UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

ORIENTACIÓN ESPACIAL EN INSECTOS SOCIALES: PATRONES FEROMONALES AUTORGANIZADOS

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

MATEMÁTICO

P R E S E N T A :

MAURICIO FLORES PEÑA

TUTOR: DR. OCTAVIO MIRAMONTES VIDAL

2006

Hoja de Datos del Jurado

1. Datos del alumno
   Flores
   Peña
   Mauricio
   5554 4734
   Universidad Nacional Autónoma de México
   Facultad de Ciencias
   Matemático

2. Datos del tutor
   Dr.
   Octavio
   Miramontes
   Vidal

3. Datos del sinodal 1
   Dr.
   Pablo
   Padilla
   Longoria

4. Datos del sinodal 2
   Dr.
   Ricardo
   Mansilla
   Corona

5. Datos del sinodal 3
   Dr.
   José
   Marquina
   Fábrega

6. Datos del sinodal 4
   M. en C.
   Alberto
   Salazar
   Martínez

7. Datos del trabajo escrito.
   Organización Espacial en Insectos Sociales:
   Patrones Feromonales Autorganizados
   74 páginas
   2006

Dedicatoria:

A mi abuela, madre y hermana, en orden causal, por su incondicionalidad. A Octavio Miramontes por su inagotable paciencia. A Talía por su amable compañía. A Julián por su integridad. A Fidel por la sugerencia, a Edgar y Marcela Santillán por la oportunidad.

ÍNDICE

RESUMEN.....iv

INTRODUCCIÓN.....v
1. Orden implícito en la materia
2. Representabilidad del seguimiento feromonal
3. Formalismos discretos de representación
4. Cómputo masivo: interacciones estocásticas puntuales

ANTECEDENTES FENOMENOLÓGICOS.....1
1. Ciclo de vida de un hormiguero en el desierto de Arizona
2. Comunicación en sociedades de insectos
3. Comunicación químicamente mediada
4. Dinámica espacial a representarse

MODELO HEURÍSTICO.....12
1. Antecedentes teóricos del modelo
2. Descripción del modelo propuesto
3. Características específicas funcionales
4. Parámetros involucrados

IMPLEMENTACIÓN DIGITAL.....20
1. El procesamiento paralelo y distribuido
2. Clasificación y arquitectura
3. La programación orientada a objetos
4. El uso de RMI
5. División lógica y funcional del sistema
6. Seguimiento feromonal y parámetros del modelo
7. Interacción entre los módulos del sistema

RESULTADOS.....37
DISCUSIÓN.....43
CONCLUSIONES.....47
BIBLIOGRAFÍA.....56

RESUMEN

El tema de este trabajo es la emergencia de estructuras ordenadas a nivel global a partir de múltiples microinteracciones entre partículas y pequeños cambios que introducen dichas partículas en el medio. El mecanismo particular a representar es quizás más representativo de esta clase de sistemas. Se trata de la retroalimentación mútua entre deposición y seguimiento feromonal, de suma importancia para la articulación colectiva de millares de insectos sociales en un todo funcional, altamente eficiente y especializado.

La introducción delimita el marco teórico en el que se inscribe el modelo planteado y recapitula sobre el desarrollo histórico de las ideas que sustentan este enfoque. A continuación sigue un capítulo descriptivo de la fenomenología biológica donde se observa la dinámica representada por el modelo desarrollado; la comunicación químicamente mediada en una colonia de insectos sociales. En el capítulo dos se discute que aspectos del fenómeno biológico serán retomados para y cómo se cuantificarán en la elaboración de un autómata celular bidimensional; posteriormente se presenta la implementación del modelo como un programa digital paralelo, orientado a objetos y en seguida se reportan los resultados obtenidos de los experimentos numéricos con este simulador. Finalmente, se discuten estos resultados y se enuncian las conclusiones.

INTRODUCCIÓN

Se han demostrado isomorfismos parciales entre los mecanismos de adaptación, reconocimiento y cognición. De aquí podemos afirmar que los procesos semióticos abarcan una gran parte de la biología; de manera que podemos hablar de la semiótica como la base de los procesos biológicos (Kull, 1993).

El interés por el estudio de los principios organizativos en la dinámica espacial colectiva de los insectos sociales se debe a su extraordinaria capacidad de coordinar miles de individuos en la realización de tareas altamente especializadas sin un mecanismo de control central. La organización coherente del total de la colonia de insectos surge de las interacciones puntuales, a nivel local y aparentemente erráticas entre los individuos que se ubican a una escala espaciotemporal menor que el patrón total. Las reacciones de cada individuo a los estímulos que percibe varían en función de su estado motivacional, inducido por el contexto socioespacial. ¿De qué mecanismos se sirve una colonia de insectos sociales para inducir y modificar estos estados variables de respuesta individual e integrarlos en patrones colectivos?

La interacción puntual entre estructuras existentes a una escala espaciotemporal que dan lugar a un nuevo patrón dinámico perceptible, como estructura por derecho propio en una escala mayor, es un principio común al surgimiento de redes de interacción coherente (orden) a través de todos los niveles de organización. Así pues, la coordinación espacial en insectos sociales es relevante por estar fundamentada en principios de organización comunes a todas las escalas de la fenomenología. El tamaño de los insectos sociales permite observar a simple vista tanto las interacciones individuales como las configuraciones globales que de allí emergen; por tanto, proveen un espacio ideal para explorar los principios generales detrás del cómputo colectivo difuso.

Este trabajo tiene como punto de partida una situación experimental en laboratorio y busca reproducir cualitativamente los patrones espaciales ocurridos. Con esto se busca establecer que el surgimiento de estructuras

ordenadas a nivel global puede descansar sobre principios simples de interacción local. Por otra parte, espero que este documento tenga valor didáctico como material introductorio al modelaje computacional de sistemas biológicos, así como a la programación orientada a objetos y al cómputo paralelo. En lo que sigue se revisa el desarrollo y posturas del enfoque teórico sistémico en el que se inscribe el modelo que aquí se presenta.

Orden implícito en la la materia

En la tradición occidental podemos remontar las ideas de orden estructural a Goethe quién fue pionero en estudiar la biología desde una perspectiva dinámica y de desarrollo a partir del "orden en movimiento" manifiesto en la naturaleza, de la forma como un "patrón de relaciones en el seno de un todo organizado" y de la evolución como transformación estructurales. Es Cuvier el primero en plantear un sistema de clasificación zoológica basado en las similitudes de sus relaciones estructurales. El organismo es un ser organizado y autorganizador para Kant y un sistema unificado de coevolución orgánica para Von Humboldt (Capra, 1996, pp.41-42).

El término de "propiedad emergente" fue acuñado por el filósofo C.D. Broad para aquellas propiedades que surgen a un cierto nivel de complejidad y están ausentes en niveles inferiores. Durante los años 20 en Alemania, tanto la biología organicista como la psicología Gestalt formaron parte de una corriente intelectual mayor que denunciaba la creciente fragmentación y alienación de la cultura humana. También a principios del siglo XX se acuñaron las nociones de ecología, entorno, ecosistema y de la vida como consecuencia pero también como fuerza reguladora del entorno planetario Hacia los 30 la mayoría de los criterios clave del pensamiento sistémico habían sido ya formulados por los biólogos organicistas, los psicólogos de la Gestalt y ecólogos (Capra, 1996, pp.52,53).

No obstante, en un movimiento pendular más del desarrollo científico, con la teoría celular de Rudolph Virchow, la atención se desplazó de los organismos a las células. Como alternativas históricas a la perspectiva mecanicista, los vitalistas adoptan la postura de que existe una fuerza o campo extrafísico responsable de las propiedades autorganizativas en la materia viva; mientras que los organicistas parten de que la autorganización es una propiedad emergente, implícita ya en la materia que conforma estos sistemas.

El hallazgo de indicios de autorganización, presentes desde el nivel molecular, ha resultado sorprendente puesto que, hasta bien entrado el siglo XX, se consideraba a la materia como una entidad inherentemente irresponsiva y paciente ante fuerzas externas. Esta postura, en el terreno de la teoría evolutiva, se instaló hegemónicamente con la síntesis neodarwinista, postulando la selección natural sobre mutaciones aleatorias al código genético como única fuente de orden biológico[1]. Se trata de un prejuicio histórico de una sociedad basada en la competencia, organizada centralmente y con una relación utilitarista respecto a su entorno. Es el agotamiento de este mismo modelo, tanto en lo teórico como en sus implicaciones sociales[2] lo que posibilita y promueve el planteamiento de modelos teóricos alternativos que replantean nuestra concepción respecto al fenómeno de lo vivo y nuestra relación con éste.

De acuerdo a Boltzman no existe ninguna ley física que se oponga al surgimiento espontáneo de orden sino que éste es altamente improbable bajo interacciones puramente aleatorias. Cuanto mayor el número de configuraciones moleculares posibles a disposición de un sistema mayor desorden y mayor energía requerida en propiciar un estado ordenado. En el caso de las estructuras disipativas las moléculas no se hallan en estados de alta probabilidad aleatoria sino interrelacionadas mediante ciclos de retroalimentación describibles mediante ecuaciones diferenciales que pueden presentar atractores no triviales e incluso, mediante bifurcaciones, cambiar de cuenca de atracción. Desde estas bifurcaciones pueden surgir estados altamente ordenados, lo cual no contradice la 2da ley puesto que los sistemas vivos aumentan la entropía circundante para disminuirla internamente.
De acuerdo a la biología sistémica, el surgimiento de los procesos orgánicos no responde a una muy larga serie de mutaciones aleatorias con efectos determinísticos que hayan propiciado la adquisición gradual de características funcionales, ni es producto de una fuerza extrafísica. Los elementos químicos no se combinan aleatoriamente, sino de modo pautado, por lo que es factible que cierto tipo de reacciones favorecidas por el medio abiótico de la tierra

1 Mutación puntual y selección no son un mecanismo suficiente para explicar el orden biológico; el espacio paramétrico es demasiado grande (los párrafos alternativos para secuencias de tan solo 300 nucleótidos es superior al número de átomos en el universo, (Eigen M., 1997) y cada mutación puntual difícilmente conferiría alguna ventaja adaptativa , por lo que el surgimiento de estructuras nuevas no puede basarse en una secuencia específica de aquellas.

2 Los sistemas de control contemporáneos devienen más costosos que las tareas que regulan, por ejemplo, la UNAM gasta el 60% de su presupuesto en la administración de sus tareas docentes y científicas (periódico la Jornada).

primigenia se hayan entrecruzado formando redes autocatalíticas de creciente complejidad hasta convertirse en estructuras disipativas con propiedades autopoiéticas al pasar por puntos sucesivos de bifurcación en coevolución con su medio.

La serie completa de genes de un organismo forma una vasta red interconectada, rica en ciclos de retroalimentación en la que los genes regulan, directa o indirectamente sus actividades. No se trata de una disposición lineal de genes independientes sino que la red presenta múltiples efectos recíprocos mediatizados por represores y derrepresores, exones e intrones, genes móviles e incluso proteínas estructurales (Varela, 1991).

La estructura de una red genética es la misma para todas las células pero los patrones de actividad genética presentados difieren, puesto que éstos son representables como distintos ciclos de estados en la red binaria, donde los diversos tipos de célula corresponden a atractores distintos. Este modelo es también apoyado por la predicción respecto al número de atractores (tipos celulares) a partir del número de genes y por la correspondencia en que al diferir cada atractor por pocos elementos los tipos celulares de un organismo en efecto difieren por la expresión de un pequeño número de genes. Una red binaria con un núcleo dinámico fijo e islas separadas de actividad fluctuante. Los tripletes de ácidos nucleicos son capaces de especificar un aminoácido sólo en el contexto del metabolismo celular, es decir, como parte de una red compleja de interacciones enzimáticas. Sólo a partir de las propiedades emergentes de esta red podemos considerar a los tripletes como un código (Kauffman, 1993).

Mientras las estructuras disipativas reciben energía del exterior, las inestabilidades y saltos a nuevas formas de organización son el resultado de fluctuaciones internas, amplificadas por ciclos de retroalimentación positiva. Estas consideraciones teóricas fueron puestas en práctica durante los sesentas con el estudio de las reacciones enzimáticas autocatalíticas. De acuerdo a estos resultados sería posible extrapolar que la vida tiene sus raíces profundas en el reino de la materia muerta (Eigen 1992).

Podemos caracterizar los organismos como sistemas dinámicos disipativos (que presentan correlaciones estructurales bioquímicas que los mantienen en equilibrio dinámico respecto al intercambio térmico con el medio) y memoria, es

decir, capacidad de replicación. Los genes que restringen los procesos de autorganización a aquellas configuraciones históricamente dotadas de alta probabilidad de éxito, no son, por tanto, el motor del proceso evolutivo sino aquéllo que reduce el espectro de su variabilidad a lo viable (Kauffman 1993). La evolución ocurre cuando se vuelven accesibles nuevas vías de degradación energética más eficientes, a medida que un nuevo conjunto de interacciones y actividades emergen se dan cambios evolutivos de manera abrupta. Bajo esta óptica, la vida es un balance entre los imperativos de optimización para la supervivencia y la capacidad de degradación energética. Estos procesos dan lugar a un continuo de complejidad de sistemas dinámicos basados unos sobre otros, suprasistemas que imponen un conjunto de restricciones a los subsistemas que contienen (Schneider *et. al.* en Murphy, 2000). Siendo muy lejano al equilibrio, el estado de un ser vivo se mantiene estable a lo largo de prolongados periodos, mantiene su estructura general a pesar del incesante flujo y cambio de componentes y entorno. En un sistema vivo el producto de su operación es su propia organización.

A medida que el concepto de red ha cobrado más relevancia en nuestros días, se ha ido aplicando a todos los niveles de organización sistémica, así, los flujos de materia y energía en los ecosistemas son vistos como una extensión de las vías metabólicas que fluyen a través de un organismo. En cuanto a la división en tres niveles de sistemas vivos: organismos, partes de organismos y comunidades de organismos; cabe notar que en ciertas comunidades de bacterias, en los insectos sociales o ciertos pares simbióticos los individuos que componen la comunidad se relacionan con ésta de un modo que asemeja mucho la relación del organismo con sus partes.

La ciencia sistémica establece que las características de una estructura disipativa (y por ende los sistemas vivos) no pueden ser reducidas a la interacción de sus partes; del análisis de las partes no se deducen propiedades intrínsecas al sistema que sólo pueden entenderse en el contexto del todo. “En el fondo no hay partes, lo que denominamos parte es meramente un patrón dentro de una red inseparable de relaciones”. Aparecen correlaciones de largo alcance en la transición de fase del desorden al orden y a partir de entonces el sistema se comporta como un todo. Por otra parte, el comportamiento de una estructura disipativa no sigue ninguna ley universal sino que es exclusivo del sistema específico. Existen al menos dos fuentes de impredecibilidad, la que ocurre en los puntos de bifurcación de la dinámica del sistema y la no

linearidad que hace poco viables las predicciones numéricas al amplificar cualquier error. Estas estructuras son predecibles por sólo estrechos márgenes temporales.

En la visión mecanisista el mundo es una colección de objetos, en la visión sistémica los objetos en sí mismos son redes de relaciones inmersas en redes mayores. De acuerdo con el pensamiento procesal sistémico son las redes de interacción las que son prioritarias. En la ciencia mecanicista hay estructuras fundamentales y luego fuerzas y mecanismos mediante los que interactúan dando lugar a procesos; la ciencia sistémica considera cada estructura como la manifestación de procesos subyacentes. De hecho, nuestras descripciones forman también una red interconectada de conceptos y modelos sin cimientos últimos : "lo que observamos, no es la naturaleza en sí misma, sino la naturaleza expuesta a nuestro método de observación" (Heisenberg en Capra, 1996). La física, por tanto, no se puede considerar como el nivel fundamental de la ciencia; cada rama del conocimiento modela cierto nivel sistémico sin que ninguno conforme su fundamento.

Este trabajo es un ejercicio de modelaje enmarcado en el intento por establecer conjeturas viables respecto a las "...estrategias inferenciales vigentes en la materia" (Esté, 1997). Algunos autores han denominado "inteligencia en masa"[3] a las propiedades dinámicas que caracterizan las estrategias de optimización operacional observadas entre los insectos sociales. Las propiedades emergentes obtenidas son estructuras espaciales globales altamente estructuradas, lo cual resulta notable puesto que la dinámica subyacente está dada en términos de interacciones locales estocásticamente definidas para cada individuo. Podemos entonces afirmar que la cooperación no supervisada o dirigida surge principalmente de la autorganización de interacciones puntuales entre individuos y de cada individuo con su entorno inmediato. Con este trabajo se ilustran y exploran principios dinámicos que, a partir de principios locales simples, dan lugar a soluciones globales óptimas para tareas no triviales (cfr. Bonabeau & Theraulaz, 2000).

Representabilidad del seguimiento feromonal

Las evidencias de complejidad material y logística que exhibe una colonia de insectos sociales sugieren para éstas una alta capacidad cognitiva distribuida sobre los individuos que la conforman. Por ejemplo, tenemos que la hormiga

3 *Swarm intelligence (Bonabeau E. & Thérelauz G., 2000)*

Formica polyctena construye sus montículos con piñas de pino; se requieren de 100 000 000 de acciones individuales para cada metro cúbico de estructura (cfr. Van Damme 1998 en Theraulaz *et.al.*, 2003). Debido al sofisticado manejo de información por parte de la colonia, así como por la complejidad de las tareas que llevan a cabo, se ha indagado metafóricamente respecto al funcionamiento de la mente humana (y de los procesos de generación y procesamiento de información en general) tendiendo paralelismos de un dominio a otro en busca de esclarecer su origen y mecanismos.

Maturana (1988) ha logrado sintetizar la búsqueda de los principios de formación de estructuras biológicas con la de la naturaleza de la cognición en cuanto a que ambos están basados estructuralmente en ciclos de retroalimentación; procesos organizativamente cerrados que permiten la evolución a partir de un patrón global en el que la función de cada componente es ayudar a producir y transformar a los otros componentes manteniendo la ciclicidad de la red. El sistema nervioso no sólo es autorganizado sino autorreferente, de modo que la percepción no puede ser tomada por la representación de una realidad externa sino como la creación continua de nuevas relaciones al interior de una red neuronal. La relación entre cognición y realidad no es de representación sino de especificación y actualización.

La mente no es una estructura sino un proceso, la actividad organizadora de los sistemas vivos es una actividad mental (facilitada por el cerebro en los animales superiores). En el contexto del organismo humano hay indicios suficientes para afirmar que los sistemas nervioso, inmunológico y endocrino conforman una sola red cognitiva basada en la síntesis de polipéptidos. Los sistemas vivos son sistemas cognitivos, el proceso de vivir es una cognición. Esta afirmación es válida para todos los organismos, tengan sistema nervioso o no. El enfoque sistémico no se centra en las propiedades de los componentes sino en los procesos y relaciones entre los procesos realizados entre componentes en términos de patrón, estructura y procesos. La estructura de un sistema es la manifestación física de su organización interna.

Kauffman (cfr. 1993, 2000) ha utilizado el formalismo discreto de las redes booleanas para mostrar que cuando el número de relaciones, n, entre elementos supera n/2, las relaciones comienzan a mostrar ciclos estables y que estos “conjuntos autocatalíticos” pueden evolucionar haciéndose más

complejos. De acuerdo con Kalevi Kull (1993) la célula es el primer sistema semióticamente completo. Dadas las condiciones "...la materia se desarrolla y se constituye en seres vivos, incluso en seres vivos conscientes" (Bohm 1987:19 en Esté, 1997); con lo que el surgimiento de comportamiento coherente sería una propiedad implícita en interacción fisicoquímica de un sistema termodinámicamente complejo. Estos argumentos justifican la utilización de reglas heurísticas al explorar los mecanismos que generan un comportamiento dinámico global coherente a partir de reglas simples, enunciables computacionalmente, a nivel individual y hacen posible la coordinación de multitudes en el caso de los insectos sociales.

Formalismos discretos de representación

Por la virtual imposibilidad de representar los niveles de complejidad desplegados por los sistemas vivos sino en términos de una descripción analítica o discreta de la interacción de sus componentes físicos básicos, se hacen relevantes diversas técnicas de modelaje matemático y cibernético, así como el estudio de las propiedades de estos formalismos capaces de representar algunos de sus principios dinámicos.

A grandes rasgos se pueden dividir estas herramientas en dos clases, las basadas en el empleo de ecuaciones parciales integrodiferenciales y aquéllas sustentadas en el uso de formalismos discretos. En ocasiones es muy fina la línea que divide una simulación numérica y un autómata celular, puesto que ambos modelos conceptuales dependen de su implementación sobre un procesador digital, de naturaleza esencialmente discreta (cfr. Ermoentrout & Edelstein-Keshet, 1993).

Los métodos continuos tienen la ventaja de ser respaldados por las poderosas herramientas del análisis real, permitiendo la predicción numérica y una comprensión detallada de la dinámica de las variables involucradas. Por otro lado, este tipo de representación es exigente en cuanto a la comprensión detallada de la relación cuantitativa de los parámetros clave interactuantes. En tanto que los mecanismos discretos son fácilmente implementables, es posible elaborar un modelo para evaluar conjeturas aún en el caso de fenómenos de difícil cuantificación, para así evaluar la plausibilidad general de una hipótesis. A pesar de no lograr la predicción numérica sí contribuyen a la comprensión de los mecanismos responsables por el surgimiento de orden macroscópico. En muchas ocasiones se ha tenido éxito en obtener resultados relevantes a

problemas prácticos y específicos, por ejemplo, la representación y evaluación de mecanismos de regulación de tráfico en ciudades. Hay una gran cantidad de sistemas que son demasiado grandes para poder ser representados en términos de la mecánica clásica y demasiado pequeños para los métodos de la dinámica estadística. Es en esta escala espacio-temporal donde se ubica mucha de la fenomenología de los sistemas lejos del equilibrio (sistemas que involucran la interacción de cientos o miles de componentes) frecuentemente se encuentra una representación cómoda y útil en formalismos discretos tales como autómatas celulares, redes neurales o algoritmos genéticos.

El estudio de los sistemas complejos ha reunido un grupo de herramientas para representar sistemas naturales complejos, pero de manera más relevante, sus premisas constituyen un replanteamiento contrapuesto al discurso reduccionista y positivo característico del modernismo (Kauffman, 2000). Su énfasis está puesto en la capacidad de autorganización y surgimiento espontáneo de estructuras ordenadas de una escala a otra de organización de la materia; es decir en la relación de un todo con sus partes. Se continua avanzando en la exploración de fenómenos específicos de distinta índole sin paralizarse ante la falta de una teoría general que de un marco conceptual abarcante acabado.

El enfoque sistémico busca establecer los principios dinámicos que permiten el surgimiento de orden y dinámicas manifestadas a un nivel explorando las interacciones que ocurren entre los elementos constitutivos del fenómeno. No se busca reducir las relaciones observadas a términos “atómicos” específicos a un nivel privilegiado por leyes enunciadas respecto a su funcionamiento sino determinar si las restricciones en las interacciones al interior de un dominio determinan el surgimiento de estructuras en el otro o si existen niveles intermedios con elementos significantes involucrados.

Es importante contemplar que aunque la manifestación de estructuras macroscópicas ordenadas con base en microprocesos es un problema común a un espectro amplio de la problemática biológica, en sus términos generales, es prematuro especular respecto a los posibles alcances de este tipo de investigaciones dado que no se han logrado describir más que aspectos específicos de los fenómenos complejos abordados (Gordon, 1999).

Cómputo masivo: interacciones estocásticas puntuales

A pesar de las dificultades en describir estos sistemas en términos analíticos y con claridad respecto a los principios subyacentes es posible tomar ventaja de sus características en aplicaciones que retomen su arquitectura computacional. Las arquitecturas masivamente paralelas que caracterizan sistemas como el nervioso, las redes genéticas y las dinámicas colectivas de los insectos sociales pueden responder localmente a una gran cantidad de piezas dispersas de información. Lo anterior hace de estas arquitecturas candidatos ideales para ciertas aplicaciones que requieran del procesamiento masivo de datos.

El estudio del comportamiento de los insectos sociales es paradigmático respecto a este tipo de abordajes por involucrar al menos dos niveles de organización de escala macroscópica. Por una parte, tenemos al individuo, limitado en cuanto a tamaño, capacidad de procesamiento y recolección de datos, tiempo de vida, capacidad reproductiva y por el otro tenemos a la colonia, entidad altamente organizada con un ciclo de vida complejo que implica la realización de muchas y complejas tareas que requieren la coordinación de miles de individuos durante ciclos temporales largos y un comportamiento muy dúctil en cuanto a respuestas exigidas por situaciones altamente cambiantes.

La gran disponibilidad numérica de una colonia de insectos sociales, en contraste con la capacidad de procesamiento individual de cada elemento, permite aventurar que las estrategias de estas colonias se basan en la exploración *de facto* de la miríada de posibilidades existentes en contraste con los mecanismos operantes en organismos con mayor capacidad cerebral que parecieran evaluar mentalmente las posibilidades existentes y sólo aventurarse a llevar a cabo las más prometedoras.

Con todo, el misterio fundamental de las colonias de insectos sociales es su capacidad de llevar a cabo tareas altamente especializadas y complejas sin una jerarquía organizativa que administre instrucciones. Las hormigas se mueven independientemente, no obstante, nada de lo que hacen tiene sentido excepto en función del hormiguero. Mirando de cerca hay hormigas, de lejos, una colonia (Gordon, 1999). Por otra parte, la evidencia fenomenológica sobre la organización distribuida en estos insectos no descarta la existencia de niveles organizativos autónomos intermedios entre individuo y colonia (Hoffstater, 1979). Los resultados, aunque prematuros y limitados, han sido

fructíferos en cuanto a semillero de nuevas hipótesis acerca de los principios generadores de dinámicas de generación de orden espacial y en la generación de aplicaciones basadas en estos principios computacionales.

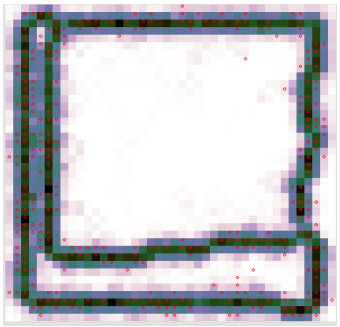
El hecho de que los dos niveles de interacción individuo/colectividad sean ambos fenómenos macroscópicos convierte a los insectos sociales en buenos candidatos para observar, medir y modelar en distintas situaciones buscando aclare algo general respecto a la semiosis implícita en la interacción masiva de entidades que generan configuraciones coherentes abstraibles como unidades funcionales a una escala mayor.

En todas las disciplinas surgen tendencias explicativas alternativas en las que crece la importancia asignada a la microdinámica intrínseca a los procesos. Se enfatizan relaciones intra e interespecíficas más allá de la competencia selectiva. Se cuestiona el proceso de mutación/selección como explicación cabal de la plétora de formas vivientes en favor de cambios cualitativos en las dinámicas morfogénicas que responderían a propiedades genéricas de la materia. Se reconocen fenómenos regulatorios importantes más allá de los genéticos. Se habla de asociaciones y patrones de activación cerebral en el caso de la producción lingüística, en contraste con reglas y gramáticas generativas, y se toma en cuenta la gran importancia de lo local, lo cotidiano y el hábito en la llamada microhistoria, en detrimento del papel determinante tradicionalmente otorgado a eventos y personajes particulares.

Los insectos sociales siempre han sido de interés para los estudiosos, además de que tradicionalmente han sido utilizados como fuente de alimento y medicina. El orden y complejidad inherentes a sus actividades (excavación de nido, construcción de panales, recolección de polen, cultivo de hongos, migraciones, búsqueda y obtención de alimento) es tanto más impresionante en cuanto que se trata de organismos antiquísimos que no cuentan con la evolución de estructuras nerviosas y sensoriales que asociamos típicamente con la inteligencia. Debido a la masificación de la dinámica urbana cotidiana y de la elaboración de bienes y prestación de servicios; es de esperar que los principios algorítmicos utilizados por los insectos sociales pueden encontrar un vasto campo de aplicación en el manejo de bases de datos, la administración de tráfico vial o virtual así como otros problemas de optimización relativos al manejo masivo de datos.

