INSTITUTO TECNOLOGICO
de nuevo laredo

" Con la Ciencia por la Humanidad "

INTEGRANTES DEL EQUIPO:

- 01100186 HÉCTOR COSS GARCIA
- 01100240 ENRIQUE JARA PEQUEÑO
- 01100298 DAMIAN PEREA CENICEROS
- 01100285 FILIBERTO QUEBEDO CORTAZAR

TRABAJO: ALGORITMOS GENETICOS (EXPOSICION – EQUIPO 5)

MATERIA: INTELIGENCIA ARTIFICIAL

MAESTRO: ING. BRUNO LÓPEZ TAKEYAS

CARRERA: ING. EN SISTEMAS COMPUTACIONALES

Nuevo Laredo, Tamaulipas a 2 de septiembre de 2005

Algoritmos Genéticos

INDICE

Introducción..... 1
Definición..... 1
Forma de Proceder..... 2
Ventajas y Desventajas..... 3
Limitaciones..... 4
Como Saber si es Posible usar un Algoritmo Genético..... 5
Nacimiento de los Algoritmos Genéticos..... 6
Codificación de los Datos..... 8
Selección..... 11
Cross-Over..... 12
Mutación..... 13
Aplicaciones Reales de los Algoritmos Genéticos..... 14
Acústica..... 14
Ingeniería Aeroespacial..... 14
Astronomía y Astrofísica..... 14
Química..... 14
Ingeniería Eléctrica..... 14
Mercados Financieros..... 15
Juegos..... 15
Geofísica..... 15
Ingeniería de Materiales..... 15
Matemáticas y Algoritmia..... 15
Ingeniería de Sistemas..... 15
Aplicaciones de Búsqueda y Optimización..... 15
Aprendizaje Automático..... 15
Bibliografía y Anexos..... 16

Introducción

Definición

Los Algoritmos Genéticos (AGs) son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización.
Están basados en el proceso genético de los organismos vivos. 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. Por imitación de este proceso, los Algoritmos Genéticos son capaces de ir creando soluciones para problemas del mundo real. La evolución de dichas soluciones hacia valores óptimos del problema depende en buena medida de una adecuada codificación de las mismas. Un algoritmo genético 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.

Una definición bastante completa de un algoritmo genético es la propuesta por John Koza:

((Es un algoritmo matemático altamente paralelo que transforma un conjunto de objetos matemáticos individuales con respecto al tiempo usando operaciones modeladas de acuerdo al principio Darwiniano de reproducción y supervivencia del mías apto, y tras haberse presentado de forma natural una serie de operaciones genéticas de entre las que destaca la recombinación sexual. Cada uno de estos objetos matemáticos suele ser una cadena de caracteres (letras o números) de longitud fija que se ajusta al modelo de las cadenas de cromosomas, y se les asocia con una cierta función matemática que refleja su aptitud. *))*

Versiones mas complejas de algoritmos genéticos generan un ciclo iterativo que directamente toma a la especie (el total de los ejemplares) y crea una nueva generación que reemplaza a la antigua una cantidad de veces determinada por su propio diseño. Una de sus características principales 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 en los que se requiere una gran interacción humana para definir el modus operandi del programa.

Forma de proceder

Dado un problema especifico a resolver, la entrada del AG es un conjunto de soluciones potenciales a ese problema, codificadas de alguna manera, y una métrica llamada función de aptitud que permite evaluar cuantitativamente a cada candidata. Estas candidatas pueden ser soluciones que ya se sabe que funcionan, con el objetivo de que el AG las mejore, no obstante de forma experimental se suelen generar aleatoriamente.

Luego el AG evalúa cada candidata de acuerdo con la función de aptitud. En un acervo de candidatas generadas aleatoriamente, por supuesto, la mayoría no funcionarían en absoluto, y serán eliminadas. Sin embargo, por puro azar, unas pocas pueden ser prometedoras (pueden mostrar actividad), aunque sólo sea actividad débil e imperfecta, hacia la solución del problema.

