Sistemas Lógicos de Múltiples Agentes: Arquitectura e Implementación en Simuladores de Conflictos

Eduardo Alonso, Daniel Kudenko

Department of Computer Science
University of York, York YO10 5DD, UK
{ea,kudenko}@cs.york.ac.uk

Este artículo presenta algunos resultados preliminares en la arquitectura e implementación de un sistema de múltiples agentes basado en lógica en un dominio de simulación de conflictos. En el micronivel, se introducen los mecanismos de planificación y aprendizaje de los agentes individuales. En el macronivel, presentamos una jerarquía de mando y control. Finalmente, se consideran algunos detalles de la implementación.

Palabras clave: Sistemas de múltiples agentes, organización, planificación, aprendizaje.

1. Introducción

Nuestra investigación se basa en la hipótesis de que los agentes necesitan grandes cantidades de conocimiento del dominio en sistemas de múltiples agentes complejos y dinámicos. Una aproximación lógica es, por tanto, un candidato idóneo para diseñar e implementar tales sistemas.

Por otra parte, en escenarios complejos es por lo general imposible para un diseñador preveer todas las situaciones en las que un agente puede encontrarse, de modo que los agentes deben evolucionar y adaptarse al entorno. Nosotros proponemos dos mecanismos de aprendizaje basados en lógica, el aprendizaje basado en explicaciones y la programación lógica inductiva, para dar cuenta de dicho problema. Nuestra hipótesis es que las técnicas lógicas de aprendizaje mejoran el comportamiento de los agentes en entornos complejos debido a que los agentes pueden así incoporar directamente conocimiento del dominio en el proceso de aprendizaje. Para validar tal hipótesis hemos implementado un simulador de conflictos.

En este artículo se discuten la arquitectura e implementación de los agentes individuales y el sistema de múltlipes agentes, prestando especial atención a aspectos relacionados con la organización, la planificación y el aprendizaje.

El resto del artículo está estructurado de la siguiente manera: En la próxima sección, motivamos la simulación de conflictos como el dominio de aplicación de nuestro sistema. La tercera sección describe la arquitectura individual de los agentes. En la cuarta sección, consideramos la estructura organizativa (las reglas de control y de comunicación) del sistema de múltiples agentes. En las secciones 5 y 6, se presentan las técnicas de planificación y aprendizaje. En la sección 7, se discute la planificación de múltiples agentes y el aprendizaje de múltliples agentes en nuestro sistema. Algunos detalles técnicos acerca de su implementación aparecen en la sección 8. Finalmente, comentamos en las dos últimas secciones el trabajo relacionado y las líneas de investigación futura.

2. Simuladores de Conflictos

Los simuladores de conflictos (o juegos de guerra) pueden ser definidos como modelos de confrontaciones militares (Dunnigan, 1992). Estos modelos incluyen un mapa del campo de batalla (formado normalmente por hexágonos), las unidades militares, y un conjunto de reglas de simulación que definen las habilidades de las mismas (movilidad, capacidad de combate, rango de tiro, etc.), los efectos del terreno, etc. (véase, la Figura 1 como ejemplo). La escala de los modelos de simulación varía desde escaramuzas tácticas hasta grandes campañas estratégicas.

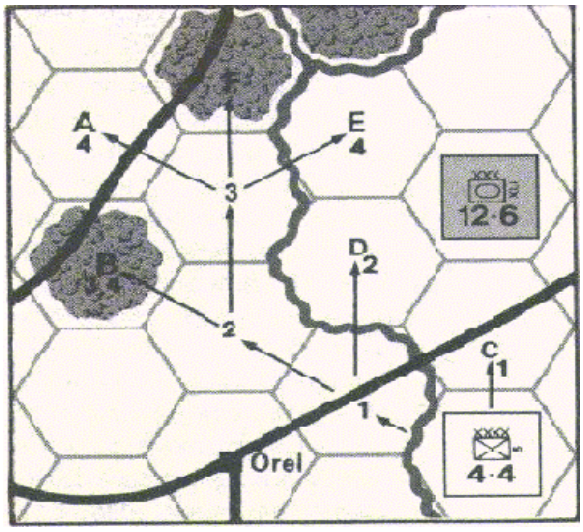

Figura 1. Un simulador de conflictos.

