TRABAJO FINAL
ESPECIALIDAD EN INGENIERÍA DE SISTEMAS EXPERTOS

Estudio de una Herramienta de Obtención de Sub-óptimos Basada en Algoritmos Genéticos

Autor: Lic. Andrea Cottone

Directora: M.Ing. Paola Britos

Octubre 2004

Contenido

Capitulo 1: Introducción A Los Algoritmos Genéticos

Capitulo 2: Manual De Usuario De La Herramienta

Capitulo 1: Introducción A Los Algoritmos Genéticos

Los Algoritmos Genéticos

Los *algoritmos genéticos* son métodos sistemáticos para la resolución de problemas de búsqueda y optimización que aplican los mismos métodos de la evolución biológica: selección basada en la población, reproducción y mutación.

Fueron desarrollados por John Holland, junto a su equipo de investigación, en la universidad de Michigan en la década de 1970.

El objetivo de un AG es buscar una "buena" solución a un problema determinado.

El cromosoma es el representante, dentro del algoritmo genético, de una posible solución al problema.

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

Pasos

1. Evaluar la aptitud de cada uno de los individuos.
2. Permitir a cada uno reproducirse, de acuerdo con su puntuación.
3. 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.

De esta manera el algoritmo genético va creando nuevas “generaciones” de la población, cuyos individuos son cada vez mejores soluciones al problema.

Cada uno de los pasos consiste en hacer algo con las cadenas de bits, es decir, la aplicación de un *operador* a una cadena binaria. A estos operadores se los denominan, *operadores genéticos*, y los tres principales son: *selección, cruza (crossover* o recombinación) y *mutación.*

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. Poblaciones pequeñas corren el riesgo de no cubrir adecuadamente el espacio de búsqueda y poblaciones grandes llevan a un excesivo costo computacional.

- *Condición de terminación:* lo más habitual es que además de la convergencia exista otra condición de terminación como ser el número máximos de generaciones.

Evaluación de la Aptitud

En el sistema biológico el material genético, sobrevive a cada generación, combinándose y ampliando su presencia en la población, en la medida en que las estructuras que lo contienen manifiesten aptitud relativamente buena.

Llevando esto al ámbito de la solución de problemas, un individuo representaría una solución posible y su aptitud indicaría cuan buena es la solución.

Durante la evaluación, se decodifica el gen, convirtiéndose en una serie de parámetros de un problema, se halla la solución del problema a partir de esos parámetros, y se le da una puntuación a esa solución en función de lo cerca que esté de la mejor solución. A esta puntuación se le llama aptitud (en ingles fitness).

La aptitud ayuda a determinar los cromosomas que se van a reproducir, y aquellos que se van a eliminar, pero hay varias formas de considerar a este valor durante la selección

Selección

Una vez evaluada la aptitud, se tiene que crear la nueva población teniendo en cuenta que los *buenos* rasgos de los mejores individuos. Esta selección, y la consiguiente reproducción, se puede hacer de las siguientes formas principales:

- *Rueda de ruleta*: Este método consiste en construir una ruleta particionada en ranuras de igual tamaño, las cuales se numeran. A cada individuo de la población se le asigna una cantidad de ranuras proporcional a su aptitud
  El proceso se repite hasta completar la cantidad de individuos deseados. Este método de selección otorga mayor probabilidad de contribuir a la siguiente generación a los individuos con mayor aptitud.
  Hay algunas otras variantes como por ejemplo, incluir en la nueva generación el mejor representante de la generación actual. En este caso, se denomina método *elitista*.

- *Selección por torneo*: En este caso dos individuos son elegidos al azar de la población actual y el mejor o más apto de los dos se coloca en la generación siguiente. Esto continúa hasta que se complete la nueva población.

- *Basado en el rango*: en este esquema se mantiene un porcentaje de la población, generalmente la mayoría, para la siguiente generación. Se coloca toda la población por orden de aptitud, y los *M* menos dignos son eliminados y sustituidos por la descendencia de alguno de los *M* mejores con algún otro individuo de la población.
  A este esquema se le pueden aplicar otros criterios; por ejemplo, se crea la descendencia de uno de los padres, y esta sustituye al más parecido entre los perdedores. Esto se denomina *crowding*, y fue introducido por *DeJong***.** En realidad, para este esquema se escoge un *crowding factor, CF*. Cuando nace una nueva criatura, se seleccionan CF individuos de la población, y se elimina al más parecido a la nueva criatura. [JJMG]

- *Método Estocástico:* Por cada individuo se calcula la aptitud relativa al promedio de aptitudes de la población, y en función de esto se asignan las copias. Por ejemplo, si la aptitud promedio de la población es 15 y la aptitud del individuo es 10; entonces su aptitud relativa es 1.5. Esto significa que se colocará una copia en la próxima generación y que se tiene el 0.5 (50 %) de chance de colocar una segunda copia.

Cruza o Crossover

Consiste en el intercambio de material genético entre dos cromosomas
La cruza es el principal operador genético, hasta el punto que se puede decir que no es un algoritmo genético si no tiene *cruza*, y, sin embargo, puede serlo perfectamente sin mutación, según descubrió Holland.
Para aplicar la cruza (crossover o recombinación), se escogen aleatoriamente dos miembros de la población.

Dos descendiente de los mismos padres pueden cruzarse; ello garantiza la perpetuación de un individuo con buena puntuación (y, además, algo parecido ocurre en la realidad; es una práctica utilizada, por ejemplo, en la cría de ganado, llamada *inbreeding*, y destinada a potenciar ciertas características frente a otras). Sin embargo, si esto sucede demasiado a menudo, puede crear problemas: toda la población puede aparecer dominada por los descendientes de algún gen, que, además, puede tener caracteres no deseados. Esto se suele denominar en otros métodos de optimización *atranque en un mínimo local*, y es uno de los principales problemas con los que se enfrentan los que aplican algoritmos genéticos [JJMG].

El operador de cruza es el encargado de mezclar bloques buenos que se encuentren en los diversos progenitores, y que serán los que den a los mismos una buena puntuación.
La presión selectiva se encarga de que sólo los buenos bloques se perpetúen, y poco a poco vayan formando una buena solución.

El *teorema de los esquemas* dice que la cantidad de *buenos bloques* se va incrementando con el tiempo de ejecución de un algoritmo genético, y es el resultado teórico más importante en algoritmos genéticos.

El intercambio genético se puede llevar a cabo de muchas formas, pero los grupos principales son:

- *Cruza Simple*: los dos cromosomas padres se cortan por un punto, y el material genético situado entre ellos se intercambia [RGM].

  Dada las siguientes estructuras de longitud 1 = 8, y eligiendo 3 como el punto de cruza se intercambian los segmentos de cromosoma separados por este punto.

  XYX | XYYYX
  YYX | YYXXY

  ⇨

  XYX | YYXXY
  YYX | XYYYX

- *Cruza de dos puntos*: En este método de cruza de dos puntos, se seleccionan dos puntos aleatoriamente a lo largo de la longitud de los cromosomas y los dos padres intercambian los segmentos entre estos puntos.

- *Cruza Multipunto:* el cromosoma es considerado un anillo, y se eligen *n* puntos de cruza en forma aleatoria. Si la cantidad de puntos de cruza es par, se intercambian las porciones de cromosomas definidas entre cada par de puntos consecutivos, si es impar se asume un punto de cruza adicional en la posición cero y se procede de igual modo [RGM].

  Dadas dos estructuras de longitud 1 = 8, con *n* = 4 puntos de cruza. Intercambiando los segmentos de la posición 2 a 4 y 6 a 7, se tiene:

  X | YXX | Y | YY | X
  Y | YXY | Y | XX | Y

  ⇨

  X | YXY | Y | XX | X
  Y | YXX | Y | YY | Y

- *Cruza binomial*: Para generar un cromosoma hijo por cruza binomial, se define la probabilidad P0 como la probabilidad de que el Alelo de cualquier posición del descendiente se herede del padre, y 1 – P0 como la probabilidad de que lo herede de la madre1.
  En este caso se puede construir un único hijo por cada aplicación del operador, o bien generar un segundo hijo como complemento del primero.
  Cuando existe igual probabilidad de heredar del padre como de la madre, P0 = 0,5 la cruza se denomina *uniforme.* Para estructuras de longitud l la cruza uniforme implica un promedio de *l*/2 puntos de cruza [DEJ/92].

Mutación

En la Evolución, una mutación es un suceso bastante poco común (sucede aproximadamente una de cada mil replicaciones), en la mayoría de los casos las mutaciones son letales, pero en promedio, contribuyen a la diversidad genética de la especie. En un algoritmo genético tendrán el mismo papel, y la misma frecuencia (es decir, muy baja).

Una vez establecida la frecuencia de mutación, por ejemplo, uno por mil, se examina cada bit de cada cadena. Si un número generado aleatoriamente está por debajo de esa probabilidad, se cambiará el bit (es decir, de 0 a 1 o de 1 a 0). Si no, se dejará como está. Dependiendo del número de individuos que haya y del número de bits por individuo, puede resultar que las mutaciones sean extremadamente raras en una sola generación.
No hace falta decir que no conviene abusar de la mutación. Es cierto que es un mecanismo generador de diversidad, y, por tanto, la solución cuando un algoritmo genético está estancado, pero también es cierto que reduce el algoritmo genético a una búsqueda aleatoria. Siempre es más conveniente usar otros mecanismos de generación de diversidad, como aumentar el tamaño de la población, o garantizar la aleatoriedad de la población inicial.

Capitulo 2: Manual De Usuario De La Herramienta

Sección 1

- Requisitos e Instalación del Sistema -

Introducción

El WinGA o Gwin2 es un programa basado en ventanas interactivas que se utiliza para demostrar y experimentar con algoritmos genéticos. El programa se ha diseñado para ser fácil de utilizar, no obstante se debe tener cierto conocimiento sobre algoritmos genéticos y cómo trabajan antes de usarlo. Por esto se recomienda comenzar por el capítulo 1.
Permite experimentar con los problemas incluidos, como así también es posible introducir nuevos problemas en el sistema para encontrarles solución.
El programa almacena problemas en las bibliotecas dinámicas DLLs. Esto significa que el usuario del software puede escribir un problema que desee solucionar en cualquier lenguaje de programación que se apoye en DLLs.

Este manual se desarrolla a lo largo de las siguientes secciones:

Sección 1: Requisitos e instalación del sistema.

Sección 2: Operadores.

Sección 3: Menúes, diálogos y ventanas.

Sección 4: Escribiendo funciones de evaluación.

Sección 5: Comienzo rápido.

Requisitos Del Sistema

Hardware

Requerimientos mínimos de sistema
- PC 386+
- memoria 2Mb
- disco duro del 1Mb.
- Monitor de EGA/VGA 640 x 350.

Sistema recomendable
- PC 486+
- memoria 4Mb
- disco duro del 1Mb.
- Monitor de SVGA 800 x 600.

