cada componente de la estrategia.

Figura 4.19: Tiempo de respuesta de la estrategia.

- El tiempo de cómputo requerido por la estrategia se mantiene estable y cercano al promedio.
- Los valores elevados que aparecen como picos en la gráfica se deben a:
  - la inicialización del sistema de toma de decisiones que requiere la carga dinámica de los predicados difusos.
  - el emborronado a demanda que puede provocar la carga dinámica de predicados que no son utilizados desde el comienzo.
  - la finalización del reconocimiento de las zonas de actividad del oponente (aproximadamente a las 4000 iteraciones), disparando la generación del resultado de la predicción. Esto requiere una gran cantidad de cálculos que se computan concurrentemente con la estrategia, afectando los tiempos de respuesta debido al despacho de los múltiples procesos que componen el sistema.
- El tiempo total promedio requerido por la estrategia en cada iteración es muy inferior a 16 ms (ciclo de simulación). Esto permite perfeccionar las soluciones actuales incrementando su complejidad o sofisticación, aprovechando el tiempo ocioso o disponible.

4.6. Análisis de resultados obtenidos

En las secciones anteriores de este capítulo, se han presentado los resultados de la evaluación del equipo FIbRA, tanto en un ambiente de prueba, como en un ambiente de competencia oficial, así como de los tiempos de respuesta. Estos muestran las deficiencias competitivas del equipo, siendo necesario continuar con el mejoramiento de la planificación de movimientos y el ajuste de la toma de decisiones, además de optimizar el uso de la información suministrada por la predicción. Asimismo, el mejoramiento de la precisión, velocidad y potencia en los movimientos de los robots puede repercutir notablemente en la obtención de mejores resultados.

A pesar de que las evaluaciones específicas de los sistemas de toma de decisiones y predicción parecen indicar que funcionan correctamente, una vez reunidos en la solución final, los resultados globales denuncian limitaciones que pueden no haberse detectado en las pruebas individuales. La información generada para la predicción del comportamiento oponente permite mejorar algunos aspectos defensivos; sin embargo, admite mejoras que permitan manejar el impacto general de esta predicción, a través del uso que se le da en la toma de decisiones. Por otro lado, la toma de decisiones asume una correcta ejecución de las acciones asignadas, pero en muchos casos esto no ocurre (ver sección 3.8.4).

El sistema de toma de decisiones no tiene en cuenta, durante el transcurso de un partido, la contribución o impacto real que significa utilizar la información de predicción. Esta consideración permitiría descartar el uso de dicha información cuando no aporte beneficios para alcanzar el objetivo global. Tampoco tiene en cuenta situaciones particulares que involucran el cumplimiento de las reglas de juego; por ejemplo, cuando los *clusters* generados se encuentran muy cercanos al arco provocan la aglomeración de defensas en torno al golero, limitando sus movimientos o incluso generando faltas en favor del oponente (penales).

A grandes rasgos, el equipo se muestra poco agresivo, con baja tasa de llegadas al arco oponente y poco efectivas. Estas carencias, sumadas a la elevada cantidad de goles en contra, llevan a que los resultados numéricos (tanteador) no sean buenos. A continuación se listan algunas de las posibles causas del bajo rendimiento competitivo:

- Aparición de falsos positivos o falsos negativos[6] en el resultado de la predicción (obtenido, por ejemplo, frente a oponentes con comportamientos aleatorios).
- Utilización de información caduca de la predicción (por ejemplo, frente a oponentes que tengan la capacidad de aprender o adaptar su comportamiento).
- Sub-explotación de la información de la predicción por parte del sistema de toma de decisiones.
- Asignación inadecuada de acciones cuando hay oportunidad de utilizar la información de la predicción.
- Implementación incorrecta de las acciones asignadas.
- Divergencia entre lo que idóneamente debería hacer un robot y lo que efectivamente hace (traducción de acciones a comandos de bajo nivel).

Las conclusiones generales y el trabajo a futuro, que se proponen a partir de éstas y otras observaciones se presentan en el capítulo siguiente.

