

Pathfinding utilizando Algoritmos de Hormigas

Aplicado a laberintos 3D

Emmanuel Iarussi
Adrián Ignacio Pereyra
Director: Virginia Cifuentes

Facultad de Ciencias Exactas - UNICEN
Concurso de trabajos estudiantiles 2009
Categoría: Trabajo de Cátedra

15 de mayo de 2009

Resumen

Se propone un algoritmo evolutivo que resuelve un problema clásico como el de encontrar el camino más corto entre cada par de nodos de un grafo que modela a un extenso laberinto virtual tridimensional. Aplicando ideas de sencilla resolución e inteligencia distribuida, el algoritmo basado en hormigas resuelve el problema en grafos grandes, altamente densificados y también da la posibilidad de encontrar quasi-óptimos y/u óptimos absolutos en tiempos razonables para el tratamiento del problema. Además se propone una estrategia que reduce el número de actualizaciones necesarias para evitar caer en mínimos locales, mejorando la performance general del algoritmo. Los resultados obtenidos se han comparado con algoritmos clásicos, obteniendo que para grafos de hasta 10.000 nodos el algoritmo de hormigas reduce aproximadamente en un 95 % los tiempos de respuesta.

1. Introducción

A la hora de resolver problemas que involucran grafos, es ya conocida la utilidad que poseen algoritmos como el de Floyd para encontrar todos los caminos mínimos entre cualquier par de vértices, el de Kruskal, para el del árbol de recubrimiento mínimo o el algoritmo de Prim, para objetivos similares. Todos estos algoritmos, aunque eficientes, tienen un costo polinómico que a pesar de ser relativamente bajo, escalan considerablemente para cantidades (ya sea de arcos o de vértices) muy elevadas. En este contexto surge la necesidad de encontrar caminos heurísticos a las soluciones que no aseguran el óptimo pero incorporan una serie de técnicas que nos acercan a ellos en la mayoría de los casos.

Hace aproximadamente 15 años, un científico Belga descubrió la capacidad extraordinaria que tenían ciertos insectos, las hormigas, para resolver problemas de forma colectiva e implicando un mínimo de inteligencia por parte de cada uno de sus miembros. Esta fue la fuente de inspiración que lo llevaría a crear una nueva generación de algoritmos: los algoritmos de Hormigas.

Nuevamente, a pesar de ser algoritmos altamente eficientes, para cantidades considerablemente grandes de nodos se vuelven muy problemáticos. Además, es imposible asegurar su convergencia a una solución óptima, por lo cual es de vital importancia contar con métodos que nos den cierta certeza a la hora de evaluar la optimalidad de las respuestas. En el presente trabajo intentaremos dar solución a los mencionados problemas a fin de conseguir mejores resultados en los algoritmos de hormigas aplicados a problemas de ruteo.

Históricamente, los laberintos resultaron ser un clásico para la aplicación de métodos de pathfinding y combinados con la graficación 3D se vuelven un desafío muy interesante. Explorar las posibilidades que da la creación de ambientes tridimensionales a través de herramientas como OpenGL es una experiencia enriquecedora que facilita la visualización de resultados.

Este trabajo integra los aspectos antes mencionados en una aplicación que resuelve caminos mínimos con algoritmos de hormigas, optimizando al máximo los resultados para grafos de gran tamaño que modelan complejos laberintos tridimensionales.

2. Modelado del problema

Las hormigas, al igual que todos los seres vivos con comportamiento social, se valen de esta característica para realizar tareas que les serían inviables si las intentaran individualmente. Estos particulares insectos se agrupan en grandes conglomerados y crean complejas estructuras de organización a partir de simples reglas y comportamientos. Tan exitosas son sus estructuras sociales que les han permitido sobrevivir en la tierra por más de 120 millones de años convirtiendo a las hormigas en una de las especies vivas más antiguas del planeta.

Ciertos comportamientos pueden ser modelados en agentes inteligentes, para que en conjunto puedan resolver complejos problemas computacionales. Para esto, es necesario modelar los agentes involucrados (hormigas) y algún ambiente que los contenga y que sirva de medio para que puedan coordinar y comunicarse entre sí. Carentes de una buena visión y de oídos, y en constante contacto con la superficie, estos insectos han desarrollado un eficiente sistema de mensajes químicos a través de feromonas. En general, este método de comunicación indirecta a través de mensajes depositados en el ambiente se denomina: “stigmergy”. Así, si una hormiga encuentra alimento, vuelve a la colonia depositando feromonas provocando con esto que otras la sigan. El concepto de “stigmergy” fue introducido por Grassé, un paleontólogo francés en 1959.