Software

Obligatorio
- DOS 3.3 o superior
- Windows 3.1 o superior

Opcional (requerido para escribir funciones de evaluación propias)
- Borland C++ para las ventanas
- PASCAL de Borland para las ventanas
- Cualquier lenguaje que se apoye en DLLs

Instalación

Copiar los archivos en un nuevo directorio en su disco rígido, por ej. C:\GWIN

El directorio debe contener los siguientes archivos:

GWIN2.EXE	Archivo ejecutable
GWIN2.HLP	Archivo de ayuda
BWCC.DLL	Librería de Borland
MASTER1.DLL	Funciones de evaluación.
EXAMPLE1.DLL	Ejemplo de funciones.
EXAMPLE1.PAS	Ejemplo de código fuente.

Sección 2

- Operadores -

Operadores Provistos

Esta sección cubre brevemente los operadores provistos por el WinGA. Estos pueden ser divididos en selección, combinación y mutación.

Selección

Los individuos elegidos con el operador de selección aportarán sus genes a la generación siguiente.

- **Remainder Stochastic**

  Por cada individuo se calcula la aptitud relativa al promedio de aptitudes de la población, y en función de esto se asignan las copias. Por ejemplo, si la aptitud promedio de la población es 15 y la aptitud del individuo es 10; entonces su aptitud relativa es 1.5. Esto significa que se colocará una copia en la próxima generación y que se tiene el 0.5 (50 %) de chance de colocar una segunda copia.

- **Tournament - Torneo**

  En este caso dos individuos son elegidos al azar de la población actual y el mejor o más apto de los dos se coloca en la generación siguiente.
  Esto continúa hasta que se llene la nueva población.

Combinación

Este operador trabaja sobre pares de estructuras (padres) para producir nuevas estructuras (hijos) por el intercambio de segmentos entre los padres.

- **One point crossover**

  En este método, de cruza en un punto, también conocido como cruza simple, se elige al azar uno de los posibles puntos de cruza. Luego de esta elección se intercambian los segmentos del cromosoma separados por este punto.

- **Two point crossover**

  En este método de cruza de dos puntos, se seleccionan dos puntos aleatoriamente a lo largo de la longitud de los cromosomas y los dos padres intercambian los segmentos entre estos puntos.

- **Two point ring crossover**

  Los extremos de los cromosomas se juntan para formar un lazo o anillo y dos puntos se seleccionan aleatoriamente. Los padres intercambian los segmentos entre estos puntos.

Mutación

La mutación permite mantener diversidad en la población disminuyendo el riesgo de convergencia prematura.

- **Mutación normal**

  Cada gen en los cromosomas se selecciona con una probabilidad p y cambian su valor con una probabilidad v. En este caso v es fijo en 0.5 y p es modificable pero debe ser pequeño por ejemplo 0.00001.

Sección 3

- Menúes, Diálogos y Ventanas -

Menúes

Esta sección contiene una breve descripción de los comandos de menú disponibles y de sus propósitos.

Archivo
Este menú se utiliza para abrir y para grabar archivos de log

- **Open...**

  El comando Open abre la ventana de dialogo Open File.
  En esta ventana se selecciona el archivo que desea abrirse.

- **Save**

  El comando de File/Save graba el archivo de la ventana de edición activa al disco.
  Si se está utilizando un archivo existente, WinGA pregunta si desea sobreescribir el archivo.

- **Save as...**

  File/Save As abre el dialogo *File/Save As*, donde se puede guardar el archivo de log con un nombre distinto o a un directorio distinto.
  Se puede incorporar el nuevo nombre del archivo, incluyendo el disco y el directorio.
  Todas las ventanas que contienen este archivo se ponen al día con el nuevo nombre.
  Si se elige un nombre de un archivo existente, WinGA pregunta si desea sobreescribir el archivo existente.

- **Exit**

  El comando de File/Exit sale del WinGA y lo quita de memoria.

Edit

Este menú se utiliza para editar archivos de log.

- **Undo**

  El comando Undo, deshace la acción más reciente.

- **Cut**

  El comando cut quita el texto seleccionado del archivo de log y lo coloca en el portapapeles.
  Se puede elegir Edit/Paste para pegar el texto en cualquier otro documento (o a otra parte en el mismo archivo).

- **Copy**

  El comando del copy deja el texto seleccionado intacto hace una copia exacta de él en el portapapeles.

- **Paste**

  El comando de la paste inserta el texto seleccionado del portapapeles en la ventana actual en la posición del cursor.

- **Delete**

  Este comando borra el texto seleccionado del documento, pero no lo pone en el portapapeles.

- **Clear All**

  Este comando suprime todo el texto del archivo actualmente seleccionado. Es equivalente a marcar todo el texto y después a seleccionar la opción de Edit/Delete.

Search

Este menú se utiliza para encontrar datos en archivo de log.

- **Find**

  Se abre un cuadro de diálogo para permitir que el usuario ingrese el texto que desea buscar.

- **Replace..**

  Se abre un cuadro de diálogo para permitir que el usuario ingrese texto que desea buscar y el texto por el cual lo desea reemplazar.

- **Next**

  Repite el último comando Search/Find o de Search/Replace para encontrar la secuencia de texto siguiente.

Window

Este menú se utiliza para manejar ventanas del escritorio.

- **Tile**

  Esta opción arregla las ventanas de manera que cubren el tablero del escritorio entero sin solaparse.

- **Cascade**

  Esta opción arma cada ventana del mismo tamaño dejando solamente la barra del título de cada una visible.

- **Arrange Icons**

  Los iconos se espacian uniformemente, comenzando en la esquina izquierda más baja del tablero del escritorio.
  Al menos una ventana debe estar minimizada para que esta opción esté activa.

- **Close all**

  Window/Close all permite cerrar todas las ventanas abiertas del escritorio.

- **Save state**

  Al elegir esta opción WinGA guardará el escritorio y el estado actual del programa.
  Esto se almacena en un archivo con extensión .DSK.

- **Restore state**

  Este comando restaurará el programa a una condición anterior usando el archivo DSK.

Functions
Este menú se utiliza para seleccionar funciones de la evaluación.

- **Load DLL**

  Al seleccionar Functions/Load Dll se abre un cuadro de dialogó que permite seleccionar una DLL.
  Luego WinGA verifica que sea una DLL que contiene funciones de la evaluación.

- **Select Function**

  Este menú permite seleccionar la función de evaluación, o bien modificar los parámetros de la función actualmente seleccionada.

- **Status**

  Esta opción exhibe el estado actual de la función de evaluación.

Setup
Este menú se utiliza para seleccionar los operadores genéticos y para configurar el AG

- **Selection**

  Este menú exhibe un cuadro de dialogo que permite que se seleccione o se cambie el operador de selección.

- **Combination**

  Este menú exhibe el cuadro de dialogo *Combination que perrmite* seleccionar o cambiar el operador de combinación o cruza.

- **Mutation**

  Este menú exhibe el cuadro de dialogo *Mutation* que permite seleccionar o cambiar el operador de mutación.

- **General**

  Este menú exhibe a cuadro de diálogo *General* que se utiliza para alterar parámetros generales como el tamaño de la población, etc.

- **Status**

  Este menú exhibe el estado de la configuración actual del WinGA.

Views
Se utiliza para ir mostrando información cuando el WinGA está funcionando.

- **Text Stats**

  Esto exhibe un texto basado en *ventana de la estadística* (véase la sección de Windows para más información).

- **Fitness Graph**

  Este gráfico muestra la *progresión de la aptitud*. (véase Sección de Windows para más información).

- **Distribución total**

  Este menú exhibe *histograma de la distribución total*. (véase la sección de Windows para más información).

- **Distribución actual**

  Este menú exhibe el *histograma de la distribución actual*. (véase la sección de Windows para más información).

Reports

Se utiliza para la grabación de información sobre el AG

- **Log file**

  Este menú permite indicarle al sistema la información que se desea guardar en el archivo de log. Permitiendo registrar la actividad del WinGA en funcionamiento.

Run

Este comando comienza la ejecución del WinGA. Si el WinGA no se ha configurado correctamente se mostrará un cuadro indicando el problema.
Si por el contrario todo está en orden comenzará la ejecución. Seguido por todas las etapas normales de la evolución. El AG continuará ejecutándose hasta se de una de las siguientes situaciones:

- la población converja o,
- el AG alcance el número máximo de las generaciones permitidas.

Para parar o detener brevemente el GA antes de este punto debe seleccionar la opción de menú stop.

Diálogos

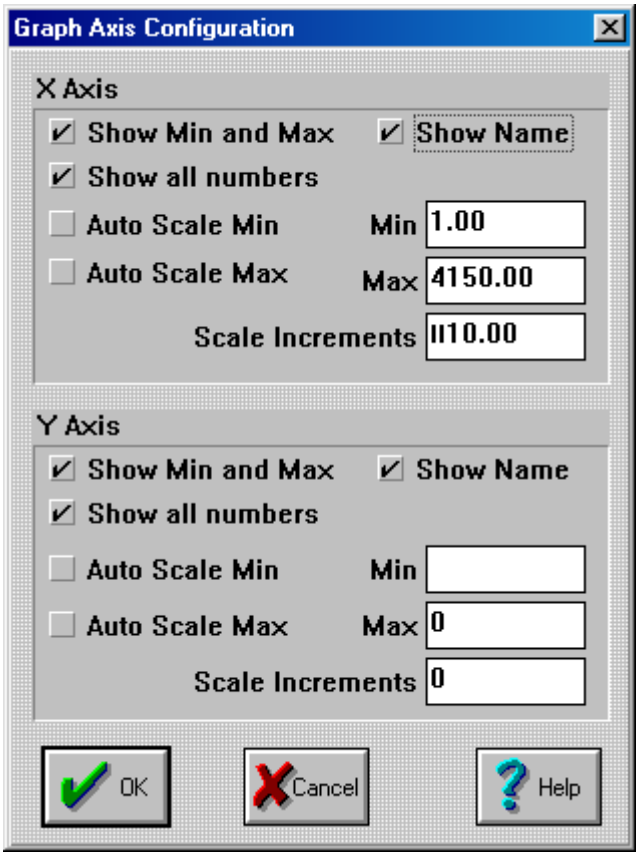
Esta sección contiene una lista de cuadros de diálogo que pueden aparecer al ir utilizando el WinGA. Se mostrará qué hacen, cómo se deben usar y como se ven.

Graph Axis Configuration

Propósito

Permite que la reconfiguración del eje de un gráfico, para seleccionar distintas opciones de display

Apariencia



Uso

Esta cuadro de diálogo permite la alteración del eje X e Y en una ventana del gráfico. Se accede presionando el botón derecho del mouse sobre el gráfico.

Contiene los siguientes controles:

Show Min and Max
Si se selecciona se mostraran los valores mínimos y máximos en el gráfico.

Show all numbers
Si se hace selecciona y el eje tiene un incremento, entonces cada incremento se verá impreso el gráfico.

