FIbRA: Toma de decisiones difusa y predicción del comportamiento oponente

Armagno Gustavo, Benavides Facundo, Rostagnol Claudia
Tejera Gonzalo, Copello Ernesto

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

garmagno@gmail.com, facundo.benavides@gmail.com, crostagnol@gmail.com
gtejera@fing.edu.uy, ecopello@fing.edu.uy

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

Resumen

FIbRA es un equipo de la liga simulada SimuroSot de la FIRA que utiliza un sistema de toma de decisiones basado en lógica difusa (*fuzzy logic*) para determinar las acciones a ejecutar. Para potenciar esta toma de decisiones, se genera meta-información que permite predecir, entre otras cosas, el patrón de comportamiento del equipo oponente. Esta predicción se realiza mediante la combinación de meta-información generada por diferentes componentes. Uno de estos componentes es modelado siguiendo teorías del comportamiento de insectos, simulando el uso de rastros de feromonas utilizado por las hormigas para comunicarse. Se presentan los resultados obtenidos en materia de predicción de jugadas y comportamientos emergentes derivados de la toma de decisiones difusa, así como los pasos futuros en este sentido.

Palabras claves: Fútbol-de-Robots, Predicción-de-patrones-de-juego, Comportamientos-de- insectos, Toma-de-decisiones, Lógica-difusa.

1. Introducción

El equipo FIbRA se construye en el marco de un proyecto de grado de la carrera Ingeniería en Computación de la Facultad de Ingeniería de la Universidad de la República. Surge como resultado de la investigación de Inteligencia Artificial, Aprendizaje Automático, Sistemas Multi-Agente y sistemas de toma de decisiones, aplicadas al fútbol de robots. Se desarrolla una estrategia basada en un sistema de toma de decisiones de tres niveles. Este sistema de toma de decisiones utiliza predicados basados en lógica difusa para determinar el estado del ambiente y seleccionar la acción que cada robot debe llevar a cabo. Para potenciar la toma de decisiones se incorpora un módulo de predicción que permite, entre otras cosas, obtener meta-información sobre el comportamiento del equipo oponente y el estado del juego.

Este artículo presenta las características más relevantes del sistema, centrándose en las dos características más destacadas: predicción del comportamiento oponente y toma de decisiones a través de predicados basados en lógica difusa. La sección 2 ofrece un esquema general

de la solución a nivel estructural. La sección 3 se refiere a la predicción que el sistema es capaz de realizar. La estrategia y la toma de decisiones se analizan en la sección 4. Por último, la sección 5 presenta los resultados, conclusiones y trabajos a futuro.

2. Estructura general del sistema

El sistema se compone de varios módulos, cada uno de los cuales cumple una tarea específica. La figura 1 resume los conceptos claves del esquema del sistema.

Toda la comunicación con el entorno, en este caso el simulador SimuroSot, se realiza a través del módulo *simulator*. Los datos del ambiente son "percibidos" (en este caso recibidos desde el simulador) y encapsulados para ser enviados a los restantes módulos del sistema. Las modificaciones del ambiente o de la forma de percibir los datos son absorbidas por este módulo, siendo responsable de obtener toda la información que el resto del sistema espera o necesita.

El módulo *prediction* es capaz de generar meta-información derivada de la observación del ambiente. Cuenta con submódulos especializados en determinada sección de los datos. La combinación de la meta-información generada por los submódulos conforma el modelo del mundo que el sistema utiliza.

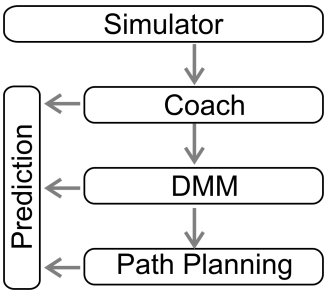

Figura 1: Estructura general del sistema FIbRA.

La determinación de la acción que debe realizar cada robot se lleva a cabo en el módulo *DMM* (*Decision Making Module*). La toma de decisiones se basa en el modelo del mundo generado a partir del ambiente y la predicción.