La mayoría de las especies de hormigas tienen un comportamiento muy simple a la hora de llevar a cabo la tarea de encontrar alimento. Por ejemplo, la Linepithema humile, popularmente conocida como hormiga argentina se limita únicamente a comunicarse a través de feromonas. Las hormigas de esta especie en busca del alimento van y vienen a la fuente depositando feromonas. Las otras hormigas las siguen con una determinada probabilidad, que además dependerá de la cantidad de feromonas en el camino.

Desde que se descubrió la forma en que las hormigas hallaban el camino, una multiplicidad de campos de la ciencia se dedicó a estudiarlas. En 1990, el Dr. Jean Louis Deneubourg diseñó un ex-

perimento para observar a una colonia encontrando su alimento. En dicho experimento, denominado “de los dos puentes” colocó una colonia de hormigas en un extremo y en el otro el alimento. Ambos sitios fueron conectados primeramente por dos puentes de igual longitud e igual ángulo de bifurcación. Seguidamente habilitaron a las hormigas a desplazarse libremente por el camino. Al cabo de cierto tiempo, se tomaron muestras de la cantidad de estos insectos en cada ramal del puente, y resultó que estaban equitativamente distribuidas.

Luego, conectaron al hormiguero un nuevo tipo de puente, esta vez contenía una rama considerablemente más larga que la otra. Nuevamente se liberó a las hormigas y al cabo de cierto tiempo se observó que todas transitaban por el camino más corto al alimento. Además, esporádicamente algunas de las hormigas rompían la fila y exploraban nuevos caminos.

Como ya se mencionó con anterioridad, cuando una hormiga explora un camino lo hace depositando en él sustancias químicas que ayudan a las otras de su especie a seguirlo. Cuando una halla una fuente de alimento y emprende el regreso, duplica la cantidad de feromonas en el camino aumentando las posibilidades de que otras las sigan. Las hormigas que eligieron el camino corto aventajan a las que optaron por el largo, y para un mismo instante habrá más feromonas en el camino corto que en el largo. Este proceso se repite en el tiempo, aumentando la concentración de químicos en un camino y reduciendo así las posibilidades de que el otro sea elegido.

Finalmente un tercer experimento fue realizado. En este, solo un camino fue colocado entre el nido y el alimento. Se dejó a las hormigas recorrerlo, y cuando se obtuvo un flujo constante que iba y venía, se anexó un segundo puente mucho más corto. Al principio, solo unas pocas hormigas lo eligieron por mero azar, pero paulatinamente el número se incrementó hasta que en cierto punto el camino original fue abandonado por completo. Este fenómeno se dio básicamente por combinación de dos factores: por un lado, las hormigas que volvieron por el camino más corto incrementaron más rápidamente el número de feromonas en el camino, aumentando así la posibilidad de que otras hormigas también lo elijan. Por otro lado, gracias a los efectos del ambiente, a medida que más hormigas transitaban el nuevo recorrido, la concentración de feromonas en el camino original disminuyó, haciendo que la nueva opción sea cada vez más elegida.

Es muy importante destacar que este fenómeno nunca hubiese sucedido de no ser por la siempre presente posibilidad (aunque a veces muy pequeña) de que una hormiga transite un camino con baja concentración de feromonas. Además, la evaporación de estos químicos favorece la exploración de nuevos recorridos, que de otra forma quedarían confinados, haciendo que las hormigas no se conformen con un camino óptimo local.

Deneubourg, en su intentó por capturar de alguna manera la esencia del comportamiento de las hormigas, desarrolló un modelo estocástico para el experimento de los dos puentes. En él, definió los siguientes parámetros:

- φ : Es la cantidad de hormigas que cruzan el puente a una velocidad v m/seg, y depositan en él una única unidad de feromona. Es importante aclarar que en este modelo no es considerada la evaporación de estos químicos. Es decir, es necesario que el tiempo de vida de las feromonas sea el mismo que demora la simulación en converger al camino más corto.
- t_s : Tiempo en el que una hormiga atraviesa la rama corta. Este valor está dado por el cociente l_s/v , siendo l_s la longitud del camino más corto.
- $p_{is}(t)$: Probabilidad de que una hormiga elija la rama corta. Además, como $p_{il}(t)$ es la posibilidad de elegir el camino largo, entonces será:

$$p_{is}(t) + p_{il}(t) = 1$$

- α :Este parámetro fue deducido de numerosos experimentos y en general vale 2. Depende, entre otros factores, de las hormigas que se utilicen para el experimento.

