Escuela de Computación e Informática
Paradigmas Computacionales CI-1440 Grupo 2
Profesora Yadyra Solano Sabatier

Algoritmos Genéticos
Piglets: Simulación de Supervivencia de Individuos

Francisco Rodríguez Zamora
A54818

09/6/2009

Índice de contenido

Introducción

Gracias a los postulados evolucionistas propuestos por Darwin se cuenta hoy en día con uno de los métodos de optimización más poderosos: los algoritmos genéticos. El estudio de los fenómenos naturales dan gran fundamento a su teoría, fenómenos como por ejemplo el de la resistencia a los insecticidas en las plagas de cultivos, a los antibióticos en las bacterias, a la quimioterapia en las células cancerosas, y a los fármacos antiretrovirales en virus como el VIH- es una consecuencia abierta de las leyes de la mutación y la selección, y comprender estos principios nos ha ayudado a desarrollar estrategias para enfrentarnos a estos nocivos organismos.

El proyecto es una forma simple de estudiar los algoritmos genéticos, utilizando un ambiente muy parecido a un juego de video. El propósito es tener una población de animales (cerditos) los cuales tienen que luchar por conseguir comida, al final del tiempo de cada generación se escogen los individuos que más comida hayan conseguido y se cruzan para conseguir la siguiente generación. El usuario puede intervenir poniendo obstáculos y comida.

La principal utilidad que tiene es el de analizar el comportamiento de los cerditos a lo largo del tiempo, al mismo tiempo que el usuario puede intervenir para cambiar el proceso aleatorio de generación de recursos, así como obtaculizar a los cerditos a la hora de buscar comida.Es una aplicación muy simple que sirve para observar el proceso evolutivo en un acervo de especímenes.

Objetivos

General

- Simular un “sistema” en el cual diversos individuos luchan por sobrevivir y perpetuar sus genes.

Específico

- Desarrollar un ecosistema en el cual varios individuos puedan habitar y buscar comida.
- Desarrollar un algoritmo genético que cruze a los individuos más aptos y genere descendencia.
- Analizar el desarrollo de los individuos en el tiempo.

Marco Teórico

Descripción del paradigma

Un algoritmo genético es una forma de programación en la cual se imita a la evolución genética para resolver problemas de búsqueda y optimización.

Consiste en una función matemática o una rutina de software que toma como entradas a los ejemplares y retorna como salidas cuales de ellos deben generar descendencia para la nueva generación. A lo largo de las generaciones, las poblaciones evolucionan en la naturaleza de acorde con los principios de la selección natural y la supervivencia de los más fuertes, postulados por Darwin.

Se usa una analogía directa con el comportamiento natural. Se trabaja con una población de individuos, cada uno de los cuales representa una solución factible a un problema dado. A cada individuo se le asigna un valor ó puntuación, relacionado con la bondad de dicha solución.

Teniendo un conjunto de individuos, se les pone a interactuar con el ambiente, luego cada individuo se evalúa de acuerdo a una función de aptitud definida previamente. En un acervo de candidatos, por supuesto, la mayoría no funcionarán en absoluto, y serán eliminados. Estos candidatos prometedores se conservan y se les permite reproducirse y se introducen cambios aleatorios durante el proceso de copia a manera de mutación.

Luego, esta descendencia digital prosigue con la siguiente generación, formando un nuevo acervo de individuos, y son sometidos a una ronda de evaluación de aptitud. De nuevo, se selecionan y copian estos individuos vencedores hacia la siguiente generación con cambios aleatorios, y el proceso se repite. Las expectativas son que la aptitud media de la población se incrementará en cada ronda y, por tanto, repitiendo este proceso cientos o miles de rondas, pueden descubrirse soluciones muy buenas del problema.

Los algoritmos genéticos requieren que el conjunto se codifique en un cromosoma. Cada cromosoma tiene varios genes, que corresponden a sendos parámetros del problema. Para poder trabajar con estos genes en el ordenador, es necesario codificarlos en una cadena, es decir, una ristra de símbolos (números o letras) que generalmente va a estar compuesta de 0s y 1s.

Para comenzar la competición, se generan aleatoriamente una serie de cromosomas. El algoritmo genético procede de la forma siguiente:

- Evaluar la puntuación (fitness) de cada uno de los genes.
- Permitir a cada uno de los individuos reproducirse, de acuerdo con su puntuación.
- Emparejar los individuos de la nueva población, haciendo que intercambien material genético, y que alguno de los bits de un gen se vea alterado debido a una mutación espontánea.