[6] Falsos negativos son aquellos resultados que deberían haber sido producidos en el proceso de detección de zonas, porque reflejarían correctamente cierto nivel de regularidad en el comportamiento del oponente, pero que no fueron detectados.

Capítulo 5

Conclusiones

La construcción de programas inteligentes que permitan dotar de la habilidad de jugar al fútbol a un conjunto de cinco robots cúbicos de dos ruedas, plantea -cuanto menos- un gran desafío. Esto se debe, en primer lugar, a la complejidad inherente al problema y a la multiplicidad de propuestas para resolverlo, inabordables en su totalidad en un proyecto de grado. En segundo lugar, a los ajustes que requiere una propuesta, una vez elegida, para ser aplicada en el entorno simulado y bajo las reglas de juego que propone la categoría SimuroSot de la FIRA. Finalmente, si se trata de otorgarle un enfoque académico al proceso, el desafío se extiende a la búsqueda de una solución que aspire a la innovación y a la calidad.

Los campeonatos, como el CAFR o el Mundial de la FIRA, son el medio definitivo para medir el perfil competitivo de una solución. En ellos, los diferentes equipos enfrentan las últimas versiones de sus soluciones con las de sus rivales en una arena de juego moderada por un árbitro humano. Como en toda competencia, justa o injustamente, gana la solución que obtiene mejores resultados en cada instancia de juego.

A diferencia de la RoboCup, donde se promueve el desarrollo de soluciones orientadas hacia lo académico, la FIRA propicia un ámbito más inclinado hacia lo competitivo, donde no se estila el intercambio de experiencias, ideas o soluciones como forma de potenciar el aprendizaje. Esta realidad, por un lado, acota la velocidad de superación de los equipos, que sin duda podría catalizarse si -por ejemplo- se fomentara el intercambio de los ejecutables al final de los campeonatos; por otro lado, no ayuda a la motivación de construir soluciones que contribuyan a la comunidad. Como excepción a la regla, en el último CAFR no sólo se evaluó y premió el perfil académico de las soluciones, sino que también se insistió en el intercambio amistoso de los ejecutables que se presentaron en la competencia. El resultado de esta iniciativa de higiene competitiva sin duda se verá capitalizado en el próximo campeonato, cuando los equipos presenten soluciones mejor preparadas.

El aprendizaje cosechado durante la etapa de investigación, y sobre todo la experiencia práctica producto de la construcción del sistema FIbRA permite, tal vez tardíamente, subrayar qué aspectos son de resolución ineludible, para los cuales conviene enfocar el mayor esfuerzo, y cuáles son secundarios, prescindibles o incluso inútiles de resolver.

Como punto de partida, para obtener buenos resultados es indispensable lograr un balance equilibrado entre la "inteligencia" de los robots y su habilidad motora: de nada sirve contar con un elaborado mecanismo que permita tomar muy buenas decisiones si la planificación de movimientos es incapaz de ejecutar correctamente las tareas asignadas; y, por el contrario, de nada sirve que la planificación de movimientos permita movimientos precisos y eficientes si, a mayor nivel de abstracción, el sistema toma decisiones inadecuadas. El aspecto cooperativo también es señalado como un medio para mejorar el desempeño, robustez, tolerancia a fallas, adaptabilidad, disminución de la complejidad del sistema y paralelismo, entre otros.

La solución propuesta muestra que es posible construir un equipo de fútbol de robots capaz de fusionar satisfactoriamente lógica difusa en la toma de decisiones y teoría del comportamiento de insectos en un mecanismo que permite predecir, de manera aceptable, algunas facetas del comportamiento oponente. Además, aporta una arquitectura en capas que encapsula el modelo del mundo -que incluye predicción, detección y monitoreo de distintos aspectos del entorno-, la planificación de movimientos, la toma de decisiones y la interacción con otras aplicaciones, garantizando

la cohesión interna de cada una de las partes, así como el bajo acoplamiento entre ellas.

Los párrafos que siguen sintetizan las conclusiones más sobresalientes acerca de los principales componentes que conforman la solución, así como de aquellas herramientas que pueden facilitar la puesta a punto del equipo.