Así, la probabilidad p_{is} (de elegir la rama más corta) se define como:

$$p_{is} = \frac{(t_s + \varphi_{is}(t))^\alpha}{(t_s + \varphi_{is}(t))^\alpha + (t_s + \varphi_{il}(t))^\alpha}$$

Esta ecuación se combinó con otras dos para terminar de modelar el comportamiento de las hormigas. La simulación Monte Carlo realizada por Marco Dorigo [Dor04] de las ecuaciones para 1000 tiradas arrojó resultados sorprendentemente similares a los comportamientos observados por el doctor Deneubourg. Las hormigas encontraron siempre, en distintas superficies, el camino más corto hacia el alimento, inclusive si en su ambiente había cambios luego de que iniciasen la exploración.

3. Aplicación

A la hora de construir hormigas artificiales se pensó básicamente en el experimento de los dos puentes y el comportamiento que se debía esperar de ella en esta situación, ya que el problema del camino más corto, en grafos aún mayores puede desglosarse en una cantidad finita de decisiones del mismo tipo que las del mencionado experimento.

El primer elemento a definir fue el modelado de la superficie. Era preciso establecer una estructura que modelara los dos puentes y potencialmente terrenos mayores, y que además pudieran recorrer las hormigas. Un grafo no dirigido fue la solución más apropiada ya que vinculaba ambos aspectos y resultaba sencilla su implementación. De todas formas, se intentó simplificar más aún su estructura debido a un particular problema con los arcos.

Supongamos que debemos modelar un terreno en el que se encuentran tres posibles caminos para ir del nido al alimento, dos de ellos con costo 3 y uno con costo 1. La forma normal de modelar esto con un grafo es colocando dos nodos (uno para origen y otro como destino) y tres arcos entre ellos, con los correspondientes costos. Ahora bien, una hormiga que transita el camino deberá optar por uno de los tres. Dado que solo puede hacer un paso por turno, si la hormiga elige alguno de los arcos con costo 3, deberá permanecer en el camino por tres turnos hasta alcanzar un nodo. Por este motivo, para simplificar la tarea de la hormiga, en el grafo se prohíben arcos con costo mayor a 1, y se construye un camino equivalente. Así, en cada paso la hormiga alcanza un nuevo nodo.