Estos fenómenos donde la acción de un individuo modifica el entorno, con lo cual las condiciones preexistentes a la acción de otro individuo se ve afectada por este cambio. Este tipo de comunicación indirecta que genera un "cálculo colectivo" se denomina estigmergia, del griego *stigma*-marca y *ergon*-trabajo, (Moura, 2002). Las termitas en África y Australia utilizan este procedimiento para edificar las grandes estructuras de sus nidos partiendo de azarosos depósitos de arcilla impregnada de feromona que promueven la deposición ulterior de más arcilla. Se ha observado que el mismo mecanismo que produce columnas da lugar a túneles ante la presencia adicional de un trazo feromonal (Bruinsma 1979 en Theraulaz *et.al* 2003); también se relata que el gradiente de concentración generado por la emisión feromonal de la termita reina de *M. Subhyalinus* determina la construcción de paredes que limitan su cámara, límites modificados a lo largo de su vida por su crecimiento (Theraulaz *et.al.,* 2003) y se observa el mismo fenómeno a partir de la ubicación de las larvas. De manera que el plano del nido se encuentra implícito en el entorno en forma de heterogeneidades físicas y químicas del medio a las constructoras se adaptan (Theraulaz *et.al* 2003).

CAPÍTULO 1. ANTECEDENTES FENOMENOLÓGICOS



"El termitario es un animal definido y compuesto, exactamente de la misma manera que el hombre lo es". (Maeterlinck, *The Life of the White Ant*)

La noción de complejidad quizás esté más claramente ejemplificada en los fenómenos sociales puesto que exhiben simultáneamente, a escala macroscópica, al menos dos niveles de organización implicados: el individuo y la colectividad[1]. Las termitas australianas construyen termiteros que pueden alcanzar los 6 metros de alto con varios metros cuadrados de superficie y albergar a no menos de 300.000 a 500.000 habitantes. Normalmente, las construyen con dos lados planos orientados al este y al oeste para mantener la temperatura interior. Dado que a las termitas no les gusta la luz, construyen túneles por los que viajan para buscar comida y agua. Se sabe que en el desierto han llegado a cavar pozos de 30 metros de profundidad para conseguir agua.

No hay más que concluir que la información requerida para excavar un nido de hormigas queda distribuida en toda la colonia mediante su división fisiofuncional en grupos, edades y en las propiedades fisioquímicas de reacción ante estímulos específicos (Bert *et.al.*, 1990). La naturaleza no lineal de las interacciones y la fluctuación propia del comportamiento individual parece ser una manera efectiva de responder a un entorno impredecible en virtud de la flexibilidad de respuesta dada por la naturaleza probabilística y socialmente mediada de la reacción individual. Esto es, una inteligencia colectiva emerge a partir de las fluctuaciones amplificadas de un sistema lejos del equilibrio. "Una cadena de decisiones en una situación de inestabilidad canaliza un sistema emergente a una estructura global" (Solé & Miramontes *et.al.*, 1993).

1 Cabe plantearse, por ejemplo, considerar como estructuras grupos funcionales de hormigas que conformarían un estrato intermedio entre cada individuo y la totalidad del hormiguero.

A continuación se hará la descripción de la historia natural de una colonia de insectos sociales y sus diversas actividades colectivas para ilustrar los mecanismos de coordinación no centralizada que hacen de este conjunto de miles de unidades extremadamente simples un asombrosamente eficiente sistema de procesamiento distribuido y paralelo.

Una consideración importante antes de comenzar a describir el ciclo de vida de la especie *Pogonomyrmex barbatus* estudiada en Arizona por Deborah Gordon es la magnitud de la variabilidad etológica interespecífica en el ámbito del orden *Hymenoptera* que incluye abejas, avispas y hormigas. Inclusive dentro de la familia *Formicidae* la plétora de dietas y etologías que varían desde especies depredadoras hasta las que cultivan, pasando por las recolectoras; abarcando a lo largo de sus distintas especies la gama entera de actividades de sustento que nuestra especie ha desarrollado. Ante tal variabilidad, es importante notar que la información con que contamos proviene siempre del estudio de un tipo específico de insecto social, lo cual supone una de las mayores dificultades al estudiarlos. Por otra parte vemos que es posible encontrar más similitudes etológicas entre una colonia de hormigas en el desierto de Arizona y una de termitas del desierto de Sonora que entre dos especies de hormiga o de termita que, aunque mucho más cercanas filogenéticamente (las termitas pertenecen al orden *Isoptera* representan una divergencia en el grupo *Dictiyoptera* que comprende a las cucarachas y las mantis religiosas, *http://tolweb.org*), enfrentan condiciones de vida radicalmente distintas a partir del hábitat que ocupan (Gordon, 1999 y Smith, 2000). Este proceso de convergencia evolutiva en la orientación espacial permite que al representar la distribución espacial por comunicaciones químicamente mediadas, en el caso de los insectos sociales, sea razonable partir del común denominador a la mayoría de sus especies. En este sentido y desde el punto de vista del fenómeno de deposición y seguimiento feromonal nos referiremos en lo subsecuente a hormigas, termitas o insectos sociales indistintamente.

1.1. **Ciclo de vida de una colonia del desierto de Arizona**

Todo hormiguero tiene su origen en una sola hormiga alada que dejó su nido volando al encuentro de otros miles de insectos alados; machos y hembras fértiles que hicieron lo mismo de forma sincrónica obedeciendo, desde colonias

separadas, alguna señal que escapa nuestra percepción. De sobrevivir a un índice de mortandad del 99%, una hembra fecundada crea un hormiguero que contará al final del primer año con alrededor de 500 individuos y al cabo de 2 años, de forma más o menos constante, con hasta 10 000 hormigas y el 95% de probabilidades de sobrevivir de 10 a 20 años más (tantos como la reina). A lo largo de su vida (calculada en alrededor de un año para las trabajadoras y unas semanas para los machos) cada hormiga pasa de trabajar en las cámaras más profundas del hormiguero a las más superficiales y al exterior, donde se encuentran las hormigas más experimentadas (alrededor del 25% del total de la colonia). Entre las hormigas presentes al exterior del hormiguero (en general hormigas con sólo semanas más de vida) se pueden distinguir varios grupos asociados a diferentes funcionalidades (limpiar, cuidado del nido, exploración, abastecimiento). Las exploradoras son las primeras en activarse, más viejas e independientes respecto a los mecanismos de coordinación grupal; son también las que delimitan el área de abastecimiento de alimentos del día (Gordon, 1999).

1.2. **La comunicación entre los insectos sociales**

La alta complejidad de los sistemas de comunicación de los insectos sociales queda evidenciada en las tareas que llevan a cabo coordinadamente (Bert *et.al.*, 1990). Estos sistemas son la base sobre la que descansa el complejo sistema de organización social que caracteriza a estos insectos y que desde siempre ha fascinado al hombre, al punto de inspirar modelos del funcionamiento cerebral (Hoffstater, 1979). Por otra parte, hay también modelos para insectos sociales basados en redes neuronales (Solé *et. al.*, 1993). La analogía neurona-hormiga, al margen de diferencias claras como la conectividad "fija" del cerebro maduro contra la conectividad fluida del hormiguero, está sustentada en que ambas estructuras comparten una topología ramificada (Acosta *et.al*, 1993) altamente conexa y las características típicas del procesamiento paralelo y distribuido.

Se ha establecido que la mayoría de las hormigas tienen una visión limitada, por lo que la mayoría de sus interacciones se dan mediante señales químicas y somáticas. Cada hormiga posee hasta 14 glándulas distintas que secretan distintas sustancias, algunas se dispersan en el aire mientras que otras se acumulan sobre la superficie en la que andan (Gordon, 1999). Los distintos

mecanismos de coordinación en colonias de insectos sociales incluyen percusión, estridulaciones, frotamientos, intercambio antenal, palativo, gaseoso y señalamiento químico. Se han identificado más de una decena de categorías de reacciones suscitadas: alarma, atracción simple, reclutamiento, trofilaxis (intercambio de líquidos anal y oral), intercambio de partículas de comida sólidas, reconocimiento (entre especies y castas), marcadores territoriales, interacción sexual. Éstos habrían sido desarrollados mediante la llamada ritualización que consiste en la especialización y reinterpretación de comportamientos que previamente no tenían una función comunicativa, siendo ejemplo clásico la danza de las abejas, la cual reproduciría a menor escala el recorrido efectuado por la exploradora hasta la fuente de néctar.

En el caso de las hormigas la señal para advertir de la presencia de intrusos está asociada a los movimientos de ataque para repelerlos, en aquella para reclutar trabajadoras a una fuente de alimento se utilizan movimientos mandibulares que tendrían el propósito de liberar partículas de comida que detectarían las antenas de los receptores; para las señales químicas, parece sensato suponer su origen en la defensa (Bert *et.al.*, 1990). Dadas las características de longevidad y capacidad de procesamiento presentes en las hormigas, la comunicación química que permite un sistema de coordinación descentralizado resulta en un sistema muy diferente al humano (en el otro extremo de acuerdo a longevidad y capacidad de procesamiento neuronal), los mecanismos de interpretación de las señales que no dependen exclusivamente del mensaje emitido. Hay una sola señal química que reclutan trabajadores a una fuente de alimento o a terreno inexplorado (la diferencia radica en el comportamiento corporal de las emisoras). La señal química utilizada para promover la predación y para embestir contra intrusos es la misma en otro tipo de hormiga, *Oecophylla*. Otra sustancia es utilizada por *Solenopsis* (en distintas cantidades) tanto como medio de defensa como para mantener un ambiente aséptico para las crías. En comparación con la comunicación química, la comunicación acústica está poco desarrollada en los insectos sociales funcionando apenas como modulador de otras señales químicas de alarma y reclutamiento. En todos estos casos se utilizan vibraciones a través de un substrato y no transmisión acústica aérea (Bert *et.al.*, 1990).

Quizá el mecanismo más importante y ajeno a nuestra experiencia sea la comunicación químicamente mediada. Desde 1874 Thomas Belt estableció que las hormigas utilizaban señales químicas depositadas como medio de coordinación espacial. Queda por establecer todavía el origen y composición específica a cada marcador químico (Gotwald, 1995). De acuerdo a una clasificación común (Norlund 1981 en Bert *et.al.*, 1990) una feromona es un semioquímico (cualquier químico utilizado para fines de comunicación), usualmente una secreción glandular, utilizado intraespecíficamente. Sus efectos son de reacción cuando suscitan un cambio en el comportamiento de los individuos que las perciben, o primarios en el caso en que los individuos receptores presentan reacciones asociadas como la modificación de sus sistemas reproductivos o endocrinos.

Esta forma de comunicación se caracteriza por una alta economización en cuanto a su factura y transmisión. Se han registrado respuestas específicas a ciertas sustancias en cantidades apenas medibles; los descubridores de una feromona de marcaje de rutas en *Atta texana* estimaron que ...”un miligramo de ésta (lo que contendría en total una colonia completa)...sería suficiente para guiar una columna de hormigas tres veces alrededor de la tierra”. La desventaja de este tipo de comunicación es que la velocidad de desactivación de mensajes impediría la emisión rápida de una serie de ellos. La solución desarrollada por las colonias de hormigas consiste en respuestas graduadas a la intensidad de la concentración feromonal o a la combinación de dos o más sustancias (Bert *et.al.*, 1990). Tomando en cuenta esto y la clasificación de hasta 20 tipos distintos de feromonas, se tiene un campo de respuestas conductuales muy variado. Esto se complica aún más por la variedad en cantidad, composición y funcionalidad de las feromonas a través de distintas especies, que además se combinan con señales táctiles. Hay evidencia de que las señales más básicas (alarma o reclutamiento) fueron genéricas en un principio y posteriormente se les añadieron componentes específicos a especies y/o colonias, las cuáles se relacionan con las divisiones territoriales. Las hormigas son más agresivas en un territorio que hayan marcado previamente.

Otra característica de la comunicación químicamente mediada es que en muy pocas ocasiones está asociada con reacciones determinísticas por parte de los

individuos receptores. Estos parecen actuar sin estar forzados a la ejecución puntual de un comportamiento específico, sino más bien bajo una probabilidad alterada en su respuesta ante otros estímulos, es decir, mediante modificaciones a su estado motivacional. La respuesta conductual a una señal química varía de acuerdo a lo que el receptor se encuentre realizando. Se podría esperar este tipo de comportamiento en sociedades evolucionadas, donde muchos individuos llevan a cabo distintas tareas en distintos momentos. Esto se cataloga como comunicación modulada, basada en comportamientos que alteran la respuesta de los individuos, reduciendo la probabilidad de desviaciones en su comportamiento. En ocasiones, dos estímulos que provocan reacciones parecidas se combinan para provocar un efecto más fuerte (Gordon, 1999).

En el caso de la orientación espacial hay ejemplos de especies que siguen rastros formados por las glándulas pigoidial, glándula venenosa (presente en toda *Formiacea* pero con diferentes derivaciones funcionales), productos intestinales o la glándula de Dufou (Holldoeble *et.al*, en Gotwald, 1995) aunque en condiciones experimentales hormigas con los poros y glándulas bloqueadas han sido capaces de depositar marcadores espaciales, probablemente un marcador volátil no propiamente feromonal utilizado para otros fines y secretado por múltiples glándulas (Topoff *et.al.*, 1975 en Gotwald, 1995). Se observa una redundancia variable al punto que hay especies que siguen cualquier concentración química que no las repela.

Por lo general, las secreciones utilizadas por las hormigas en procesos de orientación espacial son altamente activas y duraderas. La estabilidad de los patrones feromonales depende principalmente del tipo de substrato, de si la especie tiene actividad superficial o solamente subterránea, la funcionalidad del marcaje espacial y la cantidad de lluvia a la que se expone. La especificidad del seguimiento del marcaje feromonal es variable, habiendo especies que pueden seguir el marcaje de otra especie y otras altamente especializadas sobre rastros de su propia colonia.

Hay hormigas que pueden responder a agentes químicos presentes sobre el cuerpo de otras. Existen también modos de seguimiento feroespacial

específicos a casta o edad. Las exploradoras no están tan atadas a la concentración depositada sino más bien al patrón de encuentros con otros individuos de la colonia (Gordon, 1999), aunque este comportamiento no se ha preservado en situaciones de laboratorio (Bert *et.al*., 1990). A pesar de su aparente simplicidad, cada insecto de una colonia es un sistema móvil con sensores sensibles a cambios químicos, mecánicos, térmicos y de humedad, lo cual favorece su capacidad de respuesta colectiva ante cambios ambientales, externos o autoinducidos (Theraulaz *et.al.* 2003).

Entre las hormigas soldado se ha observado el fenómeno de la marcha en círculo; las hormigas son incapaces de modificar este comportamiento y mueren de inanición (Shneirla, 1948) . Al ser arrastrado por el agua parte del rastro feromonal que las conectaba con el nido, las hormigas aisladas terminaron dando vueltas en lo que fue el borde de la congregación que se convirtió luego en dos discos independientes. Según los autores esta situación se debería a un conflicto entre el impulso de reanudar la exploración del terreno y aglomerarse. Esta condición ha sido reproducida en laboratorio en un espacio confinado mediante condiciones de homogeneidad espacial (Miramontes, comunicación personal) con termitas.

Debido a la complejidad del sistema de comunicación en insectos sociales es necesario restringir la atención a un aspecto de este fenómeno. El establecimiento de patrones feromonales en el espacio es favorecido, puesto que es un mecanismo observable *in situ* y experimentalmente en la coordinación de muchas de sus actividades a la vez que que los resultados de esta clase de modelo pueden ser evaluados intuitivamente en términos de un patrón de distribución espacial (Gotwald, 1995). Por esta razón se cuenta con investigaciones que plantean modelos de formación de patrones específicos a distintas situaciones: patrones de depredación en hormigas soldado (Deneubourg *et.al*., 1989; Watmough *et.al*., 1995), optimización de rutas ante obstáculos (Beckers *et.al*., 1992), formación de ramales en ángulos como estrategia de optimización (Acosta *et.al*., 1992) y otros que se ocupan de la formación de una "red vial" feromonalmente estructurada (Edelstein-Keshet, *et.al.,* 1995).

1.3. **El papel de la comunicación químicamente mediada en la orientación espacial**

Los mecanismos de comunicación feromonal en hormigas pueden coordinar cientos de miles de individuos sobre 1km cuadrado de terreno (Franks *et.al.*, 1991) y está mediado por niveles en la concentración feromonal superficial. El marcador químico es una sustancia volátil que se vuelve detectable estereoscópicamente por medio de los receptores en las antenas al volatilizarse en el aire. Lo sorprendente de este mecanismo es que la capacidad de respuesta de la colonia como un todo para sobrevivir a depredadores, alimentarse o realizar movimientos migratorios, dependería de ligeras fluctuaciones en el comportamiento individual (Edelstein-Keshet, 1994) en función de los gradientes locales de concentración feromonal, la estructura del espacio interno y externo al nido y las interacciones puntuales con otros individuos en ambos. Las modificaciones conductuales a nivel individual respecto al entorno feromonal están mediados por factores como la tasa de deposición y evaporación feromonal en distintos contextos químico-espaciales con distintas densidades de individuos y los correspondientes patrones de encuentro. Todos éstos, como también edad y casta, inciden en fluctuaciones al estado motivacional interno que alteran la respuesta de cada insecto, por ejemplo, una mayor o menor tasa de seguimiento feromonal. A pesar de ser estos parámetros, en principio, cuantificables y algunos de ellos estar documentados en la literatura (Edelstein-Keshet, 1994), hay una gran variación interespecífica en la importancia, contextos y combinación de estos factores.

Descrita en términos generales, en función de la concentración y en referencia a un solo individuo, la dinámica de orientación y formación de rutas consiste en la determinación de una dirección de desplazamiento por parte del insecto con base en la distribución feromonal. El insecto se desplaza a la nueva posición y deposita cierta cantidad de feromona (Deneubourg *et.al.*, 1989) modificando el propio rastro que sigue. El aparato perceptual de cualquier especie tiene limitaciones y distintos espectros perceptuales están abiertos a cada especie. Al principio la elección es casi al azar, en función de una capacidad limitada para detectar las pequeñas diferencias en la distribución feromonal (Watmough *et.al.*, 1993); no obstante, cada deposición modifica la probabilidad en la elección de la siguiente que transite por allí. A pesar de su simplicidad, este

mecanismo es factor importante en las complejas maniobras de hasta 200 000 hormigas soldado en falanges de 15 metros o más que pueden predar sobre un área de 1 km^2 en un solo día (Franks *et.al.*, 1991). Además, hay evidencia de que este mismo principio organizativo es suficientemente flexible como para explicar los distintos patrones de distribución espacial asociados a colonias de diferentes especies, que corresponden a la distribución específica de las especies que depreda cada una de aquellas. La sensibilidad de las hormigas a la feromona depositada puede ser estimada mediante el aumento en la velocidad media en función de la concentración (Franks *et.al.*, 1991); las colonias pueden detectar restos de ciertas redes feroespaciales después de hasta una semana (Franks *et.al.*, 1991).

El estudio de los insectos sociales tiene el interés adicional de que son nuestras competidoras en el uso de recursos. Esto nos ha llevado a una confrontación con algunas de sus especies pues además depredan también sobre árboles y campos de cultivo. El papel de las termitas dentro del orden ecológico es el de degradar la celulosa; sustancia orgánica nutricionalmente pobre, abundante y de difícil desintegración que le da estructura a las plantas. Curiosamente, las termitas no son en sí capaces de procesar la celulosa, sino que se encuentran en relación de mutualismo con "parásitos" intestinales que sí la procesan; de esta suerte la supervivencia de las termitas descansa sobre su asociación mutua y con una población huésped de protozoarios. Se estima que sin la actividad de las termitas en 50 años el 20% de los desiertos de la frontera de México y los E.E.U.U. estarían cubiertos por excremento de ganado y esqueletos de plantas fibrosas, lo cual derivaría en la desaparición de cantidad y variedad de plantas y en el eventual colapso del ecosistema en cuestión (Smith, 2000).

1.4. **Dinámica espacial a representarse**

La simulación que constituye el elemento central de este trabajo está basada en un experimento llevado a cabo con termitas (Miramontes, comunicación personal) que recrea en laboratorio las condiciones de homogeneidad espacial que inducen la ya citada procesión circular en contingentes de hormigas aisladas del rastro conducente al nido. La razón detrás de la elección de termitas es que con estos insectos la deposición feromonal cobra una mayor

importancia debido a que son ciegas y viven en oscuridad. Ambos tipos de insecto depositan feromona durante todo su desplazamiento, actividad que queda restringida a ciertas situaciones en el caso de la generalidad de las hormigas. Una característica importante en cuanto al uso de feromonas como indicador espacial es que existen dos mecanismos para seguir un rastro: la detección de la sustancia volatilizada en el aire mediante sus antenas, o directamente sobre el substrato utilizando sus sensores palativos (Tschinkl *et.al.*, 1973 en Dumpert, 1981). El último, llamado orientación clinotáctil, consiste en la comparación de varias muestras y la elección del camino en función del máximo detectado mediante "el palpar del área impregnada"; el primero, llamado osmotropotaxis, implica a los organismos con órganos sensoriales bilaterales percibiendo simultáneamente dos áreas y, midiendo la diferencia, ajustar la posición para procurar una intensidad sensorial simétrica en ambos, lo cual significa seguir sobre el camino. Se ha encontrado evidencia de uso de ambos métodos en distintas situaciones y especies de hormigas (Dumpert, 1981). Lo mismo pareciera ocurrir en termitas.

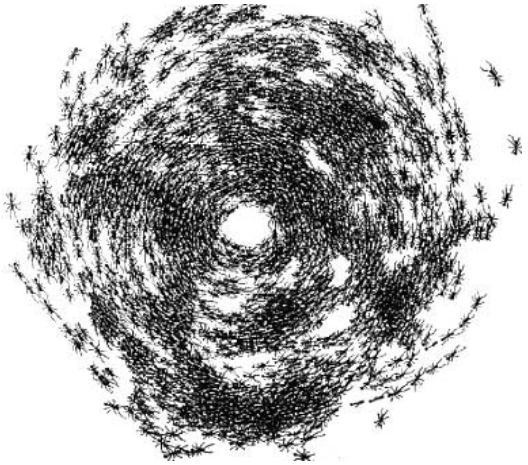


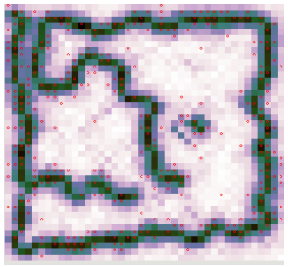
fig 1.1: patrón de marcha circular

www.filterfine.com/images/ants.jpg

El experimento que se representa con este esquema de simulación computacional consistió en ingresar a un número variable de termitas a distintas cajas con las mismas dimensiones y tratadas de forma que las termitas permanecieran sobre la superficie horizontal. Al transcurrir una noche se observó que a partir de un número crítico de termitas en el contenedor los grupos habían formado un anillo cerrado alrededor del cual se movían. Hay dos diferencias importantes entre las condiciones experimentales y el fenómeno

observado en la naturaleza; una está dada por las condiciones de frontera impuestas por los bordes del contenedor en laboratorio, otra es que el punto de partida para el fenómeno natural no es de homogeneidad espacial puesto que al quedar aislado del nido un grupo está ya montado sobre su rastro feromonal, con la configuración espacial asociada. Queda advertir que el contexto social en el que un comportamiento ocurre es fundamental para sus causas y evolución; los experimentos en laboratorio inevitable y tácitamente asumen que se puede siempre muestrear un número significativo de veces respecto a la variedad de las situaciones posibles que pudieran surgir naturalmente, con lo cual se podría subestimar el rango de experiencias posibles para un ser vivo (Esté, 1997). Al ser la experiencia de la termita principalmente táctil y química y estar determinado su éxito como colonia en los patrones o redes de movimiento, parece ser un buen guía en la indagación de la relación entre ambos.

CAPÍTULO 2.
MODELO HEURÍSTICO



...los sistemas vivos son fundamentalmente
sistemas fuera del equilibrio, los patrones emergen y se
mantienen de forma relativamente autónoma
(Scott-Kelso, Haken en Murphy et. al. (eds.) 1997)

En este capítulo se discuten las características dinámicas de la deposición, elección direccional a nivel individual y otras consideraciones que determinan la plasticidad de las configuraciones feroespaciales emergentes. Los patrones espaciales de movimiento en una especie están directamente relacionados con su comportamiento, de manera que se pueden considerar como un campo de análisis importante en relación a su etología. El marcaje espacial no es específico a las termitas; se han encontrado orugas, moluscos y hasta mamíferos que emplean feromonas con tal fin (Edelstein *et.al.*, 1995). Incluso en el caso humano, la conformación de brechas en el medio rural se da de forma estigmérgica, es decir, como resultado de la acción colectiva y no planificada a través de la modificaciones puntuales al entorno introducidas por el quehacer de cada individuo, no necesariamente químicas[2]. Este mecanismo presenta altos niveles de optimización que derivan en rutas óptimas (Moura, 2002).

La hipótesis de trabajo es que los mecanismos de deposición, respuesta y seguimiento respecto al gradiente feromonal local son los factores principales en el establecimiento de la dinámica que genera los patrones espaciales observados; aún cuando para operaciones espaciales más complejas (por ejemplo migración, depredación) otros mecanismos de coordinación, en particular de interacción entre individuos, serían cruciales.

2 Es sabido que los indios norteamericanos eran capaces de seguir el rastro dejado por una persona incluso con varios días de diferencia y desarrollaron también estrategias tanto para hacer sus pasos evidentes como también para eliminar su rastro.

2.1. **Antecedentes teóricos del modelo**

Sin duda alguna, el sistema social es una organización, al igual que el individuo, cohesionada por un sistema de comunicación e imbuida en una dinámica en la que los procesos circulares de retroalimentación tienen un papel de suma importancia. La retroalimentación es una clara y dramática diferencia entre los mecanismos cartesianos y cibernéticos, introduce la irreversibilidad en procesos donde el propio mecanismo se ve transmutado por su propio actuar. Los ciclos de retroalimentación son una característica fundamental, omnipresente en los patrones de redes no lineales de los sistemas vivos.

En los ochentas se dio no tanto el desarrollo formal de una Teoría de Sistemas sino el de una serie de modelos sistémicos basados en las nociones de retroalimentación y autorregulación, aplicados exitosamente a aspectos específicos en la biología, psicología cognitiva y cibernética. Estos modelos surgen alrededor del concepto de autorregulación, tal vez el concepto central de la visión sistémica, íntimamente ligado al de redes. En muchos de estas abstracciones, de un estado aleatorio elegido al azar surgían espontáneamente patrones ordenados. A la emergencia espontánea de orden se le denominó autorganización. Esta propiedad permite la aparición espontánea de nuevas estructuras y dinámicas de comportamiento en sistemas lejos del equilibrio, se puede caracterizar mediante ciclos de retroalimentación interna o en términos de ecuaciones diferenciales no lineales.

2.2. **Descripción del modelo propuesto**

El modelo propuesto se ocupa de un fenómeno que, si bien fue reproducido en laboratorio bajo condiciones controladas, no ha podido ser cabalmente caracterizado y cuantificado en cuanto a los parámetros interactuantes. Las tasas de evaporación o deposición feromonal son valores altamente cambiantes tanto en función del substrato en que se depositan como de la especie del insecto. Además, no existe una clara correspondencia entre la función y composición de una feromona específica siendo que el contexto en el que ocurre la deposición media sobre el efecto de estas sustancias; por lo general secretadas en diversas situaciones para distintos fines e incluso combinadas entre sí (Bert *et.al.*, 1990).

El modelo diverge del comportamiento observado en termitas, aún en laboratorio, en varios aspectos: por ejemplo, no se han incluido variaciones en la velocidad dependiendo de la concentración feromonal. Lo observado es un patrón espacial y en este trabajo se explora la posibilidad de representarlo en términos de unas reglas de seguimiento propuestas para una señal química. En estas condiciones un autómata celular se perfila como el formalismo más adecuado pues, una vez que se cuenta con el esquema básico para la simulación, permite explorar una amplia gama de reglas, parámetros y situaciones con relativa facilidad y rapidez, además de proveer una representación natural de un espacio rectangular plano.

El punto de partida es la revisión de los modelos propuestos anteriormente para problemas similares. En la simulación que se presenta, las reglas fueron implementadas con base, tanto en modelos planteados anteriormente, como en la descripción fenomenológica de entomólogos. En esos casos se reproducen y relatan situaciones (elección de dirección en una bifurcación, determinación de la ruta más corta, formación de caminos de rastreo de alimento) típicas para colonias de hormigas en su medio natural, y no parten de un experimento controlado ni analizan la configuración espacial total después de un espacio temporal "largo", en esta situación las condiciones de frontera devuelven el rastro feromonal sobre sí mismo lo cual la aleja de las situaciones que naturalmente experimentan estos insectos. La procesión anular descrita es un patrón degenerado por la homogeneidad espacial inducida. Con los resultados del programa de simulación se evaluará si una dinámica feromonal simple es suficiente para dar cuenta de los patrones espaciales observados.