Predicción Para el componente que permite descubrir el juego posicional del oponente, se encontró que el entorno de aplicación plantea dificultades para alcanzar resultados útiles: el alto dinamismo, la propia morfología de los robots, su gran capacidad de reacción y la relación entre el área de apoyo y el área total de la cancha, reducen la oportunidad de cooperación para aumentar la eficiencia; en particular, de realizar pases, que es lo que -paradójicamente- se pretende detectar.

Toma de decisiones El sistema de toma de decisiones difuso aumenta el nivel de expresividad para el modelado del comportamiento de los robots, frente al nivel que ofrecen otros enfoques más tradicionales. Permite formalizar razonamientos subjetivos e imprecisos más fácilmente, repercutiendo positivamente en la capacidad de mantenimiento del conjunto de reglas que describen el comportamiento. Además, el enfoque difuso, al ser más intuitivo, habilita una interacción directa entre el sistema y el usuario experto, sin exigir que éste posea necesariamente conocimientos de computación.

Como contrapartida, a medida que aumenta la complejidad del modelo, la capacidad de ajustar los parámetros manualmente se vuelve menos factible. Para superar esta limitante podría evaluarse la viabilidad de utilizar alguna técnica de aprendizaje que permita realizar esta tarea automáticamente. Sin embargo, para posibilitar el entrenamiento o aprendizaje, es necesario contar con una herramienta de simulación que admita la ejecución programática de secuencias de partidos o jugadas, característica que no contempla el simulador de la FIRA. Con miras a la construcción de un simulador alternativo, se elaboró un prototipo de motor de simulación para la categoría SimuroSot, que puede ser tomado como punto de partida (ver [ABR06d]).

Planificación de movimientos El alto grado de precisión y eficiencia en los movimientos alcanzado hoy en día por los equipos, queda de manifiesto en los partidos donde compiten los referentes en la materia. Si el objetivo es lograr buenos resultados competitivos, es indispensable contar con un sistema de planificación de movimientos correcto, de alto desempeño, preciso y adaptable a los agresivos cambios propios del entorno. Estos factores motivan una urgente revisión y actualización del sistema de planificación de movimientos que utiliza FIbRA.

Herramientas auxiliares Para la realización de pruebas, ajustes y puesta a punto del equipo es fundamental la utilización de una herramienta auxiliar que permita, por un lado, reproducir partidos o jugadas y, por otro, facilitar la visualización del estado interno del sistema durante la ejecución. La adaptación de la herramienta *LogViewer*, por ejemplo, ayudó al desarrollo de varios componentes del sistema, alentando su uso en emprendimientos de similares características.

Los resultados particulares de la predicción de la formación del oponente y del sistema de toma de decisiones, las ventajas cualitativas emergentes de una arquitectura intuitiva y el poder expresivo que admite el modelado del comportamiento de los robots, dan la pauta de que el camino elegido es, cuanto menos, alentador: la solución propuesta deja abierta la posibilidad de continuar investigando y trabajando sobre lo ya construido.

5.1. Trabajo futuro

Si la terminología que se utiliza usualmente para designar la etapa de desarrollo en que se encuentra un sistema pudiera trasladarse para calificar el grado de evolución que posee un equipo de fútbol de robots, podría decirse que FIbRA, como equipo, se encuentra en la versión *alfa*. Por un lado, los movimientos de los robots carecen del nivel de eficiencia requerido en la actual competencia. Por otro, tanto el modelado del comportamiento de los robots, a través de la declaración

de reglas difusas propuesta, como la especificación de predicados difusos, acciones, roles y estrategias, requieren de más pruebas y ajustes para mejorar el comportamiento existente, o incluso de la ampliación del modelo para lograr una mayor expresividad.

A continuación se propone un resumen con las líneas de trabajo a futuro más trascendentes.