Auto Scale Min
Si se selecciona cualquier valor menor que el valor actual en el gráfico forzará el gráfico a redibujarse en la escala elegida.

Auto Scale Max
Si se selecciona cualquier valor mayor que el valor actual en el gráfico forzará el gráfico a redibujarse en la escala elegida.

Show Name
Esto determina si se muestra la etiqueta del eje del gráfico. Si no se selecciona entonces habrá más espacio para el cuerpo del gráfico.

Min
Esto fija el valor mínimo de un eje del gráfico. Si Auto Scale (escalamiento automático) está seleccionado, entonces este valor puede ser eliminado.

Max
Esto fija el valor mínimo de un eje del gráfico. Si el Auto Scale (escalamiento automático) está seleccionado entonces este valor puede ser eliminado.

Scale Increments

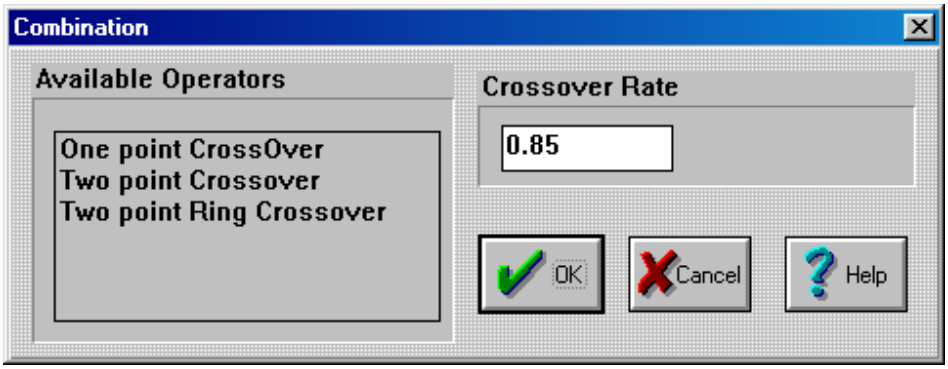
Esto determina en qué puntos de la escala se exhiben en el eje del gráfico. Por ejemplo, si esto se fija en 10 entonces se colocará un marcador para cada aumento en valor a lo largo de un eje que sea divisible por 10.

Combination

Propósito

Permite seleccionar y modificar los parámetros de la cruza

Apariencia



Uso

La caja "available operators" contiene todos los operadores disponibles. Si se ha seleccionado a un operador ya aparecerá resaltado. Para seleccionar un nuevo operador se debe mover el puntero del mouse sobre él y hacer clic.

También se puede modificar la tasa de cruza:

Crossover Rate

Este número determina qué porcentaje de la población experimentará la cruza. Por ejemplo el valor prefijado es 0.85 es decir, que el 85% de la población se cruzará.

El rango permitido es 0.0 - 1.0.

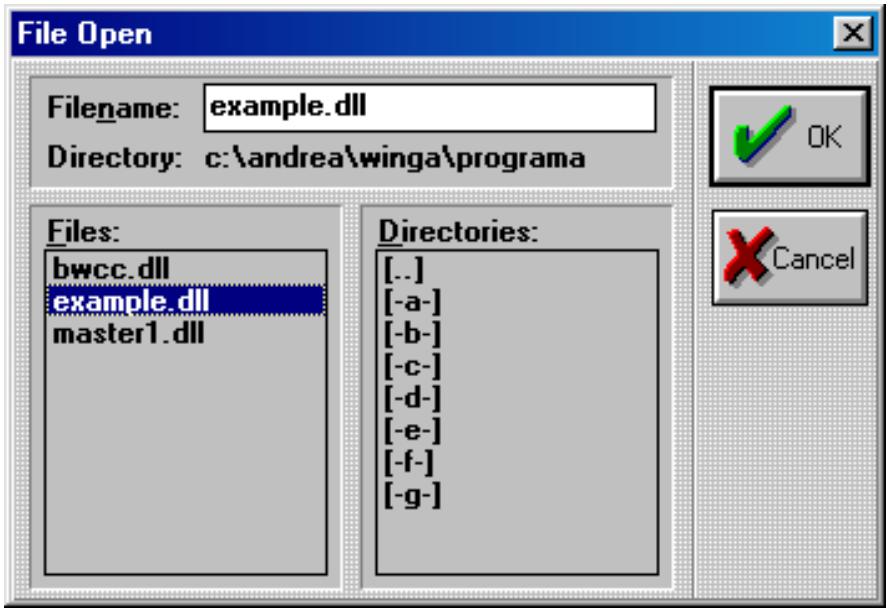
File Open /DLL

Propósito

Permite la selección de una DLL que lleva a cabo funciones de la evaluación.

Se accede por el menú Functions – Load Dll

Apariencia



Uso

Permite cargar una Dll tipeando el nombre o bien seleccionándolo de la lista

File Name input Box

Aquí es donde se ingresa en nombre de la Dll que se desea.

Files

Aquí se muestra todas las Dll disponibles en el directorio seleccionado.

Directories

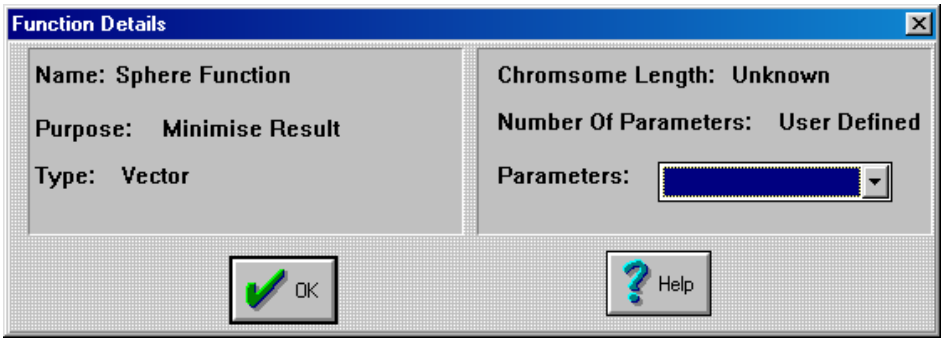
Permite moverse por los distintos directorios o carpetas

Function Details

Propósito

Este cuadro muestra información sobre la función de evaluación/aptitud seleccionada. Se accede por el botón Info de la ventana, Evaluation Function.

Apariencia



Uso

Este cuadro contiene la siguiente información (no editable):

Nombre

Éste es el nombre externo/alias de una función de la evaluación.

Purpose

Ésta es la meta de la función. Las metas actuales son reducir al mínimo o maximizar el resultado de una función.

Type

Esto indica el tipo de parámetros que la función espera recibir.

Chromosome Length

Esto es cuántos bits se requieren para representar el cromosoma requerido por la función. Puede exhibir a veces a “unkown” (desconocido) o al 'user defined' (definido por el usuario).

Numbers of Parameters

Indica cuantos parámetros/ variables son requeridos por la función de evaluación. Esto puede decir a veces 'user defined'

Parameters

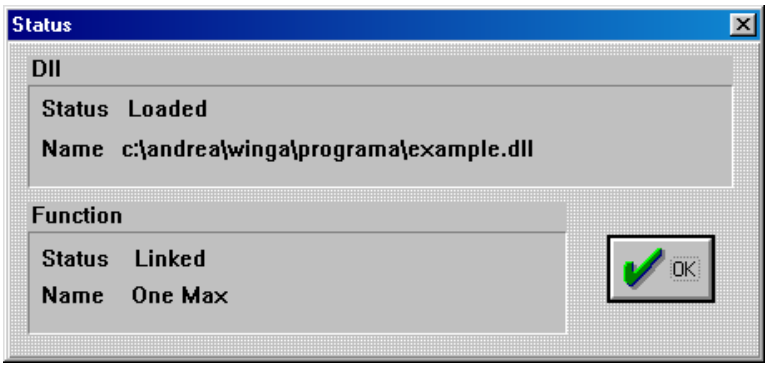
Si la función de evaluación /aptitud requiere de parámetros, en este cuadro se mostrara el tipo y en rango de estos parámetros.

Function Status

Propósito

Muestra el estado actual de la DLL y de la función

Apariencia



Uso

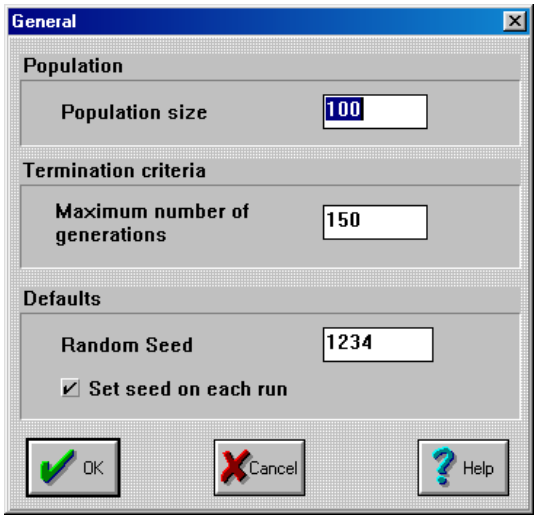
Muestra el nombre y el estado de la Dll seleccionada, así como también, el nombre y el estado de la función.

General

Propósito

Permite alterar los parámetros generales asociados al AG

Apariencia



Uso

Contiene los siguientes controles:

Population size

Esto fija el tamaño de la población es decir del AG. Cuanto más grande es el número, mayor es la variedad en la población; pero esto es expensas del tiempo de ejecución. El valor por defecto es 100.

Maximun number of generations

Este es el número máximo de generaciones para la que el AG se va a ejecutar. Es decir, que si el AG no converge se ejecutara este número de veces.

Random Seed

Cuando el AG comienza a ejecutarse crea una población de cromosomas al azar. Para permitir al usuario recrear una situación determinada; el sistema permite el ingreso de un generador random. Esto garantiza el mismo resultado en cada ejecución.

Set seed on each run

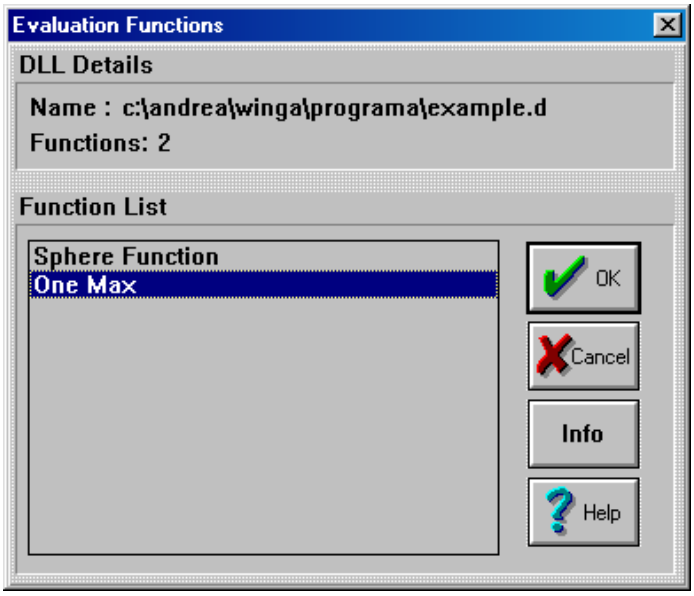
Si se selecciona, el número random se introducirá en cada ejecución del AG. En este caso el AG producirá exactamente los mismos resultados para una serie de ejecuciones. Si no se selecciona en cada ejecución se podrán obtener distintos resultados.