Los autómatas celulares se han utilizado para modelar aspectos de dinámica espacial en fenómenos pertenecientes a todos los niveles de organización biológica, desde el bioquímico hasta interacciones entre individuos y aún entre especies; medios excitables, diferenciación celular durante la morfogénesis, actividad eléctrica en la corteza cerebral, relaciones parásito huésped, sistemas de reacción difusión, reacciones autocatalíticas, modelos depredador presa, osciladores acoplados, modelos inmunológicos (Ermoentrout *et.al.*, 1993).

La malla cuadriculada se convierte en la superficie plana y limitada del experimento y las termitas en cuestión se convierten en el conjunto de partículas sobre la malla. Dependiendo la cantidad de parámetros interactuantes y las dificultades inherentes a su cuantificación es generalmente difícil, en los ámbitos biológico y social, generar modelos continuos que reflejen relaciones cuantitativas detalladas entre los parámetros sin acabar con un sistema de ecuaciones difícilmente analizable o tan simple que sea imposible extrapolar los resultados a situaciones reales. El que se trate de fenómenos que no están detalladamente cuantificados permite, no obstante, un tipo de análisis cualitativo que evalúa la relación de factores hipotéticos involucrados con los patrones observados. Éstos factores se pueden así saber correlacionados y ciertas inferencias cualitativas pueden ser hechas sin necesidad, o ante la falta de oportunidad, de representar el sistema cuantitativamente. Al utilizar metodologías analíticas existe el riesgo de perder el fenómeno biológico de vista a cambio de lidiar con un problema difícil de cálculo numérico o una descripción dinámica en la que sea difícil distinguir los principios físicos básicos operando (Ermoentrout *et.al.*, 1993). Kauffman y sus colegas utilizaron redes binarias para representar sistemas enormemente complejos, cuya descripción analítica requeriría de un sistema de ecuaciones con miles de variables acopladas. En cierto sentido, los dos formalismos, ecuaciones diferenciales y autómatas celulares pueden ser concebidos como correspondientes a las dos dimensiones conceptuales de estructura y patrón en la teoría de los sistemas vivos. En ocasiones no resulta obvia la distinción entre un autómata de estados finitos y una simulación numérica, excepto que en el caso de los autómatas la discretización del tiempo,espacio y estado es siempre explícita (Ermoentrout *et.al.*, 1993).

Es importante enfatizar que el papel de los modelos es complementario al trabajo empírico encaminado a corroborar las implicaciones sugeridas por los estos e inspirar nuevos que evalúen diferentes aspectos de la cuestión (Gordon, 1999:142). Conviene tomar esto en cuenta para evitar la común situación en que "...los modelos tienden a ser tan generales que no pueden hacer predicciones concretas respecto a sistemas específicos o tan particulares que resultan inaplicables a la realidad fenomenológica del problema. (Levin en Gordon,1999:149).

2.3. **Características funcionales específicas**

El sistema de software de simulación (que permite y constituye el núcleo de este ejercicio de modelaje biológico) es una implementación distribuida y orientada a objetos de un autómata celular basada en el comportamiento individual de las partículas (en este caso insectos) que lo componen. El modelaje biológico basado en formalismos discretos (redes neurales, algoritmos genéticos, autómatas celulares) se ha popularizado por tratarse de un mecanismo en que poder plantear y evaluar cualitativamente hipótesis en muy distintos contextos. Como autómata celular, esta simulación se sitúa dentro de la clasificación de modelo de partículas de gas, consistentes en una malla rectangular y un conjunto de partículas que se mueven e interactúan de acuerdo a ciertas reglas; a diferencia de los modelos determinísticos, estos autómatas operan bajo reglas dependientes de valores probabilísticos. Estos modelos exhiben un tipo de organización colectiva que puede ser representada mediante sistemas dinámicos continuos, en ocasiones más simples, pero a riesgo de perder información espacial explícita (Ermoentrout *et.al.*, 1993).

2.4. **Parámetros involucrados**

En cada turno cada termita es elegida en orden aleatorio. Esta estrategia alude a la cuestión de sincronía respecto a un paso de tiempo discreto. Por lo general al implementar un autómata celular se sigue la estrategia de mantener en memoria dos gradillas de estado (t, t+1). Esto se hace para calcular el siguiente estado de cada individuo (o celda) con base en el anterior pero sin tomar en cuenta las modificaciones que inducirían la transiciones de estado en las celdas ya computadas. Esto se hace necesario en virtud de la naturaleza estrictamente secuencial del procesamiento digital, es decir, se utiliza una estrategia de modificación pseudosimultánea de los estados individuales. La implementación del sistema de simulación digital en cuestión permite, en cambio, que una gradilla se vea afectada por cada modificación individual previa, de manera que cada individuo involucrado es influenciado por el cambio de estado inducido por los precedentes. Aquí la estrategia seguida es la de sortear el orden en el que cada individuo efectúa un movimiento, no se intenta reproducir un régimen de simultaneidad sino de asincronía en la que cada individuo actúa en un momento distinto siguiendo un orden aleatorio.

El algoritmo de desplazamiento es sencillo, en primer lugar el individuo detecta los valores feromonales de las casillas pertenecientes a su vecindad perceptual, de acuerdo a una función probabilística en términos de la concentración local, la respuesta feromonal, se aplica o no la regla de seguimiento. Esto resulta en un desplazamiento a una nueva casilla con una orientación específica; si la casilla no está ocupada ingresará el insecto, en caso contrario girará en una nueva dirección para, en el siguiente turno, aplicar nuevamente el algoritmo.

2.4.1 **Evaporación, deposición y respuesta feromonal**

La consideración central de esta representación digital es la capacidad de los insectos para seguir y modificar un rastro feromonal. Estas consideraciones se encuentran codificadas en la clase o tipo principal del programa **Termita** que conforma la simulación y que representa el funcionamiento de cada individuo, está constituida, esencialmente, por un objeto tipo **Rumbo** que le confiere una orientación específica y las reglas de transición que llevan a una **termita** de un **sitio** de la gradilla a otro. Se tiene una tasa fija de evaporación ξ por unidad de tiempo, ζ es el valor constante de deposición feromonal fijo[3] y $\delta_{y,x}(n)$ es cero o uno dependiendo de si hay o no una termita presente en la casilla en ese lapso temporal. De aquí que el cambio iterativo en la concentración local $\chi_{y,x}(n+1)$ para una casilla arbitraria sea:

$$\chi_{y,x}(n+1) = (1 - \xi)\, \chi_{y,x}(n) + \zeta\delta_{y,x}(n)$$

El desplazamiento de cada isóptero Υ_i por unidad temporal depende de la concentración local promedio en su vecindad perceptual, $\nu(\Upsilon_{i\,(y,x)}) = \{ (y,x)+(v,w) \mid v=0,1;\ w=-1,0,1\}$, de manera sigmoidal.

3 Se sabe que el contexto sociomotivacional modifica la tasa de deposición individual; por ejemplo, con el hallazgo de una fuente de alimento se aumenta considerablemente la tasa de deposición. Sin embargo, considerando que aquí se reproduce una situación espacial homogénea se justifica una tasa fija de deposición.

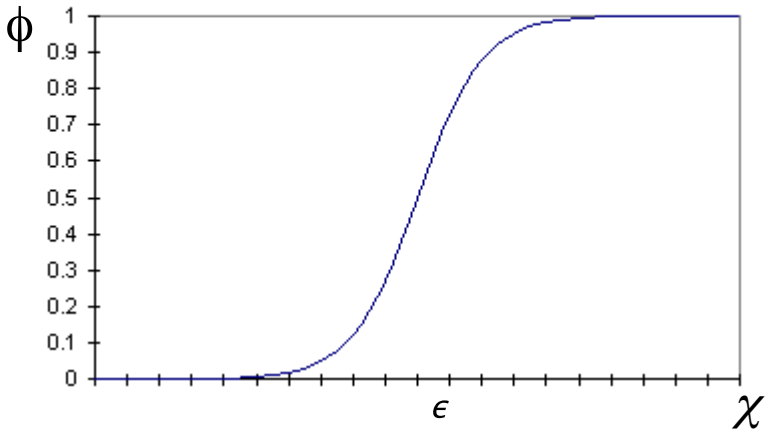


fig 2.1: responsividad feromonal respecto a la concentración local

La función de respuesta feromonal no lineal ϕ (figura 2.1) se refiere a la respuesta feromonal de las termitas ante el gradiente de concentración en su vecindad perceptual; la probabilidad con la que cada termita se desplazará de acuerdo al algoritmo de seguimiento feromonal, este parámetro representa el estado motivacional de cada individuo. La constante ϵ determina el punto de inflexión, α la pendiente de la tangente en este punto; en este modelo el valor de ϵ regula la respuesta feromonal mínima, la probabilidad de seguimiento cuando la concentración feromonal es cero. Mientras tanto, α determina la pendiente de la tangente a este punto, es decir la velocidad de crecimiento de la curva. En el modelo discreto este parámetro es representado como la taza de deposición feromonal por turno e individuo. Estas consideraciones quedan plasmadas en la siguiente fórmula para el insecto Υ en el sitio (y,x) con vecindad perceptual ν, con concentración feromonal media χ.

$$\phi(\chi(\nu(\Upsilon_{(y,x)}))) = 1/ (1 + e^{-\alpha(\chi-\epsilon)})$$

2.4.2 **Las reglas de movimiento**

La vecindad perceptual de un termita está definida partiendo de su orientación espacial. La segunda parte del algoritmo define la regla de elección de una casilla en función de la distribución de valores feromonales en las casillas de la vecindad perceptual.

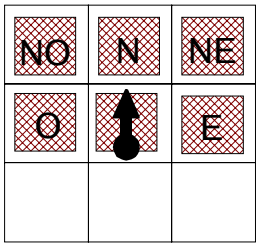


fig 2.2: Vecindad perceptual de una termita

La regla de seguimiento que intenta capturar el mecanismo referido en (Dumpert, 1981) como osmotropotaxis, se basa en la medición de las diferencias perceptuales en las concentraciones del "área izquierda" y "derecha" percibidas por cada antena para después dirigirse al **sitio** adyacente en esa dirección.

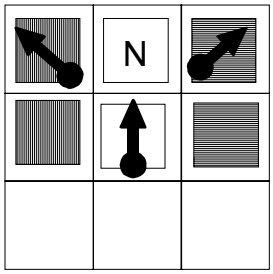
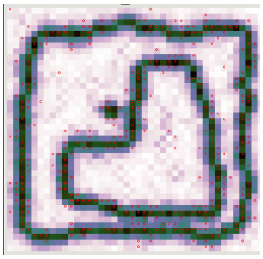


fig 2.3 Regla osmotrópica

Operando con la última regla, basada en el mecanismo clinotáctil, una **termita** se dirige al **sitio** que tenga la máxima concentración feromonal.

CAPÍTULO 3.
IMPLEMENTACIÓN DIGITAL



¿Qué es lo que gobierna aquí?
J.M. Maeterlinck
L'ouvrier ne dirige pas son travail,
il est guidé par lui. Grassé

La implementación, como simulación digital, del modelo conceptual descrito en el capítulo anterior constituye la infraestructura experimental de este trabajo. Esta implementación se describe a partir de su clasificación como un sistema de cómputo paralelo orientado a objetos. Como apéndice se incluyen el código fuente y documentación para instalar, ejecutar o modificar el sistema.

3.1. Procesamiento paralelo o distribuido

El cómputo distribuido se refiere a aquellos sistemas que dependen de subprocesos que se llevan a cabo en varias computadoras pero que en conjunto integran uno sólo; por ejemplo, un juego de ajedrez en línea, que involucra dos programas cliente y un servidor. En el caso del cómputo paralelo tenemos una situación similar, con la salvedad de que, generalmente, cada nodo de la simulación está ejecutando el mismo código, o más estrictamente, la ejecución simultánea de partes complementarias de un sólo proceso. En última instancia, no es posible trazar una delimitación tajante entre estos dos tipos de sistema, por lo general, el procesamiento paralelo tiene la simultaneidad de la ejecución en cada nodo como prioridad mientras que en un proceso distribuido está enfatizada la posibilidad misma de efectuar una operación que incluya distintos nodos.

Los sistemas de procesamiento paralelo eran ejecutados casi exclusivamente en costosas y enormes máquinas con varios procesadores que, en la mayoría de los casos, compartían una sola memoria. El costo del cómputo en paralelo fue abatido con el surgimiento de *clusters* formados mediante una red local de computadoras interconectadas mediante un *switch* de alta velocidad, por lo que el cómputo paralelo está abierto a un grupo menos exclusivo de programadores.

La implementación como sistema paralelo responde a que algunos escenarios en el ciclo de vida de los insectos sociales implican la interacción de una gran cantidad de elementos sobre una escala espacial y temporal “grande” con geometrías variables. Ésto crea la necesidad de comunicación bidireccional entre varios nodos, para permitir el tránsito de las ***termitas*** entre ellos***.*** Un nodo es un hilo de ejecución que puede intercambiar datos con otros; la comunicación puede ocurrir en la memoria de la misma máquina, como en el caso de una máquina con varios procesadores donde cada procesador ejecuta un hilo, o a distancia, a través de un *switch* en una red (local *LAN* o virtual *VLAN*) o en un *cluster*.

Por otra parte, la paralelización implica la disociación de las funciones de visualización y procesamiento, es decir, la necesidad de un nodo especializado en representar gráficamente la progresión de estados numéricos cambiantes de los demás nodos, es decir, la simulación digital. Esto deviene en otra relación vía red con un visualizador, o ***interfaz***, que a petición exhiba el estado de cada nodo. Por esto, considerando la relación entre la ***interfaz*** de visualización y los procesos activos en los nodos que conforman la simulación tenemos un sistema distribuido con un cliente ligero que se conecta a distintos servidores para obtener los datos que despliega gráficamente. Estas necesidades de procesamiento distribuido definieron la arquitectura del sistema se muestra en la siguiente figura:

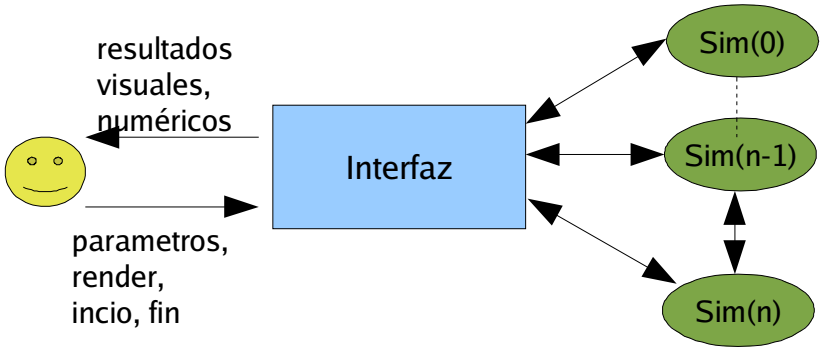


fig 3.1: esquema de interaccion con el sistema

3.2.**Clasificación y arquitectura**

En este caso, tenemos un sistema de procesamiento paralelo desde el punto de vista de la simulación y distribuido desde la perspectiva de su visualización.

Los nodos procesan sincrónicamente las mismas operaciones sobre una colección cualitativamente homogénea de datos ordenados y puesto que operan casi independientemente sobre datos locales, comunicándose entre sí resultados parciales, se puede ubicar este sistema dentro de la categoría de paralelismo de dominio con componentes homogéneos (Ortega, 1998). Un patrón de diseño para la arquitectura de esta categoría de problema es llamado Elementos Secuenciales Comunicantes que consiste de una red de nodos con una estructura espacial que refleja la del dominio del problema. Este patrón consta de elementos secuenciales (nodos de funcionamiento idéntico y simultáneo) y canales de comunicación entre ellos.

Con el uso de este patrón de diseño cada nodo ejecuta un ciclo operacional, comunica sus resultados parciales y se sincronizan todos para luego continuar con el siguiente ciclo (Ortega, 1998). Durante la ejecución del código estos procesadores intercambian mensajes asíncronos relativos a la información feromonal en los ***sitios*** frontera y al tránsito internodal de las ***tremitas***, denominadas así para evitar ambigüedad respecto a su contraparte no virtual. Al finalizar cada ciclo intercambian mensajes de sincronización. El esquema de procesamiento con el nodo visualizador desactivado se muestra a continuación:

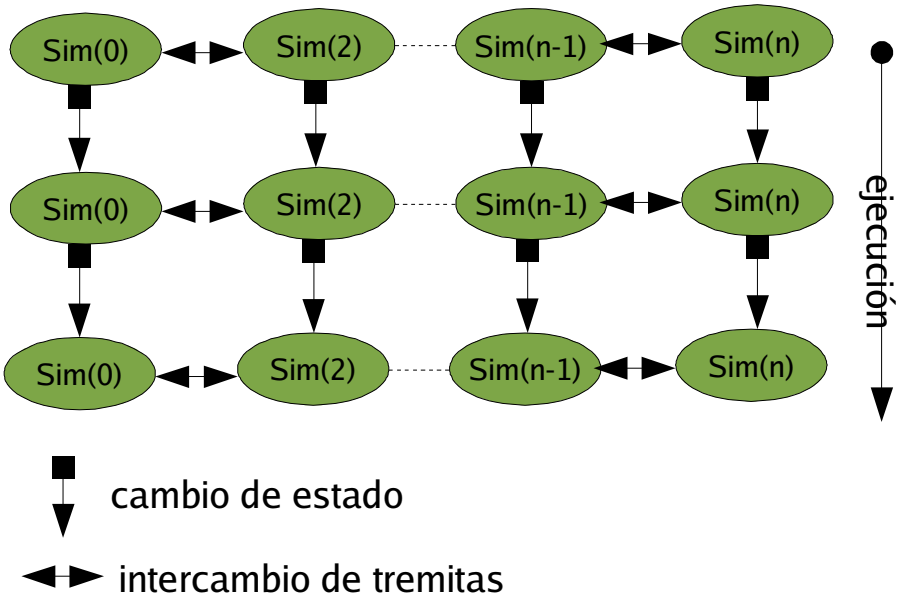


fig 3.2: esquema de ejecución nodal en paralelo

3.3. **La programación orientada a objetos**

Los lenguajes con orientados a objetos han crecido en popularidad tanto en el ámbito académico como industrial por dos razones fundamentales. La primera es la facilidad para definir nuevos tipos de datos así como operaciones sobre éstos lo cual deriva en que el código orientado a objetos es más legible y más cercano al problema modelado, lo cual facilita su ampliación o modificación subsecuente. Esto se debe a que las distintas funcionalidades del sistema quedan agrupadas en distintas clases, cuya definición y comportamiento pueden ser modificadas independientemente en tanto se mantenga coherencia en relación a los mensajes que intercambian entre sí objetos de distintas clases. Por otra parte, el lenguaje *Java* cuenta con grandes bibliotecas estándar que permiten al programador diseñar programas que se comuniquen a través de distintas computadoras o que generen interfaces de interacción en modo gráfico sin tener que programar cada detalle o utilizar bibliotecas particulares que sólo funcionen en un cierto sistema operativo o para un fin particular muy específico. La dinámica computacional en tiempo de ejecución para un sistema escrito en un lenguaje orientado a objetos está determinada por los distintos objetos en memoria que se solicitan y envían información mediante un protocolo de métodos o funciones. Cada objeto definido por una clase cuenta con métodos (o funciones) que son o no "visibles" para otros objetos dependiendo del nivel de visibilidad definido para el método. Puede ser visible para todos los objetos del sistema (método público), para aquellos cuyas definición de clase pertenece al mismo grupo o sólo para objetos de la misma clase (método privado). De esta manera otro objeto que cuente con una referencia al primero puede invocar sus métodos visibles, llevando a cabo entre todos una tarea que al ser descompuesta en intercambios y transformaciones graduales de información pasando de objeto a objeto es más comprensible en la medida que las clases que definen los objetos emulan el dominio de problema.

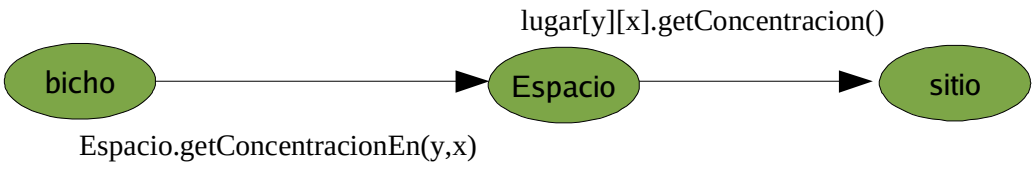


fig 3.3: intercambio de mensajes entre objetos

Cada uno de los “objetos” se crea a partir de su definición como tipo o miembro de una clase y tiene un estado interno cambiante que está determinado por los valores de sus variables que pueden ser tanto valores numéricos así como otros objetos. Cada objeto también se ve determinado por las operaciones definidas para él, lo cual define su papel en la dinámica de intercambio de mensajes con el resto de objetos en el sistema.

El reuso y la extensibilidad de las clases se tienen como una de las ventajas de este paradigma de programación puesto que la característica principal de la orientación a objetos es la descomposición del dominio teórico en unidades conceptuales relacionadas que posteriormente son mapeadas a distintas definiciones de clase. Lo anterior facilita la elaboración de programas pues la sintaxis del código fuente se sigue de las relaciones existentes en el problema modelado. En otras palabras, durante la elaboración del sistema y a partir de un buen diseño, el programador trabaja en los términos del problema a modelar.

3.4. **El uso de RMI (***Remote Method Invocation)*

El paralelismo del sistema está basado en la partición del dominio espacial del problema -la cuadrícula de **sitios** del autómata celular- distribuidos en subgradillas, agrupados en objetos de clase **Espacio**, residentes en cada procesador; las **tremitas** quedan repartidas y son intercambiadas entre objetos **Simulador** que interconectan las subcuadrículas albergadas en cada nodo de un *cluster*.

La distribución del procesamiento sobre varios nodos fue desarrollada a partir de las bibliotecas de *Java* para la invocación de métodos para objetos remotos, *RMI,* la cual proporciona referencias a objetos presentes en la memoria de otra computadora de manera que es posible invocar métodos definidos para estos desde otro nodo de la red como si se tratara de un objeto en la memoria local.

Esta biblioteca provee las funciones necesarias para exportar un objeto, hacerlo visible desde otros nodos y obtener referencias a éste. De esta manera la **interfaz** así como cada **simulador** cuenta con referencias a los demás, pudiéndose comunicar con estos para coordinar su funcionamiento como un proceso único.

El protocolo aludido, como MPI o PVM, está basado en el paradigma de intercambio de mensajes, constituido por tres capas. La primera es una definición de clase vacía que sirve sólo como mapa lógico del objeto solicitado pues declara qué métodos exporta. Esta definición es parte del código en el nodo del objeto solicitante. En tiempo de ejecución las llamadas son canalizadas a un objeto puramente referencial o *stub*, perteneciente a la siguiente capa, *stub layer*, de implementación de la biblioteca. Los *stubs* suplantan a los objetos que en el caso de un programa de ejecución local se generarían en memoria a partir de la definición de clase. Su papel es el de transmitir las invocaciones hechas a partir de la definición de clase vacía a la siguiente capa (*remote reference layer)*, la cual maneja la lógica de creación e intercambio de referencias a objetos remotos, la última capa (*transfer protocol layer)* se encarga de transferir de un extremo a otro los módulos con el código binario que representa las referencias y valores paramétricos implicados en las capas anteriores (Flanagan *et.al.*, 1999). Las últimas dos capas de comunicación conciernen no al sistema desarrollado aquí sino al funcionamiento interno de las bibliotecas utilizadas.

3.5. **División lógica y funcional del sistema**

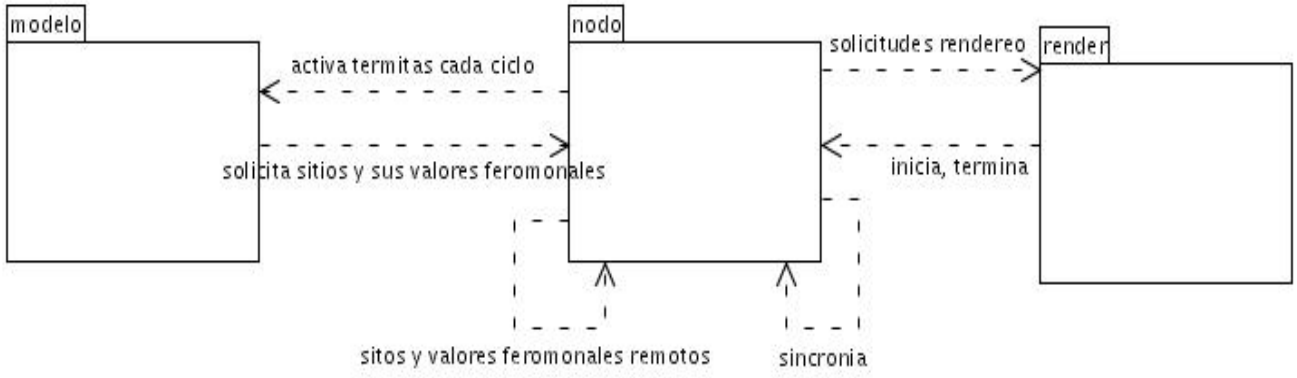


fig. 3.4 agrupación funcional de las clases del sistema: visualización, ejecución y lógica del modelo

Las consideraciones logísticas y de sincronía relativas a la implementación del procesamiento distribuido y paralelo generaron las clases agrupadas en los módulos de clases **simula** y **render**. Las clases que codifican la lógica funcional del modelo teórico; estas clases fueron clasificadas en **modelo**.

El módulo **render** se relaciona con el módulo **simula** a partir de sus funciones de parametrizar, inicializar, conectarse, desconectarse y finalizar la ejecución de cada nodo, es decir, para permitir la interacción del usuario con la simulación. El módulo **simula** se relaciona con el módulo **render** a partir de la funcionalidad de la **interfaz** de representar el estado de cada **simulador**. **Simula** se relaciona con **modelo** por el hecho de que cada nodo en cada uno de sus ciclos activa a cada **tremita** para la ejecución de su algoritmo de movimiento. Finalmente **simula** aparece relacionado consigo mismo por la funcionalidad de sincronizar los nodos de la simulación. Así cada nodo contacta a sus vecinos al cabo de cada ciclo de ejecución para mantenerse en sincronía con ellos. El **modelo** esta relacionado con **simula** a partir de las llamadas que las **tremitas** hacen sobre el **espacio** tanto para obtener la concentración de un sitio determinado como para obtener una referencia a un **sitio** y gestionar su movimiento a éste. Gracias al uso de las librerías de *RMI* no existe una diferencia en función de las relaciones recién mencionadas respecto a si el **sitio** en cuestión pertenece a la subgradilla local o a la de un nodo vecino. La figura 3.5 muestra el agrupamiento de clases en los distintos nodos en tiempo de ejecución.