El *Coach* es responsable de alimentar al módulo *prediction* con los datos del ambiente percibidos por el módulo *simulator*, así como de determinar qué componentes de la predicción estarán disponibles para el *DMM*. Esto último conlleva a que el *Coach* determine la meta-información que el sistema genera, afectando directamente el modelo del mundo.

La puesta en práctica de la acción asignada a cada robot, es llevada a cabo por el módulo *PathPlanning*. Este módulo conoce las características físicas de los robots y, por tanto, tiene la responsabilidad de generar los movimientos necesarios para cumplir una acción específica. Se han incorporado las acciones y controles de movimiento del equipo FRUTo [CCTC05] para implementar este módulo.

3. Predicción

La generación de meta-información del módulo *prediction* se realiza en dos niveles de abstracción.

En un primer nivel se definen detectores o filtros. Cada detector se especializa en determinado aspecto de los datos del ambiente, generando la meta-información asociada. Por ejemplo, se implementa un detector de goles que se especializa en observar la pelota y determina cuándo se ha producido un gol.

En un segundo nivel de abstracción, se cuenta con predictores y monitores. Éstos utilizan la meta-información generada por los detectores, combinándola o refinándola, para generar nueva meta-información más enriquecida. Por ejemplo, se implementa un monitor del resultado parcial del juego, basado en la información proporcionada por el detector de goles.

Los predictores se caracterizan por procesar información durante un período determinado, finalizando su proceso al generar el resultado. En contraposición, los monitores pueden procesar información sin tiempo límite, generando resultados parciales, sin necesidad de finalizar su ejecución.

El procesamiento realizado por algunos detectores, o el procesamiento del resultado de éstos por parte de los predictores y monitores, puede insumir un tiempo mayor que el tiempo de respuesta esperado del sistema. Para mejorar e independizar los tiempos de respuesta del sistema de los tiempos de procesamiento de la predicción se implementa un sistema que

paraleliza la ejecución de la predicción con la del resto del sistema. Esta implementación se basa fuertemente en la utilización de hilos para lograr paralelismo y en el patrón *pipes & filters* para combinar detectores que generan la meta-información esperada.

3.1. Pipes & filters en la predicción

Una red de *pipes & 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 el *sink* o en los *pipes* de salida. Al final del proceso, el *sink* contendrá el resultado final.

Para el sistema FIbRA se implementa un *framework* que maneja redes de *pipes & filters*. 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. La figura 2 muestra un ejemplo de red de *pipes & filters*.

3.2. Predicción del patrón de juego del oponente

Esta habilidad o capacidad refiere a intentar predecir el comportamiento del equipo contrario para poder anticipar sus jugadas, mejorar la defensa y eventualmente el ataque del equipo FIbRA. Para lograrlo, primeramente es necesario analizar el juego del equipo oponente durante un cierto periodo de tiempo. Este análisis implica la alimentación constante del predictor, con la información del ambiente. Una vez transcurrido el tiempo de detección, se dispara el procesamiento de toda la información recolectada para generar la meta-información que representa el comportamiento del equipo oponente hasta ese momento.

Parte de esta meta-información continúa actualizándose a lo largo de todo el juego, mientras que otra parte se mantiene estática, debido a la complejidad que conlleva su actualización.

El predictor de comportamiento se construye sobre una red de *pipes & filters* con la topología mostrada en la figura 2. A continuación se describe el proceso realizado para obtener la meta-información de comportamiento.

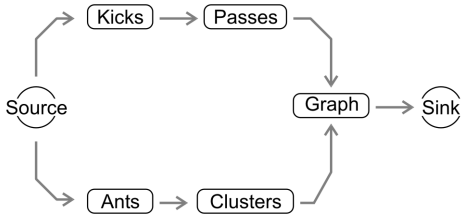

Figura 2: Red de pipes & filters del predictor de comportamiento del oponente.