Evaluation Functions

Propósito

Seleccionar una función de evaluación.

Apariencia



Uso

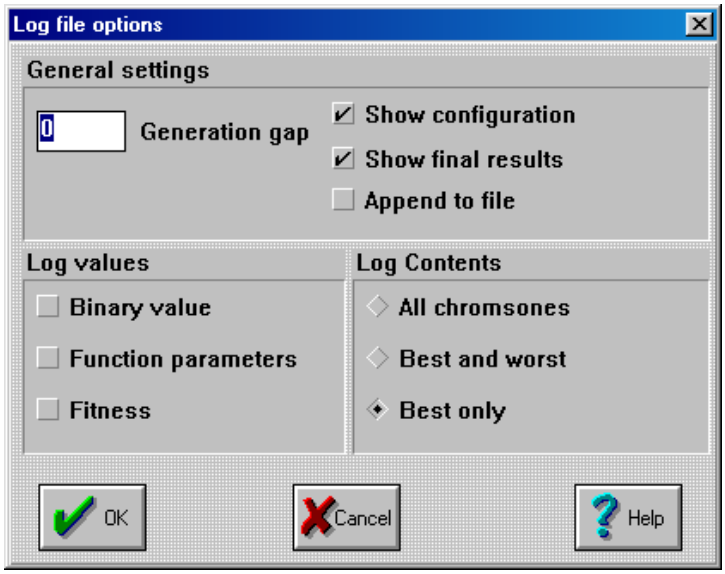
Se utiliza para seleccionar una función de evaluación. Para seleccionar una función hay que hacer doble clic o marcarla y presionar el botón OK.

Log File Options

Propósito

Configurar el archivo de log.

Apariencia



Uso

El archivo de log registra el estado de un AG funcionado. Se utiliza para guardar un registro permanente de los resultados. Al presionar Enter o el botón 'OK' el sistema solicita un nombre para el archivo de log.

Este diálogo tiene los controles siguientes:

Generation Gap

Esto determina la frecuencia de actualización del archivo de log. Por ejemplo si el valor es 1, el archivo de log será actualizado en cada generación. Si el número es 3 entonces el log será puesto al día cada tercera generación. El valor por defecto es 0, donde el archivo de log no es actualizado en ninguna generación.

Show configuration

Si se selecciona, una cabecera conteniendo información de la configuración será agregado al archivo de log.

A continuación se muestra una cabecera de ejemplo y su correspondiente explicación:

Start Time: 17:55:42 Dll Name: c:\gwin2\master1.dll Function Name: Sphere Function's Objective: Minimise Result Function Type: Vector Bit Length: 30 Parameter Count: 3 Function Parameter 0) Real -5.120 to 5.110 Function Parameter 1) Real -5.120 to 5.110 Function Parameter 2) Real -5.120 to 5.110 Selection type: Remainder Stochastic Recombination type: Two point Crossover Recombination Rate: 0.85000 Mutation type: normal mutation Mutation Rate: 0.00050 Population Size: 100 End of Header	Hora de comienzo: 17:55:42 Nombre de la dll: c:\gwin2\master1.dll Nombre de la función: Sphere Objetivo de la función: minimizar el resultado Tipo de la función: vector Longitud del bit: 30 Cuenta del parámetro: 3 Parámetro de la función 0) -5 real .120 a 5.110 Parámetro de la función 1) -5 real.120 a 5.110 Parámetro de la función 2) -5 real.120 a 5.110 Tipo de la selección: estocastica Tipo de la recombinación: cruza de dos puntos Tasa de recombinación:0.85000 Tipo de la mutación: mutación normal Tarifa de la mutación:0.00050 Tamaño de la población: 100 Fin

Show final results

Si se selecciona, los resultados finales del funcionamiento de AG se agregan al final del archivo de log.

Un ejemplo se demuestra abajo :

Final Results End Time: 17:55:46 Total Run Time: 0:0:4 Overall Best: 0.01100 Overall Worst: 64.11410 Last Average: 0.01100 Last Generation: 18 Total number of function Evaluations: 555 Final Best result Fitness: 0.01100 BitString 100000101001111111010111111111 Paramter 0) 0.10000 Paramter 1) -0.03000 Paramter 2) -0.01000	Resultados Finales Tiempo de finalización:17:55:46 Tiempo total de ejecución: 0:0:4 Lo mejor Posible:0.01100 La peor: 64.11410 Ultimo Promedio:0.01100 Ultima Generación: 18 Número total de las evaluaciones de la función: 555 Mejor resultado final Aptitud:0.01100 BitString 100000101001111111010111111111 Paramter 0) 0.10000 Paramter 1) -0.03000 Paramter 2) -0.01000

Append to file

Si se selecciona esto, las ejecuciones consecutivas del AG irán agregando sus informes al final de un archivo especificado en lugar de sobreescribir el contenido del archivo.

Log Contents

Si el generation gap es mayor que cero el estado de varios cromosomas será escrito al archivo de log. Hay una opción que permite registrar el mejor cromosoma, los cromosomas mejores y peores o todos los cromosomas. Cabe observar que la registración de todos los cromosomas producirá un archivo de log muy grande.
La información registrada sobre un cromosoma se define con los controles siguientes.

Binary value

Si se selecciona esta opción, el valor binario (bit string) se registra en el archivo de log.

Function parameters

Si se selecciona esta opción, se incluyen en el archivo los parámetros pasados a la función.

Fitness

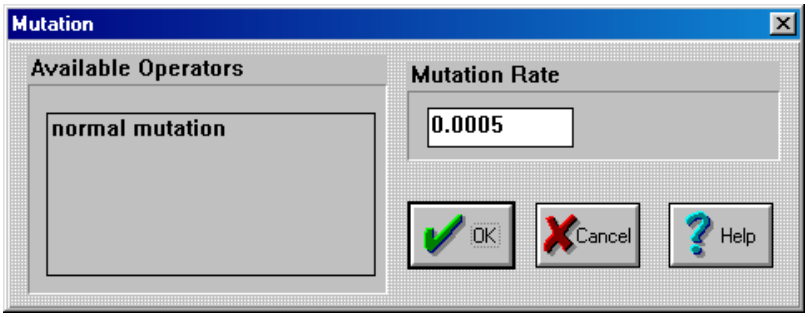
Si se selecciona esta opción, la aptitud del cromosoma se registra en el archivo de log.

Mutation

Propósito

Permite la selección y la alteración del parámetro del operador de mutación.

Apariencia



Uso

Se muestra la lista de operadores disponibles. Si se ha seleccionado un operador, el mismo aparecerá destacado.

Tasa de Mutación

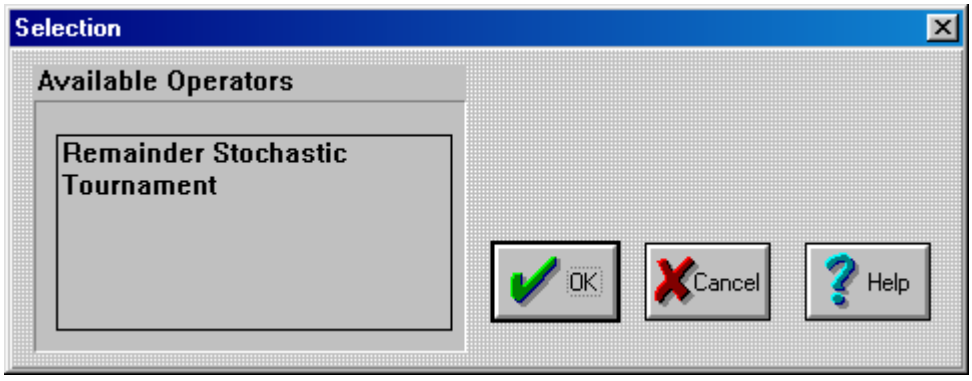
Este número determina cuál es la chance de ser mutado al azar que tiene un gen. Por ejemplo el valor prefijado es 0.0005. El gen tiene una posibilidad en 2000 de ser mutado. El rango de valores permitidos es 0.0 - 1.0 .

Selection

Propósito

Permite la selección de los operadores de la selección.

Apariencia



Uso

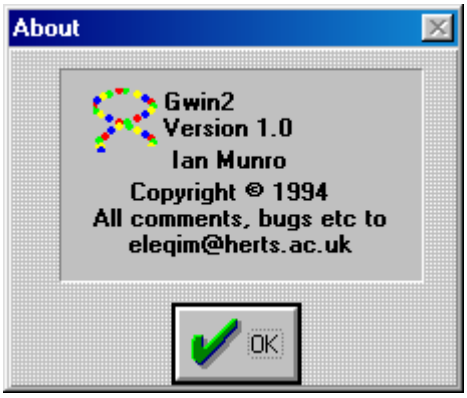
Se muestra la lista de operadores disponibles.
Si se ha seleccionado un operador, el mismo aparecerá destacado. Para seleccionar un operador distinto simplemente hay que clikear con el mouse sobre él.

About

Propósito

Muestra la versión actual del programa.

Apariencia



Uso

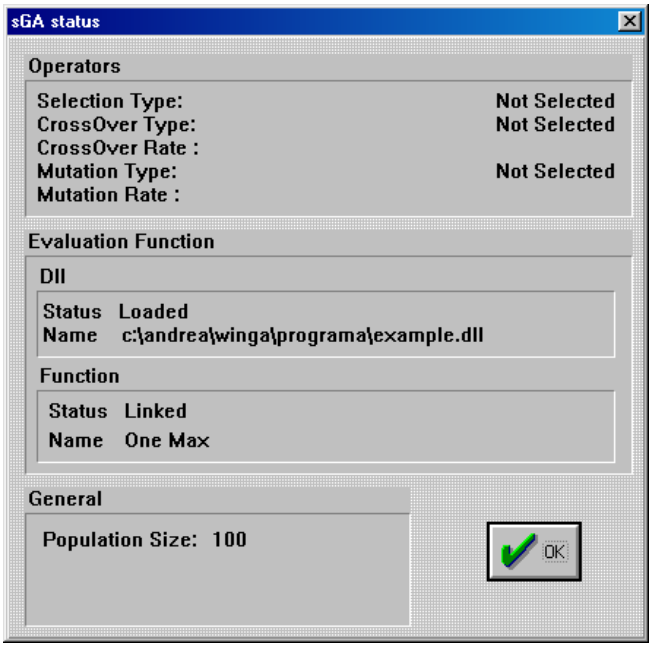
Para salir hay que presionar el botón ok

GA Status

Propósito

Exhibe el estado actual del sistema de AG.