Los simuladores de conflictos han sido ampliamente utilizados por los militares para entrenamiento de tropas y el desarrollo y prueba de estrategias. Como hemos explicado en (Alonso y Kudenko, 2000; Kudenko y Alonso 2000), los simuladores de conflictos son un dominio de aplicación idóneo para evaluar sistemas lógicos de múltiples agentes debido a las siguientes propiedades:

- Aportan una gran de cantidad de conocimiento, disponible en la forma del modelo de simulación.
- La diversidad de los modelos de simulación de conflictos (desde las campañas de Aníbal hasta la guerra de Chechenia) supone un reto a la generalidad y adaptabilidad del sistema.
- La complejidad del modelo puede variarse con facilidad, lo que permite evaluar cómo se comporta el sistema cuando aumenta, por ejemplo, el número de agentes o de las tareas a realizar.

A su vez, las técnicas lógicas de adaptación en sistemas de múltiples agentes resultan interesantes para la creación de oponentes inteligentes por tres razones:

- El conocimiento del dominio es crucial para la conducta inteligente. Los agentes basados en lógica son capaces de utilizar el conocimiento disponible para planificar y aprender.
- Dado que la búsqueda global de una estrategia óptima es normalmente imposible, el proceso de búsqueda tiene que ser distribuido utilizando una approximación basada en múltiples agentes.
- Los sistemas de múltiples agentes son útiles para la implementación de jerarquías de mando, que son la base de las organizaciones militares.

3. Arquitectura de los agentes

La arquitectura de los agentes individuales se ilustra en la Figura 2. En esta sección describimos brevemente los componentes individuales del sistema. En las próximas secciones se puede encontrar información más detallada de los procesos de planificación y aprendizaje.

Bases de conocimiento: Un agente tiene tres bases de conocimiento: una conteniendo información completa de las reglas de simulación, otra con conocimiento inducido sobre la conducta de los oponentes, y una tercera donde se almacena la historia de éxitos y fracasos, ejemplos con que se alimenta el módulo de aprendizaje.

Planificación: Los agentes en nuestro sistema planifican de manera pro-activa, orientada al objetivo. Las distintas fuentes de conocimiento sirven como las bases para el módulo de planificación. Tras ejecutar una acción, el agente verifica si el plan es aún válido. En caso contrario, se replanifica.

Aprendizaje: Técnicas de aprendizaje basado en explicaciones y/o de programación lógica inductiva se usan después de que una acción ha sido ejecutada y una vez que se ha observado sus consecuencias. El resultado del aprendizaje se añade a la base de conocimiento correspondiente: los resultados del aprendizaje basado en explicaciones se integran en la base de conocimientos completa, y los resultados de la programación lógica inductiva en la base de

conocimiento incompleta ya que no hay garantía de corrección para el razonamiento inductivo.

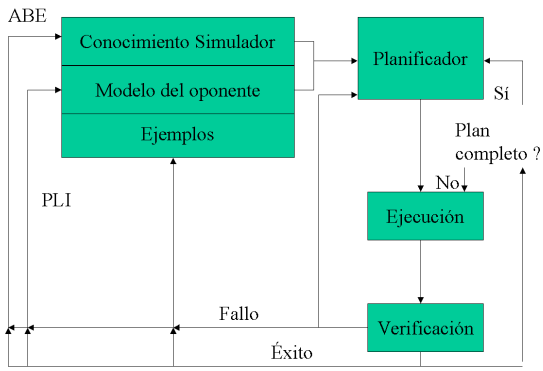

Figura 2. Una arquitectura integrada de planificación y aprendizaje.

Hay dos clases de agentes en nuestro sistema: unidades y comandantes. Una unidad es un agente que esta físicamente presente en el mapa e interctúa directamente con el entorno. Por contraste, los comandantes son agentes ‘’virtuales’’ que sólo actúan cuando mandan órdenes a sus inferiores. En la siguiente sección se da cuenta detallada de la estructura organizativa del sistema.

4. Organización del sistema de múltipes agentes

Hemos elegido una jerarquía de mando y control como una estructura organizativa natural para nuestro simulador de conflictos. En este tipo de estructura, la autoridad a la hora de tomar decisiones esta concentrada en un único solucionador de problemas (o grupo especializado) en cada nivel de la jerarquía. En los niveles superiores de la jerarquía los agentes deben formar y ejecutar planes que consigan objetivos de alto nivel. Para ello, son dotados de conocimiento abstracto de propósito general. Los agentes en la parte inferior de la jerarquía, por el contrario, utilizan conocimineto específico y detallado del dominio para ejecutar acciones concretas. La Figura 3 muestra un ejemplo de este tipo de jerarquías. A cada nivel, sólo un agente considera la información al nivel de detalle que corresponde, es decir, un comandante no tiene que generar planes concretos al nivel de las unidades.

