Universidad Don Bosco

Facultad de Ingeniería

"Estudio monográfico sobre algoritmos genéticos y su implementación práctica"

Presentado por

Francisco Javier Zamora Santana

Asesor

Ingeniero Eduardo Rivera

Agosto 2006

El Salvador, Centro América

Índice

Introducción

Desde sus orígenes el ser humano ha tenido que superar dificultades de acuerdo a las condiciones bajo las que se encuentra. Está en su misma naturaleza buscar e idear soluciones a los diversos problemas que puede enfrentar. Conforme encuentra soluciones y desarrolla nuevas tecnologías, cada obstáculo se convierte en un nuevo reto que involucra un análisis mas profundo. Actualmente los avances en la ciencia y la tecnología ponen a disposición del hombre mucha información y, especialmente en la ingeniería, la solución de problemas requiere de estudios cada vez más complejos.

Una estrategia común del hombre a lo largo de la historia ha sido la búsqueda de soluciones en su entorno natural. Se dice que el desarrollo tecnológico inició una vez este comenzó a vivir en grupo, es decir cuando se formaron las sociedades primitivas. Los expertos afirman que fue gracias a la convivencia grupal lo que dio la oportunidad para el desarrollo de un lenguaje, la forma más primitiva de comunicación se logró gracias a que los miembros del grupo ensayaban imitando a los demás.

Desde las comunidades primitivas hasta el día de hoy, el ser humano ha podido perfeccionar soluciones funcionales aprendiendo del comportamiento a sus alrededores, encontrando una aplicación útil para su beneficio y descubriendo formas de cómo mejorar procedimientos y corregir errores; esto es lo que favorece su desarrollo. La imitación y el aprendizaje permiten al hombre avanzar en cognición y enfocarse en nuevos retos, pero aún cuando los problemas son diferentes y las variables a considerarse cambian, la técnica continua siendo siempre valida, pues diversos descubrimientos continuarán expandiendo los conocimientos hacia nuevos horizontes.

La cantidad de aptitudes que el hombre ha logrado acumular a través de los años es impresionante y por ser un desarrollo progresivo muchas veces podría parecer que no hay campo de la ciencia del cual no se hayan realizado estudios previos. Sin embargo, conforme nuevas tecnologías se implementan y el pensamiento humano evoluciona, los problemas se vuelven más

complicados, por lo que requieren de soluciones más exactas, eficientes e involucran el uso de herramientas más avanzadas.

Seguramente el reto más grande para todo científico se encuentra en si mismo. El mismo pensamiento humano ha sido siempre un misterio para el hombre y conforme se estudia su razonamiento surgen nuevas formas de enfocar los problemas. Muchos científicos[i] estiman que lo más avanzado en el conocimiento es la inteligencia artificial y su dominio ha llegado a ser considerado como la meta última del desarrollo tecnológico. Sin embargo en la actualidad todavía no se ha podido simular un método de aprendizaje que pueda ser comparado con la eficiencia de un individuo promedio.

La inteligencia artificial básicamente tiene como finalidad emular el funcionamiento del cerebro del hombre en una máquina o computadora para que esta posea la habilidad de aprender y pensar de la misma forma que un ser humano. Si se alcanzara dicho objetivo muchas de las actividades que requieren de la supervisión y criterio del hombre podrían ser automatizadas y probablemente surgirían nuevas perspectivas para la resolución de problemas.

La búsqueda intensa por simular comportamientos humanos, aprendizaje y toma de decisiones ha llevado a desarrollar áreas de estudio como las redes neuronales, el aprendizaje de máquina y la computación evolutiva. Siendo los algoritmos genéticos los más representativos de esta última área mencionada.

La computación evolutiva es en términos generales una técnica de procesamiento de información que esta basada hasta cierto grado en la evolución de la vida biológica dentro del mundo natural. Su importancia radica en su utilidad y para lograr comprender claramente las oportunidades que esta presenta es necesario comprender claramente como evolucionan los organismos vivos.

Los primeros capítulos de este documento contienen información relacionada con este tipo de problemas y las bases biológicas de los algoritmos genéticos. Finalmente se describe el

[i] Revista SCIENTIFIC AMERICAN Volumen 294 Número 6- . Junio 2006

desarrollo de una aplicación desarrollada con el propósito de ejemplificar este tipo de aplicaciones.

Capitulo I

Descripción del proyecto

1.1 Importancia de la investigación

Esta investigación podrá ser de gran utilidad al incorporar técnicas de programación relacionadas con la inteligencia artificial a la solución de problemas. Una de las áreas de gran importancia dentro de la inteligencia artificial son los algoritmos genéticos y esta tesis podrá ser utilizada posteriormente para aplicarlos a la solución de problemas.

1.1.1 Planteamiento del problema a resolver

Actualmente en nuestro medio existe muy poca bibliografía que permita asistir los estudios de los algoritmos genéticos y su implementación en la programación. La mayor parte de la información disponible se encuentra en otros idiomas o está polarizada entre la teoría y la práctica. Por una parte se encuentran documentos muy científicos que dificultan su comprensión o uso práctico y por otro lado se encuentran los documentos que describen sorprendentes aplicaciones de los algoritmos genéticos pero que para discernirlos se necesita tener un amplio conocimiento previo de la materia.

Esencialmente el proyecto planteado busca desarrollar un documento bibliográfico que permita comprender los algoritmos genéticos desde una perspectiva práctica, para este fin se auxiliará del desarrollo una aplicación que, a pesar de utilizar información ficticia para servir de ejemplo, esta orientada a resolver un problema muy común en el país.

1.1.2 Definición del tema

El tema para la tesis de graduación se ha definido como “Estudio monográfico sobre algoritmos genéticos y su implementación práctica”. El proyecto consistirá en realizar una investigación bibliográfica enfocada a la implementación de la teoría en la solución de problemas prácticos. A continuación se describe mas detalladamente lo que formará parte de esta investigación y como pretende realizarse.

Como punto de partida se estudiará información relevante de otras fuentes bibliográficas adaptándola a un modelo del documento diseñado con el propósito de guiar al lector hacia una aplicación práctica. Además de exponer los conceptos y elementos involucrados en el uso de estos algoritmos, en esta tesis también explicará como solucionar un problema incorporándolos en la programación de sistemas, para esto se hará uso de un ejemplo práctico que será utilizado a lo largo de la investigación para demostrar y aplicar los temas en estudio.

La aplicación que se utilizará como ejemplo será un programa que, con un punto de origen y un punto de destino, utilizará los algoritmos genéticos para determinar la ruta más descongestionada al trasladarse dentro de un mapa diseñado con fines académicos. Este mapa simulará las calles de una ciudad imaginaria y las posibles rutas representarán las diferentes calles en las que un conductor podría transitar.

Esta simulación de una ciudad será diseñada con algunas de las propiedades que se deben considerar al estudiar problemas de tránsito en una ciudad. Para que esta aplicación pueda ser de utilidad en la búsqueda de soluciones a los problemas de tráfico, se utilizarán y se asignarán valores a ciertas variables de tráfico que necesariamente deben ser monitoreadas en una ciudad real.

Este programa, de ser implementado en una ciudad real con datos reales, podría ser de utilidad para que los conductores puedan optar por las rutas menos congestionadas y así reducir los tiempos y costos de transporte. Si en una ciudad todos los conductores pudieran estar constantemente informados del tráfico vehicular, se lograría un mejor balance de vehículos en las calles de la ciudad.

El desarrollo de esta aplicación a lo largo del documento no solo tiene como objetivo mostrar un ejemplo práctico sino también mantener el interés del lector en las diferentes etapas del documento. Como producto final se presentará un documento de utilidad para comprender el origen de estos algoritmos, su funcionamiento y aplicación. Anexo a este se incluirá, como complemento a la teoría, una aplicación práctica de software que exponga los conceptos estudiados en el documento.

1.1.3 Justificación

Los motivos por los cuales se opto por esta investigación fueron:

- La necesidad de mejorar y enfatizar la importancia de la inteligencia artificial en el aprendizaje actual.
- El potencial en la implementación de algoritmos genéticos dentro de los sistemas de información.
- La importancia de mostrar a los estudiantes de ingeniería en sistemas, que a pesar de ser una materia optativa, la inteligencia artificial esta al alcance de todos y debe ser de interés para optimizar procesos en sus aplicaciones.
- La falta de disponibilidad de bibliografía útil para instruirse y aplicar los algoritmos genéticos en la solución de problemas prácticos.
- La necesidad de incentivar la búsqueda de una solución para el problema de aglomeración de vehículos en nuestro país.

Este trabajo de tesis se llevará a cabo teniendo siempre presente los puntos previamente mencionados y buscando ofrecer una solución congruente a los mismos.

El interés por desarrollar este proyecto de tesis proviene de la certeza que en un futuro los algoritmos genéticos tendrán un papel muy importante en la programación cotidiana. Como resultado de este trabajo se espera proporcionar una investigación lo suficientemente completa como para servir de apoyo bibliográfico.

Como se mencionó anteriormente no se busca producir una gran cantidad de información, más bien, se busca crear una guía práctica en la que se describa el funcionamiento de estos algoritmos, sus variaciones e implementación en situaciones reales. Actualmente la mayoría de estudios en este campo se han hecho en otros países y prácticamente todo el material de apoyo para una nueva investigación se encuentra en el idioma inglés. Aun cuando es conveniente que todo estudiante complemente sus investigaciones con otra bibliografía, especialmente escrita en el idioma inglés, este trabajo se podría utilizar como una fuente de información más accesible, es decir como un punto de partida.

Orientado a ser herramienta para el estudio de estos algoritmos, el proyecto de tesis debe ser de utilidad para estudiantes de informática que en algún momento necesiten emplearlos con sus propios proyectos de programación. Con este propósito en mente, la investigación se complementará haciendo referencia al estudio de una aplicación de algoritmos que se llevará a cabo paralelamente.

1.2 Objetivos

1.2.1 Objetivo general

Proporcionar una fuente bibliográfica útil para los estudiantes de ingeniería que, sin tener amplios conocimientos en la inteligencia artificial, estén interesados en instruirse sobre los algoritmos genéticos o que deseen optimizar sus aplicaciones mediante la implementación de los mismos.

1.2.2 Objetivos específicos

- Aportar a la biblioteca de la UDB material de apoyo para el estudio de algoritmos genéticos e inteligencia artificial.
- Asistir a un estudiante, sin conocimientos previos en la materia, proporcionando una visión acertada de los algoritmos genéticos así como su utilidad.
- Presentar los algoritmos genéticos y sus conceptos auxiliándose de una aplicación práctica expuesta para ejemplificar las fases de su desarrollo.
- Desarrollar una simulación sencilla en software para PC que haciendo uso de los algoritmos genéticos demuestre su utilidad en la solución de problemas sociales y sirva como punto de partida para el desarrollo de otras aplicaciones.
- Proveer, con esta investigación, una guía de carácter descriptivo, mostrando datos acompañados de una explicación orientada a su posible utilidad en otras aplicaciones.
- Exponer la forma en que afecta la alteración de las variables genéticas y como se realiza la selección de la configuración más adecuada para un problema específico.

1.3 Alcances

- El material bibliográfico que se pretende documentar incluirá los orígenes de los algoritmos genéticos, los personajes que han estado directamente involucrados en el desarrollo de los mismos, sus avances y actuales aplicaciones. También estará contenida la información necesaria para comprender los procesos y funciones involucrados en el funcionamiento de estos algoritmos.

- El documento será presentado en un formato accesible al estudiante de manera tal que toda la información incluida será relevante en el proceso de desarrollo de aplicaciones en las que se implementen los algoritmos genéticos.

- Los algoritmos genéticos que serán estudiados y utilizados a lo largo de este proyecto estarán basados en los diseñados en la dedada de los 60 por el profesor John H. Holland de Universidad de Michigan en Ann Arbor.

- La bibliografía estará apoyada en el desarrollo de una aplicación demostrativa que será documentada desde proyección hasta su finalización y funcionamiento. Esta aplicación será utilizada para ejemplificar como puede abstraer una codificación adecuada de variables y procesos necesarios para la solución de problemas mediante el uso de algoritmos genéticos.

- La aplicación demostrativa a desarrollar buscará encontrar la solución a un problema de transporte en el cual se busca desplazarse desde un punto dado A hasta otro punto dado B. En la simulación se tomarán en cuenta una serie de parámetros que servirán para simular condiciones de tráfico específicas en una ciudad.

- La aplicación tendrá como finalidad facilitar la comprensión de los algoritmos genéticos y mostrar como la alteración de variables genéticas puede influir en el desempeño de estos algoritmos y su efectividad. Sin embargo, como la aplicación tratará un problema de tráfico vehicular, la solución planteada será lograr una mejor distribución de automóviles en las calles de la ciudad bajo el supuesto de que un alto porcentaje de usuarios utilizarían esta aplicación para evitar transitar en calles congestionadas.

1.4 Limitaciones

A pesar de tener establecido un tiempo prudencial para el desarrollo de la investigación, el período de desarrollo para esta tesis limita en cierta forma los detalles y la decoración de la aplicación práctica. Como se mencionó anteriormente el propósito central de la tesis servir como herramienta de estudio, por lo que la mayor parte del tiempo deberá ser invertido en ejemplificar su utilidad didáctica y no en su decoración.

1.5 Delimitaciones

- Todo material bibliográfico estará enfocado a facilitar el estudio de los algoritmos genéticos y a la comprensión de su funcionamiento por lo que mucha de la información disponible que no sea relevante o que no este acorde con el propósito de este proyecto será desechada.

- La recopilación de material bibliográfico no tiene como propósito generar grandes volúmenes de información. Más bien, busca generar material útil explicado bajo un formato accesible que, apoyándose en ejemplos reales, faciliten una visión práctica de estos algoritmos.

- Tanto la aplicación demostrativa a desarrollar como el documento bibliográfico, tienen como propósito facilitar el estudio de los algoritmos genéticos. Si bien es cierto están estrechamente relacionados, no se busca facilitar la comprensión de la aplicación con el documento ni viceversa, mas bien ambos se complementan para alcanzar el mismo fin.

- Esta aplicación práctica se desarrollará bajo Visual Basic 6.0 para proveer una interfaz gráfica que facilite a los usuarios la comprensión de su funcionamiento. El programa también tendrá la opción de alterar ciertas variables de entrada para demostrar su incidencia en el resultado del algoritmo.

- La aplicación estará limitada a encontrar la ruta más conveniente para trasladarse de un punto a otro en un mapa específicamente diseñado para explicar la codificación y funcionamiento de los algoritmos genéticos. El mapa estará limitado a un radio fijo de aproximadamente dieciséis cuadras, cuyas calles estarán establecidas con un sentido y una cantidad de carriles especifica, no variable.

- Los resultados proporcionados por la aplicación estarán sujetos a los cambios en las variables genéticas. Las condiciones de tráfico sobre las calles podrán ser modificadas de manera que la aplicación mostrará la solución que mejor se adapte al ambiente bajo el cual se ejecuta el algoritmo. Esta solución podrá variar dependiendo de las alteraciones realizadas en las variables genéticas.

- La aplicación no deberá ser tomada como una solución absoluta o una única solución al problema. Deberá ser tomada como una guía para ser estudiada. El problema utilizado de ejemplo podría ser resuelto con otras variaciones y otros parámetros en las variables genéticas. Sin embargo la eficiencia de las variantes en la solución estaría ligada a la

sensibilidad de los parámetros que describan los diferentes segmentos de las calles de la ciudad. Por ser una aplicación con fines demostrativos, no tendría sentido presentar una gran cantidad de pruebas exhaustivas con alteraciones mínimas.

1.6 Metodología de la investigación

Este proyecto de investigación, como se mencionó anteriormente, consiste en hacer una recopilación de información bibliográfica, analizarla y exponerla de forma práctica implementándola paralelamente en una aplicación de software.

En una primera instancia se deberá obtener la información que será estudiada, para esto se utilizará un método de selección analítica bajo el cual se determinará si la bibliografía encontrada es aplicable o no para los objetivos de la tesis.

Este método consiste en consultar diversas de fuentes de información como por ejemplo libros, trabajos de tesis o documentos de investigación ya publicados, disponibles en las bibliotecas universitarias del país. Por otro lado, también se tratarán de explotar al máximo los recursos disponibles en Internet, ya sea en idioma español o inglés. Esta recopilación será la base de la cual se partirá para la elaboración del proyecto.

La información será analizada dentro del contexto de la investigación y se evaluará la utilidad que esta podría presentar para un estudiante de computación. Si la información no demuestra una utilidad concreta en el estudio o se encuentra aplicada dentro de un proceso específico que no sea de interés, la información será descartada.

Una vez determinada la utilidad del material bibliográfico este deberá ser incorporado al trabajo de tesis adaptándolo al formato y al esquema original. Para esto se utilizará una metodología explicativa estructuralista, ya que su finalidad será explicar el porqué y el cómo influye el elemento o concepto del algoritmo en estudio, sin alterar su estructura inherente. En otras

palabras, la información bibliográfica será analizada y adaptada para poder ser explicada, mas no será distorsionada de su concepto original.

A lo largo del desarrollo del proyecto se utilizará principalmente la técnica documental pues estará constantemente refiriéndose a las diferentes fuentes de información. Sin embargo, la investigación no es absolutamente teórica. Se realizará una aplicación práctica enlazada correspondientemente con la teoría, para la cual será necesario aplicar técnicas de campo que faciliten la descripción de los conceptos utilizados en la aplicación.

Capitulo II

Reseña histórica y bases teóricas

2.1 Que son las especies y como se relacionan

Uno de los principios más importantes dentro de la biología es la evolución de las especies, sin embargo este principio también ha demostrado ser de gran utilidad para las ciencias de la computación. Es indispensable tener un claro concepto de lo que es una especie y como se limita antes de comprender como estas se modifican con el tiempo.

La palabra especie proviene del latín *species* y en general es un conjunto de individuos que, además de los caracteres genéricos, tienen en común otros por los cuales se asemejan entre sí y pueden ser distinguidos de individuos pertenecientes a las demás variedades. A lo largo de los años este concepto ha cambiado constantemente, desde los tiempos de Aristóteles ya se establecía que los géneros se dividen en especies pero esta, a su vez, puede convertirse en género con respecto a las subdivisiones secundarias.

Entre las diferentes definiciones podemos encontrar varias citas que muestran las formas en que se maneja el concepto.

"Contamos tantas especies cuantas formas distintas fueron creadas en el principio"[ii].

"Especie es el conjunto de los individuos descendientes uno de otro o de padres comunes y de los que se les parecen tanto como aquellos entre si"[iii].

"Especie es la colección de todos los individuos que se parecen más entre si que a otros; que por fecundación recíproca pueden dar individuos fértiles, y que se reproducen por generación, de tal

[ii] Linneo, "Phylosophya botanica"
[iii] Georges Cuvier (1769-1832)

manera que, por analogía, se les puede suponer a todos procedentes originariamente de un solo individuo"[iv].

"Especie es el conjunto de todos los individuos cualitativamente idénticos que no presentan entre sí, en sus elementos vivos, mas que diferencias cuantitativas"[v].

"Todos los individuos fecundos entre sí y cuyos descendientes son también indefinidamente fecundos"[vi].

A pesar de que la definición de una especie ha sido analizada muchas veces, el verdadero dilema ocurre al momento de limitar que es y hasta donde las características de un individuo se asemejan lo suficiente como para no ser considerado parte de una diferente. La determinación de los límites es puramente subjetiva y, por tanto, expuesta a las modalidades de la interpretación personal. Algunos conceptos usuales son muy antiguos y en ciertas ocasiones anteriores a su establecimiento científico.

Si no se dieran los cambios en las especies, sería muy fácil definir cada una de ellas y sus miembros, estableciendo que grupo de individuos son cualitativamente idénticos. Pero en la realidad una entidad definida de esa forma no es realmente una especie, sino lo que usualmente se llama una línea pura o un clon.

Sin embargo, existen muchas diferencias entre las cualidades de cada miembro de una especie y a su vez existen cambios que como conjunto se llevan a cabo. La clave esta en definirlas como agrupaciones de organismos biológicos que compongan una unidad natural de reproducción. Este es el punto definitivo, originalmente se puede reproducir y engendrar crías que como adultos serán similares a sus padres.

Existen algunas excepciones, pues se da el caso en que dentro de una misma especie se encuentren individuos que no puedan reproducirse entre ellos. Como ejemplo podemos tener a

[iv] Augustin Pyrame de Candolle (Ginebra, suiza, 4 de febrero de 1778– 9 de septiembre de 1841)
[v] Félix le Dantec
[vi] Rennolls, K. and Laumonier, Y. Analysis of species hyperdiversity in the tropical rain forests of Indonesia: the problem of non-observance. Environmental Forest Science. 1998; 54355-362.

los perros, en que las razas de perros grandes no pueden reproducirse con perros pequeños, sin embargo ambas razas forman una misma especie.

Este es uno de los puntos más interesantes al momento de estudiar las especies y su evolución. Así como en los perros pueden existir amplias variaciones que hoy conocemos como razas, también existen otros animales que no permiten diversificaciones. Los chita por ejemplo, solamente se pueden reproducir entre ellos y por tanto solo pueden procrear crías con ninguna o pocas mutaciones.

Las especies, como las que ahora habitan en el planeta, proceden de otras distintas que existieron en el pasado, a través de un proceso de descendencia con modificación. Esto es precisamente la evolución biológica, es el proceso histórico de transformación de unas especies en otras descendientes, e incluye la extinción de la gran mayoría de las que hasta el momento han existido.

Es claro que la evolución biológica es un proceso largo y muchas veces imperceptible. Sin embargo, esto nos indica que no es posible que una especie pueda procrear otra totalmente diferente. Dentro de la población de individuos se pueden originar nuevas subespecies, finalmente cuando una de estas pierde el vinculo característico que la une con las demás se considera una unidad diferente. La idea de evolución por modificación y derivación implica la existencia de antepasados comunes para cualquier par de especies.

2.2 Darwin, Wallace y el origen de las especies

Charles Darwin publicó en 1859 su obra "El origen de las especies" y con esta publicación quedó establecida de forma definitiva la idea de la evolución biológica. Sin embargo los estudios en la biología evolutiva habían producido numerosas teorías, de hecho en el mismo año que Darwin nació se publicaba la obra *filosofía de la evolución*, por Jean-Baptiste Monet, Caballero de Lamarck, autor de la primera teoría organizada de la evolución[vii].

[vii] http://natureduca.iespana.es/bio_teorias_evol.htm

La teoría de Lamarck fue vivamente atacada en su tiempo, hasta el extremo de ser silenciada. El mayor rechazo llegó de los grupos religiosos, que preveían el derrumbe de las explicaciones basadas en la Biblia sobre el origen de los seres vivos. Sin embargo, se mantuvo esta corriente de pensamiento evolucionista, sirviendo de base para lo que terminaría siendo una verdadera revolución en las ideas biológicas del momento, y que desembocaría en los trabajos de Darwin[viii].

El naturalista británico recopiló e interpretó un gran número de observaciones y experimentos de muy diversas disciplinas de investigación y los presentó como un argumento irrefutable en favor del hecho de la evolución. Pero a su vez suministró un mecanismo para explicar las adaptaciones complejas y características de los seres vivos: la selección natural.

Desde 1802 cuando el teólogo W. Paley publica la obra *Teología natural*, donde argumentaba que el diseño funcional de los organismos evidenciaba la existencia de un creador omnisapiente, existía un gran reto para las ideas de biología evolutiva. Él afirmaba, utilizando el ojo humano como ejemplo por su delicado diseño, que únicamente podría haber sido obra directa de Dios. Para los naturalistas que querían explicar los fenómenos biológicos por procesos naturales, explicar la adaptación, la maravillosa adecuación de los organismos a su ambiente, constituía el problema fundamental[ix].

El argumento del diseño de Paley tenía una gran influencia en los naturalistas del siglo XIX, a pesar de que esta visión en la que existía una intervención directa de un creador, violaba incuestionablemente el concepto de naturaleza que se había establecido con el desarrollo de la física en los siglos XVI y XVII. Los fenómenos del Universo, según esta nueva concepción, eran explicables por procesos naturales. La naturaleza, por si sola, era un objeto aceptable para preguntar y contestar científicamente. Con la publicación del estudio de Darwin se introduce esta revolución en la biología. Lo verdaderamente revolucionario en Darwin fue el proponer un mecanismo natural para explicar la génesis, diversidad y adaptación de los organismos[x].

[viii] http://natureduca.iespana.es
[ix] http://biologia.uab.es/divulgacio/sn/sn.htm
[x] http://biologia.uab.es/divulgacio/sn/sn.htm

El gran reto de Darwin era explicar las complejas adaptaciones de los organismos vivos, como el diseño funcional de un ojo, por mecanismos naturales. La solución de Darwin fue proponer el mecanismo de la selección natural. Como se podrá ver mas adelante, este concepto es de gran interés no solo para comprender la evolución biológica sino también la computación evolutiva.

Para obtener las respuestas que buscaba, Darwin se embarcó como naturalista en el velero Beagle, a bordo del cual viajó alrededor del mundo durante cinco años. Allí recogió infinidad de datos de carácter geológico, zoológico y botánico en los que se inspiró para formular sus puntos de vista. De regreso a Inglaterra, en 1837 se instaló en Londres, ocupándose de la redacción de su diario del viaje y de la elaboración de su estudio sobre los arrecifes de coral[xi].

La publicación de los estudios de Darwin requirió ser apresurada por las circunstancias del momento, si bien es cierto se exponía a los ataques de los seguidores de la teoría creacionista, recibió una carta procedente del archipiélago malayo, de Sir Alfred Wallace, un biólogo británico, nacido en Monmouth (hoy Gwent)[xii].

Él se encontraba realizando una investigación y en la expedición observó las diferencias zoológicas fundamentales entre las especies de animales de Asia y las de Australia y estableció la línea divisoria zoológica -conocida como línea de Wallace- entre las islas malayas de Borneo y Célebes. Aquí formuló su teoría de la selección natural que comunicó a Darwin en 1858[xiii].

Finalmente, en 1859, el 24 de Noviembre, a los doce meses de haber recibido el manuscrito de Wallace, publicó su obra "Origin of Species", de la que Wallace recibiría un ejemplar y del cual opinó: *"Perdurará tanto como los Principia de Newton. El señor Darwin ha donado al mundo una ciencia nueva, y su nombre, a juicio mío, se destaca por encima del de muchos filósofos antiguos y modernos. ¡¡La fuerza de la admiración me impide decir más!!"*.

El hecho de que ambos investigadores se interesaran y estuvieran determinados a desarrollar estudios relacionados en diferentes partes del mundo fue una de las coincidencias más

[xi] http://natureduca.iespana.es/biog_darwin.htm
[xii] http://www.terra.es/personal/cxc_9747/transformismo.html
[xiii] http://www.terra.es/personal/cxc_9747/el_origen.html

portentosas de la historia de la ciencia. Refiriéndose a Darwin, Wallace escribió una vez: *"Ni en sueños me hubiera acercado yo a la perfección de su libro. Confieso mi agradecimiento de que no me incumbiera presentar la teoría al mundo"*.

Sin embargo la teoría de Wallace difiere de la de Darwin en algunas cuestiones importantes; por ejemplo, niega que la selección natural sea suficiente para dar cuenta del origen del hombre, lo cual requiere, según Wallace, la intervención divina directa. También creyó que el proceso evolutivo había finalizado en los hombres y que la evolución sería imposible en adelante[xiv].

2.3 La evolución, principio unificador de la biología

Se dice que el concepto de evolución llego a unificar muchos de los diferentes estudios dentro de la biología, al definir apropiadamente la evolución es posible comprender las propiedades que diferencian a los organismos unos de otros y como estos se adaptan al medio en el que habitan. Por medio de la evolución se puede explicar la proximidad existente entre especies diferentes. El genético evolucionista Theodosius Dobzhansky una vez afirmó que *"la teoría evolutiva se relaciona con el resto de la biología de forma análoga a como el estudio de la historia se relaciona con las ciencias sociales"*[xv].

Realmente el concepto de evolución es sencillo pero, posiblemente por lo difícil que fue su aceptación, la mayor parte de las personas tienden a confundir lo que en sí afirma esta teoría. Comúnmente se piensa que las especies pueden ser ordenadas en una cadena evolutiva, empezando desde las bacterias a través de los animales "inferiores", hasta los animales "superiores" y, finalmente, hasta el hombre. En realidad, la idea de una gran cadena del ser, que se remonta a Carolus Linnaeus (1707-78), fue echada por tierra por la idea de Darwin de la descendencia común[xvi].

Las pequeñas, pero muy significativas, equivocaciones de este concepto afectan en gran medida los estudios de la alteración en las especies y de toda la biología en general. Por tanto afectan

[xiv] http://www.terra.es/personal/cxc_9747/el_origen.html
[xv] "Genética y el origen de las especies" publicada en 1937
[xvi] http://bioinformatica.uab.es/divulgacio/evol.html

también los estudios de computación evolutiva que tiene su fundamento en estos mismos enunciados. La palabra evolución tiene una variedad de significados. Frecuentemente se le menciona como el hecho de que todos los organismos estén relacionados a través de la descendencia con un antepasado común. También se le llama de esta forma a la teoría de cómo aparecieron los primeros organismos vivos, lo cual es errado puesto que debe llamarse abiogénesis. Y, otras veces, la gente utiliza la palabra cuando realmente quieren decir selección natural, lo que es en realidad uno de los mecanismos de la evolución.

Básicamente lo que Darwin afirmaba es que la alteración en las especies es un proceso de adaptación natural en la cual, en una primera etapa se produce la mutación, recombinación y acontecimientos al azar, es decir producción de la variabilidad genética, para en una segunda etapa quedar regulada esa variabilidad mediante la selección natural, y en la cual el proceso artificial generado por el hombre no produce variabilidad[xvii].

2.4 Especies y su población, fundamento de la teoría de Darwin

El pensamiento poblacional

Hay grandeza en esta concepción de la vida,... que mientras este planeta ha ido girando según la constante ley de la gravitación, se han desarrollado y se están desarrollando, a partir de un comienzo tan sencillo, infinidad de formas cada vez más bellas y maravillosas

Charles Darwin

La mayor dificultad que tuvo que enfrentar Darwin para introducir su teoría de la evolución fue sin lugar a dudas el tener que desafiar las ideas que se tenían en su tiempo sobre los seres vivos. Se consideraban las especies como entidades fijas que no cambiaban, puesto que habían sido creadas directamente por Dios estas eran perfectas y no tenían necesidad de cambio. Para imponer su teoría de la evolución y de la selección natural, Darwin tuvo que introducir una nueva forma de entender la variación en la naturaleza, el pensamiento poblacional[xviii].

[xvii] http://anthro.palomar.edu/evolve/evolve_2.htm
[xviii] http://bioinformatica.uab.es/divulgacio/evol.html#revolucion

En aquel entonces las diferencias en la forma, en la conducta o en la fisiología de los organismos de una misma especie no eran más que imperfecciones, errores en la materialización de la idea de la especie. En contraste con esta visión, la variación individual, lejos de considerarla superficial, fue expuesta por Darwin como *la piedra angular de la evolución*.

La variación en el seno de las especies o poblaciones es lo único real, es la materia prima de la evolución, a partir de la que se va a crear toda la diversidad biológica. Son las diferencias existentes entre los organismos de una especie las que, al magnificarse en el espacio y en el tiempo, producirán nuevas poblaciones, nuevas especies, y por extensión, toda la diversidad biológica. Bajo la visión darviniana, la variación es la única realidad de las especies. No hay un color de piel en la especie humana ideal. Cada individuo con su variación característica es un elemento esencial de nuestra especie[xix].

Cuando Darwin exponía sus ideas respecto al hombre, afirmaba que éste condiciona la evolución de determinadas especies para su propio aprovechamiento mediante la selección artificial, sin embargo en su obra explicó que el hombre por si mismo no produce variabilidad; lo único que hace es exponer intencionadamente seres orgánicos a nuevas condiciones de vida, y luego la naturaleza actúa sobre la organización, y causa la variabilidad.

Es claro que el hombre puede seleccionar y selecciona las variaciones que la naturaleza le da, y de este modo las puede controlar como desee. Adapta así animales y plantas a su propio beneficio o placer. Puede hacerlo metódicamente o puede hacerlo inconscientemente, preservando los individuos que le son más útiles de momento, sin pensar en alterar la especie. No hay motivo aparente para que los principios que han actuado con tanta eficacia en la domesticación no hayan actuado en la naturaleza. Nacen más individuos de los que pueden sobrevivir. La más ligera ventaja de un ser sobre los demás con los cuales entra en competencia, una mejor adaptación a las condiciones físicas que le rodean, por mínima que sea, cambiará el equilibrio en su favor.

Pero es importante evitar confundir los enunciados de Darwin en su obra "El origen de las especies" con los términos de evolución. De lo contrario se puede llegar a malinterpretar la evolución con los cambios morfológicos de una especie, es decir los cambios físicos en el

[xix] http://bioinformatica.uab.es/divulgacio/evol.html#revolucion

organismo de algunos individuos de la población. En realidad puede ocurrir evolución sin cambio morfológico; y puede ocurrir cambio morfológico sin evolución. Un ejemplo claro somos los humanos, en general en las poblaciones los individuos son más altos ahora que en el pasado reciente, como resultado de una dieta y medicina mas favorables. Cambios como éste, inducidos solamente por alteraciones en el entorno, no cuentan como evolución, porque no son hereditarios; en otras palabras, el cambio no se transmite a la descendencia del organismo.