Apariencia



Uso

Este cuadro de diálogo exhibe el estado actual y los ajustes del WinGA. Para salir hay que presionar el botón OK.

Ventanas

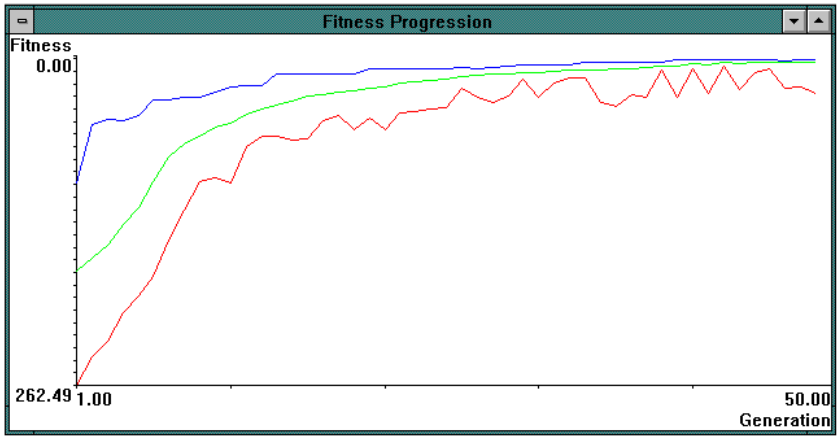
Esta sección contiene detalles sobre las ventanas que están disponibles al usar el WinGA. Se detalla lo que hacen, cómo utilizarlas y como se ven.

Fitness Progression

Propósito

Sigue la mejora de la aptitud de la población.

Apariencia



Uso

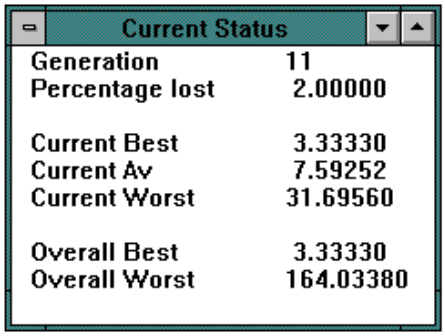
Un gráfico típico de la aptitud aparecerá similar al cuadro de arriba. La línea superior representa el progreso del mejor individuo de la población. La línea más baja representa al peor individuo de la población, y el centro representa la aptitud media de la población. El gráfico se redibuja siempre para acomodar el peor al valor actual. Abriéndose otro gráfico de la aptitud mientras que el sistema está funcionando se obtiene un zoom de la población actual. Esto puede ser útil en funcionamientos largos cuando las tres líneas comienzan a converger a una.

Current Status

Propósito

Exhibe el estado actual del sistema de AG

Apariencia



Uso

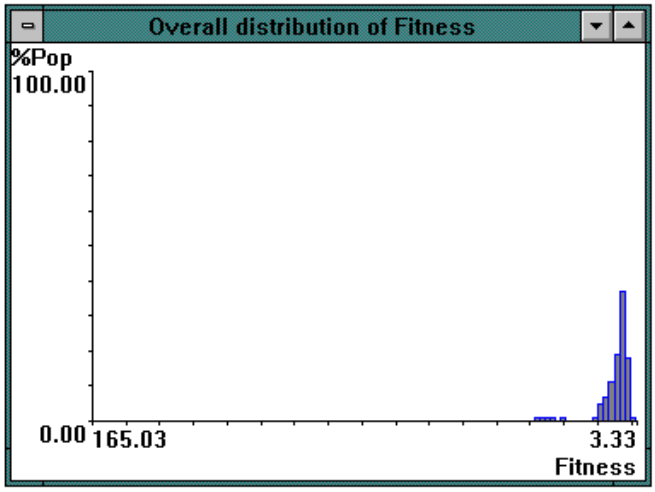
Este cuadro de dialogo muestra la información básica sobre el estado actual del AG.

Overal distribution of fitness

Propósito

Muestra un histograma de la distribución de la aptitud.

Apariencia



Uso

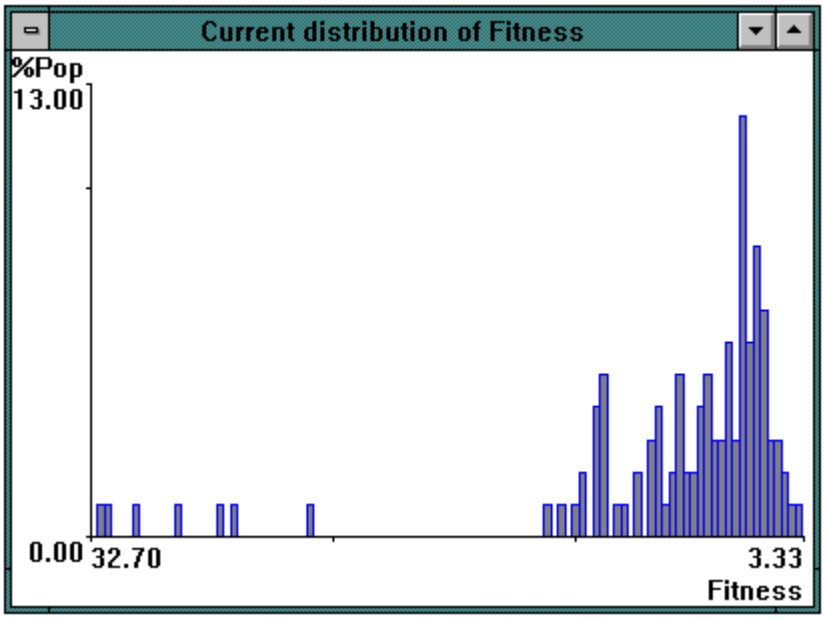
Esta ventana demuestra cómo la distribución de la aptitud fue a cambiado en el tiempo. La escala se dibuja con a la aptitud mejor y peor total. El eje de Y muestra el porcentaje de la población que comparte el mismo valor. En el ejemplo de arriba es un AG que ha estado funcionando por una cantidad de tiempo.

Current Distribution of Fitness

Propósito

Muestra un histograma de la distribución de la aptitud sobre la población actual.

Apariencia



Uso

Esta ventana demuestra la distribución de la aptitud de la generación actual. Es una versión ampliada (zoom) de la distribución total. La aptitud se escala entre el mejor y peor individuo actual.

Sección 4

- Escribiendo Funciones de Evaluación -

Introducción

Escribir funciones de evaluación en WinGA es una cuestión relativamente simple.
Se mostrará el desarrollo de una DLL simple la cual contiene dos funciones; ' OneMax ' y ' Sphere '. OneMax es una función cuyo propósito es minimizar el número de ceros en una cadena de caracteres binaria. Sphere devuelve la suma del cuadrado de una serie de números reales.

Pasaje de Información

Cualquier función que se escriba tiene que recibir dos variables. La primera variable es un puntero y la segunda es un número entero.
Por ejemplo:

Function fc1(Vect: puntero n: número entero): Real;

El puntero señala a donde se almacenan los parámetros requeridos por la función. Por ejemplo si la función requiere tres reales entonces el puntero contendrá la dirección de un buffer intermediario que contenga tres Reales almacenados consecutivamente.
El número entero contiene el número de los parámetros almacenados en la dirección.
Esto permite que la función reciba datos del algoritmo genético en uno de los cuatro niveles de abstracción siguientes:

- **Cadena empaquetada - PackedFunc**

  En el nivel más bajo, la función puede recibir un puntero al cromosoma comprimido y a su longitud. Por ejemplo:

  Function MyFunc(PackedVal: Puntero; Longitud: Número entero);
  ....
  End;

  la interpretación de la cadena depende enteramente del usuario, de manera que permite gran flexibilidad y eficacia. Sin embargo, la forma empaquetada del cromosoma puede ser difícil de tratar.

- **Cadena Bit - BitFunc**

  Hay un nivel intermedio de la representación donde el valor desempaquetado del cromosoma (un puntero a la cadena binaria) y la longitud se pasan a la función.

  Function BitFunc(Buff: Puntero; Long: número entero);
  ...
  End;

  Donde Buff es un vector de caracteres que contiene ceros y unos y long es la longitud del vector intermediario del arsenal.

- **Vectores – VectFunc**

  En el nivel siguiente, se pasa un puntero a un vector y el número de vectores. Un puntero a un vector consiste en un puntero a una serie de tipos de datos idénticos.

- **Lista de Parámetros - ValFunc**

  En el nivel más alto, se pasa un puntero a un buffer y el número de parámetros en el buffer.

  ```
  Function  ValFunc(Buffer: Indicador; Longitud: Número entero);
     ....
  End;
  ```

  el buffer contiene una serie de parámetros predefinidos por el usuario Por ejemplo: el buffer puede contener dos reales, un número entero y un booleano.

¿Qué información debe proveer la Dll?

Todo lo que el usuario tiene que hacer es especificar qué parámetros necesita que sean pasados a la función y el software de AG calculará automáticamente la longitud del cromosoma requerido y la codificará y descifrará cuando lo requiera.

Toda DLL debe tener dos funciones que son usadas por el AG para determinar si la Dll es valida o no. Estas dos funciones son:

```
Function GetCount: número entero; export;
Begin
  ....
End;
```

Get count debe volver el número de las funciones de evaluación dentro del DLL. En este ejemplo devolverá el valor dos. Get count debe tener un índice de referencia de 2 dentro del DLL de modo que la posición sea siempre fija.

```
Function GetFunc(indx: número entero):Pchar; exportación;
Begin
  ....
End;
```

GetFunc devuelve un puntero a una cadena terminada en nulo que contiene la descripción de una función usando el lenguaje descriptivo de funciones (FDL).

El string a devolver se decide mediante *indx* , el cuál define que cadena devolver de un vector con un rango [0.. GetCount-1]. En este caso los únicos valores del índice que son válidos son 0 y 1.

Definiciones del FDL

Lo que sigue es el código en FDL de las dos funciones, OneMax y Sphere previamente descritas:

```
[ "One Max" f1:Min, BitFunc; ]

[ "Sphere " F2:Min, VectFunc, Real -5.12, . 5.11 1024; ]
```

Esta secuencia tiene el sintaxis siguiente
([*Función*])

{} Denota opcional

() Denota 1 o más ocurrencias

Cada función se describe entre corchetes, tiene el sintaxis siguiente:

["*Función alias*" *NombreFunción*: *Objetivo, Tipo {(, parámetros)}*;]

- **Función Alias**

  Es el nombre de la función de la evaluación que ve el usuario final. Por ejemplo: al usar la DLL en cuestión el usuario verá que hay dos funciones llamadas "One Max" y " Sphere " .

- **Nombre De la Función**

  Es el nombre real de la función dentro de la DLL. Por ejemplo si el usuario selecciona " One Max" entonces el AG se mapeará a una función llamada f1.

