UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

MÁSTER EN INGENIERÍA EN INFORMÁTICA Y DE TELECOMUNICACIONES

TRABAJO FIN DE MÁSTER

ESTUDIO DE LA INFLUENCIA DE TOPOLOGÍAS DE COMUNICACIÓN EN SISTEMAS MULTIAGENTE BIOINSPIRADOS

Autor: D. Antonio González Pardo
Tutor: Prof. D. David Camacho Fernández

Febrero de 2011

Índice

Índice de figuras

Resumen

El estudio de las redes de comunicaciones ha sufrido un importante crecimiento en las últimas décadas en diferentes áreas de investigación como Inteligencia Artificial, Biología, Medicina y Psicología entre otras. Gran parte de esos trabajos estudian problemas de sobrecarga en las comunicaciones, la conectividad dentro de las redes, o protocolos de comunicación. Desde el punto de vista de la Inteligencia Artificial o Sistemas Multi-Agente, estas redes de comunicaciones están constituidas por un conjunto de nodos, o agentes, que pueden ser tanto software como hardware. Los problemas de saturación en redes de comunicaciones pueden aparecer en dos niveles diferentes: a nivel de nodos y a nivel de red. La saturación a nivel de agentes se produce cuando los agentes reciben más mensajes que los que ellos son capaces de procesar. Por otro lado, cuando los agentes envían más mensajes de los soportados por la red se produce la saturación a nivel de red. Este trabajo fin de máster tiene como objetivo la reducción de la sobrecarga en Sistemas Multi-Agente. Para reducir la sobrecarga a nivel de agente, estos están dotados de un sistema de discriminación de información bio-inspirado que les permite analizar los mensajes recibidos dependiendo del emisor de los mismos. Para reducir la sobrecarga en la red, el trabajo estudia la topología de comunicación óptima para la cual el número de mensajes enviados es el mínimo. Esta topología óptima se caracteriza por la utilización de una probabilidad de redirección de los enlaces, de manera que se introducen atajos en la red inicial. Una vez que todos los enlaces han sido analizados usando esta probabilidad de redirección, se inicia la ejecución del Sistema Multi-Agente y la topología no vuelve a cambiar. Los resultados muestran una importante reducción en el número de mensajes enviados y además, se ha podido determinar los valores de los parámetros en los cuales los sistemas logran su mejor rendimiento. Con estos valores fijados, se han analizado varios sistemas compuestos por un mayor número de agentes y los resultados obtenidos de estos nuevos experimentos muestran que los valores fijados de los parámetros son extrapolables en grandes sistemas sin influir significativamente en el rendimiento de estos.

Abstract

The study of Network communication have grown in the last decades in research fields such as Artificial Intelligence, Biology, Medicine or Psychology amongst others. Some of those works studies the overhead problems, connectivity or communication protocols. When these problems are restricted to Artificial Intelligence or Multi-Agent Systems, the networks are composed by a set of nodes, or agents, that can be software or hardware. Overhead problems in network communications can appear in two different ways: overhead problem in nodes or agents, and overhead in the network. Overhead problems in nodes are produced when agents received more messages than they are able to analyzed. On the other hand, overhead problems in network communications are produced when the number of sent messages in the network exceeds the maximum number of messages that the network is able to handle. The aim of this Masterś thesis is the reduction overhead in Multi-Agent Systems. In order to reduce the agent's overhead, agents have a bio-inspired discrimination process that allows them to analyze messages depending on the sender. To reduce network overhead, this work studies the sub-optimal communication topology for which the minimum number of sent messages is needed to solve the problem for which the system was designed. This sub-optimal topology is characterized by the use of a redirection probability of the network links, in such a way that shortcuts are created in the network. Once links are redirected, or not, according to a specific redirection probability, the topology is fixed and remains unchanged during the execution of the Multi-Agent System. The experimental results show an important improvement in terms of number of sent messages. Furthermore, a range value for the redirection probability in which the system provides the best performance has been defined. With this range value, larger multi-agent systems have been tested and the results of these new set of experiments show that the range value selected is applicable in larger multi-agent systems without deteriorating the performance of the system.

1. Introducción

Durante la última década, el interés en los Sistemas Multiagente (MAS) ha aumentado considerablemente. Este crecimiento se debe a que un gran número de problemas se pueden resolver creando sociedades de agentes que se comunican entre sí para resolver el conjunto de tareas, u objetivos, para los cuales fueron creados. Sin embargo, cuando el número de agentes se incrementa y la conectividad entre ellos es intensa, el sistema de comunicaciones se puede saturar debido a la enorme cantidad de mensajes que viajan por el sistema.

Existen numerosas plataformas sobre las que implementar un sistema multi-agente, como pueden ser Jade [1] o Grassoppher [2], sin embargo el problema de la saturación de mensajes es independiente de la arquitectura utilizada [3]. En la literatura se han seguido varios enfoques a la hora de solucionar este problema:

- **Utilizando agentes móviles**. Este tipo de agentes se denominan móviles porque viajan por la red y se ejecutan en los nodos de dicha red. La idea es ejecutar en el mismo nodo aquellos agentes con alta comunicación entre ellos para que estos mensajes no saturen la red, ya que las comunicaciones dentro del mismo nodo no afectan a la sobrecarga del sistema, [4].
- **Utilizando agentes BDI**. Los agentes BDI (*Belief, Desire and Intention*) están dotados de un sistema de razonamiento, o de un proceso de inferencia, que les permite determinar los mensajes que contienen información importante dependiendo de un registro histórico, [5, 6].

Este trabajo fin de máster tiene un enfoque diferente a las dos ideas anteriormente expuestas, ya que los agentes utilizados serán agentes *SWARM* bio-inspirados. Los agentes de tipo SWARM [7] tienen un comportamiento muy básico ya que no poseen reglas de inferencia, ni módulos de razonamiento, que permitan al agente cambiar de manera dinámica su comportamiento. Sin embargo, los agentes utilizados en este trabajo poseen un sistema de discriminación de mensajes basado en el comportamiento de células vivas [8, 9]. Este proceso de discriminación permite a los agentes analizar, o descartar, mensajes dependiendo de la identidad del emisor. De manera que los agentes ahorran tiempo de procesamiento de los mensajes recibidos ya que sólo analizarán aquellos mensajes que les interesen para solucionar el problema modelado.

Otra manera de evitar la saturación de las comunicaciones puede consistir en la optimización de la red. La idea principal consiste en la construcción de topologías que minimicen el envío de mensajes en la red, de manera que los agentes envíen mensajes al menor número de agentes y que se reduzca el coste computacional de resolución del problema.

El objetivo de este trabajo fin de máster es el estudio de los diferentes parámetros que influyen en el rendimiento de un sistema multiagente bio-inspirado. Se pretende realizar un estudio de la influencia de dichos parámetros para generar sistemas multiagentes que se comuniquen de una manera sub-óptima [10, 11].

Este documento se estructura de la siguiente manera. En la Sección 2, se describen los conceptos básicos necesarios para el correcto entendimiento de este trabajo fin de máster. Estos conceptos básicos quedan englobados en dos áreas principales que son los Sistemas Multiagente y la Neurociencia. Después de esto, se describe el sistema multiagente bio-insipirado que se ha desarrollado en la Sección 4. Durante la Sección 5 se describen las diferentes pruebas realizadas, se presentan los resultados obtenidos y se realiza un análisis de los mismos. En la Sección 6, se realiza un resumen general de todo el Trabajo Fin de Máster presentando los problemas que se estudian en este trabajo y describiendo y analizando la plataforma propuesta. La Sección 7 presenta las aportaciones científicas que se han conseguido con este trabajo fin de máster y que se anexan en el presente documento. A parte del trabajo presentado en este documento, se han realizado algunas investigaciones en otras áreas diferentes. De estos trabajos, aquellos que ya han sido publicados son citados en la Sección 8, mientras que los trabajos que se han enviado y están pendientes de notificar se presentan en la Sección 9.

2. Estado del arte

En esta sección se presentan los conceptos básicos de las dos ramas que sustentan este Trabajo Fin de Máster. La primera rama esta formada por los sistemas multiagente, que quedan descritos en la sección 2.1. Los sistemas multiagente son muy utilizados en diferentes áreas de investigación debido a que permiten solucionar problemas globales usando la información local de cada uno de los agentes. Con el uso de estos sistemas se han observado varios problemas que afectan al rendimiento de la plataforma. Uno de estos problemas es la sobrecarga del sistema.

En este trabajo, para solucionar parte de la sobrecarga del sistema se ha dotado a los agentes de un sistema de discriminación de información bio-inspirado. Este sistema de discriminación tiene su inspiración en el comportamiento de las células vivas, las cuales discriminan las entradas en función del emisor de las mismas. Una explicación más amplia de este sistema de discriminación, así como los conceptos biológicos necesarios para su correcto entendimiento, es proporcionada por la sección 2.2.

2.1. Introducción a los Sistemas Multiagente

A continuación se realiza una breve descripción de los Sistemas Multiagente, SMA o MAS del inglés *Multi-Agent Systems*. Esta descripción comienza con la definición de *agente* para después introducir las propiedades de un *Sistema Multiagente*, por último se describe el problema de la sobrecarga en estos sistemas ya que este problema es el que se pretende reducir en este Trabajo Fin de Máster.

La definición de agente es una tarea difícil ya que no existe una definición formal de este concepto que sea totalmente aceptada. Las diferentes definiciones que se pueden encontrar dependen de la perspectiva del área de investigación que realiza la definición. Por eso se van a mostrar algunas definiciones del término *agente* para después intentar resumir todas ellas en una definición común.

- Según el Diccionario de la Real Academia Española [12], se define agente como:

  > *"Agente: Del lat. agens, -entis, part. act. de agere, hacer".* 1. adj. Que obra o tiene virtud de obrar. 2. m. Persona o cosa que produce un efecto. 5. Persona que obra con poder de otra".

- Según el Object Management Group [13]:

  > Un agente es un programa de ordenador que actúa autónomamente en nombre de una persona u organización.

- Según Russell y Norving en su libro *Artificial Intelligence: a Modern Approach* [14]:

  > Un agente puede verse como aquello que percibe su entorno a través de sensores y que actúa sobre mediante efectores.

Existen diversas características que pueden utilizarse para definir, de manera general, las características básicas de un agente, [15, 16, 17]:

1. **Situación**: localización en un determinado entorno.
2. **Flexibilidad**: sus acciones no están prefijadas.
3. **Autonomía**: puede actuar sin intervención directa del hombre o de otro programa. Ejerce control sobre sus propias acciones.
4. **Aprendizaje adaptativo**: cambia su comportamiento basándose en la experiencia previa. Puede aprender y adaptarse al entorno y al usuario.
5. **Persistencia**: puede ser almacenado y recuperado.
6. **Colaboración/Actividad social**: es capaz de colaborar y compartir información con otros agentes.
7. **Comunicativo**: capaz de comunicarse con personas y otros agentes con lenguajes semejantes a lenguajes hablados humanos.
8. **Proactivo/Reactivo**: no sólo es capaz de responder a su entorno sino que esta orientado a objetivos.

Existen algunas clasificaciones de agentes que definen algunas características de los mismos. Teniendo en cuenta la localización, o situación del agente, podemos hablar de agentes **estáticos** o **móviles**. Los agentes estáticos son aquellos que se ejecutan única y exclusivamente en una máquina, mientras que los agentes móviles ejecutan partes de su código en diferentes máquinas. El objetivo de los agentes móviles es el tratamiento de información local, de manera que en lugar de enviar los datos a la máquina donde se ejecuta el agente, es el agente el que se traslada al equipo donde están almacenados los datos. Este tipo de agentes no es muy utilizado en la actualidad debido a los problemas de seguridad que pueden generar [18].

Si se tiene en cuenta el comportamiento del agente, se habla de agentes de tipo SWARM [7] o agentes de tipo BDI [19]. Los agentes de tipo SWARM son agentes con muy poca autonomía y con un comportamiento muy básico. Son agentes ligeros lo que permite realizar simulaciones utilizando un gran número de agentes. Los agentes BDI reciben su nombre del inglés *Beliefs-Desires-Intentions* y como su propio nombre indica, estos agentes tienen un conjunto de creencias y un conjunto de intenciones, que utilizarán para alcanzar unos deseos u objetivos. Estos agentes poseen un proceso de inferencia por el cual a medida que avanza su ejecución el conjunto de creencias o intenciones se modifica. Esto hace que el comportamiento de cada agente se convierta en algo dinámico que cambia a lo largo de su ejecución. Estos agentes suelen ocupar mucha memoria en los equipos, debido a los procesos de inferencia que llevan asociados, y por tanto es difícil la simulación de sistemas formados por un gran número de estos agentes.

Cuando un sistema complejo se enfrenta a la posibilidad de disponer de un conjunto de agentes que pueden desarrollar diferentes tareas, deben de caracterizarse y definirse dos aspectos básicos de este tipo de sistemas. En primer lugar, el comportamiento local de cada agente, sus roles, funciones, etc... Y en segundo lugar, las diferentes características globales, o de grupo, que permiten coordinar a los diferentes agentes dentro de la sociedad que conforman. Este tipo de sistemas complejos formados por varios agentes se denominan Sistemas Multiagente (SMA), pueden definirse como:

> Sistemas computacionales que tratan sobre la coordinación inteligente entre una colección de *agentes* autónomos, y sobre cómo pueden coordinar sus conocimientos, metas, propiedades y planes para tomar una decisión o resolver un problema [20]

Los SMA presentan una serie de propiedades que caracterizan su definición e implementación como son: la descripción de competencias, la modelización del conocimiento, la comunicación, el comportamiento y los puntos de interacción.

- La descripción de competencias consiste en la definición del problema que se pretende resolver como un conjunto de tareas, subtareas y relaciones, de manera que se determina cómo resolver el problema, cómo distribuirlo entre los diferentes agentes y cómo los agentes deben interaccionar entre ellos.

- Generalmente, los SMA solucionan un problema global haciendo uso de información local almacenada en cada uno de los agentes. Por ello, es necesaria la identificación y modelización de la información que va a conocer cada agente.

- Otro aspecto importante es la definición del lenguaje que utilizarán los agentes para comunicarse unos con otros y así interaccionar propagando su información local. Los dos lenguajes más utilizados en la actualidad son el lenguaje *Knowledge Query and Manipulation Language*, KQML [21] o *Foundation for Intelligent Physical Agents*, FIPA [22].

- El comportamiento de un sistema se ve influido por el tipo de organización de los agentes. Algunos ejemplos de estos modelos pueden ser la organización centralizada, donde un agente es el que toma las decisiones, o modelos de comunidad plural donde un agente toma la decisión y el resto refina dicha decisión.

- Los puntos de información son las localizaciones iniciales para las actividades cooperativas entre los agentes.

Uno de los problemas que surgen cuando se trabaja con SMA, es que a medida que aumenta el número de agentes en el sistema y aumentan la conectividad de dichos agentes, el número de mensajes que se intercambian sufre un importante incremento que puede llegar a sobrecargar el sistema. Los agentes necesitan establecer conexiones para intercambiar mensajes con el objetivo de cumplir los objetivos definidos y solucionar el problema para el cual el sistema fue diseñado.

Este problema de la sobrecarga de SMA ha sido estudiado con anterioridad y se sabe que es independiente de la arquitectura utilizada, [3]. Una manera de afrontar este problema es utilizando agentes móviles [4]. La idea principal consiste en determinar cuales son los agentes que tienen una alta comunicación entre ellos para colocarlos en la misma máquina. De esta manera, al tratarse de comunicación dentro de la propia máquina no se envían los mensajes por los canales de comunicación de la red y se evita la sobrecarga. La dificultad de este enfoque radica en la identificación de aquellos agentes que necesitan intercambiar grandes volúmenes de datos. En [23], se propone una solución a este problema basada en la evaluación de comunidades de agentes.

Otra manera de reducir el problema de la sobrecarga consiste en la discriminación de mensajes, [5, 6]. En este caso los agentes determinan la probabilidad que tienen los mensajes de contener información relevante. Los agentes pueden crear sus propias reglas de comportamiento analizando un histórico de inferencias pasadas.

Este trabajo fin de máster estudia el problema de la sobrecarga en sistemas multiagente, donde los agentes son estáticos (y por tanto no se pueden utilizar las soluciones propuestas en [4, 23]) y donde los agentes son del tipo SWARM. Este último requisito es necesario para poder escalar el sistema sin tener problemas de memoria en los equipos. Además los agentes poseen un sistema de discriminación de información que está inspirado en el comportamiento de células vivas [8, 24].

2.2. Proceso de discriminación de información de firmas neuronales

A continuación se describen los conceptos biológicos necesarios para el correcto entendimiento del proceso de discriminación bio-inspirado que poseen los agentes. Este proceso de discriminación está basado en estudios recientes sobre la actividad en ráfagas de neuronas vivas, [8, 24].

Las neuronas son las unidades básicas de procesamiento de información del sistema nervioso [25]. Son células con un alto grado de especialización que se diferencian del resto de tipos celulares por su alta capacidad de recibir, procesar y transmitir información. Toda neurona se compone de *soma*, *axón* y *dentritas*. El *soma* es el núcleo, o cuerpo, de la neurona. Las dentritas son ramificaciones que parten del soma y forman los *árboles dentríticos*. Su principal función es la recepción de estímulos, o señales nerviosas, y propagarlos hasta el soma, donde serán analizados. El *axón* es el encargado de propagar la señal de salida a las neuronas vecinas.

Además las neuronas poseen una *membrana citoplasmática* que las aisla de su ambiente extracelular. La carga eléctrica a ambos lados de la membrana es diferente, lo que genera una diferencia de potencial llamada *potencial de membrana*. La evolución temporal de dicho potencial de membrana caracteriza el comportamiento de la neurona.

La membrana citoplasmática presenta una permeabilidad selectiva a ciertas moléculas, lo que permite que dichas moléculas entren y salgan del interior de la neurona a través de los llamados *canales iónicos* [25].

Con el movimiento de moléculas a través de los canales iónicos, el potencial de membrana varía y estas variaciones son las que definen el comportamiento de las neuronas. Si el potencial de membrana aumenta se denomina *despolarización* y si disminuye, *hiperpolarización*. Normalmente las neuronas presentan pequeñas despolarizaciones e hiperpolarizaciones cercanas al potencial en reposo de la neurona, en este caso este tipo de actividad se denomina *actividad subumbral*. Sin embargo, existe un umbral de despolarización que una vez ha sido superado se produce un incremento y un decremento muy rápido del potencial de membrana en un corto periodo de tiempo. Este comportamiento, representado en la Figura 1, se denomina *spike* o *potencial de acción* y es el encargado de codificar la información que se va a propagar a lo largo de la red neuronal. La generación de diferentes spikes actúan en diferentes escalas temporales, lo que permite que los potenciales de acción interactúen entre sí y se produzca un procesamiento temporal de la información.

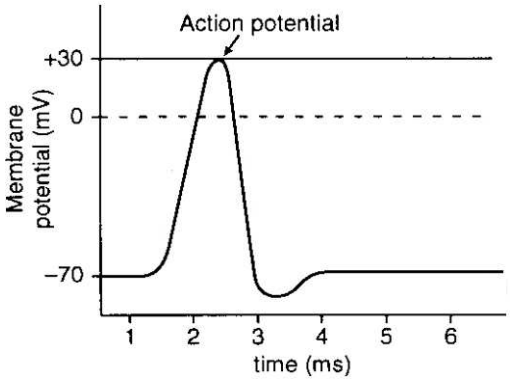


Figura 1: Representación de un potencial de acción o spike

Hay ocasiones en las que los spikes no se producen de manera aislada, sino que se agrupan en ráfagas o *burst*. Estos bursts, ver Figura 2, se caracterizan por la generación de un grupo spikes en un corto intervalo de tiempo. Después de cada burst se produce un período de actividad subumbral, conocida como *período de quiscencia*, antes de la generación del siguiente burst. Este tipo de comportamiento se conoce como *actividad en ráfagas* o *bursting activity*.

Cuando una neurona produce un spike o un burst, la actividad se propaga a lo largo de todo el axón hasta llegar a las sinápsis que son los puntos de conexión entre dos neuronas. Las sinápsis pueden ser eléctricas, químicas o graduales, y permiten la propagación de los impulsos nerviosos desde la neurona emisora, *neurona presinápica*, hasta la neurona de destino, *neurona postsináptica*. La recepción de potenciales de acción en una neurona puede provocar cambios en el potencial de membrana que pueden significar la generación de nuevos spikes y su consiguiente propagación.

Dentro de una ráfaga, se puede identificar el momento preciso en el que se produce un spike y por tanto se puede calcular el período entre spikes de una ráfaga llamado ISI, del inglés *InterSpike Interval*. Estos ISIs se suelen representar usando mapas de retorno, en los cuales el ISI del instante n es representado frente al ISI del instante $n+1$. En la Figura 2 se muestra la actividad en burst de unas neuronas del ganglio estomatogástrico de algunos crustáceos. Estas neuronas son LD y DP, y sus ISIs están representados en la Figura 3 donde la primera fila se corresponde mapas de retorno de diferentes mediciones de la neurona LD, y la segunda fila se corresponde con las mediciones de la neurona PD.