Planificación de movimientos

Para lograr una mejor eficiencia en el movimiento de los robots es fundamental contar con un sistema de planificación de movimientos adecuado a las exigencias competitivas de hoy en día. Tomando como punto de partida las limitaciones del sistema reutilizado, se presentan las siguientes alternativas:

- Determinar la viabilidad de reformar el algoritmo de planificación de movimientos reutilizado para mejorar sus prestaciones.
- Desarrollar un sistema de planificación de movimientos (o modificar el existente) que integre explícitamente otras restricciones al modelo, como el tiempo, para aumentar la eficiencia.
- Modificar el modelo predictivo de la trayectoria de la pelota y de la configuración futura de los robots para que:
  - contemple los límites de la cancha;
  - contemple colisiones entre la pelota y los robots;
  - contemple colisiones entre los robots;
  - se puedan realizar predicciones a mayor plazo.
- Mejorar el control de movimiento *Direct LV* agregándole una maniobra de aparcamiento -del mismo modo que lo realizado para *Direct Ang*-, para aumentar la estabilidad de los robots entorno al objetivo.

Predicción del comportamiento oponente

El enfoque utilizado agrega al reconocimiento del comportamiento oponente un componente subjetivo muy particular, lo cual dificulta el proceso de ajuste. Por otro lado, las propias características del entorno que proporciona el simulador oficial obstaculizan la extracción de información sobre el juego posicional del oponente. Las alternativas de trabajo presentadas a continuación intentan abordar esta problemática, así como alentar la reutilización de este sistema en otros entornos:

- Continuar con el ajuste de los parámetros que controlan la cantidad de feromona depositada y evaporada, la política de evaporación, el tiempo de procesamiento, el umbral de aceptación de celdas y el tamaño de los *clusters*.
- Intentar formalizar aspectos relacionados con la frecuencia de la regularidad del comportamiento oponente con la finalidad de enriquecer el modelo, evaluando el impacto que esta incorporación puede tener en la cantidad de falsos positivos y negativos.
- Evaluar la factibilidad de utilizar la predicción del juego posicional del oponente en otras ligas simuladas, como la RoboCup, cuyo entorno facilita la cooperación directa entre robots.
- Ampliar el modelo sobre el que sustenta la detección de pases a los efectos de que las aristas del grafo que representa el comportamiento del juego oponente no se ponderen exclusivamente en función de los pases consumados, sino también cuando el oponente tiene intención de realizarlos.

Toma de decisiones

El sistema construido para modelar el comportamiento de los robots, basado en predicados difusos, alcanza para cubrir el funcionamiento básico de un equipo de fútbol de robots. Aunque este enfoque admite mejoras, debe tenerse en cuenta el balance entre habilidad motora e inteligencia, priorizando las correcciones al sistema de planificación de movimientos sobre las que se listan a continuación.

- Modificar la forma en que se cargan los predicados difusos para posibilitar que el sistema de toma de decisiones considere conjuntos de datos multi-particionados, aumentando aún más su poder de expresividad.
- Utilizar alguna técnica de aprendizaje automático para ajustar los parámetros del sistema, sobre todo los ponderadores utilizados para relacionar acciones, roles y estrategias.
- Definir nuevos roles y acciones para aumentar la cooperación directa entre los robots.

Herramientas auxiliares

- Ajustar el prototipo de motor de simulación construido para aproximarlo al modelo de simulación de la FIRA (ver Conclusiones y Trabajo futuro del documento [ABR06d]).
- Tomando como punto de partida este motor de simulación, desarrollar una herramienta de simulación alternativa que posibilite el entrenamiento automático del equipo.
- Mejorar la interfaz y las prestaciones del reproductor de *logs* y depurador *LogViewer* para que permita detener la reproducción de manera automática en algún punto de la ejecución (por ejemplo, a través de *breakpoints* condicionales e incondicionales), así como definir modos de ejecución que permitan el avance y retroceso paso a paso.

Apéndice A

Glosario

Actuador Los actuadores o efectores de un robot son las partes que le permiten modificar el entorno en el que están situados.

CAFR El IV Campeonato Argentino de Fútbol de Robots tuvo lugar en la Universidad Interamericana Argentina (UAI) junto al II *Workshop* en Inteligencia Artificial aplicada a la Robótica Móvil, durante los días 11, 12, 13 y 14 de Julio de 2006. El sitio oficial del evento es: `www.vaneduc.edu.ar/cafr/`. Disponible Oct/2006.