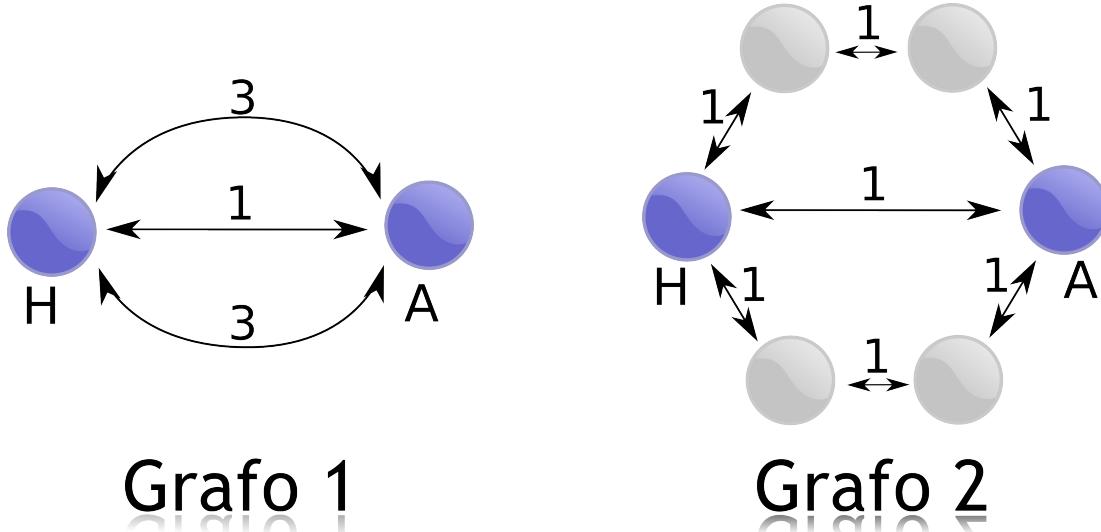


Figura 1: Antes y después del proceso de separación de arcos en nodos

Una vez definido el terreno, es de vital importancia crear la forma en que las hormigas decidirán el camino a seguir. Para esto, basándonos en los experimentos del doctor Deneubourg y de Dorigo [Dor00] recreamos la siguiente ecuación que define el valor de probabilidad de que una hormiga transite el camino ab :

$$P_{ab} = \frac{C_{ab}}{\sum_{i=0}^n C_{ai}}$$

en donde C_{ab} es el costo de ir de a hasta b y C_{ai} son los costos de los vértices vecinos de a .

Como dijimos en secciones anteriores las hormigas depositan sus feromonas tanto a la ida como en el regreso, sin importar cuan largo sea el camino ni si hallaron el alimento. Una hormiga podría perderse irremediablemente y guiar a otras con ella. Aún más peligroso, es el caso de un ciclo: si una hormiga entra en él, comienza a depositar sus feromonas y lo vuelve cada vez más atractivo para las demás, que también comenzarán a girar en un bucle en el que al fin y al cabo todo el sistema quedará atrapado (liveloop). Este caso no es poco común y es irrecuperable. Sin embargo, hay una forma muy sencilla de solucionarlo. En vez de depositar las feromonas durante todo el trayecto, es decir, ida y vuelta, solo se depositará cuando se regrese de la fuente de alimento. Esto impide que otras hormigas queden atrapadas en ciclos, ya que estos pueden ser eliminados con facilidad cuando una hormiga regresa al hormiguero, y también evita que un conjunto de hormigas que pierde el rumbo, guíe por mal camino al resto. A pesar de que parece una solución brillante, tiene un error fatal. El sistema entero no funciona si las feromonas son depositadas solo de regreso. Por ello, hay que agregar una serie de mecanismos que ayudan a la convergencia de las hormigas al camino más corto.

Se analizaron entonces diferentes alternativas a la hora de establecer pautas para el stigmergy. Depositar siempre una unidad de feromonas en el arco resultó no ser lo suficientemente justo. Ciertos caminos (los más cortos) merecen tener más chance de ser elegidos, aunque tampoco podemos privar a las hormigas de explorar nuevas rutas. Por ello, la cantidad de feromonas que se depositan son directamente proporcionales a la longitud del camino. Así, la cantidad de feromonas a depositar en el arco responde a la fórmula:

$$F_{ab} = \frac{1}{pathlong}$$

Además, se incorporó un sistema de evaporación de feromonas. En cada ciclo, el nivel de feromonas de todo el grafo es actualizado, en base a un factor. En fórmulas, por cada par de arcos ab en el grafo, la nueva cantidad de feromonas F' se calcula como:

$$F'_{ab} = (1 - P) * F_{ab}$$

en donde F es la cantidad original de feromonas en el grafo, y P es el factor de evaporación.

El proceso de evaporación de feromonas impulsa a las hormigas a buscar otros caminos, y a no conformarse con mínimos locales. Experimentamos con varios factores de evaporación, y el que mejora considerablemente la convergencia al camino más corto, en tiempos razonables es $P = 0,01$.

Finalmente fue preciso crear una entidad que encapsule todo el comportamiento que se espera obtener de una hormiga y todos los datos que debe almacenar: terreno, nodo origen, nodo destino, camino recorrido y posición actual. Una hormiga puede verse entonces como una máquina de dos estados: "Forward mode" y "Backward mode".

Cuando una hormiga es creada, es seteada a modo *avanzar*. De esta forma, cuando se desea que la hormiga avance por el camino, ella eligirá entre sus posibilidades siguiendo los cálculos mencionados anteriormente. Una vez que la hormiga alcanza el destino, su modo cambia a *retroceder*, haciendo que cada vez que debe elegir el próximo nodo hacia donde ir, lo haga recorriendo su lista de camino recorrido, para así depositar la feromona en los mismos lugares por los que transito a la ida.