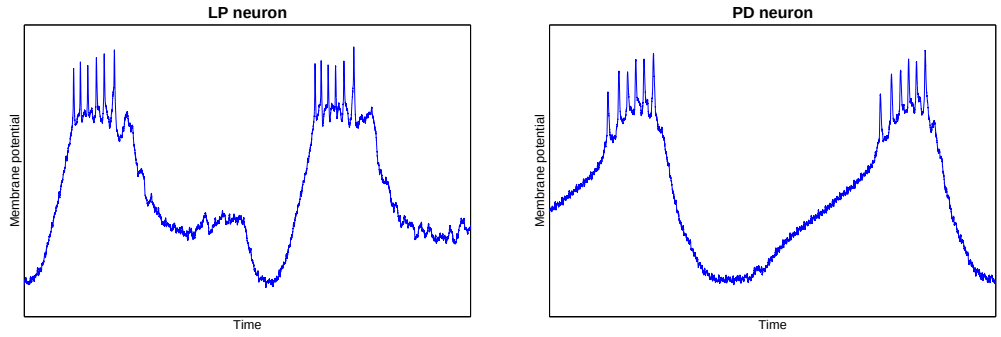


Figura 2: Ejemplos de actividad en ráfagas de las neuronas LP y PD

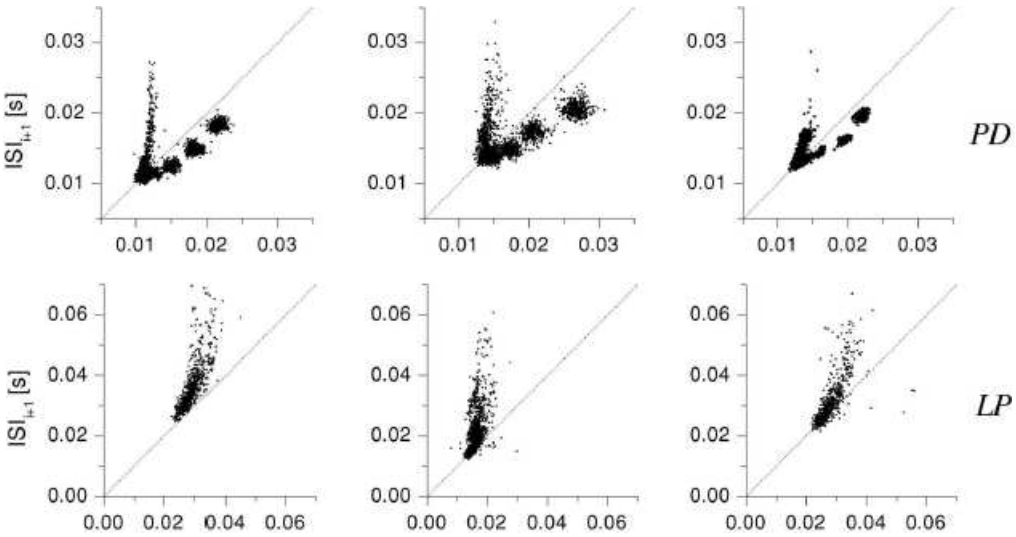


Figura 3: Mapas de retorno de los ISIs de las neuronas LP y PD

Como se puede observar en la Figura 3 los mapas de retorno de los ISIs varían mucho cuando se trata de neuronas diferentes. En cambio, cuando se trata de diferentes mediciones de una misma neurona el mapa de retorno es bastante similar. Este hecho demuestra que los ISIs identifican de manera única a la neurona y además son reproducibles [8, 26].

Que los ISIs sean reproducibles significa que diferentes neuronas tienen diferentes actividades de ráfagas y por tanto, sus correspondientes ISIs tienen una distribución temporal distinta. Esta distribución temporal de los ISIs se conoce como *firma neuronal*.

La información que se propaga en la red neuronal está codificada dentro de los *spikes* o de la actividad en ráfagas de la neurona emisora. Las neuronas que reciben la señal pueden procesar la información en base a la firma neuronal o en base a la frecuencia de aparición de los *spikes*, lo que permite realizar múltiples tareas en función de quién emita la señal. Existen redes biológicas reales, como los *CPGs* (*Central Pattern Generators*), que generan comportamientos rítmicos entre un conjunto de neuronas conectadas entre sí. Diferentes experimentos [9, 27] muestran cómo la modificación de la firma neuronal de algunas de estas neuronas producen un comportamiento rítmico completamente diferente al comportamiento cuando el sistema no ha sido alterado.

3. Descripción del sistema multiagente bio-inspirado

En esta sección se describe el Sistema Multi-Agente (SMA) utilizado en este trabajo. Los SMA se componen fundamentalmente de los agentes y de una topología que permita a los agentes comunicarse y solucionar el problema para el cual el sistema fue creado.

3.1. Modelo de Agentes Bio-Inspirados

Los agentes modelados en el sistema están basados en agentes de tipo SWARM [7]. Como ya se ha explicado en la sección 2.1, los agentes SWARM se caracterizan por ser agentes ligeros con baja autonomía. Además estos agentes son indistinguibles y no poseen ningún proceso de razonamiento o inferencia que permita cambios en su comportamiento durante el tiempo de ejecución. Los agentes utilizados en este trabajo son ligeros y tienen baja autonomía, pero cada agente tiene un identificador que permite la diferenciación entre agentes. Esta diferenciación es el concepto clave del sistema de discriminación bio-inspirado ya que los agentes analizarán, o no, los mensajes dependiendo del emisor de los mismos. Este proceso evita que los agentes analicen toda la información que reciben y por tanto, sólo la información que es relevante para el receptor será procesada. Con este sistema de discriminación, se reduce la sobrecarga a nivel de agentes ya que hay un ahorro en la cantidad de información procesada.

Como en cualquier SMA, los agentes poseen parte de la información necesaria para solucionar el problema, información local. En este trabajo, ese fragmento de información recibe el nombre de *Información de Agente*, *[AIn]* de su nombre en inglés *Agent Information*. Además de esto, cada agente posee una identificación que permite su diferenciación del resto de agentes llamada *[AId]*. Estos dos campos forman un registro que se enviará dentro de los mensajes.

Un mensaje es una colección de uno o más registros de la forma *[AId|AIn]*. La primera parte de cada registro identifica al emisor del mismo, y la segunda parte contiene la información local que dicho agente conoce. El contenido de estos campos es dependiente del problema y de cómo se modele el SMA.

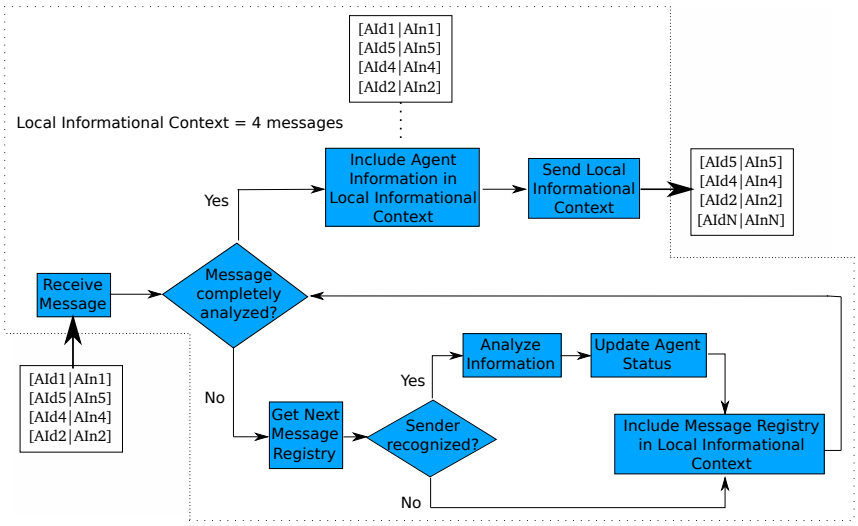


Figura 4: Comportamiento de los agentes

Para conseguir el mejor rendimiento de cualquier sistema de discriminación, es necesaria la utilización de una memoria temporal donde almacenar sucesos anteriores durante un periodo de tiempo determinado. Por esta razón, los agentes tienen una memoria temporal llamada *local informational context* donde los registros de mensaje recibidos serán almacenados. Cualquier registro de mensaje recibido es incluido en esta memoria temporal, esto significa que todos los registros se incluirán independientemente de si el sistema de discriminación determina que la información es relevante o no. Cuando todos los registros de mensajes recibidos han sido analizados, el nuevo mensaje a enviar contendrá toda la información almacenada hasta el momento en la memoria temporal.

Por último, el sistema de discriminación de mensajes bio-inspirado permite a los agentes analizar solo aquellos registros que contienen información relevante para el receptor. Esto se lleva a cabo analizando el AId de cada registro. Si el emisor identificado en este campo es reconocido, entonces la información almacenada en el AIn es analizada y el registro es añadido a la memoria del agente; pero en el caso de que el emisor no sea reconocido, el registro se almacena en la memoria temporal y el agente no analiza el campo AIn porque no contiene información relevante. Los campos AId y AIn son dependientes del problema. Dependiendo de cómo se modele el sistema, estos campos tendrán una estructura y una información específica de la cual depende la descripción de cuándo un emisor es reconocido.

Este sistema de discriminación está pensado para reducir la sobrecarga de los agentes de la red, ya que estos no analizan toda la información recibida, sino aquella que es relevante para ellos. Por ello, cuanto más largo sea *AIn* frente a *AId*, mayor ahorro computacional tendrá el sistema.

En la figura 4 se muestra un diagrama con el comportamiento de cada agente y el algoritmo 1 muestra el pseudocódigo de dicho comportamiento. Es importante tener en cuenta que no hay ningún proceso de inferencia de aprendizaje ni auto-adaptación, y por tanto, el comportamiento de cada agente es siempre el mismo.

Algoritmo 1 Agent information processing

```
1: Receive messages
2: for each received message do
3:    for each message registry do
4:       Analyze AId
5:       if Sender recognized then
6:          Analyze AIn
7:          Update status
8:       end if
9:       Include message registry in local informational context
10:   end for
11: end for
12: Send local informational context
```

3.2. Topología Inicial Básica

Cualquier SMA necesita de una topología de comunicaciones que permita a los agentes interactuar y solucionar el problema para el cual el sistema fue diseñado. Esta topología puede ser desde una organización jerárquica hasta una organización completamente aleatoria.

La primera topología utilizada en este trabajo fin de máster es una topología regular en forma de anillo. En esta topología los agentes se colocan formando un anillo y se conectan a sus vecinos más cercanos. El número de vecinos viene definido por un parámetro llamado *grado de conectividad* o k. La figura 5 muestra un ejemplo de un sistema compuesto por 16 agentes usando esta topología y con grado de conectividad 1.

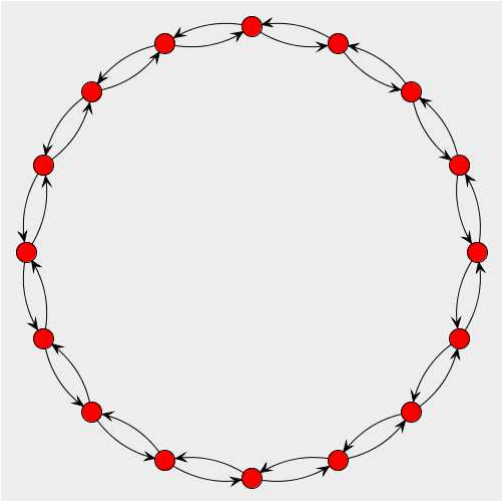


Figura 5: Topología de comunicación básica regular

Como se puede observar en la Figura 5, para un determinado grado de conectividad k, cada agente se conectará a $2k$ agentes. Además, los enlaces son dirigidos por lo que si un nodo A tiene una conexión a otro nodo B, este segundo nodo no podrá utilizar la conexión para enviar un mensaje a A sino que necesitará otra conexión.

Uno de los problemas que tienen las topologías regulares está relacionado con el camino característico del grafo subyacente. El camino característico es la longitud media de cualquier ruta entre dos nodos y determina el tiempo medio que tardará un mensaje para llegar a su destino. Que el camino característico tenga un valor alto, como es el caso de las topologías regulares, significa que por termino medio el mensaje tardará mucho tiempo en llegar a su destino, porque pasará por muchos nodos.

Una posible solución a este problema consiste en el aumento de las conexiones de la red, aumentando el parámetro k. Con esto conseguimos que el camino característico disminuya pero se enviarían más mensajes, ya que cada agente envía el mensaje por todas sus conexiones de salida, y aumentaría la sobrecarga de la red.

3.3. Topología Inicial Compleja

La segunda topología utilizada está basada en la topología anteriormente descrita. Se trata de una topología con conexiones dirigidas donde los agentes están colocados formando una topología de anillo. La principal diferencia con la topología anterior es que el grado de conectividad de cada agente está determinado por una distribución de potencias.

El uso de la ley de potencias en el grado de conectividad de cada nodo está basado en algunos trabajos que sugieren que la topología de Internet tiene esa distribución [28, 29]. Con la ley de potencias conseguimos redes que tengan características similares a la topología de Internet y además se elimina el parámetro k del sistema. Cualquier ley de potencias queda definida por la Ecuación 1.

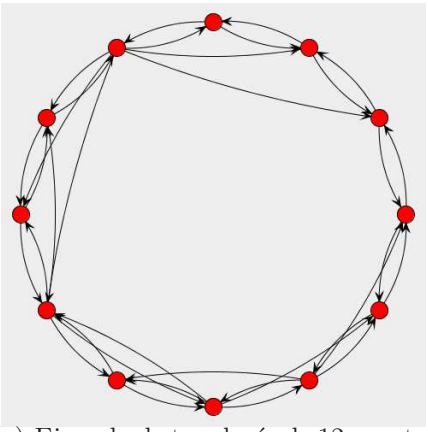
$$p(x) = \gamma x^{-\mu} \tag{1}$$

Como se puede observar existen dos parámetros en la ley de potencias μ y γ. El valor del exponente μ se ha fijado en 1.22 porque [29] se demuestra que es el valor utilizado por la topología de Internet. El valor del factor γ depende del número de agentes que existen en el sistema. Dado un sistema con N agentes, el factor γ queda definido como:

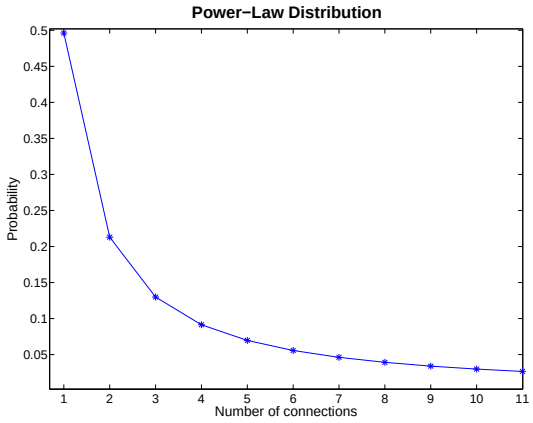
$$\gamma = \frac{1}{\sum_{i=2}^{N} i^{-\mu}} \tag{2}$$

La Figura 6(b) muestra cómo se usa la ley de potencias en este trabajo, para un sistema formado por 12 agentes. La ley de potencias relaciona el grado de conectividad con la probabilidad de que un nodo tenga dicho grado. Usando la ley de potencias se van a generar topologías donde hay muy pocos nodos con un gran número de conexiones, mientras que la mayoría de los nodos tendrán poca conectividad. La Figura 6(a) muestra la topología de comunicaciones para un sistema formado por 12 agentes y que use la ley de potencias en la distribución del grado.

Usando la ley de potencias se reduce el camino característico, sin embargo el valor del mismo aumenta a medida que el sistema se compone de más agentes. Por lo tanto se necesita optimizar la topología para que se reduzca el camino característico y así hacer que los mensajes tarden menos tiempo en llegar a su destino.



(a) Ejemplo de topología de 12 agentes



(b) Ley de potencias con 12 agentes

Figura 6: Ejemplo de topología inicial compleja con su ley de potencias

3.4. Optimización de la Topología Inicial

Como se ha visto en la sección anterior, el principal problema de las topologías regulares es que el camino característico es muy elevado. Este valor determina el tiempo medio que tardará en mensaje en llegar a su destino, y por tanto es necesario modificar la topología inicial de manera que el camino característico disminuya sin incrementar la sobrecarga de la red producida por el aumento del número de conexiones.

En este trabajo para reducir el camino característico se introduce un nuevo parámetro, p, que es la probabilidad de redirección de los enlaces. Dado un valor para este parámetro, todos los enlaces serán redireccionados, o no, de acuerdo a esta probabilidad. Los enlaces se redireccionarán a otro nodo aleatorio de la red, teniendo en cuenta que no se permiten ramas paralelas ni tampoco autoconexiones.

Una de las ventajas de la utilización de este parámetro es que se introducen atajos en la red que reducen el camino característico. Además por cada valor de p se tiene una familia de topologías con las mismas características, lo que facilita el estudio de las mismas.

La probabilidad de redireccionar un enlace fue introducida por Watts y Strogatz cuando definieron las redes de tipo Small World [30]. En estos tipos de redes, cuando el valor de p es cercano a 0 se obtienen topologías muy parecidas a la topología inicial ya que muy pocas conexiones son redireccionadas. En el caso de tener valores de p cercanos a 1 se obtienen topologías similares a las redes aleatorias. Las redes de tipo Small World se caracterizan por tener un bajo camino característico, por lo que se produce una rápida propagación de los mensajes, y un alto índice de clusterización, que hace que la red sea robusta frente a ataques. Las redes del tipo Small World han sido utilizadas en [31] para optimizar la información mutua que se intercambian entre los nodos. En este trabajo, cuando todos los enlaces han sido analizados de acuerdo a la probabilidad de redirección, la topología se fija y ya no cambia durante la ejecución del SMA.

4. Dominio de Aplicación: Jigsaw Puzzle

El sistema propuesto en este TFM se puede aplicar en aquellos sistemas en los que los agentes necesiten conocer información de su entorno para solucionar el problema. El objetivo de estos sistemas es determinar la topología sub-óptima de comunicación en la cual se resuelve el sistema enviando el menor número de mensajes posibles. Algunos ejemplos de este tipo de problemas son los problemas de *scheduling* o *job shop problems*, donde se requiere hacer una ordenación pero no se disponen de reglas de ordenación globales que ayuden a identificar la posición de los elementos.

La aplicación donde se ha probado este sistema es un conocido problema de ordenación n-dimensional, el puzle jigsaw. Resolver un puzle equivale a ordenar las diferentes piezas de manera que todas ellas encajen, pero no existe ninguna regla de ordenación global según la cual dada una pieza se pueda conocer su posición. En nuestro sistema cada pieza estará representada por un agente, que intercambiarán mensajes para solucionar el puzle.

El problema de resolución de puzles se puede afrontar de diferentes maneras. Algunos trabajos utilizan la forma de las piezas para determinar si dos piezas encajan [32, 33, 34], otros en cambio hacen un análisis de la foto que contiene cada pieza y analizan los contornos dibujados [35, 36, 37]. Existen trabajos que utilizan algoritmos de *image merging* [38], redes neuronales [39] o algoritmos genéticos [40]. En este trabajo, se intenta resolver un puzle ciego donde no hay ninguna imagen final, por lo tanto la única referencia para determinar si dos piezas son complementarias está basada en la forma de cada pieza.

La discriminación de información basada en el contexto o la identidad del emisor ya ha sido estudiada previamente [41]. El trabajo presentado en este documento contiene detalles y objetivos que lo diferencian de [41]. El objetivo de este trabajo fin de máster es el estudio de la influencia del sistema en el problema de la sobrecarga. La manera de reducir la sobrecarga estará basada, en gran medida, en la optimización de la topología de comunicación. De esta forma los agentes bio-inspirados envían el menor número de mensajes necesarios para solucionar el problema modelado.

A continuación se describe la codificación de los agentes, o cómo se realiza el mapeo agente-pieza, y las diferentes funciones de coste utilizadas en el sistema.

4.1. Codificación del agente pieza

El sistema descrito en este Trabajo Fin de Máster se ha aplicado a un problema de resolución de puzles, donde cada agente contiene una pieza del puzle y el objetivo de cada agente es encontrar aquellas piezas que son compatibles con su pieza asignada. El puzle a resolver es un puzle ciego, donde no se forma una imagen cuando el puzle está resuelto. Por lo tanto, para determinar si dos piezas encajan no se puede realizar un análisis de los colores o de la forma de la imagen contenida en cada pieza sino que se realiza es un análisis de la forma de cada uno de los lados de cada pieza.

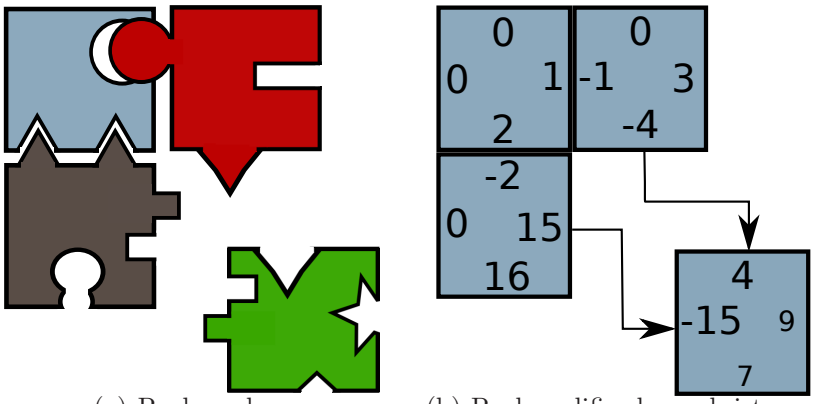
Los puzles que se pretenden resolver, además de ser puzles ciegos, son puzles jigsaw, lo que significa todas las piezas tendrán siempre 4 lados. La forma de cada lado es representado mediante un número entero teniendo en cuenta las siguientes restricciones:

- Los lados que se corresponden con el borde del puzle tienen asignados el valor 0.
- Los lados salientes de cada pieza serán asociados a valores negativos.
- Los lados que entren dentro de la pieza tendrán un valor positivo.

Para determinar si dos lados (i,j) de dos piezas encajan, el sistema comprueba si se cumple la siguiente propiedad:

$$(i + j = 0) \wedge (ij \neq 0) \tag{3}$$

Es decir, dos caras son complementarias si la suma de sus valores es 0 y ninguno de las caras se corresponde con el borde del puzle. Esta definición es posible porque en el sistema las caras son únicas y por tanto, no existen dos o más piezas que encajen en la misma cara de una tercera pieza. En la figura 7 se muestra la correspondencia entre un puzle real y su codificación dentro del sistema.



(a) Puzle real (b) Puzle codificado en el sistema

Figura 7: Codificación de un puzle en el sistema

Como ya se ha explicado anteriormente, los mensajes son una lista de registros de la forma $[AId|AIn]$. Donde el primer campo identifica al emisor del registro y el segundo contiene la información local que conoce el agente. En el caso de los puzles, el objetivo de cada pieza es conocer la posición de aquellas piezas que encajan con ella. Cada pieza conoce la posición (x, y) que ocupa dentro de la topología de anillo y dicha posición es la información que necesitan conocer el resto de piezas, por tanto estas coordenadas componen la información del agente o *AIn*. Además, cada pieza está definida e identificada unívocamente por su forma, como ya se ha dicho no existen dos lados iguales, por lo tanto el campo *AId* o identificador del agente está compuesto por la forma de la pieza. Este campo estará compuesto por 4 pares de valores $(caraIdent, valor)$ donde $caraIdent$ tendrá el valor $0, 1, 2$, o 3 dependiendo de si el valor que sigue se corresponde con la cara superior, derecha, inferior o izquierda, respectivamente. Por ejemplo, la Figura 7 hay una pieza que no está unida al puzle. Esta pieza, cuyos lados tiene los valores $4, 9, 7$ y -15 será identificada como $[(0, 4), (1, 9), (2, 7), (3, -15)]$.