3.2.1. Detectar zonas de mayor actividad del equipo contrario

Se determinan las zonas por donde se mueven los robots oponentes. Para lograr esto, se utilizan ideas derivadas de teorías sobre el comportamiento de algunos insectos, en particular las hormigas (*Ants*).

Las hormigas son insectos casi ciegos; sin embargo, son capaces de salir del nido, encontrar la fuente de alimento y regresar nuevamente a su nido. Cada hormiga deposita determinada cantidad de una sustancia química denominada feromona a lo largo del camino que va recorriendo. Es esta feromona la que le permite encontrar el camino de regreso. A medida que más y más hormigas pasan por un mismo camino, la concentración de feromona va aumentando. Por otra parte, la feromona se evapora con el tiempo, por lo que el rastro desaparece si no es reforzado con el tránsito continuo. De esta forma, las zonas más transitadas quedan determinadas por una mayor concentración de feromona, mientras que las zonas menos transitadas pierden su rastro con el tiempo [PZ04].

Estas ideas han sido llevadas a los sistemas informáticos desde hace unos años, principalmente en problemas de optimización [DG97].

Para implementar la detección de zonas de mayor actividad del equipo oponente, se toman como base estas ideas y se modela el equipo

oponente como un grupo de hormigas. Cada robot es visto como una hormiga, que deposita cierta cantidad de feromona sobre la posición en la que se encuentra. Asimismo, esta feromona es evaporada cada cierto intervalo de tiempo, con lo cual, los rastros débiles desaparecen rápidamente. No se distingue entre una hormiga u otra, todas son idénticas y depositan la misma cantidad de feromona. Al final del proceso, sólo los rastros fuertes perduran, o sea, aquellos lugares donde la concentración de feromona es muy alta, debido al tránsito continuo de robots.

La figura 3 muestra una imagen obtenida como resultado de este proceso luego de tres minutos de juego. El color oscuro de fondo (valor 0 en la referencia) indica una concentración de feromona nula. Las manchas con color se deben a la presencia de feromona, determinando la concentración de la misma el color de la mancha. El color celeste (valor cercano a 3) indica baja concentración, mientras que el color rojo (valor superior a 6) indica una concentración de feromona alta.

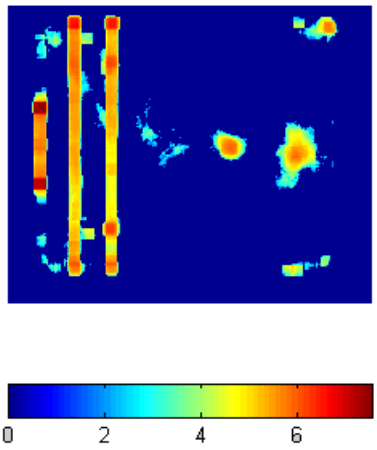

Figura 3: Ejemplo de zonas calientes detectadas utilizando ideas de *Ants*.

Las manchas alargadas sobre la izquierda de la figura determinan el comportamiento de la defensa del equipo, mientras que las manchas más circulares de la derecha determinan el comportamiento del equipo en ataque. Se observan dos grandes zonas de influencia frente al arco, lo que puede interpretarse como que el oponente busca el gol por el medio del campo de juego. Derivada de la observación del partido y el resultado de la imagen, las manchas sobre las esquinas de la derecha se deben al desborde que genera el equipo por las puntas para generar el pase (centro) al atacante central.

Finalizado el proceso basado en hormigas, se debe procesar la meta-información generada, para obtener las zonas de influencia relevantes. Este procesamiento se lleva a cabo a través de un algoritmo de clasificación, segmentación e identificación de las manchas de feromona. Este proceso se compone de tres fases y está basado en el algoritmo de procesamiento de imágenes presentado por el proyecto de grado VisRob [FST04].

La primer fase recorre el campo de juego eliminando los rastros de feromona débiles. La segunda fase implica eliminar zonas muy pequeñas, en este caso, aquellas menores que el tamaño de un robot. Por último, la tercer fase corrige la información para obtener zonas de forma convexa. La figura 4 muestra el resultado obtenido luego de procesar el resultado de la figura 3.