- **Objetivo**

  Dice al AG cuál es el objetivo de la función. Las funciones soportadas son maximizar o minimizar el resultado de una función. Por lo tanto los únicos valores permitidos para el objetivo son MIN y MAX.

- **Tipo**

  Esto informa al AG qué tipo de datos espera la función recibir. Como se dijo anteriormente, los tipos actualmente soportados son:

Tipo	Tipo Descripción
PackedFunc	Función espera recibir una cadena empaquetada
BitFunc	Función espera recibir una cadena Bit
VectFunc	Función espera recibir vectores
ValFunc	Función espera recibir una lista del parámetros

- **Parámetros**

  Los parámetros le indican al AG qué tipos de datos espera recibir la función de la evaluación. Esto no se aplica a los tipos de PackedFunc y de BitFunc que reciben siempre el mismo estilo de los datos.
  Los parámetros tienen el sintaxis siguiente:

  TipoDeDato {rango} { repeticiones}

- **Tipo de Dato**

  Esto denota cuál es el tipo de dato de un parámetro particular por ejemplo: Real, Integer, short, etc.

- **Rango**

  El rango es opcional y denota los valores válidos que espera la función.
  Tiene la siguiente sintaxis:

  MinVal, MaxVal, Resolución

- **Repeticiones**
  Esto denota cuántas copias de este parámetro son iguales, esto es para reducir el tamaño de la llamada a la función.
  Tiene el sintaxis siguiente:

  Rep *n*

  Donde n es el número de repeticiones.

Ejemplos de FDL

Los siguientes son ejemplos del lenguaje para demostrar cómo se utiliza.

["función Prueba1" func1:Min, PackedFunc;]

Ésta es una función minimizante que espera un cromosoma empaquetado como parámetro.

["función Prueba2" tmp: Max VectFunc, Short;]

Ésta es una función de maximización que espera un vector de números enteros cortos como parámetros.

["función Prueba3 ", F2:Min, VectFunc. Real, -5.12, 5.12,. 1024;]

Ésta es una función de minimizante que espera un vector de números reales, los cuales están entre -5.12 a 5.12 y tienen una resolución de 1024.

["función Prueba4" ,f3:Min, ValFunc, Real, Integer, Rep 3, Real 1, 10, 16 , Rep 10;]

Ésta es una función minimizante que espera 1 número real, 3 números enteros y 10 números reales en el rango de 1 a 10.

Código de fuente del ejemplo en PASCAL

A continuación se expone el código de fuente escrito en Turbo Pascal implementa las funciones descriptas.

```
{$R-}

uses Strings;
Const
MaxFunc = 2;

{Esto es cuantas funciones de evaluación contiene la Dll}

FunctionList: Array[0..MaxFunc-1] of Pchar =
                                        ( '[ "Sphere " f1:Min, VectFunc, Real -5.12, 5.11 1024;]',
            '[ "One Max" f2:min bitfunc ;]');
Type
     PArray = ^TArray;
     TArray = Array[0..0] of Real;


{Esto es un tipo de datos}

Function GetFunc(indx:integer):Pchar;  export;
Begin
```

```
    If indx <MaxFunc Then
        GetFunc := FunctionList[indx]
    Else
        GetFunc := nil;
End;

Function GetCount: integer; export;
Begin
    GetCount := MaxFunc;
End;

Function f1(Vect:Pointer;  n:integer):Real; export;

{Esta es la función Sphere}

    Sum : Real;
    f : integer;
    P : PArray;
Begin
    P := Vect;
  {Convierte el puntero a un array}

    Sum := 0.0;
    for f := 0 to n-1 do
    Begin
        Sum := Sum +  Sqr(p^[f]);
    End;
    f1 := sum;
end;

Function f2(Vect:Pointer;  n:integer):Real; export; {Esta es la función OneMax}
Var
 f : integer;
 P : Pchar;
 Sum : integer;
Begin
    P := Vect;
    sum := 0;
    For F := 0 to n-1 Do
        If P[f] = '0' Then
            Inc(sum);
    f2 := sum;
End;

Exports
{Acá es donde se define que funciones están visibles} GetFunc index 1,
       GetCount index 2,
       f1 index 3,
       f2 index 4;
Begin
end.
```

Sección 5

- Comienzo Rápido -

Esta sección describe brevemente cómo conseguir rápidamente resultados usando WinGA.

El funcionamiento de WinGA se puede dividir en diferentes etapas:

1. Seleccionar la función

Esto implica cargar la biblioteca de la función y mapear a una función de la evaluación.

- **Seleccione la DLL desde Functions/Load del menú.**
  Se muestra la lista de bibliotecas disponibles.

- **Doble click sobre Example.DLL.**
  Se muestra la lista de funciones disponibles dentro de esta DLL

- **Doble click sobre la función One Max**
  La función One Max es una función simple que maximiza el número de unos en una secuencia binaria (en este caso reduce al mínimo la cantidad de ceros en la secuencia pero el resultado es igual).

- **Introduzca 100 en el tamaño de cromosoma**
  Esto determina cuán largos son los cromosomas usados por el WinGA, en este caso 100 bits.

2. Configuración

Esto implica el setear varios operadores y parámetros requeridos por WinGA para que funcione correctamente.

- **Seleccione Setup/Selection del menú principal.**
  Se muestra una lista de los operadores de selección disponibles.

- **Doble click sobre Tournament Method**

- **Seleccione Setup/Combination del menú principal.**
  Se muestra una lista de los operadores de cruza/ crossover disponibles.

- **Doble click sobre One Point method**

- **Seleccione Setup/Mutation del menú principal.**
  Se muestra una lista de los operadores de mutación disponibles.

- **Doble click sobre normal mutation method**

- **Seleccione Setup/General del menú principal**
  Se muestra un cuadro de dialogo con los parámetros generales por defecto.
- **Click sobre OK.**

En este punto el WinGA esta listo para funcionar.

3. Pantalla

- **Click sobre Views/Fitness graph en el menú principal.**
  Se vera un gráfico vacío de dos ejes.

- **Click sobre Views/Text stats en el menú principal.**

  Se abrirá una ventana más pequeña que contiene texto. Se puede mover esta ventana a un lado para evitar de ocultar el gráfico principal.

- **Click sobre Run en el menú principal.**

  La barra de estado debe indicar qué está sucediendo y las dos ventanas deben comenzar a cambiar su aspecto. Se verá un resultado similar al que se muestra en la figura A1.1

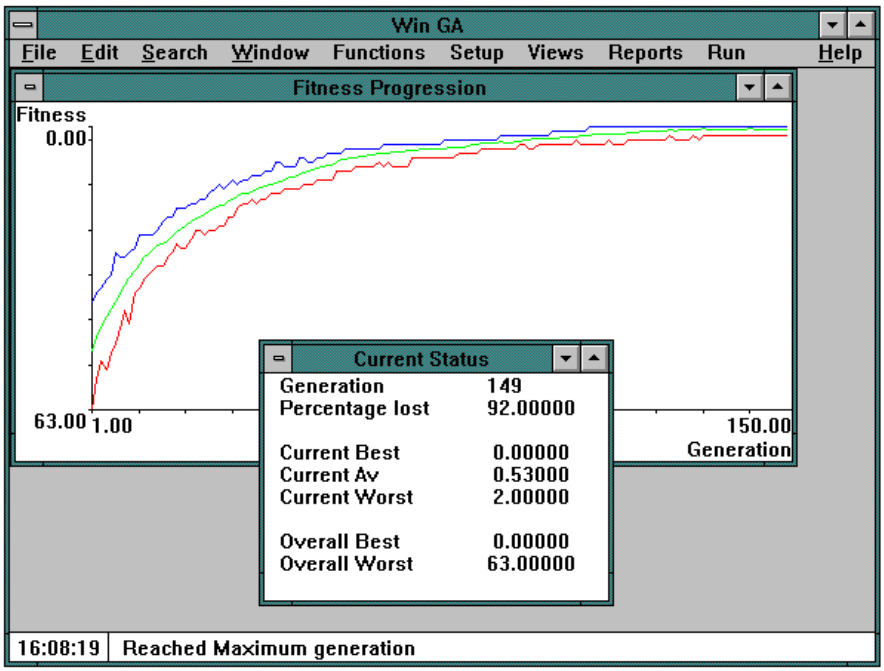


Figura A1.1

Capitulo 3: Aplicación Práctica

Ejemplo: Problema del viajante (TSP del ingles Traveling Salesman Problem)

Definición del problema

Un viajante debe recorrer n ciudades, pasado solamente una vez por cada una y retornar a la ciudad de origen.
El objetivo es minimizar el costo del viaje, entendiéndose por costo total la suma de la distancia a recorrer entre todos los puntos de la ruta escogida.

Espacio de Búsqueda

Permutaciones de n sitios, cada permutación de las n ciudades produce una solución (que es un viaje completo de n sitios) la solución óptima es una permutación que produce el mínimo costo del viaje. El tamaño del espacio de búsqueda es n!

Aproximación de la solución

Justamente en el punto anterior, en el tamaño del espacio de búsqueda, es donde tenemos un problema. Este espacio puede convertirse en un volumen inmanejable cuanto mas grande sea N.

Los algoritmos genéticos permiten encontrar una solución aproximada a este tipo de problema y la solución encontrada puede ser muy cercana a la mejor y, verdaderamente, la mayoría de las veces lo es.

Por ejemplo, si identificamos a cada ciudad con un número (1,2,3...20) y generamos 20 rutas (el número 20 es arbitrario) con los 20 números de las ciudades ordenados de manera aleatoria, tendremos 20 rutas diferentes que hemos generado al azar.

A partir de cada una de las rutas generaremos 10 más: la primera de ellas la dejamos igual que la ruta “padre” (es decir la que hemos usado para generar ésta) y en cada una de las otras 9, permutamos dos ciudades al azar.