Cuando un agente recibe un conjunto de mensajes empieza a analizar cada uno de los registros de estos mensajes, si el emisor del registro es reconocido, el campo de *AIn* es analizado y la pieza actualiza su estado. Que el emisor sea reconocido en este dominio de aplicación, significa que el emisor del registro tiene algún lado que encaja con el lado correspondiente del receptor. En este caso, el agente receptor memoriza la posición de la pieza emisora y cuál es el lado complementario. Cuando una pieza encuentra a otra pieza complementaria, la primera actualiza su estado. Este estado se corresponde con el número de caras que no tiene casadas. Finalmente, si el emisor del registro es reconocido como si no lo es, el registro se añade a la memoria de cada agente receptor para que sea propagado por la red.

4.2. Comunicación de los agentes

En cualquier sistema real existe un coste asociado a la comunicación. Dicho coste puede estar expresado en términos de tiempo, de calidad o monetarios. En este sistema se han diseñado dos funciones de coste diferentes asociadas al envío de mensajes entre dos agentes. El resultado de estas funciones determina el número de iteraciones que tardará en llegar a su destino un mensaje.

La primera función de coste utilizada tiene en cuenta la distancia euclídea que separa al agente emisor y receptor. Dado un agente emisor i y un agente receptor j, el coste de enviar un mensaje desde i hasta j, $C(i,j)$, se calcula de la siguiente manera:

$$C(i,j) = \lfloor \frac{EuclideanDistance(i,j)}{minDistance} \rfloor \tag{4}$$

Donde $EuclideanDistance(i,j)$ representa la distancia euclídea desde el agente i al agente j y $minDistance$ representa la distancia mínima existente entre dos agentes del sistema. Dado un sistema compuesto por N agentes, la distancia mínima se calcula con la siguiente ecuación:

$$minDistance = \sqrt{(1-\cos\alpha)^2 + (-\sin\alpha)^2} \tag{5}$$

Donde α es el ángulo que separa a dos agentes consecutivos de la red y se calcula como $\alpha = \frac{2\pi}{N}$. En la Ecuación 5 se asume que los agentes están localizados de tal manera que forman una circunferencia con centro $(0,0)$ y radio 1. La función de coste y algunos de los conceptos explicados, como el término $minDistance$, quedan representados en la figura 8.

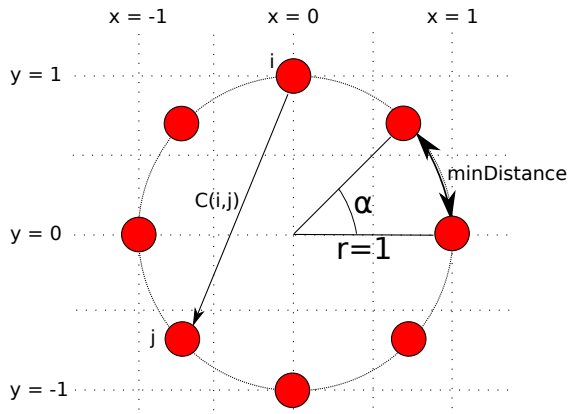


Figura 8: Representación de los costes en el envío de mensajes

La segunda función de coste tiene como base la función anterior, y queda definida de la siguiente manera. Dado un agente emisor i y un receptor j, la distancia que separa ambos agentes se calcula usando la siguiente ecuación:

$$D(i,j) = \beta \frac{EuclideanDistance(i,j)}{minDistance} \tag{6}$$

En esta ecuación $minDistance$ se calcula de la manera ya explicada en la Ecuación 5. El factor β es un factor constante cuyo objetivo es penalizar los atajos en la red. Como ya se ha dicho, los atajos son necesarios para reducir el camino característico de la red, sin embargo el objetivo es redireccionar el menor número de enlaces posibles, por esta razón una de las labores del factor β es penalizar el uso excesivo de los atajos. De todos los experimentos llevados a cabo el valor de este factor se ha fijado en 15.

Una vez que se ha calculado la distancia entre dos agentes, el coste de envío se define como:

$$C(i,j) = \begin{Bmatrix} \lceil D(i,j) \rceil & \text{if } D(i,j) - [D(i,j)] \geqslant 0.5 \\ \lfloor D(i,j) \rfloor & \text{if } D(i,j) - [D(i,j)] < 0.5 \end{Bmatrix} \tag{7}$$

Donde $[X]$ representa la parte entera de X.

5. Experimentos y Resultados

En esta sección se describen todos los experimentos realizados y se analizan los resultados obtenidos. Como ya se ha explicado en las secciones anteriores, tenemos dos topologías de comunicaciones diferentes y dos funciones de coste. Esto se debe a que la primera fase de experimentación utiliza la topología inicial básica y la función de coste simple, definida en la Ecuación 4. Con los resultados obtenidos en esa primera fase de experimentos, se han realizado diversos cambios tanto en la topología de comunicaciones como en la función de coste, dando como resultado la topología compleja (aquella que tiene una distribución de potencias en los grados de los nodos) y la función de coste definida en la Ecuación 7. A continuación se describen los experimentos y los resultados obtenidos en las dos fases de experimentos.

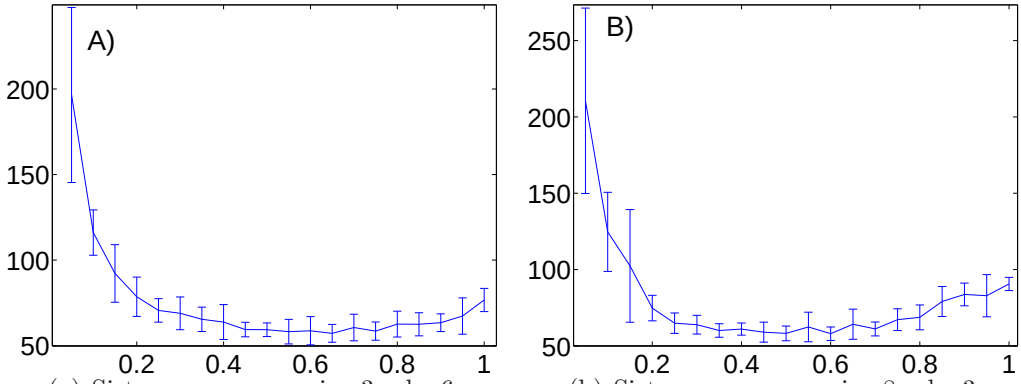
5.1. Fase de Experimentación Número 1

En la primera fase de experimentos se ha utilizado la topología inicial básica (aquella en la cual se define un grado de conectividad para los nodos) y la ecuación de coste simple definida por la Ecuación 4. El objetivo de este primer bloque de experimentos es realizar un estudio inicial de la influencia del tamaño de la memoria y de la probabilidad de redirección en el rendimiento del sistema.

El rendimiento del sistema se mide en función del número de iteraciones necesarias para solucionar el puzle. El sistema se ejecuta en local y por tanto, una iteración se corresponde con el tiempo empleado por el sistema en ejecutar una vez el algoritmo 1 de todos sus agentes.

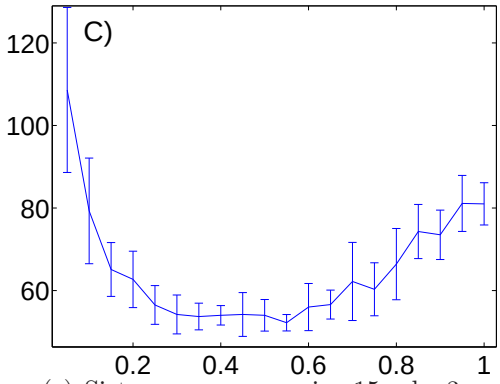
Todos los experimentos de esta fase intentan resolver un puzle compuesto por 100 piezas. Las piezas tienen lados únicos, por lo que no puede haber dos piezas que encajen en el mismo lado de una tercera, además la acción de rotar piezas no está considerada.

Para analizar la influencia de la probabilidad de redirección, se han definido 21 valores diferentes para este parámetro, que van desde 0 hasta 1 con incrementos de 0.05. Se han probado diferentes valores para el tamaño de la memoria de los agentes, los resultados más interesantes se muestran en la Figura 9.

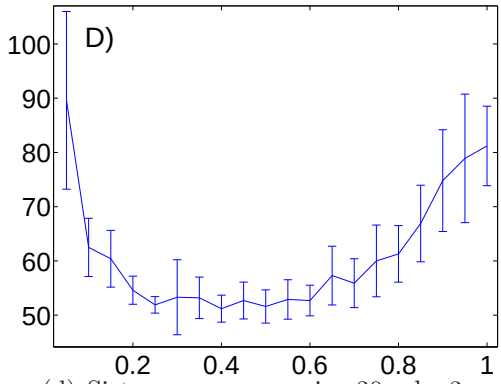


(a) Sistema con memoria=3 y k=6

(b) Sistema con memoria=8 y k=3



(c) Sistema con memoria=15 y k=3



(d) Sistema con memoria=30 y k=3

Figura 9: Resultados de la fase de experimentación número 1

Como se pueden observar en la Figura 9, pequeños cambios en la topología inicial (esto es probabilidad de redirección baja) reducen significativamente el número de iteraciones que son necesarias para solucionar el sistema. Además existe un umbral para esta probabilidad de redirección a partir del cual el número de iteraciones se mantiene estable (como es el caso de la figura 9(a)) o aumenta (figuras 9(b), 9(c) y 9(d)). De esta observación se puede concluir que existe un valor limite óptimo para la redirección de probabilidad.

En la Figura 10 se realiza un análisis comparativo del número de ejecuciones necesarias para resolver los puzles entre los sistemas cuyos resultados se muestran en la Figura 9. Como se puede observar, el tamaño de la memoria también influye en el rendimiento del sistema ya que a medida que aumenta la memoria de los agentes, el sistema necesita menos iteraciones para solucionar el problema. Sin embargo, el comportamiento del sistema con memoria 15 es muy similar al comportamiento con memoria 30, esto su-

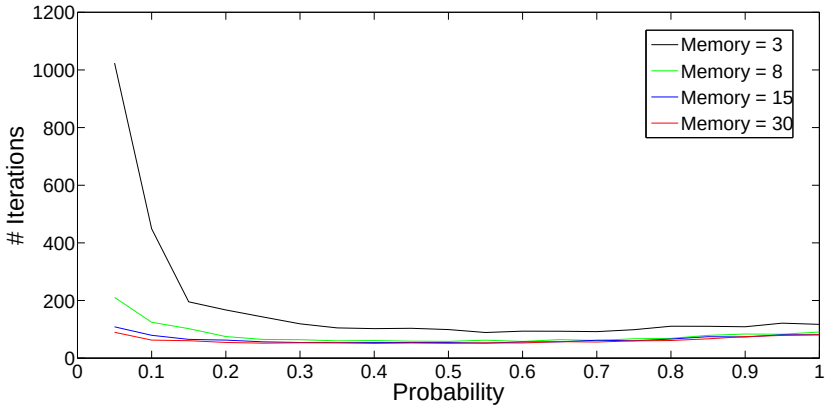


Figura 10: Estudio comparativo de la influencia de la memoria

giere que también existe una memoria límite a partir de la cual el sistema apenas se ve afectado. La identificación de estos límites, tanto en la probabilidad de redirección como en el tamaño de la memoria, es una tarea muy importante para diseñar sistemas que utilicen de manera óptima sus recursos.

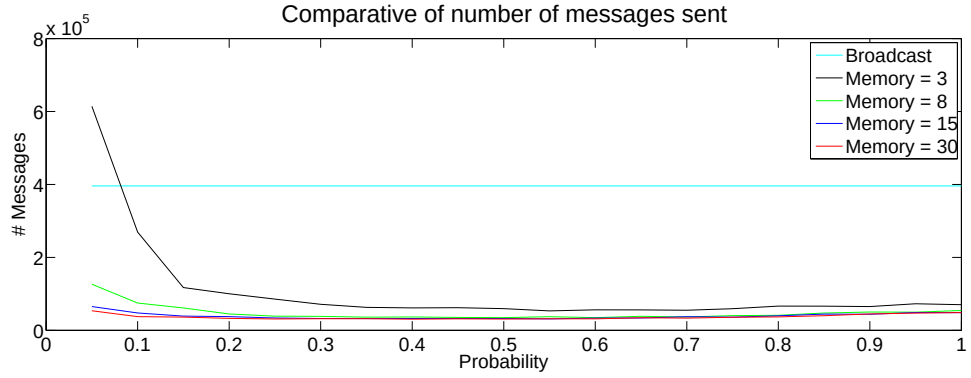


Figura 11: Estudio comparativo del número de mensajes enviados

El siguiente objetivo de la experimentación consiste en analizar si el sistema propuesto reduce el problema de la sobrecarga. Para ello, de los experimentos mostrados en la Figura 9 se analizan el número de mensajes enviados por el sistema y se comparan con los mensajes que se enviarían en la situación de broadcast. En el caso de que se tuviera un sistema broadcast, todos los nodos enviarían su información al resto. La construcción de sistemas broadcast garantizan la solución del problema, porque todos los mensajes llegan a

todas las piezas, sin embargo la construcción de estos sistemas se complica a medida que aumentan los nodos de la red. Como se puede observar en la Figura 11 con valores pequeños de redirección se necesitan menos mensajes para solucionar el problema. Esto significa que se reduce la sobrecarga de la red provocada por el envío masivo de mensajes.

5.2. Fase de Experimentación Número 2

En objetivo de la segunda fase de experimentación es realizar un estudio más detallado de la influencia que tienen en el rendimiento del sistema los parámetros probabilidad de redirección y el tamaño de la memoria de los agentes. En este nuevo bloque se utiliza una topología y una función de coste diferente al bloque anterior. En este caso, se utiliza una topología de comunicaciones donde el grado de conectividad de los nodos está determinado por una ley de potencias, y la función de coste está definida en la Ecuación 7. El rendimiento del sistema es medido por el número de iteraciones necesarias para solucionar el puzle, y por el error relativo global (ε_R).

Dada una pieza i se puede definir su error individual (ε_i) como el número de lados que la pieza no tiene casados. Con este enfoque, el error máximo que puede tener una pieza es 4 (en el caso de que no tenga ningún lado encajado con otra pieza) y el error mínimo es 0 (cuando la pieza esté completamente integrada en el puzle). Los lados de las piezas que se corresponden con el borde del puzle no contribuyen en el error de la pieza. Una vez que se define el error individual de una pieza, se puede definir el error de un puzle (ε) como la suma del error individual de todas sus piezas. Dado un puzle compuesto por N piezas, el error de dicho puzle calcula usando la siguiente ecuación:

$$\varepsilon = \sum_{k=1}^{N} \varepsilon_k \tag{8}$$

Además, el error máximo de un puzle, ε_T, se puede definir como el número de caras distintas de cero que tiene el puzle, ya que las caras que valen 0 representan el borde del puzle y no se tienen en cuenta a la hora de calcular el error. Este error se correspondería con la situación inicial del puzle en la que no hay ninguna pieza conectada a otra. La fórmula que determina el error máximo de un puzle formado por N piezas es:

$$\varepsilon_T = 4(N - \sqrt{N}) \tag{9}$$

Por último, el error relativo global de un puzle, ε_R, define el porcentaje del puzle que no está resuelto. Este error está definido por la Ecuación 10 y es utilizado para medir el rendimiento del sistema.

$$\varepsilon_R = \frac{\varepsilon}{\varepsilon_T} \tag{10}$$

Para el estudio de la influencia de los parámetros se han analizado puzles compuestos por 100, 144 y 225 piezas. El objetivo de estas pruebas es analizar el comportamiento del sistema con los diferentes valores de la probabilidad de redirección y el tamaño de la memoria, y definir los rangos de valores de estos parámetros en los que se observa el mejor rendimiento del sistema.

La probabilidad de redirección toma valores desde 0 hasta 1 con incrementos de 0.1, para asegurarnos de que estamos estudiando todo el rango de valores de dicho parámetro. Los valores del tamaño de la memoria de los agentes es más difícil de definir, ya que el valor mínimo para el tamaño es 1 y el máximo es el número de piezas que componen el puzle. Si el valor se fija en 1 el sistema nunca resolverá el puzle, ya que los agentes sólo memorizarán la información de pieza que tienen asignada. Por otro lado, el valor máximo puede exceder el número de piezas del puzle pero en este caso habría un desperdicio de los recursos del sistema ya que los agentes tienen más memoria que la que van a utilizar. Si el tamaño máximo de la memoria es el número de piezas del sistema, habrá un momento en el sistema en el que uno o varios mensajes contengan todos los registros de mensajes de todas las piezas del puzle.

En la siguiente tabla se describen los valores mínimos, máximos y el incremento del tamaño de la memoria utilizados para los diferentes puzles utilizados.

Num. Agents	Min. Memory	Max. Memory	ΔMemory
100	10	100	10
144	20	140	20
225	60	140	20

Tabla 1: Valores utilizados para el tamaño de la memoria.

Una vez que los rangos de valores para los diferentes parámetros se han fijado, se realiza el estudio del comportamiento del sistema. Este estudio consiste en la ejecución del sistema con todas las posibles combinaciones de valores de los parámetros. Por ejemplo, si se definen 3 valores diferentes para la probabilidad de redirección (0, 0.5 y 1) y 2 valores para el tamaño de la memoria (3 y 6), se ejecutará el sistema con probabilidad 0 y memoria 3; después con probabilidad 0 y memoria 6; a continuación se ejecuta con probabilidad 0.5 y memoria 3, y así sucesivamente hasta analizar todas las posibles combinaciones de valores de ambos parámetros.

La Figura 12 muestra el número de iteraciones necesarias para resolver puzles por sistemas compuestos por 100 agentes. Como se puede observar en estas figuras, cuando se realizan pequeños cambios en la topología inicial, el sistema necesita menos iteraciones para solucionar el puzle.

Respecto al análisis de la memoria, analizando la figura 12(a) puede parecer que el tamaño de la memoria no afecta al rendimiento del sistema ya que no se observan cambios significativos en el número de iteraciones provocados por el tamaño de la memoria. Sin embargo, como ya se observó en la fase de experimentación anterior, existe un límite los valores de este parámetro a partir del cual el sistema no se beneficia significativamente de dicho tamaño.

Si analizamos el comportamiento del sistema con pequeños tamaños de memoria (Figura 12(b)), sí se observan cambios en el número de iteraciones. Siendo más específicos, se puede ver cómo para un tamaño de memoria de 2 registros, el sistema necesita aproximadamente 6000 iteraciones, mientras que para una memoria máxima de 10 registros, se necesitan solamente 2000. Además se observa que a partir del tamaño de memoria de 6 registros de mensajes, el número de iteraciones permanece estable, por lo que se puede concluir que el limite en el tamaño de memoria está fijado en 6 registros.

Comportamientos similares se han observado en los sistemas compuestos por 144 y 225 agentes, que resuelven puzles de 144 y 225 piezas. El comportamiento del sistema compuesto por 144 agentes se muestra en la Figura 13, mientras que los resultados del sistema que soluciona puzles de 225 piezas se muestran en la Figura 14. En ambos casos se observa el mismo comportamiento del sistema que en el caso anterior. Es decir, una memoria pequeña en los agentes provoca que el sistema necesite más iteraciones para resolver los puzles.

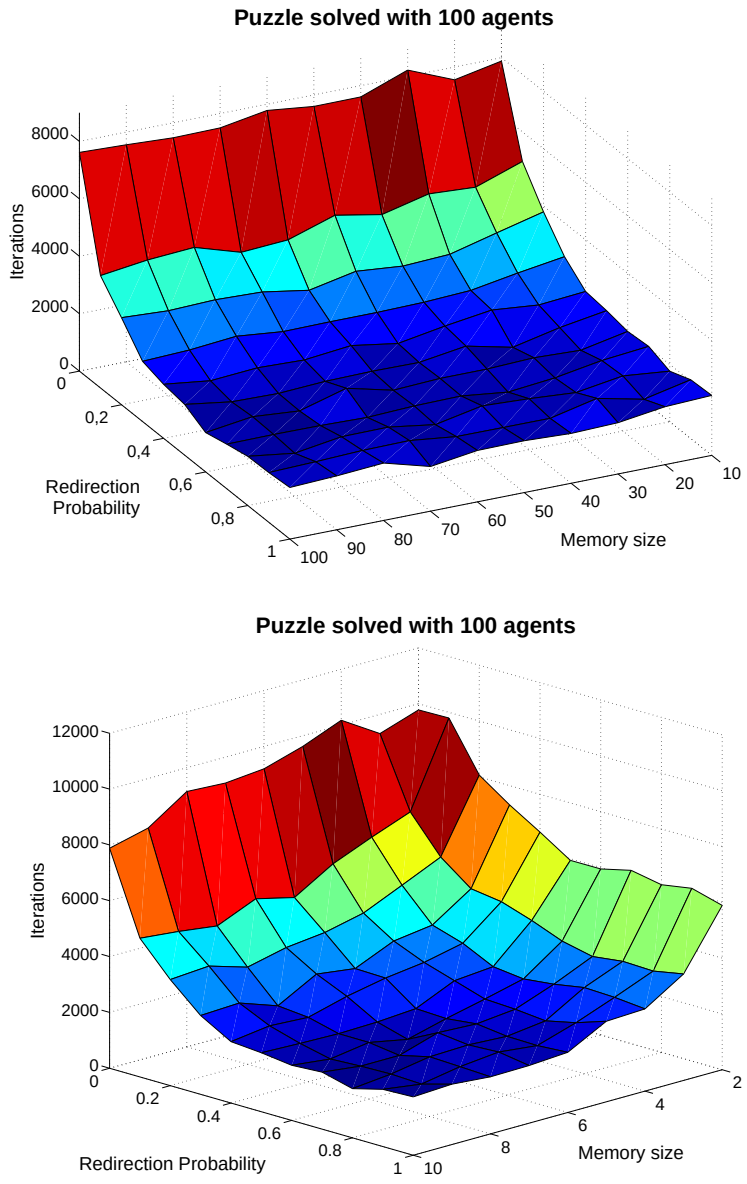


Figura 12: Rendimiento del sistema compuesto por 100 agentes

Como ya se ha explicado anteriormente, otra manera de medir el rendimiento del sistema es utilizando el error relativo global del puzle (ε_R) definido en la Ecuación 10. Este error representa el porcentaje del puzzle que no está resuelto.