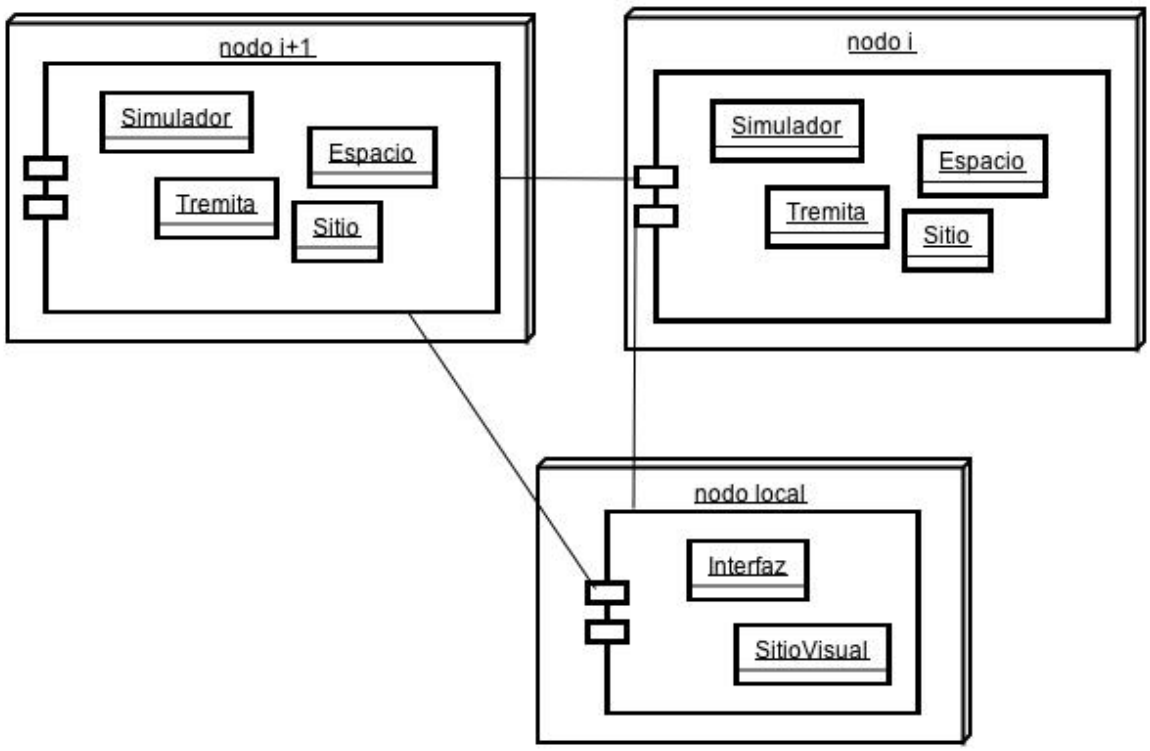


fig 3.5: distribución de objetos en tiempo de ejecución

Quedan así separadas las clases que describen la lógica del modelo y aquellas necesarias para su implementación. Esto es importante puesto que modulariza las modificaciones inevitables durante la exploración del espacio paramétrico en la clase **tremita** sin que las modificaciones a las reglas de movimiento afecte la lógica de implementación del esquema desarrollado y viceversa.

3.5.1 *Render*

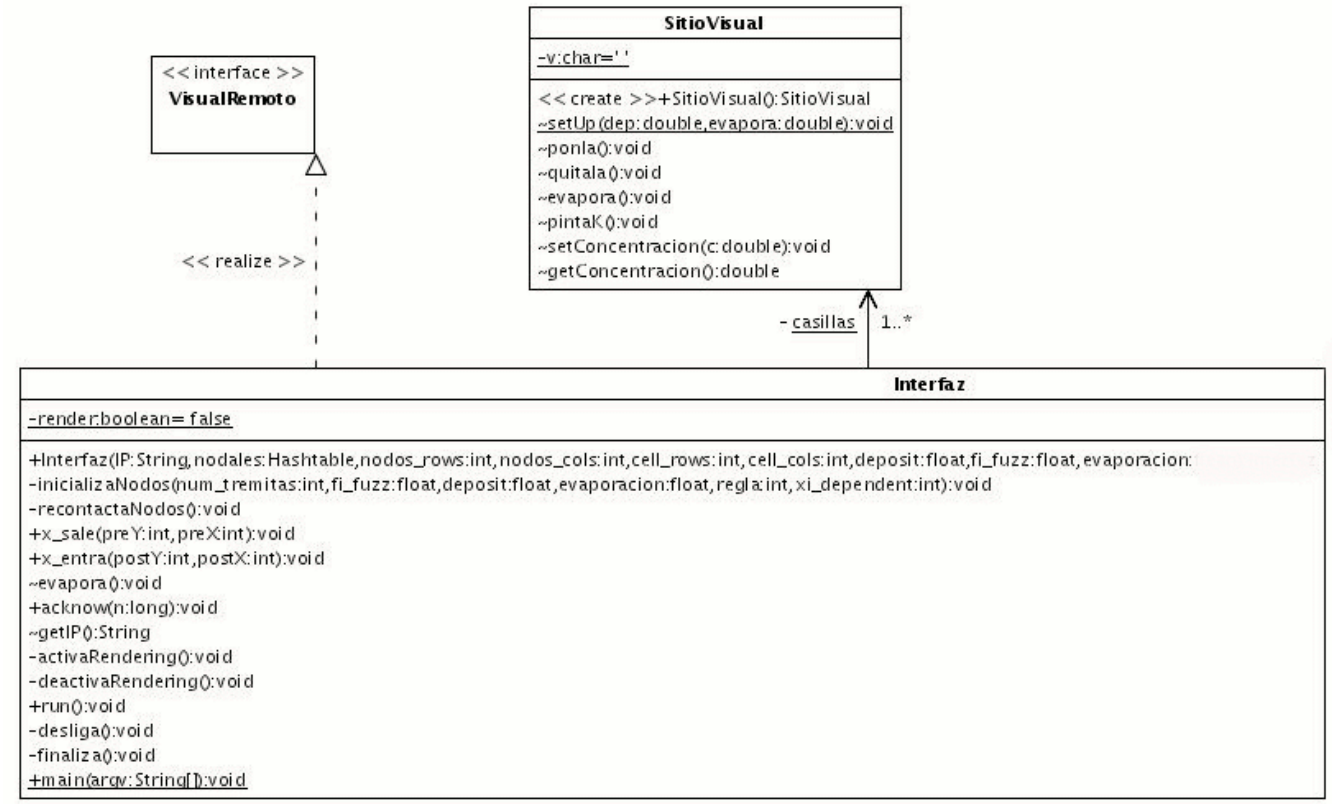


fig 3.6: clases del módulo de visualización

Se puede considerar al procesador a cargo de la ejecución de este módulo como una suerte de nodo maestro por el hecho de parametrizar, dar inicio, fin y visualización a los nodos. La figura anterior muestra las relaciones al interior de este módulo, conformado por las clases **Interfaz**, **SitioVisual** y la clase vacía **VisualRemoto**, necesaria para definir las operaciones invocables remotamente.

Interfaz es la capa externa de la simulación, a esta clase pertenece el objeto del sistema con el cual se interactúa directamente tanto para dar inicio, ver el estado y finalizar una simulación. Reúne los parámetros definidos en un archivo

de configuración, los transfiere a cada nodo y recibe peticiones de representar visualmente el movimiento de las **tremitas** y la concentración feromonal de cada **sitio**. Para esto se vale de la clase **SitioVisual**, que es una clase que permite desplegar caracteres (un símbolo es igual a un **tremita**) y cambiar el color de fondo (representando con la intensidad del color el nivel de concentración feromonal).

3.5.2 **Simula**

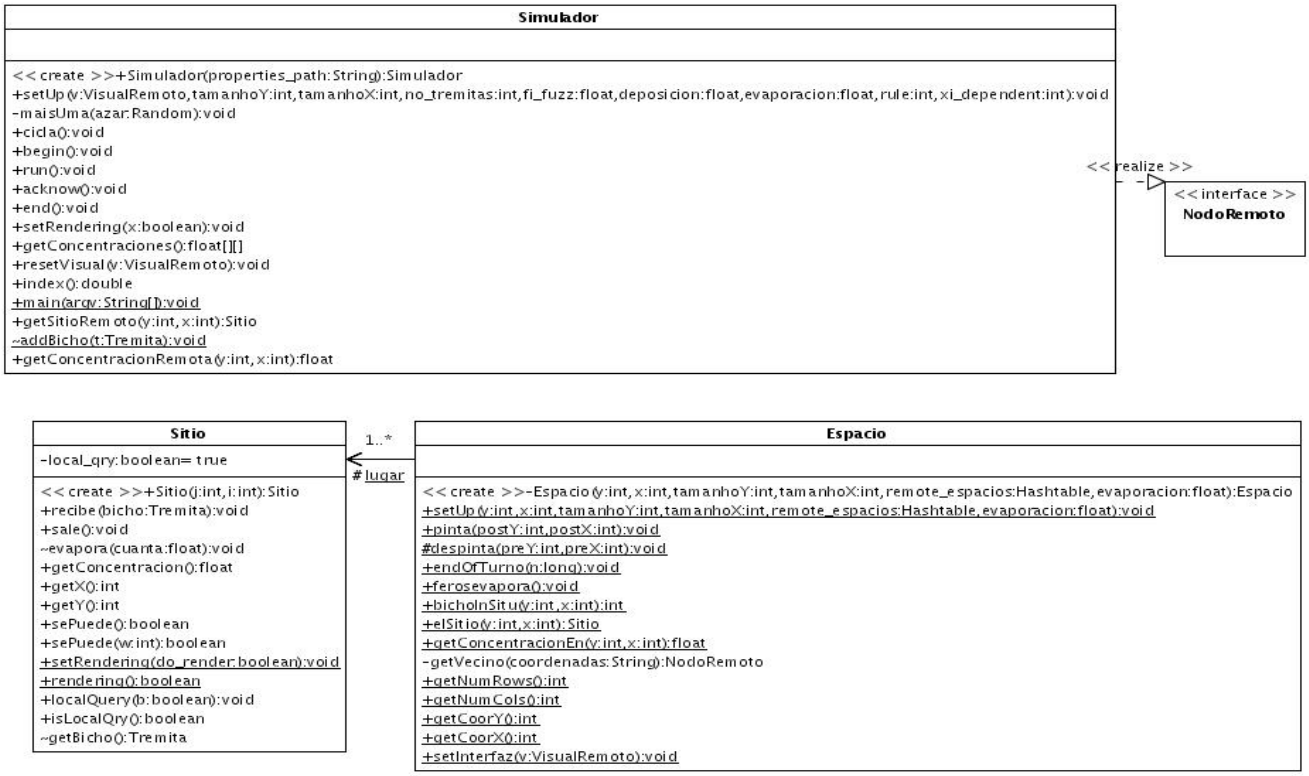


fig 3.7: clases en el módulo de ejecución

Un **simulador** constituye el objeto responsable por los intercambios de información que ocurren entre los procesadores participantes en la simulación; contiene un **espacio** y una serie de **tremitas**, interactúa con estos objetos respondiendo a las peticiones de sondeo feromonal y traslado con otros nodos; cada ciclo activa a cada **tremita** en su turno y al término solicita la evaporación feromonal. Su función principal es ser visible desde los subprocesos en los otros nodos que componen la simulación y responder a sus peticiones y

aquellas del usuario desde **interfaz** y a los mensajes internodales de sincronía. Un **espacio** contiene una matriz bidimensional de **sitios** que reciben y acumulan la deposición feromonal y, en caso de estar habilitado el modo gráfico, generan peticiones de visualización cuando un **tremita** entra. El **espacio** canaliza las peticiones de pintado y despintado provenientes de cada **sitio** a la **interfaz**. Por otra parte se encarga de determinar si la frontera de la subgradilla que representa es la frontera del espacio total de la simulación y, en caso negativo, obtener la referencia a un **sitio** remoto para canalizarlo a la **tremita** solicitante.

Para permitir el uso de varios nodos, así como para poder representar situaciones espaciales diversas, se definió la gradilla espacial en términos de rectángulos homogéneos con vecinos definidos (o no) en sus cuatro bordes, de manera que fuese posible representar cualquier espacio obtenido de pegar entre sí rectángulos de cualquier medida fija; al llegar a la frontera de una subgradilla (representada por el objeto de tipo **Espacio** en ese nodo) en dirección de otra subgradilla pegada sobre ese borde la **tremita** es transportada de un nodo a otro.

3.5.3 **Modelo**

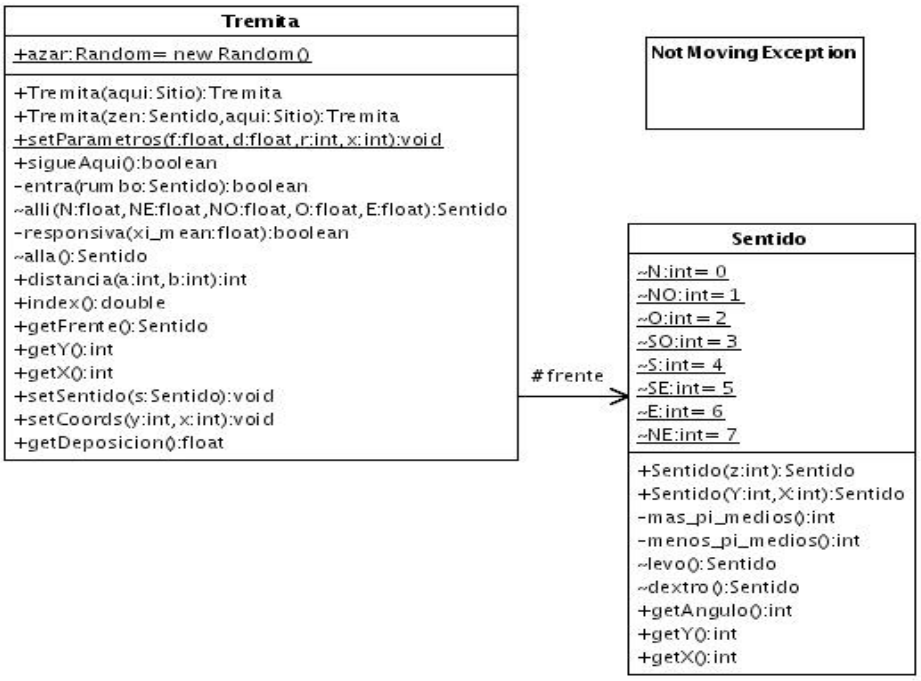


fig. 3.8: clases que implementan la lógica del modelo

Este módulo contiene la clase T**remita** que encapsula los procedimientos relativos a las reglas de movimiento discutidas en el capítulo anterior y **Sentido** que facilita el manejo de las **tremitas** como entidades orientadas. Es este el módulo que contiene las consideraciones teóricas del modelo, siendo que las clases de otros módulos sirven para implementar y visualizar la lógica representada en estas dos clases. Las funciones asociadas a esta clase son bastante auto explicativas y de alguna forma constituyen la esencia del trabajo presentado puesto que son las que definen el comportamiento individual de las partículas que constituyen el autómata distribuido y cuyas reglas de movimiento constituyen el modelo conceptual.

Desde cualquier **sitio** que una **tremita** ocupa, éste primeramente efectúa un sondeo feromonal sobre los **sitios** en la mitad frontal de las casillas circundantes. La concentración promedio y la distribución diferencial de feromona en la vecindad determinan la elección en dirección de movimiento para cada **tremita** en la siguiente unidad de tiempo.

3.6. **Seguimiento feromonal y parámetros del modelo**

El modelo busca retratar dos dinámicas de seguimiento observadas en distintos contextos. La primera corresponde al mecanismo de osmotropotaxis en el cual los insectos buscarían ajustar su posición y desplazamiento para que la concentración feromonal percibida en cada una de sus antenas sean igual. El otro mecanismo tiene como base la elección direccional en el sentido de la mayor concentración. Los dos fragmentos de código que siguen ilustran más específicamente estos dos modos de seguimiento feromonal.

```
if( N >= Math.max( Math.max(NO,O), Math.max(NE,E) )  )
    return frente;

if( NO >= Math.max( E,O ) && NO > NE )
    return frente.levo();

if( NE >= Math.max( E,O ) && NE > NO )
    return frente.dextro();

if( O > Math.max(NE,NO) && O > E )
    return frente.levo().levo();

if( E > Math.max(NO,NE) && E > O )
    return frente.dextro().dextro();
```

fig 3.9 regla clinotáctil

```
if( (NE+E/2)/(NO+O/2) > xi_fuzz )

        return frente.dextro();

if( (NO+O/2)/(NE+E/2) > xi_fuzz )

        return frente.levo();
```

fig 3.10: osmotropotaxis

Aparte de estas dos reglas heurísticas existen otros parámetros que regulan la ejecución del sistema de modelaje. El tamaño y geometría del **espacio**, en este caso un cuadrado de m**x**m unidades, la cantidad de **tremitas** presentes n, el umbral perceptual (la exactitud con la que son detectadas las diferencias concentracionales entre casillas), las tazas de deposición y evaporación por unidad de tiempo y la taza de incremento en la respuesta, la velocidad con que la fidelidad feromonal aumenta con la concentración.

```
double fi= Math.atan( xi_off*xi_mean )/Math.PI + .5f;
if( fi > Math.random() )
    return true;
  else return false;
```

fig 3.11: respuesta feromonal probabilística en función de la concentración local

3.7 **Interacción entre los módulos del sistema**

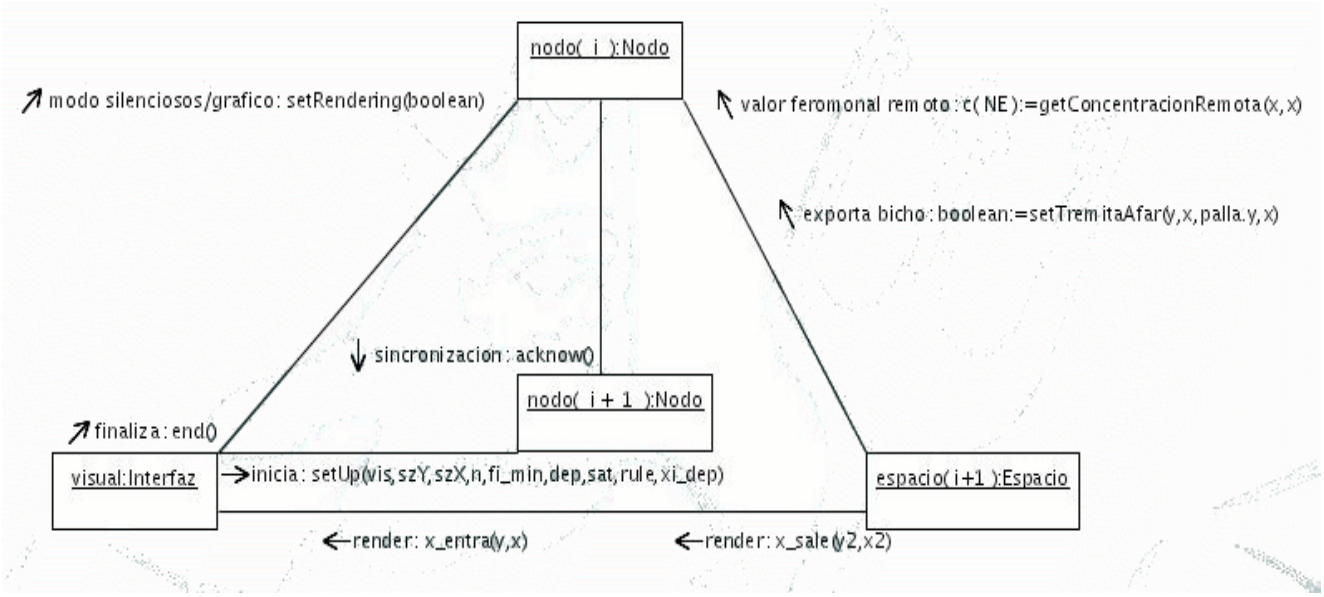


fig 3.12: intercambio de mensajes entre objetos a nivel internodal

En la figura anterior se representa la manera en la que se realiza la comunicación internodal, ya no a nivel de módulos o procesadores sino de las clases específicas que dentro de cada uno se comunican entre sí.

3.7.1 **Sincronía entre nodos**

Esto se logra mediante mensajes intercambiados con cada ciclo y **simulador** transfiriendo **tremitas** de y a otros nodos. En términos de la implementación esta sincronía es resultado del envío y recepción de mensajes entre nodos vecinos.

Cuando un **simulador** termina de procesar un ciclo de ejecución contacta a sus vecinos enviando un mensaje:

```
Enumeration synclist = vecinos.elements();
while( synclist.hasMoreElements() )
       ((NodoRemoto)synclist.nextElement()).acknow();
```

Cada vez que un ***nodo*** es contactado por uno de sus vecinos aumenta uno a un contador que se vuelve nulo al inicio de cada ciclo.

```
public void acknow() throws RemoteException
   //cicla();
   nodos_ack++;
```

Sólo cuando el **nodo** es contactado por todos sus vecinos inicia éste un nuevo ciclo.

```
if( nodos_ack == num_vecinos )
 try{
     cicla();
```

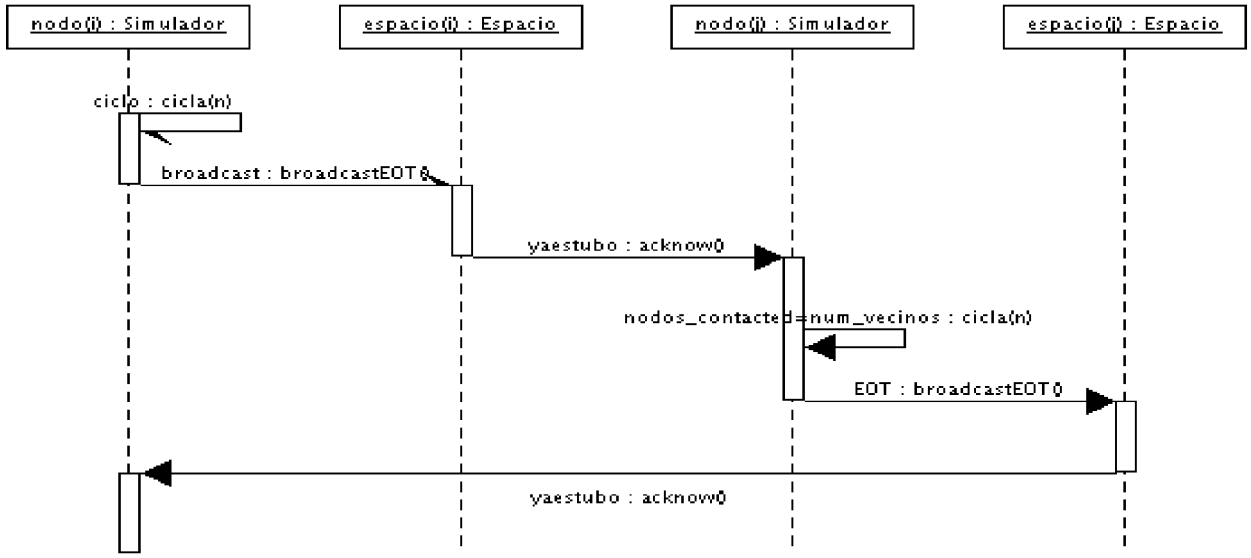


fig 3.13: la sincronización entre nodos como mensajes entre objetos

3.7.2 **Los modos de ejecución gráfico y discreto**

La clase **Interfaz** está activa únicamente cuando existe una interacción con el sistema, ya sea para iniciar, finalizar, visualizar o mandar al **simulador** al modo de ejecución silenciosa. Actúa como cliente ligero, a manera de un "navegador" *renderea* los proceso remoto***s*** y permite interactuar con el sistema.
La diferencia en cuanto a la carga de procesamiento que el funcionamiento de este componente impone al sistema queda determinado por el aumento exponencial del número de mensajes que se intercambian entre procesadores cuando está activo. En la siguiente sección se ilustran detalladamente los dos modos de funcionamiento y la diferencia en cantidad de mensajes requeridos por cada movimiento que efectúa cada **tremita**.

3.7.3 **Funcionamiento a nivel individual**

Los siguientes diagramas ilustran el funcionamiento del sistema a un nivel más detallado, representando no sólo las llamadas internodales sino también las llamadas entre objetos al interior de un **simulador**. El primer diagrama de interacción (figura 3.14) corresponde al caso en que una **tremita** pasa de un **sitio** local a un **sitio** correspondiente en un **simulador** vecino durante el modo silencioso, es decir, sin solicitudes de *visualización*.

El segundo diagrama a continuación (figura 3.15) ilustra el movimiento de una **tremita** de un **sitio** a otro dentro del mismo **simulador**, con la funcionalidad de *visualización* activada. Estos dos diagramas abarcan el funcionamiento básico del sistema. El desacoplamiento de la simulación respecto a la **interfaz** no sólo en el sentido de que ésta reside en una computadora distinta sino por que este módulo posibilita conectarse a una simulación en ejecución desde cualquier computadora con una conexión de red y observar su estado. La flexibilidad derivada de esta arquitectura ha permitido la ejecución de la simulación en diversos ambientes de ejecución, no sólo bajo distintos sistemas operativos sino en las distintas computadoras de una LAN, en una computadora con varios procesadores para cálculos numéricos, desde una estación de trabajo, sobre la propia estación de trabajo y en los distintos nodos de un *cluster*. Ésto gracias a la autonomía del módulo de visualización que se conecta en tiempo real con los nodos de la simulación ya sea localmente o vía red.