Lo que en la biología se conoce como el fenotipo de un organismo esta constituido por sus propiedades morfológicas, fisiológicas, bioquímicas y de comportamiento. El fenotipo de un organismo está determinado por sus genes y su entorno. La mayoría de los cambios debidos al entorno son bastante sutiles, por ejemplo, las diferencias en el tamaño. Los cambios fenotípicos a gran escala son debidos obviamente a cambios genéticos, y por tanto a la evolución.

Otro de los conflictos que se dan al confundir los enunciados de Darwin con la evolución misma es que se tiende a considerar como progreso, pero existe un gran peligro al hacer esa afirmación, las poblaciones simplemente se adaptan a su entorno actual. No se hacen necesariamente mejores con el tiempo en ningún sentido absoluto. Un carácter o estrategia que es exitosa en una ocasión puede no serlo en otra.

Paquin y Adams demostraron esto experimentalmente. Sembraron un cultivo de levadura y lo mantuvieron durante muchas generaciones. Ocasionalmente, surgía una mutación que permitía a su portador reproducirse mejor que sus contemporáneos. Estas cepas mutantes acababan sustituyendo a las cepas anteriormente dominantes. Se tomaron muestras de las cepas más exitosas del cultivo en varias ocasiones. En posteriores experimentos de competición, todas las cepas vencían a las cepas inmediatamente anteriores que dominaban en el cultivo. Sin embargo, algunas muestras del principio podían vencer a las cepas que surgieron al final del experimento.

La habilidad competitiva de una cepa siempre era mejor que la de la anterior, pero la competitividad, en un sentido general, no estaba aumentando. El éxito de cualquier organismo depende del comportamiento de sus contemporáneos. No es probable que exista un diseño o estrategia óptimos para la mayoría de los caracteres o comportamientos, sólo contingentes[xx].

[xx] http://the-geek.org/intro-biologia.html

Por otra parte, los organismos no son objetivos pasivos de su entorno. Todas las especies y en especial el ser humano modifican su propio entorno. Como mínimo, los organismos recogen nutrientes de sus alrededores y depositan desechos. A menudo, los productos de desecho benefician a otras especies. El estiércol animal es un fertilizante para las plantas. Inversamente, el oxígeno que respiramos es un producto de desecho de las plantas. Las especies no cambian simplemente para adaptarse a su entorno; también modifican su entorno para adecuarlo a ellas. Los castores construyen presas para crear un estanque apropiado para mantenerse y sacar adelante a las crías. Alternativamente, cuando el entorno cambia, las especies pueden migrar a climas adecuados o buscar micro entornos a los que estén adaptadas[xxi].

Es importante aclarar que en las publicaciones de Darwin aún existen afirmaciones que no han podido ser demostradas por los registros fósiles, esto muchas veces puede generar confusiones y provocar dudas sobre la evolución en si. En el siguiente capítulo se presentarán aclaraciones al respecto, sin embargo, los conceptos que se manejarán a lo largo de este estudio no entrarán en conflicto con los debates existentes puesto que la computación evolutiva se basa en las adaptaciones de los organismos a su entorno y la supervivencia del más apto.

2.5 Evolución. Debates, evidencias y pruebas

Actualmente rara vez se discute el tema públicamente. Sin embargo, hay una discusión constante entre científicos sobre casi todos los aspectos de la teoría evolutiva. La controversia no es en si la evolución, más bien, los medios por los cuales sucede. El punto en discusión no esta basado en la evolución, sino en la teología. Se debate si las formas de vida se originaron debido a una coincidencia de circunstancias favorables o no.

La teoría de Darwin de la selección natural es la única que pretende explicar cómo los hombres y otras especies son exclusivamente el resultado de fuerzas naturales. Esta es la razón por la cual la teoría del Darwin entra en excesiva discusión, la evolución realmente no juega un papel tan importante. Es la teoría de Darwin la que se enseña en las escuelas, y el hecho de que la mayoría de textos en el tema no hacen la distinción crucial entre la "evolución" y "Darwinismo" tiende a

[xxi] http://www.ewtn.com/library/HOMELIBR/DEATHDAR.TXT

complicar más las cosas. Aunque su nombre es sinónimo de su teoría, la idea de la evolución ya se había mencionado desde los antiguos filósofos griegos, si bien es cierto las publicaciones de Darwin mostraron pruebas concretas de la evolución, a su vez también pretendían proporcionar una explicación de cómo ocurrió la evolución, una en que el origen de la vida era un proceso puramente mecánico y sin intervención alguna de Dios[xxii].

La evolución que se da en una escala reducida, en el interior de una especie y en el intervalo de unas pocas generaciones, se denomina micro evolución. La macro evolución es la evolución a gran escala, y abarca periodos considerables de tiempo, y grandes procesos de transformación; en el caso más extremo comprendería toda la evolución de la vida.

Por medio de su teoría de la selección natural, Darwin pretendía afirmar que una cierta forma de vida original desconocida fue desarrollada y diversificada convirtiéndose en una variedad extensa de todas las plantas y animales que vemos hoy. Este es un punto crucial que frecuentemente entra en debate por los críticos científicos. Darwin observó las evidencias de micro evolución, sin embargo no de macro evolución. Él se refiere constantemente a los cambios pequeños que ocurren dentro en un cierto plazo a la población de una especie. Tal evolución es común, realmente las variedades de pinzones que Darwin observó en las islas de las Islas Galápagos no son mas que otro ejemplo de micro evolución.

Sin evidencia empírica directa, Darwin afirmaba que dentro de períodos de tiempo excesivamente largos estos micro cambios podrían dar lugar a la macro evolución, que consiste en saltos realmente grandes como por ejemplo el de la ameba al reptil y luego al mamífero. Este es el punto en que su teoría funciona únicamente con problemas que todavía no se resuelven en las mentes de muchos científicos. Existen dos lugares posibles para buscar la demostración de la teoría de Darwin: los experimentos fósiles del expediente y de la crianza con los animales.

Si la teoría de Darwin está correcta, el expediente fósil debe demostrar innumerables adaptaciones graduales entre especies anteriores y las más recientes. Darwin estaba enterado, sin embargo, que el expediente fósil de época no demostraba nada que lo apoyara. Había discontinuidades enormes entre los cambios en especies animales y entre los grupos importantes de plantas. Él dio como hecho, en su capítulo "Las imperfecciones del expediente geológico",

[xxii] http://www.ewtn.com/library/Theology/zschonevo2.HTM

que el futuro de la paleontología completara los vacíos, que él admitió ser "la objeción más grave a mi teoría". Las cantidades enormes de fósiles que se han obtenido en excavaciones desde entonces, en todo caso, hacen más evidentes los espacios que preocuparon Darwin.

Jay Gould, biólogo de Harvard, llama a esta carencia del cambio gradual en el expediente del fósil el "secreto comercial" de la paleontología moderna. El expediente fósil demuestra exactamente lo que se demostró en el aquellos días de Darwin, que las especies aparecen repentinamente en un estado completamente desarrollado y cambiaron poco o nada en absoluto antes de desaparecer. Hace cerca de 550 millones de años, al principio de la era cambriana, había una explosión de las formas de vida complejas, como moluscos, medusas, trilobites, para lo cuál no se puede encontrar una sola forma ancestral en rocas anteriores[xxiii].

El paleontólogo Stephen Stanley escribe que "el expediente fósil no demuestra convincentemente una sola transición a partir de una especie a otra". Muchas de las secuencias representativas de la pendiente ancestral son conjeturas y se están desechando constantemente. Paleontólogos, en efecto, encuentran en un fósil de una especie extinta y hacen escalando poco a poco un panorama que la conecta con un animal último o anterior, pero nunca encuentran las formas transitorias que la teoría de Darwin exige[xxiv].

No obstante, si nos centramos en la evolución por si misma no existen dudas al respecto, independientemente la micro evolución pueda convertirse en macro evolución con el pasar del tiempo, es un hecho que mediante observaciones a poblaciones de especies actuales a pequeña escala se puede obtener evidencia directa de evolución. La selección artificial efectuada por el hombre en el perro o el caballo son claros ejemplos que muestran el potencial de modificación de una especie.

Por su propia dimensión temporal, no es posible demostrar la macro evolución directamente, para esto se requeriría de un registro fósil extenso. La historia de la vida es una historia de extinciones, con unos pocos supervivientes. El 99,9% de las especies que han existido alguna vez

[xxiii] http://www.stephenjaygould.org/
[xxiv] http://www.digisys.net/users/hoppnrmt/transitionfossils.htm

están hoy extintas. Grupos enteros de organismos, como los dinosaurios, los trilobites, los ameboideos, se han extinguido sin dejar descendiente alguno[xxv].

Como señala el reconocido paleontólogo S. Gould, el registro fósil no es un relato convencional que conduce a los diferentes linajes a más excelencia, más complejidad, más diversidad. La historia de la vida no muestra dirección ni sentido. La evolución es una narración de eliminación masiva seguida de diferenciación en el interior de unos cuantos supervivientes. Es prácticamente imposible determinar la dirección de la evolución porque la importancia de los acontecimientos concretos son los verdaderos agentes de la historia, ya sean hechos contingentes, como la extinción masiva, o la posesión de una variante adaptativa adecuada cuando ésta es requerida[xxvi].

2.6 Principios de genética en la biología y en los algoritmos genéticos[xxvii]

Todos los organismos vivos están compuestos por células, cada célula contiene el mismo sistema de uno o más cromosomas, los cromosomas son secuencias de ADN, que sirven como un tipo de "modelo" para el organismo. Un cromosoma, por concepto, se puede dividir en genes, cada uno de los cuales codifica una proteína particular. En cierta forma, se puede considerar un gen como la codificación de un rasgo o característica del organismo, como ejemplo el color de los ojos. Las diferentes "codificaciones posibles" para un rasgo, como un ojo puede ser azul, verde o café, se llaman *alelos*. Cada gen está situado en un lugar geométrico particular, es decir una posición especifica en el cromosoma.

Muchos organismos tienen múltiples cromosomas en cada célula. La colección completa de material genético que son todos los cromosomas tomados juntos, se llama el *genoma* del organismo. El término *genotipo* se refiere al sistema de genes contenido en un genoma particular. Se dice que si dos individuos tienen genomas idénticos, entonces también tienen el mismo genotipo. El genotipo da origen a las características físicas del *fenotipo.* El fenotipo es el

[xxv] http://bioinformatica.uab.es/divulgacio/evol.html#revolucion
[xxvi] http://bioinformatica.uab.es/divulgacio/evol.html#revolucion
[xxvii] An Introduction to Genetic Algorithms, Mitchell Melanie -A Bradford Book The MIT Press, Cambridge, Massachusetts • London, England - Fifth printing, 1999, First MIT Press paperback edition, 1998 - Copyright © 1996 Massachusetts Institute of Technology

conjunto de rasgos o características visibles de un organismo, por ejemplo, el color del cabello, el peso o la presencia o ausencia de una enfermedad. Los rasgos fenotípicos no son necesariamente genéticos.

Los organismos cuyos cromosomas están ordenados en pares se llaman *diploide*; los organismos cuyos cromosomas están desapareados se llaman *haploide*. En la naturaleza, la mayoría de las especies que se reproducen sexualmente son diploides, entre estos los seres humanos, que cada uno tiene 23 pares de cromosomas en cada célula somática en el cuerpo. Durante la reproducción sexual, la recombinación o cruce, los genes de cada uno de los padres se intercambian entre cada par de cromosomas para formar un *gameto,* es decir un solo cromosoma, y entonces los gametos de los dos padres se aparean hasta crear un sistema de cromosomas diploide completo.

En la reproducción sexual haploide, los genes se intercambian entre cromosomas singulares de los dos padres. El descendiente está sujeto a una mutación, en la cual las partículas elementales de la DNA llamados nucleótidos se heredan al descendiente, los cambios resultan a menudo por errores de copiado. La aptitud de un organismo se define típicamente como la probabilidad que tiene el organismo para vivir y reproducirse, esto es conocido también como viabilidad. También se define en función del número de descendientes que el organismo logra tener, lo cual se conoce también como fertilidad.

En el contexto de algoritmos genéticos, los términos biológicos se utilizan con la intención de mantener la analogía con biología verdadera, aún cuando las entidades a las que se refieren son mucho más simples que las biológicas. Para conocer la evolución en los algoritmos genéticos, es necesario conocer primero como se desarrolla este campo en el ambiente natural.

En los algoritmos genéticos, el término cromosoma se refiere típicamente a una solución posible a un problema, codificado a menudo como cadena de bits. Los "genes" son por lo general bits aislados o pequeñas cadenas adyacentes de bits que codifican un elemento particular de la solución del candidato. Un ejemplo sería, en el contexto de la optimización de una función de múltiples parámetros, los bits que codificaban un parámetro particular se podrían considerar como un gen.

Un alelo en una secuencia de bits tiene un valor ya sea un cero o uno; Si se utilizan alfabetos más complejos más alelos son posibles en cada posición geométrica. El cruce consiste en por lo general en intercambiar material genético entre dos sujetos haploides, que contienen un solo cromosoma. La mutación consiste en el mover elementos dentro de una secuencia de bits, intercambiando el símbolo en un lugar geométrico aleatoriamente elegido por un nuevo símbolo también aleatoriamente elegido.

La mayoría de los algoritmos genéticos emplean individuos haploides, particularmente, los individuos de un solo cromosoma. El genotipo de un individuo en un algoritmo genético que usa secuencias de bits es simplemente la configuración de bits en el cromosoma de ese individuo. En general no existe una noción de *fenotipo* en el contexto de los algoritmos genéticos, aunque muchos investigadores han experimentado más recientemente con algoritmos en los cuales se utiliza un nivel genotípico y un nivel de fenotipo.

Los cromosomas en una población de algoritmos genéticos toman típicamente la forma de cadenas de bits. Cada posición geométrica en el cromosoma tiene dos alelos posibles, uno y cero. Cada cromosoma se puede visualizar como un punto en el espacio de la búsqueda de las soluciones del candidato. El algoritmo procesa las poblaciones de cromosomas substituyendo sucesivamente a una población por otra. Esto frecuentemente requiere de una función de aptitud que asigne un valor a cada cromosoma en la población que esta siendo procesada. La aptitud de un cromosoma depende de que tan eficientemente el cromosoma solucione el problema tratado.

Capitulo III
Algoritmos genéticos, variaciones y aplicaciones

Realmente no existe una definición rigurosa de lo que es un algoritmo genético. No existe un modelo aceptado por todos los investigadores de computación evolutiva y que diferencie a estos algoritmos de otros métodos evolutivos en la computación. Sin embargo, se puede afirmar que la mayoría de los métodos llamados algoritmos genéticos tienen por lo menos ciertos elementos en común. Entre estos tenemos las poblaciones de cromosomas, la selección según aptitud, el cruce para producir nuevos descendientes y la mutación al azar del nuevo elemento. A continuación se citarán algunas de las definiciones actualmente publicadas.

Un algoritmo genético es un proceso de búsqueda que optimiza una función objetivo F manteniendo una población P de soluciones posibles y que implementa operaciones inspiradas por la genética (conocidos cruce y mutación) para generar una nueva población a partir de la generación anterior. Generalmente las soluciones posibles se encuentran codificadas como cadenas. - Bob Carter[xxviii]

Son algoritmos que codifican las soluciones con una estructura parecida a la de un cromosoma o genoma. El algoritmo genético genera una población de genomas y luego aplica cruces y mutaciones a cada individuo para generar nuevos individuos. Utiliza un criterio de selección en el cual los individuos más aptos sean utilizados para aparearse. Siendo la función objetivo la que determina la aptitud o que tan bueno un individuo es. - Matthew Wall[xxix]

Los algoritmos genéticos son una parte de la computación evolutiva, la cual continúa creciendo rápidamente en el área de la inteligencia artificial. Como su nombre lo indica están inspirados por la teoría de Darwin de la evolución. En pocas palabras, la solución encontrada por un algoritmo genético es mas bien evolucionada. - Marek Obitko[xxx]

Los algoritmos genéticos son algoritmos diseñados para buscar, enfatizar y procrear buenas soluciones para un problema de forma altamente paralela. - James Matthews[xxxi]

[xxviii] Carter and Park, "How good are genetic algorithms at finding large cliques" (BU-CS-TR 93-015) Section 4 discusses implementation on CM-5

[xxix] A Genetic Algorithm for Resource-Constrained Scheduling - by Matthew B Wall, Submitted to the Department of Mechanical Engineering on 14 May 1996 in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Mechanical Engineering.

[xxx] http://cs.felk.cvut.cz/~xobitko/ga/ - August and September of 1998 at the Hochschule für Technik und Wirtschaft Dresden (FH) (University of Applied Sciences).

[xxxi] James Matthews, A "hello world!" genetic algorithm example, http://www.generation5.org /content/2003/gahelloworld.asp

Un algoritmo genético es un método de búsqueda que espera un resultado igual o muy cercano a la solución de un problema dado. - Yi Wang[xxxii]

3.1 Factores de la evolución y la resolución de problemas

Con base en los conceptos de la evolución biológica, vale la pena analizar que es lo que convierte la evolución y adaptación al medio de los organismos uno de los mecanismos mejor diseñados para solucionar problemas de computación en muchos campos. Ya es conocido que la gran mayoría de problemas en los sistemas de computación requieren buscar con un número enorme de posibilidades para encontrar soluciones.

Un ejemplo concreto que puede facilitar esta explicación es el problema de la proteína, planteado a la ingeniería en sistemas para encontrar un algoritmo que buscará entre un número muy extenso de secuencias posibles del aminoácido de una proteína con las características especificadas[xxxiii]. Otro ejemplo en el que se hace evidente la necesidad de búsqueda es en un sistema de reglas o las ecuaciones que pronostican índices económicos y mercados financieros, tal como la devaluación respecto a la moneda extranjera[xxxiv]. Tales problemas de búsqueda a menudo se pueden beneficiar del paralelismo en los algoritmos genéticos, pues muchas posibilidades se exploran simultáneamente de una manera eficiente.

Cuando se habla del paralelismo en sistemas de cómputo, básicamente se refiere a muchos procesos ejecutándose y evaluándose al mismo tiempo, esto es una estrategia inteligente para decidir cual es el sistema de secuencias disponibles más conveniente para evaluar. Muchos problemas de cómputo requieren un programa de computadora que se adapte dinámicamente, es decir que continuamente muestre un buen desempeño en un ambiente cambiante.

Una de las características de estos sistemas es que tienden a ser robustos, ya que por lo general pretenden controlar tareas en ambientes variables y que las interfaces de la aplicación deben adaptarse a las preferencias de los diferentes usuarios. Por otro lado se encuentran también los

[xxxii] Jahau Lewis Chen and Yi-Cheng Tsao. Optimal design of machine elements using genetic algorithms. Chung-Kuo Chi Hsueh Kung Ch'eng Hsueh Pao, 14(2):193-199, April 1993.
[xxxiii] Hunter, Searls, and Shavlik 1993
[xxxiv] Esben Sloth Andersen, Evolutionary Economics: Post-Schumpetrian Contributions

programas que buscan ante todo innovar, la construcción de algo verdaderamente nuevo y original, por ejemplo un nuevo algoritmo para lograr una tarea o aún un nuevo descubrimiento científico.

Finalmente, muchos problemas de cómputo requieren las soluciones complejas que son difíciles de programar de forma lineal. Como ejemplo principal tenemos el problema de crear inteligencia artificial. En los inicios de esta ciencia, los pioneros de la inteligencia artificial creyeron que simular la inteligencia humana sería una tarea que consistiría de manera directa codificar a un programa las reglas que proveerían de inteligencia.

Los sistemas expertos fueron el resultado de esta concepción optimista pero demasiado simplificada. En la actualidad, la mayoría de investigadores de Inteligencia Artificial están convencidos que la inteligencia profunda es demasiado compleja para poder ser codificada manualmente de forma deductiva. No obstante, creen que la mejor ruta a la inteligencia artificial es por medio de un paradigma inductivo, es decir empezando por lo básico y simulando su desarrollo, en el cual los seres humanos escriban solamente reglas muy simples, y los comportamientos complejos tales como la inteligencia emergen del uso y de la interacción masiva, siempre paralelo a las reglas simples originales.

El estudio de los programas de computadora inspirados por el sistema nervioso humano es un ejemplo claro de la filosofía que busca hacer surgir la inteligencia a partir de interacciones y aprendizaje; la computación evolutiva es otro. En las simulaciones del sistema nervioso las reglas son en cierta forma el principio mismo de las neuronas, es decir, una activación que se separa y se consolida o debilita dependiendo de las conexiones; el comportamiento al que se aspira es el reconocimiento y el aprendizaje sofisticados de patrones.

En el cómputo evolutivo las reglas son típicamente "selección natural" con variaciones debido al cruce, la mutación o ambos. En este caso el comportamiento al que se aspira es uno con capacidad de diseño de soluciones eficientes a problemas complejos y capacidad de adaptar estas soluciones ante un ambiente constantemente cambiante. La evolución biológica es una fuente de inspiración atractiva para tratar estos problemas. En efecto, la evolución un método de búsqueda entre un número enorme de las posibilidades para encontrar "soluciones óptimas".

En biología el sistema enorme de posibilidades es el sistema de secuencias genéticas posibles, y las "soluciones deseadas" son los organismos capaces de sobrevivir, adaptarse y de reproducirse en sus ambientes naturales. La evolución se puede también considerar como método para diseñar soluciones innovadoras a los problemas complejos. Por ejemplo, el sistema inmune mamífero es una solución maravillosa desarrollada al problema de los gérmenes que invaden el cuerpo. Desde esta perspectiva, los mecanismos de la evolución pueden inspirar métodos de cómputo para realizar búsquedas. Por supuesto la aptitud de un organismo biológico depende de muchos factores, como por ejemplo si este puede resistir a las características físicas de su ambiente y si puede competir o cooperar con los otros organismos alrededor de ella.

Los criterios de aptitud en los organismos cambian continuamente mientras las criaturas se desarrollan, por lo que la evolución está buscando en un sistema que cambia de posibilidades constantemente. Una búsqueda de soluciones ante condiciones en constante cambio es exactamente lo qué se requiere para los programas de "computación adaptiva". Por otra parte, la evolución es un método de búsqueda masiva en paralelo, es decir que el trabajo que se realiza sobre una especie y a la vez, tal como lo muestra la evolución, cambia millones de especies en paralelo.

Desde cierta perspectiva, las reglas bajo las cuales se rige la evolución son notablemente simples. Las especies se desarrollan por medio de la variación aleatoria, ya sea mediante la mutación, la recombinación u otro operador. Luego la selección natural en la cual los más aptos tienden a sobrevivir y reproducirse, les permite propagar así su material genético a las generaciones futuras. Pero a pesar de ser reglas simples se considera que han producido, por lo menos en gran parte, la variedad y complejidad extraordinaria que existe actualmente en la biosfera

3.2 Computación evolutiva[xxxv]

El desarrollo de los algoritmos genéticos como una rama de la inteligencia artificial fue el producto de un proceso en el que diferentes estrategias fueron diseñadas con el enfoque

[xxxv] http://www.redcientifica.com/doc/doc199904260012.html

deductivo que se exponía anteriormente. La clave para el desarrollo de estos algoritmos fue sin duda alguna la computación evolutiva.

Cuando se habla de computación evolutiva se hace referencia a varias técnicas de resolución de problemas, por lo general bastante complejos, en las que se pretende implementar los procesos naturales de evolución. Bajo el término de Computación evolutiva se engloba a un amplio conjunto de técnicas basadas en la emulación de los procesos naturales de evolución.

El mayor logro de esta ciencia en la solución de problemas es el uso de mecanismos de selección de soluciones potenciales y de construcción de nuevos candidatos por recombinación de características de otros ya presentes, de modo parecido a como ocurre en la evolución de los organismos naturales. Es importante aclarar que el propósito principal no es reproducir los fenómenos naturales, más bien aprovechar las ideas generales que se pueden obtener a partir de estos.

Efectivamente, en el momento en que se tienen varios candidatos para solución de un problema surge inmediatamente la necesidad de establecer criterios de calidad y de selección, siempre basándose en la idoneidad de la solución. También surge la idea de combinar características de buenas soluciones para obtener otras mejores. Dado que fue en el comportamiento natural donde inicialmente se plantearon problemas de ese tipo, es evidente que al aplicar tales ideas en la resolución de problemas científicos y técnicos, se obtengan procedimientos bastante parecidos a los ya encontrados por la naturaleza tras un largo periodo de adaptación.

La computación evolutiva surge a finales de los años 60 cuando John Holland se planteó la posibilidad de incorporar los mecanismos naturales de selección y supervivencia para resolver una gran cantidad de problemas de Inteligencia Artificial que ya habían sido resueltos muy eficientemente por la naturaleza pero que resultaban decepcionantemente inabordables mediante computadoras. Los resultados de sus investigaciones fueron registrados en su libro “Adaptación en sistemas naturales y artificiales”, que con el paso del tiempo ha sido considerado como el punto de arranque de la Computación Evolutiva.

Los intereses de Holland y sus colaboradores fueron en principio académicos, buscaban introducir un marco más amplio para relacionar la inteligencia artificial con la intención de

resolver ciertos problemas genéricos que constantemente aparecían en dicha disciplina. Al llevar a la práctica las ideas de Holland resultaban, cuando menos, poco eficientes. Sin embargo, a mediados de los 80 con la aparición de computadores de altos recursos y bajo costo el problema cambió por completo. Actualmente las computadoras pueden ser utilizadas para resolver exitosamente cierto tipo de problemas de ingeniería y de las ciencias sociales que anteriormente eran tratados con dificultad.

Posiblemente como consecuencia del esfuerzo divulgador de los autores de esta nueva ciencia y debido también a las nuevas posibilidades que se descubrían, el desarrollo de la computación evolutiva a partir de entonces fue espectacular.

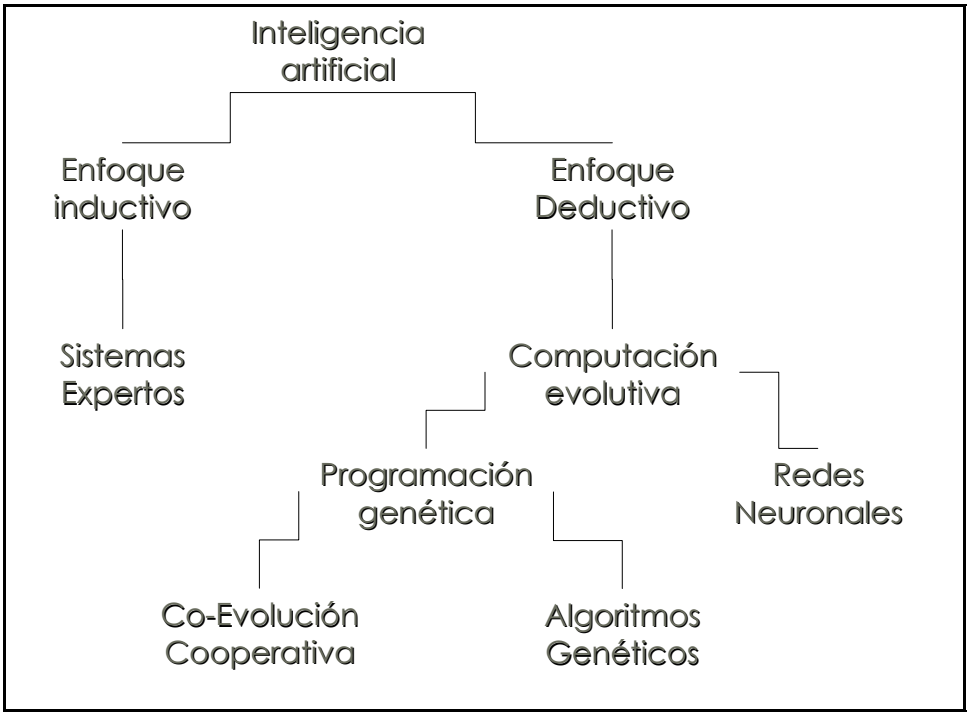


Figura 3.1 – Subdivisiones de la inteligencia artificial

3.3 Funcionamiento de los algoritmos genéticos[xxxvi]

[xxxvi] An Introduction to Genetic Algorithms
Mitchell Melanie
A Bradford Book The MIT Press
Cambridge, Massachusetts • London, England
Fifth printing, 1999

Dado un problema claramente definido y una representación de las soluciones posibles en cadenas de bits, el funcionamiento de un algoritmo genético simple puede describirse de la siguiente forma:

1. Se comienza con una población aleatoriamente generada de *n* cromosomas representados por cadenas de *l* bits, las cuales son soluciones posibles para el problema.
2. Se calcula la aptitud *f(x)* de cada cromosoma *x* en la población.
3. Se repiten los siguientes pasos hasta que han generado *n* descendientes:
    a. Seleccionar un par de cromosomas de la generación padre actual, siendo la probabilidad de selección una función de aptitud en aumento. La selección se hace "con el reemplazo", es decir que un mismo cromosoma se puede seleccionar más de una vez para convertirse en padre de la nueva generación.
    b. La probabilidad *Pc*, es decir "probabilidad de cruce" o "tasa de cruce", intercambia la pareja en un punto aleatoriamente elegido con una probabilidad uniforme, para formar dos descendientes. Si no ocurre ningún cruce, los dos descendientes son copias exactas de sus padres respectivos. Es necesario tomar en cuenta que aquí la tasa de cruce está definida como la probabilidad que dos padres se intercambien sobre un solo punto. Hay también versiones de algoritmos genéticos con “tráfico de múltiples puntos" en el cual la tasa de cruce para un par de padres es el número de puntos en los cuales ocurre el intercambio.
    c. Se hace mutar a los descendientes en cada lugar geométrico con la probabilidad *Pm,* que es la probabilidad de la mutación o la tasa de la mutación, y se colocan los cromosomas alterados en la nueva población.
    d. Si n es impar, un nuevo miembro de la población puede ser desechado al azar.
4. Se sustituye la población actual por una nueva población.
5. Se repite el proceso a partir del paso 2.

Cada repetición de este proceso se llama generación. Un algoritmo genético se repite típicamente de 50 a 500 o más veces. El sistema entero de generaciones se le llama una iteración. Al final de una iteración hay por lo general uno o más cromosomas altamente aptos en la población. Puesto

First MIT Press paperback edition, 1998

que la aleatoriedad desempeña un papel importante en cada iteración, dos iteraciones con diferentes números de semillas aleatoriamente generadas producirán diversos comportamientos. Tanto así que los investigadores de algoritmos genéticos han reportado diferentes promedios estadísticos para un mismo problema, tales como la mejor aptitud encontrada en una iteración o la generación en la que se descubre al individuo con la mejor aptitud.

El procedimiento simple descrito es la base para la mayoría de los algoritmos genéticos. Hay un número de detalles que es necesario tomar en cuenta, por ejemplo el tamaño de la población y las probabilidades de cruce y de mutación, el éxito del algoritmo depende muy a menudo en gran parte a estos detalles.

Con este ejemplo se describirá más detenidamente el algoritmo simple definido anteriormente. Suponiendo que es l =8, longitud de la cadena; que $f(x)$ es igual a la cantidad de números unos en la secuencia de bits x, esta es una función extremadamente simple de aptitud usada aquí solamente con propósitos ilustrativos. Suponiendo que el tamaño n de la población es igual a 4, que Pc = 0.7 y que Pm = 0.00. La población inicial generada al azar puede ser similar a la tabla 2.1.

Realmente los valores típicos de l y n suelen estar entre los 50 y 1000, pero los valores dados para las probabilidades Pc y Pm son bastante comunes.

Nombre del Cromosoma	Cadena del cromosoma	Aptitud
A	00000110	2
B	11101110	6
C	00100000	1
D	00110100	3

Tabla 2.1 – Población de genomas aleatoriamente generados

Un método común de selección en los algoritmos genéticos es la selección del mas apto, en la cual el número de veces que se espera que un individuo se reproduzca es igual a su aptitud

dividida entre la aptitud promedio de toda la población. Esto es equivalente a lo que los biólogos llaman "selección de viabilidad".

Un método simple de implementar la selección en base a la aptitud es utilizando el "muestreo de la ruleta" (Goldberg 1989a), que es conceptualmente equivalente a asignar a cada uno de los individuos una rebanada de una rueda circular, donde el tamaño del área es proporcional a la aptitud del individuo. Se hace girar la ruleta y se detiene sobre una rebanada, seleccionando así al individuo correspondiente.

En el ejemplo mencionado $n = 4$, la ruleta se haría girar cuatro veces; las primeras dos vueltas podrían elegir los cromosomas B y D para ser padres, y las segundas dos vueltas pudieron elegir los cromosomas B y C para ser padres. Es posible que A no sea seleccionada es debido a la aleatoriedad. Si la ruleta fuera hecha girar muchas veces, los resultados medios estarían más cerca a los valores previstos.