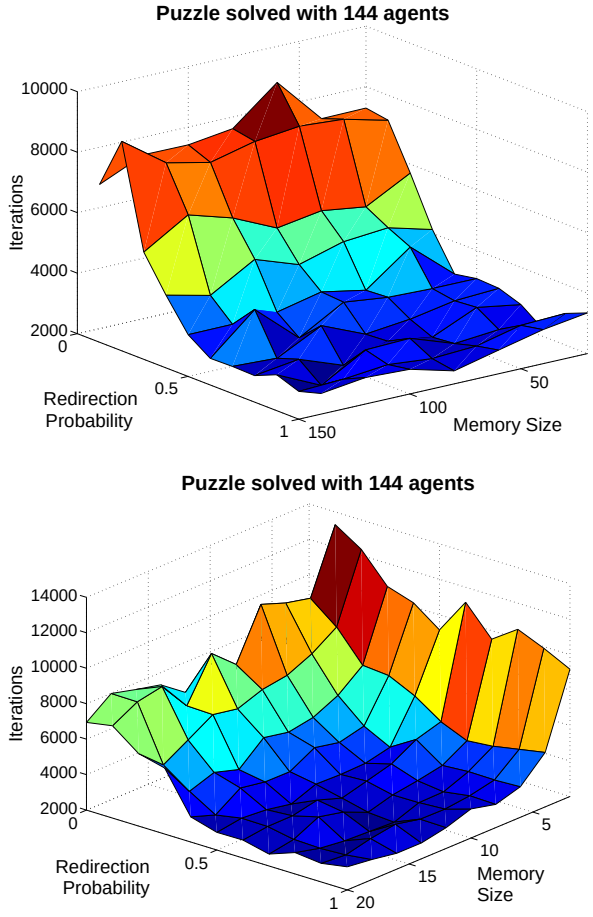


Figura 13: Rendimiento del sistema compuesto por 144 agentes

La Figura 15(a) muestra los resultados expuestos en la Figura 12(a), pero sin mostrar la dimensión del tamaño del a memoria. Esta dimensión puede ser ignorada porque en dichos experimentos ya se ha alcanzado el límite en el tamaño de la memoria. La Figura 15(b) muestra el error relativo global de los experimentos y se puede observar que con una probabilidad de redirección mayor o igual a 0.5 no hay errores en el sistema, esto significa que el puzle es resuelto completamente.

Analizando los resultados mostrados en la Figura 15, se podría concluir que el mejor rendimiento del sistema se alcanza con una redirección de probabilidad entre 0.5 y 0.6 ya que estos son los menores valores en los cuales el puzle es resuelto completamente. Sin embargo, con una redirección de probabilidad entre 0.2 y 0.4 el sistema necesita más iteraciones en resolver el puzle

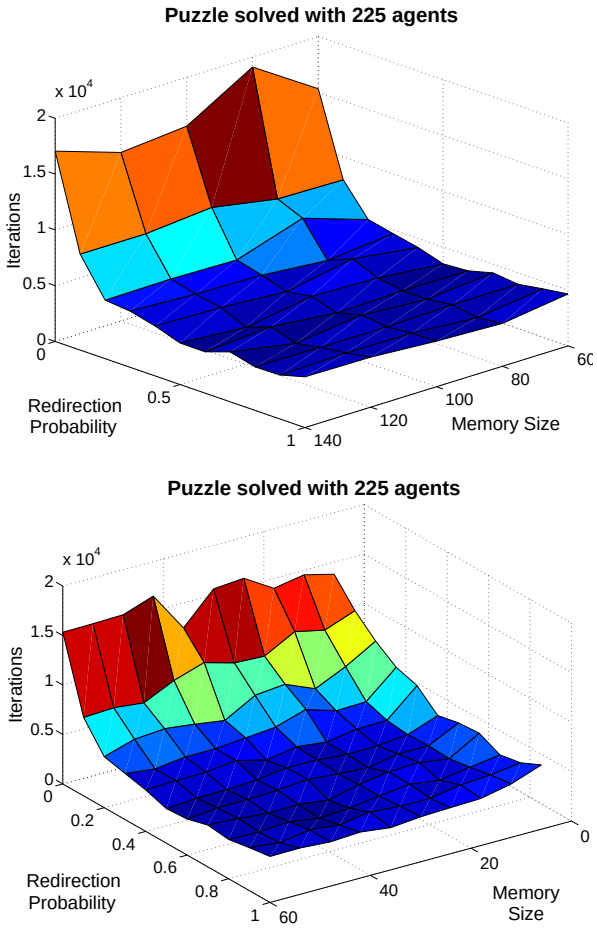
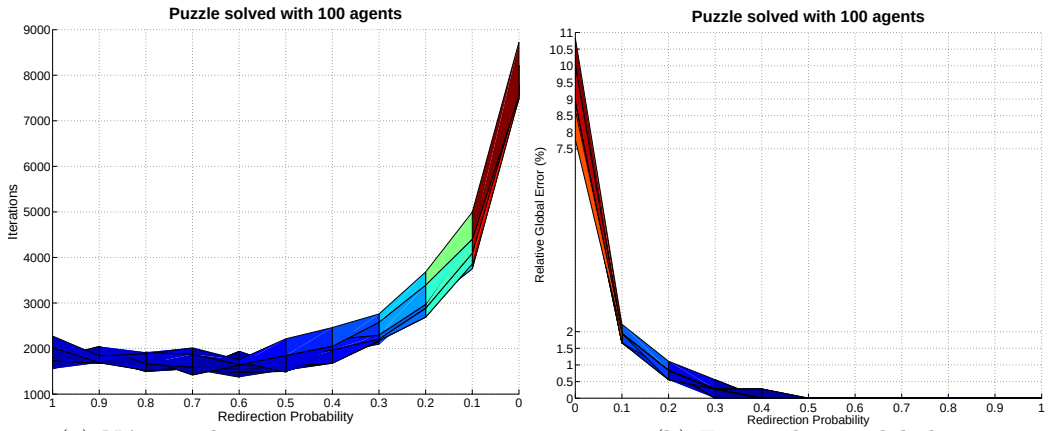


Figura 14: Rendimiento del sistema compuesto por 225 agentes

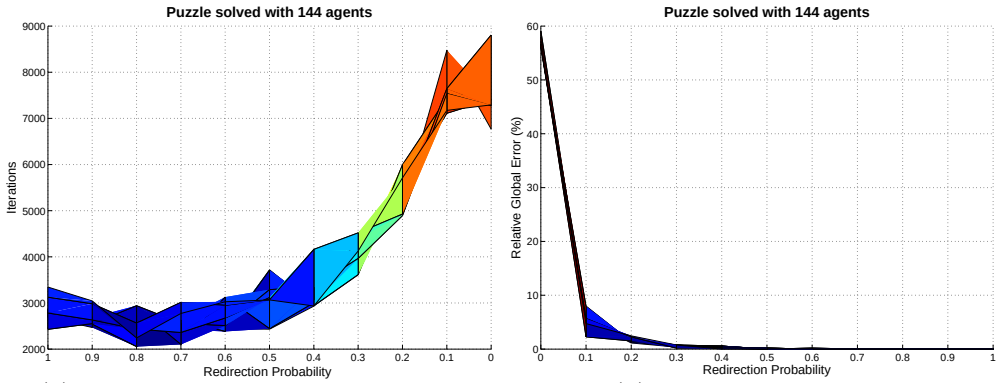
pero el error relativo global es despreciable, ya que es menor o igual al 1 %. En esta situación, el puzle está prácticamente resuelto y se han necesitado redireccionar menos enlaces que usando una probabilidad de redirección de 0.5.

En el caso de los sistemas que están compuestos por 144 agentes, el mejor intervalo de valores para la probabilidad de redirección es entre 0.3 y 0.4, donde se obtienen errores relativos globales de 0.54 % y 0.29 % respectivamente. Estos resultados se muestran en la Figura 16.



(a) Número de iteraciones necesarias (b) Error relativo global

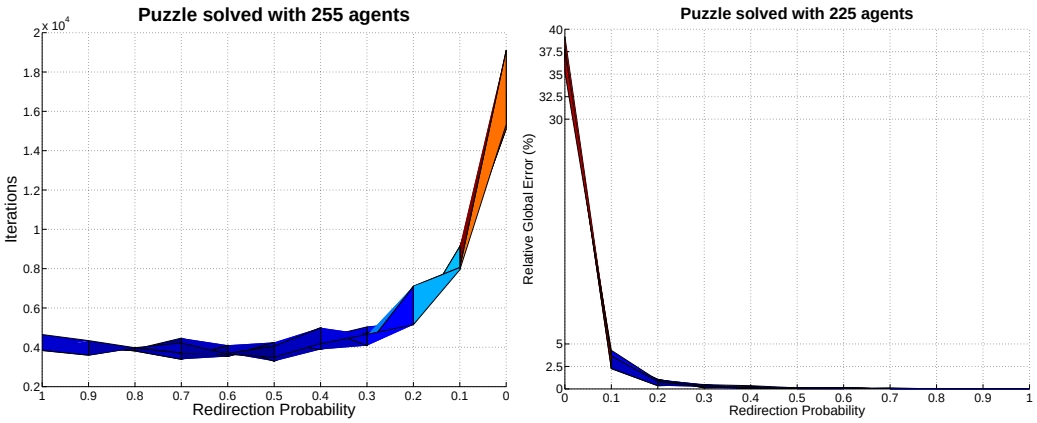
Figura 15: Iteraciones y error relativo en puzles de 100 piezas



(a) Número de iteraciones necesarias (b) Error relativo global

Figura 16: Iteraciones y error relativo en puzles de 144 piezas

Finalmente, la Figura 17 muestra las iteraciones y el error relativo global asociado a sistemas que intentan solucionar puzles de 225 piezas. En este caso sucede lo mismo que en el caso anterior, con una probabilidad de redirección igual o superior al 50 % el puzle es resuelto en su totalidad, pero el número de enlaces que son redirigidos es muy alto. Por tanto, un buen rendimiento del sistema se obtiene cuando la probabilidad de redirección toma valores entre 0.2 y 0.4, porque el error relativo global es 0.83 % y 0.14 % respectivamente.



(a) Número de iteraciones necesarias (b) Error relativo global

Figura 17: Iteraciones y error relativo en puzles de 225 piezas

En la tabla 2 se muestran los rangos de valores para la probabilidad de redirección y sus correspondientes errores relativos asociados para los sistemas compuestos por 100, 144 y 225 agentes.

Agents	Range Values for Redirection Probability	ε_R
100	$0.2 - 0.4$	$0.8 - 0.02$
144	$0.3 - 0.4$	$0.54 - 0.29$
225	$0.2 - 0.4$	$0.83 - 0.14$

Tabla 2: En esta tabla se muestran para cada sistema el rango de valores seleccionado para la probabilidad de redirección y su correspondientes errores relativos globales.

El objetivo de la tabla 2 es determinar el rango de valores sub-óptimos de la redirección de probabilidad para poder usar dicho rango en la construcción de sistemas compuestos por un mayor número de agentes. De esta tabla, se puede concluir que el mejor rendimiento del sistema se obtiene, de manera general, redireccionando entre el 20 % y el 30 % de los enlaces de la red.

Con estos valores, se han ejecutado sistemas compuestos por 400, 625 y 1024 agentes. El tamaño máximo de la memoria de los agentes en los sistemas de 400 y 625 agentes ha sido fijado en el 20 % del número de piezas que componen los puzles. Los resultados ya analizados determinan que con un tamaño de memoria del 20 %, el límite en el tamaño de la memoria se ha alcanzado y por tanto, nos aseguramos el menor número de iteraciones debido al tamaño de memoria. Por otro lado, para evitar problemas de memoria en el equipo en el que se lanzan las pruebas, el tamaño máximo de memoria para el sistema compuesto por 1024 agentes se ha fijado en 50 registros. La tabla 3 muestra las iteraciones y el error relativo global de estos sistemas.

Agents	Memory Size	Redirection Probability	Iterations	ε_R
400	80	0.3	10353	1.184
		0.4	10055	1.711
625	125	0.3	19016	3.833
		0.4	17522	2.25
1024	50	0.3	23063	0.806
		0.4	17697	0.958

Tabla 3: Resultados de las ejecuciones con sistemas grandes

Como se puede observar en esta tabla, el número de iteraciones aumenta, hecho que es lógico debido a que ha aumentado el tamaño de los puzles que se están resolviendo. Los datos importantes son los errores relativos globales asociados a estos sistemas, donde se puede ver que el peor error relativo global es inferior al 4 % de un puzle de 625 piezas. Esto supone que hay 25 piezas de 625 que no están encajadas con el resto del puzle. En estos casos al tratarse de errores relativos tan bajos se puede concluir que los rangos para los valores de la redirección de probabilidad son extrapolables a sistemas más complejos.

6. Conclusiones

Uno de los problemas más comunes cuando se trabaja con sistemas multiagente es el problema de la sobrecarga. Este problema aparece cuando el sistema se compone de un gran número de agentes que tienen que intercambian información para la consecución de unos determinados objetivos. El problema de la sobrecarga puede aparecer en dos niveles diferentes: a nivel de red y a nivel de agente.

Este trabajo fin de máster supone un estudio sobre la reducción del problema de la sobrecarga en sistemas multiagente compuestos por un gran número de agentes. Para ello, los agentes utilizados son agentes bio-inspirados, que poseen un sistema de discriminación basado en la identidad del emisor de la información recibida. Este sistema de discriminación disminuye el tiempo de cómputo de mensajes en los agentes receptores (ya que no todos los mensajes van a ser analizados sino solo aquellos que contienen información relevante) y por tanto se reduce la sobrecarga de los agentes.

La reducción de la sobrecarga de la red se realiza redireccionando canales de comunicaciones conforme a una probabilidad de redirección dada. Redireccionando los enlaces de esta manera se reduce el camino característico de la red, y por consiguiente, se reduce el tiempo medio que los mensajes necesitarán para llegar a su destino. Además se han analizado topologías similares a la topología de Internet donde el grado de conectividad de cada nodo sigue una distribución de ley de potencias.

El sistema multiagente bio-inspirado se ha aplicado a tareas de resolución de puzles donde cada agente contiene una pieza y el objetivo es resolver el puzle. A nivel individual, cada agente necesita identificar la posición de aquellas piezas que encajan con su pieza asignada. Para ello, los agentes poseen dos características llamadas *Agent Identification*, que identifica al agente, y *Agent Information* que contiene la información local que posee el agente sobre el problema modelado.

El campo *Agent Information*, que identifica al agente emisor, permite realizar las tareas de discriminación y por tanto, si el emisor es reconocido se analiza el campo *Agent Information*. Es importante tener en cuenta que para obtener el mayor beneficio de este sistema de discriminación es necesario que el campo *Agent Information* sea mucho mayor que *Agent Identification*. Por otro lado, los agentes tienen una memoria temporal donde van almacenando durante cierto tiempo todos los registros de mensaje recibidos. El tiempo que

esta información estará almacenada depende del tamaño de dicha memoria, ya que la memoria tiene una política FIFO y cuando se completa la memoria, la información más antigua es la que se va desechando. Sin embargo, que se deseche información de la memoria, no significa que la información desaparezca de la red porque ese registro que es desechado puede estar en la memoria de alguna otra pieza.

Los resultados de las fases de experimentación demuestran que pequeños cambios en la topología de comunicaciones reducen significativamente el número de mensajes intercambiados para solucionar los puzles, reduciendo por tanto el problema de la sobrecarga de las comunicaciones del sistema.

El tamaño de la memoria así como la probabilidad de redirección son parámetros esenciales en el comportamiento del sistema, ya que ambos parámetros afectan al número de iteraciones necesarias para solucionar los puzles. También se han observado unos valores límite en dichos parámetros a partir de los cuales el rendimiento del sistema no mejora significativamente.

Los resultados experimentales han permitido, partiendo de sistemas compuestos por 100, 144 y 225, establecer los valores sub-óptimos de la redirección de probabilidad donde se obtiene el mejor rendimiento del sistema. Dicho rendimiento está determinado tanto por el número de iteraciones necesarias para solucionar el puzle, como por el porcentaje del mismo que no está resuelto. El rango de valores se ha determinado entre 0.3 y 0.4 y ha servido para analizar el rendimiento de sistemas más complejos. Estos nuevos sistemas estaban compuestos por 400, 625 y 1024 agentes y los resultados muestran que el error relativo global es, por término medio, inferior al 2 %, obteniendo en el peor de los casos un error cercano al 4 %. Lo que hace indicar que en el rango de valores seleccionado para la probabilidad de redirección se obtiene un buen rendimiento del sistema.

A pesar del trabajo presentado en este Trabajo Fin de Máster, existen varias cuestiones que deberían ser analizadas para el correcto entendimiento de la plataforma propuesta. En primer lugar será necesario el análisis de estos sistemas multiagente bio-inspirados con un gran número de agentes. Para realizar esta tarea se deben utilizar plataformas software de *Large Multi-Agent Systems* como Hadoop [42] o ZASE [43]. También se deberá analizar el comportamiento del sistema en otros dominios de aplicación donde el volumen de datos a analizar excedan el campo identificativo de cada agente, ya que en estos casos es donde se obtendrán los mejores resultados del proceso de discriminación.

7. Aportaciones

Durante estos dos años de máster la investigación realizada se ha centrado en Algoritmos Genéticos, Sistemas multiagente y Recuperación de información mediante algoritmos de clustering basados en compresión de datos. Estas áreas están íntimamente ligadas con asignaturas ofertadas en el máster como Computación Evolutiva o Análisis y Simulación de Sistemas Complejos, entre otras. Más concretamente, este trabajo fin de máster se basa en conocimientos adquiridos en las asignaturas de Análisis y Simulación de Sistemas Complejos, Neurociencia I, Neurociencia II y Modelos de conectividad.

A pesar de tener resultados en otras áreas, se ha decido presentar este trabajo de Sistemas Multiagente con un proceso de discriminación bio-inspirado debido a que se han obtenido dos publicaciones: una de ellas es una conferencia internacional que ya está publicada y el otro trabajo se ha enviado a la revista *Concurrency and Computation: Practice and Experience* que pertenece al JCR y tiene un índice de impacto de 1.004.

El artículo de la conferencia se titula: *Optimal Message Interchange in a Self-organizing Multi-agent System* [10] (ver apéndice A). Este trabajo fue presentado el pasado mes de Septiembre en Tánger, en la conferencia *Intelligent Distributed Computing* cuyos artículos han sido publicados por Springer.

Dicho trabajo hacía un estudio inicial sobre cómo el sistema de discriminación bio-inspirado de los agentes junto con el diseño topológico de la red de comunicaciones influía en la sobrecarga del sistema. Se observó que tanto la definición de la topología como la memoria de los agentes bio-inspirados afectan significativamente al rendimiento del sistema, medido tanto en número de iteraciones necesarias para resolver el sistema como en número de mensajes enviados.

Partiendo de ese trabajo se estudió en detalle la influencia de los parámetros del sistema en el rendimiento del mismo. El uso de una ley de potencias en la distribución del grado de conectividad de los agentes ha permitido la eliminación de uno de los parámetros del sistema. El estudio realizado ha permitido definir el rango de valores para los parámetros del sistema en los cuales se obtiene el mejor rendimiento, medido como el número de iteraciones necesarias para resolver el sistema y el error relativo global asociado [11] (ver Apéndice B). Ambos trabajos se encuentran anexados al final del presente documento.

8. Trabajos Publicados

En esta sección se presentan los trabajos realizados que han sido publicados. Estos trabajos han sido publicados en conferencias internacionales, y pertenecen a diferentes áreas. A continuación se citan los trabajos relacionados con la extracción de patrones mediante la evolución de expresiones regulares:

- **D.F. Barrero, A. Gonzalez-Pardo, D. Camacho, M.D. R-Moreno**. Parameter Tunning for Genetic Algorithm. Journal: *Computer Science and Information Systems (COMSIS)*. 2010, Publisher ComSIS Consortium Vol. 7, pp.661-677.
- **A. Gonzalez-Pardo, D.F. Barrero, D. Camacho, M.D. R-Moreno**. A Case Study on Grammatical-Based Representation for Regular Expression Evolution. *Trends in Practical Applications of Agents and Multiagent Systems*. Springer 2010, Vol. 71, pp.379-386.
- **D.F. Barrero, A. Gonzalez-Pardo, M.D. R-Moreno, D. Camacho**. Variable Length-Based Genetic Representation to Automatically Evolve Wrappers. *Trends in Practical Applications of Agents and Multiagent Systems*. Springer 2010, Vol. 71, pp.371-378.

También se ha estudiado la repercusión que tienen los semáforos en los niveles de congestión de una carretera. Este trabajo está más relacionado con los Sistemas Multiagente, ya que utiliza agentes de tipo SWARM para simular los vehículos y los semáforos. El trabajo en cuestión es el siguiente:

- **R. Cajias, A. Gonzalez-Pardo, D. Camacho**. A Multi-Agent Traffic Simulation Framework for Evaluating the Impact of Traffic Lights. *3rd International Conference on Agents and Artificial Intelligence*. 2010 pp.443-446.

Por último, también se han realizado algunos trabajos en temas de clustering de archivos musicales utilizando algoritmos genéticos y medidas de similaridad basadas en compresión, y clustering de comportamiento de avatares en mundos virtuales. El primer trabajo estudiaba la influencia de diferentes representaciones de archivos musicales en el rendimiento de tareas de clustering evolutivo. El segundo trabajo estudia la posibilidad de clasificar el comportamiento personas en mundos virtuales con fines educativos, mediante la extracción de características de los avatares.

- **A. Gonzalez-Pardo, A. Granados, D. Camacho, F.B. Rodriguez Ortiz**. Influence of music representation on compression-based clustering. *EEE Congress on Evolutionary Computation (CEC)*. IEEE Xplore 2010 pp.2988-2995.
- **A. Gonzalez-Pardo, F.B. Rodriguez Ortiz, E. Pulido, D. Camacho**. Using Virtual Worlds for Behaviour Clustering-based Analysis. *ACM Workshop on Surreal Media and Virtual Cloning*. ACM 2010 pp.9-14.