Estas candidatas prometedoras se conservan y se les permite reproducirse. Se realizan múltiples copias de ellas, pero las copias no son perfectas; se introducen cambios aleatorios durante el proceso de copia. Luego, esta descendencia digital prosigue con la siguiente generación, formando un nuevo acervo de soluciones candidatas, y son sometidas a una ronda de evaluación de aptitud.

Las candidatas que han empeorado o no han mejorado con los cambios en su código son eliminadas de nuevo; pero, de nuevo, por puro azar, las variaciones aleatorias introducidas en la población pueden haber mejorado a algunos individuos, convirtiéndolos en mejores soluciones del problema, más completas o más eficientes. De nuevo, se seleccionan 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 incrementara en cada ronda y, por tanto, repitiendo este proceso cientos o miles de rondas, pueden descubrirse soluciones buenas del problema.

Ventajas y Desventajas

- No necesitan conocimientos específicos sobre el problema que intentan resolver.

- Operan de forma simultánea con varias soluciones, en vez de trabajar de forma secuencial como las técnicas tradicionales.

- Cuando se usan para problemas de optimización (maximizar una función objetivo) resultan menos afectados por los máximos locales (falsas soluciones) que las técnicas tradicionales.

- Resulta sumamente fácil ejecutarlos en las modernas arquitecturas masivamente paralelas.

- Usan operadores probabilisticos, en vez de los típicos operadores deterministicos de las otras técnicas.

- Pueden tardar mucho en converger, o no converger en absoluto, dependiendo en cierta
- medida de los parámetros que se utilicen tamaño de la población, numero de generaciones, etc.

- Pueden converger prematuramente debido a una serie de problemas de diversa índole.

Limitaciones

El poder de los Algoritmos Genéticos proviene del hecho de que se trata de una técnica robusta, y pueden tratar con sexito una gran variedad de problemas provenientes de diferentes careas, incluyendo aquellos en los que otros métodos encuentran dificultades. Si bien no se garantiza que el Algoritmo Genético encuentre la solución óptima, del problema, existe evidencia empírica de que se encuentran soluciones de un nivel aceptable, en un tiempo competitivo con el resto de algoritmos de optimización combinatoria. En el caso de que existan técnicas especializadas para resolver un determinado problema, lo más probable es que superen al Algoritmo Genético, tanto en rapidez como en eficacia. El gran campo de aplicación de los Algoritmos Genéticos se relaciona con aquellos problemas para los cuales no existen técnicas especializadas. Incluso en el caso en que dichas técnicas existan, y funcionen bien, pueden efectuarse mejoras de las mismas hibridándolas con los Algoritmos Genéticos.

Como Saber si es Posible usar un Algoritmo Genético

La aplicación más común de los algoritmos genéticos ha sido la solución de problemas de optimización, en donde han mostrado ser muy eficientes y confiables. Sin embargo, no todos los problemas pudieran ser apropiados para la técnica, y se recomienda en general tomar en cuenta las siguientes características del mismo antes de intentar usarla:

- Su espacio de búsqueda (i.e., sus posibles soluciones) debe estar delimitado dentro de un cierto rango.

- Debe poderse definir una función de aptitud que nos indique qué tan buena o mala es una cierta respuesta.

- Las soluciones deben codificarse de una forma que resulte relativamente fácil de implementar en la computadora.

El primer punto es muy importante, y lo más recomendable es intentar resolver problemas que tengan espacios de búsqueda discretos aunque estos sean muy grandes. Sin embargo, también podrá intentarse usar la técnica con espacios de búsqueda continuos, pero preferentemente cuando exista un rango de soluciones relativamente pequeño.

La función de aptitud no es más que la función objetivo de nuestro problema de optimización. El algoritmo genético únicamente maximiza, pero la minimización puede realizarse fácilmente utilizando el reciproco de la función maximizante (debe cuidarse, por supuesto, que el recíproco de la función no genere una división por cero). Una característica que debe tener esta función es que tiene ser capaz de "castigar" a las malas soluciones, y de "premiar" a las buenas, de forma que sean estas últimas las que se propaguen con mayor rapidez.