Una vez que seleccionen a un par de padres, con la probabilidad *Pc*, se cruzan para formar a 2 descendientes. Si no se cruzan, entonces los descendientes son copias exactas de cada padre. En el ejemplo, si los padres B y D se cruzan a partir de la primera posición de bits para formar los descendientes entonces E = 10110100 y F = 01101110.

Después, cada descendiente estaría sujeto a la mutación en cada posición geométrica con la probabilidad *Pm*. Por ejemplo, si el descendiente E es transformado en el sexto lugar geométrico formaría E’ = 10110000, los descendientes F y C no sufrirían ninguna mutación, Pero el descendiente B sería transformado en el primer lugar geométrico para formar B’ = 01101110. La nueva población estaría formada por los cromosomas de la tabla 2.2.

Nombre del Cromosoma	Cadena del cromosoma	Aptitud
E’	10110000	3
F	01101110	5
C	00100000	1
B’	01101110	5

Tabla 2.2 – Población después de haber evolucionado

Es importante recalcar que en la nueva población, a pesar de que la mejor secuencia que tenía aptitud de 6, se perdió, sin embargo la aptitud media incremento de 12/4 a 14/4. La continua iteración de este procedimiento dará eventualmente como resultado una secuencia de solo números 1.

3.4 Usos típicos de los algoritmos genéticos[xxxvii]

Por lo general los algoritmos genéticos son utilizados en problemas de búsqueda, sin embargo es importante definir de manera exacta el término "búsqueda" y compararlo con sus otros significados en la informática. Hay por lo menos tres significados diferentes de la "búsqueda" y estos se describen a continuación.

Inicialmente tenemos la *búsqueda de datos almacenados*, aquí el problema consiste en recuperar eficientemente la información almacenada en una computadora. Por ejemplo en una base de datos grande se puede realizar una búsqueda de los nombres y de las direcciones almacenadas bajo un orden específico. En estos casos al investigar la manera mas conveniente de buscar, la "búsqueda binaria" es un método eficiente para encontrar los registros deseados[xxxviii].

También tenemos la *búsqueda de trayectorias para alcanzar metas*, aquí el problema es encontrar eficientemente una serie de acciones que moverán un sistema desde un estado inicial dado a una meta predeterminada. Esta forma de búsqueda es fundamental para muchos enfoques en la inteligencia artificial. Un ejemplo simple y bastante familiar en los estudios de inteligencia artificial es la búsqueda de una ruta para desplazarse. Un sistema requiere encontrar una ruta para desplazarse de un punto a otro. A lo largo del recorrido muchas decisiones son necesarias para finalmente encontrar el destino, la solución es el conjunto de decisiones que definen la ruta a utilizar. A cada una de las posibles decisiones tomadas se le puede llamar un movimiento.

[xxxvii] GENETIC ALGORITHMS AND TRADITIONAL SEARCH METHODS – Melanie Mitchell

[xxxviii] Data Structures and Algorithms II
Lecture 3:
String Search Algorithms
http://www.macs.hw.ac.uk/~alison/alg/lectures/l3.pdf

Los algoritmos de búsqueda discutidos en la mayoría de textos de inteligencia artificial, son métodos para encontrar eficientemente la mejor o, así como en este caso, más corta trayectoria del estado inicial al estado final. Los algoritmos típicos son "búsqueda de primera profundidad", "rama y límite," y "A *"[xxxix], los cuales no serán explicados puesto que están fuera del contexto de este documento.

Finalmente tenemos la *búsqueda de soluciones*, la cual es una clase más general de búsqueda que la "búsqueda de trayectorias para alcanzar metas". La idea es encontrar eficientemente la solución a un problema en un espacio más grande de soluciones posibles. En especial este tipo de problemas son los que comúnmente se resuelven implementando algoritmos genéticos.

Existe una clara diferencia entre la primera clase de búsquedas y las otras dos. En el primer caso se refiere a los problemas en los cuales se requiere encontrar un dato o alguna secuencia especifica en una colección de información explícitamente almacenada. En los siguientes dos tipos de búsqueda, la información que se requiere no se encuentra explícitamente almacenada. En realidad se crean las soluciones posibles mientras el proceso de la búsqueda se realiza.

Por ejemplo, los métodos de búsqueda utilizados en inteligencia artificial para solucionar problemas de trayectorias no parten de un árbol completo de búsqueda en el que todos los nodos, que en este caso serían todos los movimientos posibles, están previamente almacenados en memoria.

Para la mayoría de los problemas de este tipo hay demasiados nodos posibles en el árbol para almacenarlos todos. Mas bien, el árbol de búsqueda se elabora gradualmente de manera que depende del algoritmo particular y la meta es encontrar una solución óptima o de alta calidad, examinando solamente una porción pequeña del árbol. Asimismo, al buscar un espacio de soluciones posibles con un algoritmo genético, no todas las soluciones posibles se formulan y se evalúan inicialmente; En realidad el algoritmo es un método para encontrar soluciones óptimas o aceptables examinando solamente una fracción pequeña de los candidatos posibles.

[xxxix] A parallel depth first search branch and bound algorithm for the quadratic assignment problem – Bernard Mans, Thierry Mautor and Catherine Roucairol (Jan 18 1994)

La *búsqueda de soluciones* incluye a la *búsqueda de trayectorias*, puesto que una trayectoria a través de un árbol de búsqueda se puede codificar como una solución posible. Para el problema del desplazamiento en una ciudad, las soluciones posibles podrían ser listas de movimientos del estado inicial a un cierto otro estado, pero en este caso seria correcto solamente si el estado final es la meta. Sin embargo, muchas *búsquedas de trayectorias* son resueltas mejor por las técnicas de inteligencia artificial utilizando árboles binarios, en los que soluciones parciales pueden ser evaluadas, ya que por medio de las técnicas de algoritmos genéticos las soluciones posibles completas deben ser generadas antes de que puedan ser evaluadas.

Es importante tener en cuenta que los métodos normales de búsqueda por árboles de inteligencia artificial no pueden ser aplicados siempre. No todos los problemas requieren encontrar una trayectoria de un estado inicial a una meta. Realmente muchas soluciones buscadas tienden a cambiar conforme nuevas alternativas son encontradas y en estas búsquedas no es de interés la forma en que se obtiene la mejor alternativa. Los algoritmos genéticos ofrecen un método general para solucionar las *búsquedas de soluciones* así como otras técnicas inspiradas en la computación evolutiva. En términos generales se pueden resumir de la siguiente manera.

1. Se elije al azar una solución posible, codificada como cadena de bits. Y se denomina a esta cadena “solución actual”.

2. Sistemáticamente se altera por mutación cada bit en la cadena, empezando de izquierda a derecha y uno a la vez, registrando la aptitud de las cadenas mutantes que resultan.

3. Si cualquiera de los resultados de la mutación muestran un aumento de la aptitud, entonces se asigna como nueva “solución actual” a la cadena que muestre la mejor aptitud.

4. Si no hay aumento de la aptitud, entonces se mantiene la cadena de “solución actual” y se regresa al paso 1. Si no, se pasa al paso 2 con un nuevo “solución actual”.

5. Cuando un número necesario de evaluaciones de aptitud del sistema han sido realizadas, se selecciona como resultado la cadena con la mejor aptitud encontrada.

La aplicación de este simple algoritmo será discutido en el siguiente capítulo, en el cual se analizarán las circunstancias en las que este puede desarrollarse de manera optima.

En la inteligencia artificial tales métodos generales que pueden trabajar en una gran variedad de problemas, se les llama los *métodos débiles* para distinguirlos de los *métodos fuertes* diseñados especialmente para trabajar en problemas específicos. Todas las *búsquedas de soluciones* generan inicialmente un conjunto de soluciones posibles que dentro del algoritmo genético es la población inicial, luego evalúan las soluciones posibles según algunos criterios de aptitud, decide en base de esta evaluación cuales candidatos serán almacenados y cuales desechados. Finalmente se producen variantes usando una cierta clase de operadores de mutación en los candidatos sobrevivientes.

3.5 Paralelismo en los algoritmos genéticos [xl]

Aunque los algoritmos genéticos pueden ser descritos de forma simple y programados rápidamente, su comportamiento puede ser complicado y todavía existen diferentes opiniones sobre cómo trabajan de de forma optima y para qué tipos de problemas es posible adecuarlos. Muchos estudios se han realizado en los fundamentos teóricos de los algoritmos genéticos, pero tradicionalmente se asume que, en un nivel muy general, un algoritmo genético funciona descubriendo, acentuando, y recombinado bloques de valores de manera paralela. La idea en si consiste en que las mejores soluciones están compuestas por bloques que combinando sus valores por segmentos forman soluciones eficientes y por tanto son mas aptos dentro de las secuencias a las que pertenecen.

Holland introdujo la noción de esquemas en 1975 para formalizar la noción informal de los denominados "bloques de edificio". Un esquema es un sistema de cadenas de bits que se pueden describir por una plantilla compuesta de unos, ceros y asteriscos, los asteriscos que representan datos indiferentes. Por ejemplo, el esquema H = 1 * * * * 1 representa el sistema de todas las

[xl] Genetic Algorithms
Computer programs that "evolve" in ways that resemble natural selection can solve complex problems even their creators do not fully understand
by John H. Holland - http://www.econ.iastate.edu/tesfatsi/holland.GAIntro.htm

cadenas de 6 bits que comiencen y terminen con 1. Las secuencias que caben esta plantilla, por ejemplo 100111 y 110011 forman parte del esquema H. Este esquema contiene dos bits definidos lo que es equivalente a decir de orden 2.

Es importante aclarar que no cada subconjunto posible del sistema de cadenas de bits con longitud x se puede describir como un esquema; de hecho, la gran mayoría no lo son. Hay 2^x cadenas posibles de longitud x, por lo que los subconjuntos posibles de secuencia son 2^{2x}, pero hay solamente 3^x esquemas posibles. Sin embargo, un principio fundamental en la teoría tradicional de algoritmos genéticos, es que los esquemas son en si los "bloques de edificio" que el algoritmo puede procesar con eficacia bajo los operadores de selección, como por ejemplo la mutación y lo conocido como crossover.

Estas definiciones tienen como propósito simplificar los procesos del algoritmo genético, pues no se operan directamente las cadenas de bits sino los esquemas, Cualquier cadena de bits dada de longitud x es un caso de 2^x esquemas diferentes. Por ejemplo, la secuencia 11 es un caso del **, es decir forma parte de las cuatro cadenas de bits posibles de longitud 2. Pero a su vez pertenece al esquema, * 1, 1 *, y al esquema 11 que contiene únicamente la secuencia, 11.

Entonces cualquier población dada de n cadenas de bits puede contener casos entre 2^x y nx2^x esquemas posibles. Si todas las secuencias son idénticas, entonces hay casos de exactamente 2^x esquemas diferentes; si no, el número es menor o igual a nx2^x. Esto significa que, en una generación dada, mientras el algoritmo genético está evaluando explícitamente la aptitud de las n secuencias en la población, él realmente está estimando implícitamente la aptitud media de un número mucho más grande esquemas, donde la aptitud media de ese esquema se define como la aptitud media de todos los casos posibles de ese esquema.

Por ejemplo, en una población de n secuencias aleatoriamente generadas, como promedio la mitad de las secuencias serán casos del esquema 1***··· * y la otra mitad serán casos de 0 ***···*. Las evaluaciones de aproximadamente $n/2$ secuencias que son casos de 1***··· * dan un estimado de la aptitud promedio de ese esquema, esto es una estimación porque los casos evaluados adentro una población son solamente una muestra pequeña de todos los casos posibles. Así como los esquemas no son representados ni son evaluados explícitamente por el algoritmo

genético, las estimaciones de la aptitud promedio del esquema no son calculadas ni son almacenadas explícitamente. Sin embargo, el comportamiento del algoritmo, en términos del aumento y disminución del número de casos en los esquemas dados de la población, puede ser descrito como si realmente calculara y almacenara estos promedios.

3.6 Operación de los esquemas en la solución de problemas[xli]

En la solución de problemas computacionales se utiliza como base la noción de los "esquemas" y se hace evidente su importancia para los algoritmos genéticos. El propósito original de John Holland para desarrollar los algoritmos genéticos era construir un marco teórico para la adaptación según se ve en la naturaleza y aplicarlo al diseño de sistemas artificiales adaptivos. Según Holland, un sistema adaptivo debe persistentemente identificar, examinar e incorporar las características estructurales de la teoría evolutiva para desempeñarse óptimamente en un ambiente específico. Los esquemas están diseñados para ser una formalización de tales características estructurales.

En el contexto de la genética, los esquemas corresponden a las agrupaciones de genes que en conjunto efectúan una cierta adaptación en un organismo; la evolución descubre y propaga tales agrupaciones. Por supuesto, la adaptación es posible solamente en un mundo en el cual haya un ambiente con una estructura que pueda ser descubierta y explotada. La adaptación es imposible en un ambiente que carece de cambios o de cierto grado de aleatoriedad.

El análisis de los esquemas de Holland demostró que un algoritmo genético, al mismo tiempo en que explícitamente calcula la aptitud de los miembros de una población *n*, estima implícitamente la aptitud promedio de un número mucho más grande de esquemas, esto se da expresamente al calcular las aptitudes promedio de los casos observados dentro de esquemas en la población. Hace esto sin necesitar de memoria o tiempo adicional de cómputo, únicamente el necesario para procesar los *n* miembros de la población.

[xli] Muhlenbein, H, 1992, "How genetic algorithms really work: I. mutation and hill-climbing." In Parallel Problem Solving from Nature 2, Manner, R & Manderick, B,

Holland llamaba a esto "paralelismo implícito". Por supuesto que la exactitud de estas estimaciones esta claramente sujeta a las variaciones de los esquemas en cuestión. El análisis de Holland también demostró que para los esquemas cuya aptitud se calcula sobre el promedio reciben un alto número de "ensayos", es decir, numero de casos en la población. El teorema del esquema se ha interpretado para implicar eso específicamente, en un algoritmo genético, los esquemas cortos y de “bajo orden” cuya aptitud media esta sobre el valor promedio recibirá un número exponencial de muestras a lo largo del tiempo.

El análisis de Holland sugiere que la selección se enfoque, de forma incremental, en buscar en los subconjuntos con aptitud estimada por encima del promedio. Estos estarán definidos por los esquemas observados con aptitud por encima del promedio. Mientras tanto, mediante el cruce los "bloques de edificio" de alta capacidad de adaptación se unen en una misma secuencia para crear otras secuencias de una aptitud cada vez más alta. La mutación desempeña un papel semejante a una póliza de seguro, cerciorándose de que la diversidad genética nunca se pierda completamente en cualquier lugar geométrico.

Holland enmarca la adaptación como tensión entre la "exploración", es decir la búsqueda de nuevas y útiles adaptaciones, y la "explotación", que es el uso y la propagación de estas adaptaciones. La tensión se hace visible puesto que cualquier movimiento enfocado en la exploración, es decir en la prueba de nuevos esquemas no vistos o en los esquemas que de casos vistos que han probado tener hasta cierto punto baja aptitud, reducen en la misma medida la explotación de esquemas previamente probados y de alta adaptación.
Para cualquier sistema o población que deba de hacer frente a ambientes con un cierto grado de imprevisión, un equilibrio óptimo entre la exploración y la explotación deben ser encontrados. El sistema tiene que mantenerse continuamente probando nuevas posibilidades, de lo contrario se podría “sobre adaptar” y volverse completamente inflexible ante nuevos cambios, pero a su vez tiene que incorporar y utilizar continuamente sus experiencias previas como guía para el comportamiento futuro.

Los algoritmos genéticos originales propuestos por Holland tenían un "plan de adaptación" para lograr un equilibrio apropiado entre la exploración y la explotación en un sistema adaptivo. El

análisis de esquemas demostró que, dadas ciertas condiciones, el algoritmo genético alcanza de hecho un equilibrio casi óptimo. Los argumentos para esto se basan en secuencias binarias y el cruce de un solo punto. Los esquemas útiles, según lo definido por Holland, son una clase de los subconjuntos de secuencias binarias de alta adaptación que evitan la interrupción significativa mediante el cruce y la mutación y pueden sobrevivir así para re-combinarse con otros esquemas.

3.7 Cuando utilizar algoritmos genéticos[xlii]

En los diferentes estudios documentados de los algoritmos genéticos se pueden encontrar una gran cantidad de usos acertados, pero hay también muchos casos en los cuales estos algoritmos se desempeñan mal. En muchas ocasiones se ha intentado determinar bajo que circunstancias de uso potencial particular, un algoritmo genético es buen método a utilizar, sin embargo, no existe una respuesta rigurosa.

Muchos investigadores están de acuerdo con las nociones que es conveniente utilizarlos si el espacio en el que se buscará es grande o si se sabe que no es perfectamente liso y uniforme, como por ejemplo en una única curva continua. En caso de que el espacio de búsqueda no esta bien definido o si la función de aptitud es ruidosa, si la tarea no requiere encontrar el grado óptimo global o en caso es suficiente encontrar rápidamente una solución relativamente buena, un algoritmo genético tendrá muy buenas probabilidades de ser competitivo o de sobrepasar otros métodos "débiles", es decir los métodos que no utilizan conocimientos específicos del espacio en su procedimiento de la búsqueda.

Si un espacio de búsqueda no es grande, entonces puede ser recorrido exhaustivamente y finalmente se puede garantizar que se ha encontrado la mejor solución posible, mientras que un algoritmo genético podría converger en un grado óptimo local en lugar de encontrar la mejor solución global. Si el espacio es liso o uniforme, un algoritmo enfocado al gradiente tal como el

[xlii] An Introduction to Genetic Algorithms
Mitchell Melanie
A Bradford Book The MIT Press
Cambridge, Massachusetts • London, England
Fifth printing, 1999
First MIT Press paperback edition, 1998

de la pendiente mas inclinada será mucho más eficiente que un algoritmo genético al explotar la gráfica de aptitudes para las soluciones posibles.

Si el espacio esta bien definido, al igual que el espacio para el conocido problema del vendedor que viaja por ejemplo, los métodos de búsqueda que usan la heurística de dominios específicos se pueden diseñar a menudo de forma que superan cualquier método del propósito general tal como lo son los algoritmos genéticos.

Si la función de aptitud es ruidosa, es decir que implica evaluar soluciones de medidas de un proceso real, un método de búsqueda que examina una posible solución a la vez, tal como los algoritmos basados en las pendientes se puede desviar de forma irrecuperable por el ruido. Sin embargo un algoritmo genético tiene dificultades para mantener un desempeño estable ante la presencia de ruido en cantidades pequeñas, esto se debe a que trabaja en base a información estadística de aptitud que se acumula sobre muchas generaciones.

Estas guías generales, por supuesto, no predicen estrictamente cuando un algoritmo genético será un método eficaz de búsqueda y competitivo con otros procedimientos conocidos. El funcionamiento de un algoritmo genético dependerá mucho de detalles tales como el método para codificar las soluciones posibles, los operadores, los ajustes de parámetros y el criterio particular para el éxito de adaptarse.

3.8 Algunos ejemplos específicos de algoritmos genéticos [xliii]

Mientras el poder de la evolución gana reconocimiento cada vez más generalizado, los algoritmos genéticos se utilizan para abordar una amplia variedad de problemas en un conjunto

[xliii] Algoritmos genéticos y computación evolutiva
Adam Marczyk
2004
http://the-geek.org/docs/algen/algen.html#key-58

de campos sumamente diverso, demostrando claramente su capacidad y su potencial. Esta sección analizará algunos de los usos más notables en los que han tomado parte.

3.8.1 Acústica

Sato et al. 2002[xliv]utilizaron algoritmos genéticos para diseñar una sala de conciertos con propiedades acústicas óptimas, maximizando la calidad del sonido para la audiencia, para el director y para los músicos del escenario. Esta tarea implica la optimización simultánea de múltiples variables. Comenzando con una sala con forma de caja de zapatos, el AG de los autores produjo dos soluciones no dominadas, ambas descritas como "con forma de hoja" (p. 526). Los autores afirman que estas soluciones tienen proporciones similares al Grosser Musikvereinsaal de Viena, el cual está considerado generalmente como una de las mejores -si no la mejor- salas de conciertos del mundo, en términos de propiedades acústicas.

Porto, Fogel y Fogel 1995[xlv] utilizaron programación evolutiva para adiestrar a redes neuronales para distinguir entre reflexiones sonoras desde distintos tipos de objetos: esferas metálicas hechas por el hombre, montañas submarinas, peces y plantas, y ruido aleatorio de fondo. Tras 500 generaciones, la mejor red neuronal que evolucionó tenía una probabilidad de clasificación correcta que iba desde el 94% al 98%, y una probabilidad de clasificación errónea entre un 7,4% y un 1,5%, que son "probabilidades razonables de detección y falsa alarma" (p. 21). Esta red evolucionada igualó las prestaciones de otra red desarrollada mediante recocido simulado, y superó consistentemente a redes entrenadas mediante propagación hacia atrás, las cuales "se atascaban repetidamente en conjuntos de pesos subóptimos que no producían resultados satisfactorios" (p. 21). En contraste, ambos métodos estocásticos demostraron su capacidad para superar estos óptimos locales y producir redes más pequeñas, efectivas y robustas; pero los autores sugieren que el algoritmo evolutivo, a diferencia del recocido simulado, opera sobre una

[xliv] Sato, S., K. Otori, A. Takizawa, H. Sakai, Y. Ando y H. Kawamura. ``Applying genetic algorithms to the optimum design of a concert hall." Journal of Sound and Vibration, vol.258, no.3, p. 517-526 (2002)
[xlv] Porto, Vincent, David Fogel y Lawrence Fogel. ``Alternative neural network training methods." IEEE Expert, vol.10, no.3, p.16-22 (junio de 1995).

población, y por tanto se beneficia de la información global sobre el espacio de búsqueda, conduciendo potencialmente hacia un rendimiento mayor a la larga.

Tang et al. 1996[xlvi] analizan los usos de los algoritmos genéticos en el campo de la acústica y el procesamiento de señales. Un área de interés particular incluye el uso de AGs para diseñar sistemas de Control Activo de Ruido (CAR), que eliminan el sonido no deseado produciendo ondas sonoras que interfieren destructivamente con el ruido. Esto es un problema de múltiples objetivos que requiere el control y la colocación precisa de múltiples altavoces; los AGs se han utilizado en estos sistemas tanto para diseñar los controladores como para encontrar la colocación óptima de los altavoces, dando como resultado una "atenuación efectiva del ruido" (p. 33) en pruebas experimentales.

3.8.2 Ingeniería aeroespacial

Obayashi et al. 2000[xlvii] utilizaron un algoritmo genético de múltiples objetivos para diseñar la forma del ala de un avión supersónico. Hay tres consideraciones principales que determinan la configuración del ala -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). Estos objetivos son mutuamente exclusivos, y optimizarlos todos simultáneamente requiere realizar contrapartidas.

El cromosoma de este problema es una cadena de 66 números reales, cada uno de los cuales corresponde a un aspecto específico del ala: su forma, su grosor, su torsión, etcétera. Se simuló una evolución con selección elitista durante 70 generaciones, con un tamaño de población de 64 individuos. Al final de este proceso había varios individuos paretianos, cada uno representando una solución no dominada del problema. El artículo comenta que estos individuos ganadores tenían características "físicamente razonables", señalando la validez de la técnica de optimización (p. 186). Para evaluar mejor la calidad de las soluciones, las seis mejores fueron comparadas con un diseño de ala supersónica producido por el Equipo de Diseño SST del Laboratorio Aeroespacial Nacional de Japón. Las seis fueron competitivas, con valores de

[xlvi] Tang, K.S., K.F. Man, S. Kwong y Q. He. ``Genetic algorithms and their applications." IEEE Signal Processing Magazine, vol.13, no.6, p.22-37 (noviembre de 1996).

[xlvii] Obayashi, Shigeru, Daisuke Sasaki, Yukihiro Takeguchi, y Naoki Hirose. ``Multiobjective evolutionary computation for supersonic wing-shape optimization." IEEE Transactions on Evolutionary Computation, vol.4, no.2, p.182-187 (julio de 2000).

resistencia y carga aproximadamente iguales o menores a los del ala diseñada por humanos; en particular, una de las soluciones evolucionadas superó al diseño del LAN en los tres objetivos. Los autores señalan que las soluciones del AG son similares a un diseño llamado "ala flecha", sugerido por primera vez a finales de los años 50, pero que finalmente fue abandonado en favor del diseño más convencional con forma de delta.

En un artículo posterior (Sasaki et al. 2001[xlviii]), los autores repitieron el experimento añadiendo un cuarto objetivo, a saber, minimizar el momento de torsión (un conocido problema en los diseños de alas flecha en el vuelo supersónico). También se añadieron puntos de control adicionales para el grosor al conjunto de variables de diseño. Tras 75 generaciones de evolución, se compararon dos de las mejores soluciones paretianas con el diseño de ala que el Laboratorio Aeroespacial Nacional japonés realizó para el avión supersónico experimental NEXST-1. Se descubrió que ambos diseños (además de un diseño óptimo de la simulación anterior, explicada arriba) eran físicamente razonables y superiores al diseño del LAN en los cuatro objetivos.

Williams, Crossley y Lang 2001[xlix] aplicaron algoritmos genéticos a la tarea de situar órbitas de satélites para minimizar los apagones de cobertura. Mientras la tecnología de telecomunicaciones sigue progresando, los humanos somos cada vez más dependientes de las funciones vitales que realizan los satélites en órbita alrededor de la Tierra, y uno de los problemas con los que se enfrentan los ingenieros es el diseño de las trayectorias orbitales. Los satélites que se encuentran en una órbita terrestre alta, a unos 35.000 kilómetros de altitud, pueden ver amplias secciones del planeta al mismo tiempo y estar en contacto con las estaciones terrestres, pero son mucho más caros de lanzar y más vulnerables a las radiaciones cósmicas. Es más económico colocar satélites en órbitas bajas, en algunos casos a sólo unos pocos cientos de kilómetros; pero, a causa de la curvatura de la Tierra, es inevitable que estos satélites pierdan durante un tiempo la línea de visión con los receptores terrestres, y por lo tanto se vuelven inútiles. Incluso las constelaciones de varios satélites tienen apagones ineludibles y pérdidas de cobertura por esta razón. El reto consiste en colocar las órbitas de los satélites para minimizar este tiempo muerto. Esto es un

[xlviii] Sasaki, Daisuke, Masashi Morikawa, Shigeru Obayashi y Kazuhiro Nakahashi. ``Aerodynamic shape optimization of supersonic wings by adaptive range multiobjective genetic algorithms." En Evolutionary Multi-Criterion Optimization: First International Conference, EMO 2001, Zurich, Switzerland, March 2001: Proceedings, K. Deb, L. Theile, C. Coello, D. Corne y E. Zitler (eds). Notas de la confenrencia en Computer Science, vol.1993, p.639-652. Springer-Verlag, 2001

[xlix] Williams, Edwin, William Crossley y Thomas Lang. ``Average and maximum revisit time trade studies for satellite constellations using a multiobjective genetic algorithm." Journal of the Astronautical Sciences, vol.49, no.3, p.385-400 (julio-septiembre de 2001).

problema multiobjetivo que implica la minimización de el tiempo medio de apagón para todas las localizaciones y el tiempo máximo de apagón para cada una de las localizaciones; en la práctica, estos objetivos resultan ser mutuamente exclusivos.

Cuando se utilizó el AG en este problema, los resultados que evolucionaron para constelaciones de tres, cuatro y cinco satélites eran extraños, configuraciones orbitales muy asimétricas, con los satélites colocados alternando huecos grandes y pequeños, en lugar de huecos de igual tamaño como habrían hecho las técnicas convencionales. Sin embargo, esta solución redujo significativamente los tiempos medio y máximo de apagón, en algunos casos hasta en 90 minutos. En un artículo periodístico, el Dr. William Crossley señaló que "ingenieros con años de experiencia aeroespacial quedaron sorprendidos con el rendimiento ofrecido por el diseño no convencional".

Keane y Brown 1996[l] utilizado un AG para producir un nuevo diseño para un brazo o jirafa para transportar carga que pudiese montarse en órbita y utilizarse con satélites, estaciones espaciales y otros proyectos de construcción aeroespacial. El resultado, una estructura retorcida con aspecto orgánico que se ha comparado con un fémur humano, no utiliza más material que el diseño de brazo estándar, pero es ligera, fuerte y muy superior a la hora de amortiguar las vibraciones perjudiciales, como confirmaron las pruebas reales del producto final. Y sin embargo "Ninguna inteligencia produjo los diseños. Simplemente evolucionaron" (Petit 1998). Los autores del artículo comentan además que su AG sólo se ejecutó durante 10 generaciones, debido a la naturaleza computacionalmente costosa de la simulación, y la población no se había estancado todavía. Haber proseguido la ejecución durante más generaciones habría producido indudablemente mayores mejoras de rendimiento.

[l] Keane, A.J. y S.M. Brown. ``The design of a satellite boom with enhanced vibration performance using genetic algorithm techniques." En Adaptive Computing in Engineering Design and Control '96 - Proceedings of the Second International Conference, I.C. Parmee (ed), p.107-113. University of Plymouth, 1996.
Ver también: Petit, Charles. ``Touched by nature: Putting evolution to work on the assembly line." U.S. News and World Report, vol.125, no.4, p.43-45 (27 de julio de 1998). Disponible en http://www.genetic-programming.com/published/usnwr072798.html

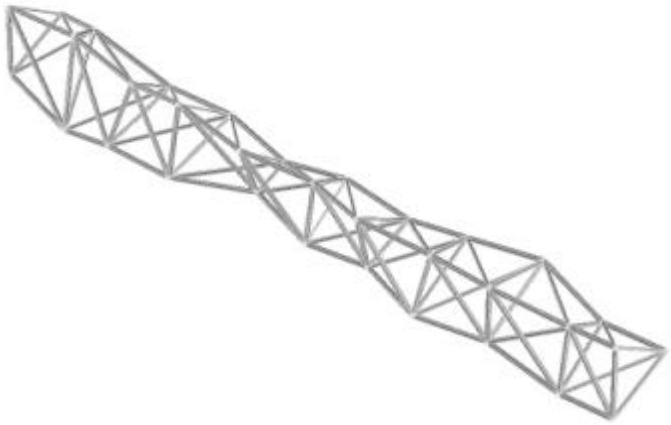


Figura 3.2 - Un brazo tridimensional optimizado genéticamente, con una respuesta mejorada a la frecuencia (adaptado de http://www.soton.ac.uk/~ajk/truss/welcome.html).

Finalmente, como informa Gibbs 1996[li], Lockheed Martin ha utilizado un algoritmo genético para producir mediante evolución una serie de maniobras para mover una nave espacial de una orientación a otra, dentro del 2% del tiempo mínimo teórico para tales maniobras. La solución evolucionada era un 10% más rápida que una solución producida manualmente por un experto para el mismo problema.

3.8.3 Astronomía y astrofísica

Charbonneau 1995[lii] sugiere la utilidad de los AGs para problemas de astrofísica, aplicándolos a tres problemas de ejemplo: 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 magnetohidrodinámico del viento solar. Son tres difíciles problemas no lineales y multidimensionales.

El algoritmo genético de Charbonneau, PIKAIA, utiliza selección generacional y proporcional a la aptitud, junto con elitismo, para asegurar que el mejor individuo se copia una vez hacia la siguiente generación sin ninguna modificación. PIKAIA tiene un ritmo de cruzamiento de 0,65 y un ritmo de mutación variable que se pone a 0,003 inicialmente y luego aumenta gradualmente,

[li] Gibbs, W. Wayt. ``Programming with primordial ooze." Scientific American, octubre de 1996, p.48-50.
[lii] Charbonneau, Paul. ``Genetic algorithms in astronomy and astrophysics." The Astrophysical Journal Supplement Series, vol.101, p.309-334 (diciembre de 1995).

mientras la población se aproxima a la convergencia, para mantener la variabilidad en el acervo genético.

En el problema de la curva de rotación galáctica, el AG produjo dos curvas, y ambas estaban bien ajustadas a los datos (un resultado común en este tipo de problema, en el que hay poco contraste entre cimas cercanas); observaciones posteriores pueden distinguir cuál es la preferible. En el problema de la serie temporal, el AG fue impresionantemente exitoso, generando un ajuste de los datos de gran calidad, aunque otros problemas más difíciles no se ajustaron tan bien (aunque, como señala Charbonneau, estos problemas son igualmente difíciles de resolver con técnicas convencionales). El artículo sugiere que un AG híbrido que emplee tanto evolución artificial como técnicas analíticas estándar, podría funcionar mejor. Finalmente, en el problema de obtener los seis parámetros críticos del viento solar, el AG determinó con éxito el valor de tres con una precisión de menos del 0,1% y los otros tres con precisiones entre el 1 y el 10%. (Aunque siempre serían preferibles unos errores experimentales menores para estos tres parámetros, Charbonneau señala que no existe ningún otro método eficiente y robusto para resolver experimentalmente un problema no lineal 6-dimensional de este tipo; un método de gradiente conjugado funciona "siempre que se pueda proporcionar un valor inicial muy acertado" (p. 323). En contraste, los AGs no requieren un conocimiento del dominio tan bien afinado).