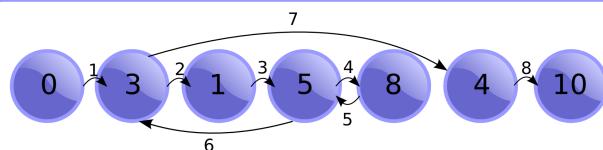
3.1. Eliminación de ciclos

Como habíamos mencionado, una vez que se alcanza el nodo destino, la ruta almacenada por la hormiga debe ser procesada para eliminar cualquier vestigio de ciclos ya que de no hacerlo, podríamos guiar a las hormigas a bucles que se autoalimentan de feromonas, volviéndose cada vez más atractivos.

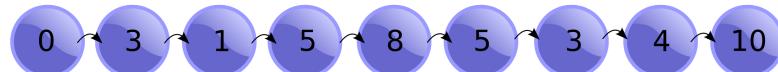
Para poder eliminar los ciclos, se implementó un algoritmo que se ejecuta cada vez que una hormiga llega a destino, antes de emprender el regreso. Dicho algoritmo posee un comportamiento muy sencillo. Se recorre la lista de nodos visitados de izquierda a derecha, y por cada uno, se vuelve a

Algoritmo de eliminación de ciclos

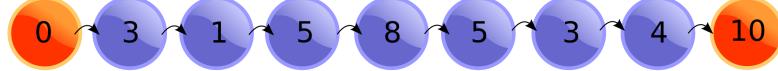
1.



2.



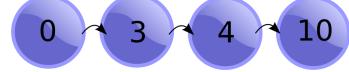
3.



4.



5.



6.



Figura 2: Secuencia del algoritmo de eliminación de ciclos para un camino

recorrer la lista de derecha a izquierda intentando encontrar nodos repetidos. Cuando son encontrados dos elementos iguales estamos en presencia de un ciclo, con lo cual el camino comprendido entre ellos es completamente prescindible y se elimina.

3.2. Algoritmo

Una vez definidas las características de una hormiga, es preciso definir un algoritmo que utilice todo su potencial. Básicamente, el algoritmo se basa en los siguientes componentes: Elección de un camino en cada nodo en el que se encuentre una hormiga, avance de todas las hormigas por el camino elegido, actualización de la cantidad de feromonas y evaporación general de los “químicos” en los arcos.

Para poder ejecutar el algoritmo debemos definir dos parámetros importantes: el tiempo de ejecución y la cantidad de hormigas. Estos parámetros dependen del terreno por el que las hormigas transitarán. Por ejemplo, si la cantidad de nodos en el grafo es de 100, con 20 hormigas bastará para hallar rápidamente el camino más corto entre cualquier par de nodos; ahora bien, si el terreno tiene 100.000 nodos, 20 hormigas podrían perderse en la enorme cantidad de bifurcaciones.

El algoritmo devuelve el camino más corto hallado hasta el momento de corte. Claramente, al ser un algoritmo heurístico, no podemos asegurar completamente que el resultado obtenido sea realmente el camino óptimo. De todas formas, si el tiempo de ejecución es lo suficientemente largo, aumentan considerablemente las posibilidades de hallarlo. Por esta razón determinar el tiempo de ejecución para un grafo es de vital importancia en el hallazgo de buenas soluciones.

3.2.1. Pseudocódigo

```

antPathFinding (Terrain, Source, Food)
{
    antsVector[NUM_ANT];
    initialize(antsVector,Terrain,Source,Food); //Hormigas a modo "forward"

    while(not converge() or not execLimitReached()) //Comienzo del algoritmo
    {
        for each ant
        {
            if destinyReached(ant_i)
            {
                dropLoops(ant_i);
                setToBackwardMode(ant_i);
                savePath(ant_i);
            }
            if(isBackwardMode(ant_i) and sourceReached(ant_i))
            {
                ant_i=new Ant; //Reemplazo por una nueva hormiga
                initialize(ant_i,Terrain,Source,Food);
            }
            chooseNextPlace(ant_i); //Avanzar un paso
        }

        if(timeToEvaporate())
            for each edge
                pheromonesUpdate(edge_i);
    }
}

```

3.3. Optimizaciones

Con motivo de perfeccionar aún más el algoritmo, y de aumentar su performance tanto como fuera posible, se pensaron algunas mejoras que resultaron fundamentales para la calidad de la solución final.