Las zonas derivadas de este análisis no son modificadas durante el resto del partido. Si el equipo oponente cambia sus zonas de ataque, el sistema no podrá detectarlo.

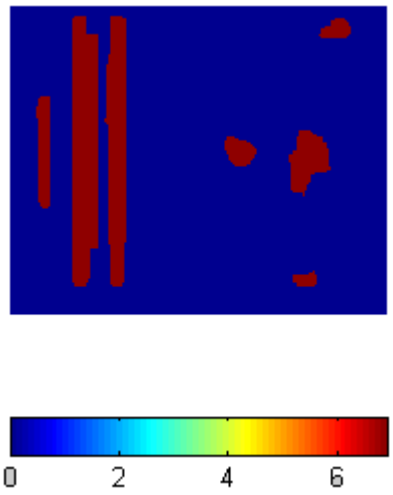

Figura 4: Procesamiento de la detección de zonas calientes mostradas en la figura 3.

3.2.2. Detectar pases del equipo oponente

En simultáneo con la detección de zonas calientes, se detectan los golpes (*kicks*) entre los robots y la pelota, y, a partir de estos golpes, construir la secuencia de pases del equipo oponente. En cierta forma se busca detectar patrones de cooperación entre los jugadores del equipo contrario al momento de realizar jugadas de ataque o de defensa.

Al finalizar la recolección de estos datos, se cuenta con un mapa de juego del oponente que permite determinar las zonas más utilizadas por el equipo oponente para atacar, y los pases realizados en dicho ataque.

3.2.3. Detectar jugadas del equipo contrario

Una vez detectadas las zonas de ataque del equipo oponente y los pases realizados en situaciones ofensivas, se procesa esta información, combinando el resultado de ambos análisis, generando meta-información que contenga la relación de precedencia entre las zonas identificadas. Esta relación indica el origen y destino probable de un pase desde una región de la cancha donde el oponente tiene la pelota.

El resultado puede interpretarse como un mapa del juego oponente, donde se indican las zonas de influencia del equipo, las secuencias de pases realizadas entre dichas zonas y la frecuencia o probabilidad con la que se realizan estos pases. Asimismo, puede ser interpretado como un grafo que conecta las zonas de influencia mediante pases realizados en alguna jugada de ataque del equipo oponente. Cada arista de este grafo, derivada de un pase realizado, contiene un peso que indica la probabilidad de ocurrencia del mismo, obtenida en base a la frecuencia de pases realizados en la etapa de recolección de información.

Este resultado final de la predicción, puede ser utilizado para anticipar las jugadas del oponente, pudiendo interceptar sus pases o mejorar la defensa reubicando los robots propios en las zonas de mayor presencia oponente.

Debido a las características del ambiente no se han obtenido grafos de juego que permitan anticipar pases o jugadas, pero sí se utilizan las zonas calientes para mejorar la defensa del equipo. La colocación de robots propios en las zonas calientes pretende obstaculizar las jugadas más frecuentes del oponente, con lo cual se evitarían jugadas de riesgo.

3.3. Otros predictores y monitores

Además de predecir el comportamiento del equipo oponente, se ha dotado al sistema de otras herramientas de predicción, que potencian la toma de decisiones.

Una de éstas *(Tracking Predictor)* tiene la responsabilidad de "observar" los objetos del ambiente y sus movimientos, para poder predecir el comportamiento esperado en el corto plazo. Permite determinar también los obstáculos que pueden interponerse en una trayectoria determinada. Este predictor esta basado en el módulo de *tracking* implementado por el equipo FRUTo [CCTC05], el cual almacena la historia reciente de los objetos para poder realizar luego, los cálculos de predicción a futuro.

La toma de decisiones podría verse afectada por el resultado parcial del partido, por lo que se implementa un monitor del resultado del juego. Este monitor ofrece la información actualizada de los goles convertidos por cada equipo.