Cada uno de los pasos consiste en una actuación sobre las cadenas de bits, es decir, la aplicación de un operador a una cadena binaria. Se les denominan, por razones obvias, operadores genéticos, y hay tres principales: selección, crossover o recombinación y mutación; aparte de otros operadores genéticos no tan comunes, todos ellos se verán a continuación.

Un algoritmo genético tiene también una serie de parámetros que se tienen que fijar para cada ejecución, como los siguientes:

- Tamaño de la población: debe de ser suficiente para garantizar la diversidad de las soluciones, y, además, tiene que crecer más o menos con el número de bits del cromosoma, aunque nadie ha aclarado cómo tiene que hacerlo. Por supuesto, depende también del ordenador en el que se esté ejecutando.
- Condición de terminación: lo más habitual es que la condición de terminación sea la convergencia del algoritmo genético o un número prefijado de generaciones.

Una de sus características principales de los algoritmos genéticos es la de ir perfeccionando su propia heurística en el proceso de ejecución, por lo que no requiere largos períodos de entrenamiento especializado por parte del ser humano, principal defecto de otros métodos para solucionar problemas, como los Sistemas Expertos.

Descripción del aplicación

El proyecto es una forma simple de estudiar los algoritmos genéticos, utilizando un ambiente muy parecido a un juego de video. El propósito es tener una población de chanchos los cuales tienen que luchar por conseguir comida, al final del tiempo de cada generación se escogen los individuos que más comida hayan conseguido y se cruzan para conseguir la siguiente generación. El usuario puede intervenir poniendo obstáculos y comida.

Manual de Usuario

Plataforma

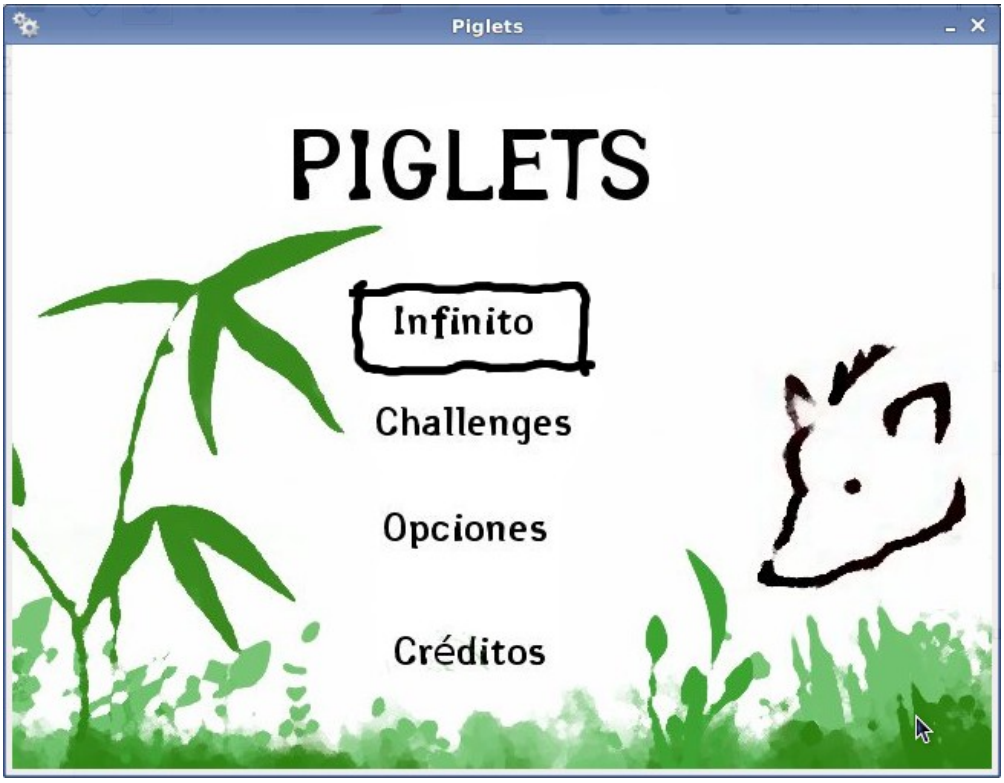
Se usó el lenguaje de programación C para el desarrollo de la aplicación, por lo que se le dió un acercamiento estructurado al problema. La implementación de la aplicación es bastante directa, ya que el algoritmo que se va a usar es realmente simple, y el paradigma estructurado facilita las cosas.