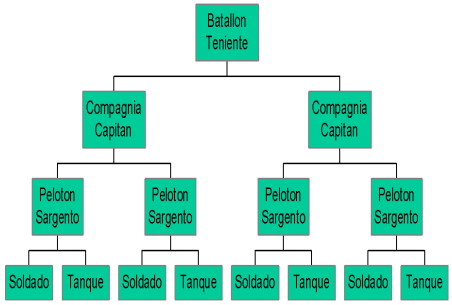

Figura 3. Una jerarquía de mando y control.

Las ventajas derivadas de utilizar este tipo de jerarquías son múltiples, a saber:

- Se ha demostrado que la distribución de la solución de problemas en varios niveles de abstracción reduce el tiempo de computación del proceso de planificación (Korf, 1987; Montgomery y Durfee, 1993).
- En un sistema jerárquico, el módulo de aprendizaje puede ser aplicado a cada espacio del problema. Dado que los espacios abstractos definen teorías y problemas más generales y, por lo tanto, más simples, los agentes aprenden reglas más eficientes (Knoblock et al, 1991).
- La revisión de planes es más sencilla. Si la situación cambia localmente, sólo los planes al nivel de las unidades se ven afectados ---no hay necesidad de cambiar el plan entero. Este no sería el caso si el comandante de la compañía hubiese planificado desde el inicio al nivel más bajo de detalle. Sin embargo, si hay una desviación significativa de las situación esperada, entonces los planes de alto nivel también deben ser revisados (ver sección 5).
- La cobertura, la conectividad, y la capacidad del sistema (Corkill y Lesser, 1983) están aseguradas. Se evitan interacciones negativas distribuyendo las responsabilidades entre diferentes comandantes en diferentes áreas del mapa y manteniendo las unidades lo más independientes posible.
- Los costes de coordinación y comunicación son reducidos ya que la comunicación horizontal (la comunicación entre agentes del mismo nivel) está prohibida y la comunicación vertical es restringida a órdenes y peticiones.

Los diferentes niveles en la jerarquía corresponden con diferentes abstracciones del espacio del problema.

En nuestro simulador, la abstracción se refiere a la diferente granularidad del conocimiento que los agentes tienen acerca del terreno y los demás agentes. La granularidad del conocimiento (es decir, la imagen del mundo) de un agente depende del nivel de la jerarquía en el que está situado.

Abstracción del terreno: Al nivel más bajo de abstracción, las posiciones concretas (coordinadas [X,Y]) corresponden con una característica del terreno (por ejemplo, plano, bosque, montaña, ciudad). Por su parte, los comandantes ignoran las posiciones individuales y las abstraen en unidades más amplias como áreas o subáreas. Por ejemplo, las posiciones cuyas coordinadas cubren del 0 al 30 pertenecen al área nordeste. Aquéllas cuyas coordinadas comprenden del 0 al 10 serían el subárea noroeste del área nordeste.

Las características del terreno en las áreas y en las subáreas dependen de la característica del terreno dominante en el correspondiente conjunto de posiciones. Por ejemplo, una subárea es montañosa si más de un porcentaje predefinido de posiciones en esa subárea son montañas. Las subáreas (y las áreas) son consideradas ‘’mixtas’’ si incluyen más de una característica dominante.

Abstracción de agentes: Los agentes están organizados en grupos. Los comandantes no consideran tales grupos como un conjunto de agentes individuales, sino como un ‘’super-agente’’. Las propiedades de tal super-agente son la media de las propiedades de los agentes constituyentes.

Como se ha mencionado anteriormente, la comunicación en nuestra jerarquía se reduce al paso de mensajes vertical. Un agente se comunica con sus superiores inmediatos mandando peticiones y confirmaciones, y con sus subordinados mediante órdenes. No se contempla otro tipo de comunicación.

5. Planificación

Hemos elegido STRIPS (Fikes y Nilsson, 1971) como el formalismo ideal para la representación de acciones. En STRIPS, una acción es representada mediante tres listas:

- Precondiciones: una lista de fórmulas atómicas que deben ser verdaderas para que ocurra la acción;
- Lista de borrar: una lista de las relaciones primitivas que dejan de ser verdaderas cuando se ejecuta la accón;
- Lista de añadir: lista de relaciones primitivas hechas verdaderas por la acción.

Aun cuando STRIPS es restringido en su expresividad (sólo se permiten fórmulas atómicas en las condiciones), no creemos que necesitemos formalismos más expresivos en nuestro dominio de aplicación. Además, los algoritmos de planificación más eficientes operan sobre representaciones STRIPS.