La codificación más común de las soluciones es a través de cadenas binarias, aunque se han utilizado también números reales y letras. El primero de estos esquemas ha gozado de mucha popularidad debido a que es el que propuso originalmente.

Nacimiento de los algoritmos genéticos

El algoritmo genético es una técnica de búsqueda basada en la teoría de la evolución de Darwin, que ha cobrado tremenda popularidad en todo el mundo durante los últimos años.

La comunidad científica internacional ha mostrado un creciente interés en una nueva técnica de búsqueda basada en la teoría de la evolución y que se conoce como el algoritmo genético. Esta técnica se basa en los mecanismos de selección que utiliza la naturaleza, de acuerdo a los cuales los individuos más aptos de una población son los que sobreviven, al adaptarse más fácilmente a los cambios que se producen en su entorno. Hoy en día se sabe que estos cambios se efectúan en los genes de un individuo (unidad básica de codificación de cada uno de los atributos de un ser vivo), y que sus atributos mas deseables (i.e., los que le permiten adaptarse mejor a su entorno) se transmiten a sus descendientes cuando este se reproduce sexualmente.

Un investigador de la Universidad de Michigan llamado John Holland era consciente de la importancia de la selección natural, y a fines de los años 60 desarrolló una técnica que permitió incorporarla a un programa. John Holland desde pequeño, se preguntaba como logra la naturaleza, crear seres cada vez mas perfectos[1]. Lo curioso era que todo se lleva a cabo a base de interacciones locales entre individuos, y entre estos y lo que les rodea. No sabía la respuesta, pero tenía una cierta idea de como hallarla: tratando de hacer pequeños modelos de la naturaleza, que tuvieran alguna de sus características, y ver cómo funcionaban, para luego extrapolar sus conclusiones a la totalidad.

En los años 50 entró en contacto con los primeros ordenadores, donde pudo llevar a cabo algunas de sus ideas, aunque no se encontró con un ambiente intelectual fértil para propagarlas.

Fue a principios de los 60, en la Universidad de Michigan en Ann Arbor, donde, dentro del grupo Logic of Computers, sus ideas comenzaron a desarrollarse y a dar frutos. Y fue, además, leyendo un libro escrito por un biólogo evolucionista, R. A. Fisher, titulado "La teoría genética de la selección natural", como comenzó a descubrir los medios de llevar a cabo sus propósitos de comprensión de la naturaleza. De ese libro aprendió que la evolución era una forma de adaptación más potente que el simple aprendizaje, y tomó la decisión de aplicar estas ideas para desarrollar programas bien adaptados para un fin determinado.

En esa universidad, Holland impartía un curso titulado Teoría de sistemas adaptativos. Dentro de este curso, y con una participación activa por parte de sus estudiantes, fue donde se crearon las ideas que mas tarde se convertirían en los algoritmos genéticos.

Por tanto, cuando Holland se enfrentó a los algoritmos genéticos, los objetivos de su investigación fueron dos:

- Imitar los procesos adaptativos de los sistemas naturales, y

- diseñar sistemas artificiales (normalmente programas) que retengan los mecanismos importantes de los sistemas naturales.

Unos 15 años más adelante, David Goldberg conoció a Holland, y se convirtió en su estudiante. Golberg era un ingeniero industrial trabajando en diseñó de pipelines, y fue uno de los primeros que trató de aplicar los algoritmos genéticos a problemas industriales. Aunque Holland trató de disuadirle, porque pensaba que el problema era excesivamente complicado como para aplicarle algoritmos genéticos, Goldberg consiguió lo que quería, escribiendo un algoritmo genético en un ordenador personal Apple II. Estas y otras aplicaciones creadas por estudiantes de Holland convirtieron a los algoritmos genéticos en un campo con base suficiente aceptado para celebrar la primera conferencia en 1985, ICGA'85. Tal conferencia se sigue celebrando bianualmente. Su objetivo era lograr que las computadoras aprendieran por si mismas. A la técnica que inventó Holland se le llamo originalmente "planes reproductivos", pero se hizo popular bajo el nombre "algoritmo genético" tras la publicación de su libro en 1975.