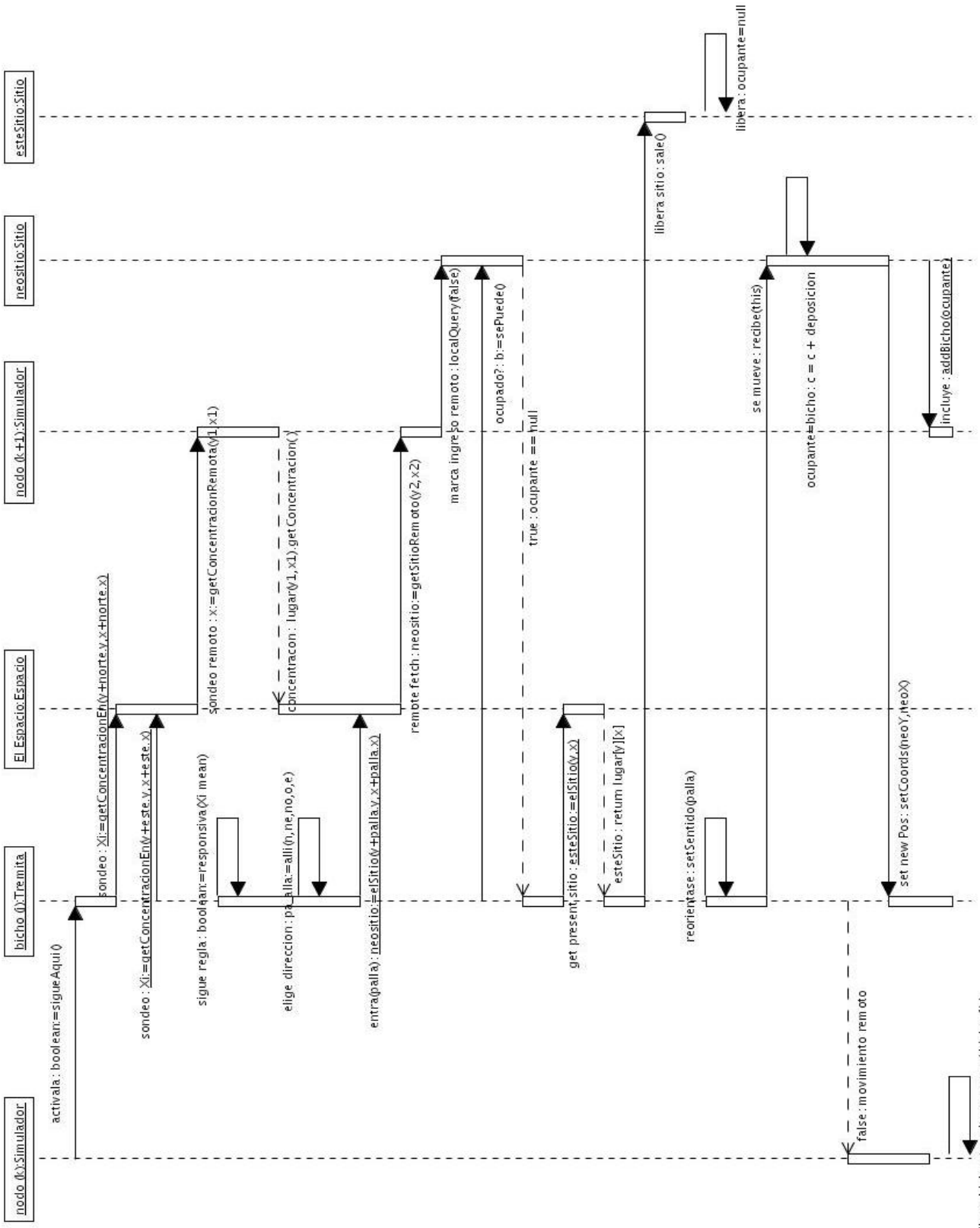


fig 3.14: traslado internodal de un individuo

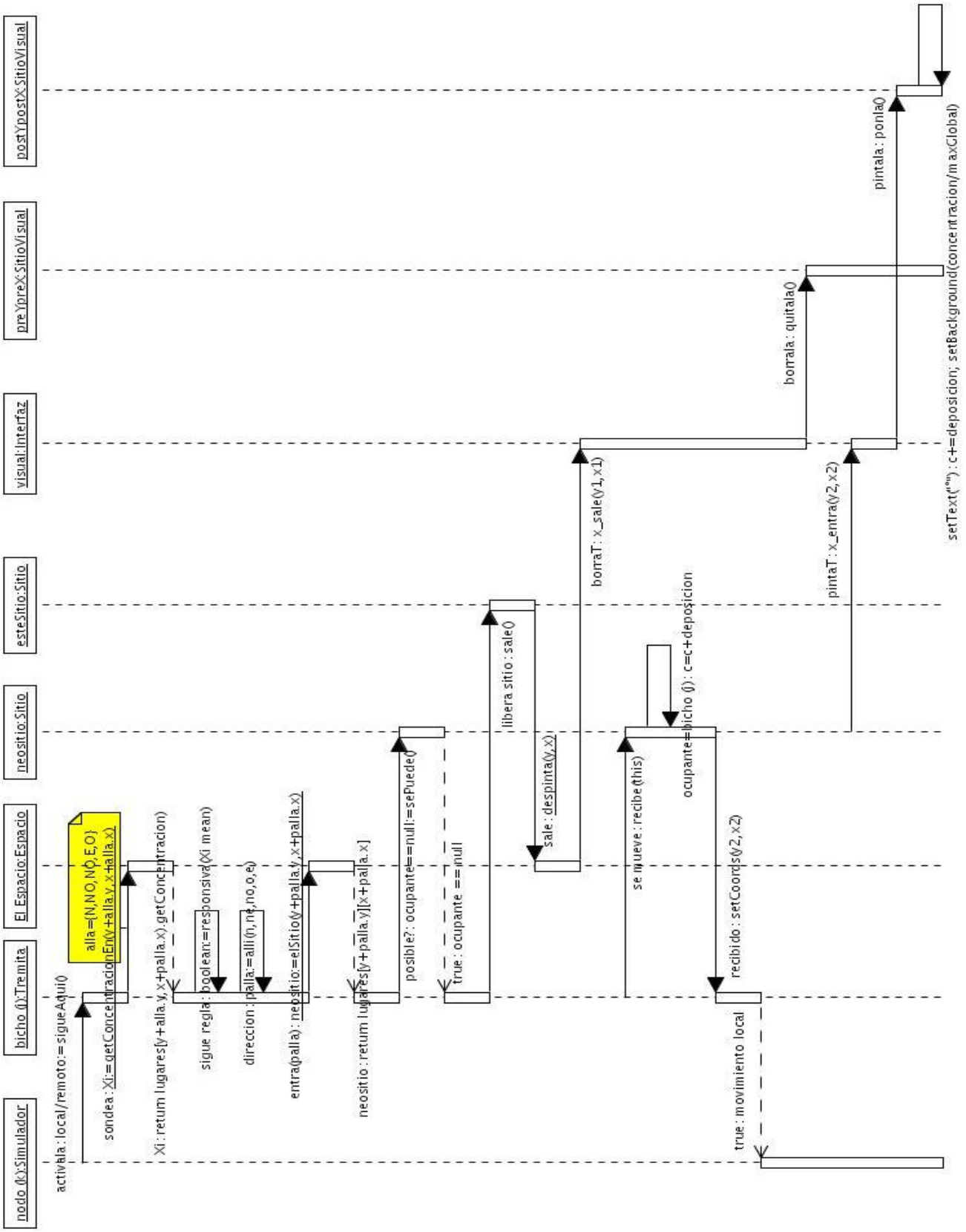
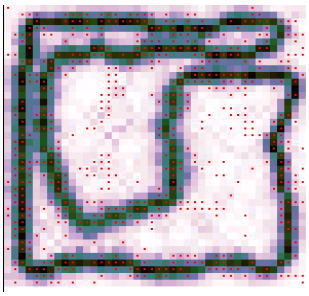


fig 3.15: traslado internodal con visualización activa

El esquema distribuido permite el uso de este sistema para representar distintas situaciones espaciales (aquellas generables pegando rectángulos de tamaño uniforme), con lo que se podrían representar otras situaciones de comportamiento de insectos sociales e incluso posibles interacciones con otro tipo de entidades: comida, individuos de otra especie, etcétera. Es decir, el modelo presentado busca ser un esquema escalable para la simulación de dinámicas socioespaciales de poblaciones y comunidades.

RESULTADOS



La simplicidad y la flexibilidad
triunfarán sobre el poder en un mundo
cuya clave sea la conectividad.

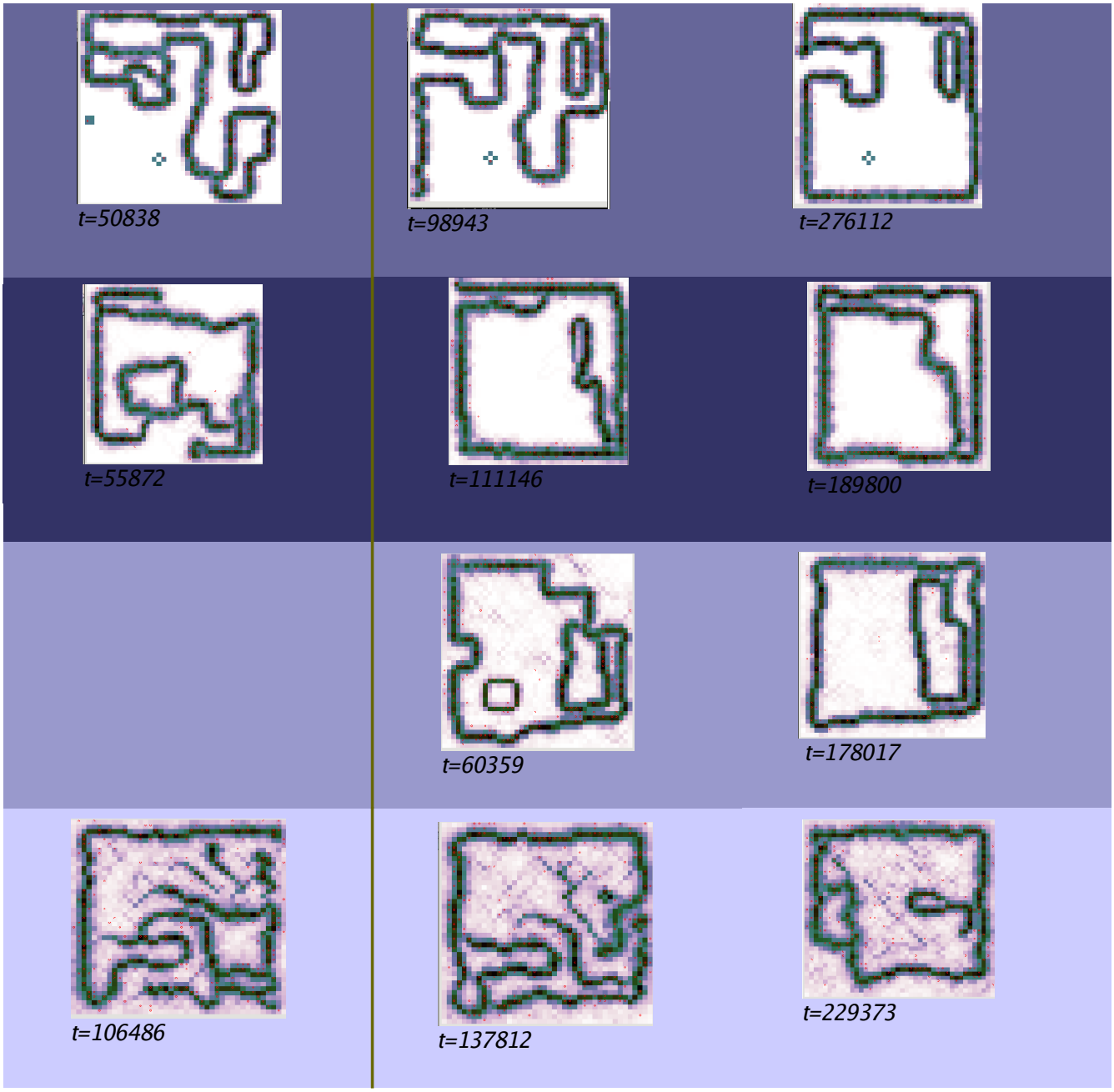
A. Bosworth

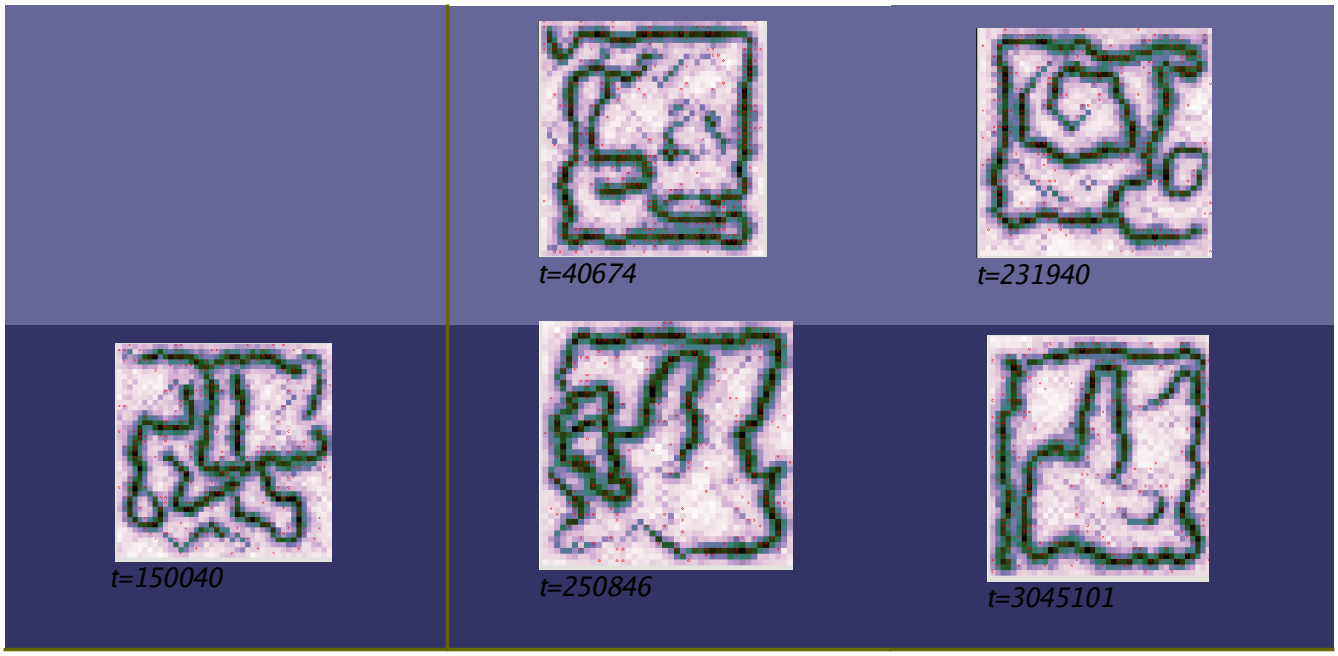
Esta sección presenta los resultados derivados de los experimentos llevados a cabo con el sistema de simulación orientado a objetos. Esta implementación del autómata celular es modelo de los principios fenomenológicos de la organización espacial guiada feromonalmente en insectos sociales en un medio homogéneo. Por no haber ninguna referente direccional, si en la exploración del espacio la trayectoria de deposición feromonal llega a autointersectarse, difícilmente se podrá romper como atractor de trayectorias por la retroalimentación positiva a que da lugar. Esto es altamente probable en una situación espacial limitada a una caja pues la geometría hace altamente probable el evento de una intersección. Por tanto, abundan resultados con ciclos cerrados.

A partir de principios los heurísticos individuales de movimiento, se generan configuraciones dinámicas de concentración feromonal que pueden ser leídas visualmente como movimientos, ya no a nivel individuo sino, a nivel de la colectividad de hormigas como líneas o trazos feromonales. Los trazos se van modificando mediante "cambios topológicos" limitados por unas condiciones de frontera fijas y tienden a distribuirse en un ciclo cerrado a lo largo de dicha frontera espacial. En lo que sigue se muestra el resultado de variar el parámetro *f*, su efecto es el de acotar el rango de la curva la sigmoidal, que determina la probabilidad de seguimiento feromonal. Se presentan los resultados en sus efectos relativos a cada regla de seguimiento. Los demás parámetros se han mantenido fijos, una vez que se encontró una ventana de valores que no derivara en la saturación feromonal que, por su falta de información, es similar a la condición inicial homogénea (posiciones y direcciones aleatorias para las tremitas y ninguna deposición feromonal) ni tampoco en una distribución feromonal precaria en la que no se distingue ningún patrón emergente.

Regla clinotáctil

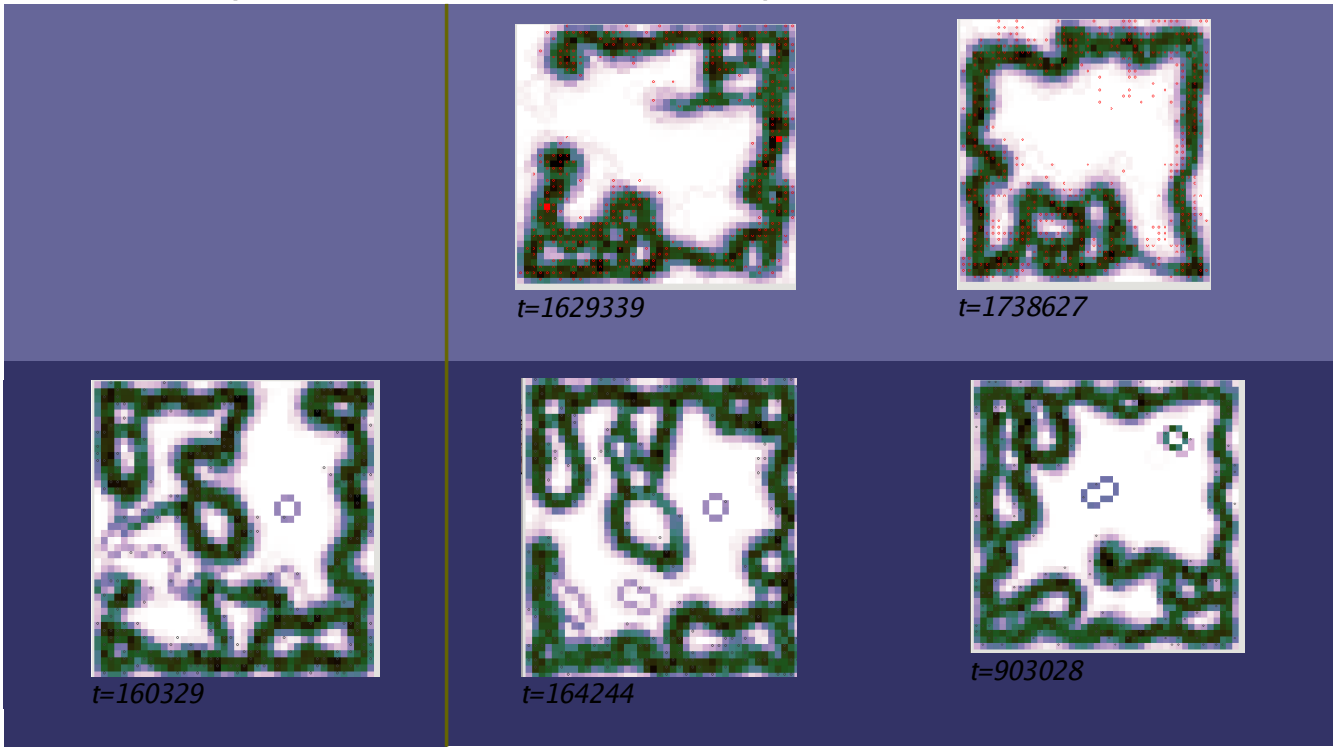
En las siguientes series temporales se ilustra el efecto de incrementar la incertidumbre de seguimiento. Los parámetros utilizados son n=400 **tremitas**, e=0.01 como índice de evaporación, 30x30 en el tamaño de la gradilla y f= 0.01, 0.025, 0.5, 0.1, 0.135, 0.15 respectivamente. Al mostrar distintos momentos de la simulación, se busca también ilustrar que las modificaciones temporales se dan a nivel de trazo feromonal y no a nivel individual.

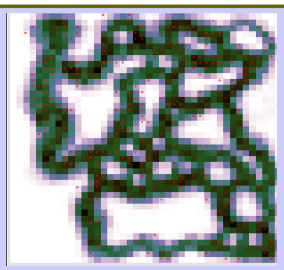




Regla osmotrópica

La inclusión de las siguientes series exhibe la diferencia cualitativa entre el uso de cada una de las reglas de seguimiento, esta segunda regla deviene en trazos más gruesos y sinuosos. Los valores iniciales son los mismos que en el caso anterior y el valor de f= 0.01, 0.1, 0.2 respectivamente, n=450





t=742295

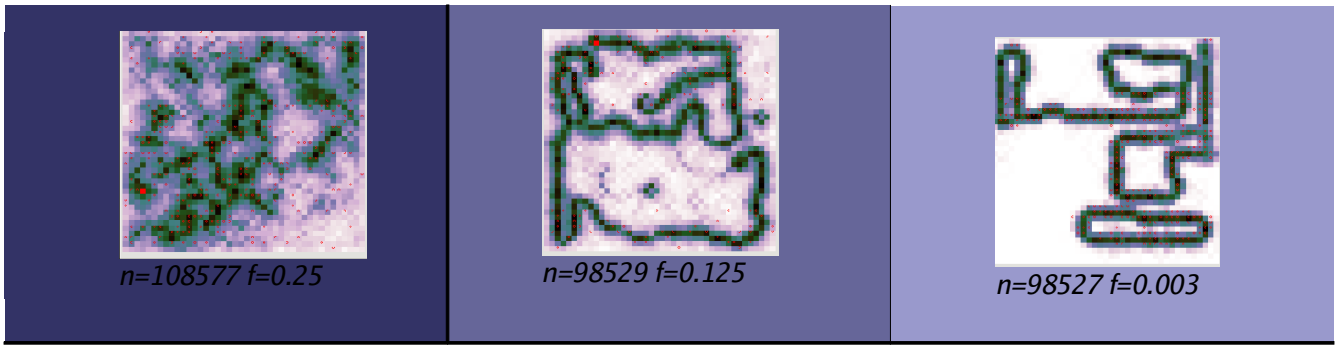
t=860481

Estos resultados ilustran que la configuración cíclica se deriva del acoplamiento colectivo a las condiciones espaciales. Esto sugiere que los mecanismos estigmérgicos son principios intrínsecamente aptos para el reconocimiento espacial. Estas series temporales ilustran que los patrones obtenidos tienen relevancia a una escala espacial y temporal superior a la de los movimientos individuales de las tremitas. En este sentido, los individuos no existen más como tal a pesar de que son sus decisiones individuales las que conforman en patrón global. Se observó que las modificaciones más significativas respecto a los patrones obtenidos están relacionadas con modificaciones a la respuesta feromonal y la regla se seguimiento.

Estas anotaciones son apoyadas por un trabajo (Anderson *et.al.*, 2002) que describe y analiza el papel biológico de los al menos 18 tipos distintos de autoensambles de individuos. Estos son estructuras funcionales como puentes, techos, balsas que son formados por el acoplamiento de individuos ante situaciones que requieren defensa, termoregulación, transporte, desplazamientos. De acuerdo a estos autores las estructuras mencionadas conforman un nuevo nivel de complejidad funcional más allá del individuo que correspondería a la organización en tejidos y órganos en los organismos pluricelulares. Asimismo, señalan similitudes entre la formación de este tipo de estructuras intermedias y procesos de morfogénesis embrionaria; lo cual hace del autoensamblaje un mecanismo adaptativo operando a varios niveles de escala biológica.

Independientemente de la regla se seguimiento feromonal utilizada, el proceso similar de ordenamiento varía de acuerdo al parámetro de incertidumbre en la respuesta individual a la distribución feromonal. Para valores bajos de este

parámetro la configuración espacial del sistema tiende a mantenerse contenido en los trazos formados inicialmente, para valores medios se encuentra la mayor correlación entre el patrón feromonal resultante y el dominio espacial asignado siendo que los trazos se deforman topologicamente, para valores altos de incertidumbre los propios trazos se vuelven inestables. A continuación se ilustran estas observaciones.



Las siguientes gráficas muestran la distinta velocidad de convergencia de los individuos hacia estructuras con menos grados de libertad. Estos resultados fueron obtenidos a partir de la relación de un índice de orden con la evolución de la simulación en el tiempo.

$$\omega = \sum \sigma(\Upsilon_\iota)/n$$

donde $\sigma(\Upsilon_\iota)$ representa el índice de cada individuo Υ_ι en la simulación y **n** el número total de individuos. El índice para cada individuo está dado por:

$$\sigma(\Upsilon_{(y,x)}) = \sum(\varrho(\Upsilon_{(y,x)}) - \varrho(\Upsilon_{\nu(y,x)})) / \| \Im(\Upsilon_{(y,x)}) \|$$

donde: $\Im(\Upsilon_{(y,x)}) = \{ \Upsilon_{\nu(y,x)} \mid \nu(y,x) = (y,x)+(v,w)\ v=-1,0,1;\ w=-1,0,1 \}$
es decir, la cantidad de **tremitas** en las casillas adyacentes y

$$\varrho(\Upsilon_{(y,x)}) - \varrho(\Upsilon_{\nu(y,x)})$$

la diferencia (módulo 8) en la orientación del individuo en cuestión con la de cada una de sus vecinas. Este índice es entonces el promedio sobre los individuos de las diferencias direccionales entre cada uno de éstos y sus vecinos.

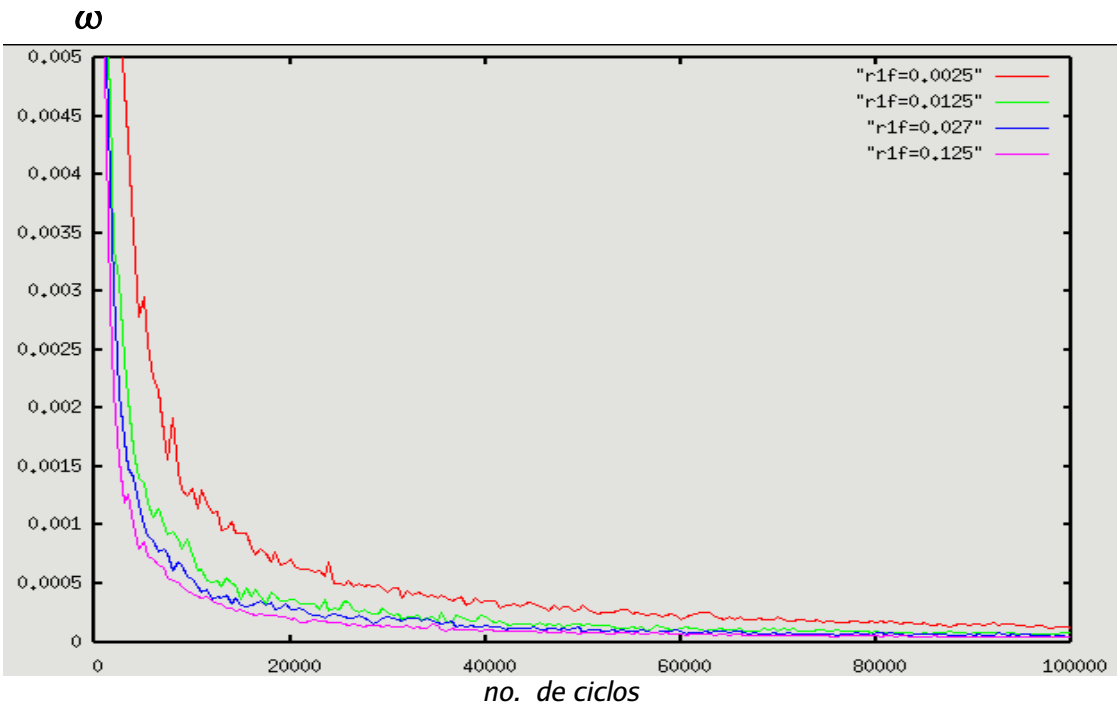


fig 4.1: velocidad de convergencia al ordenamiento con distintos valores de f

La siguiente gráfica muestra los datos anteriores en escala logarítmica. La distribución en línea recta denota una relación basada en una ley de potencias donde la pendiente de la recta es el índice de esta ley de potencias. El valor del exponente de la regla de potencias es de -1, obtenido mediante un análisis de regresión lineal.

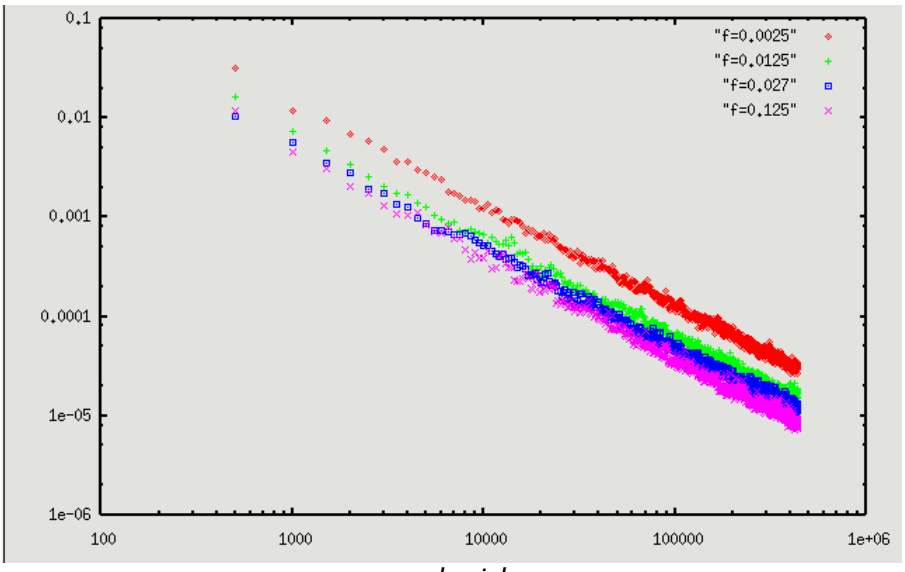
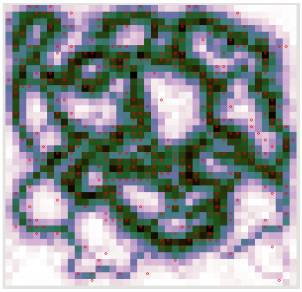


fig 4.2: ploteo a escala logaritmica de los valores de convergencia

DISCUSIÓN.

Los genes ejercen el papel de restringir el proceso de autorganización a opciones con alta probabilidad de éxito ... no son el mecanismo de evolución, la autorganización constituye este mecanismo. (Schneide & Kay en Murphy & O'Neill, eds.)