Para el despliegue de imágenes y sonido se utilizará la biblioteca Simple DirectMedia Library (SDL) que provee gran cantidad de funciones para el manejo de multimedia.

El para ejecutar solo es necesario hacer doble click en el archivo ejecutable.

Aplicación

La aplicación consta de una interfaz fácil de usar. La pantalla principal del programa es la siguiente:



Al hacer click con la opción “Infinito” empezará el juego sin límite de tiempo.
La pantalla de juego deberá ser similar a la siguiente:

EL juego provee la siguiente información:

Número de Generaciones

Cronómetro

Número de generación en la que nació el chancho

Ya en el juego se pueden alterar las formas de juego de varias maneras diferentes.

Haciendo click derecho sobre cualquier lugar por donde andan los chanchos hará que aparezca comida. Al encontrar la comida, los chanchos correrán hacia ella.

Al hacer click izquierdo aparecerá un tronco de madera, el cual impedirá el paso a cualquier chancho.

Si se presiona la tecla 'r', la comida aparecerá aleatoreamente sobre la superficie.

Detalles técnicos

La aplicación consta de tres archivos de código fuente:

- main.c : Contiene la función principal del programa, inicializa todos los factores necesarios
- imagenes_sdl.c: Contiene las funciones principales para manejar las imágenes del programa, así como inicializar la biblioteca SDL.
- game.c: Funciones del juego. Funciones del algoritmo genético.

Los individuos tiene los siguientes genes:

```
struct _piglet {
  int _x;//posición x del chancho
  int _y;//posición y del chancho
   int _inteligencia;//inteligencia del chancho aka cantidad de
veces que cambia de dirección
  int _vel_run;//velocidad del chancho cuando corre
  int _vel_walk;//velocidad del chancho cuando camina
  int _stamina;//stamina: cantidad de tiempo que puede correr
  int _vision;//rango de visión del chancho
  int _direccion; //dirección en la que va el piglet
  char id[3];//identificador del chancho
  int tipo;//tipo de chancho al pintar :p
  int _eaten;//para saber cuanto se ha comido
} _piglet;
```

Se crean 10 chanchos con los genes al azar. Estos se dejan caminar por el mapa durante 15 segundos. Al terminar estos 15 segundos se ejecutará la función de aptitud, la cual seleccionará a los chanchos que más hayan logrado comer durante la generación.

game.c Línea 985: `void select_piglets(void *data) // Función de Selección`

El cruce de los chanchos para crear nuevos individuos se hace eligiendo cada gen al azar entre alguno de los dos padres.

game.c Línea 1050: `void create_piglet( const struct _piglet *parent1, const struct _piglet *parent2, struct _piglet *child, const unsigned int i ) //función de cruze`

Luego de crear los nuevos chanchos se puede pasar a mutar. Esto se decide al azar y tiene 1/3 de probabilidades de ocurrir. Los chanchos que se mutarán serán entre 1 y 4. La mutación cambiará 1 gen al azar de cada chancho.

Game.c Línea 1073: `void mutate(const unsigned n)//función de mutación`

Evaluación

Ventajas de los Algoritmos genéticos[1]

El primer y más importante punto es que los algoritmos genéticos son intrínsecamente paralelos. La mayoría de los otros algoritmos son en serie y sólo pueden explorar el espacio de soluciones hacia una solución en una dirección al mismo tiempo, y si la solución que descubren resulta subóptima, no se puede hacer otra cosa que abandonar todo el trabajo hecho y empezar de nuevo. Sin embargo, ya que los AGs tienen descendencia múltiple, pueden explorar el espacio de soluciones en múltiples direcciones a la vez.

Otra ventaja notable de los algoritmos genéticos es que se desenvuelven bien en problemas con un paisaje adaptativo complejo -aquéllos en los que la función de aptitud es discontinua, ruidosa, cambia con el tiempo, o tiene muchos óptimos locales. Los algoritmos evolutivos han demostrado su efectividad al escapar de los óptimos locales y descubrir el óptimo global incluso en paisajes adaptativos muy escabrosos y complejos.

Otro área en el que destacan los algoritmos genéticos es su habilidad para manipular muchos parámetros simultáneamente. Muchos problemas de la vida real no pueden definirse en términos de un único valor que hay que minimizar o maximizar, sino que deben expresarse en términos de múltiples objetivos, a menudo involucrando contrapartidas: uno sólo puede mejorar a expensas de otro. Los AGs son muy buenos resolviendo estos problemas: en particular, su uso del paralelismo les permite producir múltiples soluciones, igualmente buenas, al mismo problema, donde posiblemente una solución candidata optimiza un parámetro y otra candidata optimiza uno distinto, y luego un supervisor humano puede seleccionar una de esas candidatas para su utilización.