Cuando una hormiga halla un camino, no evita que otras encuentren una ruta aún más larga. Es decir, no contábamos con un mecanismo que permita a una hormiga “saltarse” el Stigmergy y comunicarse a través de otros medios para informarse el camino más corto encontrado hasta el momento. Agregando esta funcionalidad, es decir, permitiendo a todas las hormigas conocer la ruta más corta que se ha hallado, se puede evitar que salgan en busca de caminos que ya no serán solución. De esta manera, una hormiga cuya ruta supera la longitud mínima permitida hasta el momento, es regresada al origen, para que busque una nueva solución. La reducción del espacio de búsqueda es notable a simple vista.

Se ha observado experimentalmente que una vez encontrado un camino, el algoritmo se acelera vertiginosamente debido al estrechamiento del avance permitido, haciendo que a esta primer solución le sigan otras aún mejores casi inmediatamente.

Haciendo numerosas pruebas con diferentes tamaños de terrenos, notamos un decremento importante en el rendimiento del algoritmo en grafos extremadamente grandes debido únicamente al costo excesivo de la evaporación de feromonas. Recorrer en cada iteración todos los arcos existentes en el grafo retrasaba intensamente el proceso de avance de las hormigas. Por otro lado, nos era imposible eliminar la evaporación ya que sin ella es altamente improbable que el algoritmo halle la solución óptima, y en general no converge. Por este motivo se adoptó una solución intermedia que debió ser adaptada para alcanzar la convergencia. Hasta que no se encuentra al menos un camino hacia el alimento, la evaporación se realiza cada N iteraciones (N depende pura y exclusivamente del número de arcos en el grafo). Adicionalmente, más feromona es quitada en cada paso, debido a la demora entre los períodos de evaporación. Es importante notar que con este cambio no alteramos la posibilidad de convergencia del algoritmo. Una vez hallada alguna solución parcial, es decir, habiendo reducido el espacio de búsqueda, se quitan feromonas de los arcos cada $l/2$ ciclos, siendo l la longitud del camino más corto hallado hasta el momento. Este técnica de “avance y evaporación” mejora extremadamente el rendimiento del algoritmo permitiendo encontrar soluciones en grafos de 100.000 nodos en menos de 4 minutos, sin poner en riesgo la convergencia.

4. Experimentos y Resultados

Con motivo de comparar el rendimiento de nuestro algoritmo de hormigas con otro ya conocido, se implementó el algoritmo de Dijkstra para encontrar el camino más corto, y los resultados fueron los siguientes:

Los grafos de estas pruebas son generados aleatoriamente, n es la cantidad de nodos, y la cantidad de arcos por nodo puede variar entre 1 y 4. La cantidad de hormigas en el grafo es 300. En todos los casos ambos algoritmos hallaron la solución óptima.

Como se puede observar, el costo del algoritmo de Dijkstra es similar al del algoritmo de hormigas para grafos relativamente pequeños (100 nodos). Pero mientras el costo del primero sigue creciendo según una curva de orden cuadrático, el algoritmo implementado en este trabajo mantiene el costo temporal muy bajo.

Para la evaluación del rendimiento del algoritmo, se realizaron 100 ejecuciones del mismo, para grafos con 1000, 2000 y 3000 nodos. Dichos grafos son generados aleatoriamente y como representan laberintos, poseen 1, 2, 3 y hasta 4 nodos adyacentes. Esta cantidad es también aleatoria. Es importante remarcar que estos test se realizaron en una PC con un procesador AMD - Athlon x 2 de 1.9 GHz y 1Gb de Memoria Ram DDR2 667 MHz.

En la primer tabla (figura 4) podemos ver el promedio de cantidad de arcos para los diferentes tamaños del terreno. Por otro lado, el promedio de iteraciones hasta alcanzar la mejor solución al camino mínimo, y un tiempo aproximado en milisegundos (en segundos: 1, 4 y 53 respectivamente).