9. Trabajos Enviados

Además del journal internacional [11], se ha enviado otro trabajo relacionado con la recuperación de expresiones regulares mediante la evolución de Gramáticas de Christiansen. La información sobre este trabajo se muestra a continuación:

- **A. Gonzalez-Pardo, D. Camacho**. Analysis of Grammatical Evolution Approaches to Regular Expression Induction. *2011 IEEE Congress on Evolutionary Computation (CEC)*.

A. Optimal message interchange in a self - organizing MAS

- **Title**: Optimal message interchange in a self-organizing multi-agent system
- **Authors**: Antonio Gonzalez-Pardo, Pablo Varona, David Camacho, Francisco B. Rodriguez Ortiz
- **Publication:** 4th International Conference on Intelligent Distributed Computing (ICD 2010). Vol. 315, pages 131-141.
- **Editorial**: Springer-Verlag
- **ISBN**: 978-3-642-15210-8

- **Abstract**:Over the last decade there has been a growing interest on Intelligent Agents and Multi-Agent Systems (MAS) in several fields such as Artificial Intelligence (AI), Software Engineering, Psychology, etc... Different problems can be solved in these fields by creating societies of agents that communicate with each other. Nevertheless, when the number of agents is large and the connectivity is extensive, the system suffers from overhead in the communication among agents due to the large number messages exchanged. This work addresses the search for an optimal communication topology to avoid these situations. This optimal topology is characterized by the use of a redirecting probability in the communication. The redirection of a communication is performed before the execution of the MAS. Once agents start the execution, the topology is fixed and remains unchanged. This characteristic is useful in those systems where a given topology can not be changed as, for example, in wired networks. On the other hand, in the proposed solution agents contain a local message discrimination process as a function of the sender of the message. Experiments show an important improvement in terms of a reduction in the number of iterations needed to solve the problem and also in the number of messages exchanged.

Optimal Message Interchange in a Self-organizing Multi-agent System

Antonio González-Pardo, Pablo Varona, David Camacho, and Francisco de Borja Rodriguez Ortiz

Abstract. Over the last decade there has been a growing interest on Intelligent Agents and Multi-Agent Systems (MAS) in several fields such as Artificial Intelligence (AI), Software Engineering, Psychology, etc... Different problems can be solved in these fields by creating societies of agents that communicate with each other. Nevertheless, when the number of agents is large and the connectivity is extensive, the system suffers from overhead in the communication among agents due to the large number messages exchanged. This work addresses the search for an optimal communication topology to avoid these situations. This optimal topology is characterized by the use of a redirecting probability in the communication. The redirection of a communication is performed before the execution of the MAS. Once agents start the execution, the topology is fixed and remains unchanged. This characteristic is useful in those systems where a given topology can not be changed as, for example, in wired networks. On the other hand, in the proposed solution agents contain a local message discrimination process as a function of the sender of the message. Experiments show an important improvement in terms of a reduction in the number of iterations needed to solve the problem and also in the number of messages exchanged.

1 Introduction

A problem using Multi-Agent Systems appears when the system contains a large number of agents connected extensively. Agents need to stablish connections to interchange messages and realize a set of goals, or tasks, for which the system is

Antonio González-Pardo · Pablo Varona · David Camacho
· Francisco de Borja Rodriguez Ortiz
Departamento de Ingeniería Informática, Escuela Politécnica Superior,
Universidad Autónoma de Madrid
C/Francisco Tomás y Valiente 11, 28049 Madrid, Spain
e-mail: {antonio.gonzalez,pablo.varona,david.camacho,
f.rodriguez}@uam.es

M. Essaaidi et al. (Eds.): Intelligent Distributed Computing IV, SCI 315, pp. 131–141.
springerlink.com

designed. Depending on the topology of the network created and the number of agents in the system, overhead could appear due to the large number of messages interchanged, for example in broadcast topologies. To deal with the communication overhead, this paper studies the influence of the topology of the communication among agents. The main characteristic of the agents in the proposed solution is the discrimination of messages as a function of the sender in combination with an optimal topology that provides a faster propagation of the information without overhead the system.

The overhead problem in Multi-Agent Systems has been studied before and it is independent on the architecture used, [Jurasovic et al., 2006]. For example, in mobile agents it has been proposed the use of a dynamic agent distribution that reduces the communication between servers, [Jang and Agha, 2006]. This distribution is based on the idea of allocating agents with high communication rate between them, in the same server. This is useful because inter-node communication does not affect the overhead. But with this approach, a new problem appears which is how to select the agents with heavy traffic between them to migrate all of them into the same server. To solve this problem a new algorithm is proposed in [Miyata and Ishida, 2008].

Other works have addressed the discrimination of messages, [Sugawara and Lesser, 1993] and [Sugawara and Kurihara, 1998]. In these works agents learn what messages have high probability of being important based on rules created by each agent. Analysing the history of past inferences, agents create local rules that allow them to behave in a different way depending on the received message.

The discrimination technique described in this paper is a bio-inspired method based on the bursting activity of living neurons, [Szücs et al., 2003]. The bio-inspiration comes from the role of identity codes in neural systems which has been extensively with realistic models in [Latorre et al., 2006].

The main contribution of this paper is the search of an initial communication topology for a self-organizing Multi-Agent System. Static agents discriminate messages depending on the sender but their behaviour is not influenced by inference rules or learning procedures. This discrimination procedure has been applied previously in a new self-organizing neural network paradigm [Latorre et al., 2010] where the communication topology is dynamic and changes during the execution of the system. Although there are other approaches based on mobile agents ([Jang and Agha, 2006]), this work uses the optimization of the network topology to reduce the communication impact in the network. The communication topology suggested in this work deals with the communication overhead problem. This topology is based on a regular topology where each edge is redirected randomly according to a specific probability. Once the topology is created, it does not change during the execution of the system. Finally, our approach is illustrated by solving a jigsaw puzzle problem.

2 Description of the Model

This section describes the agent model, and the topology used to allow the optimal communication in the system.

2.1 Description of the Agent Model

Agents modelled in the system are static lightweight agents with a very low autonomy. However, most of the approaches of swarm and collective intelligence agents have the same behaviour and agents are indistinguishable. This work does not use this idea and each agent contains an unique identification that enables the differentiation between them. This characteristic is a key concept of the system because it will allow the discrimination of information as a function of the sender of the message.

In Multi-Agent Systems, agents have an incomplete information for solving the problem at hand. The piece of information contained in each agent is called, in this work, *Agent Information*, *[AIn]*, and the identification of each agent is called *Agent Identification*, *[AId]*. Both concepts compose the message that will be sent to the agents neighbourhood. Therefore, messages have the following structure $[AId|AIn]$. The first part of the message contains information that allows the identification of the sender, and the rest are data needed to solve the problem. Note that information stored in each part depends on the problem, *[AIn]* could contain much more information than *[AId]*. In these situations where Agent Information is large, it is very important the discrimination based on the recognition of the sender to avoid analyzing non relevant information.

The tasks performed by the agents will depend on the content of the messages received. That is, the behaviour of the agents will depend on who sends the information because depending on the sender, the message will be ignored or not. Nevertheless, the set of actions performed by the agents is always the same, there are not self-adaptation nor learning in its behaviour. Figure 1 shows the behaviour of agents.

Agents contain a temporal memory, called *local informational context*. Using this memory, agents can retain, during a certain period of time, a set of messages received by their neighbourhood. All received messages are stored in this memory, which means that it is not important whether the sender is recognized or not, because the message will be added to that memory in both cases. The fact that an agent does not recognize the sender of a message does not mean that any agent belonging to its neighbour will not recognize the sender. In the case where the received messages exceed the memory size, the oldest messages are replaced by the most recently ones. Finally, the information of the agent is added to this memory and all the content of the memory is sent.

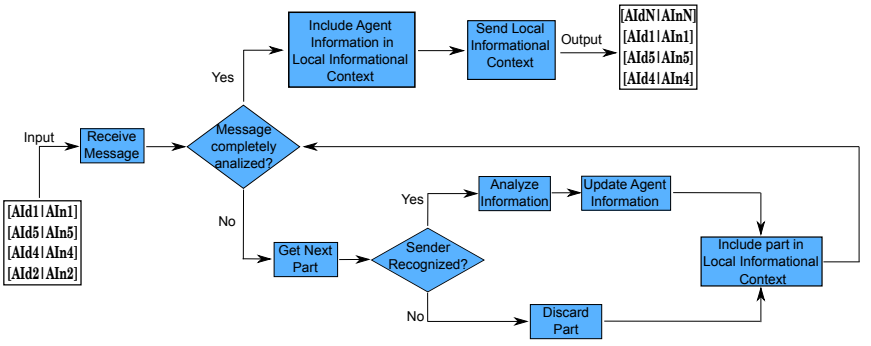


Fig. 1 Flow diagram that describes the behaviour of agent *N*. Each agent receives a set of messages, and starts analysing the sender of each message ($[AId]$). If the sender is not recognized, the message is discarded without analyzing the information contained ($[AIn]$). This discrimination procedure is really useful in environments where $[AIn]$ is bigger than $[AId]$ because agents does not analyze uninteresting $[AId]$ saving processing time.

2.2 Network Topology

The organization of any MAS can be analyzed from different perspectives which goes from hierarchical structures to completely random structures. However, it is necessary to provide some kind of structure to allow the interactions between agents. This organization can be studied from the point of view of a network topology.

In this work the topology selected to connect different agents is the Ring topology. This topology depends on a parameter named *connectivity degree (k)* that defines the number of connections that each node will have. Figure 2.a shows an illustrative example of a Ring Topology with $k = 1$, and 9 agents (nodes).

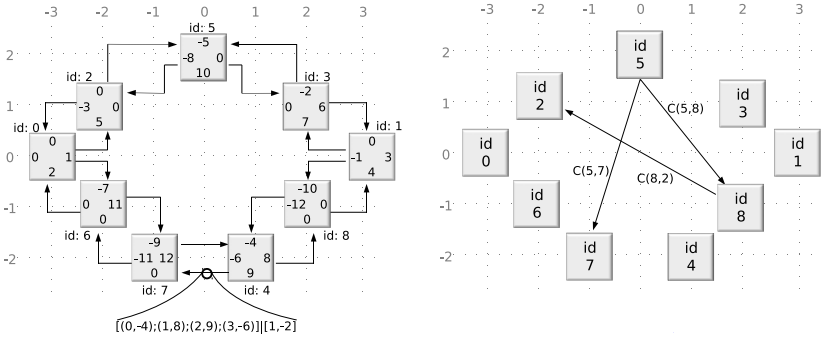


Fig. 2 Details of the communication topology. Figure on the left shows an example of a ring topology with 9 agents, and connectivity degree (k) 1. The figure on the right is a graphical representation of communication costs in the topology.

As it can be seen in Figure 2.a, given a particular connectivity degree, k, each node will be connected to $2 * k$ agents. Note also that the connections are unidirectional. This means that if node A is the origin of a connection to node B, node B is not able to use that connection to send a message to node A, and thus node B need another connection that goes from B to node A.

One of the problems of regular topologies is that the propagation of the messages in the network is very slow. This is produced due to the characteristic path length is very high. This metrics represent the mean length of the paths in the network.

For a specific value of k, each node will be connected to its $2 * k$ nearest nodes. Increasing the k of the topology, nodes will be highly clustered and the characteristic path will be decreased because the network will contain more edges. The minimum value for k is 1, in this situation each node will be connected to its 2 nearest nodes. On the other hand, the maximum connectivity corresponds with the broadcast mode, and this situation is produced when the connectivity degree is $n/2$, where n is the number of nodes in the network.

It is important to take into account that by increasing the connectivity degree of the topology, the system will converge in less iterations but the number of messages exchanged will also increase. Therefore, the value chosen for k is a trade off between the number of messages exchanged and the number of iterations to converge.

2.3 Searching an Optimal Communication Topology

As it was described in Section 2.2, the main problem in regular topologies is the value of the characteristic path length. From a communicational point of view, the lower the value of the characteristic path length is, the faster the messages are propagated.

In order to reduce the characteristic path length, a new parameter p is introduced in the system. This parameter is the probability of redirecting a connection. Given a specific value of p, each connection is analyzed and connections will be redirected with a probability p.

Using this new parameter, the connections are redirected to a randomly chosen node in the network, so the characteristic path length is reduced. Moreover, considering only the connectivity degree of the node (k, see Section 2.2) there is only one topology for each connectivity degree. Considering the parameter p, for each k and p fixed there are a family of topologies.

The probability of redirecting a connection was introduced by [Watts and Strogatz, 1998] in the definition of a Small World Network. It is important to note that for values of p close to 0 the network will have a regular topology, because few connections will be redirected. On the other hand, values close to 1 will generate Random Networks.

Small World Networks are characterized by having short path lengths between their nodes, which provides fast propagation of information, and high clustering rate that makes the network robust to attacks. This type of topology has been used to optimize the mutual information exchanged between nodes [Dominguez et al., 2009].

Finally when all connections have been treated by a fixed probability, agents start the execution and the topology created does not change.

3 Solving Puzzles Using an Optimal Communication Topology

There are many practical applications in which some agents need to know certain information about their environment in order to solve the problem. These applications can take advantage of the system described in this paper because the aim is to determine the optimal communication topology to solve the system by reducing the number of iterations and avoiding the overhead problem. Some examples of those applications, where our approach could be used, are scheduling problems, job shop problems, routing problems, etc.

The application selected to apply our approach tries to solve a puzzle where each piece needs to know which pieces match with its sides. Note that the neighbour of a piece does not necessarily contain pieces that matches with it. Using the technique described in this paper, each piece sends its information to the minimum number of agents to solve the puzzle in a reasonable amount of time.

Using the model to solve a puzzle, each agent will represent a single piece of the puzzle. Puzzles compose an image, and each piece contains a part of that global image. For that reason the agent information will be that part of the whole image contained in the piece.

3.1 Piece-Agent Codification

In order to model the shape of a side, each shape is represented by an integer value. For example, the border of the puzzle is represented as 0. With this approach, two pieces are compatible if the addition of the corresponding side values are equals 0 and none of these values are 0. This means that a piece with a side value X will match with the piece with value $-X$.

Using this representation, a message is relevant for an agent when the message contains, at least, one agent information compatible with any receiver agent sides. When a relevant message is received, the agent extracts the information about the compatible agent and memorizes that it must be connected to the sender through the corresponding side.

Each agent will be referred by its coordinates in a plane. That identification is unique because the agents are static and two agents cannot be located in the same place.

When an agent processes a relevant message, it updates its agent information, as shown in Figure 1. This means that agents need to delete from its agent information the side just matched. Each piece has four sides: upper side, right side, bottom side and left side. Those sides are identified by 0, 1, 2 and 3 respectively. In Figure 2.a, the piece with *AId* 4, which is located in $(1,-2)$, sends the message *[(0,-4);(1,8);(2,9);(3,-6)]|[1,-2]* to its neighbour.

3.2 Piece-Agent Communication

In real world, the communication between two partners has a cost which can be expressed in terms of monetary, time or performance cost. In this work, the cost of sending a message is proportional to the distance between the sender and the receiver. The cost of a communication is defined by equation 1.

$$C(i,j) = \lfloor \frac{EuclideanDistance(i,j)}{\min(EuclideanDistance(l,m))} \rfloor \forall l,m \mid l \neq m \quad (1)$$

The cost of a communication between two agents is the first positive number that exceeds the euclidean distance between both agents, normalized by the minimum distance in the system. Costs resulting from Equation 1, describe the number of iterations that a message will take to go from the sender to the receiver. The minimum cost is 1 and means, that receiver will have the message available in the next iteration. Figure 2.b shows a representation of the cost in the network.

4 Experimental Results

This section describes the different experiments carried out in this work. As it was described in section 3, the model has been adapted to solve a puzzle as an illustrative example.

In order to analyse the impact of the probability on the performance of the system, the original topology will be tested with with 21 different values of this probability. The values go from 0 to 1 with increases of 0.05.

All the experiments carried out in this work are based in a puzzle with 100 pieces, and all the shapes of the pieces are unique. This means that each piece will match, exactly, with one piece for each side. The algorithm initializes the agents and creates the topology defined by a fixed probability. Finally, agents are executed to solve the puzzle.

Figure 3 shows the iterations taken by the system to solve the puzzle. Those charts represents the number of iterations taken by the system to solve the problem for a specific probability value. The value of k is 3 for Figure 4.A, while the rest of figures corresponds to executions with $k = 8$.

In order to analyze how the memory size affects the performance of the system in terms of iterations taken to solve the puzzle, four experiments have been carried out. All experiments try to solve a puzzle with 100 pieces, and all pieces have $k = 3$. The memory size changes in each experiment, and it takes values 3, 8, 15 and 30. Figure 4 shows the performance of these experiments. From this figure, it is deduced that a larger memory provides an important improvement in the system with lower values of probability.

Furthermore, there is a limit probability from which there is no important improvement in the system. This limit is located between 0.3 and 0.4 and means that with this probability the characteristic path is very low.

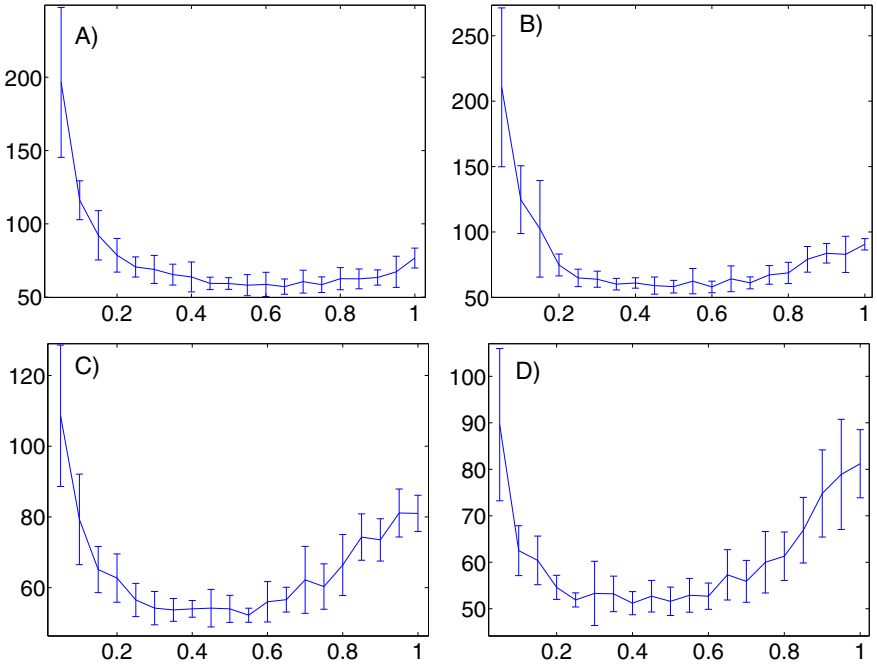


Fig. 3 Iterations per probability resulting from the execution of the system with different values for the memory and the connectivity degree of the topology. The system is composed by 100 agents that represent a puzzle with 100 pieces. For a specific probability, the system has been executed 10 times. Y-axis represents the number of iterations taken by the program, while X-axis shows the value of probability p. Figure A shows the performance of the system with memory 3 and $k = 6$. Figure B represents the system with memory 8 and $k = 3$. Finally, performance of the system with $k = 3$ and memory 15 and 30 is shown in Figures C and D.

Finally, as it was stated previously, the memory size affects the performance of the system. Nevertheless, the performance of a system with size memory 15 is very similar to the performance with memory 30. This fact suggest the existence of a limit in the memory size. This limit must be studied in future works because it is important to build a system that optimizes resources.

In order to measure the overhead problem in the system, the number of messages sent by the agents is compared with the a broadcast situation. The broadcast mode guarantees a solution, because each agent will received a message from the rest but the cost of sending a message affects the number of iterations taken to achieve the solution. Apart from this, designing a broadcast mode in a real system, for example, communication between hosts, it is not useful because to build a system with these characteristic could be very expensive.

Figure 5 shows the performance of the system taken into account the total number of messages sent. Although these messages could have different length, (because if

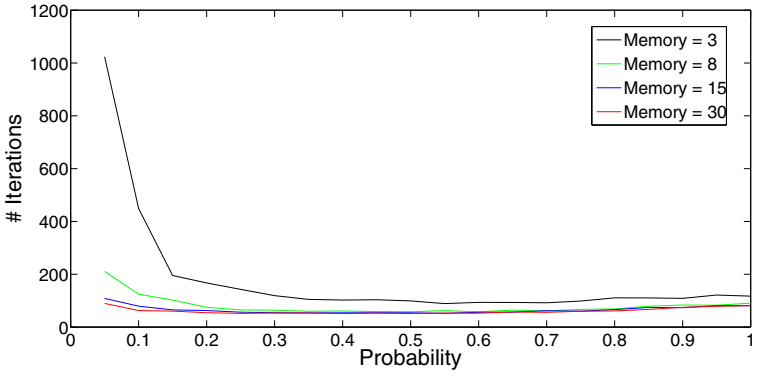


Fig. 4 Performance of the system, in terms of number of iterations taken to solve the puzzle, with memory 3, 8, 15 and 30. All executions try to solve a puzzle with 100 pieces, and each agent has $k = 3$.

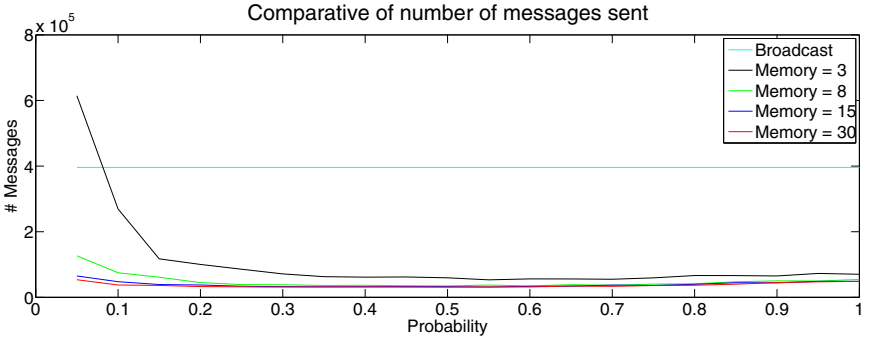


Fig. 5 Difference between the system and the broadcast. The connectivity degree of the topology in the broadcast mode is 50 ($n/2$), in the rest of executions the value of k is 3.

a piece has a side matched, the information belonging to that side is not sent) in these experiments, the suggestion that all messages have the same length is taken into account. This is not strange because, in this problem, the difference between the largest message and the smaller message is not relevant. Nevertheless, a more specific study on the total number of messages is required.