Hemos utilizado una variante del algoritmo Graphplan (Blum y Furst, 1997) para el módulo de planificación de nuestros agentes. Los sistemas Graphplan son muy eficientes (Weld, 1999), a la vez que mantienen la corrección y la completud. Una ventaja adicional es su extensibilidad, por ejmplo, para la planificación condicional (Anderson et al., 1998), contingente (Weld et al., 1998), temporal (Smith y Weld, 1999), y probabililista (Blum y Langford, 1999).

Dado que las acciones tienen un coste asociado en nuestro modelo de simulación, hemos modificado el algoritmo básico de Graphplan de manera que los agentes puedan computar el plan de coste mínimo.

En nuestro sistema, la planificación puede ser entendida como un proceso de refinamiento de objetivos. El proceso comienza cuando un comandante en lo alto de la jerarquía adopta un objetivo abstracto y forma un plan (un CONcept PLAN en terminología militar) para alcanzarlo. Este plan consiste en órdenes que sus subordinados inmediatos adoptan como objetivos. El proceso se repite hasta el nivel de las unidades, donde los agentes forman y ejecutan planes detallados, (Operational PLANs), es decir, acciones, para conseguir sus objetivos concretos.

6. Aprendizaje

Uno de los mayores problemas a la hora de implementar sistemas de agentes es la dificultad del diseñador para preveer todas las situaciones que un agente puede encontrar ni, por lo tanto, especificar la conducta óptima del agente por adelantado. Esto es especialmente grave en entornos de múltliples agentes donde la fuente de incertidumbre no es sólo la Naturaleza sino la presencia de otros agentes con creencias, objetivos e intenciones diferentes e incluso antagónicas. Es por lo tanto ampliamente reconocido en la comunidad de agentes que una de las características más importantes de los sistemas inteligentes es su capacicidad para adaptarse y aprender (por ejemplo, Russell y Norvig, 1995). Este campo de aprendizaje en agentes ha sido especialmente fructífero en las últimas dos décadas, en particular en lo que se refiere al aprendizaje en sistemas de múltiples agentes (ver Sen, 1998; Weiss, 1997).

Sin embargo, debemos hacer notar que mientras se suele utilizar un lenguaje lógico para especificar la arquitectura de los agentes, la mayoría de los algoritmos de aprendizaje que se han aplicado en agentes no están basados en lógica, sino que se utilizan otras técnicas como el aprendizaje por refuerzo o las redes neuronales (Kaelbling et al., 1996). Aun cuando estas técnicas han sido exitosas en dominios restringidos, desnudan a los agentes de la habilidad de adaptarse de una manera dependiente del dominio. Nosotros creemos que tal capacidad es crucial en dominios complejos donde el conocimiento del dominio tiene un gran impacto en la calidad de las decisiones que toman los agentes. Como una solución a este problema, proponemos la aplicación de técnicas lógicas de aprendizaje.

El aprendizaje en nuestro sistema puede ser dividido en dos categorías:

- Aprendizaje de aceleración, es decir, la optimización del conocimiento existente para el proceso de planificación de manera que se mejoren la capacidad de solución de problemas del sistema;
- Adquisición de nuevo conocimiento, en concreto, conocimiento acerca de la conducta del oponente.

Utilizamos técnicas de aprendizaje basado en explicaciones (ABE) y programación lógica inductiva (PLI) respectivamente para tratar con estas tareas de aprendizaje.

6.1. Aprendizaje basado en explicaciones

En nuestro sistema, los agentes están dotados de conocimiento perfecto acerca de las reglas de simulación. No obstante, sabemos que computar una solución (es decir, un plan) directamente a partir de principios básicos es un problema extremadamente complejo. ABE añade macro-operadores útiles a la base de conocimiento y, por lo tanto, acelera el proceso de búsqueda de una solución óptima.

ABE ha sido utilizado ampliamente en Inteligencia Artificial para acelerar la actuación de los planificadores (como en PRODIGY, Carbonell et al., 1990). Obviamente, los solucionadores de problemas, cuando se les presenta con el mismo problema repetidas veces, no deberían solucionarlo de la misma manera en la misma duración de tiempo. Por el contrario, parece sensato usar conocimiento general para analizar o explicar cada ejemplo de la solución del problema para optimizar la actuación futura del planificador. Este tipo de aprendizaje no es sólo una manera de hacer que un programa funcione más rápidamente, sino que también es un modo de producir cambios cualitativos en la actuación del solucionador de problemas.

En dos palabras, ABE extrae reglas generales de ejemplos particulares generando y generalizando una explicación acerca del éxito o fracaso del sistema. Ello proporciona un método deductivo (no estadístico) para convertir conocimiento general en experiencia práctica. Las reglas así aprendidas permiten al planificador tomar la decisión correcta si se le presenta una situación similar durante la solución de problemas subsecuentes.