Los patrones obtenidos muestran el surgimiento de una colectividad organizada que reconoce y actúa en estrecha relación con las condiciones espaciales del entorno distinguiendo el centro y bordes, organizándose colectivamente en términos de las fronteras del dominio espacial. Para el proceso de transformación de los patrones espaciales cabe una lectura, no en términos de deposición puntual y movimiento de individuos sino, a través de transformaciones topológicas del trazo feromonal. En este sentido es que las reglas individuales dan lugar a una inteligencia espacial de orden superior que, en particular, representa información geométrica global; lo cual no es el caso de cada individuo. Lo anterior queda ilustrado con tres ejemplos adicionales:

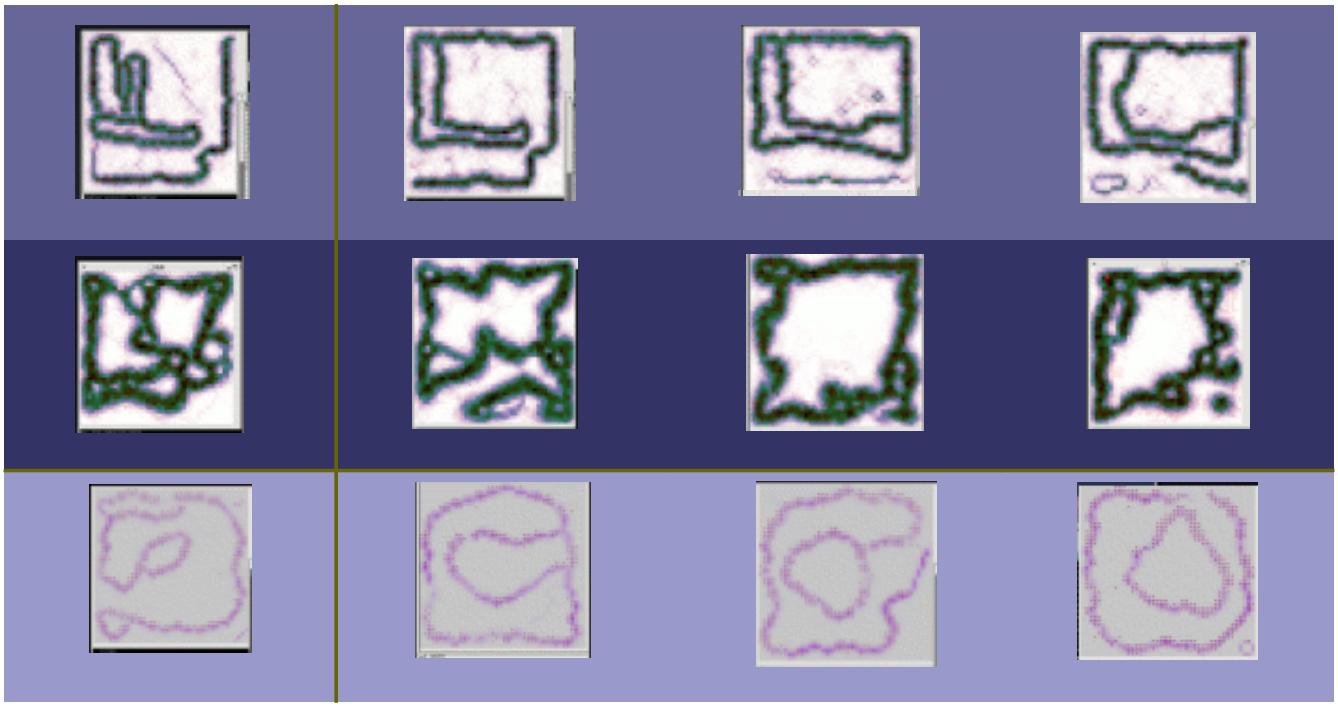
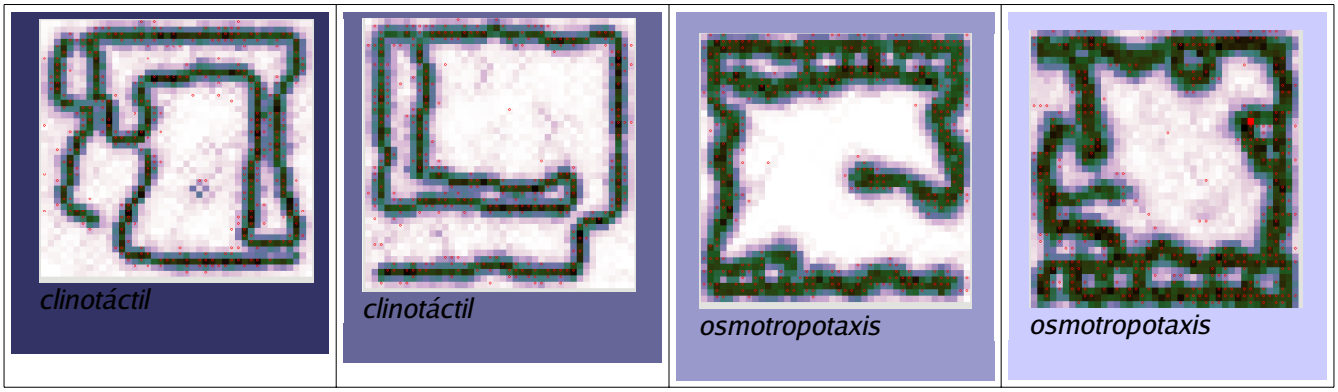


fig 5.1: transformaciones topológicas a nivel de trazo colectivo

Existe una emergencia autorganizada de estructuras globales ordenadas a partir de reglas locales puntuales. Un hallazgo interesante derivado de las simulaciones son las diferencias cualitativas entre los patrones obtenidos a partir de una y otra regla de seguimiento feromonal. La regla osmotrópica está conformada localmente por trazos más gruesos y sinuosos, además, el patrón global presenta autointersecciones que faltan en los patrones más "lisos" derivados de la regla clinotáctil; estas dos características hacen que el trazo feromonal en el caso de osmotropotaxis cubra una superficie significativamente mayor. Las tendencias anteriores corresponderían, intuitivamente, a un contexto motivacional de exploración y de seguimiento, respectivamente. Las concentraciones mayores tienden a atraer más individuos y los trazos delgados cercanos se extienden hasta romper; se "satura" el área concurrida y se extiende un brazo bajo la "presión" en alguna dirección libre disponible. Una vez que el patrón es más homogéneo, los patrones osmotrópicos se modifican poco. Lo anterior apoya las observaciones de campo de que cada mecanismo debe ser utilizado en distintos contextos sociomotivacionales. La siguiente figura ilustra lo anterior.



La mayor cobertura de superficie del trazo derivado de esta regla y las autointersecciones locales podrían asociarla a procesos de búsqueda aleatoria. Por otra parte, la regla clinotáctil, por ocupar una superficie menor y dar lugar a rutas más directas, podría corresponder al seguimiento de rutas establecidas por parte del los individuos más experimentados, las exploradoras. Una regla favorecería a la cohesión, una mayor cobertura de la superficie y una elevación en la tasa de encuentros mientras la otra la optimización de rutas.

El nivel de incertidumbre probabilística juega un papel definitivo en la regularidad o coherencia de las rutas obtenidas con cualquier regla. Una vez que se encontraron valores balanceados para la densidad espacial y las tasas de deposición y evaporación fue este factor que se encontró como el parámetro con mayor influencia en la configuración global, siendo que la regla de seguimiento es lo más determinante en la constitución local del trazo. Esto sugiere que el llamado estado motivacional de los insectos (representado justamente por la regla de seguimiento aplicada y la respuesta a la distribución feromonal) es el que factor más determinante en tanto al tipo de dinámica espacial obtenida; con un rango de resultados ubicados entre el orden y el desorden absoluto. A continuación se ilustra esta afirmación para los dos tipos de movimiento analizados.

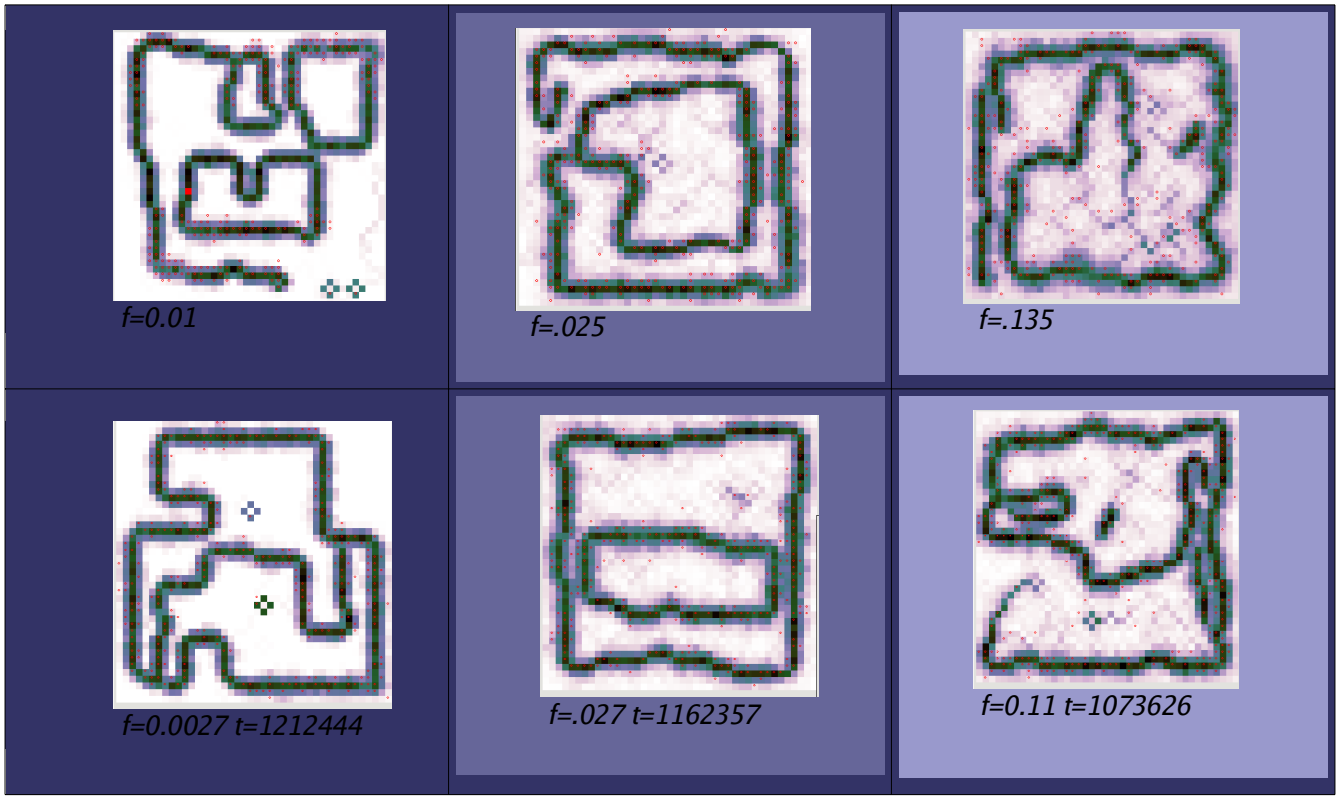


fig 5.3: diferencias topológicas con el aumento de incertidumbre probabilística modo clinotáctil de seguimiento

Sigue la ilustración de este fenómeno con la otra regla de seguimiento:

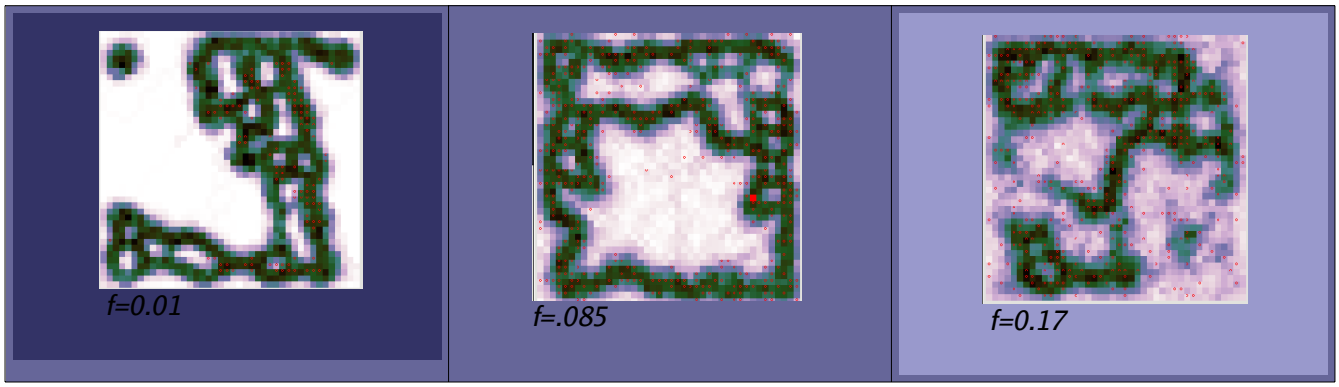
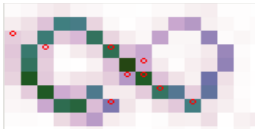


fig 5.4: diferencias topológicas con aumento en la incertidumbre probabilística modo osmotrópico de seguimiento

Debido a la escala espacial del experimento efectuado en laboratorio no se explotaron de lleno las posibilidades de paralelismo del sistema desarrollado, lo cual sería posible (y quizás necesario) en un subsecuente estudio de situaciones sociales más complejas que involucren un mayor número de individuos y configuraciones espaciales no homogéneas.

CONCLUSIONES.



Evolution is the increasing exhibition of consciousness from apparently, not truly, unconscious matter. In existence there are no rigid partitions. 'Life', appears with organized matter but is not something entirely new which had no sort of being before. The difference between stone, plant, animal and man, has always been one rather of degree than of kind.

Woodroffe, J. 1918

Una lección básica proveniente del estudio de los insectos sociales es que, para representar una dinámica, es preciso representarla como un sistema por derecho propio en términos de reglas relevantes a ese nivel fenomenológico. Lo anterior es válido para dinámicas de patrones neuronales, redes inmunológicas o ecosistemas. A pesar de las características comunes a todos éstos, es importante enfatizar que conforme avanzamos en su comprensión comenzamos a encontrar sus diferencias características (Gordon, 1999), lo cual dificulta y pone en tela de juicio la plausibilidad del desarrollo de una teoría general de sistemas.

Este modelo no retoma ningún mecanismo de interacción entre individuos. Gordon analiza un principio de relación entre individuos como determinante en muchos contextos; se trata de la tasa de encuentros que percibe un insecto y sugiere también la posibilidad de que en estos encuentros se intercambia información que afecta el estado motivacional de los individuos; distintos estados con base en los cuales se dan distintas respuestas a un solo estímulo. El propio estado motivacional en relación con el estado de los otros individuos encontrados afectaría el estado del individuo en cuestión.

Las observaciones que hace del comportamiento de hormigas en el desierto de Arizona apoyan lo encontrado en cuanto a que asigna un papel de importancia al nivel de sinuosidad de las rutas. Gordon asocia esta propiedad local a la asignación de tareas distintas en el contexto de la exploración y recolección de alimento. Las hormigas exploradoras que habitan la periferia del hormiguero son las que inician las exploraciones siguiendo cursos más directos e independientemente del grueso de la colonia. Las hormigas más jóvenes, más

cercanas a la entrada al hormiguero, son activadas por las primeras y casi no realizan tareas fuera del nido; siguen rutas más convolucionadas, posiblemente asociadas al mecanismo de seguimiento osmotrópico de partículas feromonales suspendidas, que tienden a formar una masa compacta en torno a la entrada del nido y sólo participar en la colecta de alimentos con rutas recientes y bien marcadas (Gotwald, 1995). De esta manera se asignan diferencialmente las tareas de exploración y recolección de acuerdo a la edad de los individuos. El mecanismo que controlaría el paso de un régimen de movimiento convoluido a aquel más directo puede ser la tasa de encuentros que experimenta cada hormiga puesto que en densidades bajas tenderían a agregarse mientras que en densidades altas se evitarían (Gordon 1999). Los resultados de las simulaciones concuerdan con estas observaciones experimentales respecto las características de las trazas feromonales obtenidas con las dos reglas de seguimiento.

La formación de patrones espaciales es el resultado del efecto combinado de reglas sencillas a nivel individual respecto a un entorno y un estado interno cambiante. Las estructuras emergentes no sólo se forman a pesar de las fluctuaciones azarosas resultantes sino que son justamente estas fluctuaciones las que permiten la desestabilización del estado homogéneo. El proceso morfogénico se refiere a la formación de estructuras donde cada modificación hecha se retroalimenta sobre los subsecuentes cambios puntuales, de manera que el propio sistema en formación constituye para sí la información sobre sus rutas de desarrollo que se lee como *estímulo* motivacional. En el caso de los insectos sociales la codificación conductual no está dada a nivel individual mediante signos sino a través de una miríada de interacciones individuo-individuo e individuo-entorno: gradientes feromonales, modificaciones materiales, estructuras funcionales de agregación de insectos, así como la propia distribución espacial de elementos en la colonia. Todas estas señales asociativas de retroalimentación negativa y positiva dependen para su efecto de la intensidad y el contexto, totalmente asociados a los mecanismos de competencia entre amplificaciones locales e inhibición a largo plazo propuestos por Alan Turing en los cincuentas (Theraulaz *et.al.* 2003). En este sentido cabe notar que, aún en el caso de las galerías, columnas, paredes y arcos en los termiteros, los patrones no son estructuras fijas sino que son el reflejo concreto

de un proceso continuo de construcción y deconstrucción. Observado a partir de la escala individual este proceso puede parecer torpe o absurdo, como cuando un insecto quita lo que otro acaba de colocar; no obstante desde un punto de vista global los insectos sociales son capaces de encontrar soluciones ergonómicamente óptimas para problemas complejos que tienen su contraparte humana en áreas avanzadas de la ingeniería de operaciones (Theraulaz G. *et.al.*, 2003).

Una posible estrategia futura sería la evaluación de las reglas y conjuntos paramétricos del modelo ante otras situaciones socioespaciales documentadas en el medio natural. De esta manera, se contaría con patrones espaciales típicos a situaciones específicas contra los cuales comparar nuevos resultados generados y determinar si cada una de las reglas postuladas describe mejor ciertas actividades de estos insectos. Sería importante también incluir las reglas explícitas de interacción entre individuos descritas, pues parece que el contacto fisicoquímico de un individuo con otro es un principio de organización de suma importancia entre los insectos sociales (Gordon, 1999:168). Estas reglas deben provenir de observaciones directas hechas en el medio natural para que el modelo tenga mayor relevancia biológica puesto que las reacciones individuales varían con los estados motivacionales, indisolubles al entorno físico. Estas modificaciones fundamentarían la investigación, ya no como un ejercicio sino como modelo de organización espacial relevante a nivel biológico. Es posible que a raíz de esto el modelo perdiese generalidad, puesto que sería difícil extrapolar los resultados a otros ámbitos naturales, pero tal sería el precio por la comprensión cabal del fenómeno específico. El sistema desarrollado tiene la capacidad de representar diversas situaciones espaciales y con pocas modificaciones pudiera ser extendido para la representación de distintos contextos sociales.

Este tipo de modelo (que reproduce situaciones concretas, a partir de principios de dinámica individual, relativas a la generación, manutención, transformación y disipación de estructuras ordenadas) encuentra su lugar no tanto en función de la construcción de un cuerpo teórico abarcante a partir del cual se puedan describir una amplia gama de fenómenos basados en dinámicas cualitativamente similares. Por el contrario, me parece que su principal virtud

está en la posibilidad de describir fenómenos particulares evaluando a profundidad la plausibilidad de los principios postulados en función de un interés, también específico, por el fenómeno. En otras palabras, el sentido de la aplicación de los modelos de la matemática discreta (autómatas celulares, redes neurales, algoritmos genéticos, redes booleanas) estaría en explorar los principios cualitativos causales detrás de fenómenos específicos en caso de no contarse con una teoría general. A partir de este esquema de trabajo se puede visualizar una imagen alterna a la visión piramidal del edificio científico del conocimiento; la superposición de franjas relacionales de conocimiento que emparenten niveles de organización colindantes sin una relación semántica explícita entre los distintos estratos. Este sistema de "parches" cubriría el "área total" que a principios del siglo XX parecía representable a partir de un solo sistema de leyes y principios mecanicistas.

Las líneas de pensamiento sobre las que descansa este trabajo cuestionan no sólo la estructura reduccionista del pensamiento científico tradicional sino también la viabilidad, conveniencia y eficacia de una estructura social de poder centralizado que permea toda nuestra cultura, desde la familia, escuelas, empresas e instituciones, hasta la noción misma del estado nación; están todas orientadas a una organización jerárquica que queda plasmada en el paradigma de cómputo digital basado en la Unidad Central de Procesamiento. Esto proviene de una de las grandes diferencias, no sólo entre los insectos sociales y los humanos sino entre los humanos del presente y los antiguos. El control central deviene necesario y posible en la misma medida que una marcada tendencia hacia la individualización de intereses en detrimento del llamado interés colectivo que, teóricamente regulado por instituciones sociales, deja de ser una responsabilidad. Esto también permite que dichas instituciones sean controladas por pequeños grupos de gran poder económico, antes locales, ahora trasnacionales.

Es argumentable que la autogestión sería el más eficaz sistema organizativo humano (Moura, 2002). A diferencia de los insectos sociales que actúan desinteresadamente, sin opción propia, en el caso humano el desinterés puede surgir sólo de una firme voluntad consciente que mediante la atención minuciosa desarticule el egocentrismo como motor de accción. A pesar de que la

condición humana parece descansar sobre el principio de elección personal, podemos observar mecanismos fuera del nivel consciente que dan lugar a configuraciones globales coherentes. Éstas son producto de acciones puntuales que normalmente no serían consideradas como trascendentes, tal es el caso, por ejemplo, del surgimiento de veredas en el campo que surgen a partir de las modificaciones hechas por cada individuo al pisar sobre el terreno y desaparecen por desuso en favor de alternativas más interesantes; las más de las veces sin que los actores se comuniquen directamente y elaboren un plan consciente. A este tipo de operación podríamos denominarlo "cómputo ambiental" y tiene la característica de basarse en la memoria espacial colectiva, con base en marcas espaciales reconocibles por el grupo. Al margen de las políticas urbanas gubernamentales es argumentable que este mecanismo está atrás de muchas de las decisiones de ordenamiento en el caso del levantamiento de ciudades y pueblos. "Calles, casas y plazas aparecen como resultado de un proceso constante de construcción y deconstrucción de expectativas e intereses, producto de acciones individuales y consensos colectivos puntuales" ... "son el resultado global de la acción de múltiples agentes actuando cada uno a nivel local" (Moura, 2002). La producción cultural está también determinada, en parte, por este tipo de proceso. El estímulo para producir está dado tanto en términos cuantitativos como cualitativos por las características de la producción local; culturalmente nunca se parte de un "ambiente monocromático" como en las simulaciones efectuadas. "La producción cultural está caracterizada por el refuerzo de ciertas rutas y el abandono de otras en un permanente reconstrucción del mapa cognocolectivo" (Moura, 2002). Es importante reconocer el importante papel de este tipo de procesos en la dinámica social humana pues sólo así existe la posibilidad de que tengan como trasfondo un estado motivacional adecuado, producto de la responsabilidad individual.

En una sociedad como la contemporánea, basada en directivas superiores y líderes, los grupos tienden a ser incapaces de generar alternativas más allá de las decisiones individuales de estas figuras de poder (Moura, 2002). Esto sólo puede cambiar a partir de una cabal responsabilidad sobre los propios actos donde la interacción entre individuos estuviera conscientemente basada en sugerencias o estímulos a partir de actos concretos, en vez de basarse en

órdenes y directivas. Theraulaz caracteriza estos sistemas como descentralizados y cooperativos mediante unidades autónomas y distribuidas, basados en comportamientos simples de estímulo-respuesta con acceso restringido a información local. Las características de autorganización y el uso de templetes (el uso de estructuras previas como base para elaboraciones posteriores), tan exitosamente explotados por los insectos sociales, han sido reconocidos como mecanismos potencialmente útiles para la solución de problemas de optimización y en el diseño de dispositivos artificiales distribuidos.

De acuerdo a Wheatley (1996), los principales fallos en la estructura organizativa de nuestra sociedad provienen del supuesto fundamental y erróneo de que las organizaciones deben de funcionar, como las máquinas, bajo los principios de la predicción y el control. El cambio y el "ruido" pueden ser vistos como la fuente de orden, no una intrusión problemática. Incluso hoy, las soluciones locales predominan más no son reconocidas como modelos organizativos viables. Unas pocas reglas de interacción local llevan inexorablemente a respuestas colectivas coordinadas.

La crisis como sistema social tiene origen en el desfase semántico de la realidad global que habitamos con el paradigma científico oficialmente vigente. La tensión se da principalmente entre las partes y el todo en un movimiento pendular de golpes y contragolpes hacia uno y otro lado. Veinticinco años después del análisis de Khun, reconocemos el cambio de paradigma implicado por los descubrimientos en la física subatómica como parte de una transformación cultural más amplia que constituye la crisis finimilenarista. El "orden natural", de ser percibido como piramidal (con el hombre en la cima), ha pasado a ser inteligido como una red, carente de puntos privilegiados. A nivel psicológico esto implica a la expansión de la identificación de sí con la naturaleza en su totalidad.

A pesar de las implicaciones sociopolíticas implícitas en estos desarrollos teóricos, hay abundantes ejemplos pasados donde discursos han sido absorbidos y reinterpretados por el aparato dominante de manera que se convierten en un mecanismo más de su reproducción (Focault , 1980). Como ejemplos de este proceso podemos citar el movimiento global juvenil del 68 que

de un replanteamiento en cuanto a la producción como fin social devino en la integración a la economía de mercado de actores inusitados y productos insólitos (aún cuando se cataloguen bajo el rubro de tráfico ilegal) haciendo de la otrora disidencia un artículo de consumo suntuario. La esencia de estos movimientos ha sido diluida hasta que la mayoría de la sociedad (incluso al interior de éstos) no puede leer en ellos sino opciones profesionales y de consumo con que llenar el vacío del maquinismo contemporáneo.

En el caso de los nuevos avances en la teoría de sistemas, los mismos modelos que evidencían la insuficiencia explicativa del discurso científico reduccionista están siendo empleados en la detección de movimiento para proyectos militares de armas robotizadas (redes neuronales), mayor eficiencia en la línea de producción (lógica difusa), altas finanzas y desarrollo de estrategias de mercado a partir de la información desestructurada almacenada e interpretada en servidores *proxy.* La discusión anterior no tiene un afán derrotista sino el de plantear que en la búsqueda de la verdad no bastan la teoría y la forma. Es indispensable una intención subjetiva con la claridad suficiente para que los ideales que procura y contiene puedan ser destilados hasta convertirse en una realidad material reflejada en un estilo de vida, actitudes y actividades cotidianas que mediante mecanismos estigmérgicos moldeen un futuro menos jerarquizado. Este tipo de resultados sugiere que la necesidad de organismos y legislaciones normativas pudiera ser más bien creada y tendientes a justificar una estratificación social que no sólo no es necesaria sino perjudicial, tanto a nivel individual (indistintamente de la ubicación en la jerarquía) como social y ecológico.

Contamos con evidencia que apunta a que las dinámicas colectivas más eficientes no necesariamente son aquéllas dependientes de una concienzuda planificación central; por lo contrario, la impredicibilidad asociada a algoritmos basados en información y efectos locales parece ser un punto a favor de las dinámicas estigmérgicas que permiten responder de manera flexible a escenarios cambiantes e impredecibles. Las redes binarias complejas (que han sido usadas para la representación de redes genéticas, inmunológicas, redes neurales, sistemas de órganos y ecosistemas) exhiben tres regímenes de comportamiento: un régimen ordenado, con componentes congelados, un

régimen caótico sin componentes fijos y una región limítrofe donde los componentes congelados comienzan a fundirse (Capra, 1996:215).

Ya existen hoy aplicaciones basadas en el comportamiento de agentes múltiples que, mediante interacciones locales, resuelven el ruteamiento de tráfico en redes, el análisis y agrupamiento de datos de clientes bancarios (mediante algoritmos provenientes de la clasificación espacial de huevos, larvas, pupas y ninfas en el hormiguero), el ensamblaje de líneas de producción en fábricas o el problema de la ruta más corta visitando n lugares. *Ruteo* de enlaces telefónicos a través de estaciones de direccionamiento, planeación de itinerarios y tiempos de vuelo con correcciones dinámicas ante mal tiempo o congestión de rutas (obtenido de una tasa de deposición asociada al nivel de tráfico en una ruta). Estos principios algorítmicos generan desempeños computacionales superiores al del protocolo existente en el ruteo de paquetes sobre internet que se vuelve muy demandante, pues exige una comunicación continua entre nodos de ruteo. Otro campo de aplicación ha sido la robótica basada en la cooperación de robots simples. La comunicación masiva entre insectos sociales a partir de encuentros puntuales locales ha sido utilizada para un modelo de intercambio entre usuarios de internet asociados a distintos grupos y la calificación diferencial de éstos a los contenidos electrónicos de la Red (Bonaneau & Théraulaz, 2000).

En la medida que cada vez más artículos tienen un chip implantado, es posible plantearse una comunicación local en términos "útiles" entre distintos aparatos y objetos urbanos cotidianos que facilitarían la operación global de éstos de forma operativa. Un ejemplo inmediato es el tráfico de automóviles, si éstos contaran con sensores de congestionamiento se podrían comunicar para dar con distribuciones óptimas sobre la red vial.