Codificación de los datos

El primer paso para conseguir que un ordenador procese unos datos es conseguir representarlos de una forma apropiada. En primer término para codificar los datos es necesario separar las posibles configuraciones posibles del dominio del problema en un conjunto de estados.

Una vez que obtenida esta clasificación el objetivo es representar cada estado de forma univoca con una cadena de caracteres compuesta (en la mayoría de casos por 1's y 0's). Aunque cada estado puede codificarse con alfabetos de diferente cardinalidad; sin embargo, uno de los resultados fundamentales de la teoría de algoritmos genéticos es el teorema del esquema, que afirma que la codificación óptima es aquella en la que los algoritmos tienen un alfabeto de cardinalidad

El enunciado del teorema del esquema es el siguiente [2]:

Teorema 1 (Teorema del Esquema) *Esquemas cortos, de bajo orden y aptitud superior al promedio reciben un incremento exponencial de representantes en generaciones subsecuentes de un Algoritmo Genético.*

Un ejemplo de una posible población para describir un problema podría ser la siguiente:

Atributo 1	Atributo 2	Atributo 3
101001	010100	000001

Otra de las ventajas de usar un alfabeto binario para la construcción de configuraciones de estados es la sencillez de los operadores utilizados para la modificación de estas. En el caso de que el alfabeto sea binario, los operadores se denominan operadores binarios. Es importante destacar que variables que estén próximas en el espacio del problema deben estarlo en la codificación ya que la proximidad entre ellas condiciona un elemento determinante en la mutación y reproducibilidad de estas, es decir dos estados que en nuestro espacio de estados del universo del problema estén consecutivos preferiblemente deben estarlo en la representación de los datos, esto es para que una vez que haya mutaciones los saltos se den entre estados consecutivos. En términos generales cumplir esta premisa mejora los resultados obtenidos con algoritmos genéticos.

La longitud de las cadenas que representen los posibles estados no es necesario que sea fija, representaciones como la de Kitano para representar operaciones matemáticas son un ejemplo de esto [3].
Experimentalmente el factor que condiciona en mayor grado el fracaso o el éxito de la aplicación de Algoritmos Genéticos a un problema dado es codificación acorde de los datos.

Ejemplo de codificación 1

Si un atributo "tiempo" puede tomar tres valores posibles –despejado, nublado, lluvioso– una manera de representarlo es mediante tres bits de forma que: (Tiempo = Nublado o Lluvioso) y (Viento = Fuerte) se representaría con la siguiente cadena: 01110.

De esta forma podemos representar fácilmente conjunciones de varios atributos para expresar restricciones –precondiciones mediante la concatenación de dichas cadenas de bits.