Ilustramos cómo funciona el ABE con un ejemplo simple. Asumamos que un comandante ha enviado sus unidades a eliminar una unidad enemiga y que tales unidades han elegido sus posiciones de manera que el enemigo está rodeado. Asumamos que el ataque es exitoso y que eliminan al enemigo. Para explicar este éxito, el comandante utiliza las siguientes reglas que no fueron usadas anteriormente en la formación del plan original (aunque habían estado siempre en su base de conocimiento):

Hecho: Cada unidad tiene una zona de control, los seis hexágonos adyacentes a su posición actual.

Regla 1: Una unidad que entra en una zona de control enemiga debe terminar su fase de movimiento.

Regla 2: Una unidad que se retira a una zona de control enemiga es eliminada.

El comandante puede explicar ahora el éxito de sus unidades. Alcanzaron su objetivo porque rodearon al enemigo, cortándole todas sus posibles rutas de retirada. Esta regla es entonces generallizada:

Regla ABE: Si una unidad es rodeada, entonces es eliminada.

La regla así deducida es entonces añadida a la base de conocimiento y usada directamente para elaborar planes más precisos más rápidamente.

Por supuesto, no se aprende nada nuevo. En teoría, los agentes podrían haber deducido tal regla de la base de datos original. Sin embargo, ello hubiese sido computacionalmente prohibitivo.

6.2. Programación lógica inductiva

Una cosa es deducir de reglas generales el orden y asignación de las subtareas que deberían, en teoría, llevar un ejército a la victoria. Otra cosa es aprender de ejemplos (y conocimiento previo) si las condiciones para que tales subtareas sean ejecutadas

exitosamente se dan en realidad. En contraste con los métodos ABE, PLI computa una hipótesis basada no sólo en las reglas de simulación que se conocen de antemano sino también en circunstancias que son inicialmente desconocidas, como las estrategias del oponente. En general, confiar en exclusivo en las reglas generadas con ABE puede resultar poco práctico en dominos reales donde los agentes trabajan con conocimiento incompleto, de modo que PLI es importante para la eficacia general del sistema.

Los métodos PLI computan una hipótesis inductiva con la ayuda de ejemplos positivos y negativos y conocimiento del dominio. Como se ilustra en la figura 4., los agentes coleccionan ejemplos basados en las acciones y planes que han ejecutado y sus resultados. Ello, junto con la base de conocimiento del dominio y un predicado a aprender (por ejemplo, ‘’éxito’’ o ‘’fracaso’’) forma la base del proceso de aprendizaje inductivo que computa una hipótesis (es decir, una definición del predicado objeto) que puede ser utilizada en los próximos procesos de planificación. Los predicados a aprender son especificados por el diseñador del sistema.

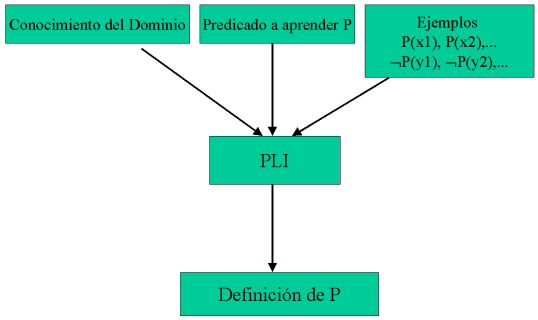

Figura 4. Programación lógica inductiva.

Cuando un número de ejemplos han sido clasificados incorrectamente (es decir, el agente comete un cierto número de errores en sus predicciones acerca del resultado de sus acciones) una nueva hipótesis es computada en base a la base de ejemplos extendida. Si la hipótesis no cubre los nuevos ejemplos, entonces es generalizada. Si tales ejemplos la contradicen, entonces la hipótesis debe ser refinada.

Ilustramos el uso de PLI en nuestro sistema con un ejemplo. Un agente podría deducir la siguiente regla de su base de conocimientos original (tomando en consideración, por ejemplo, la capacidad de combate de las unidades):

Regla: Si un tanque ataca una unidad de infantería situada en un puente, entonces es exitoso.

La unidad usa tal regla para computar y ejecutar el plan apropiado (moverse a la posición adyacente, etc.). Supongamos, sin embargo, que el plan falla en repetidas ocasiones (por ejemplo, la unidad de infantería se retira y vuela el puente). Estos nuevos ejemplos contradicen la regla anterior. PLI encontrará entonces la siguiente hipótesis:

Hipótesis: Si un tanque ataca una unidad de infantería situada en un puente, entonces no es exitoso.