Basándose en los resultados obtenidos hasta ahora, Charbonneau sugiere que los AGs pueden y deben encontrar uso en otros problemas difíciles de astrofísica, en particular, problemas inversos como las imágenes por Doppler y las inversiones heliosísmicas. Para terminar, Charbonneau sostiene que los AGs son un "contendiente poderoso y prometedor" (p. 324) en este campo, del que se puede esperar que complemente (no sustituya) a las técnicas tradicionales de optimización, y concluye que "el punto decisivo, si es que tiene que haber alguno, es que los algoritmos genéticos funcionan, y a menudo colosalmente bien" (p. 325).

3.8.4 Química

Un pulso láser ultracorto de alta energía puede romper moléculas complejas en moléculas más sencillas, un proceso con aplicaciones importantes en la química orgánica y la microelectrónica.

Los productos específicos de una reacción así pueden controlarse modulando la fase del pulso láser. Sin embargo, para moléculas grandes, obtener la forma del pulso deseado de manera analítica es demasiado difícil: los cálculos son demasiado complejos y las características relevantes (las superficies de energía potencial de las moléculas) no se conocen con suficiente precisión.

Assion et al. 1998[liii] resolvieron este problema utilizando un algoritmo evolutivo para diseñar la forma del pulso. En lugar de introducir información compleja, específica del problema, sobre las características cuánticas de las moléculas iniciales, para diseñar el pulso conforme a las especificaciones, el AE dispara un pulso, mide las proporciones de las moléculas producto resultantes, muta aleatoriamente las características del rayo con la esperanza de conseguir que estas proporciones se acerquen a la salida deseada, y el proceso se repite. (En lugar de afinar directamente las características del rayo láser, el AG de los autores representa a los individuos como un conjunto de 128 números, en el que cada número es un valor de voltaje que controla el índice de refracción de uno de los pixeles del modulador láser. De nuevo, no se necesita un conocimiento específico del problema sobre las propiedades del láser o de los productos de la reacción). Los autores afirman que su algoritmo, cuando se aplica a dos sustancias de muestra, "encuentra automáticamente la mejor configuración... no importa lo complicada que sea la respuesta molecular" (p. 921), demostrando un "control coherente automatizado de los productos que son químicamente diferentes uno del otro y de la molécula padre" (p. 921).

A principios y mediados de los 90, la amplia adopción de una novedosa técnica de diseño de fármacos, llamada química combinatoria, revolucionó la industria farmacéutica. Con este método, en lugar de la síntesis precisa y meticulosa de un sólo compuesto de una vez, los bioquímicos mezclan deliberadamente una gran variedad de reactivos para producir una variedad aún mayor de productos -cientos, miles o millones de compuestos diferentes en cada remesa- que luego pueden aislarse rápidamente para su actividad bioquímica. Hay dos formas de diseñar las bibliotecas de reactivos en esta técnica: diseño basado en los reactivos, que elige grupos optimizados de reactivos sin considerar qué productos saldrán como resultado, y diseño basado en los productos, que selecciona los reactivos que producirán con mayor probabilidad los productos con las propiedades deseadas. El diseño basado en los productos es más difícil y

[liii] Assion, A., T. Baumert, M. Bergt, T. Brixner, B. Kiefer, V. Seyfried, M. Strehle y G. Gerber. ``Control of chemical reactions by feedback-optimized phase-shaped femtosecond laser pulses." Science, vol.282, p.919-922 (30 de octubre de 1998).

complejo, pero se ha demostrado que genera bibliotecas combinatorias mejores y más diversas, y tiene más probabilidades de ofrecer un resultado útil.

En un artículo patrocinado por el departamento de investigación y desarrollo de GlaxoSmithKline, Gillet 2002[liv] describe el uso de un algoritmo genético multiobjetivo para el diseño basado en los productos de bibliotecas combinatorias. Al elegir los componentes que van en una biblioteca particular, deben considerarse características como la diversidad y peso molecular, el coste de los suministros, la toxicidad, la absorción, la distribución y el metabolismo. Si el objetivo es encontrar moléculas similares a una molécula existente con una función conocida (un método común en el diseño de nuevos fármacos), también se puede tener en cuenta la similaridad estructural. Este artículo presenta un enfoque multiobjetivo, donde puede desarrollarse un conjunto de resultados paretianos que maximicen o minimicen cada uno de estos objetivos. El autor concluye diciendo que el AG fue capaz de satisfacer simultáneamente los criterios de diversidad molecular y eficiencia sintética máxima, y también fue capaz de encontrar moléculas parecidas a un fármaco que eran "muy similares a las moléculas objetivo dadas, tras explorar una fracción muy pequeña del espacio de búsqueda total" (p. 378).

En un artículo relacionado, Glen y Payne 1995[lv] describen el uso de algoritmos genéticos para diseñar automáticamente moléculas nuevas desde cero que se ajustan a un conjunto de especificaciones dado. Dada una población inicial, bien generada aleatoriamente o utilizando la sencilla molécula del etano como semilla, el AG añade, elimina y altera aleatoriamente átomos y fragmentos moleculares con el objetivo de generar moléculas que se ajusten a los requisitos dados. El AG puede optimizar simultáneamente un gran número de objetivos, incluyendo el peso molecular, el volumen molecular, el número de enlaces, el número de centros quirales, el número de átomos, el número de enlaces rotables, la polarizabilidad, el momento dipolar, etcétera, para producir moléculas candidatas con las propiedades deseadas. Basándose en pruebas experimentales, incluyendo un difícil problema de optimización que implicaba la generación de moléculas con propiedades similares a la ribosa (un componente del azúcar imitado a menudo en los fármacos antivirales), los autores concluyen que el AG es un "excelente generador de ideas" (p. 199) que ofrece "propiedades de optimización rápidas y poderosas" y puede generar "un

[liv] Gillet, Valerie. ``Reactant- and product-based approaches to the design of combinatorial libraries." Journal of Computer-Aided Molecular Design, vol.16, p.371-380 (2002).

[lv] Glen, R.C. y A.W.R. Payne. ``A genetic algorithm for the automated generation of molecules within constraints." Journal of Computer-Aided Molecular Design, vol.9, p.181-202 (1995)

conjunto diverso de estructuras posibles" (p. 182). Continúan afirmando: "Es de interés especial la poderosa capacidad de optimización del algoritmo genético, incluso con tamaños de población relativamente pequeños" (p. 200). Como prueba de que estos resultados no son simplemente teóricos, Lemley 2001[lvi] informa de que la empresa Unilever ha utilizado algoritmos genéticos para diseñar nuevos componentes antimicrobianos para su uso en productos de limpieza, algo que ha patentado.

3.8.5 Ingeniería eléctrica

Una matriz de puertas programable en campo (Field Programmable Gate Array, o FPGA), es un tipo especial de placa de circuito con una matriz de celdas lógicas, cada una de las cuales puede actuar como cualquier tipo de puerta lógica, interconectado con conexiones flexibles que pueden conectar celdas. Estas dos funciones se controlan por software, así que simplemente cargando un programa especial en la placa, puede alterarse al vuelo para realizar las funciones de cualquier dispositivo de hardware de la amplia variedad existente.

El Dr. Adrian Thompson ha explotado este dispositivo, en conjunción con los principios de la evolución, para producir un prototipo de circuito reconocedor de voz que puede distinguir y responder a órdenes habladas utilizando sólo 37 puertas lógicas -una tarea que se habría considerado imposible para cualquier ingeniero humano. Generó cadenas aleatorias de bits de ceros y unos y las utilizó como configuraciones de la FPGA, seleccionando los individuos más aptos de cada generación, reproduciéndolos y mutándolos aleatoriamente, intercambiando secciones de su código y pasándolo hacia la siguiente ronda de selección. Su objetivo era evolucionar un dispositivo que pudiera en principio discriminar entre tonos de frecuencias distintas (1 y 10 kilohercios), y luego distinguir entre las palabras habladas "go" (adelante) y "stop" (para).

Su objetivo se alcanzó en 3.000 generaciones, pero el éxito fue mayor de lo que había anticipado. El sistema que evolucionó utilizaba muchas menos celdas que cualquier cosa que pudiera haber diseñado un ingeniero humano, y ni siquiera necesita del componente más crítico de los sistemas diseñados por humanos -un reloj. ¿Cómo funcionaba? Thompson no tiene ni idea, aunque ha rastreado la señal de entrada a través de un complejo sistema de bucles re alimentados del

[lvi] Lemley, Brad. ``Machines that think." Discover, enero de 2001, p.75-79.

circuito evolucionado. De hecho, de las 37 puertas lógicas que utiliza el producto final, cinco de ellas ni siquiera están conectadas al resto del circuito de ninguna manera -pero si se les retira la alimentación eléctrica, el circuito deja de funcionar. Parece que la evolución ha explotado algún sutil efecto electromagnético de estas celdas para alcanzar su solución, pero el funcionamiento exacto de la compleja e intrincada estructura evolucionada sigue siendo un misterio (Davidson 1997[lvii]).

Altshuler y Linden 1997[lviii] utilizaron un algoritmo genético para evolucionar antenas de alambre con propiedades especificadas a priori. Los autores señalan que el diseño de tales antenas es un proceso impreciso, comenzando con las propiedades deseadas y luego determinando la forma de la antena mediante "conjeturas... intuición, experiencia, ecuaciones aproximadas o estudios empíricos" (p. 50). Esta técnica requiere mucho tiempo, a menudo no produce resultados óptimos y tiende a funcionar bien sólo con diseños simétricos y relativamente simples. En contraste, con el método del algoritmo genético, el ingeniero especifica las propiedades electromagnéticas de la antena, y el AG sintetiza automáticamente una configuración que sirva.

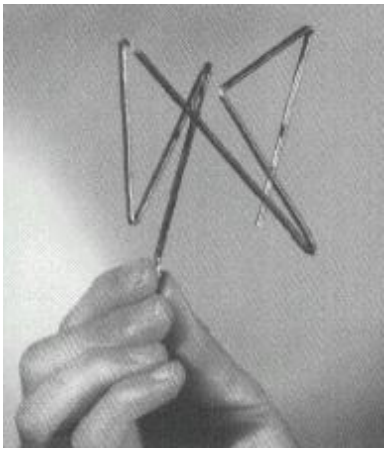


Figura 3.3 - Una antena genética de alambre doblado (de Altshuler y Linden 1997).

Altshuler y Linden utilizaron su AG para diseñar una antena de siete segmentos polarizada circularmente con una cobertura hemisférica; el resultado se muestra a la izquierda. Cada individuo del AG consistía en un cromosoma binario que especificaba las coordenadas tridimensionales de cada extremo final de cada alambre. La aptitud se evaluaba simulando a cada candidato de acuerdo con un código de cableado electromagnético, y el individuo mejor de cada

[lvii] Davidson, Clive. ``Creatures from primordial silicon." New Scientist, vol.156, no.2.108, p.30-35 (15 de noviembre de 1997) - http://www.newscientist.com/article/mg15621085.000.html

[lviii] Altshuler, Edward y Derek Linden. ``Design of a wire antenna using a genetic algorithm." Journal of Electronic Defense, vol.20, no.7, p.50-52 (julio de 1997).

ronda se construía y probaba. Los autores describen la forma de esta antena, que no se parece a las antenas tradicionales y carece de una simetría obvia, como "inusualmente extraña" y "anti intuitiva" (p. 52), aunque tenía un patrón de radiación casi uniforme y con un gran ancho de banda tanto en la simulación como en la prueba experimental, adecuándose excelentemente a la especificación inicial. Los autores concluyen que un método basado en algoritmos genéticos para diseñar antenas se muestra "excepcionalmente prometedor". "... este nuevo procedimiento de diseño es capaz de encontrar antenas genéticas capaces de resolver de manera efectiva difíciles problemas de antenas, y será especialmente útil en situaciones en las que los diseños existentes no sean adecuados" (p. 52).

3.8.6 Mercados financieros

Mahfoud y Mani 1996[lix] utilizaron un algoritmo genético para predecir el rendimiento futuro de 1.600 acciones ofertadas públicamente. Concretamente, al AG se le asignó la tarea de predecir el beneficio relativo de cada acción, definido como el beneficio de esa acción menos el beneficio medio de las 1.600 acciones a lo largo del periodo de tiempo en cuestión, 12 semanas (un cuarto del calendario) en el futuro. Como entrada, al AG se le proporcionaron datos históricos de cada acción en forma de una lista de 15 atributos, como la relación precio-beneficio y el ritmo de crecimiento, medidos en varios puntos del tiempo pasado; se le pidió al AG que evolucionara un conjunto de reglas si/entonces para clasificar cada acción y proporcionar, como salida, una recomendación sobre qué hacer con respecto a la acción (comprar, vender o ninguna predicción) y un pronóstico numérico del beneficio relativo. Los resultados del AG fueron comparados con los de un sistema establecido, basado en una red neuronal, que los autores habían estado utilizando para pronosticar los precios de las acciones y administrar las carteras de valores durante tres años. Por supuesto, el mercado de valores es un sistema extremadamente ruidoso y no lineal, y ningún mecanismo predictivo puede ser correcto el 100% del tiempo; el reto consiste en encontrar un predictor que sea preciso más de la mitad de las veces.

[lix] Mahfoud, Sam y Ganesh Mani. ``Financial forecasting using genetic algorithms." Applied Artificial Intelligence, vol.10, no.6, p.543-565 (1996)

En el experimento, el AG y la red neuronal hicieron pronósticos al final de la semana para cada una de las 1.600 acciones, durante doce semanas consecutivas. Doce semanas después de cada predicción, se comparó el rendimiento verdadero con el beneficio relativo predicho. Globalmente, el AG superó significativamente a la red neuronal: en una ejecución de prueba, el AG predijo correctamente la dirección de una acción el 47,6% de las veces, no hizo predicción el 45,8% de las veces y realizó una predicción incorrecta sólo un 6.6% de las veces, una precisión predictiva total de un 87,8%. Aunque la red neuronal realizó predicciones precisas más a menudo, también hizo predicciones erróneas más a menudo (de hecho, los autores especulan que la mayor capacidad del AG para no realizar predicciones cuando los datos eran dudosos fue un factor de su éxito; la red neuronal siempre produce una predicción a menos que sea restringida explícitamente por el programador). En el experimento de las 1.600 acciones, el AG produjo un beneficio relativo de un +5,47%, contra el +4,40% de la red neuronal -una diferencia estadísticamente significativa. De hecho, el AG también superó significativamente a tres índices bursátiles importantes -el S&P 500, el S&P 400 y el Russell 2000- en este periodo; la casualidad fue excluida como causa de este resultado con un margen de confianza de un 95%. Los autores atribuyen este convincente éxito a la capacidad del algoritmo genético de percatarse de relaciones no lineales difícilmente evidentes para los observadores humanos, además del hecho de que carece del "prejuicio contra las reglas anti intuitivas y contradictorias" (p. 562) de los expertos humanos.

Andreou, Georgopoulos y Likothanassis 2002[lx] lograron un éxito similar utilizando algoritmos genéticos híbridos para evolucionar redes neuronales que predijeran los tipos de cambio de monedas extranjeras hasta un mes en el futuro. Al contrario que en el ejemplo anterior, donde competían AGs y redes neuronales, aquí los dos trabajaron conjuntamente: el AG evolucionó la arquitectura (número de unidades de entrada, número de unidades ocultas y la estructura de enlaces entre ellas) de la red, que luego era entrenada por un algoritmo de filtro.

Se le proporcionaron al algoritmo 1.300 valores brutos diarios de cinco divisas como información histórica -el dólar estadounidense, el marco alemán, el franco francés, la libra

[lx] Andreou, Andreas, Efstratios Georgopoulos y Spiridon Likothanassis. ``Exchange-rates forecasting: A hybrid algorithm based on genetically optimized adaptive neural networks." Computational Economics, vol.20, no.3, p.191-210 (diciembre de 2002).

esterlina y el dracma griego- y se le pidió que predijera sus valores futuros para los 1, 2, 5 y 20 días posteriores. El rendimiento del AG híbrido mostró, en general, un "nivel excepcional de precisión" (p. 200) en todos los casos probados, superando a otros varios métodos, incluyendo a las redes neuronales en solitario. Los autores concluyen que "se ha logrado un excepcional éxito predictivo tanto con un horizonte predictivo de un paso como de varios pasos" (p. 208) -de hecho, afirman que sus resultados son mejores con diferencia que cualquier estrategia predictiva relacionada que se haya aplicado en esta serie de datos u otras divisas.

La utilización de los AGs en los mercados financieros ha empezado a extenderse en las empresas de corretaje bursátil del mundo real. Naik 1996[lxi] informa de que LBS Capital Management, una empresa estadounidense con sede en Florida, utiliza algoritmos genéticos para escoger las acciones de los fondos de pensiones que administra. Coale 1997[lxii] y Begley y Beals 1995[lxiii] informan de que First Quadrant, una empresa de inversiones de California que mueve más de 2.200 millones de dólares, utiliza AGs para tomar decisiones de inversión en todos sus servicios financieros. Su modelo evolucionado gana, de media, 225 dólares por cada 100 dólares invertidos durante seis años, en contraste con los 205 dólares de otros tipos de sistemas de modelos.

3.8.7 Juegos

Una de las demostraciones más novedosas y persuasivas de la potencia de los algoritmos genéticos la presentaron Chellapilla y Fogel 2001[lxiv], que utilizaron un AG para evolucionar redes neuronales que pudieran jugar a las damas. Los autores afirman que una de las mayores dificultades en este tipo de problemas relacionados con estrategias es el problema de la asignación de crédito -en otras palabras, ¿cómo escribir una función de aptitud? Se ha creído

[lxi] Naik, Gautam. ``Back to Darwin: In sunlight and cells, science seeks answers to high-tech puzzles." The Wall Street Journal, 16 de enero de 1996, p. A1
[lxii] Coale, Kristi. ``Darwin in a box." Wired News, 14 de julio de 1997. Disponible en http://www.wired.com/news/technology/0,1282,5152,00.html
[lxiii] Begley, Sharon y Gregory Beals. ``Software au naturel." Newsweek, 8 de mayo de 1995, p.70.
[lxiv] Chellapilla, Kumar y David Fogel. ``Evolving an expert checkers playing program without using human expertise." IEEE Transactions on Evolutionary Computation, vol.5, no.4, p.422-428 (agosto de 2001) – Disponible en http://www.natural-selection.com/Library/2001/IEEE-TEVC.pdf

ampliamente que los criterios simples de ganar, perder o empatar no proporcionan la suficiente información para que un algoritmo genético averigüe qué constituye el buen juego.

En este artículo, Chellapila y Fogel echan por tierra esa suposición. Dados sólo las posiciones espaciales de las piezas en el tablero y el número total de piezas que posee cada jugador, fueron capaces de evolucionar un programa de damas que jugaba a un nivel competitivo con expertos humanos, sin ninguna información de entrada inteligente acerca de lo que constituye el buen juego -es más, ni siquiera se les dijo a los individuos del algoritmo evolutivo cuál era el criterio para ganar, ni se les dijo el resultado de ningún juego.

En la representación de Chellapilla y Fogel, el estado del juego estaba representado por una lista numérica de 32 elementos, en donde cada posición de la lista correspondía a una posición disponible en el tablero. El valor de cada posición era 0 para una casilla desocupada, -1 si esa casilla estaba ocupada por una pieza enemiga, +1 si la casilla estaba ocupada por una de las piezas del programa, y -K o +K si la casilla estaba ocupada por una dama enemiga o amiga. (El valor de K no se especificaba a priori, sino que, de nuevo, era determinado por la evolución durante el curso del algoritmo). Acompañando a todo esto había una red neuronal con múltiples capas de procesamiento y una capa de entrada con un nodo para cada una de las 4x4, 5x5, 6x6, 7x7 y 8x8 posibles casillas del tablero. La salida de la red neuronal para una colocación de las piezas dada era un valor entre -1 y +1, que indicaba cómo de buena le parecía esa posición. Para cada movimiento, se le presentaba a la red neuronal un árbol de juego que contenía todos los movimientos posibles hasta cuatro turnos en el futuro, y el movimiento se decidía basándose en qué rama del árbol producía los mejores resultados.

El algoritmo evolutivo comenzó con una población de 15 redes neuronales con pesos y tendencias, generados aleatoriamente, asignados a cada nodo y conexión; luego, cada individuo se reprodujo una vez, generando una descendencia con variaciones en los valores de la red. Luego estos 30 individuos compitieron por la supervivencia jugando entre ellos; cada individuo compitió en cada turno con 5 oponentes elegidos aleatoriamente. Se otorgó 1 punto a cada victoria y se descontaban 2 puntos por cada derrota. Se seleccionaron los 15 mejores jugadores en relación a su puntuación total, y el proceso se repitió. La evolución continuó durante 840 generaciones más (aproximadamente seis meses de tiempo de computación).

Clase	Puntuación

Gran Maestro	+2.400
Maestro	2.200-2.399
Experto	2.000-2.199
Clase A	1.800-1.999
Clase B	1.600-1.799
Clase C	1.400-1.599
Clase J	<200

El mejor individuo que surgió de esta selección fue inscrito como competidor en la página web de juegos http://www.zone.com. Durante un periodo de dos meses, jugó contra 165 oponentes humanos que componían una gama de niveles altos, desde clase C a maestros, de acuerdo con el sistema de clasificaciones de la Federación de Ajedrez de Estados Unidos (mostrado a la izquierda, con algunos rangos omitidos en aras de claridad). De estas partidas, la red neuronal ganó 94, perdió 39 y empató 32; en base a las clasificaciones de los oponentes en estas partidas, la red neuronal evolucionada era equivalente a un jugador con una puntuación media de 2.045,85, colocándola en el nivel experto -una clasificación superior a la del 99,61% de los 80.000 jugadores registrados en la página web. Una de las victorias más significativas de la red neuronal fue cuando venció a un jugador clasificado en la posición 98 de todos los jugadores registrados, cuya puntuación estaba tan sólo 27 puntos por debajo del nivel de maestro.

Las pruebas realizadas con un sencillo programa diferencial en las piezas (que basa sus movimientos solamente en la diferencia entre el número de piezas que quedan en cada lado) con una capacidad de anticipación de 8 movimientos demostró que la red neuronal era significativamente superior, con una puntuación de más de 400 puntos por encima. “Un programa que se basa sólo en el número de piezas y en una búsqueda de ocho capas vencerá a muchas personas, pero no es un experto. La mejor red neuronal evolucionada sí lo es” (p. 425). Aunque podía buscar posiciones dos movimientos más lejos que la red neuronal, el programa diferencial en las piezas perdió contundentemente 8 de 10 partidas. Esto demuestra concluyentemente que la red neuronal evolucionada no sólo está contando piezas, sino que de alguna manera procesa las características espaciales del tablero para decidir sus movimientos. Los autores señalan que los oponentes de zone.com que los movimientos de la red neuronal eran

“extraños”, pero su nivel global de juego fue descrito como “muy duro” o con términos elogiosos similares.

Para probar más a la red neuronal evolucionada (a la que los autores nombraron “Anaconda” porque a menudo ganaba restringiendo la movilidad de sus oponentes), jugó contra un programa de damas comercial, el Hoyle Classic Games, distribuido por Sierra Online (Chellapilla y Fogel 2000[lxv]). Este programa viene con un surtido de personajes incorporados, cada uno con un nivel de juego distinto. Anaconda se puso a prueba con tres personajes (“Beatrice”, “Natasha” y “Leopold”) designados como jugadores expertos, jugando una partida con las rojas y otra partida con las blancas contra cada uno de ellos con una capacidad de anticipación de 6 movimientos. Aunque los autores dudaban de que esta profundidad de anticipación pudiera darla a Anaconda la capacidad de juego experto que demostró anteriormente, consiguió seis victorias seguidas de las seis partidas jugadas. Basándose en este resultado, los autores expresaron escepticismo sobre si el software Hoyle jugaba al nivel que anunciaba, ¡aunque debe señalarse que llegaron a esta conclusión basándose solamente en la facilidad con la que Anaconda le venció!

La prueba definitiva de Anaconda se detalla en Chellapilla y Fogel 2002[lxvi], cuando la red neuronal evolucionada jugó contra el mejor jugador de damas del mundo: Chinook, un programa diseñado principalmente por el Dr. Jonathan Schaeffer, de la Universidad de Alberta. Con una puntuación de 2.814 en 1996 (mientras que sus competidores humanos más cercanos andan por los 2.600), Chinook incorpora un libro de movimientos de apertura proporcionado por grandes maestros humanos, un conjunto sofisticado de algoritmos de juego para la parte central de la partida, y una base de datos completa de todos los movimientos posibles cuando quedan en el tablero 10 piezas o menos, de manera que nunca comete un error durante un final de partida. Se invirtió una cantidad enorme de inteligencia y experiencia humana en el diseño de este programa.

Chellapilla y Fogel enfrentaron a Anaconda y Chinook en un torneo de 10 partidas, con Chinook jugando al nivel de 5 capas de anticipación, aproximándolo más o menos al nivel de maestro. Chinook ganó esta competición, cuatro victorias a dos, con cuatro empates. (Curiosamente, como señalan los autores, en dos de las partidas que acabaron con empate, Anaconda lideraba con

[lxv] Chellapilla, Kumar y David Fogel. ``Anaconda defeats Hoyle 6-0: a case study competing an evolved checkers program against commercially available software." En Proceedings of the 2000 Congress on Evolutionary Computation, p.857-863. IEEE Press, 2000

[lxvi] Chellapilla, Kumar y David Fogel. ``Verifying Anaconda's expert rating by competing against Chinook: experiments in co-evolving a neural checkers player." Neurocomputing, vol.42, no.1-4, p.69-86 (enero de 2002)

cuatro damas mientras que Chinook tenía tres. Además, una de las victorias de Chinook vino tras una serie de movimientos con búsqueda de 10 capas sacados de su base de datos de finales de partida; unos movimientos que Anaconda, con una anticipación de 8 capas, no pudo anticipar. Si Anaconda hubiera tenido acceso a una base de datos de finales de partida de la misma calidad de la de Chinook, el resultado del torneo bien podría haber sido el de victoria para Anaconda, cuatro a tres). Estos resultados "proporcionan un buen sustento a la puntuación de experto que se ganó Anaconda en www.zone.com" (p. 76), con una puntuación global de 2.030-2.055, comparable a la puntuación de 2.045 que ganó jugando contra humanos. Aunque Anaconda no es un jugador invulnerable, es capaz de jugar competitivamente en el nivel experto y comportarse ante una variedad de jugadores de damas humanos extremadamente hábiles. Cuando uno considera los criterios de aptitud tan sencillos con los que se obtuvieron estos resultados, el surgimiento de Anaconda proporciona una espectacular corroboración del poder de la evolución.

3.8.8 Geofísica

Sambridge y Gallaguer 1993[lxvii] utilizaron un algoritmo genético para los hipocentros de los terremotos basándose en datos sismológicos. (El hipocentro es el punto bajo la superficie terrestre en el que se origina un terremoto. El epicentro es el punto de la superficie directamente encima del hipocentro). Esto es una tarea sumamente compleja, ya que las propiedades de las ondas sísmicas dependen de las propiedades de las capas de roca a través de las que viajan. El método tradicional para localizar el hipocentro se basa en lo que se conoce como algoritmo de inversión sísmico, que empieza con la mejor estimación de la ubicación, calcula las derivadas del tiempo de viaje de la onda con respecto al punto de origen, y realiza una operación de matriz para proporcionar una ubicación actualizada. Este proceso se repite hasta que se alcanza una solución aceptable. (Glenn Morton 2003[lxviii], proporciona más información). Sin embargo, este método requiere información diferencial y es propenso a quedar atrapado en óptimos locales.
Un algoritmo de localización que no dependa de información diferencial o modelos de velocidad puede evitar esta deficiencia calculando sólo el problema directo -la diferencia entre los tiempos

[lxvii] Sambridge, Malcolm y Kerry Gallagher. ``Earthquake hypocenter location using genetic algorithms." Bulletin of the Seismological Society of America, vol.83, no.5, p.1.467-1.491 (octubre de 1993)
[lxviii] The Design Inference in Geology
Post of the Month: November 2003
by Glenn Morton – Disponible en http://www.talkorigins.org/origins/postmonth/nov03.html

de llegada de la onda observados y predichos para distintas localizaciones del hipocentro. Sin embargo, un método de búsqueda exhaustivo basado en este método sería demasiado costoso computacionalmente. Éste, por supuesto, es precisamente el tipo de problema de optimización en el que destacan los algoritmos genéticos. Como todos los AGs, el propuesto por el artículo citado es paralelo en naturaleza -en lugar de mover un solo hipocentro más y más cerca hacia la solución, comienza con una nube de hipocentros potenciales que encoge con el tiempo hasta converger en una sola solución. Los autores afirman que su método "puede localizar rápidamente soluciones casi óptimas sin una búsqueda exhaustiva del espacio de parámetros" (p. 1.467), muestra "un comportamiento muy organizado que produce una búsqueda eficiente" y es "un compromiso entre la eficiencia de los métodos basados en derivadas y la robustez de una búsqueda exhaustiva completamente no lineal" (p. 1.469). Los autores concluyen que su algoritmo genético es "eficiente para una verdadera optimización global" (p. 1.488) y "una herramienta nueva y poderosa para realizar búsquedas robustas de hipocentros" (p. 1.489).

3.8.9 Ingeniería de materiales

Giro, Cyrillo y Galvão 2002[lxix] utilizaron algoritmos genéticos para diseñar polímeros conductores de electricidad basados en el carbono, conocidos como polianilinas. Estos polímeros, un tipo de material sintético inventado recientemente, tienen "grandes aplicaciones tecnológicas potenciales" y podrían abrir la puerta a "nuevos fenómenos físicos fundamentales" (p. 170). Sin embargo, debido a su alta reactividad, los átomos de carbono pueden formar un número virtualmente infinito de estructuras, haciendo que la búsqueda de nuevas moléculas con propiedades interesantes sea del todo imposible. En este artículo, los autores aplican un enfoque basado en AGs a la tarea de diseñar moléculas nuevas con propiedades especificadas a priori, comenzando con una población de candidatos iniciales generada aleatoriamente. Concluyen que su metodología puede ser una "herramienta muy efectiva" (p. 174) para guiar a los investigadores en la búsqueda de nuevos compuestos y es lo suficientemente general para que pueda extenderse al diseño de nuevos materiales que pertenezcan virtualmente a cualquier tipo de molécula.

[lxix] Giro, R., M. Cyrillo y D.S. Galvão. ``Designing conducting polymers using genetic algorithms." Chemical Physics Letters, vol.366, no.1-2, p.170-175 (25 de noviembre de 2002).

Weismann, Hammel, y Bäck 1998[lxx] aplicaron algoritmos evolutivos a un problema industrial "no trivial" (p. 162): el diseño de revestimientos ópticos multicapa para filtros que reflejan, transmiten o absorben luz de frecuencias especificadas. Estos revestimientos se utilizan en la fabricación de gafas de sol, por ejemplo, o discos compactos. Su fabricación es una tarea precisa: las capas deben colocarse en una secuencia particular y con un grosor particular para producir el resultado deseado, y las variaciones incontrolables del entorno de fabricación, como la temperatura, la polución o la humedad, pueden afectar al rendimiento del producto acabado. Muchos óptimos locales no son robustos ante estas variaciones, lo que significa que una mayor calidad del producto se paga con una tasa mayor de desviaciones indeseadas. El problema particular considerado en este artículo también tenía múltiples criterios: además de la reluctancia, también se consideró la composición espectral (color) de la luz reflejada.

El AE actuó variando el número de capas de revestimiento y el grosor de cada una de ellas, y produjo diseños que eran "sustancialmente más robustos a la variación de parámetros" (p. 166) y tenían un rendimiento medio mayor que los métodos tradicionales. Los autores concluyen que "los algoritmos evolutivos pueden competir e incluso superar a los métodos tradicionales" (p. 167) de diseño de revestimientos ópticos multicapa, sin tener que incorporar un conocimiento específico del dominio en la función de búsqueda y sin tener que alimentar a la población con buenos diseños iniciales.

Es digno de mención otro uso de los AGs en el campo de la ingeniería de materiales: Robin et al. 2003[lxxi] utilizaron AGs para diseñar patrones de exposición para un haz de electrones de litografía, utilizado para grabar estructuras a una escala menor a la del micrómetro en circuitos integrados. Diseñar estos patrones es una tarea muy difícil; es pesado y costoso determinarlos experimentalmente, pero la alta dimensionalidad del espacio de búsqueda frustra a la mayoría de los algoritmos de búsqueda. Deben optimizarse simultáneamente hasta 100 parámetros para controlar el haz de electrones y evitar la dispersión y efectos de proximidad que arruinarían las estructuras finas que se estén esculpiendo. El problema directo -determinar la estructura resultante como función de la cantidad de electrones- es sencillo y fácil de simular, pero el

[lxx] Weismann, Dirk, Ulrich Hammel, y Thomas Bäck. ``Robust design of multilayer optical coatings by means of evolutionary algorithms." IEEE Transactions on Evolutionary Computation, vol.2, no.4, p.162-167 (noviembre de 1998).