Un aspecto importante del algoritmo, es que el tiempo de ejecución, es decir la cantidad de itera-

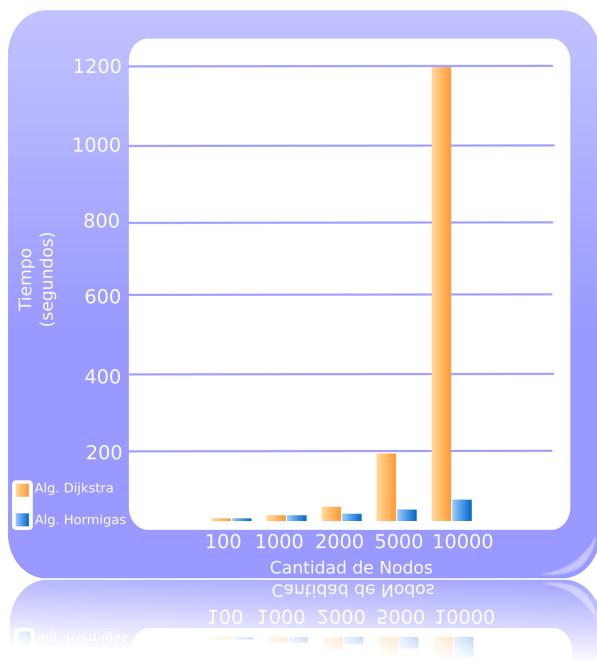


Figura 3: Gráfico comparativo de tiempos de ambos algoritmos

Nodos	Arcos	Iteraciones	Tiempo
1000	4985,63	138,92	987,04
2000	9987,49	400,65	4822,93
5000	24954,61	1892,56	53168,12

Figura 4: Tablas correspondientes a las pruebas de performance del algoritmo. Los datos aquí presentes están calculados sobre un total de 100 ejecuciones del mismo

ciones depende pura y exclusivamente de lo que el algoritmo demore en converger a un único camino. A la hora de cargar un grafo del cual se quiere hallar el camino mínimo entre dos nodos, se debe estimar en base a varios parámetros (cantidad de arcos, de hormigas, factor de evaporación, feromonas depositadas en cada paso) cuanto tiempo deberá esperarse hasta obtener la solución óptima. Por este motivo, realizamos pruebas con los siguientes parámetros del algoritmo para obtener dichas medidas estimativas.

- Cantidad de Hormigas = 300
- Factor de Evaporación = 0,01
- Feromonas en cada paso = $1/pathlong$

Percentil	Iteraciones	Tiempo Aprox. (milisegundos)	Percentil	Iteraciones	Tiempo Aprox. (milisegundos)	Percentil	Iteraciones	Tiempo Aprox. (milisegundos)
1000 nodos	0,5	94	2000 nodos	0,5	247,5	0,5	731	16000
	0,7	141,2		0,7	411,1	0,7	1084,5	38000
	0,9	250,8		0,9	779	0,9	2046,3	67000
	0,99	794,33		0,99	2881,16	0,99	24762,29	750000

Figura 5: Tablas estadísticas para distintos tamaños de grafos

En la tabla 1 de la figura 5 se observa que para poder tener 50 porciento de probabilidades de obtener el óptimo, el algoritmo debe iterar aproximadamente 94 veces. Si queremos la mejor solución con un 99 porciento de certeza, debemos iterar cerca de 800 veces. Las medidas temporales pueden variar dependiendo del sistema y la arquitectura de la máquina en la que se está corriendo el algoritmo. En general, estos tiempos están entre 1 y 6 segundos.

Como podíamos suponer, para tamaños y cantidades de nodos aún mayores la cantidad de iteraciones necesarias para seguir obteniendo soluciones óptimas se incrementa. En este caso, para un 99 porciento de probabilidades de que la solución encontrada sea la óptima, se requieren, aproximadamente 2800 iteraciones, aproximadamente un minuto de ejecución.

Finalmente, en grafos aún mayores, se requieren casi 25000 iteraciones para soluciones sumamente confiables. Aún así, los tiempos son considerablemente menores a los de los algoritmos clásicos utilizados normalmente.

5. Laberintos 3D

Junto con la investigación de técnicas que mejoren la performance de los algoritmos de hormigas aplicados a problemas de ruteo, se pensó de qué manera lograr una aplicación de dichas técnicas de forma que se facilite el ingreso de datos y la devolución del algoritmo. Por esta razón se construyó paralelamente a la investigación, una herramienta capaz de hacer simulación de recorridos en laberintos tridimensionales. De esta forma, podemos no solo resolver los laberintos, sino también visualizar los resultados y recorrerlo virtualmente.