La unidad rechazará por tanto un ataque frontal e intentará un plan alternativo (por ejemplo, moviendo en primer lugar algunas unidades ‘’amigas’’ al otro lado del puente).

7. Planificación de múltiples agentes y aprendizaje de múltiples agentes

En términos generales, la planificación en nuestro sistema puede ser vista como planificación centralizada para planes distribuidos (Fischer, 2000). Por una parte, los agentes no trabajan como un equipo en el sentido descrito en (Cohen y Levesque, 1991). En sociedades complejas, tales como un ejército, la cantidad de comunicación necesaria para que los agentes alcanzasen un acuerdo acerca de un plan conjunto sobrecargaría el sistema. Esta es la razón por la que la responsabilidad para tomar decisiones se delega en coordinadores, los comandantes. Como consecuencia, los agentes sólo actúan cuando reciben órdenes.

Por otra parte, los agentes trabajan lo más independientemente posible. Una vez que han recibido una orden, son libres de usar su conocimiento del dominio para formar y ejecutar sus propiso planes para satisfacer el objetivo descrito en dicha orden.

Los comandantes, por tanto, imponen objetivos, no planes. Las responsabilidades de planificación son delegadas en los diferentes niveles de la jerarquía de acuerdo con las habilidades y el conocimiento de los agentes y determinados parámetros geográficos.

En lo que se refiere al aprendizaje, cada agente aprende individualmente cómo mejorar sus propias habilidades. Las unidades aprenden, pues, cómo moverse y atacar con éxito al enemigo. Los comandantes, cuyas habilidades y repertorio de acción están limitados a dar órdenes, aprenden cómo

mantener a las secciones bajo su mando eficientemente coordinadas. La coordinación es, pues, un problema de mando. Consecuentemente, las unidades no aprenden cómo cooperar. Los comandantes lo hacen.

Se podría argumentar que éste es un tipo de aprendizaje ‘’aislado’’ (Stone y Veloso, 1998), ya que los agentes aprenden por separado. Por otra parte, lo que una unidad aprende afecta los procesos de planificación y de aprendizaje del resto de los agentes. Si una unidad aprende cómo conseguir su objetivo eficientemente, su comandante le asignará la misma tarea en una operación similar. Si la unidad aprende que no puede satisfacer su objetivo, entonces será reemplazada o se cambiará el plan original del comandante.

8. Implementación

La implementación de nuestro sistema consiste de dos partes, como se ilustra en la Figura 5. El interface del usuario (input y visualización del mapa, de los agentes, del movimiento, de los planes, de la comunicación y de la jerarquía) se ha implementado en JAVA. El servidor del simulador y los agentes se implementaron en PROGOL (Muggleton, 1995), un intérprete de Prolog aumentado con un algoritmo de programación lógica inductiva.

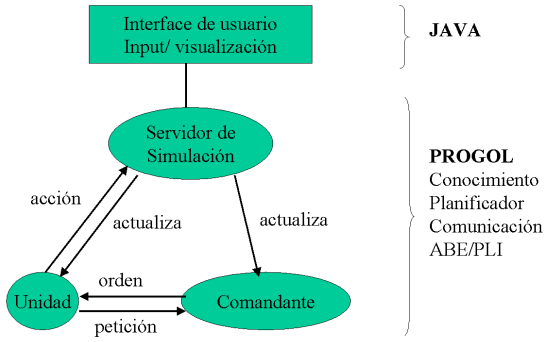

Figura 5. Diagrama de implementación.

La simulación en sí es sincronizada y basada en turnos. Todos los agentes de un ejército envían simultáneamente las acciones que deben ejecutarse al servidor, el cual actualiza el estado del mundo y manda los cambios de vuelta a los agentes. Acto seguido, los agentes del otro ejército ejecutan sus acciones. El simulador sigue esta rutina (mover-combatir-repostar-recargar) sucesivamente hasta que uno de los ejércitos gana la batalla o se alcanza un número predeterminado de fases.

La comunicación se implementa en MAS-PROGOL, una versión de Progol de múltiples agentes que se está desarrollando en el Artificial Intelligence Group en la University of York. Este programa permite a un agente mandar un objetivo PROGOL al resto de los agentes y recibir de vuelta variables ligadas o valores de verdad. El protocolo de comunicación que proporciona PROGOL puede ser visto como un subconjunto del protocolo KQML (Finin et al., 1997). Dado que nuestro sistema solo contiene agentes homogéneos que se comunican de una manera bastante restringida, no es necesario el uso de KQML.