[lxxi] Robin, Franck, Andrea Orzati, Esteban Moreno, Otte Homan, y Werner Bachtold. ``Simulation and evolutionary optimization of electron-beam lithography with genetic and simplex-downhill algorithms." IEEE Transactions on Evolutionary Computation, vol.7, no.1, p.69-82 (febrero de 2003).

problema inverso de determinar la cantidad de electrones para producir una estructura dada, que es lo que se pretende resolver aquí, es mucho más difícil y no existe una solución determinista.
Se aplicaron algoritmos genéticos a este problema, ya que "se sabe que son capaces de encontrar soluciones buenas a problemas muy complejos de alta dimensionalidad" (p. 75) sin necesidad de proporcionarles información específica del dominio acerca de la topología del paisaje de búsqueda. Los autores del artículo emplearon un AG de estado estacionario con selección por rueda de ruleta en una simulación por computador, que produjo unos patrones de exposición "muy bien optimizados" (p. 77). En contraste, se utilizó un tipo de trepacolinas conocido como algoritmo bajacolinas-simplex (simplex-downhill) en el mismo problema, sin éxito; el método BS quedaba rápidamente atrapado en óptimos locales de los que no podía escapar, produciendo soluciones de poca calidad. Un híbrido entre los métodos del AG y el BS tampoco pudo mejorar los resultados ofrecidos por el AG en solitario.

3.8.10 Matemáticas y algoritmia

Aunque algunas de las aplicaciones más prometedoras y las demostraciones más convincentes de la potencia de los AGs se encuentran en el campo de la ingeniería de diseño, también son relevantes en problemas "puramente" matemáticos. Haupt y Haupt 1998[lxxii] (p. 140) describen el uso de AGs para resolver ecuaciones de derivadas parciales no lineales de alto orden, normalmente encontrando los valores para los que las ecuaciones se hacen cero, y dan como ejemplo una solución casi perfecta para los coeficientes de la ecuación de quinto orden conocida como Super Korteweg-de Vries.
Ordenar una lista de elementos es una tarea importante en la informática, y una red de ordenación es una manera eficiente de conseguirlo. Una red de ordenación es una lista fija de comparaciones realizadas en un conjunto de un tamaño dado; en cada comparación se comparan dos elementos y se intercambian si no están en orden. Koza et al. 1999[lxxiii] (p. 952) utilizaron programación genética para evolucionar redes de ordenación mínimas para conjuntos de 7 elementos (16 comparaciones), conjuntos de 8 elementos (19 comparaciones) y conjuntos de 9

[lxxii] Haupt, Randy y Sue Ellen Haupt. Practical Genetic Algorithms. John Wiley & Sons, 1998
[lxxiii] Koza, John, Forest Bennett, David Andre y Martin Keane. Genetic Programming III: Darwinian Invention and Problem Solving. Morgan Kaufmann Publishers, 1999

elementos (25 comparaciones). Mitchell 1996[lxxiv], p. 21, describe el uso de algoritmos genéticos por W. Daniel Hillis para encontrar una red de ordenación de 61 comparaciones para un conjunto de 16 elementos, sólo un paso más allá de la más pequeña conocida. Este último ejemplo es especialmente interesante por las dos innovaciones que utiliza: cromosomas diploides y, más notablemente, coevolución de huésped/parásito. Tanto las redes de búsqueda como los casos de prueba evolucionaron conjuntamente; se les otorgó mayor aptitud a las redes de ordenación que ordenaran correctamente un mayor número de casos de prueba, mientras que se les otorgó mayor aptitud a los casos de prueba que pudieran "engañar" a un mayor número de redes de búsqueda para que ordenaran incorrectamente. El AG con coevolución rindió significativamente mejor que el mismo AG sin ella.

Un ejemplo final de AG digno de mención en el campo de la algoritmia puede encontrarse en Koza et al. 1999[lxxv], que utilizó programación genética para descubrir una regla para el problema de clasificación por mayoría en autómatas celulares de una dimensión, una regla mejor que todas las reglas conocidas escritas por humanos. Un autómata celular de una dimensión puede imaginarse como una cinta finita con un número dado de posiciones (celdas) en ella, cada una de las cuales puede contener el estado 0 o el estado 1. El autómata se ejecuta durante un número dado de pasos temporales; en cada paso, cada celda adquiere un nuevo valor basado en su valor anterior y el valor de sus vecinos más cercanos. (El Juego de la Vida es un autómata celular bidimensional). El problema de la clasificación por mayoría implica encontrar una tabla de reglas tal que si más de la mitad de las celdas de la cinta son 1 inicialmente, todas las celdas se ponen a 1; de lo contrario, todas las celdas se ponen a 0. El reto consiste en el hecho de que cualquier celda individual sólo tiene acceso a información acerca de sus vecinos más cercanos; por lo tanto, los conjuntos de reglas buenos deben encontrar de algún modo una manera de transmitir información sobre regiones distantes de la cinta.

Se sabe que no existe una solución perfecta a este problema -ningún conjunto de reglas puede clasificar con precisión todas las configuraciones iniciales posibles-, pero durante los últimos veinte años ha habido una larga sucesión de soluciones cada vez mejores. En 1978, tres investigadores desarrollaron la famosa regla GKL, que clasifica correctamente un 81,6% de los posibles estados iniciales. En 1993, se descubrió una regla mejor con una precisión de un 81,8%;

[lxxiv] Mitchell, Melanie. An Introduction to Genetic Algorithms. MIT Press, 1996.

[lxxv] Koza, John, Forest Bennett, David Andre y Martin Keane. Genetic Programming III: Darwinian Invention and Problem Solving. Morgan Kaufmann Publishers, 1999

en 1995 se encontró otra regla con una precisión de un 82,178%. Todas estas reglas requirieron para su desarrollo de un esfuerzo significativo por parte de humanos inteligentes y creativos. En contraste, la mejor regla descubierta mediante programación genética, descrito en Koza et al. 1999[lxxvi], p. 973, tiene una precisión total de 82,326% -mejor que cualquiera de las soluciones humanas desarrolladas durante las dos últimas décadas. Los autores señalan que sus nuevas reglas son cualitativamente distintas a las reglas publicadas con anterioridad, al emplear representaciones internas muy detalladas de la densidad de estados y conjuntos intrincados de señales para comunicar información a largas distancias.

3.8.11 Ejército y cumplimiento de la ley

Kewley y Embrechts 2002[lxxvii] utilizaron algoritmos genéticos para evolucionar planes tácticos para las batallas militares. Los autores señalan que "planear una batalla militar táctica es una tarea compleja multidimensional que a menudo atormenta a los profesionales experimentados" (p. 163), no sólo porque este tipo de decisiones a menudo se toman bajo condiciones de mucho estrés, sino también porque hasta los planes más sencillos requieren tomar en cuenta un gran número de variables y consecuencias: minimizar las bajas amigas, maximizar las bajas enemigas, controlar el terreno deseado, conservar recursos, etcétera. Los planificadores humanos tienen dificultades al tratar con las complejidades de esta tarea y a menudo deben recurrir a métodos "rápidos y sucios", como hacer lo que funcionase la última vez.

Para superar estas dificultades, los autores del artículo citado desarrollaron un algoritmo genético para automatizar la creación de planes de batalla, en conjunción con un programa gráfico de simulación de batallas. El comandante introduce el resultado deseado y el AG evoluciona automáticamente un plan de batalla; en la simulación utilizada, se tomaron en cuenta factores como la topografía del terreno, la cobertura vegetal, la velocidad del movimiento de tropas, y la precisión en los disparos. En este experimento también se utilizó la coevolución para mejorar la calidad de las soluciones: los planes de batalla de las fuerzas enemigas evolucionaron simultáneamente con los planes amigos, forzando al AG a corregir cualquier debilidad de su plan

[lxxvi] Koza, John, Forest Bennett, David Andre y Martin Keane. Genetic Programming III: Darwinian Invention and Problem Solving. Morgan Kaufmann Publishers, 1999

[lxxvii] Kewley, Robert y Mark Embrechts. ``Computational military tactical planning system." IEEE Transactions on Systems, Man and Cybernetics, Part C - Applications and Reviews, vol.32, no.2, p.161-171 (mayo de 2002).

que pudiese explotar el enemigo. Para medir la calidad de las soluciones producidas por el AG, se compararon con planes de batalla para el mismo escenario producidos por un grupo de "expertos militares experimentados... considerados muy capaces de desarrollar planes de acción para el tamaño de las fuerzas utilizadas en este experimento" (p. 166). Estos avezados expertos desarrollaron su propio plan y, cuando la solución del AG estuvo acabada, se les dio la oportunidad de examinarla y modificarla como vieran conveniente. Finalmente, todos los planes se ejecutaron varias veces en el simulador para determinar su calidad media.

Los resultados hablan por sí mismos: la solución evolucionada superó tanto al propio plan de los expertos militares como al plan producido por sus modificaciones sobre la solución del AG. "...Los planes producidos por los algoritmos automáticos tenían un rendimiento medio significativamente mayor al de los generados por los experimentados expertos militares" (p. 161). Es más, los autores señalan que el plan del AG tenía sentido táctico. (Consistía en un ataque a dos flancos a la posición enemiga por pelotones de infantería mecanizada apoyados por helicópteros de ataque y exploradores terrestres, en conjunción con vehículos aéreos no tripulados realizando labores de reconocimiento para el fuego directo de artillería). Por añadidura, el plan evolucionado incluía unidades amigas individuales llevando a cabo misiones doctrinales -una propiedad emergente que apareció durante el curso de la ejecución, en lugar de ser especificada por el experimentador. En campos de batalla modernos, cada vez más conectados por red, el atractivo potencial de un algoritmo evolutivo que pueda automatizar la producción de planes tácticos de alta calidad debería ser obvio.

Naik 1996[lxxviii] informa de un uso interesante de los AGs en el cumplimiento de la ley, describiendo el software "FacePrints", un proyecto para ayudar a los testigos a identificar y describir a los sospechosos criminales. La imagen cliché del artista policía haciendo un dibujo del rostro del sospechoso en base a la descripción de los testigos es un método difícil e ineficiente: la mayoría de la gente no es buena describiendo aspectos individuales del rostro de una persona, como el tamaño de la nariz o la forma de la mandíbula, pero sin embargo son mejores al reconocer caras completas. FacePrints aprovecha esto utilizando un algoritmo genético que evoluciona dibujos de caras basándose en bases de datos de cientos de características individuales que pueden combinarse de infinitas maneras. El programa muestra a los testigos imágenes de rostros generadas aleatoriamente, y éstos escogen las que más se

[lxxviii] Naik, Gautam. ``Back to Darwin: In sunlight and cells, science seeks answers to high-tech puzzles." The Wall Street Journal, 16 de enero de 1996, p. A1

parecen a la persona que vieron; las caras seleccionadas mutan y se combinan para generar nuevas combinaciones de características, y el proceso se repite hasta que emerge un retrato preciso del rostro del sospechoso. En un caso real de atraco, los retratos definitivos que crearon los tres testigos eran sorprendentemente parecidos, y el dibujo resultante apareció en el periódico local.

3.8.12 Biología molecular

En los seres vivos, las proteínas transmembrana son proteínas que sobresalen de una membrana celular. Las proteínas transmembrana realizan a menudo funciones importantes como detectar la presencia de ciertas sustancias en el exterior de la célula o transportarlas hacia el interior de la célula. Para comprender el comportamiento de una proteína transmembrana es necesario identificar el segmento de la proteína que realmente está insertado en la membrana, lo que se conoce como dominio transmembrana. Durante las dos últimas décadas, los biólogos moleculares han publicado una serie de algoritmos cada vez más precisos para este propósito.
Todas las proteínas utilizadas por los seres vivos están formadas por los mismos 20 aminoácidos. Algunos de estos aminoácidos son hidrofóbicos, lo que significa que repelen el agua, y algunos son hidrofílicos, lo que significa que atraen el agua. Las secuencias de aminoácidos que son parte de un dominio transmembrana tienen probabilidad de ser hidrofóbicas. Sin embargo, la hidrofobicidad no es una característica definida con precisión, y no existe acuerdo sobre una escala para medirla.
Koza et al. 1999[lxxix], capítulo 16, utilizaron programación genética para diseñar un algoritmo que identificase el dominio transmembrana de una proteína. Se le suministró al programa genético un conjunto de operadores matemáticos estándares con los que trabajar, además de un conjunto de funciones booleanas para la detección de aminoácidos que devuelven +1 si el aminoácido de una posición dada es el aminoácido que detectan o -1 en caso contrario. (Por ejemplo, la función A? recibe como argumento un número que corresponde a una posición dentro de la proteína, y devuelve +1 si el aminoácido de esa posición es alanina, denotado por la letra A, y si no devuelve -1). Una variable de memoria compartida contenía una cuenta de la suma total, y

[lxxix] Koza, John, Forest Bennett, David Andre y Martin Keane. Genetic Programming III: Darwinian Invention and Problem Solving. Morgan Kaufmann Publishers, 1999.

cuando el algoritmo acababa, el segmento proteínico se identificaba como dominio transmembrana si su valor era positivo. Con tan sólo estas herramientas, ¿haría falta más información para que un diseñador humano produjese una solución eficiente a este problema?
Las aptitudes de las soluciones producidas por la programación genética fueron evaluadas probándolas con 246 segmentos proteínicos de los que se conocía su condición de transmembrana. Luego se evaluó al mejor individuo de la prueba con 250 casos adicionales inéditos (out-of-sample), y se comparó su efectividad con la de los cuatro mejores algoritmos humanos para el mismo propósito. El resultado: la programación genética produjo un algoritmo de identificación de segmentos transmembrana con una tasa total de error del 1,6%- significativamente menor que las de los cuatro algoritmos humanos, el mejor de los cuales tenía una tasa de error del 2,5%. El algoritmo diseñado genéticamente, al que los autores llamaron regla 0-2-4, funciona de la manera siguiente:

- Incrementar la suma en 4 unidades por cada instancia de ácido glutámico (un aminoácido cargado eléctricamente y muy hidrofílico) del segmento proteínico.
- Incrementar la suma en 0 unidades por cada instancia de alanina, fenilanalina, isoleucina, leucina, meionina o valina (todos aminoácidos muy hidrofóbicos) del segmento proteínico.
- Incrementar la suma en 2 unidades por cada instancia de cualquier otro aminoácido.
- Si [(SUMA - 3,1544)/0,9357] es menor que la longitud del segmento proteínico, clasificar ese segmento como dominio transmembrana; de lo contrario, clasificarlo como dominio no transmembrana.

3.8.13 Reconocimiento de patrones y explotación de datos

La competición en la industria actual de las telecomunicaciones es feroz, y se ha acuñado un nuevo término -"fuga", por el inglés "churn", término de difícil traducción, para describir la velocidad a la que los usuarios se cambian de un proveedor de servicios a otro. La fuga le cuesta a las compañías de telecomunicaciones una gran cantidad de dinero cada año, y reducir las fugas

es un factor importante para aumentar la rentabilidad. Si las compañías pueden contactar con los clientes que tienen probabilidad de cambiar y ofrecerles incentivos especiales para que se queden, puede reducirse la tasa de fugas; pero ninguna compañía tiene los recursos para contactar a más de un pequeño porcentaje de sus clientes. El problema es, por tanto, cómo identificar a los clientes que más piensen fugarse con mayor probabilidad. Todas las compañías tienen grandes bases de datos con información de los clientes que teóricamente puede utilizarse para este propósito; pero ¿qué método funciona mejor para examinar esta enorme cantidad de datos e identificar los sutiles patrones y tendencias que indican la probabilidad de fuga de un cliente?

Au, Chan y Yao 2003[lxxx] aplicaron algoritmos genéticos a este problema para generar un conjunto de reglas de tipo si-entonces para predecir la probabilidad de fuga de distintos grupos de clientes. En su AG, la primera generación de reglas, todas las cuales tenían una condición, fue generada utilizando una técnica de inducción probabilística. Las generaciones posteriores las refinaron, combinando sencillas reglas de una condición con reglas más complejas con varias condiciones. Para la medición de la aptitud se utilizó una medida de correlación objetiva de la "interesantitud", que no necesitaba información de entrada subjetiva. El algoritmo evolutivo de explotación de datos se probó sobre una base de datos real de 100.000 clientes proporcionada por una compañía malasia, y su rendimiento se comparó con el de dos métodos alternativos: una red neuronal multicapa y un algoritmo basado en árbol de decisiones ampliamente utilizado, el C4.5. Los autores afirman que su AE fue capaz de descubrir regularidades ocultas en la base de datos y "fue capaz de hacer predicciones precisas de fuga con distintas tasas de fuga" (p. 542), superando al C4.5 bajo todas las circunstancias, superando a la red neuronal en tasas mensuales de fuga bajas e igualándola en tasas de fuga mayores y, en ambos casos, alcanzando las conclusiones más rápidamente. Algunas ventajas más del enfoque evolutivo son que puede funcionar eficientemente incluso cuando faltan algunos campos de datos, y que puede expresar sus descubrimientos en conjuntos de reglas fácilmente comprensibles, al contrario que la red neuronal.

Entre algunas de las reglas más interesantes halladas por el AE se encuentran las siguientes: los clientes tienen más probabilidad de fugarse si se han suscrito personalmente al plan de servicios y no han sido admitidos en ningún plan de bonificación (una solución potencial sería admitir a

[lxxx] Au, Wai-Ho, Keith Chan, y Xin Yao. ``A novel evolutionary data mining algorithm with applications to churn prediction." IEEE Transactions on Evolutionary Computation, vol.7, no.6, p.532-545 (diciembre de 2003).

todos esos clientes en planes de bonificación); los clientes tienen más probabilidad de fugarse si viven en Kuala Lumpur, tienen entre 36 y 44 años y pagan sus facturas en efectivo (supuestamente porque es más fácil cambiarse de proveedor para los clientes que pagan al contado, a diferencia de los que cargan en cuenta automáticamente); y los clientes que viven en Penang y contrataron a través de un cierto vendedor tienen más probabilidades de fugarse (este vendedor puede estar proporcionando un mal servicio al cliente y debería ser investigado).

Rizki, Zmuda y Tamburino 2002[lxxxi] utilizaron algoritmos evolutivos para evolucionar un complejo sistema de reconocimiento de patrones con una amplia variedad de usos potenciales. Los autores señalan que el reconocimiento de patrones es una tarea cada vez más realizada por algoritmos de aprendizaje automático, en particular, algoritmos evolutivos. La mayoría de ellos comienzan con un acervo de características predefinidas, del que un AE puede seleccionar combinaciones apropiadas para la tarea en cuestión; en contraste, este método empezaba desde cero, primero evolucionando detectores individuales de característica en forma de árboles de expresiones, y luego evolucionando combinaciones cooperativas de esos detectores para producir un sistema completo de reconocimiento de patrones. El proceso evolutivo selecciona automáticamente el número de detectores de característica, la complejidad de los detectores y los aspectos específicos de los datos a los que responde cada detector.

Para probar su sistema, los autores le asignaron la tarea de clasificar aviones basándose en sus reflexiones radar. Un mismo tipo de avión puede devolver señales bastante distintas dependiendo del ángulo y elevación desde el que se le observa, y distintos tipos de avión pueden devolver señales muy parecidas, así que esto no es una tarea trivial. El sistema de reconocimiento de patrones evolucionado clasificó correctamente un 97,2% de los objetivos, un porcentaje neto mayor que el de las otras tres técnicas -una red neuronal perceptrón, un algoritmo clasificador KNN y un algoritmo de base radial- con las que fue comparado. (La precisión de la red de base radial era sólo un 0,5% menor que la del clasificador evolucionado, una diferencia que no es estadísticamente significativa, pero la red de base radial necesitó 256 detectores de característica mientras que el sistema reconocedor evolucionado sólo utilizó 17). Como afirman los autores, "los sistemas de reconocimiento que evolucionan utilizan menos características que los sistemas producidos utilizando técnicas convencionales, pero consiguen una precisión de reconocimiento comparable o superior" (p. 607). También se han aplicado varios aspectos de su sistema en

[lxxxi] Rizki, Mateen, Michael Zmuda y Louis Tamburino. ``Evolving pattern recognition systems." IEEE Transactions on Evolutionary Computation, vol.6, no.6, p.594-609 (diciembre de 2002).

problemas que incluyen el reconocimiento óptico de caracteres, la revisión industrial y el análisis médico de imágenes.

Hughes y Leyland 2000[lxxxii] también aplicaron AGs multiobjetivo a la tarea de clasificar objetivos basándose en sus reflexiones radar. Los datos de una sección transversal radar de alta resolución necesitan enormes cantidades de espacio de almacenamiento en disco, y producir un modelo realista de la fuente a partir de los datos es muy costoso computacionalmente. En contraste, el método basado en el AG de los autores demostró ser muy exitoso, produciendo un modelo tan bueno como el del método iterativo tradicional, pero reduciendo el gasto computacional y las necesidades de almacenamiento hasta el punto de que era factible generar buenos modelos en un ordenador de escritorio. En contraste, el método iterativo tradicional requiere diez veces más resolución y 560.000 veces más accesos a los datos de imagen para producir modelos de calidad similar. Los autores concluyen que sus resultados "demuestran claramente" (p. 160) la capacidad del AG de procesar datos de radar bidimensionales y tridimensionales de cualquier nivel de resolución con muchos menos cálculos que los métodos tradicionales, manteniendo una precisión aceptablemente alta.

3.8.14 Robótica

El torneo internacional RoboCup[lxxxiii] es un proyecto para promocionar el avance de la robótica, la inteligencia artificial y los campos relacionados, proporcionando un problema estándar con el que probar las nuevas tecnologías -concretamente, es un campeonato anual de fútbol entre equipos de robots autónomos. (El objetivo fijado es desarrollar un equipo de robots humanoides que puedan vencer al equipo humano de fútbol que sea campeón del mundo en 2050; actualmente, la mayoría de los equipos de robots participantes funcionan con ruedas). Los programas que controlan a los miembros del equipo robótico deben exhibir un comportamiento complejo, decidiendo cuándo bloquear, cuándo tirar, cómo moverse, cuándo pasar la pelota a un compañero, cómo coordinar la defensa y el ataque, etcétera. En la liga simulada de 1997, David Andre y Astro Teller inscribieron a un equipo llamado Darwin United cuyos programas de control habían sido desarrollados automáticamente desde cero mediante programación genética,

[lxxxii] Hughes, Evan y Maurice Leyland. ``Using multiple genetic algorithms to generate radar point-scatterer models." IEEE Transactions on Evolutionary Computation, vol.4, no.2, p.147-163 (julio de 2000).
[lxxxiii] http://www.robocup.org/

un desafío a la creencia convencional de que "este problema es simplemente demasiado difícil para una técnica como ésa" (Andre y Teller 1999[lxxxiv], p. 346).

Para resolver este difícil problema, Andre y Teller le proporcionaron al programa genético un conjunto de funciones de control primitivas como girar, moverse, tirar, etcétera. (Estas funciones estaban también sujetas al cambio y refinamiento durante el curso de la evolución). Su función de aptitud, escrita para que recompensara el buen juego en general en lugar de marcar goles expresamente, proporcionaba una lista de objetivos cada vez más importantes: acercarse a la pelota, golpear la pelota, conservar la pelota en el campo contrario, moverse en la dirección correcta, marcar goles y ganar el partido. Debe señalarse que no se suministró ningún código para enseñar específicamente al equipo cómo conseguir estos objetivos complejos. Luego los programas evolucionados se evaluaron utilizando un modelo de selección jerárquico: en primer lugar, los equipos candidatos se probaron en un campo vacío y, si no marcaban un gol en menos de 30 segundos, se rechazaban. Luego se evaluaron haciéndoles jugar contra un equipo estacionario de "postes pateadores" que golpeaban la pelota hacia el campo contrario. En tercer lugar, el equipo jugaba un partido contra el equipo ganador de la competición RoboCup de 1997. Finalmente, los equipos que marcaron al menos un gol contra este equipo jugaron unos contra otros para determinar cuál era el mejor.

De los 34 equipos de su división, Darwin United acabó en decimoséptima posición, situándose justo en el medio de la clasificación y superando a la mitad de los participantes escritos por humanos. Aunque una victoria en el torneo sin duda habría sido más impresionante, este resultado es competitivo y significante de pleno derecho, y lo parece aún más a la luz de la historia. Hace unos 25 años, los programas informáticos que jugaban al ajedrez estaban en su infancia; por primera vez, una computadora había sido inscrita recientemente en una competición regional, aunque no ganó (Sagan 1979[lxxxv], p. 286). Pero "una máquina que juega al ajedrez a un nivel medio de la capacidad humana es una máquina muy capaz" (ibid.), y podría decirse que lo mismo es cierto para el fútbol robotizado. Si las máquinas de ajedrez actuales compiten al nivel de los grandes maestros, ¿qué tipo de sistemas producirá la programación genética dentro de 20 o 30 años?

[lxxxiv] Andre, David y Astro Teller. ``Evolving team Darwin United.'' En RoboCup-98: Robot Soccer World Cup II, Minoru Asada and Hiroaki Kitano (eds). Lecture Notes in Computer Science, vol.1604, p.346-352. Springer-Verlag, 1999

[lxxxv] Sagan, Carl. Broca's Brain: Reflections on the Romance of Science. Ballantine, 1979.

3.8.15 Diseño de rutas y horarios

Burke y Newall 1999[lxxxvi] utilizaron algoritmos genéticos para diseñar los horarios de los exámenes universitarios. Se sabe que, en general, el problema del horario es NP-completo, lo que significa que no se conoce un método para hallar con garantías una solución óptima en un tiempo razonable. En un problema así, hay restricciones duras -no puede asignarse el mismo aula a dos exámenes a la vez- y restricciones suaves -si es posible, no deben asignarse varios exámenes en sucesión a un mismo estudiante, para minimizar la fatiga. Las restricciones duras deben satisfacerse, mientras que las restricciones suaves deben satisfacerse lo máximo posible. Los autores llaman "algoritmo memético" a su método híbrido para resolver este problema: un algoritmo evolutivo con selección por rango proporcional a la aptitud, combinado con un trepacolinas local para optimizar las soluciones halladas por el AE. El AE se utilizó en cuatro conjuntos de datos de universidades reales (la menor de las cuales tenía 25.000 alumnos), y sus resultados se compararon con los resultados producidos por un método heurístico de vuelta atrás, un algoritmo muy consolidado que se encuentra entre los mejores que se conocen para este problema y que se utiliza en varias universidades. Comparado con este método, el AE produjo un resultado con una reducción de la penalización bastante uniforme del 40%.

He y Mort 2000[lxxxvii] aplicaron algoritmos genéticos al problema de hallar rutas óptimas en las redes de telecomunicaciones (como las redes de telefonía e Internet), que se usan para transmitir datos desde los remitentes hasta los destinatarios. Esto es un problema NP-difícil, un tipo de problema para el que los AGs son "extremadamente aptos... y han encontrado una enorme variedad de aplicaciones exitosas en esos campos" (p. 42). Es además un problema multiobjetivo, en el que hay que equilibrar objetivos en conflicto como maximizar el caudal de datos, minimizar los retrasos en la transmisión y la pérdida de datos, encontrar caminos de bajo costo y distribuir la carga uniformemente entre los conmutadores de la red. Cualquier algoritmo real satisfactorio debe también ser capaz de redirigir el tráfico de las rutas principales que fallen o estén congestionadas.

En el AG híbrido de los autores se utilizó un algoritmo de tipo "primero el camino más corto", que minimiza el número de "saltos" que debe realizar un paquete de datos dado, para generar la

[lxxxvi] Burke, E.K. y J.P. Newall. ``A multistage evolutionary algorithm for the timetable problem." IEEE Transactions on Evolutionary Computation, vol.3, no.1, p.63-74 (abril de 1999).

[lxxxvii] He, L. y N. Mort. ``Hybrid genetic algorithms for telecommunications network back-up routeing." BT Technology Journal, vol.18, no.4, p. 42-50 (octubre de 2000).

semilla de la población inicial. Sin embargo, esta solución no tiene en cuenta la congestión o fallo de los enlaces, condiciones inevitables en redes reales, y es entonces cuando el AG toma el control, intercambiando secciones de rutas. Cuando se probó sobre un conjunto de datos derivado de una base de datos en red real de Oracle, se descubrió que el AG era capaz de redirigir enlaces rotos o congestionados, equilibrar la carga de tráfico y maximizar el caudal total de la red. Los autores afirman que estos resultados demuestran la "efectividad y escalabilidad" del AG y que "se pueden conseguir soluciones óptimas o casi óptimas" (p. 49).

Esta técnica ha encontrado aplicaciones reales para propósitos similares, como informan Begley y Beals 1995[lxxxviii]. La compañía de telecomunicaciones U.S. West (ahora fusionada con Qwest) se enfrentó a la tarea de desplegar una red de fibra óptica. Hasta hace poco, el problema de diseñar la red para minimizar la longitud total de cable desplegado era resuelto por un ingeniero experimentado; ahora la compañía utiliza un algoritmo genético para realizar la tarea automáticamente. Los resultados: "El tiempo de diseño para las redes nuevas ha caído de dos meses a dos días, y le supone un ahorro a U.S. West de 1 millón a 10 millones de dólares cada una" (p. 70).

Jensen 2003[lxxxix] y Chryssolouris y Subramaniam 2001[xc] aplicaron algoritmos genéticos a la tarea de generar programas para líneas de montaje (job shop scheduling). Éste es un problema de optimización NP-difícil con múltiples criterios: deben tomarse en cuenta factores como el coste, los retrasos y el rendimiento, y puede que se tenga que cambiar al vuelo el programa de la línea de montaje debido a averías en la maquinaria, ausencia de empleados, retrasos en la entrega de piezas, y otras complicaciones, lo que hace que la robustez del programa sea una consideración importante. Ambos artículos concluyen que los AGs son significativamente superiores a las reglas de despacho de prioridad utilizadas comúnmente, al producir programas eficientes que pueden tratar con más facilidad los retrasos y las averías. Estos resultados no son simplemente teóricos, sino que se han aplicado a situaciones reales:

[lxxxviii] Begley, Sharon y Gregory Beals. ``Software au naturel." Newsweek, 8 de mayo de 1995, p.70

[lxxxix] Jensen, Mikkel. ``Generating robust and flexible job shop schedules using genetic algorithms." IEEE Transactions on Evolutionary Computation, vol.7, no.3, p.275-288 (junio de 2003)

[xc] Chryssolouris, George y Velusamy Subramaniam. ``Dynamic scheduling of manufacturing job shops using genetic algorithms." Journal of Intelligent Manufacturing, vol.12, no.3, p.281-293 (junio de 2001).

Como informa Naik 1996[xci], los organizadores de los Juegos Paraolímpicos de 1992 utilizaron un AG para diseñar los horarios de los eventos. Como informa Petzinger 1995[xcii], John Deere & Co. ha utilizado AGs para generar los programas de montaje para una planta de Moline, Illinois, que fabrica plantadoras y otras maquinarias agrícolas pesadas. Al igual que los coches de lujo, éstas pueden construirse en una gran variedad de configuraciones con muchas partes y opciones distintas, y la enorme cantidad de maneras posibles de construirlas implica que el diseño eficiente de programas de montaje sea un problema aparentemente intratable. La productividad se veía mermada por cuellos de botella en el montaje, los equipos de trabajadores discutían, y se estaba perdiendo dinero. Finalmente, en 1993, Deer acudió a Bill Fulkerson, un analista e ingeniero de personal que concibió la utilización de un algoritmo genético para producir programas de montaje para la planta. Tras superar el escepticismo inicial, el AG demostró su valía rápidamente: la producción mensual aumentó un 50 por ciento, el tiempo extra casi desapareció y otras plantas de Deer están incorporando los AGs en sus propios diseños de programas de montaje.

Como informa Rao 1998[xciii], Volvo ha utilizado un programa evolutivo llamado OptiFlex para diseñar el programa de montaje de su fábrica de Dublín, Virginia, de un millón de metros cuadrados, una tarea que requiere controlar cientos de restricciones y millones de permutaciones posibles para cada vehículo. Como todos los algoritmos genéticos, OptiFlex funciona combinando aleatoriamente distintos programas de montaje posibles, determinando su aptitud clasificándolos en base a sus costos, beneficios y restricciones, y luego haciendo que las mejores soluciones intercambien genes entre ellas y vuelvan a la población para otra prueba. Hasta hace poco, esta desalentadora tarea era responsabilidad de un ingeniero humano, al que le llevaba hasta cuatro días producir el programa para cada semana; ahora, gracias a los AGs, esta tarea se puede completar en un día con una mínima intervención humana.