From the four experiments described in this section, the only execution that exceeds the number of messages sent by the broadcast mode is with memory 3 and lower probability (0.05). This result is expected because in this situation, the

topology is very similar to the regular ring (with high characteristic paths) and the message size is very small. Nevertheless, with a little change in the probability (from 0.05 to 0.1) the number of sent messages is reduced around 56%. The rest of executions, independently of the probability, improve the number of messages sent in the broadcast situation.

5 Conclusions

This paper studies the problem of communication overhead in a network of agents analyzing the topology of the communication. The characteristics of this topology try to meet the following goals. On one hand, to reduce the geodesic path of the network in order to propagate messages in fewer iterations. On the other hand, to reduce the number of messages exchanged among agents and, in that way, to avoid the overhead problem.

This work modifies the regular topology with a specific probability, as a Small World Network. Nevertheless, it is important to notice that this redirection of communication channels is only performed once and when agents start the execution the topology is fixed. This concept is similar to a computer network where, once there is a connection between two devices or sub-networks, the connection is not redirected because the wires are physically inaccessible or the cost of redirecting is expensive.

The results show that modifying the probability of redirection, there are important improvements in the number of iterations needed to solve the problem (as compared to the regular topology). The value of the optimal probability depends on the connectivity degree, Figure 3 shows that for $k = 3$ the optimal probability is around 0.25 and 0.45, and for $k = 6$ the optimal probability is located between 0.55 and 0.65. As there is no direct relation between the connectivity degree and the probability, a deeper study is needed.

The memory of the agents plays an important role in the performance. Figure 4 shows that with larger values for the memory, the system solves the problem in fewer iterations. Taking into account the number of messages sent in the system, Figure 5 shows that with minor modifications in the communications (low probability), the number of messages sent is less than the number of messages sent in the broadcast situation.

Finally, the experimental results show how our approach allows to reduce significantly the communication overhead in a MAS by modifying the regular topology of the agent communication network.

Acknowledgements. This work has been partially supported by the Spanish Ministry of Science and Innovation under TIN 2007-65989, TIN 2007-64718 and MICINN BFU2009-08473.

References

[Dominguez et al., 2009] Dominguez, D., Gonzalez, M., Serrano, E., Rodriguez, F.B.: Structured information in small-world neural networks. Physical Review E (Statistical, Nonlinear, and Soft Matter Physics) 79(2), 021909 (2009)

[Jang and Agha, 2006] Jang, M.-W., Agha, G.: Agent framework services to reduce agent communication overhead in large-scale agent-based simulations. Simulation Modelling Practice and Theory 14(6), 679–694 (2006)

[Jurasovic et al., 2006] Jurasovic, K., Jezic, G., Kusek, M.: A performance analysis of multi-agent systems. ITSSA 1(4), 335–342 (2006)

[Latorre et al., 2006] Latorre, R., Rodríguez, F.B., Varona, P.: Neural signatures: multiple coding in spiking-bursting cells. Biological Cybernetics 95, 169–183 (2006)

[Latorre et al., 2010] Latorre, R., Rodríguez, F.B., Varona, P.: Signature neural networks: Definition and application to jigsaw puzzle solving. IEEE Transaction on Neural Networks (2010) (page to appear)

[Miyata and Ishida, 2008] Miyata, N., Ishida, T.: Community-based load balancing for massively multi-agent systems. In: Jamali, N., Scerri, P., Sugawara, T. (eds.) MMAS 2006, LSMAS 2006, and CCMMS 2007. LNCS (LNAI), vol. 5043, pp. 28–42. Springer, Heidelberg (2008)

[Sugawara and Kurihara, 1998] Sugawara, T., Kurihara, S.: Learning message-related coordination control in multiagent systems. In: Selected Papers from the 4th Australian Workshop on Distributed Artificial Intelligence, Multi-Agent Systems, pp. 29–44. Springer, Heidelberg (1998)

[Sugawara and Lesser, 1993] Sugawara, T., Lesser, V.: Learning coordination plans in distributed problem-solving environments. Technical report. In: Twelfth International Workshop on Distributed Artificial Intelligence (1993)

[Szücs et al., 2003] Szücs, A., Pinto, R., Rabinovich, M., Abarbanel, H., Selverston, A.: Synaptic modulation of the interspike interval signatures of bursting pyloric neurons. Journal of neurophysiology 89(3), 1363–1377 (2003)

[Watts and Strogatz, 1998] Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world' networks. Nature 393(6684), 440–442 (1998)

B. Comm. by identity discrimination in bio-inspired MAS

- **Title**: Communication by identity in bio-inspired multi-agent systems
- **Authors**: Antonio Gonzalez-Pardo, Pablo Varona, David Camacho, Francisco B. Rodriguez Ortiz
- **Publication**: International Journal Concurrency and Computation: Practice & Experience. 2011. ENVIADO
- **Editorial**: John Willey
- **ISSN**: 1532-0626
- **Índices de calidad**: **JCR (2009)**. Índice de impacto= 1.004; posición (COMPUTER SCIENCE, THEORY & METHODS) = 51/92 (**Q3**)

- **Abstract**:Network communications have been widely studied in the last decades in different research fields: Artificial Intelligence, Computer Science, Biology, Medicine and Psychology amongst others. Some important efforts have been carried out to analyze communication features such as overhead, connectivity, or communication protocols in these areas from their own perspectives. When this problem is restricted to Intelligent Agents or Multi-Agent Systems (MAS), the networks are built by a set of interconnected agents that can be software or hardware. In MAS, communication optimization is used to improve the overall performance of the system by reducing the information sharing (i.e. number of messages or message size) between the agents. The paper analyzes a scaled free topology network of agents to solve a sorting problem. These agents use their local information as well as a bio-inspired identity discrimination process, to select only those messages that are relevant for each agent to solve the global problem. This paper shows a comprehensive study of some essential parameters (memory information size and reconnection probability of a connection) in an agent network, and shows how they can be set to obtain a better performance in the system. The experiments show that this strategy contributes to reduce the number of iterations needed to solve the problem.

CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2010; **00**:1–19
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe

Communication by identity discrimination in bio-inspired multi-agent systems

Antonio González-Pardo, Pablo Varona,
David Camacho, Francisco de Borja Rodriguez Ortiz

Departamento de Ingeniería Informática. Escuela Politécnica Superior.
Universidad Autónoma de Madrid.
C/Francisco Tomás y Valiente 11, 28049 Madrid, Spain.
{*antonio.gonzalez, pablo.varona, david.camacho, f.rodriguez*}*@uam.es*

SUMMARY

Network communications have been widely studied in the last decades in different research fields: Artificial Intelligence, Computer Science, Biology, Medicine and Psychology amongst others. Some important efforts have been carried out to analyze communication features such as overhead, connectivity, or communication protocols in these areas from their own perspectives. When this problem is restricted to Intelligent Agents or Multi-Agent Systems (MAS), the networks are built by a set of interconnected agents that can be software or hardware. In MAS, communication optimization is used to improve the overall performance of the system by reducing the information sharing (i.e. number of messages or message size) between the agents. The paper analyzes a scaled free topology network of agents to solve a sorting problem. These agents use their local information as well as a bio-inspired identity discrimination process, to select only those messages that are relevant for each agent to solve the global problem. This paper shows a comprehensive study of some essential parameters (memory information size and reconnection probability of a connection) in an agent network, and shows how they can be set to obtain a better performance in the system. The experiments show that this strategy contributes to reduce the number of iterations needed to solve the problem. Copyright © 2010 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: Multi-Agent Systems; Network Topologies; Signature Neural Networks

1. INTRODUCTION

Communication in computer networks is always one of the most critical and essential feature that must be optimized for a good performance of the system. Important efforts, and relevant contributions, have been obtained in areas like Physics [1], NeuroScience [2], Biology [3], Computer Science [4], Artificial Intelligence [5], Intelligent Agents (IA) or Multi-Agent Systems (MAS) [6] amongst others. Different characteristics related to this problem like the communication overhead, connectivity, or communication protocols, have been carefully studied and analyzed to understand correctly these processes [7, 8].

In Intelligent Agents and Multi-Agent Systems an important problem appears when the system contains a large number of agents connected extensively. Agents need to stablish connections to interchange messages and realize a set of goals, or tasks, for which the system is designed. Depending on the topology of the network and the number of agents in the system, overhead problems (in both agents and connections) could appear due to the large number of messages interchanged (i.e. in broadcast topologies). This work analysis the effect of an agent network topology when it is tried to optimize the performance of the system in a sorting problem. This type of problems are very interesting from the computational point of view because their aim is to perform a global multi-dimensional sorting but no global sorting criteria is defined. This work is applied in a well known sorting problem called jigsaw puzzle.

Solving a jigsaw puzzle is a well known problem that has been faced using different approaches. Some works solve puzzles based on the shape of the pieces [9, 10, 11] or based on the image contained in the pieces [12, 13, 14]. And there are different algorithms that solve the problem using different computational strategies such as shape matching [15], image merging [15], neural networks [16] or genetic algorithms [17].

This work tries to solve a blind puzzle using a bio-inspired multi-agent system. Blind puzzles are those that does not compose any image and the result is a uniform colour picture. As there are not colour differences the way to solve these puzzles are based on the shape of the pieces.

One of the main characteristic of the agents in our solution is the utilization of a discrimination policy of messages which uses information about the sender to decide whether the message should be analyzed or not. This discrimination policy combined with an optimal communication topology provide a reduction in the resolution time needed to solve the problem.

Discrimination techniques in MAS has been previously studied to solve the communication problems produced in cases where the system is composed by a large number of agents. Previous works have addressed the discrimination of information, [18] and [19]. In these works agents learn what messages have high probability of being important based on rules created by each agent. Analysing the history of past inferences, agents create local rules that allow them to behave in a different way depending on the received message. For instance [20] shows how a discrimination process reduces the overhead of the agents, this is due to agents does not analyze all received messages but only those that contain relevant information. This work also shows how a redirection probability to reconnect the network can be used to reduce the characteristic length of the network, reducing the path taken by the system to send a message from any agent to other. Both characteristics provide an important reduction of the communication overhead in the system. In this previous work static agents discriminate messages depending on the sender but their behaviour is not influenced by inference rules or learning procedures. This discrimination procedure has been applied previously in a new self-organizing neural network paradigm [21] where the communication topology is dynamic and changes during the execution of the system.

DOI: 10.1002/cpe

This paper uses a power law distribution applied to the agents connectivity degree. This fact generates more realistic networks similar to Small World [22, 23]. This allows to set one of the previous parameters studied (k, or the connectivity degree of each agent) and makes the study of the system easier. This work studies the influence of the reconnection probability and the memory size of the agents in the performance of the system. An exhaustive study about these two parameters is carried out with the aim of determining the optimal values that improve the performance of the system. These new set of experiments have been applied to solve jigsaw puzzle problems.

The rest of the paper is structured as follows. Section 2 provides a brief description about the biological concepts to understand the discrimination procedure used by our agent network. Section 3 shows the agent model, the agent topology and describes what are the main characteristics to be studied in our network of agents to search for an optimal communication topology. Section 4 describes how the agent network has been dedicated to a particular problem, to solve jigsaw puzzles, in this section the codification of a puzzle piece, and how the information of a piece is shared among agents is described. Sections 5, and 6, shows the experimental setup of the experiments, and provides a detailed empirical analysis of the results obtained. Finally, Section 7 describes the main conclusions of this work.

2. BASICS ON SIGNATURE NEURAL NETWORKS

The identity discrimination technique described in this paper is a bio-inspired method based on recent researches that study the bursting activity of living neurons, [24, 25]. The bio-inspiration comes from the role of identity codes in neural systems which has been extensively analyzed with realistic models in [26]. This section makes a general overview about the basic concepts needed to understand the biological discrimination procedure.

Information in neural networks is encoded as *action potentials* also known as spikes. A single spike is a fast increase and decrease of membrane potential. Neurons usually react with a sequence of action potentials in response to a stimulus. Some neurons produce bursting activity. During bursting, a neuron repeatedly fires discrete groups or bursts of spikes. Each burst is followed by a period of quiescence before the next burst occurs, see Figure 1.

Recent electrophysiological experiments have shown that some neurons display robust and neuron-specific interspike intervals (ISI) in their bursting activity, which have been named as *neural signatures* (see Figure 1). These observations have pointed out the possibility that these neurons have multiple codes regarding the who and the what of the information. In addition living neurons are known have several different time scales in their intrinsic dynamics that endow them with transient memory mechanisms for information processing.

ISIs are very important because they are cell-specific and reproducible. ISIs identify unequivocally the neuron that has generated this burst behaviour [24, 27]. This means that different neurons have different bursting activity and the ISIs have different temporal distribution. This fact

DOI: 10.1002/cpe

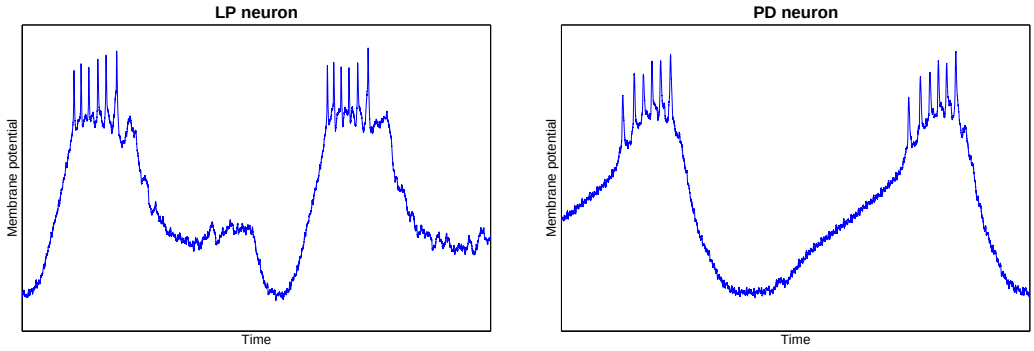


Figure 1. This figure shows the neural signatures of two different neurons located in the stomatogastric ganglion of crustaceans. Left figure corresponds with the bursting activity of the LP neuron while the right figure shows the activity of the PD neuron. Both figures are different and identify the neuron that generate each activity. Each burst have a duration of 1 sec, and a potential difference around 20 mV.

is represented in Figure 1 where the activity of two different neurons are represented. This specific temporal distribution of the ISIs is called *neural signature*. The identification of the neuron that generates the bursting activity through its neural signature is used in complex neural networks to perform input discrimination and thus perform multi-tasking in the network.

3. DESCRIPTION OF THE MULTI-AGENT MODEL

The Multi-Agent System designed is based on a Swarm approach [28]. These kind of systems are based on agents with characteristics such as, low proactiviness, low autonomy or a limited reasoning capabilities. Swarm-based approaches have been used traditionally for simulation and optimization problems. Our work uses this approach to design complex network of agents where the communication process among them is the most important characteristic to be analyzed. This section describes the agent model and how they implement the bio-inspired discrimination process, the agent topology designed, and finally describes what are the main characteristics that will be studied in our network to search for an optimal communication topology.

3.1. Description of the agent model

The agents modelled in the system can be described as static lightweight agents with a very low autonomy and proactiviness. Most of the approaches based on Swarm and Collective Intelligence implements agents that are indistinguishable, and all of them have the same behaviour (i.e. ant colonies or particle swarm optimization systems). This work uses a different approach where each agent contains an unique identification that allows the differentiation between them. This characteristic is a key concept in the system, because the discrimination of information is related to the identification of the sender of the message.

In any standard Multi-Agent System, agents only have incomplete (local) information to solve the whole problem. In this work, the local information stored by any agent is called *Agent Information*, $[AIn]$, and it contains the information needed to perform the tasks for which the

DOI: 10.1002/cpe

system was designed. Moreover, each agent is identified by the *Agent Identification*, $[AId]$. Both fields, compose a message registry $[AId|AIn]$ that conform the message that will be sent to the agents neighbourhood, the complete message is composed by a set of non empty message registries.

The performance of the whole system is related to the amount of information stored in $[AIn]$ field, that is the more extensive $[AIn]$ is, the more improvement is taken by the system. Our approach improves the global performance using the discrimination process, which avoids to analyze unnecessary Agent Informations that are not relevant for the receiver. The tasks performed by the agents will depend on the sender of the message registries, that is, the behaviour of the agents will depend on who sends the information. Depending on the sender, the $[AIn]$ part of the message registry will be ignored or not. Therefore, this discrimination procedure is really useful in those domains where $[AIn]$ is much more larger that $[AId]$ because agents does not analyze uninteresting $[AIn]$ saving computational effort. Nevertheless, the set of actions performed by the agents is always the same, there is neither self-adaptation nor learning in its behaviour. Figure 2 shows the behaviour of any agent in the system.

The information and the structure of $[AId]$ and $[AIn]$ fields are dependent on the problem modelled, in this work the application domain is a jigsaw puzzle, the description of the domain and descriptions for these fields are given in Section 4.

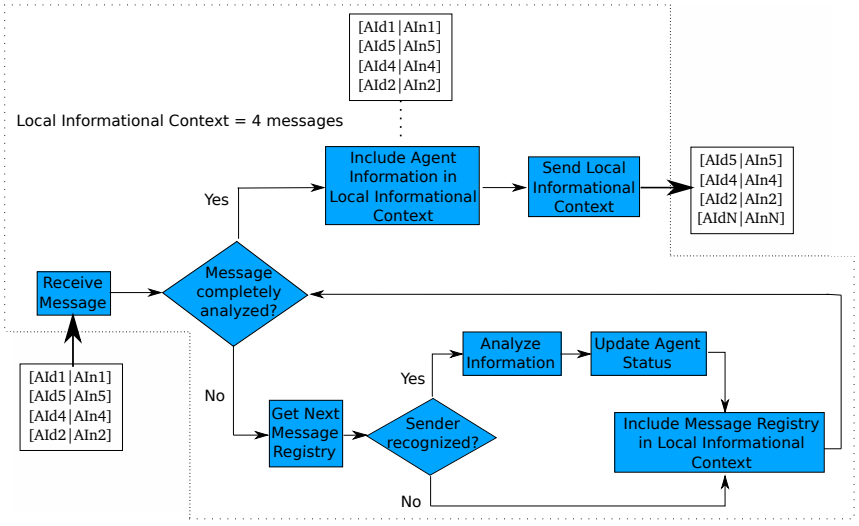


Figure 2. Flow diagram that describes the behaviour of a particular agent N in the network. Each agent receives a message composed by a set of message registries. The agent starts analysing the sender of each message registry ($[AId]$). If the sender is not recognized, the message registry is discarded without analyzing the information contained ($[AIn]$).

As it was described in Section 2, the strength of a discrimination techniques is increased if some a temporal memory is provided. The agent model uses a limited temporal memory, named *local informational context*. Using this memory, the agents can store, during a period of time, a set of message registries received by their neighbourhood. All received message registries are stored in

DOI: 10.1002/cpe

this memory. This means that it is not relevant if the sender has been recognized or not by the receiver agent, the message will be always stored in the memory. If the sender of the message is not recognized for a particular agent, it could be analyzed by other agents in the network because the content of the *local informational context* will be sent to the other agents in the neighbourhood. If the number of message registries received by a particular agent exceeds the memory size for the temporal memory, the oldest message registry are replaced by the most recently ones.

3.2. Agent Topology

The organization of any MAS can be analyzed from different perspectives that go from hierarchical structures to completely random structures. However, any type of agent organization requires some kind of structure to allow the interactions between them (i.e. horizontal, vertical, hybrid architectures). One of the possible perspectives that can be used to analyze the organization of a set of agents is based on the network topology. Our approach studies a set of topologies in order to select the parameters that optimize the performance for a final topology.

In this work the initial topology selected to connect the agents is the ring topology. In [20], the network topology was configured using a parameter named *connectivity degree (k)* was used to define the number of connections that each node in the network has. Using this parameter all the nodes have the same number of output connections. In this work the connectivity of each node depends on a Power-law distribution, this fact is based on some works that suggest the Internet topology use a Small World structure with a power-law distribution in the connectivity degree of each node [22, 23]. Any power-law function is described by Equation 1.

$$p(x) = \gamma x^{-\mu} \tag{1}$$

This function has two factors, μ and γ. The exponent μ is fixed to 1.22 because [23] shows that Internet topology uses a power-law distribution with this exponent. The value of γ factor is dependent on the number of agents in the system. In the case where there are N agents in the system, γ factor is calculated using:

$$\gamma = \frac{1}{\sum_{i=2}^{N} i^{-\mu}} \tag{2}$$

Figure 3(b) shows the power law function for a puzzle with 12 pieces. In this figure the number of connections that a piece will have is represented over the probability of having such number of connections. As can be seen, using this distribution the system will have many agents with low connectivity and few agents highly connected. A topology composed by 12 agents and a power law distribution in the connectivity degree of each node is shown in Figure 3(a).

The connections used in this work are unidirectional. This means that if node *A* is the source of a connection to node *B*, node *B* is not able to use that connection to send a message to node *A*, and

Prepared using cpeauth.cls
DOI: 10.1002/cpe

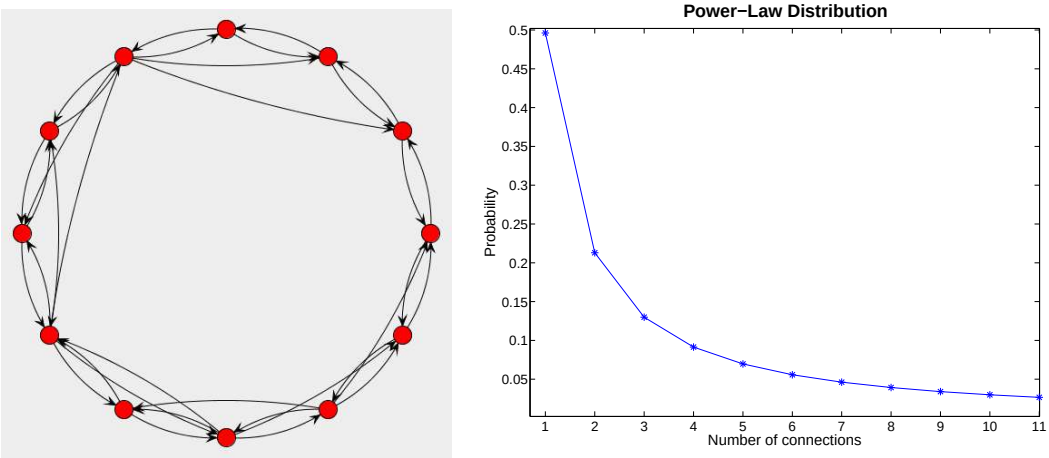


Figure 3. Details of the departing communication topology used in work. The left figure shows an example of an initial topology with 12 agents. The power law distribution that defines the number of connections for each agent is shown in right figure. This distribution makes that agents with few connections have more probability that those with high connectivity.

thus node *B* need another connection to go to *A*.