Ejemplificando, una de las 20 rutas que fueron generadas aleatoriamente que contiene un orden concreto de visita de las ciudades, podría ser:

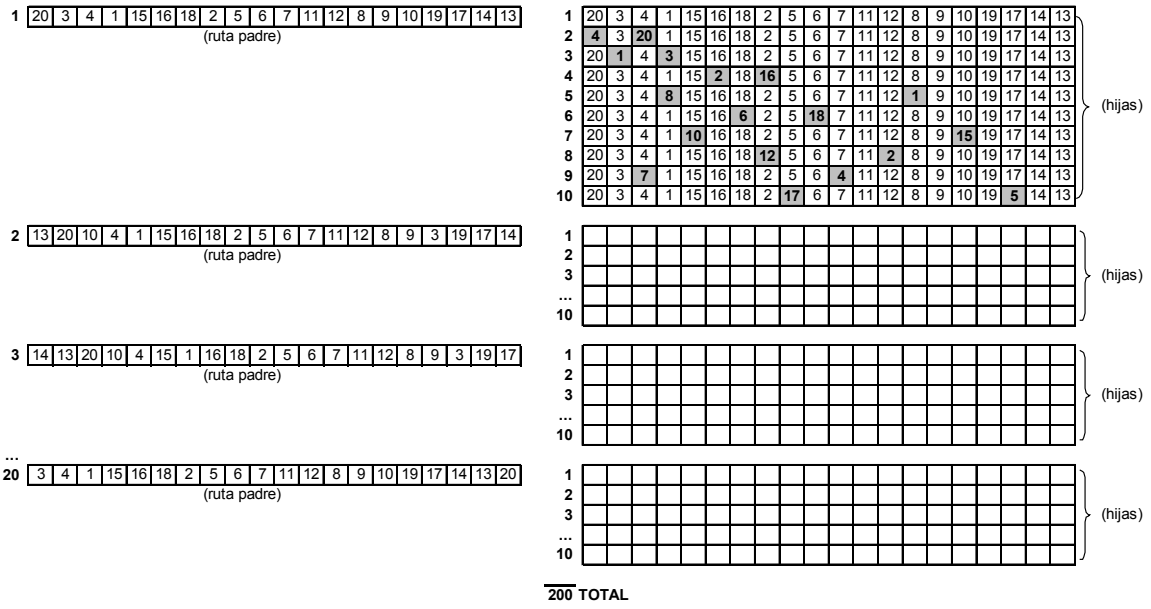
20 3 4 1 15 16 18 2 5 6 7 11 12 8 9 10 19 17 14 13

entonces la primera de las 10 rutas que llamaremos “hijas” será ella misma y a la segunda le permutamos 2 elementos (por ejemplo la posición del 20 por la del 4), obteniendo así una ruta diferente.

Básicamente, esto implica una nueva manera de recorrer las ciudades.

Debemos entonces repetir el proceso para las restantes 19 rutas “padres”. Con lo cual, si teníamos 20 rutas “padres”, de cada una hemos generado 10 rutas “hijas” diferentes, tendremos ahora un total de 200 rutas diferentes.

El siguiente esquema pretende mostrar gráficamente la explicación anterior:



Obtenidas estas 200 rutas, estamos en condiciones de calcular la distancia que supondría recorrer las ciudades. Por lo tanto, sólo haremos 200 operaciones frente a las 2.432.902.008.176.640.000 que hubiésemos tenido que hacer si hubiéramos evaluado todas las posibilidades.

De los resultados obtenidos elegiremos las 20 mejores, esto es, nos quedaremos con las rutas que generaban las 20 maneras más rentables (en distancia) de realizar el viaje.

Con estas 20 repetiremos el proceso de generar otras 200, obtendremos nuevamente sus 20 mejores y podemos repetir esta operación (es lo que se llama el algoritmo genético) todas las veces que queramos.

Se deduce, entonces, que en cada reproducción y selección de las 20 mejores rutas, se obtienen mejores resultados.

Al finalizar el programa, de la selección de las 20 mejores, solamente elegimos la mejor, estableciéndola como resultado final. Por lo tanto, será nuestra aproximación a la solución.

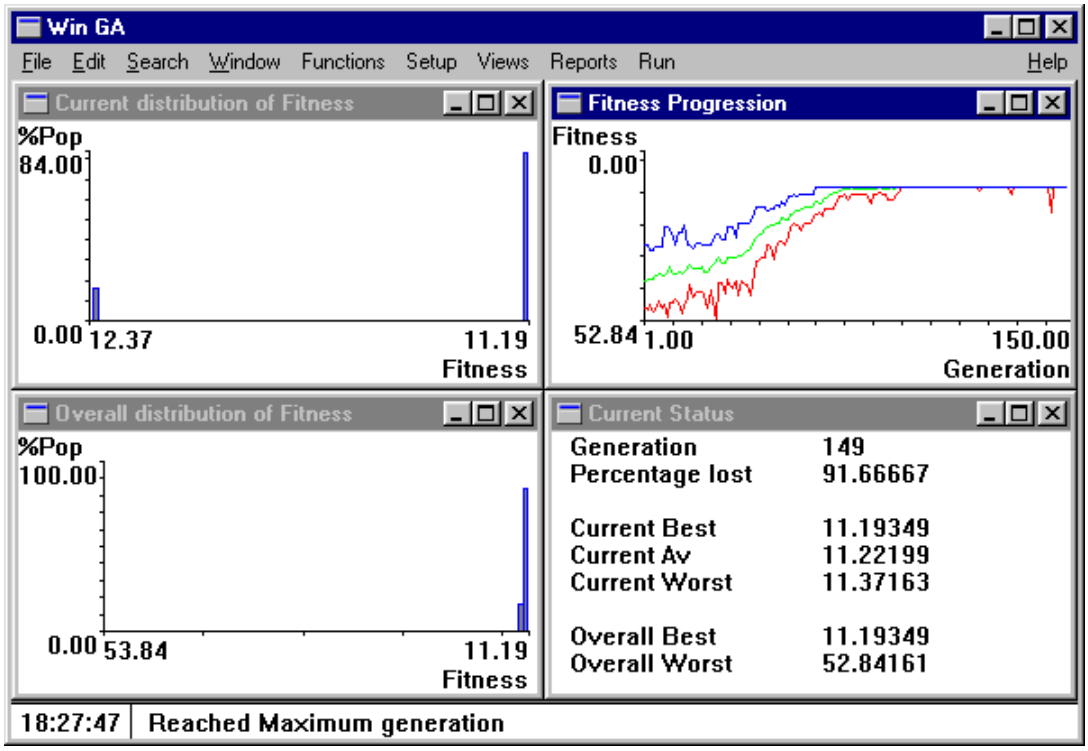
Por qué se denomina aproximación? Justamente porque hemos encontrado soluciones que en un principio eran malas pero a medida que se reiteraban los procesos se iban generando otras soluciones mejores. Luego de un cierto número de repeticiones del algoritmo genético obtendremos una solución razonablemente buena.

Usando el programa Winga

La aplicación Winga o Gwin2 tiene implementada la función de evaluación para el problema del viajante, tomando 25 ciudades.
Para ejecutarla se debe seleccionar la dll master1.dll y la función Krolak's 100 City TSP
Con esta aplicación se puede observar el efecto de los diferentes parámetros sobre el algoritmo genético. Por ejemplo como la mutación impacta sobre la diversidad de la población o como un método de selección puede ser más efectivo que otro.

A continuación se presentan algunos casos.

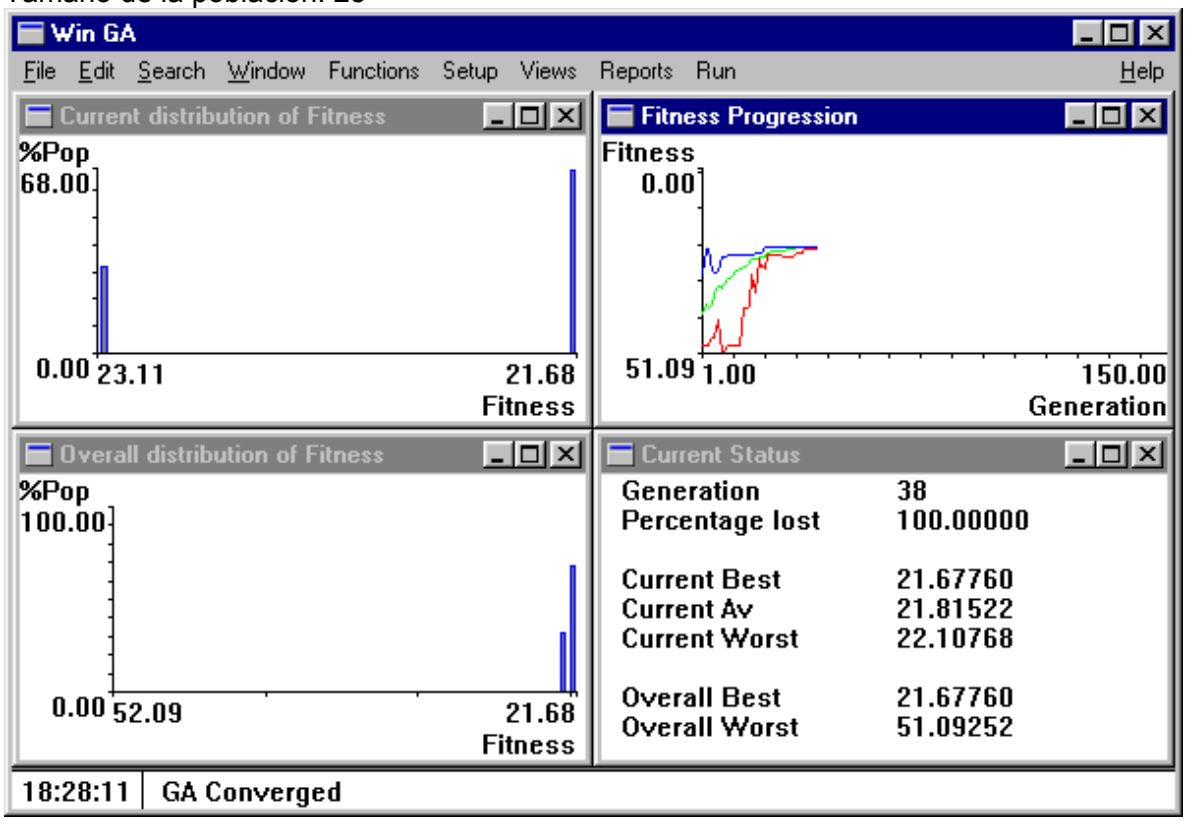
Caso 1:
Tipo de Selección: Método Estocástico
Tipo de Cruza: Cruza Simple Tasa: 0.85000
Tasa de mutación: 0.00001
Tamaño de la población: 25



Resultado: El algoritmo termina al alcanzar el numero máximo de generaciones.