Como informa Lemley 2001[xciv], United Distillers and Vintners, una empresa escocesa que es el mayor y más rentable distribuidor de licores del mundo y es responsable de más de un tercio de la producción mundial de whisky de grano, utiliza un algoritmo genético para administrar su

[xci] Naik, Gautam. ``Back to Darwin: In sunlight and cells, science seeks answers to high-tech puzzles.'' The Wall Street Journal, 16 de enero de 1996, p. A1

[xcii] Petzinger, Thomas. ``At Deere they know a mad scientist may be a firm's biggest asset.'' The Wall Street Journal, 14 de julio de 1995, p.B1

[xciii] Rao, Srikumar. ``Evolution at warp speed.'' Forbes, vol.161, no.1, p.82-83 (12 de enero de 1998).

[xciv] Lemley, Brad. ``Machines that think.'' Discover, enero de 2001, p.75-79.

inventario y sus suministros. Esto es una tarea desalentadora que exige almacenar y distribuir eficientemente más de 7 millones de barriles, que contienen 60 recetas distintas, entre un enorme sistema de almacenes y destilerías, dependiendo de una multitud de factores como la edad, el número de malta, el tipo de madera y las condiciones del mercado. Anteriormente, coordinar este complejo flujo de suministro y demanda requería de cinco empleados a tiempo completo. Hoy, unas cuantas pulsaciones de teclado en un ordenador solicitan a un algoritmo genético que genere un programa cada semana, y la eficiencia de almacenamiento casi se ha duplicado.

Beasley, Sonander y Havelock 2001[xcv] utilizaron un AG para programar los aterrizajes del London Heathrow, el aeropuerto más transitado del Reino Unido. Esto es un problema multiobjetivo que implica, entre otras cosas, minimizar los retrasos y maximizar el número de vuelos mientras se mantiene la suficiente distancia de separación entre los aviones (los vórtices de aire que se forman en la estela de un avión pueden ser peligrosos para otro avión que vuele demasiado cerca). Comparado con los horarios reales de un periodo intensivo del aeropuerto, el AG fue capaz de reducir el tiempo de espera medio en un 2-5%, implicando dos o tres vuelos extra despegando y aterrizando por cada hora -una mejora significativa. Sin embargo, se han logrado mejoras mayores: como se informa en Wired 2002[xcvi], aeropuertos internacionales y líneas aéreas importantes como Heatrhow, Toronto, Sydney, Las Vegas, San Francisco, America West Airlines, AeroMexico y Delta Airlines están utilizando algoritmos genéticos para programar los despegues, aterrizajes, mantenimiento y otras tareas, mediante el software del Ascent Technology's SmartAirport Operations Center (ver http://www.ascent.com/faq.html). Cruzando y mutando las soluciones en forma de horarios que incorporan miles de variables, “Ascent vence con comodidad a los humanos, aumentando la productividad hasta en un 30 por ciento en todos los aeropuertos en los que se ha implementado”.

3.8.16 Ingeniería de sistemas

Benini y Toffolo 2002[xcvii] aplicaron un algoritmo genético a la tarea multiobjetivo de diseñar molinos eólicos para generar energía eléctrica. Este diseño “es un procedimiento complejo

[xcv] Beasley, J.E., J. Sonander y P. Havelock. ``Scheduling aircraft landings at London Heathrow using a population heuristic." Journal of the Operational Research Society, vol.52, no.5, p.483-493 (mayo de 2001).
[xcvi] ``Adaptive Learning: Fly the Brainy Skies." Wired, vol.10, no.3 (marzo de 2002) -
[xcvii] Benini, Ernesto y Andrea Toffolo. ``Optimal design of horizontal-axis wind turbines using blade-element theory and evolutionary computation." Journal of Solar Energy Engineering, vol.124, no.4, p.357-363 (noviembre de 2002).

caracterizado por varias decisiones sobre contrapartidas... El proceso de toma de decisiones es muy difícil y no hay tendencias de diseño bien establecidas" (p. 357); como resultado, hoy existen varios tipos de turbina distintos y no hay acuerdo sobre cuál es la óptima, si alguna lo es. Deben tomarse en cuenta objetivos mutuamente exclusivos como la producción máxima de energía anual y el coste mínimo de la energía. En este artículo se utilizó un algoritmo evolutivo multiobjetivo para encontrar el mejor conjunto de contrapartidas entre estos objetivos, construyendo palas de molino con una configuración óptima de características como la velocidad de la punta de la pala, la razón buje/punta, y la distribución de cuerda y giro. Al final, al AG consiguió encontrar soluciones competitivas con los diseños comerciales, además de dilucidar más claramente los márgenes entre los que se puede aumentar la producción anual de energía sin producir diseños demasiado caros.

Haas, Burnham y Mills 1997[xcviii] utilizaron un algoritmo genético multiobjetivo para optimizar la forma, orientación e intensidad del haz de los emisores de rayos X utilizados en la radioterapia dirigida, para destruir los tumores cancerosos al tiempo que se evita el tejido sano. (Los fotones de rayos X dirigidos hacia un tumor tienden a dispersarse por las estructuras interiores del cuerpo, dañando inintencionadamente los órganos internos. El reto consiste en minimizar este efecto mientras se maximiza la dosis de radiación dirigida hacia el tumor). Utilizando un modelo de aptitud basada en rango, los investigadores comenzaron con la solución producida por el método convencional, un método de mínimos cuadrados iterativo, y luego utilizaron el AG para modificarlo y mejorarlo. Construyendo un modelo del cuerpo humano y exponiéndolo al rayo evolucionado por el AG, encontraron un buen acuerdo entre las distribuciones de radiación predicha y real. Los autores concluyen que sus resultados "muestran una protección [de los órganos sanos] que no podía lograrse utilizando las técnicas convencionales" (p. 1745).

Lee y Zak 2002[xcix] utilizaron un algoritmo genético para evolucionar un conjunto de reglas para controlar un sistema de frenos antibloqueo automovilístico. Aunque la capacidad que tienen los sistemas de freno antibloqueo de reducir la distancia de frenada y mejorar la maniobrabilidad ha salvado muchas vidas, el rendimiento del ABS depende de las condiciones de la superficie de la carretera: por ejemplo, un controlador ABS que esté optimizado para el asfalto seco no

[xcviii] Haas, O.C.L., K.J. Burnham y J.A. Mills. ``On improving physical selectivity in the treatment of cancer: A systems modelling and optimisation approach." Control Engineering Practice, vol.5, no.12, p.1.739-1.745 (diciembre de 1997).

[xcix] Lee, Yonggon y Stanislaw H. Zak. ``Designing a genetic neural fuzzy antilock-brake-system controller." IEEE Transactions on Evolutionary Computation, vol.6, no.2, p.198-211 (abril de 2002)

funcionará igual de bien en carreteras mojadas o heladas, y viceversa. En este artículo, los autores proponen un AG para ajustar un controlador ABS que pueda identificar las propiedades de la superficie de la carretera (monitorizando el patinaje y aceleración de las ruedas) y pueda actuar en consecuencia, liberando la cantidad adecuada de fuerza de frenado para maximizar la tracción de las ruedas. En las pruebas, el ABS puesto a punto genéticamente "exhibe características de rodada excelentes" (p. 206) y fue "muy superior" (p. 209) a los otros dos métodos de maniobras de frenado, encontrando con rapidez nuevos valores óptimos para el patinaje de las ruedas cuando cambia el tipo de terreno bajo un coche en movimiento, y reduciendo la distancia total de frenada. "La lección que hemos aprendido de nuestro experimento... es que un AG puede ayudar a ajustar incluso un controlador bien diseñado. En nuestro caso, ya teníamos una buena solución del problema; sin embargo, con la ayuda de un AG, conseguimos mejorar significativamente la estrategia de control. En resumen, parece que merece la pena intentar aplicar un AG incluso en un controlador bien diseñado, porque hay muchas probabilidades de que se pueda hallar una configuración del controlador mejor utilizando AGs" (p. 211).

Como cita Schechter 2000[c], el Dr. Peter Senecal, de la Universidad de Wisconsin, utilizó algoritmos genéticos de población pequeña para mejorar la eficiencia de los motores diésel. Estos motores funcionan inyectando combustible en una cámara de combustión que está llena de aire extremadamente comprimido, y por tanto extremadamente caliente, lo bastante caliente para hacer que el combustible explote y empuje un pistón que produce la fuerza motriz del vehículo. Este diseño básico ha cambiado poco desde que Rudolf Diesel lo inventó en 1893; aunque se ha empleado mucho esfuerzo en realizar mejoras, es una tarea muy difícil de realizar analíticamente, porque requiere un conocimiento preciso del comportamiento turbulento que exhibe la mezcla de combustible y aire, y de la variación simultánea de muchos parámetros independientes. Sin embargo, el método de Senecal prescindía de ese conocimiento específico del problema y, en cambio, funcionaba evolucionando parámetros como la presión de la cámara de combustión, los tiempos de inyección de combustible y la cantidad de combustible de cada inyección. El resultado: la simulación produjo un motor mejorado que consumía un 15% menos de combustible que un motor diesel normal y producía dos tercios menos de óxido nítrico de

[c] Schechter, Bruce. ``Putting a Darwinian spin on the diesel engine." The New York Times, 19 de septiembre de 2000, p. F3.
Ver también: Patch, Kimberly. ``Algorithm evolves more efficient engine." Technology Research News, junio/julio de 2000. – Disponible en http://www.trnmag.com/Stories/062800/Genetically_Enhanced_Engine_062800.html

escape y la mitad de hollín. Luego el equipo de Senecal construyó un motor diésel real de acuerdo con las especificaciones de la solución evolucionada, y obtuvieron los mismos resultados. Ahora Senecal sigue su trabajo evolucionando la geometría del propio motor, lo que con suerte producirá todavía más mejoras.

Como citan Begley y Beals 1995[ci], Texas Instruments utilizó un algoritmo genético para optimizar la disposición de los componentes de un chip informático, colocando las estructuras de manera que se minimice el área total para crear un chip lo más pequeño posible. Utilizando una estrategia de conexiones que no se le había ocurrido a ningún humano, el AG alcanzó un diseño que ocupaba un 18% menos de espacio.

Finalmente, como cita Ashley 1992[cii], empresas de la industria aeroespacial, automovilística, fabril, turbomaquinaria y electrónica están utilizando un sistema de software propietario conocido como Engineous, que utiliza algoritmos genéticos, para diseñar y mejorar motores, turbinas y otros dispositivos industriales. En palabras de su creador, el Dr. Siu Shing Tong, Engineous es "un maestro `toqueteador', ensayando incansablemente las puntuaciones de escenarios de tipo "y-si" hasta que emerge la mejor solución posible" (p. 49). En un ensayo del sistema, Engineous consiguió producir un incremento del 0,92 por ciento de la eficiencia de una turbina experimental en sólo una semana, mientras que diez semanas de trabajo de un diseñador humano sólo produjeron un 0,5 por ciento de mejora.

Supuestamente, Engineous no sólo cuenta con algoritmos genéticos; también emplea técnicas de optimización numérica y sistemas expertos que utilizan reglas si-entonces para imitar el proceso de toma de decisiones de un ingeniero humano. Sin embargo, estas técnicas dependen mucho de información específica del dominio, carecen de aplicabilidad general, y son propensas a quedar atrapadas en óptimos locales. En contraste, el uso de algoritmos genéticos permite a Engineous explorar regiones del espacio de búsqueda que pasan por alto los otros métodos.

Engineous ha obtenido un amplio uso en una gran variedad de industrias y problemas. El más famoso fue cuando se utilizó para mejorar la turbina generadora de energía del avión Boeing 777; como informan Begley y Beals 1995[ciii], el diseño optimizado genéticamente era casi un 1% más eficiente en combustible que los motores anteriores, lo que en un campo como éste es una ganancia caída del cielo. Engineous también se ha utilizado para optimizar la configuración de

[ci] Begley, Sharon y Gregory Beals. ``Software au naturel." Newsweek, 8 de mayo de 1995, p.70.
[cii] Ashley, Steven. ``Engineous explores the design space." Mechanical Engineering, febrero de 1992, p.49-52.
[ciii] Begley, Sharon y Gregory Beals. ``Software au naturel." Newsweek, 8 de mayo de 1995, p.70.

motores eléctricos industriales, generadores hidroeléctricos y turbinas de vapor, para proyectar redes eléctricas, y para diseñar generadores superconductores y generadores de energía nuclear para satélites en órbita. Rao 1998[civ] informa también de que la NASA ha utilizado Engineous para optimizar el diseño de un avión de gran altitud para analizar la disminución del ozono, que debe ser a la vez ligero y eficiente.

Capitulo IV
Selección y desarrollo de un algoritmo genético

4.1 Aplicación de algoritmos genéticos

La aplicación demostrativa a desarrollar buscará encontrar la solución a un problema de transporte en el cual se busca desplazarse desde un punto dado A hasta otro punto dado B en un mapa asignado. Se tomarán en cuenta una serie de parámetros que servirán para simular condiciones de tráfico específicas en una ciudad. Con esto se pretende mostrar como la alteración de variables genéticas puede influir en el desempeño de estos algoritmos y su efectividad.

Básicamente como el propósito de la aplicación tratará un problema de tráfico vehicular, la solución será lograr una mejor distribución de automóviles en las calles de la ciudad bajo el supuesto de que un alto porcentaje de conductores utilizarían esta aplicación para evitar transitar en calles congestionadas.

La aplicación estará limitada a un mapa específicamente diseñado para ejemplificar la codificación y funcionamiento de los algoritmos genéticos. Estará formado por calles con un sentido y una cantidad de carriles especifica, no variable.

Los resultados proporcionados por la aplicación estarán sujetos a los cambios en las variables genéticas. Las condiciones de tráfico sobre las calles podrán ser modificadas de manera que la aplicación mostrará la solución que mejor se adapte al ambiente bajo el cual se ejecuta el algoritmo.

[civ] Rao, Srikumar. ``Evolution at warp speed." Forbes, vol.161, no.1, p.82-83 (12 de enero de 1998).

El algoritmo debe ser capaz de evaluar diferentes segmentos del mapa en el que operará, pero a su vez, permitir registrar la secuencia de segmentos que formarán la ruta a seguir. Como todos los algoritmos genéticos este deberá también adaptarse con el tiempo, las condiciones cambiantes de tráfico estarán representadas por variables definidas antes de ejecutarse, sin embargo en una aplicación real podrían monitorearse constantemente.

4.2 Primer caso examinado

Para desarrollar una aplicación de algoritmos genéticos es conveniente tomar como punto de partida las bases establecidas por John Holland[cv] y su metodología de trabajo. El primer paso para programar un sistema de búsqueda o aprendizaje implementando estos algoritmos es codificar el problema, su gran importancia se verá reflejada en el desarrollo de esta propuesta.

Tradicionalmente estos algoritmos utilizan una codificación binaria, esta fue la primera codificación utilizada por Holland y sus alumnos, por lo que los estudios subsecuentes mantuvieron la misma tendencia. La codificación es aplicada a las poblaciones que se emplearán. En realidad lo que se codifica es el genoma de cada miembro de la población.

Las poblaciones que se generan con los algoritmos genéticos son básicamente las posibles soluciones al problema. En este caso, fue necesario comenzar por codificar de forma binaria las rutas que se convertirán en las posibles soluciones óptimas. Para esto, y como base para la solución del problema, se simplificó el mapa de la ciudad y se redujo a una cuadricula, la cual podría ser ajustada posteriormente al diseño real de las calles.

La idea original fue partir desde el punto de inicio y comenzar a evaluar cada uno de los segmentos adyacentes en base a las propiedades de cada uno de ellos. Cada decisión tomada por el algoritmo quedaría registrada como parte de la ruta a seguir y daría paso a una siguiente

[cv] Genetic Algorithms
Computer programs that "evolve" in ways that resemble natural selection can solve complex problems even their creators do not fully understand
by John H. Holland

generación donde sería posible utilizar de una mejor forma los parámetros para la siguiente decisión.

Para la codificación de todas las trayectorias debe incluirse cada segmento de las calles involucradas, pero también, cada segmento debe tener asignados sus propios parámetros de aptitud y en este problema sería necesario mantener un mismo criterio al evaluar toda la cadena. Utilizando los fundamentos teóricos, se consideró en un primer momento agrupar las propiedades de los segmentos para formar una parte del genoma.

Instintivamente se podría asegurar que al modificar las propiedades de las rutas en la población se puede obtener una con la combinación de propiedades óptimas para transportarse. Si este problema busca considerar factores como la distancia, estructura, señalización, circulación de vehículos pesados y congestionamiento en función del tiempo, un genoma debería comprender una sección para cada una de estas propiedades.

El primer modelo de genoma a considerarse estaba estructurado como se muestra en la tabla 4.1. Los primeros 3 genes definirían a la conexión, el primero estaría reservado como una clave para extinguir a los individuos que representasen rutas imposibles, luego un gen que almacena número correlativo a la ruta seguido por otro que almacena la posición del segmento evaluado. El resto del genoma estaría asignado a los parámetros de interés.

X	#	Posición	Distancia	Estructura	Señalización	Tráfico pesado	Congestionamiento

Tabla 4.1 – Genoma inicial en el que los primeros 3 genes determinan el segmento

El problema con este modelo es que los últimos 5 elementos tendrían que modificarse conforme se evalúa una nueva generación y las mutaciones o cruces entre estos genes pocas veces coincidirían con los valores reales del mapa por lo que rápidamente se extinguiría la especie completa.

Como solución práctica se hizo el intento de agrupar los segmentos como un mismo tipo, estrategia que puede resultar muy útil al momento de codificar estos programas. Sin embargo al agrupar segmentos con propiedades similares se hacía mas complejo el algoritmo porque se agregaba la tarea de ubicar en el mapa cada segmento agrupado una vez la solución óptima fuera encontrada.

Para simplificar parcialmente este proceso se decidió agrupar los segmentos de acuerdo a su orientación, es decir basado en los puntos cardinales. Al agrupar únicamente N, S, E y O podrían encontrarse bifurcaciones que dejarían segmentos fuera de estos grupos, sin embargo al sacrificar un poco la capacidad de procesamiento se podía utilizar un genoma mas largo y agregar los grupos NE, NO, SE y SO.

En este caso el mayor reto de la codificación consistía en que a cada grupo se debe asignar una aptitud y esta no puede ser sometida al cruce o a la mutación. La necesidad de mantener ciertos genes fijos y excluidos de toda operación es un problema común al momento de hacer codificaciones para algoritmos genéticos, fue tratado originalmente por Schaffer y Morishima en 1987 siempre inspirados por comportamientos en la naturaleza.[cvi]

Su idea no era evolucionar el orden de bits en la secuencia, más bien las posiciones en las cuales el cruce fuera permitido; a esto le llamaron cruce de "puntos calientes". A cada solución posible en la población se le asigno una segunda cadena, un tipo formato para cruces permitidos, se atribuye un 1 en cada lugar geométrico en el cual esta permitido el cruce y un 0 en cada lugar geométrico en el cual no puede ocurrir. Por ejemplo, en la cadena 10011111:00010010, la primera sección es el cromosoma y la plantilla de cruce se encuentra después de los dos puntos. Aquí se indica que el cruce puede ocurrir únicamente en la cuarta y séptima posición geométrica de la secuencia.

[cvi] Schaffer, J. D. et Morishima, A. (1987). An adaptive crossover distribution mechanism for genetic algorithms. In Grefenstette, J. J., editor, GENETIC ALGORITHMS AND THEIR APPLICATIONS: Proceedings of the Second Internal Conference on Genetic Algorithms, Hilsdale, New Jersey, Hove and London. Lawrence Erlbaum Associates, Publishers.

Otra notación común se hace usando un signo de admiración para denotar los puntos posibles de cruce. Cada uno unido a la izquierda del respectivo bit. Esa misma cadena es posible escribirla como 1001!111!1.

Al intentar adaptar esta notación a la aplicación se hace posible y evita la transferencia de propiedades de aptitud un segmento a otro diferente. Sin embargo esto no resolvería el problema. Al estudiar más detenidamente esta alternativa, se encontró que no existía ninguna codificación posible que permitiera libremente el cruce y la mutación entre los individuos de la población.

Segmento	Orientación	Aptitud	Gen X	Sumatoria aptitud
0	0	0	1	1

Tabla 4.2 – En este intento de codificación se intentaba resumir las propiedades de aptitudes y a su vez aplicar el principio de los "puntos calientes" para evitar las operaciones de los genes que mantenían registro de segmentos anteriores

El problema es que aún cuando se mantenga una posición permanente para las aptitudes de cada segmento, tal como se planteaba en la tabla 4.2, al modificar las cantidades de segmentos era imposible garantizar que el algoritmo encontraría una solución posible. Podría encontrar una combinación óptima, más no necesariamente permitida dentro de las limitaciones del mapa.

Seg. Posición / Orientación	Seg. Aptitud	Posición / Orientación Seg. Anterior	Aptitud Seg. Anterior	Gen X

Tabla 4.3 – Este prototipo mostraba una agrupación de posición y orientación del segmento para poder generar la secuencia con la ruta óptima. Igual a que en la codificación anterior aquí se pretendía utilizar los "puntos calientes" para mantener la integridad de los valores de aptitud.

Se realizó un último intento de "encapsular" las propiedades de los segmentos. Unir diferentes genes de esta forma es otra técnica que en ocasiones hace posible la codificación de problemas complejos. Simplemente consiste en considerar como un gen un segmento completo del genoma, esto permite que la secuencia genetica no se pierda. Este procedimiento aumenta la cantidad de procesos a realizar y en este caso no proporciona una solución viable, de igual forma los

segmentos del genoma no podrían someterse al cruce sino únicamente a la mutación. Por todas estas razones se descarto por completo esta codificación.

4.3 Segundo caso examinado

Aún después de varios intentos para simplificarla, la codificación binaria en este caso resultaba en un incremento de procesos debido a la gran cantidad de bits que tendrían que ser utilizados para definir un segmento en el mapa, el problema principal esta al momento de especificar un valor máximo. Cuando se codifica un genoma, para poderlo operar es necesario especificar un número fijo de bits en la cadena los cuales están asignados a una característica particular. Por ejemplo, al momento de definir que parte de la cadena que estará atribuida a la aptitud de una zona, se debe delimitar el número de bits a utilizar y esto restringe el máximo. Si se asignan únicamente 4 bits, el valor entero máximo sería 15 partiendo desde 0 y esto en cierta forma limita la codificación.

Un tipo de codificación alterna, es la codificación de múltiples caracteres o valores reales. En muchas aplicaciones resulta más natural utilizar alfabetos con muchos caracteres o incluso los valores reales. En la teoría, los argumentos de los esquemas de Holland parecen implicar que un algoritmo codificado con múltiples caracteres tiene un desempeño menor a los que utilizan codificación binaria.

Las ventajas de la codificación binaria radican en que las mutaciones o alteraciones de un gen son más significativas y permiten explorar una progresión más amplia de posibilidades. Sin embargo existen aplicaciones que, por el contrario, muestran mejor desempeño al utilizar codificaciones de valores reales[cvii]. Pero el desempeño depende siempre del problema más que de los detalles del algoritmo en si. Actualmente no existe ninguna guía específica que pueda predecir cual codificación tendrá mayor eficacia.

[cvii] Michalewicz and C. Janikow, Handling Constraints in Genetic Algorithms – 1991.

Debido a la gran cantidad de valores posibles se decidió utilizar una codificación de valores reales al intentar un nuevo diseño. Esta vez, se considero operar los genomas únicamente mediante la mutación. Inicialmente se consideró el utilizar una ruta conocida que uniera los dos puntos deseados, y partiendo de esta ruta comenzar a realizar mutaciones y cruces de manera que la población alcanzara la mayor aptitud posible.

Esta solución, valiéndose de una codificación de valores reales, permitiría definir un genoma de tamaño fijo que podría ser operado con las técnicas mencionadas anteriormente. En la tabla 4.4 se muestra la estructura del genoma en la cual una ruta conocida se podría modificar hasta encontrar la ruta optima. El funcionamiento del algoritmo sería muy similar al mencionado en la sección 3.2 e incorporaría los conceptos de "puntos calientes" para el gen de sobrevivencia.

Gen X	Cant. Segmentos N	Cant. Segmentos S	Cant Segmentos E	Cant. Segmentos O	Aptitud

Tabla 4.4 – Diseño de un genoma codificado con valores reales y longitud fija

A pesar de ser una solución efectiva, no cumple con los requerimientos mencionados en el planteamiento del problema. El programa supone que no existen rutas conocidas, pero aún si estas fueran conocidas uno de los requerimientos indispensables es que el programa pueda adaptarse, es decir que si las propiedades de las calles en el mapa cambiaran, el algoritmo debe ser capaz de adaptarse a estas modificaciones. Si alguna de estas afectara las rutas conocidas, el desempeño del algoritmo no sería el mismo.

Con la intención de cumplir con todos los requerimientos del programa, se modificó este algoritmo para que, en vez de partir de una ruta conocida, iniciara desde una que tenga una aptitud óptima imposible. En este caso se partiría de un trayecto en línea recta desde el punto de inicio hasta el de llegada, luego este iría mutando y reduciendo gradualmente su aptitud.

Con esta modificación el algoritmo continuaría utilizando una codificación de valores reales y eliminando la necesidad de mantener un registro histórico de cuales segmentos han sido seleccionados ya que la solución optima sería una única ruta que se modificaría con el tiempo.

Gen X	Cant. Segmentos N	Cant Segmentos S	Cant. Segmentos E	Cant. Segmentos O	Cant. Segmentos NE	Cant. Segmentos NO	Cant. Segmentos SE	Cant. Segmentos SO	Aptitud

Tabla 4.5 – Estructura de un genoma capaz de almacenar mas información manteniendo los mismos genes terminales

Una limitación grave que muestra este algoritmo al intentar implementarlo es la ubicación de los segmentos seleccionados. Como se puede ver en el diseño del cromosoma de la tabla 4.5 no se registra la posición de los segmentos codificada en ninguno de sus genes. Esto implica mantener un proceso paralelo a la mutación que permita ubicar las calles del mapa seleccionadas, para que una vez se encuentre una combinación óptima se pueda generar la ruta deseada.

Con intenciones de simplificar esta tarea de ubicación y reconstrucción de la ruta final, se consideró segmentar una vez mas el mapa, en este caso se dividiría el mapa por cuadrantes. Cada uno tendría asignados un número especifico de segmentos, de forma idónea se esperaba agrupar en sectores con las mismas dimensiones.

Al fraccionar el mapa de esta forma, se multiplica la cantidad de secciones necesarias para la codificación. Si utilizamos como ejemplo la figura 4.1, el diagrama de la izquierda, donde el mapa se encuentra seccionado en cuatro cuadrantes, los elementos necesarios para codificar la ruta se multiplicarían, es decir, tendríamos cuatro grupos de fragmentos y en cada grupo una subdivisión para cada una de las ocho orientaciones posibles, se necesitaría un genoma con 34 genes de longitud.

Este cambio implica un incremento sustancial en la longitud de las cadenas que se operan, sin embargo si tomamos en cuenta la considerable reducción de caracteres después de descartar la codificación binaria, el algoritmo continúa siendo perfectamente funcional.

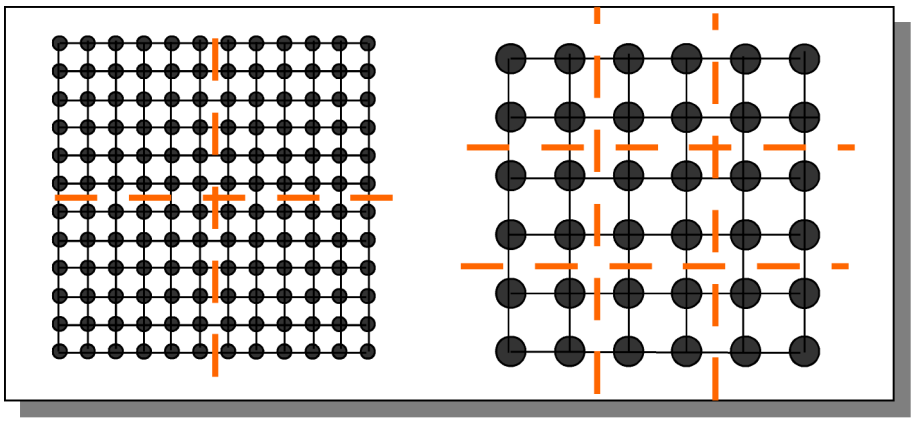


Figura 4.1 – Diagrama del mapa de la ciudad representado por segmentos y agrupado por bloques. Cuando la cantidad de bloques es mayor, el número de segmentos o líneas por bloque disminuye y se facilita su ubicación en el mapa.

Una limitación que se hace evidente al tratar de utilizar segmentaciones es que entre mas bloques contiene el cuadrante, menor es su utilidad al momento de ubicar los elementos finales. Si se intentara agrupar de manera similar a la izquierda de la figura 4.1, simplemente se reducen el proceso paralelo de ubicación a cuatro procesos similares pero un poco más sencillos.

Con esto en mente se hace necesaria una división como la derecha de la figura 4.1, en la que no se utilizan más de cuatro puntos completamente dentro del grupo. Esta disposición también requería de modificaciones adicionales al algoritmo, requería definir de forma previa donde pertenecían los segmentos que conectaban diferentes cuadrantes. Estas reglas a su vez podrían generar ambigüedades al momento de reconstruir la solución final por lo que se optó por no evaluar segmentos sino nodos.

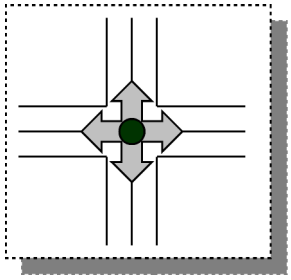


Figura 4.2 – Representación de un nodo como punto de referencia para poder recrear una ruta seleccionada

Esta nueva forma de evaluar resulto ser altamente eficiente y no solo eliminaba cualquier posibilidad de confusión sino también permitía una codificación más eficiente. Es por eso que a partir de este momento se utilizaría siempre esta misma convención.

Entre los problemas que presentan este tipo de codificación es que una vez se lleva a la práctica, entre mas grande sea el mapa a evaluar, un mayor número de segmentos son necesarios, pero sobretodo la mayor limitante es que el algoritmo no puede ser utilizado para un mapa mas grande y si la cantidad de nodos se incrementara, el algoritmo no sería capaz de adaptarse a la nueva segmentación y tendría que ser rediseñado.

Para resolver este problema, se implementaría un concepto un poco diferente al manejado tradicionalmente en la teoría, pero que al igual que todo ha sido inspirado por evolución natural. Se simularía un genoma variable, es decir, que la cantidad de genes varía conforme la cadena evoluciona. En este caso, esta modificación permite almacenar dentro de la información genética del individuo todos los nodos que formarán parte de la ruta. Esto le da al algoritmo mayor flexibilidad para mutación.

Con una ruta que gradualmente reduce su aptitud pero que a su vez registra todos los nodos involucrados, la solución de este problema se hace posible. El algoritmo que utilizaría es muy similar a los mencionados en la sección 3.4 en el que se describe como la “solución actual” se modifica con el tiempo hasta encontrar la ruta con las modificaciones óptimas. La diferencia en este caso es que no se parte de una solución posible al azar, sino de una solución ideal imposible y se modifica hasta encontrar una posible.

El algoritmo aplicado sería el siguiente:

1. Asigna A y B dentro del mapa.
2. Calcula la distancia entre ellos.
3. Agrega un gen intermedio con su ubicación en base a la distancia.
4. Asigna una edad a cada uno de los genes.
5. Realiza un proceso de mutación en base a la edad.
6. Elimina las modificaciones imposibles

7. Incorpora dos nuevos genes adyacentes a los de menor edad
8. Calcula las distancias de los nuevos elementos agregados.
9. Elimina genes innecesarios en base a las distancias y ubica nuevos.
10. Almacena y compara la longitud del genoma. Si es mayor a la que esta actualmente registrada entonces regresa al paso 4.
11. Regresa la solución óptima.

Una aplicación que utilice este algoritmo permitiría encontrar una muy buena forma para desplazarse entre dos puntos del mapa. La ventaja principal que ofrece este proceso es que el resultado que encuentra primero sería el mejor de todos los posibles, ya que si se continúa mutando aún después de encontrar una primera ruta, únicamente se encontrarían otras con mayor número de nodos por lo que eventualmente tendrían que ser eliminadas.

A pesar de que este algoritmo es capaz de encontrar una solución al problema planteado, tiene limitaciones que hasta cierto punto quebrantan los principios de los algoritmos genéticos mencionados anteriormente. En la sección 2.6 se menciona como tanto en la biología como en los programas para computadoras, lo que evoluciona es toda una población y no solo un individuo.

La limitante en este caso consiste en que se esta siempre operando una única ruta, la cual se modifica hasta encontrar las propiedades buscadas, esto deja por fuera otras rutas que podrían iniciar con diferentes segmentos y que no necesariamente estarían enlazadas con los nodos encontrados.

Una de las propiedades de los algoritmos genéticos es que permiten buscar diferentes alternativas simultáneamente, sin embargo este programa no explotaría esa característica. Con la intención de aprovechar al máximo el potencial de la programación genética se realizaron diversas modificaciones.