Las soluciones difusas en masa (*fuzzy swarming*) devienen en soluciones específicas con un alto grado de optimización, incluso para problemas cuya solución analítica crece exponencialmente con el número de elementos. El hecho de que con estos modelos de cómputo el hallazgo de las mejores rutas no elimine por completo rutas menos eficientes y solamente se aumente la probabilidad de transito por éstas, dota a estos sistemas con una forma de

memoria que resulta útil en el caso de que una de estas rutas se vea bloqueada o se congestione. Esta robustez ante cambios ambientales distingue esta forma de cómputo difuso masivo del tradicional, en el que el formato de los datos de entrada tiene que ser previsible, lo cual imposibilita que estos sistemas respondan a situaciones nuevas (Bonaneau & Théraulaz, 2000). A partir de las aplicaciones encontradas para el manejo de bases de datos, trazo de itinerarios óptimos de transporte; es posible especular que esta forma de neoanarquismo algorítmico sea la solución más eficaz para un gran número de problemas prácticos relacionados con la interacción masiva de individuos. Tal vez el tradicional enfoque centralista en cuanto a gestión social pudiera ser sustituido por uno en el que la solución local fuera favorecida.

BIBLIOGRAFÍA

- Acosta F.J., López F., Serrano J. M., *Branching angles of ant trunk trails as an optimization cue*. J.Theor.Biol. 160:297-310, 1993

- Anderson C., Theraulaz G., Deneubourg J.L., *Self Assamblages in Insect Societies.*
  Insect Sociaux Vol 49 2002:99-111

- Bonaneau E., Théraulaz G., *Swarm Smarts.*
  Scientific American, marzo 2000

- Bert H., Wilson E.O., *The Ants*, Cambridge, Massachusetts : Belknap, 1990

- Capra F., La Trama de la Vida, Anagrama: Barcelona, 1996

- Deneubourg J. L., Goss S., Franks N., Pasteels J. M., *The Blind Leading the Blind: Modelling chemically mediated army ant raid patterns*. J. Insect Behavior 2:719-725, 1989

- Dumpert K., *The Social Biology of Ants*, London: Pitman Publishing Ltd, 1981

- Edelstein Keshet L., *Simple models for trail-following behaviour; Trunk trails versus individual foragers*. J.Math.Biol. 32:303-328, 1994

- Edelstein-Keshet L., Watmough J., Ermoentrout G.B., *Trail following in ants: individual properties determine population behavior*. Behav.Ecol.Sociobiol 36:119-133, 1995
- Eigen M., *Steps Towards Life A Perspective on Evolution,* Oxford University Press, Oxford, 1992

- Eisler R., *The Chalice and the Blade*, Harper Collins, 1987

- Ermoentrout G.B., Edelstein-Keshet L., *Cellular Automata Approaches to Biological Modelling*. J.Theor.Biol. 160:97-133, 1993

- Esté A., Cultura Replicatne (El orden semiocentrista), Barcelona: Gedisa, 1997

- Foucault M., *Knowledge and Power*. The Harvester Pres, Toronto. 1980

- Franks N., Gómez N., Goss S., Deneubourg J. L., T*he Blind Leading the Blind in Army Ant Raid Patterns: Testing a model of self-organization*. J. Insect Behavior 4:583-607, 1991

- Gordon D.M., *Ants at work : how an insect society is organized*, New York, New York :
Free, 1999

- Gotwald Jr. W.H., *Army Ants : the biology of social predation*, Ithaca : Comstock Pub. Associates, 1995

- Haken en *O que é a vida? 50 anos depois*. Sao Paulo, USP 2000

- Kauffman S., *The Origins of Order*, Oxford University Press, 1993

- Kauffman S. en *O que é a vida? 50 anos depois*. Sao Paulo, USP 2000

- Kull K., Tiivel T. (eds.) 1993. *Lectures in Theoretical Biology: The Second Stage*. Tallinn: Estonian Academy of Sciences, 52-62. En http://www.zbi.ee/~kalevi/vaheleht.htm

- Maturana H. R., *Ontology of Observing: the biological foundations of self consciousness and the physical domain of existence,* Texts in Cybernetics, American Society For Cibernetics, Felton, CA., 1988

- Moura L., *Trails, Ants and Anarchy,* 2002. En www.lxxl.pt/pdf/Trails.pdf

- Ortega Arjona J.L., *Architectural Patterns for Parallel Programming.* 1998

- Smith R.L., *A Natural History of the Sonoran Desert*, 2000. The University of California Press en: http://www.desertmuseum.org

- Solé R.V., Miramontes O., Goodwin B.C., *Oscillations and Chaos in Ant Societies*. J.Theor.Biol. 161:343-357, 1993

- Theraulaz G., Gautrais J., Camazine S., Deneubourg J.L., *The Formation of Spatial Patterns in Social Insects. Philosophical Transactions Royal Society London A*, Vol. 361 2003:1263-1282

- Varela, F., *The Embodied Mind: Cognitive Science and Human Experience*, MIT Press, 1991

- Watmough J., Edelstein-Keshet L., *A one-dimensional model of trail propagation by army ants*. J.Math.Biol. 33: 459-476, 1995

- Watmough J., Edelstein-Keshet L., *Modelling the formation of trail networks by foraging ants*. J.Theor.Biol. 176:357-371, 1995

- Wheatley M.J., Kellner-Rogers M., *The Irresistible Future of Orginizing. Strategy and Leadership*, agosto 1996