Cluster En el contexto de la propuesta realizada para reconocer el comportamiento del juego oponente, se refiere a los conjuntos o aglomeraciones de celdas (de la matriz que representa la cancha) que registran cierta concentración de feromona y representan las zonas donde el oponente registra mayor actividad.

Configuración de un robot La configuración de un robot es un conjunto de parámetros independientes que definen de manera única la posición y la orientación de cada punto del robot.

Deadlock En el contexto de los sistemas operativos, representa el bloqueo permanente de un conjunto de procesos o hilos de ejecución en un sistema concurrente que compiten por recursos del sistema o bien se comunican entre ellos. En el contexto del fútbol de robots se refiere a las situaciones donde, por limitaciones de la toma de decisiones, los robots permanecen sin moverse a pesar de no tener impedimentos para hacerlo.

Efector Ver **Actuador**.

Falsos Negativos Son aquellos resultados que deberían haber sido producidos en el proceso de detección de zonas, porque reflejarían correctamente cierto nivel de regularidad en el comportamiento del oponente, pero que no fueron detectados.

Falsos Positivos Son aquellos resultados producidos en el proceso de detección de zonas, que reflejan cierto nivel de regularidad pero que no se condice con el comportamiento voluntario del oponente.

Feromona Sustancia o conjunto de sustancias químicas producidas por algunos organismos vivos que las utilizan para transmitir mensajes a otros miembros de la misma especie. Hay varios tipos de feromona, como por ejemplo de alarma, de rastros al alimento, sexuales, y otras que afectan el comportamiento de los individuos de la especie.

FIRA Acrónimo de *Federation of International Robot-soccer Association*. La FIRA se creó el 5 de junio de 1997 durante el torneo MiroSot97 (*Micro-robot Soccer Tournament*) realizado en Daejeon, Corea. Desde entonces todos los años organiza dos eventos de nivel internacional: FIRA *Robot World Cup* y FIRA *Robot World Congress* con el propósito de promover la investigación en sistemas robóticos autónomos entre otros.

Livelock Es similar a un *deadlock*, excepto que el estado de los dos procesos envueltos en el *livelock* constantemente cambia. En el contexto del fútbol de robots se refiere a las situaciones donde,

por una combinación fortuita de acontecimientos en el entorno, el comportamiento de los robots adquiere una regularidad cíclica inmovilizante.

LogViewer Es un visor de *logs* desarrollado por el proyecto FRUTo y reutilizado para el equipo FIbRA

RoboCup Originalmente llamada *Robot World Cup Initiative* es una iniciativa internacional de investigación y educación. En ese sentido intenta fomentar la investigación en Inteligencia Artificial y robótica proporcionando un problema estándar donde una amplia gama de tecnologías puede ser integrada y examinada, así como promover proyectos de educación integrada.

Sistema multi-robots Son un clase de sistema multiagente donde los agentes se encuentran integrados a un entorno físico concreto -agentes situados-.

Swarm robotics Es una aproximación que intenta resolver el problema de coordinación en sistemas multi-robots. El objetivo es construir/diseñar agentes simples a partir de los cuales emerjan comportamientos colectivos. Una fuente de inspiración son las sociedades de insectos, aunque no la única.

Visor de *logs* Herramienta auxiliar que permite la reproducción de partidos grabados previamente con un formato particular.

Zona de actividad Representa una zona de la cancha donde uno o varios robots oponentes se mueven con cierta regularidad durante el transcurso de un partido.

Zona libre Son áreas del campo de juego libres de obstáculos. En [ABR05a] se profundiza este concepto.