Por otra parte, se ha incorporado un detector de atascamientos que determina si hay robots propios atascados. Esta información puede ser utilizada para asignar acciones diferentes a los robots libres y a los atascados, dado que un robot atascado probablemente no podrá cumplir con su objetivo hasta que pueda desatascarse. Un robot se considera atascado si durante una cantidad determinada de iteraciones no ha podido cumplir con sus acciones asignadas. Esto significa que el robot no ha logrado desplazarse, siendo que sí se le han asignado acciones que implicaban su desplazamiento.

4. Estrategia Fuzzy

El problema de mantener y modificar una estrategia implementada como máquina de estados genera la búsqueda de una alternativa. Surge entonces la opción de construir una nueva estrategia, inspirada fuertemente en la idea propuesta por Uwe Egly et. al. [ENW05], construyendo un sistema de toma de decisiones conformado por predicados basados en lógica difusa.

Este enfoque esta basado en reglas lógicas, construidas a partir de estos predicados, que determinan cuál es la mejor decisión en cada momento.

4.1. Toma de decisiones

Se construye un *framework* de toma de decisiones, cuya responsabilidad es determinar cuál es la mejor acción que cada robot debe llevar a cabo en un momento determinado, para cumplir con el objetivo global del equipo.

La toma de decisiones se lleva a cabo en tres niveles. En una primer etapa se determina la estrategia de juego a ser aplicada por el equipo. Una vez determinada la estrategia de juego, se determina qué rol o tarea debe cumplir cada robot, en base a la estrategia elegida. Por último, se selecciona la acción que debe llevar a cabo cada robot, en base al rol que se le ha asignado.

Para seleccionar una alternativa apropiada en cada nivel, se debe consultar el estado global del juego, y tomar las decisiones en base a éste. Se construyen reglas lógicas, que evalúan dicho estado para determinar la adecuación de una determinada decisión.

Cada regla lógica se compone de predicados, que son la unidad atómica del *framework* de toma de decisiones, los cuales son evaluados para determinar la factibilidad del objetivo de la regla. Por ejemplo, una regla para determinar la estrategia de juego a adoptar puede ser modelada de la siguiente forma:

estrategiaX :- predicadoA AND (predicadoB OR NOT predicadoC)

donde *estrategiaX* es el objetivo de la regla y los demás componentes son los predicados basados en lógica difusa, conectados por operadores lógicos que aplican lógica difusa.

Además, cada regla está influenciada por la decisión del nivel inmediatamente superior (excepto para el primer nivel), por lo que se pondera el resultado de la evaluación de los predicados con el peso asignado a la selección anterior. Esto significa que una acción queda influenciada por el rol asignado al robot. La acción de defender el arco tiene mayor peso para un robot golero que para un atacante (como muestra la figura 5), ya que se considera que un golero está en mejores condiciones de desempañar esta tarea.

```
<Unit name="DefendGoal" desc="Action">
 <Weight name="Backward" value="0.3" />
 <Weight name="Forward" value="0.3" />
 <Weight name="Goalkeeper" value="0.7" />
 <RuleSet>
  <And>
   <Not>
    <Value term="Stucked" />
   </Not>
   <Value term="HomeAtHomeGoal" />
   <Or>
    <Value term="BallInHomeGoalDirection"/>
    <Value term="BallNearHomeGoal" />
   </Or>
  </And>
 </RuleSet>
</Unit>
```

Figura 5: Regla lógica para la acción DefendGoal definida en XML.

Para ofrecer un manejo simple y extensible de este *framework* de toma de decisiones, se utiliza un lenguaje de definición de reglas basado en XML. Todas las reglas a evaluar se detallan en un archivo para cada nivel. De esta forma, la modificación de una regla es simple: se reduce a la modificación del archivo correspondiente evitando la recompilación del código fuente. Esta característica dota al *framework* de escalabilidad en lo referente a la capacidad de modelado de la solución. Por lo tanto, éste es independiente del grado de complejidad con que se modele la solución.