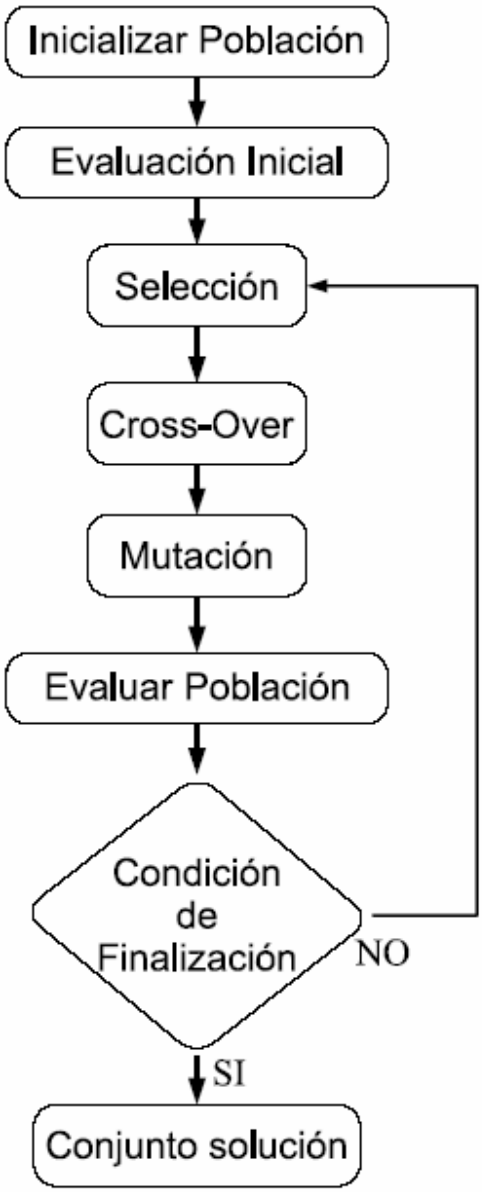
Además si tenemos otro atributo "Viento" que puede ser Fuerte o Moderado, se representaría con la siguiente cadena: 01110.

Ejemplo de codificación 2

"Jugar al Tenis" puede ser Cierto o Falso. Si Viento = Fuerte entonces Jugar al Tenis
= Cierto se representaría mediante la cadena 111 10 10. Donde los tres primeros bits a uno indican que el atributo "Tiempo" no afecta a nuestra regla.

Cabe destacar que una regla del tipo 111 10 11 no tiene demasiado sentido, puesto que no impone restricciones en la post condición. Para solucionar esto, una alternativa es codificar la post condición con un único bit (1 = Cierto y 0 = Falso). Otra opción es condicionar los operadores genéticos para que no produzcan este tipo de cadenas o conseguir que estas hipótesis tengan una adecuación muy baja (según la función de evaluación) para que no logren pasar a la próxima generación de hipótesis.

Un posible esquema que puede representar una posible implementación de algoritmos genéticos podría ser el siguiente:



Selección

La fase de selección elije los estados a reproducirse en la próxima generación, esta selección puede realizarse por muy distintos métodos. En el algoritmo mostrado en seudo código anteriormente el método de selección usado es acorde de la función *fitness* de cada estado. A continuación se describen los más comunes:

Selección elitista: se garantiza la selección de los miembros más aptos de cada generación. (La mayoría de los AGs no utilizan elitismo puro, sino que usan una forma modificada por la que el individuo mejor, o algunos de los mejores, son copiados hacia la siguiente generación en caso de que no surja nada mejor). La forma de seleccionar los mejores individuos es como se ha especificado en el punto anterior mediante el *fitness*.

Selección proporcional a la aptitud: los individuos mas aptos tienen más probabilidad de ser seleccionados, asignándoles una probabilidad de selección más alta. Una vez seleccionadas las probabilidades de selección a cada uno de los individuos se genera una nueva población teniendo en cuenta estas.

Selección por torneo: se eligen subgrupos de individuos de la población, y los miembros de cada subgrupo compiten entre ellos. Sólo se elige a un individuo de cada subgrupo para la reproducción.

Selección por rango: a cada individuo de la población se le asigna un rango numérico basado en su *fitness*, y la selección se basa en este ranking, en lugar de las diferencias absolutas en el *fitness*. La ventaja de este método es que puede evitar que individuos muy aptos ganen dominancia al principio a expensas de los menos aptos, lo que reduciría la diversidad genética de la población y podría obstaculizar la búsqueda de una solución aceptable. Un ejemplo de esto podría ser que al intentar maximizar una función el algoritmo genético convergiera hacía un máximo local que posee un *fitness* mucho mejor que el de sus congéneres de población lo que haría que hubiera una dominancia clara con la consecuente desaparición de los individuos menos aptos (con peor *fitness*).

Selección generacional: la descendencia de los individuos seleccionados en cada generación se convierte en toda la siguiente generación. No se conservan individuos entre las generaciones.