Caso 2:

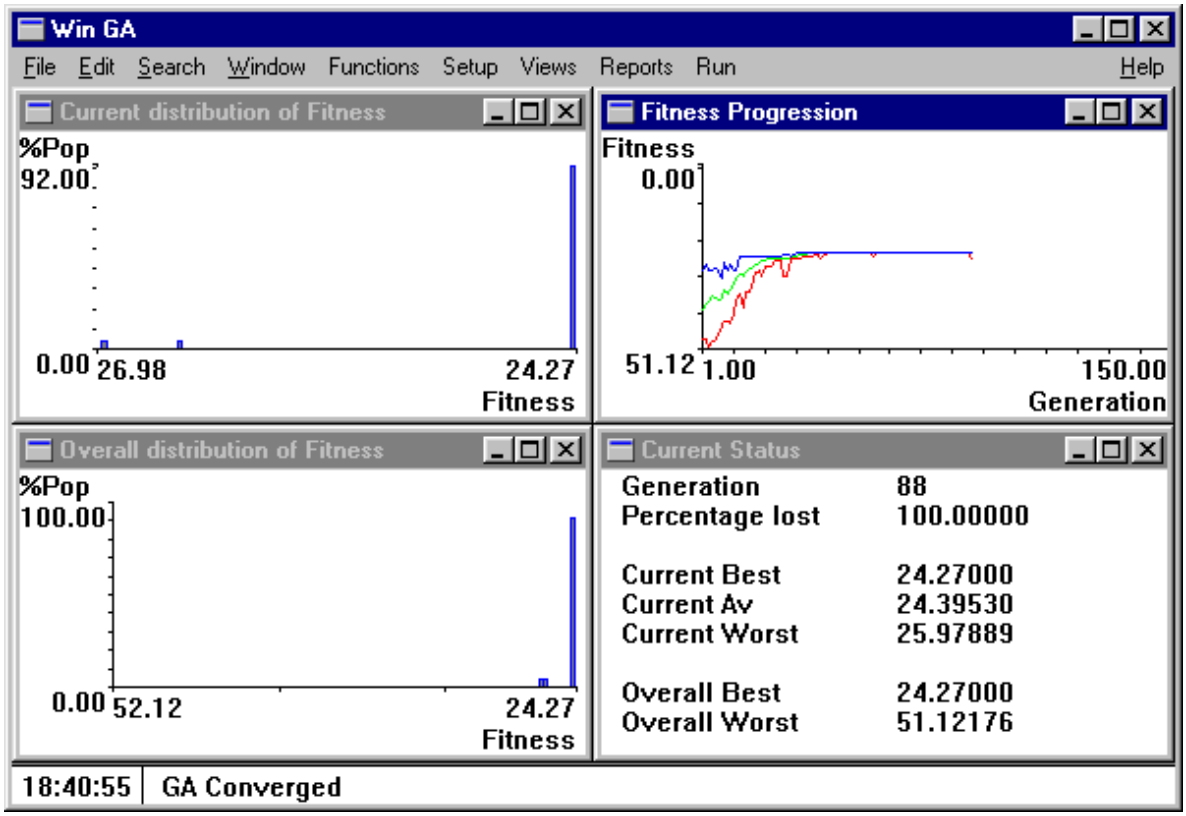
Tipo de Selección: Torneo
Tipo de Cruza: Cruza Simple Tasa: 0.85000
Tasa de mutación: 0.00001
Tamaño de la población: 25



Resultado: El algoritmo converge en la generación: 38. En la selección por Torneo siempre se seleccionan los mejores, sin embargo, en la estocástica pueden desaparecer de la población.

Caso 3:

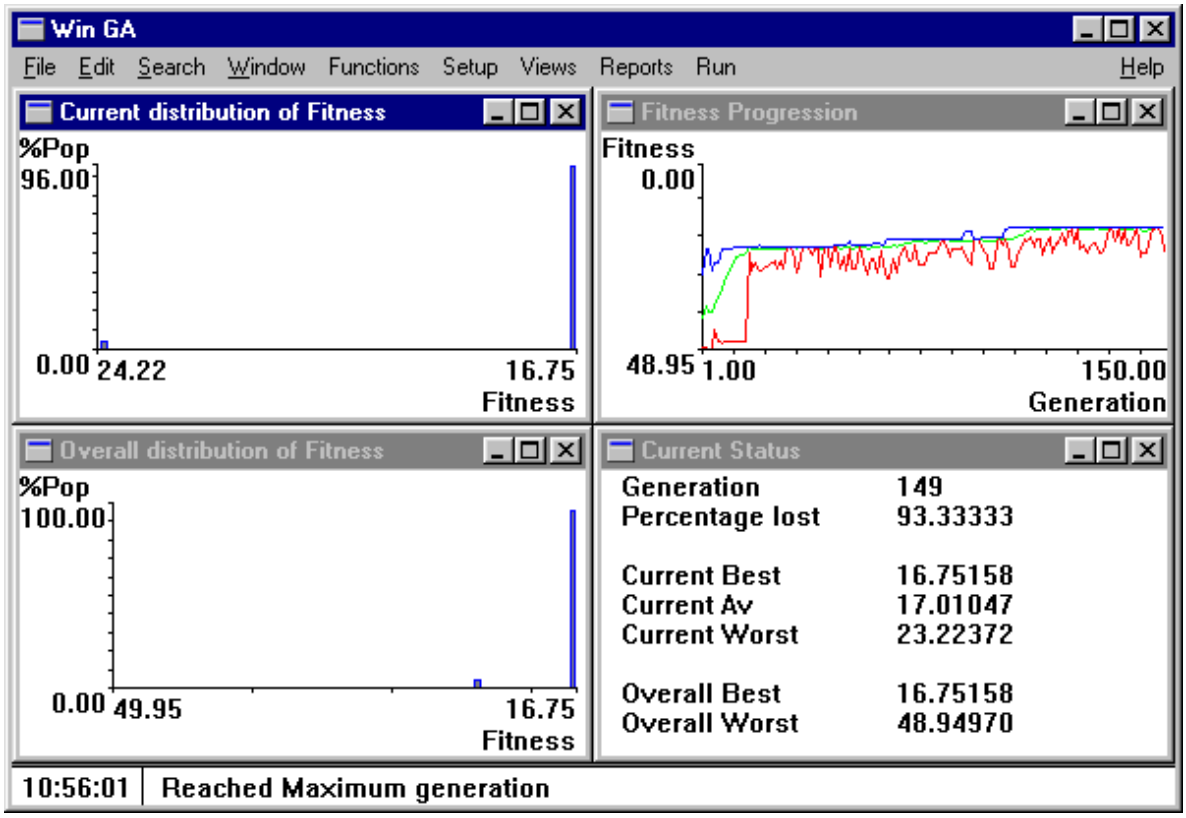
Tipo de Selección: Torneo
Tipo de Cruza: Cruza de dos puntos Tasa: 0.85000
Tasa de mutación: 0.00001
Tamaño de la población: 25



Resultado: El algoritmo converge en la generación: 88. La cruza de dos punto tiene un efecto negativo, ya que obliga al AG a ejecutarse mayor cantidad de veces para encontrar la solución.

Caso 4:

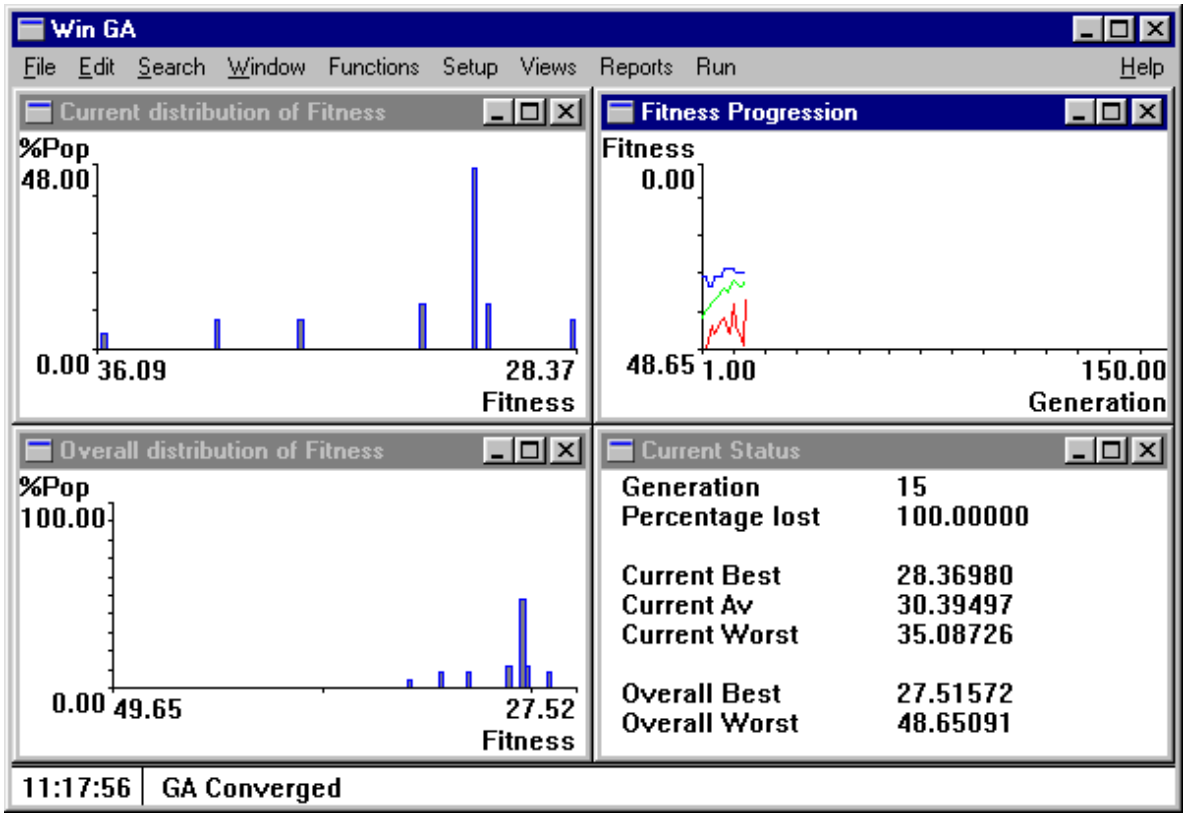
Tipo de Selección: Torneo
Tipo de Cruza: Cruza de dos puntos Tasa: 0.85000
Tasa de mutación: 0.0005
Tamaño de la población: 25



Resultado: Al aumentar la tasa de mutación usando el valor por default que propone la aplicación, el algoritmo no converge y termina al alcanzar el numero máximo de generaciones.

Caso 5:

Tipo de Selección: Torneo
Tipo de Cruza: Cruza de dos puntos Tasa: 0.50
Tasa de mutación: 0.00001
Tamaño de la población: 25



Resultado: Tomando el caso 3 y bajando la tasa de cruza el algoritmo logro una solución en solo 15 iteraciones.

Capitulo 4: Conclusiones

El programa Winga permite explorar con facilidad el impacto de los distintos operadores y su configuración en el obtención del resultado. Los gráficos muestran claramente la evolución del algoritmo, permitiendo al usuario comparar los resultados obtenidos en diferentes ejecuciones.
Al permitir ver los efectos de los distintos parámetros sobre la ejecución de un algoritmo genético simple, resulta ser una herramienta muy útil para enseñanza de los AG.

Capitulo 5: Bibliografía

[RGM] - Dr. Ramón García Martínez- “Algoritmos Genéticos” Unidad 36 del Magíster en Ing del Software de la Facultad Politécnica de Madrid y el Instituto Tecnológico de Buenos Aires.

[JJMG] – Dr. Juan Julián Merelo Guervós Prof Titular de la Universidad de Granada
“ Informática evolutiva: Algoritmos genéticos”.

[DEJ/92]. Kenneth A. De Jong, Williatn M. Spears "A Formal Analysis of the Role of Multi-Point Crossover in Genetic Algorithms" Annals of Mathematics and Artificial Intelligence Joumal, Vol 5, No. 1, 1992