Los operadores permitidos para construir las reglas son AND, OR y NOT, utilizando lógica difusa en este caso [Cob02]. Internamente, cada regla se implementa como un árbol lógico de evaluación, donde los nodos internos son los operadores lógicos, y las hojas corresponden a predicados, como se muestra en la figura 6.

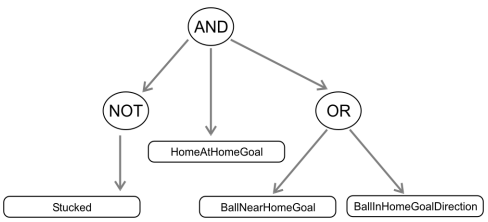

Figura 6: Ejemplo de árbol de evaluación de un predicado.

El proceso de ajuste se realiza en forma manual. Esta característica se transforma en un problema cuando el modelo crece en complejidad. Una línea de trabajo futuro es investigar la viabilidad de utilizar aprendizaje automático para lograr ajustes sub-óptimos.

5. Conclusiones

Se han observado comportamientos emergentes principalmente relacionados con el rol golero.

El intercambio de roles entre los robots es dinámico, basado exclusivamente en su aptitud para cumplir con un rol y una acción determinados.

El esquema de toma de decisiones descrito, basado en archivos de configuración, ha permitido que se mejorara el nivel de juego modificando solamente ponderadores y reglas de decisión, sin necesidad de recompilar el código.

Sin embargo, la cantidad de variables a ajustar puede crecer significativamente a medida que aumenta la complejidad del modelo. Probablemente la calidad del ajuste pueda ser mejorada utilizando técnicas de aprendizaje automático.

Otra característica de la solución, es su capacidad de adaptación a cambios en el ambiente. La misma puede ser utilizada en ligas con mayor o menor número de robots, jugando en canchas más grandes o más pequeñas, sin tener que hacer cambios significativos.

En lo referente a la predicción del comportamiento oponente, se han obtenido resultados alentadores derivados de detección de zonas calientes y procesamiento de las mismas. Sin embargo, a pesar de lograr detectar las secuencias de pases correctamente, las características morfológicas de los robots y el dinamismo del juego no permitieron que el grafo de comportamiento fuera de utilidad práctica: de todos los pases detectados, sólo una ínfima parte conecta zonas de influencia.

Si bien el sistema se encuentran en una fase inicial de experimentación y ajuste, los resultados preliminares no descartan la viabilidad de la solución propuesta para lograr un equipo competitivo. Comportamientos emergentes e intercambio dinámico de roles como los que se verificaron entre golero y defensas, podrían obtenerse también entre atacantes y defensas mejorando la configuración de la toma de decisiones.

Algunas líneas de investigación a futuro podrían ser: incorporar mecanismos explícitos de cooperación entre robots que sean compatibles con el modelo de toma de decisiones planteado; aplicar las ideas utilizadas para predecir el comportamiento del oponente en una liga donde los robots y el ambiente presenten otras características.

Referencias

[CCTC05] Alvaro Castroman, Ernesto Copello, Gonzalo Tejera, and Nelson Calero, *Fruto fútbol de robots uruguayo para torneos*, 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, Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay.

[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.

[DG97] Marco Dorigo* and Luca Maria Gambardella**, *Ant colonies for the traveling salesman problem*, BioSystems, Stockholm, Swedish (1997), *IRIDIA, Université Libre de Bruxelles, Bruxelles, Belgium. **IDSIA, Lugano, Switzerland.

[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.

[FST04] Gastón Fernández, Claudia Stocco, and Natalia Tourn, *Sistema de*

visión para fútbol de robots, Master's thesis, Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay, 2004.

[PZ04] Chris A. C. Parker and Hong Zhang, *Biologically inspired decision making for collective robotic systems*, IEEE/RSJ International Conference on Intelligent Robots and Systems (2004), Dept. of Computing Science, University of Alberta, Alberta, Canada.