One of most serious problems in regular topologies is that the propagation of the messages in the network can be very slow. This is produced due to the *characteristic path length* is very high. The characteristic path length is a metric that represents the average length of the paths in the network. This value describes the average number of jumps that the message will need to arrive to its destination, for that reason high characteristic path lengths means that sending a message from any agent to any other agent will take more time.

Connecting agents directly to their nearest neighbours generates networks with high characteristic path lengths. In order to reduce these lengths the links must be connected to nodes sufficiently far away. However, if the connected nodes are too separated we have the same problem to send a message between two nearest neighbours. Our approach shows how the network topology can be optimized searching for the best set of interconnections that allows to reduce the characteristic path length in the network.

3.3. Searching an optimal communication topology

Previous section shows how the characteristic path length is an esential characteristic that must be considered and optimized in regular topologies or other complex topologies (i.e. power-law based like Internet or Social Networks) to improve their performance. From a communication process perspective, the lower the value of the characteristic path length is, the faster the messages will be propagated in the network. Therefore, if the characteristic path length must be reduced, a new parameter p is introduced in the system. This parameter represents the probability to redirect a connection. Given a specific value of p, each connection is analyzed and any connection in the network will be redirected using this probability p. Know every connection is redirected to a randomly selected agent in the network, the new connection is analyzed to evalute how it affects to

DOI: 10.1002/cpe

the characteristic path length. This random redirection is performed taking into account that neither parallel edges nor auto connections are allowed.

The probability (p) of redirecting a connection was introduced by Watts and Strogatz in the definition of a Small World Network [29]. In this kind of networks when p is closer to 0 the network will be very similar to the initial topology, because few connections will be redirected, and when p is closer to 1 networks with similar characteristics as Random Networks will be generated. The Small World Networks have short path lengths between their nodes, which provides fast propagation of information, and high clustering rate that makes the network robust to attacks. This type of topology has been used in [30] to optimize the mutual information exchanged between nodes. Finally, in this work once all the connections have been assigned with a fixed probability, the agents can start with the execution of the problem and the topology created cannot be modified.

4. SOLVING PUZZLES USING AN OPTIMAL COMMUNICATION TOPOLOGY

In most of practical distributed applications it is usually necessary to control, or optimize, the computational effort employed to solve a problem. In these applications one of the relevant parameters to be considered is the reduction of the messages sent between nodes or agents, or control the amount of information that should be analysed to find a solution. Those applications where our approach could improve the performance of the system are related to multi-sorting problem. Some examples of those applications are scheduling problems, job shop problems, routing problems, etc.[31]. The application selected here, solving jigsaw puzzles, can be considered as a practical example where previous (and others non mentioned) techniques and algorithms could be applied. Any jigsaw puzzle can be represented as a two dimensional picture that has been divided in different fragments that later are unsorted. The goal of any algorithm is to rebuild the original picture from those fragments.

In our approach, each agent will represent a single piece of the puzzle. For that reason the agent information will be the part of the whole image contained in the piece. In the case of a basic puzzle, this information could be the colours of the pieces, or the description of the part of the figure contained, but as this work tries to solve a blind puzzle, the information is the different shapes of the piece. Each agent will be connected to some other agents, and they will interchange messages trying to find which pieces match with them. Note that the neighbour of a piece does not necessarily contain pieces that match with it. Using the technique described in this paper, each piece sends its information to the minimum number of agents to solve the puzzle reducing the computational effort.

4.1. Piece-agent codification

As it has been described in Section 3, in this work agents are composed by *Agent Information* and *Agent Identification*. Basically, *Agent Information* contains the local information known by the agent, while the *Agent Identification* contains the information that allows to identify the agent. In order to identify the information stored in both fields, it is important to take into account that *Agent Identification* must answer the question ”What agent sends the information?” while

DOI: 10.1002/cpe

Agent Information must answer the question ”What is the agent knowledge?”. Both characteristics compose a message registry that will be part of the messages.

This work is based on solving a blind puzzle. In these type of puzzles, there is not any picture and the way to solve them is taking into account the shape of the different pieces. In order to model the shape of a side, each shape is represented by an integer value. Any non-border side is assigned a non-zero integer value, while the border of the puzzle is represented as 0. Complementary sides have opposite values, this means that a piece with a side value X will match with the piece with value $-X$. Using this approach, two pieces are compatible if the addition of the corresponding side values is equal to 0 and none of these values are 0. Examples of this matching technique is shown in Figure 4(b).

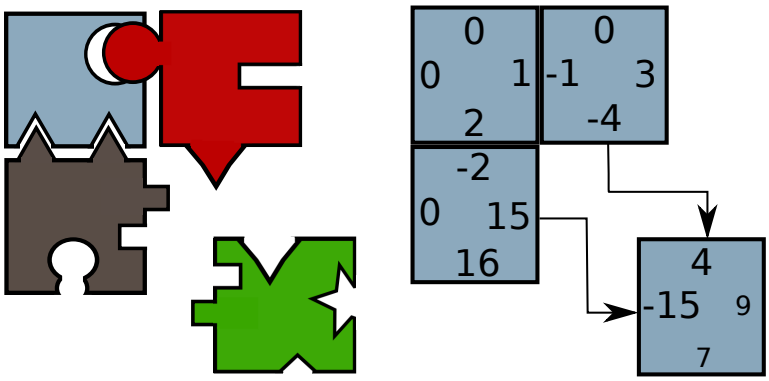


Figure 4. The left figure shows four real pieces of a puzzle while the codification of those pieces in the system is shown in the right figure.

As it was described in Section 3, agents contains two fields called *Agent Information* and *Agent Identification*. The first one contains the incomplete local information of the problem known by them, while *Agent Identification* contains information that unambiguously identifies each one. In this work, every agent represents a piece of the puzzle and their aim is to discover the position of the pieces that match them. For that reason, the *Agent information* is the position of the pieces because the agents knowledge are the coordinates of their pieces in a two-dimensional circle. Each piece is characterized by the shape of its sides. In this work, each side matches exactly with one piece, there not any piece with a side that could match with two different pieces. Therefore, the shape of the sides identify unequivocally the agent and composed the *Agent Identification*. Each piece has four sides: upper side, right side, bottom side and left side that are identified by 0, 1, 2 and 3 respectively. The *[AId]* field will be four pairs of values with the structure *(side, value)*. For example, in the right panel of Figure 4 the piece whose upper side, right side, bottom side and left side has the values 0, 3, -4 and -1 respectively, its *Agent Identification* will be $[(0,0);(1,3);(2,-4);(3,-1)]$.

Agent Information and *Agent Identification* compose a message registry that will be sent into a message. Receiver agent will analyze whether the message is relevant or not. A message is relevant for an agent when the message contains one, or more, message registries with agent informations compatible with any of the receiver agent sides. When a relevant message is received, the agent

DOI: 10.1002/cpe

extracts the information about the compatible agent and memorizes that it must be connected to the piece located in *Agent Information* field through the corresponding side.

Finally, when an agent processes a relevant message, it updates its status, as shown in Figure 2. The status of any agent is the number of sides that are not matched yet. Then the message registry is included in the local informational context, and when all received messages have been analyzed, the content of this memory is sent to agents neighbour.

4.2. Piece-agent communication

In real world, the communication between two partners has a cost which can be expressed based on time, performance or simply budget costs. In this work, the cost of sending a message is proportional to the physical distance between the sender and the receiver, in the network.

Let be $D(i,j)$ a particular distance between agent i and agent j defined by equation 3.

$$D(i,j) = \beta \frac{EuclideanDistance(i,j)}{minDistance} \tag{3}$$

Agents are located forming a circle. In two-dimensional systems, this circle has its center in position $(0,0)$ and it has a radius of 1. With this organization, the maximum distance between two agents is obtained in the case where both agents are separated by a diameter. In this case the distance is 2, as this value is very low, two different factors (β and $minDistance$) are included in Equation 3 to scale the result of the function.

β is a constant factor whose aim is to penalize shortcuts in the networks. Although shortcuts reduce the characteristic path length, it should be more expensive to send a message through them than trough the neighbours. The aim is to take the advantages of creating some shortcuts but without the design of a network composed only by shortcuts. From all the experiments carried out, the value of this factor is fixed in 15. The second factor, $minDistance$, depends on the number of agents that compose the puzzle. Given a puzzle with N pieces, the value of $minDistance$ is fixed using Equation 4.

$$minDistance = \sqrt{(1-\cos\alpha)^2 + (-\sin\alpha)^2} \tag{4}$$

Where α is the angle that separate two consecutive agent in the network and it is defined by $\alpha = \frac{2\pi}{N}$.

Once the distance between two agents is computed, the cost function used in this work is determined by equation 5

$$C(i,j) = \begin{cases} \lceil D(i,j) \rceil & \text{if } D(i,j) - [D(i,j)] \geqslant 0.5 \\ \lfloor D(i,j) \rfloor & \text{if } D(i,j) - [D(i,j)] < 0.5 \end{cases} \tag{5}$$

DOI: 10.1002/cpe

Where $[X]$ is the integer part of X. For example, $[4.05] = 4$ and $[4.9] = 4$.

Costs resulting from Equation 5, describe the number of iterations that a message will take to go from the sender to the receiver. Figure 5 shows a representation of the cost in the network.

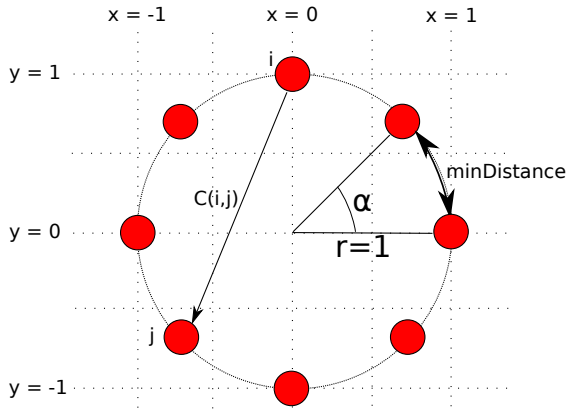


Figure 5. This figure is a graphical representation of the different variables used to compute the cost of a communication. α depends on the number of agents in the puzzle. In this case, α has a value of 0.52 radians. With this value and Equation 4, the *minDistance* of this network is 0.514.

5. EXPERIMENTAL SETUP

In this section a description about the experiments carried out in this work is provided. The aim of these experiments is the study of the different parameters and how their values affect to the performance of the system.

Although in the previous work [20] the system was composed by three parameters, in this work, only two of them are used (see Section 3.2 for more details). These two parameters are called *Memory* and *Probability of redirection*.

Memory is the name given to the *local informational context*. The value of this parameter defines the maximum number of message registries that each agent will memorize during a period of time (see Section 3.1). The size of this memory is very important because the bigger this memory is, more pieces will be memorized. Note that the content of this memory will be sent to the neighbour of the agent. For that reason in the case where the memory size is equal or bigger than the number of pieces of the puzzle, the whole puzzle will be contained in the message and the puzzle will be solved in few iterations. The number of iterations is not 1 because there is a warm period at the beginning of the system in which the different pieces are included in the memory of the rest of pieces. On the other hand, the smaller this memory is, more iterations will be taken to solve the puzzle.

The *probability of redirection*, *p*, is not a variable of the system but it has been introduced to modify the regular topology by introducing short-cuts in the agent topology. Short-cuts are needed to decrease the *characteristic path length* and reduce the average of the iterations needed to send a

Prepared using cpeauth.cls
DOI: 10.1002/cpe

message through the network. Although, the creation of short-cuts reduces the characteristic path length, only few number short-cuts are needed to optimize the metric. This is because there is a limit in the number of short-cuts in a network from which the *characteristic path length* is not reduced. For that reason, the cost function described in Section 4.2 makes short-cuts useful in the case where few short-cuts were created and makes the system works worse in the case whether the system has a high number of short-cuts.

In order to study the influence of these parameters, the minimum and the maximum value and the increase of the parameter need to be fixed. With these three values a subset of the variable domain is analyzed. For all the experiments carried out in this work, the minimum value of p is 0, the maximum is 1 and the increase of the probability is 0.1. With this configuration 11 different values for the redirection probability are used. The values for the agent memory are no fixed, because those values depends on the size of the puzzle. For puzzles with 100 agents, the maximum memory will be 100 (because a higher value will not be useful) but for a puzzle with 1000 agents, the maximum memory is 1000. Table I shows the values of those variables for all the experiments. One experiment consist on the execution of the system for all possible combination of p and *Memory* values. For example, for a given puzzle 11 values are defined for p (0, 0.1, 0.2, $\ldots$, 0.9 and 1) and 3 different values are used for the *Memory* (5, 6 and 7), one experiment consists on the execution of the puzzle with memory 5 and p set to 0; then memory 5 and p, 0.1; $\ldots$; memory 5 and p, 1; memory 6 and p, 0; and so on.

A system execution needs the number of agents, a value for the *redirection probability* and a value for the memory. The first step is the creation of the initial regular topology. Then the agent topology is modified for the given *probability of redirection*. Once the topology is modified, the communication channels are fixed, and the Multi-Agent System is executed and the topology does not change during this execution.

From each system execution two different metrics are extracted, *iteration* and *relative global error*. One iteration represents one execution of the agents in the network, the information processing is shown in Algorithm 1.

Algorithm 1 Agent information processing

```
1:  Receive messages
2:  for each received message do
3:      for each message registry do
4:          Analyze AId
5:          if Sender recognized then
6:              Analyze AIn
7:              Update status
8:          end if
9:          Include message registry in local informational context
10:     end for
11: end for
12: Send local informational context
```


DOI: 10.1002/cpe

In order to know when the system has finished, the *relative global error* is used. This *relative global error* is the percentage of sides not matched and it is based on the error of the pieces.

The error of a specific piece i (ε_i) is the number of sides not matched. The maximum error of a piece is 4, because in jigsaw puzzle, pieces have 4 sides. Using the concept of piece error, it easy the definition of the *puzzle global error* (ε) as the addition of the error of all the pieces of the puzzle. Given a puzzle with N pieces, the *puzzle global error* is defined by Equation 6.

$$\varepsilon = \sum_{k=1}^{N} \varepsilon_k \tag{6}$$

On the other hand, it is possible to define the maximum error of the puzzle. The maximum error of a puzzle (ε_T) is equal to the number of non-zero sides of the pieces. A side with value 0 represents the border of the puzzle and those sides does not have error. This maximum error is defined by Equation 7.

$$\varepsilon_T = 4(N - \sqrt{N}) \tag{7}$$

Finally, using the concept of maximum error and global error of a puzzle, it is possible to define the *relative global error*, ε_R, as the percentage of the the puzzle not matched. This error is defined by Equation 8 and will be used to evaluated the performance of the system in Section 6.

$$\varepsilon_R = \frac{\varepsilon}{\varepsilon_T} \tag{8}$$

6. EXPERIMENTAL RESULTS

The experiments carried out in this work analyze the influence of the *redirection probability* and the *memory size* in the system. The study of those parameters allow the definition of the optimal values for which the system achieve the best performance measured by the number of iterations needed to solve the puzzle and the *relative global error*.

In all the experiments, the values for the *probability of redirection* go from 0 to 1 in increments of 0.1. The values for the *memory size* are different depending on the number of pieces that compose the puzzle. The maximum value for the *memory size* is equal to the number of pieces in the puzzle and it means that, in a specific time, a message will contain the information of all the pieces. On the other hand, the minimum memory size is 1, but agents will store only the information of the piece assigned to them and thus the puzzle will not be solved. In this work three different systems are analyzed composed by 100, 144 and 225 agents. The values for the *memory size* for each system are described in Table I.

With the configuration described in this section, a total number of 7400 puzzles have been solved. The aim of all these experiments is the definition of the optimal values for the *redirection probability* and the *memory size* in order to apply them into larger systems composed by 400, 500 or 1000 agents.

DOI: 10.1002/cpe

Num. Agents	Min. Memory	Max. Memory	ΔMemory
100	10	100	10
144	20	140	20
225	60	140	20

Table I. This table shows the minimum and the maximum value and the increment (Δ) of the *memory size* in systems with different number of agents. The value of this parameter will define the size of the *local informational context*.

Figures 6 and 7 show the iterations taken by the system in the case where it is composed by 100 (Figure 6) and 225 agents (Figure 7). As can be seen in those figures there are an important improvement in the number of iterations in the case where the topology has been reconnected according to a specific probability. This fact is represented in the decrease of the iterations taken to solve the puzzle with probability 0.1 compared with the iterations need when the probability is fixed to 0.

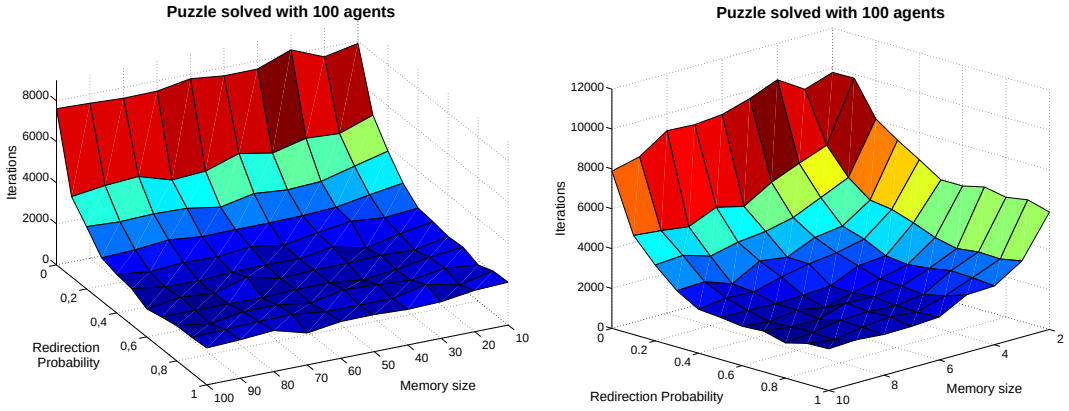


Figure 6. This figure shows the iterations taken by the system to solve a puzzle with 100 agents. Left figure shows the number of iterations when agents have short memory size. Right figure shows the performance of the system using larger memories.

Although in Figure 6(a) there are not important improvement in the performance of the system produce by the memory size, in Figure 6(b) can be seen that memory size is also a very important parameter that defines the behaviour of the system. In systems composed by 100 agents, there are a significant reduction in the number of iterations when the values of the memory size vary between 2 and 10. Nevertheless, little changes are detected in the number of iterations when the memory size is greater than 10 message registries. This fact is produced by a limit in the memory size value from which the system is not improved significantly. In the case of systems composed by 100 agents the limit in the memory size can be set in 10 message registries.

Other metrics that defines the performance of the system is the *relative global error*. This metric is used to determine the percentage of the puzzle that is not matched and it is defined by Equation 8. Figure 8(a) corresponds to the results shown in Figure 6(a) without the memory size dimension, this can be done because in 6(a) the memory size limit is reached. Figure 8(b) shows the *relative*

DOI: 10.1002/cpe

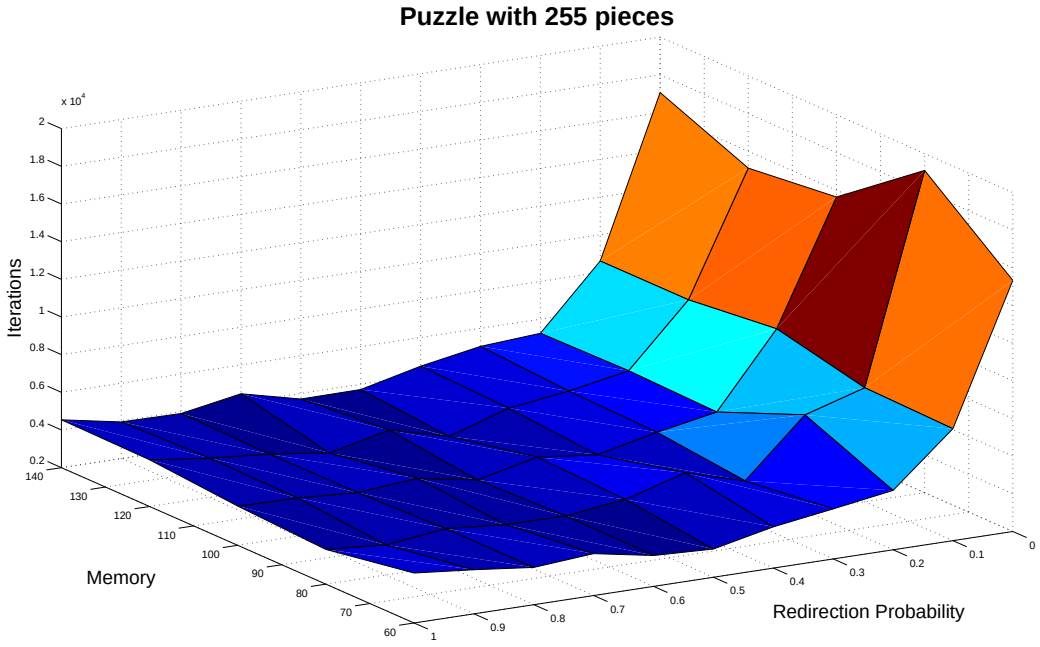


Figure 7. This figure shows the performance of the system as measured by the number of iterations taken to solve a puzzle with 225 pieces.

global error of the experiments. In this figure can be seen that from a redirection probability higher or equal to 0.5 there are not any error, this means that the system has resolved the puzzle.