Para la realización de dicha aplicación se utilizó puramente OpenGL y la librería GLUT (OpenGL Utility Toolkit). Esta librería nos permitió hacer el manejo de los gráficos 3D y de los dispositivos de entrada/salida. El factor principal que fundamentó nuestra elección es el soporte multiplataforma de GLUT, lo que nos permite trabajar en múltiples SSOO de forma sencilla.

Los laberintos son generados aleatoriamente de manera que se define una superficie (rectangular) de base y sobre ella se construyen paredes, quedando definido un laberinto con una única entrada y



Figura 6: Screenshot de la animación en el laberinto “selva”, el personaje que lo recorre en este caso: una hormiga

una única salida. Cabe destacar que se aplicaron algoritmos de construcción de polígonos mínimos para minimizar los recursos que insume la graficación y animación 3D.

La ambientación de los laberintos virtuales se realiza por medio de la utilización de luces y texturas, complementados con objetos tridimensionales que sirven como obstáculos. Todo esto es modelado en un conjunto de estructuras que contienen todos los elementos presentes en una escena.

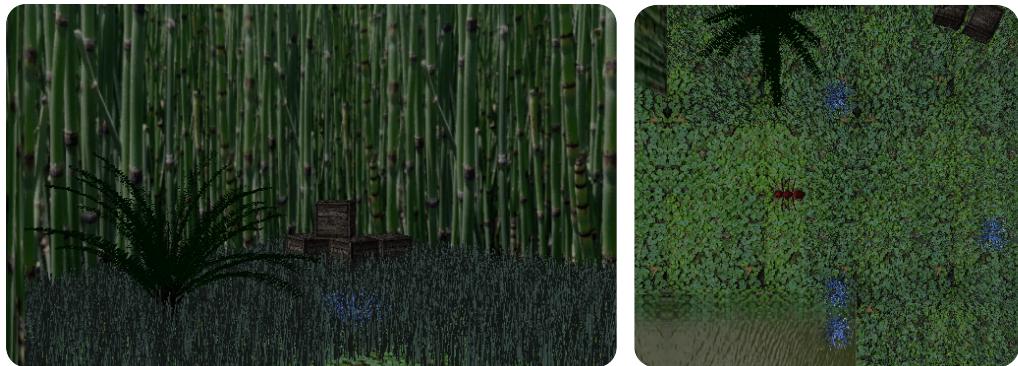


Figura 7: Screenshot de la animación en el laberinto y vista superior

6. Conclusiones

La implementación de este algoritmo de Inteligencia Artificial conocido como “Algoritmo de Hormigas” ha demostrado ser una excelente alternativa de solución a problemas complejos con altos costos computacionales. Particularmente, estos algoritmos son un terreno ampliamente explorado y existen multiplicidad de aplicaciones que los utilizan como base. Claramente hemos demostrado que a pesar de su gran difusión todavía pueden desarrollarse métodos que los optimizan al extremo consiguiendo así resultados cada vez más ajustados en tiempos razonables.

Por otro lado, las nuevas herramientas para la graficación computacional nos han permitido ofrecer los resultados de nuestro trabajo en forma sencilla y clara, haciéndolo además, mucho más atractivo.

Es sumamente importante rescatar el valor de este tipo de trabajos que instan a los alumnos a investigar y desarrollar algoritmos que ayudan a ampliar los contenidos aprendidos en la cátedra.

7. Referencias

[Bar02] Lucía Barcos, Victori M. Rodríguez y Jesús Álvarez (2002). *Algoritmo basado en la optimización mediante colonias de hormigas para la resolución del problema del transporte de carga desde varios orígenes a varios destinos*. V Congreso de ingeniería del transporte, España.

[Dor04] Marco Dorigo y Thomas Stützle (2004). *Ant Colony Optimization*. Massachusetts Institute of Technology (MIT).

[Dor00] Marco Dorigo, Eric Bonabeau y Guy Theraulaz (2000). *Ants algorithms and stigmergy*. Université Paul Sabatier, Toulouse, France.

[ShrWo] Dave Shreiner, Mason Woo, Jackie Neider, Tom Davis. *OpenGL programming guide*.

Material de la cátedra: Visualización computacional de datos de la Facultad de Ciencias Exactas (UNICEN). <http://www.exa.unicen.edu.ar/catedras/viscomp/index.html>.

Material de la cátedra: Visualización computacional II de la Facultad de Ciencias Exactas (UNICEN). <http://www.exa.unicen.edu.ar/catedras/visdat2/index.html>.