Bibliografía

[ABR05a] Gustavo Armagno, Facundo Benavides, and Claudia Rostagnol, *Estado del arte*, Tech. report, Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay, 2005. 17, 19, 20, 65, 67, 71, 74, 104

[ABR05b] ———, *Evaluación de un prototipo de simulador simil fira, desarrollado con la herramienta phi*, Tech. report, Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay, 2005. 57

[ABR05c] ———, *Reglas para simurosot*, Tech. report, Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay, 2005. 35, 38, 39, 69, 70, 76, 85

[ABR06a] ———, *Descripción de la arquitectura del sistema*, Tech. report, Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay, 2006. 29, 31, 34, 39, 58, 74

[ABR06b] ———, *Especificación suplementaria de requerimientos*, Tech. report, Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay, 2006. 32, 35, 71

[ABR06c] ———, *Fibra: Toma de decisiones difusa y predicción del comportamiento oponente*, WCAFR2005 III WorkShop de Inteligencia Artificial Aplicada a la Robótica Móvil, Buenos Aires, Argentina (2006), Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay. 15

[ABR06d] ———, *Robosoccer engine*, Tech. report, Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay, 2006. 84, 100, 102

[BKC94] H. R. Beom*, K. C. Koh*, and H. S. Cho**, *Behavioral control in mobile robot navigation using fuzzy decision making approach*, Intelligent Robots and Systems '94. 'Advanced Robotic Systems and the Real World', IROS '94. Proceedings of the IEEE/RSJ/GI International Conference on, Munich, Germany, vol. 3, Sep 1994, *Research & Development Laboratory, Gold Star Industrial Systems, Anyang, Korea. **Department of Precision Engineering and Mechatronics, Korea Advanced Institute of Science and Technology, Taejon, Korea, pp. 1938–1945. 23

[CLH+05] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun, *Principles of robot motion - theory, algorithms, and implementations*, The MIT Press, 2005, ISBN 0-262-03327-5. 24, 69, 70

[Cob02] Clifton F. Cobb, *Fuzzy logic*, Alabama Journal of Mathematics, Alabama, USA (2002), 13–24, University of West Alabama, College of Natural Science and Mathematics, Department of Mathematics, Alabama, USA. 22, 23

[dIO06] Departamento de Investigación Operativa, *Introducción a la investigación operativa: Repartido teórico*, Tech. report, Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay, 2006. 63

[DS04] Marco Dorigo and Thomas Stützle, *Ant colony optimization*, The MIT Press, Jul 2004, ISBN-10: 0-262-04219-3, ISBN-13: 978-0-262-04219-2. 19, 20

[dTdI06] GTI Grupo de Tratamiento de Imágenes, *Sistemas de reconocimiento de patrones*, Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República, Uruguay, 2006. 17, 18

[ECA+03] Francisco Escolano, Miguel A. Cazorla, M. Isabel Alfonso, Otto Colomina, and Miguel A. Lozano, *Inteligencia artificial modelos, técnicas y Áreas de aplicación*, Thomson, 2003, ISBN 84-9732-183-9. 23

[ENW05] Uwe Egly*, Gregory Novak**, and Daniel Weber*, *Decision making for mirosot soccer playing robots*, CLAWAR/EURON/IARP Workshop on Robots in Entertainment (2005), *Institute of Information Systems and **Institute of Computer Technology, Vienna University of Technology, Vienna, Austria. 23

[Fra99] Thierry Fraichard, *Trajectory planning in a dynamic workspace: a 'state-time space' approach*, Advanced Robotics (1999), INRIA Rhône Alpes, Francia. 24

[FS94] Thierry Fraichard and Alexis Scheuer, *Car-like robots and moving obstacles*, IEEE Int. Conf. on Robotics and Automation, San Diego, California (1994), LIFIA-INRIA Rhône Alpes-IMAG, Francia. 24, 25

[FS95] Paolo Fiorini* and Zvi Shiller**, *Robot motion planning in dynamic environments*, Tech. report, *Jet Propulsion Laboratory, Pasadena, California, Estados Unidos. **Department of Mechanical, Nuclear and Aerospace Engineering, University of California, Los Angeles, Estados Unidos, 1995. 25

[FST04] Gastón Fernández, Claudia Stocco, and Natalia Tourn, *Sistema de visión para fútbol de robots*, Proy. de Grado, Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay, 2004. 44

[FTIAM00] Ian Frank*, Kumiko Tanaka-Ishii**, Katsuto Arai***, and Hitoshi Matsubara****, *The statistics proxy server*, RoboCup 2000: Robot Soccer World Cup IV, Melbourne, Australia (2000), 303–308, *Complex Games Lab, ETL, Tsukuba, Japan. **Tokyo University, Japan. ***Chiba University, Japan. ****Future University, Hakodate, Japan. 84