9. Trabajo relacionado

Nuestro trabajo esta íntimamente relacionado con la investigación en otras áreas:

- Sistemas de múltiples agentes: nuestro trabajo se relaciona con los sistemas colaborativos basados en modelos tales como GRATE* (Jennings, 1995), COLLAGEN (Rich y Sidner, 1998) y STEAM (Tambe, 1997). Sin embargo, las habilidades que los agentes necesitan para colaborar en tales sistemas son diseñadas desde el comienzo, de modo que el aprendizaje no es una parte fundamental en la arquitectura de los agentes.
- Aprendizaje automático: ABE ha sido utilizado para mejorar la eficacia de los planificadores (por ejemplo, en SOAR (Laird et al., 1987) y PRODIGY (Carbonell et al., 1990)), pero sólo en dominios de agente único. Nuestro trabajo está también relacionado con la investigación en la integración de abstracción y ABE (Knoblock et al., 1991). También nos beneficiamos del trabajo reciente de Tara A. Estlin sobre cómo aplicar aprendizaje de múltiples estrategias para mejorar la eficiencia y la calidad de los planificadores (Estlin, 1998).
- Aprendizaje de múltiples agentes: la mayoría de las implementaciones de técnicas de aprendizaje en sistemas de múltiples agentes incluyen mecanismos de aprendizaje por refuerzo o redes neuronales (Weiss, 1999). Sólo Sugawara y Lesser (Sugawara y Lesser, 1998) han utilizado técnicas ABE en escenarios de múltiples agentes. Sin embargo, ellos no diferencian entre la coordinación como resultado del trabajo en equipo y la coordinación en general. Consecuentemente, no estudian cómo los compromisos y las responsabilidades adquiridas

afectan la formación de los protocolos de coordinación.

- Simulación de conflictos: en el pasado, han habido varios intentos de implementar fuerzas computerizadas para la investigación militar (Calpin, 1998; Tambe, 1997). No obstante, al carecer de técnicas de múltiples agentes, dichos modelos sufrían de falta de coordinación. Últimamente, se están empleando tales técnicas (Baxter, 1998), pero aún no se han estudiado en profundidad procedimientos lógicos para el aprendizaje en dominios militares.

10. Conclusiones y trabajo futuro

Hemos presentado algunos resultados sobre la arquitectura e implementación de un sistema de múltiples agentes basado en lógica y su aplicación en un dominio de simulación de conflictos. Al nivel de los agentes, los mecanismos de planificación y aprendizaje con que son dotados han sido introducidos. Al nivel del sistema de múltiples agentes, se ha presentado una jerarquía de mando y control. Finalmente, se han dado algunos detalles sobre la implementación.

El modelo debe ahora ser evaluado usando los siguientes criterios:

- Eficacia del sistema de múltiples agentes, analizando estadísticas de éxitos y fracasos contra humanos y sistemas reactivos.
- Habilidad para aprender y adaptarse al entorno, probando el sistema contra diferentes estrategias y en distintos escenarios.
- La eficiencia que se gana con conocimiento del dominio, comparando la actuación de diferentes versiones del sistema con diferentes tipos de conocimiento del dominio.

El trabajo futuro deberá incluir también la extensión del sistema a escenarios más realistas: se incorporarán restricciones climáticas y temporales, así como más tipos de unidades (cazas, anfibios) y de acciones (transporte, etc.). A medida que el sistema crezca, los agentes deberán evolucionar en dominios cada vez más inciertos. Para solucionar este problema, se considerará la introducción de acciones sensitivas (Weld et al., 1998), de planificación condicional (Anderson et al., 1998) y de planificación probabilística (Blum y Langford, 1999).

Referencias

Alonso, E. & Kudenko, D. (2000). Logic-based multi-agent systems for conflict simulation. In Proceedings of *The Third Workshop of the UK Special Interest Group on Multi-Agent Systems*. St. Catherine's College, Oxford.

Anderson, C., Smith, D. & Weld, D. (1998). Conditional effects in Graphplan. In Proceedings of *The Fourth Conference on AI Planning Systems*. Pittsburgh, PA.

Baxter, J.W. (1998). A model for co-ordination and co-operation between CGF agents. In Proceedings of The Eight *Conference on Computer Generated Forces and Behavioral Representation*. Orlando, Florida.

Blum, A.L. & Furst, M.L. (1997). Fast planning through planning graph analysis. *Artificial Intelligence* **90**: 281-300.