- Woodroffe J., Shiva and Shakta, 1918. En http://www.sacred-texts.com/tantra/sas/

```
/*   Autor:  Mauricio Flores Peña
     publicado bajo licencia Open Source
     mau.aum@gmail.com
                  -1,0

   -1,-1          XII        -1,1

                   |
            \ |   \ /   | /
           -- \  /  \  /  --
            \       ^      /
0,-1  IX  -- <<  ( * ) >>  -- III  0,1
            /              \
           -- o/  \o/  \o --
            / |   / \   | \

    1,-1           |          1,1

                   VI

                  1,0
*/
package modelo;

/** Representa una orientacion en el espacio */
public class Sentido{

    /** denota la direccion norte
     */  static final int N = 0;
    /** denota la direccion noroeste
     */  static final int NO = 1;
    /** denota la direccion oeste
     */  static final int O = 2;
    /** denota la direccion suroeste
     */  static final int SO = 3;
    /** denota la direccion sur
     */  static final int S = 4;
    /** dentoa la direccion sureste
     */  static final int SE = 5;
    /** denota la direccion este
     */  static final int E = 6;
    /** denota la direccion noreste
     */  static final int NE = 7;
    /** cordenada Y de este sentido como
     * componente direccional
     */ volatile int y;
    /** cordenada X de este sentido como
     * componente direccional
     */ volatile int x;
    /** representacion alterna de este sentido
     * en terminos de las variables direccionales
     * ya enumeradas
     */ volatile int w;

/** norte sur este oeste noreste noroeste sureste suroeste
  * @param z representa una de las 8 direcciones en una cuardricula respecto a un cuadro
  * central
  */public Sentido(int z){
      if( z>=0 && z<=7 ){
        switch(z){
           case N:
                     y = -1;
                     x = 0;
                     break;
           case S:
                     y = 1;
                     x = 0;
                     break;
           case E:
                     y = 0;
                     x = 1;
                     break;
           case O:
                     y = 0;
                     x = -1;
                     break;
           case NE:
                     y = -1;
                     x = 1;
                     break;
           case NO:
                     y = -1;
                     x = -1;
```

```
                    break;
          case SE:
                    y = 1;
                    x = 1;
                    break;
          case SO:
                    y = 1;
                    x = -1;
                    break;
        }
        w = z;
      }
    }

/** crea un Sentido en funcion de las coordenadas unitarias Y,X */
     public Sentido(int Y, int X){
             y=Y;
             x=X;
             if( Y==0 ){
                 if( X == 1)
                     w = E;
                 if( X == -1)
                     w = O;
             }
             if( Y==1 ){
                 if( X == 1)
                     w = SE;
                 if( X == -1)
                     w = SO;
                 if( X == 0)
                     w = S;
             }
             if( Y==-1 ){
                 if( X == -1)
                     w = NO;
                 if( X == 1)
                     w = NE;
                 if( X == 0)
                     w = N;
             }
     }
/** sirve para que este sentido transite
  * de representar un angulo al mismo
  * mas 45 graus
  * @return el entero que representa la direccion resultante
  */private int mas_pi_medios(){
        return (w+1)%8;
    }

/** sirve para que este sentido transite
  * de representar un angulo al mismo
  * menos 45 graus
  * @return el entero que representa la direccion resultante
  */private int menos_pi_medios(){
        if( w!=0 )
            return (w-1)%8;
         else return 7;
    }

/** dado una direccion regresa esta mas pi/2
  * @return el objeto que representa la direccion resultante
  * @see Sentido#mas_pi_medios
  */Sentido levo(){
       return new Sentido( mas_pi_medios() );
    }

/** dado una direccion regresa esta menos pi/2
  * @return el objeto que representa la direccion resultante
  * @see Sentido#menos_pi_medios()
  */Sentido dextro(){
        return new Sentido( menos_pi_medios() );
    }

/** regresa uno de los 8 enteros que representan las direcciones
  * @return la coordinada de este sentido
  */public int getAngulo(){
             return w;
    }

/**
  * @return la componente vertical
  */public int getY(){
        return y;
```

```
    }

/**
  * @return la componente horizontal
  */public int getX(){
        return x;
    }
}
package modelo;

import render.VisualRemoto;
import java.rmi.RemoteException;
import nodo.*;

/** representa un sitio en la estructura espacial de cada simulador
 */ public class Sitio{

    /** la coordenada Y del Sitio
     * en el Espacio local
     */ private final int y;

    /** la coordenada X del Sitio
     * en el Espacio local
     */ private final int  x;

    /** concentracion feromonal
     */ private float concentracion=1;

        private static float evaporacion;

        private boolean local_qry = true;

    /** referencia a posible ocupante
     */ protected Tremita ocupante;

        private static VisualRemoto remote_screen;

/** variable de estdo de clase que indica si el sitio enviara peticiones al servidor
     *  de visualizacion o la simulacion esta corriendo en modo silencioso
     */ private static boolean render = true;


/** constructor
  * @param j coordenada Y
  * @param i coordenada X
  */public Sitio(int j, int i) {
        y =j;
        x=i;
        concentracion = 1.0f;
        ocupante = null;
    }

/** asigna un visualizador a este nodo
  * @param v una referencia al Interfaz
  */public static void setInterfaz( VisualRemoto v ){
      remote_screen = v;
    }

/** invocado por bicho al ingresar a este Sitio al ser creado
  * (sin Sitio previo)
  * @param bicho el bicho a asignar a ocupante
  */ void recibe( Tremita bicho ){
       ocupante=bicho;
       concentracion = concentracion + 2*Tremita.getDeposicion();
       bicho.setSitio(this);
       if( !local_qry ){
           Simulador.addBicho(bicho);
           local_qry=true; // System.out.print("->");
         }
       if( render )
        try{ remote_screen.x_entra( Espacio.getNumRows()*Espacio.getCoorY() + y,
                                    Espacio.getNumCols()*Espacio.getCoorX() + x
                                  );
           } catch( Exception e ){ }
    }

/** invocado desde otro Sitio cuando recibe()
  * proporciona las coordenadas adecuadas
  * numtremitas--   feromona --
  */ void sale(){
        ocupante = null;
        concentracion = concentracion - Tremita.getDeposicion();
        if( render )
```

```
        try{ remote_screen.x_sale( Espacio.getNumRows()*Espacio.getCoorY() + y,
                                   Espacio.getNumCols()*Espacio.getCoorX() + x
                                 );
           }
          catch(Exception ef){}
    }

/** modifica su cantidad de feromona
  * @param cuanta cantidad de feromona a agregar
  */public void evapora(){
         concentracion =  concentracion*( 1 - evaporacion );
    }

/** obten variable de estado
  * @return la concentracion aqui
  */public float getConcentracion(){
        return concentracion;
    }

/** obten variable de estado
  * @return coordenada X
  */int getX(){
        return x;
    }

/** obtiene variable de estado
  * @return coordenada Y
  */int getY(){
        return y;
    }

/** obtiene variable de estado
  * @return numero de tremitas
  */public boolean sePuede(){
        if( ocupante == null )
            return true;
         else return false;
    }

/** asigna un valor booleano a partir del cual el sitio
  * solicita (o no) a IntefazVisual pinte un elemento
  * cuando recibe(un bicho)
  * @param do_render el booleano de acuerdo al cual habra o no
  * solicitudes enviadas a InsterfazVisual
  */public static void setRendering(boolean do_render){
        render = do_render;
    }

    public void setLocal(boolean b){
        local_qry=b;
    }

    boolean isLocal(){
        return local_qry;
    }

    public static void setEvaporacion(float e){
        evaporacion = e;
    }

   public static void EOT() throws RemoteException{
       if(render)
          remote_screen.acknow();
   }

   public int getBichosAngl(){
       return ocupante.getFrente().getAngulo();
   }
}
package modelo;

import java.rmi.RemoteException;
import java.util.Random;
import java.lang.reflect.Array;

import modelo.Sitio;
import nodo.Espacio;

/** representa cada tremita
 *  primordialmente se trata de un individuo
 *  con una orientacion (Sentido) y
 *  una modo de transito de una a otra posicion
 *  a partir de un algoritmo pensado en la percepcion
```

```
 *  de la concentracion feromonal espacial
 *  modificada a su vez por la deposicion de cada individuo
 */ public class Tremita{
// parametros del modelo

    /** cantidad de feromona quedetermina una probabilidad de .5=f(X)
         */ private static float xi_off;
    /** umbral de distincion diferencias perceptuales entre antenas
     *  antena1/antena2, valor maximo de responsividad
     */ private static float xi_fuzz;
    /** taze de crecimiento de la responsividad respecto al nivel feromonal
     */ private static float xi_rate;
    /** cantidad de feromona depositada por cada tremita
      * a su entrada a un Sitio
      */ private static float xi_depo;

    /** fuente de ruido
      */ private static Random azar = new Random();
    /** representa el modo de seguimiewnto osmotropico o clinotactil
      */ private static int modo;

//estado interno de cada particula

    /** objeto que representa la direccion (N,S,E,O,NE,NO,SE,SO) en
      * que se encuentra orientada la tremita
      */ private volatile Sentido frente;
                private volatile Sitio aqui;

/** constructor
  * @param aqui la posicion asignada inicialmente a la tremita
  * su orientacion es asignada al azar.  Inicializacion
  */public Tremita( Sitio aqui ){
        frente = new Sentido( azar.nextInt(8) );
        aqui.recibe(this);
    }

/** direccion, posicion inicializada especificamente
  * utilizado cuando son reconstruidas en un espacio remoto
  * @param zen una orientacion especifica de inicio
  * @param aqui una posicion de inicio especifica
  */public Tremita( Sentido zen, Sitio aqui ){  // NOT USED!!
        frente = zen;
        aqui.recibe(this);
    }

/** metodo estatico invocado al inicio para definir
  * los parametros, comunes a todas las tremitas
  * @param f incremento de la fidelidad con la concentracion local
  *  controla probabilidad de aplicacion del algoritmo de movimiento
  * @param d cantidad de feromona depositada por tremita
  *           en cada Sitio visitado
  * @param r la modo de movimiento a ser utilizada en la simulacion
  * @param x denota la dependencia de la fidelidad respecto a la concentracion feromonal
  */public static void setParametros(
                                     final float fuzz,
                                     final float dep,
                                     final int m,
                                     final float v,
                                                                    final float offset
                                    ){
        xi_fuzz= fuzz;
        xi_depo = dep;
        modo = m;
        xi_rate = v;
                xi_off = offset;
    }

/** llamado desde clase coordinadora Simulador
  * coordina los procesos (sondear, ponderar, elegir, acccionar)
  * implicados en un movimiento espacial
  * @return true si entro a un sitio local nuevo
  */public boolean camina(){

        Sentido newSentido = null;
        try{
                        float N  = Espacio.getConcentracionEn(  aqui.getY()+frente.getY(),

         aqui.getX()+frente.getX()
                                                                                                );

             if(  responsiva(  aqui.getConcentracion() ) && N>0  ){
                  float NE = Espacio.getConcentracionEn(  aqui.getY()+frente.dextro().getY(),
                                                          aqui.getX()+frente.dextro().getX()
```

```
            );
                  float NO = Espacio.getConcentracionEn(  aqui.getY()+frente.levo().getY(),
                                                          aqui.getX()+frente.levo().getX()
                                                       );
                  float E  = Espacio.getConcentracionEn(  aqui.getY()+frente.dextro().dextro().getY(),
                                                          aqui.getX()+frente.dextro().dextro().getX()
                                                        );
                  float O  = Espacio.getConcentracionEn(  aqui.getY()+frente.levo().levo().getY(),
                                                          aqui.getX()+frente.levo().levo().getX()
                                                        );
                  newSentido = alli(N,NO,NE,E,O);
                          }
                  else newSentido = alla();
                 return entra(newSentido);
           }
                  catch( NotMovingException nme ){}
             catch( Exception oe ){
                  frente = newSentido;
                          }
      return true;

    }

    private boolean entra(Sentido rumbo) throws Exception{
        Sitio neo_site = Espacio.elSitio( aqui.getY() + rumbo.getY(), aqui.getX() + rumbo.getX() );
        if( neo_site.sePuede() ){
            frente = rumbo;
                        aqui.sale();
            neo_site.recibe(this);

            return neo_site.isLocal();
          }
         else throw new Exception();
    }


/** aplica la modo elegida en funcion de las concentraciones
  * feromonales de las casillas vecinas
  * @return una instancia de Sentido representando
  *  la direccion en y hacia la que se intentara mover
  * @throws NotMovingException cuando solo gira sin desplazarse
  */private Sentido alli(float N, float NO, float NE, float E, float O) throws Exception {

        switch( modo ){
                case 0:  // osmotropic

                    if( (NE+E/2)/(NO+O/2) > 1 + xi_fuzz )

                                                return frente.dextro();

                    if( (NO+O/2)/(NE+E/2) > 1 + xi_fuzz )

                                                return frente.levo();

                                        break;
                    case 1: // clinotactil

                    if( N >= Math.max( Math.max(NO,O), Math.max(NE,E) )  )
                                                return frente;

                                    if( NO >= Math.max( E,O ) && NO > NE )
                                                return frente.levo();

                                            if( NE >= Math.max( E,O ) && NE > NO )
                                    return frente.dextro();

                                    if( O > Math.max(NE,NO) && O > E )
                                                return frente.levo().levo();

                                            if( E > Math.max(NO,NE) && E > O )
                                    return frente.dextro().dextro();

                                            break;
                            }
        return frente;
    }

/** determina si se aplicara o no la modo de
  * seguimiento con dos modos de funcionamiento:
  * en funcion de la concentracion feromonal o
  * mediante una probabilidad fija de error
  * @param xi la concentracion local promedio
```

```
  * @return if fidelidad > random
  */private boolean responsiva(float xi){
      double fi= (1 - xi_fuzz)/( 1 + Math.exp( xi_rate*(xi_off - xi) ) );///Math.PI + .5f;
          if( fi > Math.random() )
          return true;
        else return false;
   }

/** regresa una direccion de movimiento de manera
  * aleatoria N:NO:NE:O:E con probabilidades 4:3:3:2:2
  * @return un Sentido de manera azarosa
  */private Sentido alla() {
        int onde = azar.nextInt(8);
        switch( onde ) {
                case 0 :
                case 1 :
                case 2 :
                          frente = frente.levo().levo().levo();
                                              break;
                case 3 :
                case 4 :
                case 5 :
                           frente = frente.dextro().dextro().dextro();
                                               break;
                                     case 6 :
                case 7 :
                           frente = frente.dextro().dextro().dextro().dextro();
                                               break;
                          }
        return frente;
    }

 private int distancia(int a, int b){
        int distancia = Math.abs(a-b);
        if( distancia>4 )
            distancia = 8 - distancia;
        return distancia;
    }

 public double index(){
        int vecinos = 0;
        int vecinoWay = 0;
        int index = 0;
        Sentido paonde=null;

        for( int i=0; i<8; i++ ){
             paonde = new Sentido(i);
             if( (vecinoWay = Espacio.bichosAnglin( aqui.getY()+paonde.getY(),
aqui.getX()+paonde.getX() )) > 0 ){
                  index += distancia(   frente.getAngulo(), vecinoWay );
                  vecinos++;
                }
            }
        if( vecinos == 0 )
            return 0;
        return index/vecinos;
    }


/** acceso a variable de estado de orientacion
  * @return la orientacion actual del bicho
  */ Sentido getFrente(){
        return frente;
    }

    void setSitio(Sitio s){
        aqui = s;
    }

/** acceso a la cantidad de feromona depositada por termita por unidad de tiempo
  * @return la deposicion
  */ static float getDeposicion(){
         return xi_depo;
     }

}

class NotMovingException extends Exception{}
package nodo;

import java.util.Hashtable;
import java.util.Enumeration;
```

```
import java.rmi.RemoteException;

import render.VisualRemoto;
import modelo.Tremita;
import modelo.Sentido;
import modelo.Sitio;

/** controla la posicion de las temitas en este nodo y
    adecua l aepsacilidad al entorno dilstribuido, e.d.
 *  administra la posicion relativa y peticiones en referencia
    a otros espacios y al Interfaz.
 *  Asi canaliza las peticiones de concentraciones remotas,
 *  solicitudes de emigracion, mensajes al fin de cada ciclo
 *  para sincronizacion.
 */ public class Espacio {

    /** un arreglo que contiene los Sitios asignados a este Espacio
     */ protected static Sitio[][] casilla;

    /** metacoordenada espacial, e.d.
     *  identificador del simulador en los renglones de la gradilla de nodos
     */ private static int Y;

    /** metacoordenada espacial, e.d.
     *  identificador de simulador en las columnas de la gradilla de nodos
     */ private static int X;

        private static volatile float max;
/** *Constructor
  *  @param y metacoordenada espacial
  *  @param x metacoordenada espacial
  *  @param casilla.length "ancho" del Espacio
  *  @param casilla[0].length "alto" del Espacio
  *  @param remote_espacios los espacios en nodos vecinos
  */private Espacio( int y, int x, int tamanhoY, int tamanhoX, float evaporacion ){
        Y=y;
        X=x;
        Sitio.setEvaporacion( evaporacion );
        casilla = new Sitio[tamanhoY][tamanhoX];
        for( int j=0; j<casilla.length; j++)
             for( int i=0; i<casilla[0].length; i++)
                  casilla[j][i]=new Sitio(j,i);
   }

/** metodo que crea una instancia de Espacio con los
  *  parametros necesarios a peticion del Simulador
  *  @param y la coordinada de este Espacio respecto a su casilla
  *   en la gradilla de nodos
  *  @param x la coordenada de este Espacio en la gradilla de nodos
  *  @param casilla.length la cantidad de renglones de celdas en este espacio
  *  @param casilla[0].length cantidad de columnas de celdas en esta suibgradilla
  *  @param remote_espacios acumula referencias a los vecinos de esta subgradilla
  *   para accederlos en caso de sondeo o movimiento remoto
  */public static void setUp( int y, int x,
                              int sizeY, int sizeX,
                              float evaporacion ){
        new Espacio(y,x,sizeY,sizeX, evaporacion);
    }



/** "evapora" feromona de cada Sitios
  */ public static void ferosevapora(){
         for( int n=0; n<casilla.length; n++ )
              for( int m=0; m<casilla[0].length; m++ )
                   casilla[n][m].evapora();
     }


/** manera en que se puede obtener
  * la referencia a un Sitio en funcion del calculo
    de sus coordenadas locales
  * @return regresa referencia al Sitio
  * en las coordenadas proporcionadas
  * @param y Y
  * @param x X
  */public static Sitio elSitio(int y, int x) {
        try{
             return casilla[y][x];
           }
          catch( ArrayIndexOutOfBoundsException ex){
                 //System.out.print("x");
                  if( Simulador.hayVecinos() )
```

```
                try{
                     boolean so_lateral = y > -1 && y < casilla.length;
                     if(x == casilla[0].length){
                        if(so_lateral) // "este"
                           return Simulador.getSitioRemoto( Y+""+(X+1), y,0);
                         else if(y ==-1)// "noreste"
                                 return Simulador.getSitioRemoto( (Y-1)+""+(X+1), casilla.length-1,0);
                               else return Simulador.getSitioRemoto( (Y+1)+""+(X+1), 0,0);
                                 //   "sureste"
                      }
                     else if(x == -1){
                             if(so_lateral)  // "oeste"
                                return Simulador.getSitioRemoto( Y+""+(X-1), y,casilla[0].length-1);
                              else if(y==-1) // "noroeste"
                                      return Simulador.getSitioRemoto( (Y-1)+""+(X-1), casilla.length-
1,casilla[0].length-1);
                                  else return Simulador.getSitioRemoto( (Y+1)+""+(X-
1),0,casilla[0].length-1);
                                      //  "suroeste"
                                }
                               else if(y==-1) //  "norte"
                                       return Simulador.getSitioRemoto( (Y-1)+""+X,casilla.length-1,x);
                                     else if(y==casilla[0].length)
                                             return Simulador.getSitioRemoto( (Y+1)+""+X,0,x);
                                         // "sur"
                     } catch(Exception remi){ }
             }
        return null;
    }

/** provee a las tremitas con el mecanismo para obtener
  *  los valores fermonales de las celdas vecinas
  *  ya sean locales o remotas locales y remotos
  *  @param y coordenada Y de un Sitio
  *  @param x coordenada Y de un Sitio
  *  @return la concentracion correspondiente
  *  al Sitio en la coordenada indicada
  */public static float getConcentracionEn(int y, int x){
       try{ return casilla[y][x].getConcentracion(); }
         catch( Exception aiobe){
                    try{
                        boolean so_lateral = y > -1 && y < casilla.length;
                        if(x == casilla[0].length){
                           if(so_lateral) // "este"
                              return Simulador.getConcentracionRemota( Y+""+(X+1),y,0);
                            else if(y ==-1)// "noreste"
                                    return Simulador.getConcentracionRemota( (Y-
1)+""+(X+1),casilla.length-1,0);
                                 else return Simulador.getConcentracionRemota( (Y+1)+""+(X+1),0,0);
                                //   "sureste"
                          }
                         else if(x == -1){
                                 if(so_lateral)  // "oeste"
                                    return Simulador.getConcentracionRemota( Y+""+(X-
1),y,casilla[0].length-1);
                                  else if(y==-1) // "noroeste"
                                          return Simulador.getConcentracionRemota( (Y-1)+""+(X-
1),casilla.length-1,casilla[0].length-1);
                                      else return Simulador.getConcentracionRemota( (Y+1)+""+(X-
1),0,casilla[0].length-1);
                                      //  "suroeste"
                                }
                               else if(y==-1) //  "norte"
                                       return Simulador.getConcentracionRemota( (Y-
1)+""+X,casilla.length-1,x);
                                     else if(y==casilla[0].length)
                                             return Simulador.getConcentracionRemota((Y+1)+""+X,0,x);
                                         // "sur"
                     } catch(Exception remi){ }
          }
          return -2.0f;
    }

/**
  * @return el numero de renglones en este Espacio
  */public static int getNumRows(){
       return casilla.length;
    }

/**
  * @return el numero de columnas en este Espacio
  */public static int getNumCols(){
       return casilla[0].length;
```

```
    }

/**
  * @return la coordenada Y
  */public static int getCoorY(){
        return Y;
    }

/**
  * @return la coordenada X
  */public static int getCoorX(){
      return X;
    }

    public static int bichosAnglin(int y, int x) {
        try{
             Sitio s = elSitio(y,x);
          //   s.setLocal(true);
             return s.getBichosAngl();
           }
          catch( Exception e){
                 return -1;
              }
    }


}
package nodo;

import render.VisualRemoto;
import modelo.Sitio;

public interface NodoRemoto  extends java.rmi.Remote{

    // regresa la concentracion en el lugar especificado
    // para un otro simulador que solicite
    Sitio getSitioRemoto(int y, int x)
        throws java.rmi.RemoteException;

    float getConcentracionRemota(int y, int x)
        throws java.rmi.RemoteException;

    void acknow()
        throws java.rmi.RemoteException;

    double index()
        throws java.rmi.RemoteException;

        long n()
            throws java.rmi.RemoteException;

    // recibe los parametros de la simulacion y (re)inicia la simulacion asi
    void setUp( VisualRemoto vr,
                int tamanhoY,
                int tamanhoX,
                int no_tremitas,
                float fi_min,
                float deposicion,
                float evaporacion,
                float xi_rate,
                int detect_rule,
                               float xi_off
               ) throws java.rmi.RemoteException,
                        java.rmi.NotBoundException,
                        java.net.MalformedURLException;

    void setRendering(boolean toggle)
         throws java.rmi.RemoteException;

    float[][] getConcentraciones()
           throws java.rmi.RemoteException;

    void begin()throws java.rmi.RemoteException;

    void end() throws java.rmi.RemoteException;

    void resetVisual( VisualRemoto v) throws java.rmi.RemoteException;

}
package nodo;

import java.io.FileInputStream;
import java.net.MalformedURLException;
```

```
import java.util.Hashtable;
import java.util.Random;
import java.util.Collections;
import java.util.Properties;
import java.util.LinkedList;
import java.util.Enumeration;

import java.rmi.Naming;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.rmi.Remote;
import java.rmi.RMISecurityManager;

import render.VisualRemoto;
import modelo.*;
import modelo.Sitio;

/** Representa una de las subgradillas que componen la gradilla espacial de la simulacion
  * identifica y se comunica con las subgradillas vecinas (para intercambiar termitas e informacion
feroespacial)
  * mediante un sistema de coordenadas que definen la vecindad potencial
  * y una jashtabla con referencias a los vecinos que de hecho estan disponibles
    y permiten la ejecucion sincronica de la simulacion sobre los nodos que la albergan */
public class Simulador extends Thread implements NodoRemoto {

/** los bichos que participan
  */private static LinkedList tremitas;
/** coordenadas absolutas des este Simulador
  */private String self;
    /** guarda los pares (coordenada, NodoRemoto)
        e.d., referencias a espacios vecinos
     *  para exportarles bichos
     *  y sondear teleferoconcentraciones */
        static private Hashtable nodos;
/** ruta hasta el archivo de configuracion inicial
  */private String props_path;
/** numero de nodos que s e habn reportado
  *  para finalidades de sincronizacion
  */private int nodos_ack;
/** variable de control, ciclos ejecutados */
    private long ciclos;
/** variable de control, simulacion en ejecucion */
    private boolean ya_estubo;

        //private int num_vecinos;
/** lee un archivo de configuracion
  * se exporta como objeto remoto
  * @param properties_path ruta absoluta del directorio que contiene
  * los archivos con coordenadas e IPs de este simulador y sus vecinos
  */public Simulador(String properties_path){
         props_path = properties_path;
         Properties self_id = new Properties();
         try{
              self_id.load( new FileInputStream(props_path+"self"));
           } catch( Exception ioe ){
                   System.out.println(ioe.getMessage()+"no se cargo self IP file");
                   ioe.printStackTrace();
                 }
         self = (String)self_id.propertyNames().nextElement();
         try{
              UnicastRemoteObject.exportObject(this);
              Naming.rebind("//"+self_id.getProperty(self)+"/"+self,this);
              System.out.println("nodo "+self+"@"+self_id.getProperty(self)+" exportado");
           } catch( Exception rem ){
                   System.out.println("fallo exportar o registrar al directorio
RMI\n"+rem.getMessage());
                   rem.printStackTrace();
                 }
    }

   /**recibe los parmetros por parte del usuario a traves de RMI desde el Interfaz
    * obtiene las referencias a los simuladores vecinos de acuerdo a
    * un archivo de propiedades y los pasa al espacio
    * @param tamanhoY numero de renglones de celdas en cada simulador
    * @param tamanhoX nunmero de columnas de celdas en cada simulador
    * @param no_tremitas cantidad de bichos por nodo
    * @param fi_fuzz probabilidad minima con la que los bichos aplicaran
    * su regla de movimiento
    * @param deposicion cantidad de feromona depositada por unidad
    * de tiempo por termita
    * la diferencia feromonal se vuelve indperceptible
```

```
    * para una tremita
    * @param rule una regla de movimieto especifica
    * @param xi_dependent determina si la fidelidad a la aplicacion del algoritmo es funcion de la
deposicion
    * @throws RemoteException algun problema en la comunicaclion entre nodos
    * @throws NotBoundException el registro de servicios remotos no tiene este registrado
    * @throws MalformedURLException el URL que identifica el servilcio no esta bien especificado
    */ public void setUp( VisualRemoto v, int tamanhoY, int tamanhoX, int no_tremitas,
                        float xi_fuzz, float deposicion, float evaporacion, float xi_rate,
                        int modo, float xi_off ) throws java.rmi.RemoteException,
                                                                                    java.rmi.NotBoundExcepti
on,
                                                                                    java.net.MalformedURLExc
eption {
            Sitio.setInterfaz( v );
            System.out.println("visual server contactado");
            System.out.println( "\nnum renglones: "+ tamanhoY+
                                "\nnum columnas: "+ tamanhoX+
                                "\nnum tremitas: "+ no_tremitas+
                                "\nxi fuzz: "+xi_fuzz+
                                "\ndeposicion: "+deposicion+
                                "\nevaporacion: "+evaporacion+
                                "\nmodo: "+modo+
                                "\nxi rate: "+xi_rate+
                                                        "\nxi offset: "+xi_off
                            );
            Properties others = new Properties();
            tremitas = new LinkedList();
            try{
                 others.load( new FileInputStream(props_path+"vecinos"));
               } catch( Exception ioe){
                        System.out.println(ioe.getMessage()+"no se cargo vecinos IP file");
                     }
            nodos = new Hashtable();
            Enumeration noms = others.propertyNames();
            while( noms.hasMoreElements() ){
                   String otherServiceNom =  (String)noms.nextElement();
                   nodos.put( otherServiceNom, Naming.lookup( "rmi://"
                                                            + others.getProperty(otherServiceNom) + "/"
                                                            + otherServiceNom
                                                          )
                            );
                   System.out.println(otherServiceNom+"@"+others.getProperty(otherServiceNom)+": nodo
contactado");
                }

           // nodos_ack=num_vecinos;
            System.out.println("numero de vecinos "+nodos.size()+"\n");
            try{
                  Espacio.setUp( Integer.parseInt(self.substring(0,1)),
                                 Integer.parseInt(self.substring(1,2)),
                                 tamanhoY,tamanhoX, evaporacion
                               );

               } catch( Exception ex){
                        System.out.println(ex.getMessage());
                        ex.printStackTrace();
                      }
            Tremita.setParametros(xi_fuzz,deposicion,modo, xi_rate, xi_off);
            for( int k=0; k<no_tremitas; k++ )
                 maisUma(new Random());
    }

/** utilizado para ingresar tremitas al espacio
  * @param azar genera un estado inicial aleatorio
  */private  void maisUma(Random azar){
         Sitio aqui = Espacio.elSitio( azar.nextInt(Espacio.getNumRows()),
                                       azar.nextInt(Espacio.getNumCols())
                                     );
         if( aqui.sePuede() ){
             tremitas.add(new Tremita(aqui) );
             return;
           }
          else{
                maisUma(azar);
                return;
              }
     }

/** cicla la simulacion de acuerdo a la logica distribuida del programa
  * manda a mover tremitas, exporta emigrantes y reintegra rechazos
  * @throws RemoteException problemas en la red
  */public void cicla() throws java.rmi.RemoteException {
```

```
       // neoTurno( );
        Tremita esta = null;
        for( int i=0; i<tremitas.size(); i++){
             if(  ( esta = ((Tremita)tremitas.get(i)) ).camina()  );
                    else tremitas.remove( esta );
               }
            Collections.shuffle(tremitas);
        Espacio.ferosevapora();

        ciclos++;
    }

/** Inicia la ejecucion de la simulacion en cada nodo
  * @throws RemoteException network error
  */public void begin()throws java.rmi.RemoteException {
      // this.setPriority( Thread.MAX_PRIORITY );
       this.start();
    }

   public void run(){
      nodos_ack = nodos.size();
          while( !ya_estubo )
              if( nodos_ack==nodos.size() )
                  try{
                       cicla();
                       neoTurno(  );
                      nodos_ack=0;
                                      }
                   catch( Exception e ){
                                         System.out.println( "error "+ e.getMessage() );
                                                e.printStackTrace();
                                               }
          System.exit(0);
   }

/** metodo invocado por nodos vecinos para sincronizar la ejecucion global
  * @throws RemoteException error en red
  */public void acknow() throws RemoteException{
       nodos_ack++;//cicla();

   }

/** finaliza la ejecucion del programa
  * @throws RemoteException problemas de conectividad
  */public void end() throws java.rmi.RemoteException {
         ya_estubo=true;
    }

/** habilita/deshabilita el rendering del estado de este Simulador
  * @param x on o off
  * @throws RemoteException problemas de conectividad
  */public void setRendering(boolean b) throws java.rmi.RemoteException {
        Sitio.setRendering(b);
    }

/** proporciona el estado de la gradila al habilitarse el rendereo
  * de la simulacion
  * @throws RemoteException problemas de conectividad
  * @return los valores de concentracion de las casillas * de este nodo
  */public float[][] getConcentraciones() throws java.rmi.RemoteException{
        float[][] concentraciones = new float[Espacio.getNumRows()][Espacio.getNumCols()];
        for( int j=0;j<Espacio.getNumRows();j++)
             for( int i=0;i<Espacio.getNumCols();i++)
                  concentraciones[j][i] = Espacio.getConcentracionEn(j,i);
        return concentraciones;
    }

/** reconecta a este simulador con un visualizador remoto activo */
    public void resetVisual( VisualRemoto v) throws java.rmi.RemoteException {
        Sitio.setInterfaz(v);
    }

    public double index() throws java.rmi.RemoteException{
        double index = 0.0d;
        for( int i=0; i<tremitas.size(); i++)
             index +=  ( (Tremita)tremitas.get(i) ).index();
        System.out.println("individuos= "+tremitas.size());

        System.out.println("ciclos= "+ciclos);
        return index/tremitas.size();
    }

/** Crea este objeto simulador y lo pone a la escucha de
```

```
    parametros e inicio a traves del visualizador
  * @param argv ruta absoluta al directorio con los archivos de configuracion
  * respecto a la propia localizacion y la de los nodos vecinos
  * @throws Exception cualquier caso  no contemplado
  */public static void main(String argv[]) throws Exception{
        new Simulador(argv[0]);
    }

  public Sitio getSitioRemoto(int y, int x) throws java.rmi.RemoteException {
      Sitio s = Espacio.elSitio(y,x);
      s.setLocal(false);
      return s;
  }

  public static void addBicho(Tremita t){
      tremitas.add(t);
  }

  /** regresa la concentracion solicitada
  * a partir de un Sitio de un Espacio adyacente
  * @param y coordenada Y
  * @param x coordenada X
  * @return concentracion en la direccion indicada
  * @throws RemoteException posible error en la obtencion de un valor remoto
  */public float getConcentracionRemota(int y, int x)
                throws java.rmi.RemoteException {
        return Espacio.getConcentracionEn(y,x);
    }

  static Sitio getSitioRemoto(String id, int y, int x) throws RemoteException{
      return ((NodoRemoto)nodos.get( id )).getSitioRemoto(y,x);
  }

  static float getConcentracionRemota( String id, int y, int x) throws RemoteException{
      return ((NodoRemoto)nodos.get( id )).getConcentracionRemota(y,0);
  }

/**  encapsula las instrucciones de sincronizacion con
  *  los otros nodos
  */ public static void neoTurno( ) throws RemoteException{
         Enumeration synclist = nodos.elements();
         try{
              Sitio.EOT();
              while( synclist.hasMoreElements() )
                     ((NodoRemoto)synclist.nextElement()).acknow();

            } catch( Exception eofe ){ System.exit(0); }
     }

  static boolean hayVecinos(){
      return !nodos.isEmpty();
  }

  public long n(){
        return ciclos;
  }

/** regresa la referencia remota
   * @param  coordenadas  de algun vecino
   * @return referencia al Espacio vecino en cuestion o null si no existe
   */private NodoRemoto getVecino(String coordenadas){
        return (NodoRemoto)nodos.get(coordenadas);
     }
}
package render;

import java.io.FileInputStream;
import java.io.BufferedReader;
import java.io.InputStreamReader;

import java.awt.BorderLayout;

import java.awt.Frame;
import java.awt.Panel;

import java.util.Hashtable;
import java.util.Enumeration;
import java.util.Properties;
import java.util.StringTokenizer;
import java.util.Collections;

import java.rmi.Naming;
import java.rmi.RemoteException;
```

```
import java.rmi.server.UnicastRemoteObject;

import nodo.NodoRemoto;

/** es el servidor que registra los cambios
 * ocurridos en y reportados por cada nodo
 * para exhibirlos en la pantalla graficamente
 */ public class Interfaz extends Thread implements VisualRemoto{


/** IP (y puerto) del nodo donde corre ete simulador */
    private String IP;
/** referencias a los nodos de la simulacion: start/stop */
    private Hashtable nodos;
/** la representacion grafica de todas las casillas*/
    static private SitioVisual[][] casillas;
/** numero de renglones en cada subgradilla */
    static int cell_cols;
/** numero de columnas en cada subgradilla */
    static int cell_rows;
/** ventanas que representan (c/u) un nodo distinto de la simulacion */
    static private Frame repNodal;
/** numero de procesadores (sobre eje Y) */
    static int nodos_cols;
/** numero de procesadores (sobre eje X) */
    static int nodos_rows;
//  pudiera desapareer en un modelo mas desacoplado                                    !!!>
    private static boolean render = true;
/** controla la terminacion */
    protected boolean yaestubo;
    private float index;
    double avg;
        double max;
        int nodos_ack;


     /** post: Interfaz de usuario lista para aceptar los parametros de inicio de simulacion
      * @param IP (y pto) que informara a cada simulador
      * @param nodos los nodos de la simulacion
      * @param nodos_cols el numero de renglones de nodos
      * @param nodos_rows el numero de columnas de nodos
      * @param deposicion la deposicion feromonal (para representar deposiciones graficamente)
      * @param cell_rows la cantidad de renglones de casillas en un solo nodo
      * @param cell_cols la cantidad de columnas de casillas en un solo nodo
      * @param fi_fuzz la incertidumbre en la aplicacion de las reglas de movimiento
      * @param num_tremitas cantidad de bichos por simulador
      * @param regla la regla con la cual calcular el siguiente sitio
      * @param xi_dependent si la fidelidad depende de la concentracion feromonal local
      */public Interfaz( String IP, Hashtable nodales, int nodos_rows,
                        int nodos_cols, int cell_rows, int cell_cols,
                        float deposicion, float fi_fuzz, float evaporacion,
                        int num_tremitas, int modo, float xi_rate, float xi_off,
                        boolean isNew ){
            try{
                UnicastRemoteObject.exportObject(this);
                Naming.rebind("//"+IP+"/Interfaz",this);
                System.out.println(IP+": exportado y registrado");
               } catch( Exception rem){
                        System.out.println("fallo exportar o registrar Interfaz\n"+rem.getMessage());
                        rem.printStackTrace();
                        System.out.println(rem.getCause().toString());
                       }
         this.IP= IP;
         this.nodos = nodales;
         this.nodos_cols = nodos_cols;
         this.nodos_rows = nodos_rows;
         this.cell_cols = cell_cols;
         this.cell_rows = cell_rows;

        casillas = new SitioVisual[nodos_rows*cell_rows][nodos_cols*cell_cols];
        repNodal = new Frame(  );
        repNodal.setLayout(new BorderLayout());
        Panel ghost = null;

        for( int n=0; n<nodos_rows; n++)
             for( int m=0; m<nodos_cols; m++){

                       ghost = new Panel(new java.awt.GridLayout(cell_rows, cell_cols));

                       for( int j=0; j<cell_rows; j++)
                            for( int i=0; i<cell_cols; i++){
                                 casillas[n*cell_rows+j][m*cell_cols+i] = new SitioVisual();
                                 ghost.add(casillas[n*cell_rows+j][m*cell_cols+i]);
```

```
                            }

                      ghost.setSize(11*cell_cols,11*cell_rows);
                      repNodal.add(ghost,"Center");

                    }
        repNodal.setSize(11*cell_cols,11*cell_rows);

        if( isNew ) inicializaNodos( num_tremitas, fi_fuzz, deposicion, evaporacion, modo, xi_rate, xi_off
);
          else  recontactaNodos();

        SitioVisual.setUp(deposicion, evaporacion);
        //setPriority( Thread.MAX_PRIORITY );

        start();
        }

/** transmite a cada Simulador en cada nodo de la simulacion los parametros iniciales
  * y arranca sus respectivos hilos de ejecucion
  */private void inicializaNodos( int num_tremitas, float xi_fuzz, float deposicion, float evaporacion,
                                  int modo, float xi_rate, float xi_off ){
        Enumeration os_nodos=nodos.elements();
        try{
             while( os_nodos.hasMoreElements() )
                    ((NodoRemoto)os_nodos.nextElement()).setUp( this,
                                                                cell_rows,
                                                                cell_cols,
                                                                num_tremitas,
                                                                xi_fuzz,
                                                                deposicion,
                                                                evaporacion,
                                                                xi_rate,
                                                                modo,

                       xi_off
                                                               );
                  os_nodos = nodos.elements();
                  while( os_nodos.hasMoreElements() )
                       ((NodoRemoto)os_nodos.nextElement()).begin();
                                 activaRendering();
                }
              catch( Exception ree){
                     System.out.println("error al inicializar nodo \n"+ree.getMessage());
                     ree.printStackTrace();
                     os_nodos = nodos.elements();
                     while( os_nodos.hasMoreElements() )
                            try{
                                 ((NodoRemoto)os_nodos.nextElement()).end();
                             }
                              catch( Exception reee){
                                     System.out.println( "error al finalizar nodo \n"+reee.getMessage() )
;
                                     reee.printStackTrace();
                                }
                     System.exit(0);
                    }
      }

/** Se asigna a los nodos como VisualRemoto y asi poder recibir el estado de cada nodo
  */private void recontactaNodos() {
          Enumeration os_nodos = nodos.elements();
          try{
               while( os_nodos.hasMoreElements() )
                     ((NodoRemoto)os_nodos.nextElement()).resetVisual(this);
                          activaRendering();
             } catch( RemoteException x){
                     System.out.println("unset visual");
                     System.out.println(x.getMessage());
                    }
      }

/** metodo con el que se invocan los cambios en el display
  * se borra la hormiga en las preCoordenadas
  * y se incluye en las postCoordenadas con orientacion w
  * @param preY coordenada verticval de donde sale un bicho
  * @param preX coordenada horizontal de donde sale un bicho
  * @throws RemoteException network error
  */public synchronized void x_sale(int preY, int preX) throws java.rmi.RemoteException{
        casillas[preY][preX].quitala();
    }

/** responde a una solicitud de rendereo y la canaliza al SitioVisual apropiado
```

```
  * @param postY id vertical de identificacion del sitio a renderear ocupacion
  * @param postX id horizontal de identificacion del sitio a renderear ocupacion
  * @throws RemoteException network error
  */public synchronized void x_entra( int postY, int postX ) throws java.rmi.RemoteException{
        casillas[postY][postX].ponla();
    }

/** sincroniza la ejecucion de la simulacion con el visualizador */
    public void acknow( ) throws java.rmi.RemoteException {
           nodos_ack++;
           if( nodos_ack == nodos_rows*nodos_cols ){
                   avg=0.0;
                   max=0.0;
           for( int n=0; n<nodos_rows; n++)
                for( int m=0; m<nodos_cols; m++)
                     for( int j=0; j<cell_rows; j++ )
                          for( int i=0; i<cell_cols; i++ ){
                                                          max = Math.max( max,
casillas[n*cell_rows+j][m*cell_cols+i].getConcentracion() );
                                avg += casillas[n*cell_rows+j][m*cell_cols+i].getConcentracion();
                                casillas[n*cell_rows+j][m*cell_cols+i].evapora();
                               }
           avg = avg / (casillas[0].length*casillas.length);
           SitioVisual.setMx(max);
           nodos_ack = 0;
                 }
        }

    String getIP(){
       return IP;
    }

/** metodo para activar la funcion de
  * visualizacion para la simulacion
  * activando la funcion de reportar
  * el estado de los elementos de cada nodo
  */private void activaRendering(){
        NodoRemoto esse = null;
        String id = null;
        float[][] esses_values = new float[cell_rows][cell_cols];
            repNodal.setVisible(true);
            for( int n=0; n < nodos_rows; n++)
             for( int m=0; m < nodos_cols; m++)
                  try{

                      esse = (NodoRemoto)nodos.get(n+""+m);
                      esses_values
 = esse.getConcentraciones();
                      for( int j=0; j<  cell_rows; j++ )
                           for( int i=0; i<  cell_cols; i++ )
                               casillas[n*cell_rows+j][m*cell_cols+i].setConcentracion(
esses_values[j][i]  );
                      esse.setRendering(true);

                      System.out.println( "reactivose: "+n+""+m );
                      System.out.println( "rows: "+esses_values.length+" cols: "+esses_values[0].length )
;
                    }
                   catch( Exception xyz ){
                          System.out.println( "error al reactivar nodo \n"+xyz.getMessage() );
                          xyz.printStackTrace();
                        }

    }

/** desactiva la funcion de rendereo
  * notificando a los nodos de
  * detener solicitudes de rendereo
  */private void deactivaRendering(){
       repNodal.setVisible(false);
        for( int n=0; n < nodos_rows; n++)
             for( int m=0; m < nodos_cols; m++)
               try{
                    ((NodoRemoto)nodos.get(n+""+m)).setRendering(false);

                     System.out.println( "deactivose: "+n+""+m );
                  }
                catch( Exception xyz ){
                       System.out.println( "error al deactivar nodo \n"+xyz.getMessage() );
                       xyz.printStackTrace();}
    }

    public void run(){
```

```
       BufferedReader uin = new BufferedReader(new InputStreamReader(System.in));
       String cmnd = "omtatsat";

        Enumeration os_nodos=nodos.elements();
        try{
             while( os_nodos.hasMoreElements() )
                    index += ((NodoRemoto)os_nodos.nextElement()).index();
           }
          catch(Exception e){  System.out.println("index perdido\n"+ e.getMessage()); e.printStackTrace();
}
        System.out.println("indice: "+index/nodos.size());

                while( !cmnd.equals("ya") ){
               if( cmnd.equals("render") ){
                   if( render )
                       deactivaRendering();
                     else  activaRendering();
                   render = !render;
                 }
               if( cmnd.equals("desliga") ){
                   deactivaRendering();
                   desliga();
                 }
               if( cmnd.equals("index") ){
                   index=0.0f;
                          try{
                                   NodoRemoto cikla = null;
                   os_nodos=nodos.elements();
                   while( os_nodos.hasMoreElements() )
                          index += (  ( cikla=(NodoRemoto)os_nodos.nextElement() )  ).index();
                   System.out.println("indice= "+index/nodos.size());
                          System.out.println( "ciclos= "+cikla.n() );
                          if( render ){
                          System.out.println("avg= "+avg);
                                     System.out.println("max= "+max);
                                     }
                                 }
                               catch(Exception ex){  System.out.println("index perdido\n"+
ex.getMessage()); ex.printStackTrace();  }
                    }

               try{
                    cmnd = uin.readLine();
                  } catch( Exception x){}
               }
           finaliza();
       }

   /** deactiva esta instancia dejando los nodos activos
     * para que en otra sesion una nueva instancia de Interfaz
     * se conecte a los nodos como nuevo visualizador de su estado
     */private void desliga(){
           repNodal.dispose();
           repNodal=null;
           casillas=null;
           System.exit(0);
       }

   /** desactiva cada simulador en los nodos e esta simulacion
     * y se desactiva a si
     */private void finaliza(){

           try{
                Enumeration os_nodos=nodos.elements();
                os_nodos=nodos.elements();
                while( os_nodos.hasMoreElements() )
                       ((NodoRemoto)os_nodos.nextElement()).end();
                desliga();
              }
             catch( Exception reee){
                                  System.out.println( "error al finalizar nodo \n"+reee.getMessage() );
                                  reee.printStackTrace();
                                }
       }

   /** pre: Un archivo nodos con las direcciones ip correspondientes
     * a cada nodo donde reside un Simulador.
     * Simulador instanciado, registrado y exportado en cada uno de esos nodos
     * Lee un archivo de propiedades ( {coordenada-IP} )
     * Obtiene referencias a instancias de NodoRemoto con las IP
     * Las almacena en un Hashtable contra sus coordenadas
     * post: Instancia de Interfaz creada a partir de la tabla
     */public static void main(String argv[]) throws Exception {
```

```
        Properties init_props = new Properties();
        init_props.load(new FileInputStream(argv[0]));

        String[] pd = argv[0].split("\\W");
        String params_dir ="/";
        for( int i=0; i<pd.length-1; i++ )
             params_dir = params_dir+pd[i]+"/";

        Properties nodos_places = new Properties();
        nodos_places.load(new FileInputStream( params_dir+"execs"));

        Enumeration keys = nodos_places.propertyNames();
        Hashtable nodales = new Hashtable();
        String this_nodo_nom = new String();
        NodoRemoto this_nodo = null;
        // contactando nodos vecinos de acuerdo a IPs en archivo nodos
        while( keys.hasMoreElements() ){
               this_nodo_nom = ((String)keys.nextElement()).trim();
               this_nodo = (NodoRemoto)Naming.lookup( "rmi://"
                                                      + nodos_places.getProperty(this_nodo_nom) + "/"
                                                      + this_nodo_nom
                                                    );
               nodales.put(this_nodo_nom,this_nodo);
               System.out.println("nodo "+this_nodo_nom+" @"+nodos_places.getProperty(this_nodo_nom));
             }

            // inicializa con parametros es init.props
           new Interfaz(      argv[1],
                              nodales,
                              Integer.parseInt( init_props.getProperty("nodos_rows", "1") ),
                              Integer.parseInt( init_props.getProperty("nodos_cols") ),
                              Integer.parseInt( init_props.getProperty("cell_rows") ),
                              Integer.parseInt( init_props.getProperty("cell_cols") ),
                              Float.parseFloat( init_props.getProperty("deposicion") ),
                              Float.parseFloat( init_props.getProperty("xi_fuzz") ),
                              Float.parseFloat( init_props.getProperty("evaporacion") ),
                              Integer.parseInt( init_props.getProperty("bichos_por_nodo") ),
                              Integer.parseInt( init_props.getProperty("modo") ),
                              Float.parseFloat( init_props.getProperty("xi_rate") ),
                                                    Float.parseFloat(
init_props.getProperty("xi_off") ),
                              Boolean.valueOf(  argv[2] ).booleanValue()
                             );

    }

}
package render;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;

class SitioVisual extends java.awt.Label {

    private double concentracion;

    private char c = ' ';

    static private double max;

    static private double evaporacion;

    static private double deposicion;

    static private boolean show_bichos;

/** Creates a new instance of SitioVisual */
    public SitioVisual() {
        setSize(new Dimension(1,1));
        setForeground(Color.red);
        setBackground(Color.lightGray);
    }

/** se asignan los valores globales a esta clase */
    static void setUp(final double dep, final double evapora ){
        deposicion=dep;
        evaporacion=evapora;
    }

/** representa la prescencia de un bicho en esta casilla */
    void ponla() {
```

```
         concentracion = concentracion+deposicion;
         c = '*';
         setText("" + c);
    }

/** despinta a un bicho que estaba aqui */
    void quitala() {
        c = ' ';
                setText("" + c);
    }

/** sustrae al valor de la concentracion por evaporacion */
    void evapora(){
        //setText("" + c);
        concentracion = concentracion*(1 - evaporacion);
        setBackground( Color.getHSBColor(1-(float)(concentracion/max),(float)(concentracion/max),1-
(float)(concentracion/max)));

    }

/** actualiza la concentracion */
    void setConcentracion(double c){
        concentracion = c;
    }

    double getConcentracion(){
        return concentracion;
    }

    static void setMx(double avg){
        max = avg;
    }
}

package render;

import java.rmi.*;

public interface VisualRemoto extends java.rmi.Remote {
    // public String about() throws RemoteException;

    /** metodo con el que se invocan los cambios en el display
     * se borra la hormiga en las preCoordenadas
     * y se incluye en las postCoordenadas con orientacion w
     * @param fromY cordenada vertical
     * @param fromX coordenada horizontal
     */
    public void x_sale( int fromY, int fromX )
                        throws java.rmi.RemoteException;

    /** atiende solicitudes de rendereo respecto a coordenadas especificas en donde
     * ingresa un bicho
     * @param toY coordenada vertical
     * @param toX coordenada horizontal
     */
    public void x_entra( int toY, int toX )
                        throws java.rmi.RemoteException;

    /** mecanismo para coordinar la representacion de la evaporacion (sincronizacion) */
    public void acknow( ) throws java.rmi.RemoteException;

}
```