Inicialmente se consideró modificar la forma en que inicialmente se seleccionaban los nodos. Al utilizar una codificación de valores reales y genoma variable, se hace posible seleccionarlos en

un orden específico, por lo que la primera modificación planteada era seleccionar puntos adyacentes de acuerdo a su aptitud con la intención de poder establecer dos diferentes poblaciones que fueran evolucionando, una desde el punto A hacia el punto B y la otra desde B hacia A.

El uso de diferentes poblaciones en algoritmos genéticos no es una práctica muy común debido a que por lo general para generar mayor diversidad y competencia de aptitud basta con utilizar una población con un gran número de individuos, pero este caso en particular se presta para experimentar.

La suposición era que cuando finalmente se completaran las dos poblaciones diferentes de rutas sería posible incorporarlas para formar una gran población que podría estar sujeta a mutaciones reguladas y cruces entre sus miembros. De esta superpoblación se podría elegir al miembro mas apto que sería la solución al problema.

Al momento de unir las dos poblaciones se imposibilita el funcionamiento del programa debido a que los genomas, por el mismo hecho de tener una longitud variable, contienen genes que obedecen a una posición específica y secuencial que no puede ser alterada. En este caso los genes de los individuos de una población no pueden ser cambiados por los genes en la misma posición de un individuo perteneciente a la otra población.

En este caso, primero era necesario lograr mantener el orden secuencial de los genes que forman los diferentes nodos que conforman la ruta. Por lo general cuando se encuentran este tipo de limitantes en el desarrollo de los algoritmos, en lugar de intentar solucionarlos mediante funciones o procesos adicionales, es conveniente retroceder y examinar las bases en búsqueda de soluciones.

En particular para este problema reestructurar la codificación ofrecía una buena oportunidad. Una vez mas, al examinar las bases establecidas por Holland es posible contar con una propuesta de codificación muy eficiente llamada “inversión” que fue retomada posteriormente

por Goldberg.[cviii] Esta codificación consiste en realizar un re ordenamiento inspirado por un operador similar en la genética natural. A diferencia de los algoritmos genéticos simples, en la genética verdadera la función de un gen es a menudo independiente de su posición en el cromosoma, aun cuando genes en una misma zona generalmente funcionan en conjunto como una red reguladora, así que utilizar inversión en parte del cromosoma conservará mucha o toda la "semántica" del cromosoma original.

Para implementar esta técnica, se establece una notación que permita la interpretación funcional de un alelo sin importar la posición que tenga en la cadena. La propuesta de Holland consistía en que a cada alelo se le asigne un índice que indicaba la posición real. Supongamos que se necesita codificar un genoma con la siguiente cadena de bits 10001101, entonces utilizando la técnica de inversión se codificaría como se muestra en la siguiente tabla.

(1,1)	(0,2)	(0,3)	(0,4)	(1,5)	(1,6)	(0,7)	(1,8)

Tabla 4.6 – Codificación típica de un genoma binario utilizando inversión

Tal como se puede ver en el ejemplo anterior, un segundo digito almacena la posición de la información. Esto le permite cruzarse con otros individuos y cambiar el valor de los datos pero manteniendo siempre su respectivo orden. Por los mismos propósitos de este problema se mantendría la codificación de valores reales que se adaptaría a la técnica de inversión.

(1,3)	(2,8)	(6,3)	(5,9)	(4,6)	(3,2)	(7,0)	(8,7)
(5,1)	(2,8)	(3,9)	(4,6)	(1,9)	(8,2)	(6,7)	(7,4)

Tabla 4.7 – Codificación de valores reales utilizando inversión. Esta notación permite operar alterar la posición de los valores manteniendo su posición geométrica.

Una vez más al ser llevado a la práctica se hacen evidentes ciertos problemas para funcionamiento del algoritmo al permitir un cambio de orden en los genes. En esta ocasión el problema es conocido e incluso fue estudiado desde que la codificación con inversión fue planteada. Si utilizamos como padres los individuos en la tabla 4.7 y los cruzamos a partir del

[cviii] Goldberg, D.E. (1989a) *Genetic Algorithms in Search, Optimization & Machine Learning*. Reading, MA: Addison-Wesley.

tercer segmento, entonces los individuos de la nueva generación estarían compuestos de la siguiente forma.

(1,3)	(2,8)	(6,3)	(4,6)	(1,9)	(8,2)	(6,7)	(7,4)
(5,1)	(2,8)	(3,9)	(5,9)	(4,6)	(3,2)	(7,0)	(8,7)

Tabla 4.8 – Individuos de una segunda generación después de realizar un cruce simple a partir de la tercera posición

Como se puede ver, el primer "hijo" tiene duplicados los genes 1 y 6, a su vez carece de genes 3 y 5. Por su lado el segundo "hijo" se encuentra en las mismas condiciones con elementos diferentes. Para este problema, Holland propuso dos posibles soluciones. La primera propuesta consiste en limitar el cruce de la generación padre a las mismas posiciones geométricas, pero esto consiste en una gran restricción para el algoritmo. La alternativa a esta opción es un sistema "maestro/esclavo" en el que se elije uno de los "padres" como primario y luego se reordena temporalmente el otro "padre" de acuerdo al maestro. Se mantiene este orden mientras se reproduce la segunda generación, luego el esclavo regresa a su orden original una vez el cruce ha terminado.

A pesar de los diferentes intentos de implementar este diseño, las constantes modificaciones para permitir que se articulara al problema, continuamente incrementaron la complejidad del algoritmo y obstaculizaron una solución eficiente. Es por eso que aún cuando un programa que realizara este procedimiento cumple con los requerimientos del problema, se decidió buscar otras alternativas.

4.4 Tercer caso examinado

Antes de iniciar un nuevo intento por resolver un problema de esta naturaleza, vale la pena tomar en cuenta la experiencia adquirida con intentos anteriores. En esta ocasión para desarrollar el algoritmo convenía partir de todos los elementos que resultaron de gran utilidad en los casos anteriores.

Inicialmente quedó demostrado que utilizar codificación binaria resulta ineficiente para este problema, por lo que es necesario utilizar valores reales independientemente de como se codifiquen los genomas. Por otro lado, también quedo clara la necesidad de que los códigos genéticos tengan la capacidad de variar conforme el algoritmo evoluciona, los genomas de longitud fija resultan inoperantes en este caso. Finalmente, se demostró que para poder ubicar las rutas en el mapa resulta mucho más práctico utilizar nodos en lugar de calles como puntos de referencia.

Con esto en mente, nuevamente se examinó el principio mismo del algoritmo en busca de soluciones, es decir, optimizar la codificación. Una alternativa capaz de cumplir con todos los requerimientos de este problema es mediante el uso de árboles binarios. Por supuesto, esta codificación debe ser adaptada al algoritmo deseado.

Uno de los autores conocidos por experimentar con este tipo de codificación fue John Koza quien en 1992 publicó un libro "Programación genética".[cix] El planteaba entre las ventajas de este tipo de programación que es posible hacer búsquedas en espacios abiertos ya que, en principio, todo árbol binario puede ser creado por medio del cruce y la mutación.

Lo que actualmente se conoce como programación genética muchas veces llega a considerarse como parte de de una familia de algoritmos que, al igual que las estrategias de evolución, programación evolutiva y los algoritmos genéticos, han sido desarrollados con el propósito de implementar los conceptos de evolución natural en herramientas de optimización y búsqueda.

La diferencia principal entre la programación genética y los algoritmos genéticos es que la última pretende evolucionar poblaciones de programas en lugar de evolucionar poblaciones de soluciones. En su libro Koza sostiene que aún cuando los problemas parezcan ser muy diferentes,

[cix] Genetic Programming
On the Programming of Computers by Means of Natural Selection
John R. Koza
A Bradford Book
The MIT Press
Cambridge, Massachusetts
London, England

pueden resumirse en la búsqueda de un programa que valiéndose de entradas definidas proporcione las salidas necesarias.

Así fue como surgió la idea de incorporar árboles binarios a este tipo de problemas. Al introducir la programación genética, Koza se valió del lenguaje LISP para expresar los programas en formato de árboles. Este es uno de los lenguajes de programación más antiguos, de hecho es el mas antiguo dentro lenguajes de alto nivel después del FORTRAN. Este lenguaje fue originalmente creado como una notación matemática práctica para programas de computadora, pero gracias a su estructura, continua siendo de gran utilidad en los experimentos de inteligencia artificial.

Cuando se considera detenidamente el uso árboles binarios en los algoritmos genéticos, se puede percibir el riesgo que el programa quede atrapado en ciclos infinitos y los árboles continúen extendiéndose permanentemente sin encontrar la respuesta buscada. Es por eso que en los casos en los que se aplica esta técnica se implementan reglas para controlar el crecimiento y la tasa de mutación.

En este problema particular, los árboles binarios estarían limitados a la cantidad de puntos disponibles en el mapa. La estrategia a utilizar para limitar este tipo de algoritmos debe ser, al igual que el planteamiento mismo, partir de puntos extremos y progresivamente unirlos. Esto restringe la cantidad de nodos a utilizar dentro del mapa, número que podría descontrolarse si se buscara en una sola dirección.

Un programa en el lenguaje LISP común utiliza expresiones que denotan cuales son las estructuras de código y datos. Si se deseara multiplicar dos variables A y B su notación seria básicamente (* A B). Como se puede ver el operador precede a los argumentos y esta misma estructura permite la representación por medio de árboles binarios. Un ejemplo simple puede ser el cálculo de la hipotenusa de un triángulo rectángulo, (SQRT (+ (^ A 2) (^ B 2))). Esta función puede ser representada por el árbol binario de la siguiente figura.

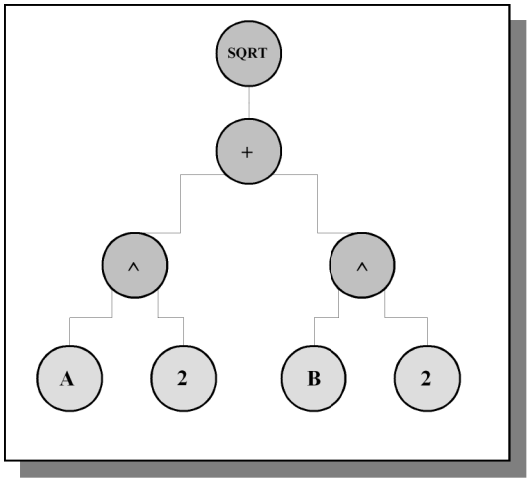


Figura 4.3 – Representación de un árbol binario partiendo del lenguaje LISP

Como el algoritmo que se desea desarrollar no estará enfocado en los cálculos matemáticos que puedan evaluar la conveniencia de la ruta, los operadores a utilizar en esta codificación deben ser muy diferentes a los comúnmente utilizados. En este caso los datos a utilizar deben ser los nodos que definirán la ruta dentro del mapa pero, al igual que en algunos programas de este tipo, en lugar de utilizar operadores lógicos o aritméticos se debe utilizar una función previamente definida, aquí sería la función que evalúa la aptitud.

Si representamos la función de aptitud mencionada como APT, el programa que buscamos evaluaría la aptitud de una conexión entre dos nodos diferentes, si estos los nombramos A y B, la operación (APT A B) nos regresaría la aptitud buscada y su representación por medio de árboles binarios seria como se muestra en la figura 4.4.

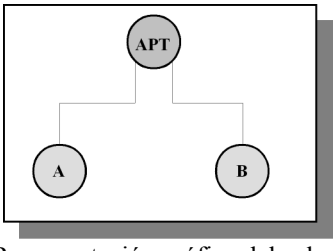


Figura 4.4 – Representación gráfica del calculo de aptitud

Con esta representación definida se hace posible el uso de un genoma de longitud variable que conforme evolucione su representación de árbol binario almacena los nodos que conformarán la

ruta óptima. Sin embargo, para resolver este problema este tipo de representación debe presentarse con aún más modificaciones.

El problema radica en que este árbol en particular utilizaría un único operador, es decir la función de aptitud. En los casos tradicionales de la programación genética es necesario almacenar la información de los operadores porque, tal como se mencionó anteriormente, se utilizan para evolucionar programas pero en este caso resulta innecesario. Tal como se puede ver en la siguiente figura, de utilizarse esta notación los genomas estarían formados principalmente por una misma función, los datos estarían almacenados únicamente al final de cada rama y reducirían la eficiencia del algoritmo.

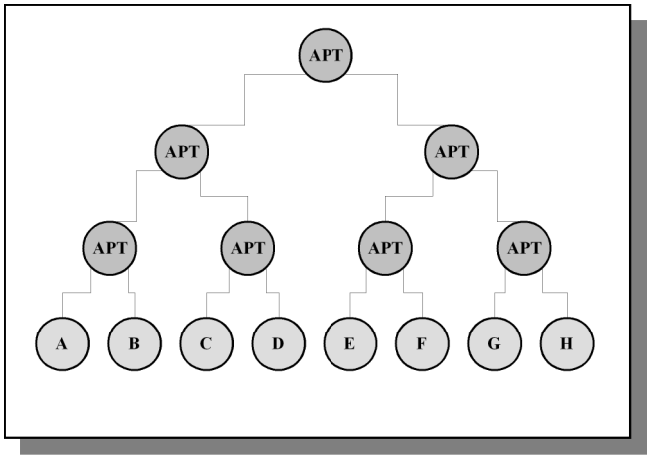


Figura 4.5 – Árbol binario que representa el calculo de aptitud para unir los nodos A y H.

Esta notación no solo es poco eficiente sino también puede ocasionar dificultades para programarla. Si tomamos como ejemplo la figura 4.5 podemos notar que fácilmente se podrá calcular la aptitud entre A y B, pero se dificulta calcular la aptitud entre C y B. Es por eso que una codificación más explicita se hace necesaria.

Como solución inmediata puede considerarse un genoma en el cual los cromosomas estén formados por 2 nodos. Esta vez, cada elemento del árbol binario estaría compuesto por el origen y el destino, con esta notación todas las ramas bajo cada elemento representarían los puntos necesarios para unir los nodos superiores. Una vez más se logra formar un genoma variable

capaz de almacenar todas las unidades involucradas en la ruta y así como se muestra en la figura 4.7, dispone de características convenientes para realizar cruces y mutaciones.

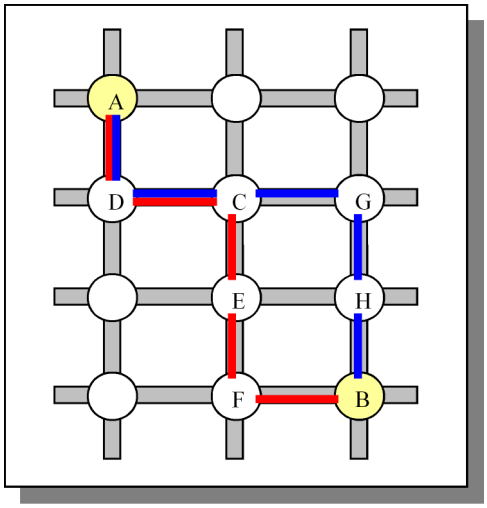


Figura 4.6 – Dos alternativas diferentes para desplazarse desde el nodo A hasta el B.

Si tomamos como ejemplo la figura anterior, podemos suponer que ambas rutas son dos individuos de la población final que han logrado desarrollarse gracias a su aptitud. Las rutas mostradas pueden estar representadas por los árboles mostrados a continuación. De igual forma se puede notar como si los individuos poseen genes en común, es decir que dentro de la ruta hay un nodo inicial y uno final comunes como en el ejemplo lo son C y B, entonces es posible realizar cruces que en rutas de mayor longitud podrían generar combinaciones con cambios mas significativos y por lo tanto mejores rutas alternas. En la figura 4.7 el árbol de la derecha representa la ruta señalada con color rojo y el que se muestra a lado izquierdo la ruta que esta en color azul.

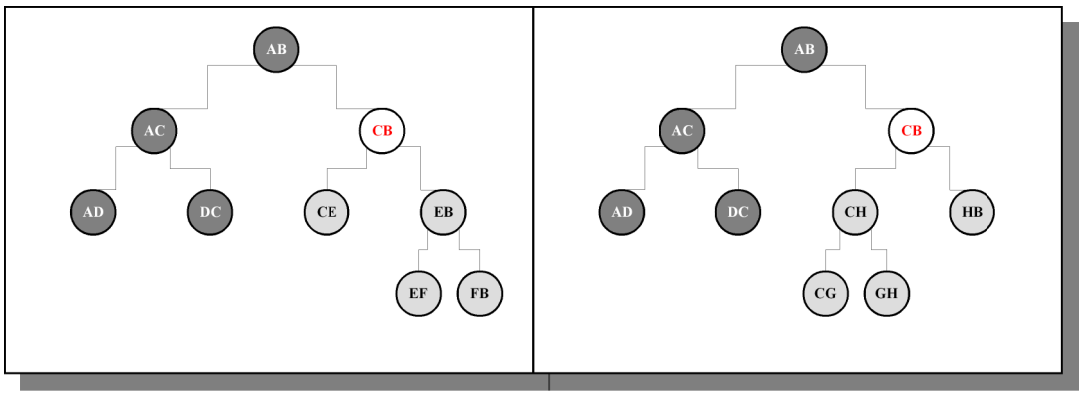


Figura 4.7 – En ambos árboles binarios existen ramas que pueden ser intercambiadas, en este caso el cruce no produce nuevas rutas, pero si se da un cruce en diferentes niveles jerárquicos las posibilidades de encontrar rutas mas eficientes son altas

El funcionamiento del algoritmo puede ser resumido con los pasos siguientes:

1. Asignar AB.
2. Calcular la distancia entre ellos.
3. Incluir un nuevo gen en desarrollo W.
4. Generar una población con candidatos posibles para W.
5. Selección de los genes más aptos.
6. Exponer los genomas a una mutación proporcional a la edad. Por lo que la mutación se realiza al final de las ramas.
7. Agregar una bifurcación para posibles nuevos nodos.
8. Calcula distancia entre ellos.
9. Compara la longitud del genoma. Si esta es diferente regresar al paso 3.
10. Validar los individuos más aptos.
11. Selección de acuerdo a la aptitud.

No hace falta describir más detalladamente el funcionamiento del algoritmo ya que tanto el programa como la codificación misma pueden ser mejorados. Se logró simplificar esta codificación para reducir el número de caracteres repetidos en una cadena de datos, facilitando de esta manera la operación entre individuos.

La codificación final parte de un operador inicial que es una función de inicialización, este es el único operador que aparecerá en el árbol, el resto de elementos serán los nodos involucrados en la ruta final. La estrategia para optimizar el algoritmo será en este caso almacenar en el genoma la menor cantidad de datos posibles. El funcionamiento estará basado en reglas que permitirán interpretar la codificación y se asistirá de una base de datos en la cual estarán almacenados los nodos del mapa junto con los parámetros que determinan la aptitud del mismo. Así se reducen los procesos para determinar la aptitud a una única evaluación lógica con valores reales.

La representación del árbol a utilizar será la siguiente, después de la función de inicialización cada una de las ramas contiene elementos que representan los puntos en el mapa. La notación parte de izquierda a derecha, con los elementos intermedios como los nodos necesarios para trasladarse. Como regla general a utilizar, un elemento puede tener únicamente dos elementos inferiores, el de la derecha representa la unidad intermedia entre el punto en cuestión y el elemento superior más a la derecha. De igual forma el elemento inferior de la izquierda representa el punto intermedio entre el nodo en cuestión y el elemento superior más a la izquierda.

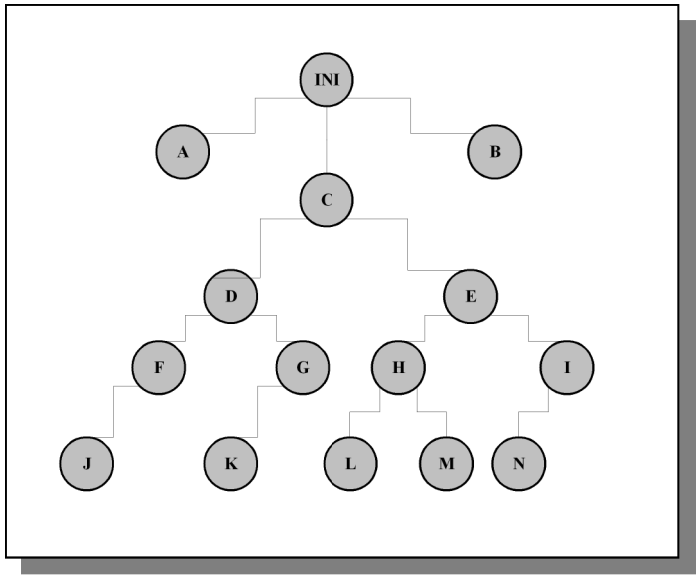


Figura 4.8 – Representación de una ruta que pasa por los nodos A, J, F, D, K, G, C, L, H, M, E, N, I, B respectivamente.

Con la figura 4.8 la explicación se hace un poco más sencilla. Por ejemplo, para unir A y B es necesario pasar por C. Pero para unir el A con C es necesario pasar por todos los nodos bajo la rama izquierda de C. Gracias a esta notación, es posible seguir una secuencia en la que se encuentran puntos intermedios continuamente y se evalúan aptitudes y distancias para encontrar las combinaciones de las unidades mas aptas.

Ahora, con una codificación funcional finalmente definida, es posible describir todo el proceso mediante el cual se obtendrá la ruta más óptima. El programa, al igual que los algoritmos genéticos tradicionales, repetirá continuamente un ciclo en el cual las mejores alternativas tienen prioridad sobre las rutas menos eficientes. Este ciclo mencionado se encuentra representado en la figura 4.9, mas adelante se detallará el funcionamiento de cada una de las etapas.

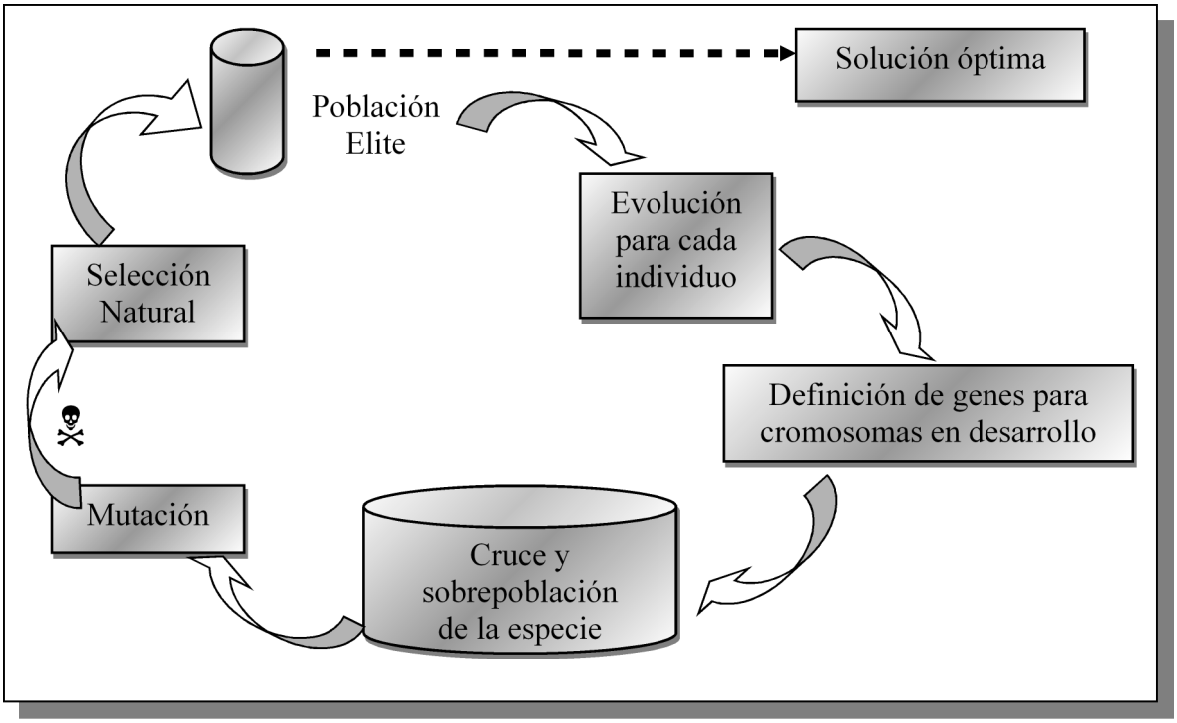


Figura 4.9 – Ciclo de procesos bajo el cual funcionará el programa

El programa inicia con información que le ha sido introducida, es decir parte de los puntos de salida y llegada. Luego cada uno de estos debe evolucionar para almacenar más información. Es

importante que esta evolución individual no sea confundida con la evolución poblacional producida por la selección natural.

La operación en esta etapa del ciclo puede verse reflejada en el siguiente flujograma, en el que se puede observar como se evalúa constantemente la distancia entre los nodos para poder determinar cuando una ruta alcanza la cantidad máxima de nodos.

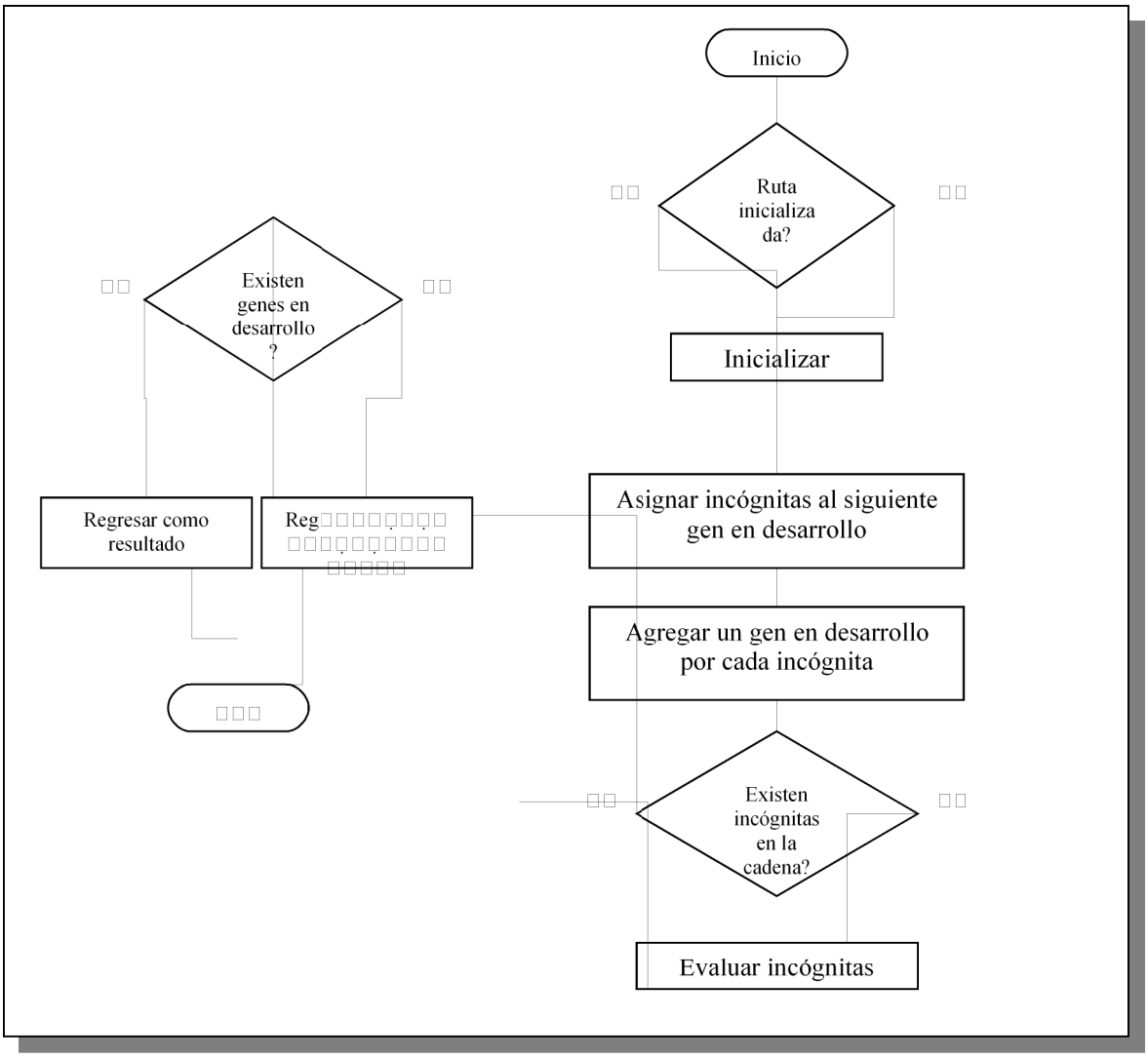


Figura 4.10 – Flujograma para la evolución individual de cada genoma

En el flujograma anterior se menciona inicialmente el proceso de inicializar la ruta. Esto consiste en definir los primeros dos genes del genoma. Como se mencionó anteriormente en la parte superior del árbol binario se manejan los nodos extremos de manera diferente, por lo que es

necesario ubicar el punto de salida y el de llegada en estos extremos, asignar una única incógnita y un solo gen en desarrollo. Una vez han sido definidos se encuentra un nodo intermedio para reemplazar la incógnita preparada, después de eso el algoritmo funciona igual independientemente de la cantidad de nodos involucrados en la ruta.

El gen en desarrollo que tanto ha sido mencionado hasta el momento consiste en una incógnita como elemento central y dos elementos secundarios que pueden llegar a convertirse en otras incógnitas en futuras generaciones ó pueden llegar a almacenar elementos terminales. Su relevancia y modo de empleo se harán mas evidentes una vez se explique más detalladamente el funcionamiento del algoritmo.

Otro proceso de la figura 4.10 que se hace necesario detallar es la evaluación de incógnitas, su propósito es determinar si las incógnitas pueden ser sustituidas por nuevos nodos dentro de la ruta. Esta función fue particularmente difícil de definir, ya que debe evaluar nodos extremos dentro de la cadena, su objetivo puede ser comprendido más fácilmente de forma gráfica con el árbol de la figura 4.8 y tomando como ejemplo el nodo D.

Si una cadena que representara un árbol binario, tal como el de la figura pero desarrollado hasta el nodo D, se introdujera a la función “evaluar incógnitas”, básicamente lo que se determinaría es si existe la posibilidad de incorporar nuevos nodos en el nivel inferior de las ramas. En este caso se determinaría que existen los nodos intermedios F y G. Su posición en la ruta indicaría que F se encuentra entre los nodos A y D, así como G se encuentra entre los nodos D y C.

Para determinar si es posible incorporar un nuevo nodo, es necesario evaluar la distancia entre los dos extremos. Uno de ellos es el nodo al final de la rama, en el caso del ejemplo uno de los extremos seria el nodo D. Lo complicado de la función fue determinar cual es el otro extremo a evaluar. Gráficamente resulta simple determinar que para un nodo inferior izquierdo debe evaluarse la distancia entre nodo en cuestión y el nodo superior al extremo izquierdo del árbol, por ejemplo A y D. De igual forma para un nodo inferior derecho debe evaluarse la distancia del nodo actual con el nodo superior derecho, por ejemplo D y C.

A nivel de código la forma de encontrar los nodos cuya distancia se debe evaluar debe ser desplazándose a lo largo del genoma. Con la codificación de cada gen como padre-derecho/ nodo-central/ padre-izquierdo, el genoma proporciona suficiente información para recrear la cadena.

Al continuar con el ciclo de evolución de acuerdo a la figura 4.9 una vez se logran evolucionar los individuos obtenidos de la población mas apta, es decir la población elite, comienza el proceso de volver a generar una población mayor de la cual se puedan seleccionar nuevas rutas mas convenientes.

Al salir del proceso de evolución es posible que se disponga de nuevos nodos intermedios que deben ser definidos, en la siguiente etapa se asignan valores a estos nuevos nodos. Esta asignación se pretende hacer de forma aleatoria, pero con la intención de hacer más eficiente el programa se manejará bajo una regla simple.

El nuevo nodo seleccionado para punto medio se genera con los nodos extremos como limites.

Esta regla se ha impuesto con el propósito de controlar las distancias recorridas en la ruta, ya que sin esta puede darse el caso que las poblaciones lleguen a ser en su mayoría rutas de largas distancias que aún cuando serían descartadas por el mismo algoritmo incrementarían el tiempo de procesamiento.

La siguiente etapa es aumentar la población, la cantidad de individuos que se podrán generar en esta etapa debe ser definida. Para multiplicar la cantidad de individuos se puede utilizar el cruce de manera similar a la descrita en la figura 4.7, por supuesto debe adaptarse a los cambios hechos en la codificación del genoma.