Selección por estado estacionario: la descendencia de los individuos seleccionados en cada generación vuelven al acervo genético preexistente, reemplazando a algunos de los miembros menos aptos de la siguiente generación. Se conservan algunos individuos entre generaciones.

Selección jerárquica: los individuos atraviesan múltiples rondas de selección en cada generación. Las evaluaciones de los primeros niveles son más rápidas y menos discriminatorias, mientras que los que sobreviven hasta niveles más altos son evaluados más rigurosamente.

Cross-Over

Cross-over También llamado recombinación. El *Cross-over* es el operador genético mas utilizado y consiste en el intercambio de material genético entre dos elementos al azar (pueden ser incluso entre el mismo elemento). El material genético se intercambia entre bloques. Gracias a la presión selectiva irán predominando los mejores bloques génicos. Existen diversos tipos de *cross-over* :

- Cross-over uniforme. Se genera un patrón aleatorio en binario, y en los elementos que haya un 1 se realiza intercambio genético.

- Cross-over de n-puntos. Los cromosomas se cortan por n puntos y el resultado de intercambia.

- Cross-over especializados. En ocasiones, el espacio de soluciones no es continuo y hay situaciones que a pesar de que sean factibles de producirse en el gen no lo son en la realidad, por lo que hay que incluir restricciones al realizar la recombinación que impidan la aparición de algunas combinaciones.

Mutación

Este fenómeno, generalmente muy raro en la naturaleza, se modela de la siguiente forma:

Cuando se genera un gen hijo se examinan uno a uno los bits del mismo y se genera un coeficiente aleatorio para cada uno. En el caso de que algún coeficiente supere un cierto umbral se modifica dicho bit. Modificando el umbral podemos variar la probabilidad de la mutación. Las mutaciones son un mecanismo muy interesante por el cual es posible generar nuevos individuos con rasgos distintos a sus predecesores.
Los tipos de mutación mas conocidos son:

- Mutación de bit: existe una única probabilidad de que se produzca una mutación de algún bit. De producirse, el algoritmo toma aleatoriamente un bit, y lo invierte.

- Mutación multibit: cada bit tiene una probabilidad de mutarse o no, que es calculada en cada pasada del operador de mutación multibit.

Mutación de gen[4]: igual que la mutación de bit, solamente que, en vez de cambiar un bit, cambia un gen completo. Puede sumar un valor aleatorio, un valor constante, o introducir un gen aleatorio nuevo.

Mutación multigen: igual que la mutación de multibit, solamente que, en vez de cambiar un conjunto de bits, cambia un conjunto de genes. Puede sumar un valor aleatorio, un valor constante, o introducir un gen aleatorio nuevo.

Mutación de intercambio: Se intercambia el contenido de dos bits/genes aleatoriamente.

Mutación de barajado: existe una probabilidad de que se produzca una mutación. De producirse, toma dos bits o dos genes aleatoriamente y baraja de forma aleatoria los bits o genes, según hubiéramos escogido-comprendidos entre los dos.

CREEP: Este operador aumenta o disminuye en 1 el valor de un gen; sirve para cambiar suavemente y de forma controlada los valores de los genes.

Aplicaciones Reales de los Algoritmos Genéticos

A continuación se citan algunos ejemplos reales en los que la utilización de algoritmos genéticos ha permitido resolver un problema real.

Acústica
Ejemplos de la aplicación de AGs en acústica puede ser el uso de los mismos para el diseño de una sala de conciertos, procurando maximizar la calidad del sonido para la audiencia, la atenuación de ruidos mediante la generación de ondas artificiales destructoras del ruido etc.