Desventajas de los algoritmos genéticos

La primera y más importante consideración al crear un algoritmo genético es definir una representación del problema. El lenguaje utilizado para especificar soluciones candidatas debe ser robusto; es decir, debe ser capaz de tolerar cambios aleatorios que no produzcan constantemente errores fatales o resultados sin sentido.

El problema de cómo escribir la función de aptitud debe considerarse cuidadosamente para que se pueda alcanzar una mayor aptitud y verdaderamente signifique una solución mejor para el problema dado. Si se elige mal una función de aptitud o se define de manera inexacta, puede que el algoritmo genético sea incapaz de encontrar una solución al problema, o puede acabar resolviendo el problema equivocado.

Además de elegir bien la función de aptitud, también deben elegirse cuidadosamente los otros parámetros de un AG -el tamaño de la población, el ritmo de mutación y cruzamiento, el tipo y fuerza de la selección. Si el tamaño de la población es demasiado pequeño, puede que el algoritmo genético no explore suficientemente el espacio de soluciones para encontrar buenas soluciones consistentemente. Si el ritmo de cambio genético es demasiado alto o el sistema de selección se escoge inadecuadamente, puede alterarse el desarrollo de esquemas beneficiosos y la población puede entrar en catástrofe de errores, al cambiar demasiado rápido para que la selección llegue a producir convergencia.

1 http://the-geek.org/docs/algen/

Un problema muy conocido que puede surgir con un AG se conoce como convergencia prematura. Si un individuo que es más apto que la mayoría de sus competidores emerge muy pronto en el curso de la ejecución, se puede reproducir tan abundantemente que merme la diversidad de la población demasiado pronto, provocando que el algoritmo converja hacia el óptimo local que representa ese individuo, en lugar de rastrear el paisaje adaptativo lo bastante a fondo para encontrar el óptimo global. Esto es un problema especialmente común en las poblaciones pequeñas, donde incluso una variación aleatoria en el ritmo de reproducción puede provocar que un genotipo se haga dominante sobre los otros.

Eficacia del paradigma para la aplicación

La aplicación no desea resolver un problema, es una simulación de la selección, por lo que las ventajas y desventajas poco aplican al ámbito de este. La meta final la da el usuario, que es el que maneja las generaciones.

El programa maneja variables simples para los cromosomas, la función de adaptación es simple.

Conclusiones

El sistema funciona adecuadamente según los objetivos que se plantearon a priori, sin embargo varios objetivos secundarios no se lograron completar, pero no afectan el objetivo principal de este, como lo son la inclusión de retos, en los cuales el usuario debiera alterar el tipo de evolución de los chanchos para lograr objetivos (ejemplo: crear 2 chanchos del tipo 1, 2 del tipo 2, 2 del tipo 3 en determinado número de generaciones), así como la inclusión de opciones para que el usuario pueda cambiar aspectos funcionales de la aplicación (volumen de la música, pantalla completa, etc).

Se dice que la evolución cultural humana ha reemplazado a la biológica -que nosotros, como especie, hemos llegado a un punto en el que somos capaces de controlar conscientemente nuestra sociedad, nuestro entorno y hasta nuestros genes al nivel suficiente para hacer que el proceso evolutivo sea irrelevante. Se dice que los caprichos culturales de nuestra cambiante sociedad son los que determinan la aptitud hoy en día, en lugar de la marcha enormemente lenta, en comparación, de la mutación genética y la selección natural. En cierto sentido, esto puede ser perfectamente cierto.

Pero en otro sentido, nada podría estar más lejos de la verdad. La evolución es un proceso de resolución de problemas cuyo poder sólo comenzamos a comprender y explotar; a pesar de esto, ya está funcionando por todas partes, moldeando nuestra tecnología y mejorando nuestras vidas, y, en el futuro, estos usos no harán sino multiplicarse. Sin un conocimiento detallado del proceso evolutivo no habrían sido posibles ninguno de los incontables avances que le debemos a los algoritmos genéticos.

Referencias

Coley D. “Introduction to genetic algorithms for cientist and engineers” World Scientific

http://the-geek.org/docs/algen/

http://eddyalfaro.galeon.com/geneticos.html

http://geneura.ugr.es/~jmerelo/ie/ags.htm

http://es.wikipedia.org/wiki/Algoritmos_genéticos

http://the-geek.org/docs/algen/

Apéndices