[Ind01] Giovanni Indiveri, *On the motion control of a non holonomic soccer playing robot*, RoboCup Symposium 2001, Seattle, USA (2001), GMD-AiS, German National Research Center on Information Technologies, Institute for Autonomous Intelligent Systems, Schloss Birlinghoven, Alemania. 71, 76

[KB91] Yehuda Koren and Johann Borenstein, *Potential field methods and their inherent limitations for mobile robot navigation*, Proceedings of the IEEE Conference on Robotics and Automation (Sacramento, California), IEEE, Apr 1991, The University of Michigan, Michigan, Estados Unidos. 25

[Lar02] Harold J. Larson, *Introduccion a la teoria de probabilidades e inferencia estadistica*, Editorial Limusa S.A., Jan 2002, ISBN: 9681807308. 18

[LF04] Tomás Laurenzo and Gabrielle Facciolo, *Una herramienta de análisis de estrategias de fútbol de robots middle league simurosot*, WCAFR2004 - Workshop en inteligencia artificial aplicada a robotica movil, Facultad de Ciencias Exactas, Universidad Nacional del Centro de la Provincia de Buenos Aires, Buenos Aires, Argentina (2004), Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay. 19, 38

[MKK04] Drago Matko, Gregor Klancar, and Rihard Karba, *Analysis of multi-agent collaborative systems*, 2nd International Conference on Autonomous Robots and Agents, Palmerston North, New Zealand, Dec 2004, Faculty of Electrical Engineering, University of Ljubljana, Slovenia, pp. 271–276. 23

[MMM05] Maximiliano L. Muzzio, Leandro M. Di Matteo, and Andrea C. Mangone, *Equipo de fútbol de robots utnfrba*, WCAFR2005 II WorkShop de Inteligencia Artificial Aplicada a la Robótica Móvil, Facultad de Informática, Ciencias de la comunicación y Técnicas especiales, Universidad de Morón, Argentina, Jun 2005, Grupo de Inteligencia Artificial, Facultad Regional Buenos Aires, Universidad Tecnológica Nacional, Buenos Aires, Argentina. 23

[Mur83] Katta. G. Murty, *Linear programming*, John Wiley & Sons, New York, USA, Oct 1983, ISBN: 0-471-09725-X. 63

[PAC04] Vasco Pires, Miguel Arroz, and Luis Custódio, *Logic based hybrid decision system for a multi-robot team*, In Proceedings of the Eighth Conference on Intelligent Autonomous Systems, Mar 2004, Institute for Systems and Robotics, Lisboa, Portugal. 21, 22

[PP02] Sankar Pal and Amita Pal, *Pattern recognition, from classical to modern approaches*, World Scientific Publishing Company, Jan 2002, ISBN: 9810246846. 18

[RN02] Stuart Russell and Peter Norvig, *Artificial intelligence: A modern approach*, segunda ed., Prentice Hall, Dec 2002, ISBN: 0137903952. 13, 20, 32

[Sic05] Jaime S. Sichman, *Arquiteturas de agentes*, Brazil Agents School 2005, SBC, Sep 2005. 20

[Sim87] G. L. Simons, *Introducción a la inteligencia artificial*, Díaz de Santos, 1987, ISBN 84-86251-54-0. 13

[TLJ97] Edward Tunstel*, Tanya Lippincott**, and Mo Jamshidi**, *Behavior hierarchy for autonomous mobile robots: Fuzzy-behavior modulation and evolution*, International Journal of Intelligent Automation and Soft Computing (1997), *NASA, Jet Propulsion Laboratory, California, USA. **NASA Center for Autonomous Control Engineering, Department of Electrical and Computer Engineering, University of New Mexico, New Mexico, USA. 23

[Val06] Steven M. La Valle, *Planning algorithms*, Cambridge University Press, Cambridge, U.K., 2006, URL http://planning.cs.uiuc.edu/, Disponible 09/2006. 25