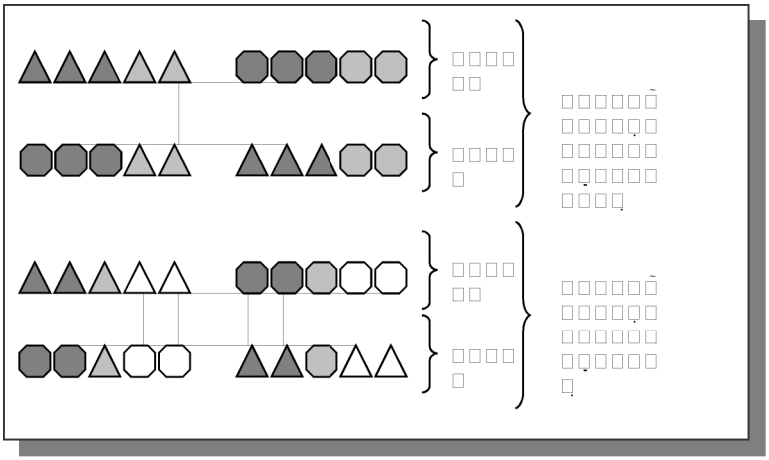


Figura 4.11 – Tipos de cruces comunes en los algoritmos genéticos

En esencia el cruce consiste en mezclar las características de una pareja de individuos. La técnica conocida como “crossover” cuza una secuencia de genes en ambos padres generando así dos nuevos individuos hijos. Los cruces entre pareja pueden ser de dos tipos, singular y doble conocidos como single-point-crossover y double-point-crossover.

Como se puede ver en la figura 4.11, la diferencia esta en la forma en que se segmenta el genoma, en el cruce singular el genoma se divide en dos cadenas, una que se mantiene estática y se complementa con genes del otro individuo padre; la otra cadena se trasfiere para complementarse con la parte estática del otro individuo. El en cruce doble son dos cadenas diferentes las que se intercambian con el otro padre, en la figura estas se pueden diferenciar por los colores utilizados.

En este problema, los genomas que representan las rutas no pueden realizar cruces tan libremente o de manera aleatoria. Aún cuando la codificación esta diseñada de tal forma que el árbol continúe creciendo hasta unir todos sus nodos, realizar cruces sin reglas podría terminar en poblaciones enteras con rutas sin sentido o ciclos infinitos. Es por esto que el algoritmo necesita de un procedimiento definido para realizar cruces efectivos.

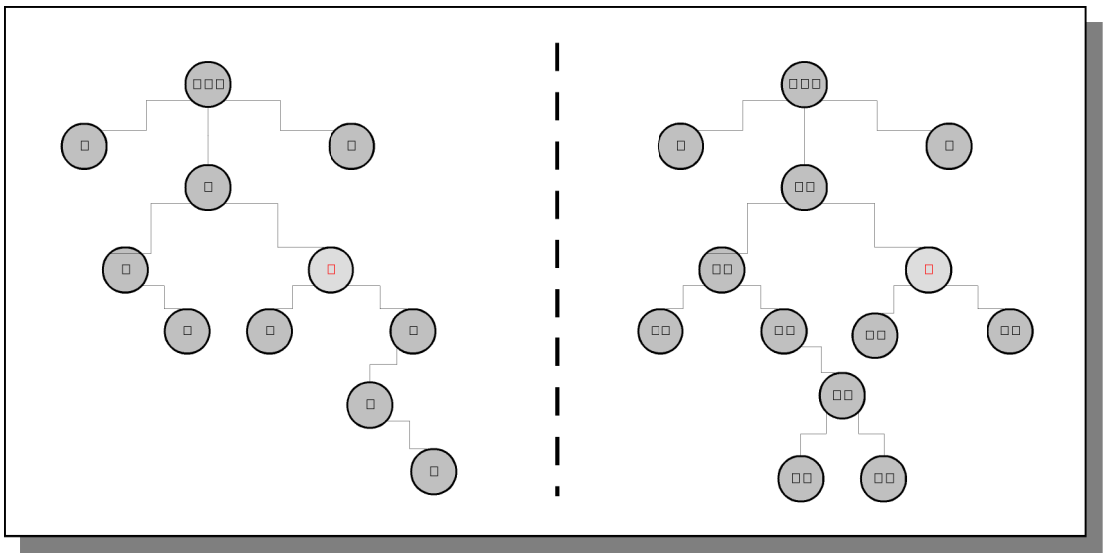


Figura 4.12 – Dos individuos diferentes que por tener un nodo común permitirían, en la etapa de cruce y sobrepoblación, engendrar otros dos individuos diferentes.

Como norma principal y para mantener la secuencia de las rutas en desarrollo, el cruce entre dos individuos se podrá realizar únicamente si ambos poseen nodos padres iguales. Esto garantiza la coherencia dentro de los genomas y aunque descarta la posibilidad de realizar cruces dobles, permite mezclar genes de diferente nivel jerárquico, lo cual puede resultar en optimizaciones muy significativas.

En el ejemplo de la figura 4.12 es posible cruzar a los dos individuos gracias al nodo que tienen en común. En este ejemplo el cruce se realizaría en el mismo nivel jerárquico, sin embargo, engendran árboles bastante diferentes.

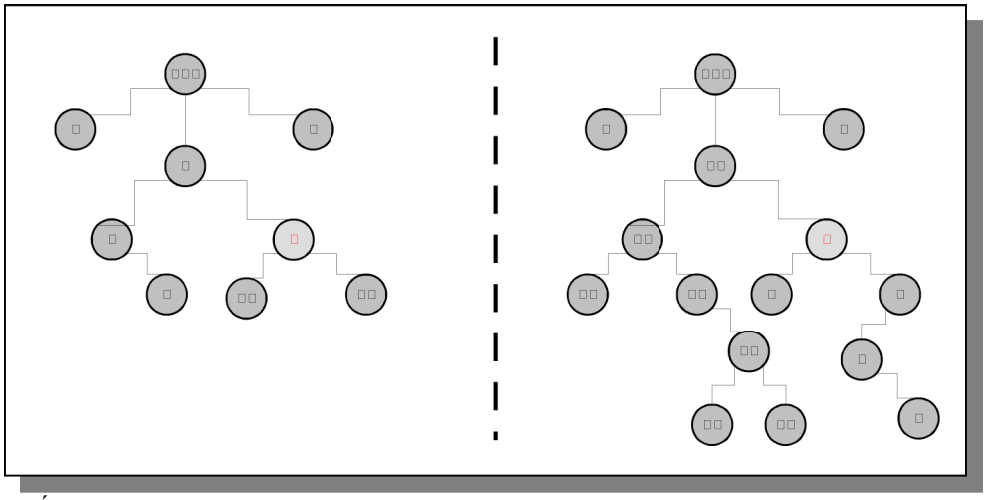


Figura 4.13 – Árboles binarios que son producto del cruce singular de los árboles mostrados en la figura anterior.

En las etapas finales del ciclo se realiza la mutación y la selección natural para formar la reducida población elite. La mutación, tal como se observo en los casos anteriores, resulta más eficiente si se hace en función de la edad de los genes que con esta notación no será necesario calcular. Finalmente la selección se realiza de acuerdo a la aptitud del individuo.

Capitulo V
Manual de usuario

En este capitulo se describe el funcionamiento del programa desarrollado para ejemplificar el funcionamiento de los algoritmos genéticos. El programa consiste en encontrar la ruta mas apta para trasladarse entre dos puntos en un mapa diseñado específicamente para simular el tráfico en una ciudad real.

Una de las características más notables de este programa es que, por ser una aplicación principalmente demostrativa, esta diseñando para operar por completo en una única pantalla que pueda desplegar toda la información necesaria. Fue una de las propiedades que al momento del diseño se evaluaron, y que para evitar distracciones innecesarias de parte del usuario final se concluyo incorporar controles que facilitaran el manejo de toda la información.

De igual forma en el programa no se incorporaron barras de menú. Por lo general, los programas que trabajan en base a algoritmos genéticos, contienen una excesiva cantidad de opciones para personalizar el desempeño del algoritmo. Sin embargo, en este caso las opciones de configuración y operación han limitado a las variables genéticas básicas, a fin de evitar confusión innecesaria.

A continuación se describirá con detalle cada uno de los componentes de la interfaz a utilizar, y se definirán brevemente las funciones que se ejecutan en cada etapa del algoritmo. Es necesario aclarar que para que un usuario pueda comprender el algoritmo y que la aplicación tenga la utilidad proyectada, se espera que los usuarios hayan leído el contenido de este documento previamente o bien tengan conocimientos previos de estos modelos de programación.

5.1 Interfaz inicial del programa.

En un primer instante se despliega de forma maximizada la siguiente pantalla, es muy importante tener en cuenta que antes de poder ejecutar cualquier rutina involucrando algoritmos genéticos, es indispensable establecer todos los parámetros de su funcionamiento. A continuación se describen los pasos necesarios para proporcionar toda la información requerida.

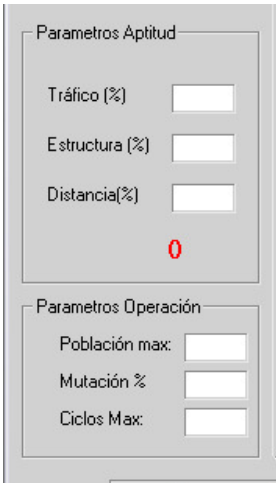
Figura 5.1 – Presentación inicial del programa

La figura 6.1 muestra como inicializa la pantalla sin ningún dato ingresado hasta el momento. Para comenzar a llenar la información requerida, el programa utiliza una serie de validaciones que primero solicitan los parámetros de aptitud y los parámetros de operación.

En el primer recuadro que hace referencia a los parámetros de aptitud se pueden asignar las ponderaciones que se les dará a las variables tomadas en consideración para encontrar una ruta óptima. La operación de estás entradas parte del supuesto que para cada usuario las variables de trafico tienen una importancia diferente, es decir, que lo que podría ser la ruta óptima para un usuario podría ser una de las menos aptas para otra.

Como era de esperarse, si se desea especificar la incidencia de una variable en el enfoque que tendrá el programa, la manera mas indicada de lograr esto es asignándoles porcentajes que reflejen su importancia en relación con las otras variables encontradas.

Para garantizar la validez de los datos, se utiliza una rutina sencilla que verifica la suma de todos los porcentajes y en la parte inferior del recuadro. De no completar el cien por ciento de aptitud, el programa despliega un mensaje de advertencia cada vez que se intente continuar a la siguiente etapa.



Note que no será permitido ingresar información referente a los nodos si los porcentajes no han sido completamente asignados. Esto se debe a que sin un cien por ciento distribuido entre las tres variables es imposible calcular la aptitud de cada nodo.

En el recuadro ubicado bajo los parámetros de aptitud se encuentran los parámetros de operación, estos están mas relacionados con la forma en que el algoritmo trabaja y son utilizados para delimitar el número de operaciones que a nivel de código el programa realiza. A diferencia del recuadro anterior, este no contiene ninguna validación especial puesto que las variables solicitadas actúan como límites y no como coeficientes de operación.

Otros datos necesarios para la ejecución de este programa son el nodo de partida y el nodo de llegada. Estos se deben incluir bajo el formato “fila , columna” tal como se manejan los nombres de los nodos a lo largo de la aplicación.

Todos los datos mencionados anteriormente son necesarios para que el programa pueda operar cadenas o genomas que reflejen las rutas diseñadas.

5.2 Captura y validación de nodos.

Tal como se mencionaba en los objetivos de este programa, el resultado que el algoritmo genético pretende generar debe estar basado en las propiedades de las rutas simuladas, estas son evaluadas por medio de las características de los nodos relacionadas a una ruta específica.

Otra información adicional que resulta crítica para el modo de operación de este algoritmo está relacionada a las entradas posibles al nodo. Es decir que para cada nodo es necesario especificar cuantas calles convergen en este nodo. El programa maneja hasta un máximo de ocho entradas que simulan calles con dirección hacia el centro del nodo. En el caso que un mapa requiera de un nodo con más accesos, estos se podrán simular modificando la codificación para agrupar un conjunto de nodos para que funcione como uno solo.

Figura 5.2 – Cuadricula de información de nodos

Como se puede observar en la figura anterior, a mano derecha se encuentra un conjunto de cajas de texto que se pueden utilizar para ingresar la información de cada nodo, el botón "Aceptar" ubicado bajo los parámetros ingresa el nodo al listado que se despliega a la izquierda de los mismos. Con este mismo juego de botones puede borrarse la información de un nodo en particular o toda la información registrada.

Existe la opción de importar toda esta información, puesto que ingresarla manualmente es un proceso que toma bastante tiempo. Con el botón importar en la parte superior de la pantalla se abre una ventana para buscar y seleccionar un archivo de texto que contenga la información respectiva.

Es importante notar que el programa esta limitado a 100 nodos, pero es específicamente una limitación de la interfaz gráfica que pretende facilitar la comprensión de estos algoritmos con un aspecto amigable. Sin embargo, el algoritmo en si podría evaluar una mayor cantidad de nodos, incrementando los recursos necesarios para operar y los tiempos de procesamiento.

Una de las características incluidas en la lista de nodos y que también busca facilitar el uso del programa es la posibilidad de modificar las propiedades de los nodos en tiempo real. Es decir que ya sea si los valores han sido importados o ingresados manualmente, pueden modificarse al hacer doble clic sobre una celda. Debido a que la aptitud del nodo se calcula en base a los porcentajes en los parámetros de aptitud, esta celda no puede ser modificada, de igual forma, dichos parámetros se bloquean para que no puedan modificarse una vez se comienzan a ingresar nodos a la lista. Esto se hace con el propósito de mantener la integridad en la aptitud de todos los nodos.

5.3 Población y población elite

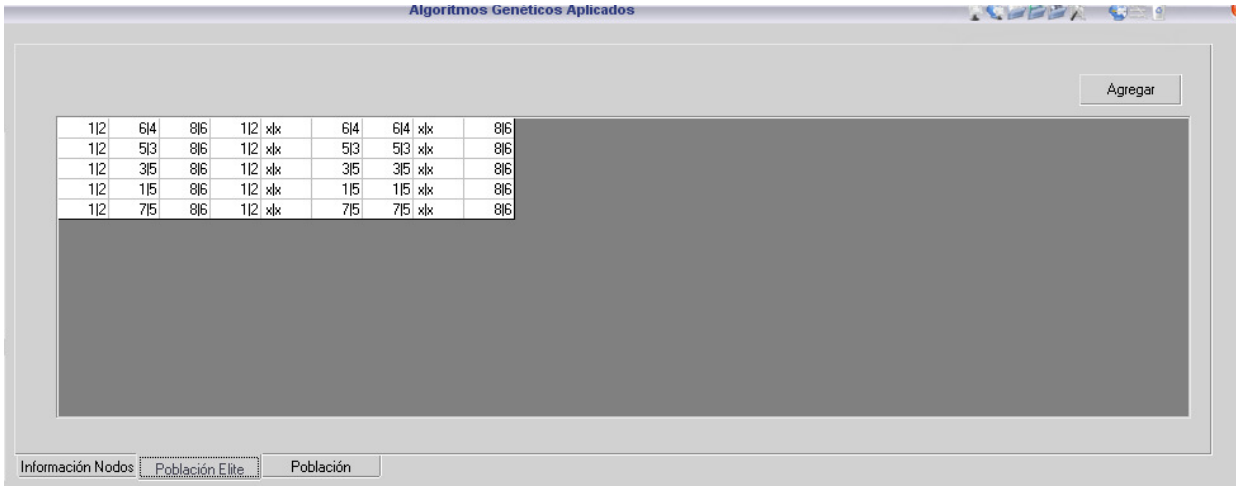


Figura 5.3 – Cuadricula de población elite.

En la parte inferior del recuadro central se encuentran tres viñetas que permiten visualizar las poblaciones de genomas y sus constantes cambios a lo largo del proceso de evolución. En el recuadro de la población elite, cada celda contiene la información de un nodo específico. Cada tres columnas se define un nuevo gen, si este se encuentra en desarrollo se podrá apreciar que el nodo intermedio todavía no ha sido definido.

En la sección de población se muestran todos los genomas o posibles soluciones que el algoritmo genera. En esta cuadricula es posible identificar cambios que sufren tanto a los genomas de forma individual por efecto de la mutación y desarrollo de nuevos genes, como la población entera que se ve afectada sobre todo por los cruces y la selección natural.

Para iniciar el ciclo evolutivo es necesario asignar una población elite inicial, que dará paso a las siguientes etapas del proceso. Esta población elite inicial no es más que un conjunto de cadenas con los nodos extremos y un nodo intermedio que permite desarrollar los nuevos nodos hasta generar la ruta buscada. Esta población elite inicial puede ser generada automáticamente utilizando el botón “Generar” o bien puede ser introducida separando las celdas por comas.

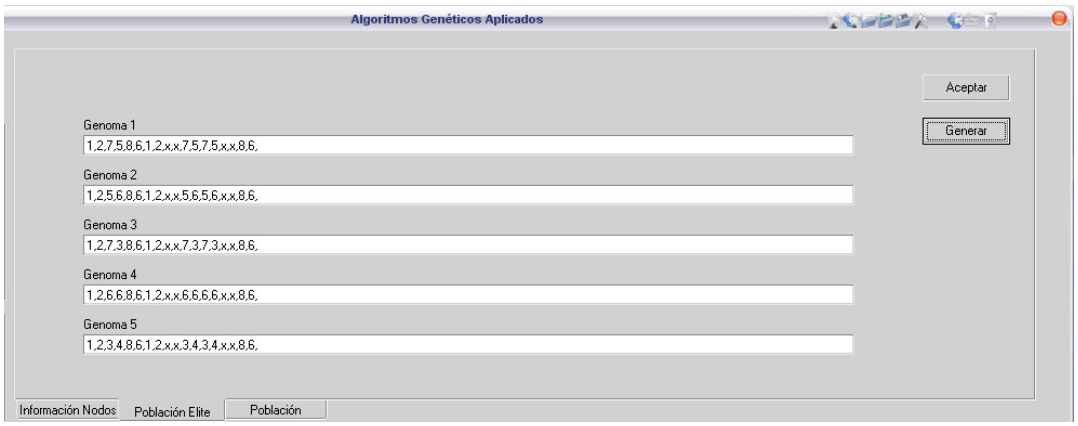


Figura 6.4 – Generación automática de población élite inicial.

Vale recalcar que la población elite inicial no tiene como requerimiento algún conocimiento previo del mapa o la conectividad entre nodos. El nodo intermedio utilizado no depende de ninguna información previa y es por eso que el algoritmo puede adaptarse a cualquier mapa generado.

5.4 Representación del genoma optimo.

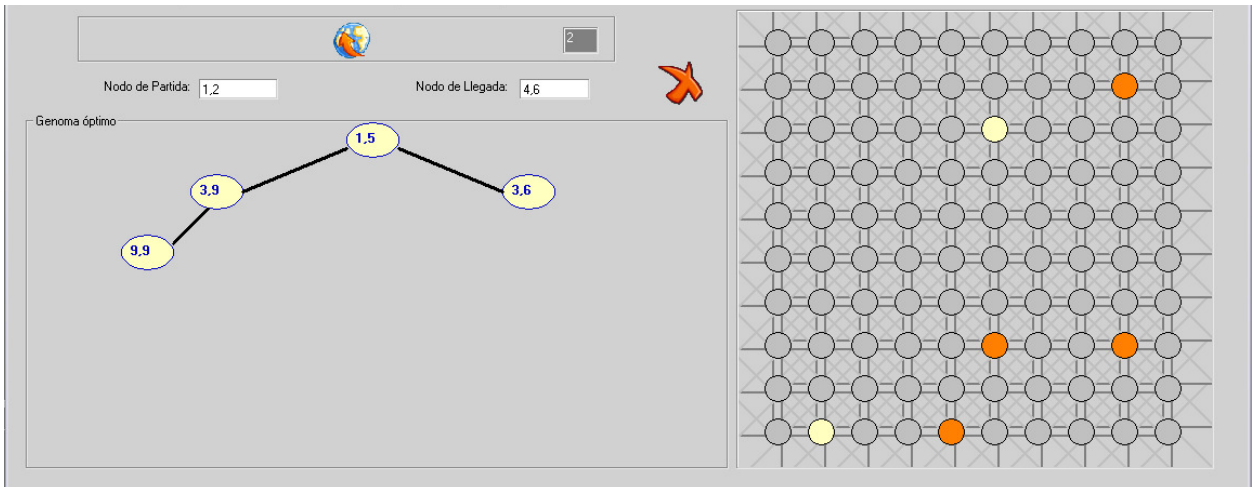


Figura 5.5 – Complementos gráficos del programa

La parte inferior de la ventana contiene dos recuadros cuyo propósito es proporcionar una representación visual de la evolución del genoma mas apto en la población. Al lado derecho, se puede observar un esquema de la infraestructura de la ciudad sobre la cual se muestran dinámicamente los puntos de intersección con sus respectivas entradas. En un estado inicial los nodos extremos son identificados y diferenciados por un tono claro (color amarillo).

Conforme el genoma optimo vaya evolucionando y nuevos nodos intermedios se van encontrando, aparecen en el esquema los nodos que formarán parte de la ruta óptima. En cada ciclo de evolución es esquema es regenerado con información actualizada. Un aspecto relevante de esta demostración gráfica es que al cargar la información listada en la tabla de nodos, el programa identifica cuales son las vías de acceso a cada nodo y resalta todas las arterias activas.

Simultáneamente, se genera un árbol binario que representa la codificación del genoma más óptimo, que aparece representado en el panel izquierdo de la ventana. Con este diagrama no se pretende mostrar el genoma completo, su finalidad esta orientada a presentar una idea de cómo se va estructurando el árbol, en la medida que la población evoluciona. En caso que la representación de la solución encontrada contenga un gran número de elementos el diagrama se verá limitado a los primeros cuatro subniveles lógicos del árbol binario.

5.5 Representación de las fases evolutivas.

En una barra ubicada al centro de la pantalla, se representan mediante iconos cada una de las fases en el ciclo de evolución. Mientras el programa evoluciona las cadenas correspondientes a cada genoma, oculta todos los iconos de la barra a excepción de aquel asignado a la función que en el momento se está ejecutando; de esta forma el usuario podrá identificar las alteraciones que cada fase desencadena en la población.

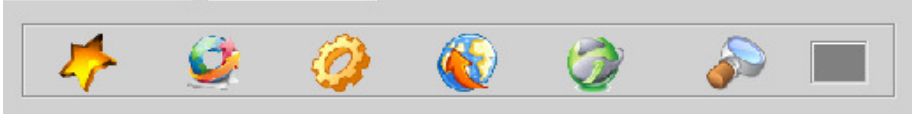


Figura 5.6 – Barra de iconos indicadores de su estado o fase.

En los párrafos siguientes se describen brevemente las funciones que se ejecutan en cada una de las fases evolutivas:

Evolución del genoma: Esta función tiene como propósito modificar la longitud del genoma e incorporar genes en desarrollo de ser necesarios. Realiza los cambios después de identificar el último gen definido, si este resulta ser un nodo terminal los nuevos genes serán nulos, de lo contrario los estos serán genes en desarrollo que darán paso a una posible nueva rama en el árbol binario,

Definición de gen intermedio: En general esta función se utiliza para que el programa pueda encontrar y definir el gen en desarrollo más antiguo. A partir de este, obtiene los nodos extremos, calcula la distancia entre ellos y verifica si son o no adyacentes, también comprueba si existe conectividad entre ellos y si es necesario sustituye el nodo desconocido por un nuevo nodo o un nodo terminal.

Reproducción y sobrepoblación: esta es la función que permite al algoritmo explorar diferentes rutas y combinarlas entre ellas para generar nuevas soluciones potenciales. El proceso para lograr este objetivo consiste en realizar búsquedas en el resto de la población para encontrar genes que coincidan con los nodos padres de manera que si existe una forma diferente de unir dos nodos, esta puede ser implementada en otro genoma.

Mutación: Esta función realiza cambios aleatorios en genomas elegidos al azar para que en cierta forma las cadenas que no pueden ser generadas por medio del cruce sean exploradas. La cantidad de mutaciones esta siempre sujeta al parámetro de operación respectivo. La mutación se realiza en el último gen definido, eso se debe a que mutaciones en los genes mas antiguos pueden generar inconsistencias en la población.

Selección natural: En la selección natural se ordenan todos los miembros de la población en función de su aptitud y se eliminan los genes marcados por el gen de supervivencia modificado en la función de mutación.

Población élite: El único propósito de esta función es actualizar el listado de genomas élite y la representación gráfica que se muestra en la parte inferior de la pantalla. Antes de finalizar la función prepara los datos existentes para iniciar un nuevo ciclo e incrementa el contador que indica el número de ciclos o generaciones.

Conclusiones finales

Probablemente todas las conclusiones encontradas después de desarrollar e implementar los algoritmos genéticos como herramienta optimización se encuentran orientadas a identificar cuando es o no conveniente considerarlos como una alternativa.

En este punto vale la pena hacer énfasis en la principal característica de este conjunto de instrucciones tal como se mencionaba en el capítulo tres. El paralelismo que estos algoritmos manejan procedente de la descendencia múltiple, permite explorar espacios más grandes de posibles soluciones. Esto hace que su uso sea ideal en búsquedas extensas que no requieren un análisis de datos exhaustivo pero que utilizando métodos tradicionales implicarían un enorme número de evaluaciones.

Claramente el tiempo de procesamiento de estos algoritmos se mucho menor al utilizado por la programación lineal, pero a su vez esta característica limita su capacidad de garantizar un valor optimo absoluto. Por otra parte, la capacidad de manejar múltiples resultados posibles simultáneamente tolera en estos programas errores emergentes que en otras circunstancias requerirían reiniciar todo el proceso. En un programa como el desarrollado en este estudio, en el que alternativas pueden en cierto momento perder características que afecten su aptitud o que estén expuestas a constante cambio, los algoritmos serán una buena primera opción.

A pesar de su versatilidad, estos algoritmos tienen la característica de que cambios sencillos tanto en la codificación o en las función de aptitud, puede tener grandes consecuencias sobre el desarrollo de genomas en la población. Por lo que en general, una eficiente codificación del genoma es determinante para encontrar una solución necesaria, de igual forma, definir adecuadamente la función operadora de aptitud garantiza que se en cada iteración se mantenga la orientación hacia la meta buscada. Se deben utilizar los algoritmos genéticos únicamente si es posible codificar por completo la información y si se cuenta con un lenguaje capaz de programar la función de aptitud.

Una conclusión final relacionada a este tipo de algoritmos y que sin embargo no esta relacionada con la implementación de estos algoritmos, es referente al desarrollo de los algoritmos genéticos como estrategia de programación y optimización. Aún cuando la computación evolutiva ha sido sujeto de estudio por poco menos de treinta años, la mayoría de los logros significativos han sido publicados en la última década. En este sentido, los algoritmos genéticos siguen siendo una estrategia de solución en desarrollo, la cual aun con todo su potencial, no ha sido limitada por reglas definidas y sigue siendo estudiada con la intensión de encontrar nuevas aplicaciones para tecnologías emergentes.

Tal como se menciona en el capítulo cuarto, se examinaron tres diferentes tipos de codificación e implementación de los cuales se seleccionó uno como mas apto para la solución de este problema específico. Las codificaciones no implementadas, aunque no resultaron aptas para los

fines de este programa clarificaban la utilidad de diferentes codificaciones y prácticas. Como resultado de estos intentos podemos formular las siguientes conclusiones.

Una codificación binaria, maximiza el impacto que tienen las modificaciones hechas al genoma ya que el cambio en uno de los dígitos es mas significativo que en cualquier otra codificación. Esto favorece el paralelismo del algoritmo, pero al mismo tiempo puede generar un exceso de procesos y operaciones que afecten negativamente el desempeño del algoritmo.

Una estrategia eficiente para optimizar la operación entre los genomas de una población, es el encapsular la información que no requiere ser directamente operada o que no sea tan relevante para los fines del programa. Mediante los ejemplos realizados quedo demostrado que no solamente facilita la programación de las operaciones genéticas sino que también reduce los tiempos de procesamiento necesarios para encontrar las soluciones buscadas.

Es correcto aseverar que en los algoritmos geneáticos se simula un ambiente natural en el que las mutaciones y el cruce de especies ser realiza principalmente de forma aleatoria. Poco después de los orígenes de estos algoritmos se comenzaron a desarrollar técnicas que permiten aislar datos o segmentos del genoma de manera que no sean afectados por el proceso de evolución.

La codificación de valores reales resulta particularmente útil en los casos en que los datos que se necesitan operar generan cadenas binarias excesivamente largas. Como se menciono anteriormente estas últimas demandan más recursos del sistema, pero a su vez garantiza la integridad de los datos ya que al realizar conversiones se dificulta la validación de la información.

Es muy común que en la codificación de genomas sea necesario incluir información que tiene asignada una posición relativa, es decir que utiliza un segmento preestablecido de la cadena y aun cuando los valores puedan cambiar dentro de un rango determinado, los valores pueden no ser compatibles con otro tipo de datos. Es por eso que con rutinas equivalentes a la técnica de inversión los algoritmos pueden adaptarse a las necesidades del programa.

Después de múltiples intentos de desarrollar una aplicación capaz de resolver el problema planteado, puede identificarse con facilidad que la etapa crítica de este proceso consiste en la codificación. Se puede concluir que una codificación inapropiada puede limitar un programa de tal forma que se convierta completamente inoperante. Por otro lado, una codificación eficiente que facilite la operación del algoritmo puede reducir recursos e incluso incrementar la precisión de los resultados.

El uso de un genoma con longitud variable no es recomendado para los casos tradicionales. El manejar genomas con longitudes idénticas facilita la programación e incluso permite el uso de recursividad en el código que maneja las poblaciones y su sobrepoblación que favorece la diversidad de la especie.

Si el algoritmo debe adaptarse a cambios constantes en el espacio que contiene los posibles resultados, una codificación de longitud variable resulta lo más indicada. Este tipo de codificación hace que el algoritmo sea capaz de adaptarse rápidamente a su entorno. Esta es una de las características que hace a algoritmos capaces incluso de generar incluso otros programas utilizando lenguajes como LISP.

La codificación utilizando árboles binarios explota las ventajas que estas estructuras de datos poseen. En este método, los cambios aleatorios pueden generarse cambiado el operador, alterando el valor de un cierto nodo del árbol o sustituyendo un subárbol por otro. Tal como se muestra en el programa realizado, este modo de codificar los datos puede modificarse y llega a permitir aplicar las operaciones genéticas sin necesidad de registrar operadores lógicos repetidos.

En la teoría tradicional de los algoritmos genéticos, se asume que estos trabajan en base a descubrimiento de nuevos nodos y combinaciones, mutaciones o selección en base a la aptitud. Estos algoritmos si bien son eficientes en casos donde la solución no se encuentra bien definida, presentan una serie de problemas notables que pueden llevarlo a generar resultados incluso hasta peores que una función aleatoria. Esto generalmente se debe a problemas de la codificación y si bien existe la posibilidad que su desempeño mejore con una codificación optimizada, también es posible llegar a encontrar problemas en los que este tipo de algoritmos no aplican.

Al igual que la codificación de genomas, los valores utilizados para las variables genéticas es de suma importancia al momento de ejecutar estos programas. Estas variables hacen referencia a parámetros porcentuales como la tasa de mutación, la tasa de cruces entre individuos, cantidad de individuos permitidos en una población etc., toda esta información determina el funcionamiento de la aplicación.

Entre más grande es una población, mas posibilidades tiene de explorar y diversificar la población de posibles soluciones. Esto proporciona la ventaja de se pueden evaluar alternativas muy diferentes y mantener la misma tendencia hacia la solución deseada. Al mismo tiempo, la diversificación trae consigo misma la desventaja de que reduce la especialización. Tal como ha sucedido con todo organismo vivo, exploración y especialización son inversamente proporcionales en el proceso de evolución. Un número recomendado para la cantidad de genes en la población es de veinte a cuarenta genomas. Por supuesto esto depende del algoritmo, la función de aptitud y la complejidad de la cadena que se estará ocupando.

La tasa de cruce entre miembros de la población y la tasa de mutación permiten encontrar espontáneamente soluciones que en otras circunstancias no podría ser encontrada. Pero estas tasas deben ser asignadas con cuidado al iniciar el proceso, ya que de estas depende, en gran parte, si el programa logrará encontrar o no una solución deseada.

Actualmente los algoritmos genéticos se plantean como una alternativa eficiente para la solución de problemas en los que la programación tradicional resulta inoperante. Sin embargo lo que hace realmente atractiva a esta manera de programar es su enorme potencial, que ha permanecido en desarrollo por años y sin embargo aún se tienen altas expectativas en cuanto a nuevas aplicaciones de inteligencia artificial, es por eso que se recomienda al lector indagar en el tema e incorporar estas estrategias de programación en su desarrollo de soluciones de software.

Bibliografía

A.- Libros consultados

Inteligencia Artificial, una nueva síntesis
Nils J. Nilsson
McGraw Hill

Design Using Genetic Algorithms
William F. Punch, Ronald C. Averill, Erik D. Goodman, Shyh-Chang Lin, Ying Ding
IEEE Expert, Jan 1995.

Data mining neural networks with genetic algorithms
Ajit Narayanan, Edward Keedwell and Dragan Savic.
School of Engineering and Computer Science
University of Exeter

B.- Sitios Web consultados

http://www.rennard.org/alife/english/entree.html (5 de Agosto 2006)

http://qudata.com/lib/genetic_algorithms/ (5 de Agosto 2006)

http://cs.felk.cvut.cz/~xobitko/ga/ (5 de Agosto 2006)

http://www.cs.bgu.ac.il/%7Esipper/ga.html (5 de Agosto 2006)

Anexos