Ingeniería Aeroespacial
Diseño de alas de aviones supersónicos, obteniendo una solución de compromiso entre los parámetros fundamentales tales como minimizar la resistencia aerodinámica a velocidades de vuelo supersónicas, minimizar la resistencia a velocidades subsónicas y minimizar la carga aerodinámica (la fuerza que tiende a doblar el ala). Otros ejemplos del uso de AGs en Ingeniería Aeroespacial es el uso de estos para la construcción de brazos mecánicos para operar con satélites en el espacio, o el problema de maximizar las rutas de los satélites de comunicaciones teniendo como objetivo minimizar las situaciones de perdida de cobertura con los centros de comunicación.

Astronomía y astrofísica
Ejemplos de la aplicación de AGs en la Astronomía y Astrofísica pueden ser problemas tales como obtener la curva de rotación de una galaxia basándose en las velocidades rotacionales observadas de sus componentes, determinar el periodo de pulsación de una estrella variable basándose en series de datos temporales, y sacar los valores de los parámetros críticos de un modelo magneto hidrodinámico del viento solar. Son tres difíciles problemas no lineales y multidimensionales.

Química
Calculo de pulsos de láser para la destrucción de moléculas y formación de elementos simples. Desarrollo de nuevos aminoácidos y proteínas mediante la llamada *química combinatoria*.

Ingeniería eléctrica
Un ejemplo clásico y el primer caso de que registrara una patente de un invento desarrollado por una maquina fue el desarrollo de una antena mediante Algoritmos Genéticos para su aplicación en un ámbito muy concreto. El desarrollo de antenas eficientes no es en absoluto una tarea fácil y requiere no solo aplicación de cálculos muy complicados sino que además requiere de varios intentos para aproximar una antena adecuada. La resolución del problema (que seria encontrar la antena óptima), actualmente es mucho menos que imposible.

Mercados financieros
Un problema recurrente en cualquier técnica de inteligencia artificial es la modelización de mercados financieros para realizar previsiones y evaluaciones. Cálculos de los tipos de cambio y de acciones bursátiles mediante algoritmos genéticos son ejemplos claros de la aplicación de los AGs a la industria.

Juegos
Los juegos siempre han sido la base de experimentación de muchos de los métodos en inteligencia artificial. En los algoritmos genéticos esta política se sigue cumpliendo.

Geofísica
Búsqueda de hipocentros[5] a partir de las mediciones de un terremoto.

Ingeniería de materiales
Diseñó de nuevos polímeros útiles para el transito de la corriente eléctrica basados en el carbono (mas conocidos como *polianilinas*). Otro ejemplo de AGs es su uso para el diseñó de patrones de exposición para un haz de electrones de litografía para la construcción de estructuras micrométricas en circuitos integrados.

Matemáticas y algoritmia
Uso de AGs para resolver ecuaciones de derivadas parciales no lineales de alto orden, resolución de problemas de ordenación etc.

Ingeniería de sistemas
Diseñó de circuitos electrónicos, control de de un sistema de frenos de un coche, control de horarios, etc.

Aplicaciones de búsqueda y optimización
Desde aplicaciones evidentes, como la biología o la medicina, hasta otros campos como la industria (clasificación de piezas en cadenas de montaje). Los algoritmos genéticos poseen un importante papel en este ámbito.

Aprendizaje automático
La capacidad que poseen para favorecer a los individuos que se acercan al objetivo, a costa de los que no lo hacen, consigue una nueva generación con mejores reglas y, por lo tanto el sistema será capaz de ir aprendiendo a conseguir mejores resultados. Es aquí donde encuentran un estupendo marco de trabajo los AGs.

Bibliografía

www.elrincondelvago.com
http://geneura.ugr.es/~jmerelo/ie/ags.htm
http://es.wikipedia.org/wiki/Algoritmos_gen%C3%A9ticos
http://the-geek.org/docs/algen/
http://casa.ccp.servidores.net/genetico.html
http://www.itnuevolaredo.edu.mx/takeyas/

Anexos

Seminario: algoritmos Genéticos
Jorge Martín Martín jorgmar@lab.fi.uva.es
Diego Garcia Morate diego.garcia.morate@gmail.com
Curso 2004-2005

Programa de Algoritmos Genéticos WinGA