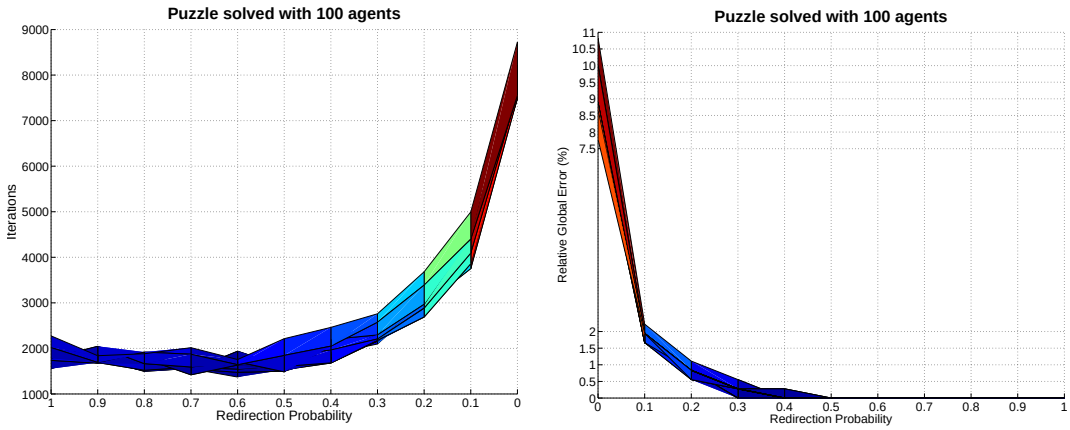


Figure 8. In this figure the iterations taken to solve puzzles with 100 pieces and the relative global error are shown. Left figure corresponds to the results shown in Figure 6(a) without the memory size dimension. Right figure shows the relative global error (computed using Equation 8).

Analyzing Figure 8, one could conclude that the best performance is carried out with redirection probability between 0.5 and 0.6 because those are the minimum probability values in which the system resolve the puzzles. Nevertheless, using a redirection probability between 0.2 and 0.4 the system takes more iterations to solve the puzzles but the *relative global error* is lower or equal to the 1%. In this situation the puzzle is almost solved and fewer connections have been redirected

Prepared using cpeauth.cls
DOI: 10.1002/cpe

than using a redirection probability of 0.5.

Figure 9 shows the iterations taken by the system to solve puzzles with 225 agents and the associated relative global error. In these type of puzzles, again, with redirection probability higher or equal than 0.5 the system solves the puzzle, but as the number of pieces increases, the number of connections increases too, and the redirection of 50% of the links in the network could be very expensive. The best results for solving puzzles with 225 pieces are obtained in the case where the probability of redirection is between 0.2 and 0.4, where the mean *relative global error* is, respectively, 0.83 and 0.14.

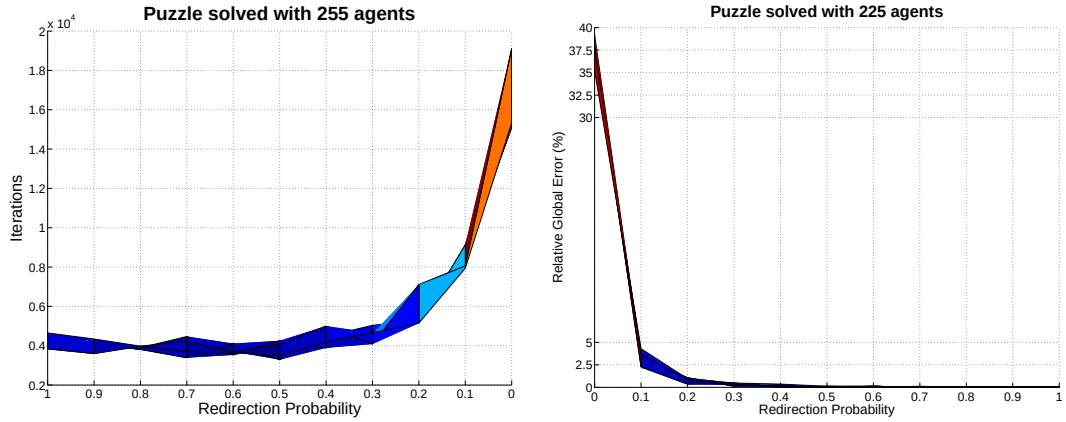


Figure 9. Left figure shows the performance of the system measured by the number of iterations taken to solve puzzles for a specific values of the redirection probability. Right figure shows the relative global error of the system when the system try to solve those puzzles.

Another puzzle, composed by 144 agents, has been analyzed. The results are shown in Table II.

Agents	Range Values for Redirection Probability	Relative Global Error
100	$0.2 - 0.4$	$0.8 - 0.02$
144	$0.3 - 0.4$	$0.54 - 0.29$
225	$0.2 - 0.4$	$0.83 - 0.14$

Table II. This table shows the summary of the different ranges values for probability of redirection and the average relative global error in different types of puzzles.

The aim of Table II is to determine the optimal values for the redirection probability in order to use them in larger puzzles. From Table II, the range values for this variable is fixed between 0.2 and 0.3. This result means that the best performance of the system is carried out redirecting between 20 and 30% of the links in the network.

With these values, some complex puzzles composed by 400, 625 and 1024 agents have been tested. The memory size of the system that tries to solve puzzles with 400 and 625 agents has been set to the 20% of the number of pieces. This value can be set because the analysis carried out in this work suggest that with this value the memory size limit is reached. System composed by 1024 agents are executed with memory size fixed to 50 to avoid stack problems in the computers. Table

DOI: 10.1002/cpe

III shows the iteration and the relative global error of those new puzzles with the configuration just explained.

Agents	Memory Size	Redirection Probability	Iterations	Relative Global Error
400	80	0.3	10353	1.184
		0.4	10055	1.711
625	125	0.3	19016	3.833
		0.4	17522	2.25
1024	50	0.3	23063	0.806
		0.4	17697	0.958

Table III. This table shows the relative global error and the iterations taken by the system to solve puzzles using 400, 625 and 1024 agents. For systems composed of 400 and 625 agents, the memory can store a maximum of 20% the number of pieces. In the cases where the systems are composed by 1024 agents, the memory has a limit of 50 pieces to avoid stack problems. The range values for the redirection probability is 0.3 - 0.4.

7. CONCLUSIONS

This paper makes a study of the different parameters that affect the performance of a bio-inspired multi-agent system. Agents are connected to interchange their local information and to solve the global problem for which the system was designed. This communication topology has been designed based on a regular topology where the connectivity degree of each node is defined by a power law distribution. This power law is used to simulate Internet-based topology [22, 23].

Communication channels in the initial topology are redirected according to a specific *probability value*. This redirection is used to reduce the *characteristic path length*. It is important to remark that this redirection is only performed before agents start to solve a given problem. Once all the communication channels are analyzed, agents start the execution and the topology cannot be changed.

Agents have been designed using a bio-inspired discrimination method based on the identification of the sender agent. This discrimination process reduces the computational efforts of the agents because only those messages that contain relevant information for the receiver agent are analysed. All received messages are included in the local informational context of the receiver agent. This local informational context is a temporal memory that stores the history of the received messages.

There many applications that could take advantage of the MAS described in this paper, an example of this kind of problems are multi-sorting problem, in which several objects need to be sorted but there are not any global sorting criteria.

The application domain shown in this work is a well-known sorting problem, i.e. jigsaw puzzles. This paper studies how the values for the redirection probability and the memory size of the agents affect the performance of the system. This performance is measured using the number of iterations

DOI: 10.1002/cpe

needed to solve the puzzle, and the relative global error, ε_R.

Finally, experimental results show that both redirection probability and memory size of agents affect the performance of the system. Both parameters have a threshold beyond which the performance of the system does not improve significantly. Moreover, the results allows the definition of an optimal range value for the redirection probability. With this range value, larger systems composed with 400, 625 and 1024 agents have been tested. The results of these new experiments show the relative global error produced is negligible with values lower than 4%. The experimental results have allowed the definition of a range values for the redirection probability in which the performance of the system is improved.

ACKNOWLEDGEMENTS

This work has been partially supported by the Spanish Ministry of Science and Innovation by the MICYT project ABANT (TIN2010-19872), TIN2010-19607 and BFU2009-08473.

REFERENCES

1. Motiee M, Khajepour A, Mansour R. A new topology optimization method for multi-physics micro domains. *Mechtronic and Embedded Systems and Applications*, 2008; 106 – 111, doi:10.1109/MESA.2008.4735689.
2. Sporns O, Tononi G, Edelman GM. Theoretical neuroanatomy: Relating anatomical and functional connectivity in graphs and cortical connection matrices. *Cerebral Cortex* 2000; (2):127–141, doi:10.1093/cercor/10.2.127.
3. Modha DS, Singh R. Network architecture of the long-distance pathways in the macaque brain. *Proceedings of the National Academy of Sciences* 2010; **107**(30):13 485–13 490, doi:10.1073/pnas.1008054107.
4. Harrison PG, Patel NM. *Performance Modelling of Communication Networks and Computer Architectures (International Computer S*. Addison-Wesley Longman Publishing Co., Inc.: Boston, MA, USA, 1992.
5. rong Gong X, Rong-cai Z, sheng Lu L. Communication optimization algorithms based on extend data flow graph. *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, ACIS International Conference on* 2007; **3**:3–8.
6. Jang MW, Agha G. Agent framework services to reduce agent communication overhead in large-scale agent-based simulations. *Simulation Modelling Practice and Theory* 2006; **14**(6):679 – 694. Distributed Systems Simulation.
7. Jurasovic K, Jezic G, Kusek M. A performance analysis of multi-agent systems. *ITSSA* 2006; **1**(4):335–342.
8. Jang MW, Agha G. Agent framework services to reduce agent communication overhead in large-scale agent-based simulations. *Simulation Modelling Practice and Theory* 2006; **14**(6):679–694.
9. Ayache N, Faugeras OD. Hyper: A new approach for the recognition and positioning of two dimensional objects 1986; **8**:44–54.
10. Wolfson H, Schonberg E, Kalvin A, Landam Y. Solving jigsaw puzzles by computer. *Ann. Oper. Res.* 1988; **12**:51–64.
11. Webster RW, LaFollette PS, Stafford RL. Isthmus critical points for solving jigsaw puzzles in computer vision 1991; **21**(5):1271–1278.
12. Schwartz JT, Sharir M. Identification of partially obscured objects in two and three dimension by matching noisy characteristic curves 1986; **8**:44–54.
13. Chung M, Fleck M, Forsyth D. Jigsaw puzzle solver using shape and color. *Proc. ICSP'98 4th Internat. Conf. on Signal Processing*, Beiijing, China, 1998; 877–880.
14. Goldberg D, Malon C, Bern M. A global approach to automatic solution of jigsaw puzzles. *Computational Geometry* 2004; **28**:165–174.
15. Yao FH, Shao GF. A shape and image merging technique to solve jigsaw puzzles. *Pattern Recogn. Lett.* 2003; **24**(12):1819–1835.
16. Suganthan PN. Solving jigsaw puzzles using hopfield network. *Proc. Internat. Conf. on Neural Networks*, Washington DC, USA, July 1999; 10–16.
17. Toyama F, Fujiki Y, Shoji K, Miyamichi J. Assembly of puzzles using a genetic algorithm. *16th International Conference on Pattern Recognition, 2002*, vol. 4, 2002; 389–392.
18. Sugawara T, Lesser V. Learning coordination plans in distributed problem-solving environments. *Technical Report*, In Twelfth International Workshop on Distributed Artificial Intelligence 1993.
19. Sugawara T, Kurihara S. Learning message-related coordination control in multiagent systems. *Selected Papers from the 4th Australian Workshop on Distributed Artificial Intelligence, Multi-Agent Systems*, Springer-Verlag, 1998; 29–44.
20. Gonzalez-Pardo A, Varona P, Camacho D, de Borja Rodriguez F. Optimal message interchange in a self-organizing multi-agent system. *Intelligent Distributed Computing IV*, *Studies in Computational Intelligence IV*, vol. 315, Springer Verlag, 2010; 131 – 141.

DOI: 10.1002/cpe

21. Latorre R, Rodríguez FB, Varona P. Signature neural networks: Definition and application to multidimensional sorting problems. *IEEE Transaction on Neural Networks* 2011; **22**(1):8 – 23.
22. Yan G, Eidenbenz S, Thulasidasan S, Datta P, Ramaswamy V. Criticality analysis of internet infrastructure. *Computer Networks* 2010; **54**:1169–1182.
23. Jin S, Bestavros A. Small-world characteristics of internet topologies and implications on multicast scaling. *Computer Networks* August 2005; **50**:648–666.
24. Szücs A, Pinto R, Rabinovich M, Abarbanel H, Selverston A. Synaptic modulation of the interspike interval signatures of bursting pyloric neurons. *Journal of neurophysiology* 2003; **89**(3):1363–1377.
25. Campos D, Aguirre C, Serrano E, Rodríguez FB, de Polavieja GG, Varona P. Temporal structure in the bursting activity of the leech heartbeat cpg neurons. *Neurocomputing* 2007; **70**(10 - 12):1792 – 1796.
26. Latorre R, Rodríguez FB, Varona P. Neural signatures: multiple coding in spiking-bursting cells. *Biological Cybernetics* 2006; **95**:169–183.
27. Szücs A, Abarbanel H, Rabinovich M, Selverston A. Dopamine modulation of spike dynamics in bursting neurons. *Eur J Neurosci* 2005; **21**(3):763–72.
28. Bonabeau E, Dorigo M, Theraulaz G. *Swarm Intelligence: From Natural to Artificial Systems (Santa Fe Institute Studies in the Sciences of Complexity Proceedings)*. 1 edn., Oxford University Press, USA, 1999.
29. Watts DJ, Strogatz SH. Collective dynamics of 'small-world' networks. *Nature* 1998; **393**(6684):440–442.
30. Dominguez D, Gonzalez M, Serrano E, Rodriguez FB. Structured information in small-world neural networks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)* 2009; **79**(2):021 909, doi:10.1103/PhysRevE.79.021909.
31. Jain AS, Meeran S. A state-of-the-art review of job-shop scheduling techniques 1999.

DOI: 10.1002/cpe

Referencias

[1] Fabio Luigi Bellifemine, Giovanni Caire, and Dominic Greenwood, *Developing Multi-Agent Systems with JADE*, Wiley, 2007.

[2] C. Bäumer and T. Magedanz, "Grasshopper ? a mobile agent platform for active telecommunication networks," in *Intelligent Agents for Telecommunication Applications*, Sahin Albayrak, Ed., vol. 1699 of *Lecture Notes in Computer Science*, pp. 690–690. Springer Berlin / Heidelberg, 1999.

[3] Kresimir Jurasovic, Gordan Jezic, and Mario Kusek, "A performance analysis of multi-agent systems," *ITSSA*, vol. 1, no. 4, pp. 335–342, 2006.

[4] Myeong-Wuk Jang and Gul Agha, "Agent framework services to reduce agent communication overhead in large-scale agent-based simulations," *Simulation Modelling Practice and Theory*, vol. 14, no. 6, pp. 679–694, 2006.

[5] Toshiharu Sugawara and Victor Lesser, "Learning coordination plans in distributed problem-solving environments," Tech. Rep., In Twelfth International Workshop on Distributed Artificial Intelligence, 1993.

[6] Toshiharu Sugawara and Satoshi Kurihara, "Learning message-related coordination control in multiagent systems," in *Selected Papers from the 4th Australian Workshop on Distributed Artificial Intelligence, Multi-Agent Systems*. 1998, pp. 29–44, Springer-Verlag.

[7] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems (Santa Fe Institute Studies in the Sciences of Complexity Proceedings)*, Oxford University Press, USA, 1 edition, 1999.

[8] A. Szücs, R.D Pinto, M.I. Rabinovich, H.D. Abarbanel, and A.I. Selverston, "Synaptic modulation of the interspike interval signatures of bursting pyloric neurons.," *Journal of neurophysiology*, vol. 89, no. 3, pp. 1363–1377, 2003.

[9] R Latorre, F B Rodríguez, and P Varona, "Neural signatures: multiple coding in spiking-bursting cells," *Biological Cybernetics*, vol. 95, pp. 169–183, 2006.

[10] Antonio Gonzalez-Pardo, Pablo Varona, David Camacho, and Francisco de Borja Rodriguez, "Optimal message interchange in a self-organizing multi-agent system," in *Intelligent Distributed Computing IV*. September 2010, vol. 315 of *Studies in Computational Intelligence IV*, pp. 131 – 141, Springer Verlag Berlin Heidelberg.

[11] Antonio Gonzalez-Pardo, Pablo Varona, David Camacho, and Francisco de Borja Rodriguez, "Communication by identity in bio-inspired multi-agent systems," *International Journal Concurrency and Computation: Practice & Experience*, 2010.

[12] Real Academia Espa nola, *Diccionario de la Real Academia de la Lengua Española*, 2010.

[13] CORBA/IIOP, "Corba/iiop rev 2..3.1 documento formal 99-10-07," in *www.omg.org/corba/corbaiiop.html*, October 1999.

[14] S. Russell and P. Norvig, *Artificial Intelligence a Modern Approach*, Prentice-Hall, 1996.

[15] W. Brenner, R. Zarnekow, and H. Wittig, *Intelligent Software Agents. Foundations and Applications*, Springer-Verlag. ISBN: 3-540-63411-8, New York, 1998.

[16] S. Franklin and A. Graesser, "Is it an agent or just a program? a taxonomy for autonomous agents," 1996, pp. 21–35, Springer-Verlag.

[17] Hyacinth S. Nwana, "Software agents: An overview," *Knowledge Engineering Review*, vol. 11, no. 3, pp. 205–224, October/November 1996.

[18] Helmut Reiser and Gerald Vogt, "Security requirements for management systems using mobile agents," in *Proceedings of the Fifth IEEE Symposium on Computers and Communications (ISCC 2000)*, 2000, ISCC '00, pp. 160–165.

[19] A. S. Rao and M. P. Georgeff, "Bdi-agents: from theory to practice," in *Proceedings of the First Intl. Conference on Multiagent Systems*, San Francisco, 1995.

[20] Alan H. Bond and Less Gasser, *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Less Gasser editors, Morgan Kaufmann Publishers: San Francisco, CA, 1988.

[21] Tim Finin, R. Fritzson, D. Mackay, and R. McEntire, "Kqml as an agent communication language," in *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM94)*, Gaithersburg, Maryland, 1994, New York: Association of Computing Machinery, pp. 456–463, ACM Press.

[22] FIPA, "www.fipa.org," .

[23] Naoki Miyata and Toru Ishida, "Community-based load balancing for massively multi-agent systems," in *Massively Multi-Agent Technology: AAMAS Workshops, MMAS 2006, LSMAS 2006, and CCMMS 2007 Hakodate, Japan, May 9, 2006 Honolulu, HI, USA, May 15, 2007 Selected and Revised Papers*. 2008, pp. 28–42, Springer-Verlag.

[24] D. Campos, C. Aguirre, E. Serrano, F. B. Rodríguez, G. G. de Polavieja, and P. Varona, "Temporal structure in the bursting activity of the leech heartbeat cpg neurons," *Neurocomputing*, vol. 70, no. 10 - 12, pp. 1792 – 1796, 2007.

[25] Eric R. Kandel, J. Schwartz, and Thomas M. Jessell, *Principles of Neural Science*, Elseiver Science Publishing, 3th edition edition, 1991.

[26] A. Szücs, H.D.I. Abarbanel, M.I. Rabinovich, and A.I. Selverston, "Dopamine modulation of spike dynamics in bursting neurons.," *Eur J Neurosci*, vol. 21, no. 3, pp. 763–72, 2005.

[27] Roberto Latorre, "Neural signatures and multicoding in spiking-bursting neurons," Ph.D. dissertation, Escuela Politecnica Superior. Universidad Autonoma de Madrid., 2008.

[28] Guanhua Yan, Stephan Eidenbenz, Sunil Thulasidasan, Pallab Datta, and Venkatesh Ramaswamy, "Criticality analysis of internet infrastructure," *Computer Networks*, vol. 54, pp. 1169–1182, 2010.

[29] Shudong Jin and Azer Bestavros, "Small-world characteristics of internet topologies and implications on multicast scaling," *Computer Networks*, vol. 50, pp. 648–666, August 2005.

[30] Duncan J. Watts and Steven H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[31] David Dominguez, Mario Gonzalez, Eduardo Serrano, and Francisco B. Rodriguez, "Structured information in small-world neural networks," *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, vol. 79, no. 2, pp. 021909, 2009.

[32] N. Ayache and O. D. Faugeras, "Hyper: A new approach for the recognition and positioning of two dimensional objects," vol. 8, pp. 44–54, 1986.

[33] H. Wolfson, E. Schonberg, A. Kalvin, and Y. Landam, "Solving jigsaw puzzles by computer," *Ann. Oper. Res.*, vol. 12, pp. 51–64, 1988.

[34] R W. Webster, P. S. LaFollette, and R. L. Stafford, "Isthmus critical points for solving jigsaw puzzles in computer vision," vol. 21, no. 5, pp. 1271–1278, 1991.

[35] J T Schwartz and M Sharir, "Identification of partially obscured objects in two and three dimension by matching noisy characteristic curves," vol. 8, pp. 44–54, 1986.

[36] M.G. Chung, M. Fleck, and D.A. Forsyth, "Jigsaw puzzle solver using shape and color," in *Proc. ICSP'98 4th Internat. Conf. on Signal Processing*, Beiijing, China, 1998, pp. 877–880.

[37] David Goldberg, Christopher Malon, and Marshall Bern, "A global approach to automatic solution of jigsaw puzzles," *Computational Geometry*, vol. 28, pp. 165–174, 2004.

[38] Feng-Hui Yao and Gui-Feng Shao, "A shape and image merging technique to solve jigsaw puzzles," *Pattern Recogn. Lett.*, vol. 24, no. 12, pp. 1819–1835, 2003.

[39] P. N. Suganthan, "Solving jigsaw puzzles using hopfield network," in *Proc. Internat. Conf. on Neural Networks*, Washington DC, USA, July 1999, pp. 10–16.

[40] Fubito Toyama, Yukihiro Fujiki, Kenji Shoji, and Juichi Miyamichi, "Assembly of puzzles using a genetic algorithm," in *16th International Conference on Pattern Recognition, 2002*, 2002, vol. 4, pp. 389–392.

[41] R Latorre, F B Rodríguez, and P Varona, "Signature neural networks: Definition and application to multidimensional sorting problems," *IEEE Transaction on Neural Networks*, vol. 22, no. 1, pp. 8 – 23, 2011.

[42] Tom White, *Hadoop: The Definitive Guide*, O'Reilly, first edition edition, june 2009.

[43] Gaku Yamamoto, Hideki Tai, and Hideyuki Mizuta, "A platform for massive agent-based simulation and its evaluation," in *6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2007, p. 132.