Blum, A.L. & Langford, J. (1999). Probabilistic Planning in the Graphplan Framework. In *The Proceedings of ECP'99*. Berlin: Springer-Verlag.

Calpin, J.A. (1998). Performance of the DARPA command forces program in the STOW97 ACTD. In Proceedings of *The Eight Conference on Computer Generated Forces and Behavioral Representation*. Orlando, Florida.

Carbonell, J., Knoblock, C. & Minton, S. (1990). PRODIGY: An integrated architecture for planning and learning. In van Lehn, K. (ed.) *Architectures for Intelligence*. Lawrence Erlbaum Associates: Hillsdale, NJ.

Cohen, P. R. & Levesque, H.J. (1991). Teamwork. *Nous* **25**, 487-512.

Corkill, D.D. & Lesser, V.R. (1983). The use of meta-level control for coordination in a distributed problem solving network. In Proceedings of *IJCAI-93*, 748-756.

Dunnigan, J.F. (1992). *The Complete Wargames Handbook*. Morrow: New York.

Estlin, T. (1998). *Using Multi-Strategy Learnig to Improve Planning Efficiency and Quality*. PhD Thesis, University of Texas at Austin.

Fikes, R.E., & Nilsson, N.J. (1971). Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* **2**(3-4): 189-208.

Finin, T., Labrou, Y. & Mayfield, J. (1997). KQML as an agent communication language. In Bradshaw, J. (ed.) *Software Agents*, 291-316. MIT Press: Cambridge, MA.

Fisher, K. (2000). Problem Solving and Planning. In Klusch, M., Fischer, K., & Luck, M. (eds.) Working Notes of *The Second European Agent Systems Summer School*, EASSS- 2000. Saarbrucken, Germany.

Jennings, N.R. (1995). Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence* **75**: 195-240.

Kaelbling, L.P., Littman, M.L. & Moore, A.W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* **4**: 237-285.

Knoblock, C.A., Minton, S. & Etzioni, O. (1991). Integrating Abstraction and Explanation-Based Learning in PRODIGY. In Proceedings *of The Ninth National Conference on Artificial Intelligence AAAI-91*, 93-102. Menlo Park, CA: AAAI Press.

Korf, R.E. (1987). Planning as search: A qualitatuve approach. *Artificial Intelligence* **33**(1): 65-88.

Kudenko, D. & Alonso, E. (2000). Machine learning techniques for logic based multi-agent systems. In Proceedings of *The First Goddard Workshop on Formal Approaches to Agent-based Systems*. Greenbelt, MD.

Laird, J.E., Newell, A. & Rosenbloom, P.S. (1987). SOAR: an architecture for general intelligence. *Artificial Intelligence* **33**(1): 1-64.

Montgomery, T. A. & Durfee, E. H. (1993). Search reduction in hierarchical distributed problem solving. *Group Decision and Negotiation* **2**: 301-317.

Muggleton, S. (1995). Inverse entailment and progol. *New Generation Computing Journal*, **13**: 245-286.

Rich, C. & Sidner, C. (1998). COLLAGEN: When agents collaborate with people. In Huhns, M. N. & Singh, M.P. (eds.) *Readings in Agents*, 117-124. Morgan Kaufmann: San Francisco, CA.

Russell, S. & Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice Hall: Upper Saddle River, NJ.

Sen, S. (1998). Special issue on evolution and learning in multi-agent systems. *International Journal of Human-Computer Studies* **48**.

Smith, D. & Weld, D. (1999). Temporal Planning with Mutual Exclusion. In *The Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, Stockholm, Sweden.

Stone, P. & Veloso, M. (1998). Towards Collaborative and Adversarial Learning: A Case Study in Robotic Soccer. *International Journal of Human Computer Studies* **48**.

Sugawara, T. & Lesser, V. (1998). Learning to improve coordinated actions in cooperative distributed problem solving environments. *Machine Learning* **33**: 129-153.

Tambe, M. (1997). Towards flexible teamwork. *Journal of Artificial Intelligence Research* **7**: 83-124.

Weiss, G. (1997).Distributed Artificial Intelligence meets Machine Learning. Proceedings of *The European Conference on Artificial Intelligence ECAI-96 Workshop*. Berlin: Springer.

Weiss, G. (1999). *Multiagent Systems: A modern approach to Distributed Artificial Intelligence*. The MIT Press: Cambridge, MA.

Weld, D., Anderson, C. & Smith D. (1998). Extending Graphplan to handle uncertainty and sensing actions. In Proceedings of *The Sixteenth National Conference on Artificial Intelligence*. Madison, WI.

Weld, D. (1999). Recent advances in AI planning. To appear in *AI Magazine*.