HERRAMIENTAS PARA LA IMPLEMENTACIÓN DE ALGORITMOS GENÉTICOS EN INGENIERÍA CIVIL CON ÉNFASIS EN HIDROINFORMÁTICA

RAFAEL ERNESTO OLARTE VALDIVIESO

TRABAJO DE GRADO
Presentado como requisito parcial
para la obtención del título de
INGENIERIO CIVIL

DIRECTOR
Ing. NELSON OBREGÓN NEIRA Ph.D.

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA CIVIL
BOGOTÁ D.C.
2003

REGLAMENTO DE LA PONTIFICIA UNIVERSIDAD JAVERIANA.

Artículo 23. "La Universidad no se hace responsable por los conceptos emitidos por sus alumnos en sus trabajos de tesis. Sólo velará por que no se publique nada contrario al dogma y a la moral católica y por que las tesis no contengan ataques personales contra persona alguna, antes bien se vea en ellas el anhelo de buscar la verdad y la justicia".

Resolución N^{o} 13 de julio de 1946.

El proyecto de grado titulado: “HERRAMIENTAS PARA LA IMPLEMENTACIÓN DE ALGORITMOS GENÉTICOS EN INGENIERÍA CIVIL CON ÉNFASIS EN HIDROINFORMÁTICA” presentado por Rafael Ernesto Olarte Valdivieso, en cumplimiento parcial de los requisitos exigidos para optar al título de Ingeniero Civil, fue aprobado el día _____ del mes de_____ de _____.

_____

Ing. Nelson Obregón
DIRECTOR

_____

Geól. Federico Fragala
JURADO

Bogotá D.C., 1º de julio de 2003

RESUMEN

En diferentes ciencias e ingenierías alrededor del mundo, los Algoritmos Genéticos constituyen una herramienta computacional que ha conducido a la resolución de problemas más complejos a los permitidos por metodologías tradicionales. Sin embargo, la ingeniería civil en Colombia no está aún suficientemente familiarizada con esta técnica y no ha publicado material de referencia básico que permita su fácil comprensión. El presente proyecto tiene por objeto (1) explicar los aspectos teóricos fundamentales de los algoritmos genéticos y de su aplicación a la ingeniería civil y (2) exponer en detalle un caso estudio que consiste en calibrar dos modelos hidrológicos lluvia-escorrentía aplicados a dos cuencas en la sabana de Bogotá. Mientras la primera cuenca se calibró satisfactoriamente, la segunda sirvió de base para exponer qué aspectos, tanto relacionados como no relacionados con los algoritmos genéticos, no permitieron su calibración. Explicaciones, ilustraciones, diagramas, ejemplos y la exposición de todo el código del software requerido para el caso estudio buscan que el principal resultado del presente proyecto sea la comprensión e inmersión del ingeniero civil en los algoritmos genéticos.

Palabras clave: algoritmos genéticos, hidroinformática, sistemas inteligentes, inteligencia artificial, soft computing, inteligencia computacional, algoritmos evolutivos, calibración, modelos hidrológicos, modelos lluvia-escorrentía, matlab, ingeniería civil, Curibital, Subachoque.

ABSTRACT

In different fields of science and engineering worldwide, Genetic Algorithms constitute a computational tool which has led to the resolution of more complex problems than those allowed by traditional methods. Nevertheless, civil engineering in Colombia is still not sufficiently familiarized with this technique and has not published basic reference material for its easy understanding. The aim of this project is to (1) explain the fundamentals of genetic algorithms and their application in civil engineering, and (2) illustrate in detail one case study consisting of the calibration of two hydrologic rain run-off models applied to two basins located at the Savannah of Bogotá. While the first watershed was calibrated with success, the second served as an illustration of the aspects, related and not related with genetic algorithms, that did not allow its calibration. Explanations, illustrations, diagrams, examples and the full exposure of the software's code required for the case study, all these are included looking toward the civil engineer's comprehension and the immersion in genetic algorithms.

Keywords: genetic algorithms, hydroinformatics, intelligent systems, artificial intelligence, soft computing, computational intelligence, evolutionary algorithms, calibration, hydrologic models, rain run-off models, matlab, civil engineering, Curibital, Subachoque.

AGRADECIMIENTOS

Quiero agradecer a Nelson Obregón por haberme introducido en tan interesante tema y por el ánimo e interés que siempre me prestó. Él hizo que desde un principio, todo se viera posible. Agradezco a Federico Fragala por su atenció y colaboración.

Doy gracias a mis padres por su paciencia, su comprensión y su apoyo para que yo culminase con éxito este trabajo. Doy gracias también a mi hermano Sergio por su interés, a Miguel por su ayuda en momentos cruciales y a Isabel por su cariño y las ganas que me transmitió.

CONTENIDO

LISTA DE FIGURAS

pág.

LISTA DE TABLAS

pág.

LISTA DE ECUACIONES

PLANTEAMIENTO DEL PROBLEMA

ANTECEDENTES Y JUSTIFICACIÓN

A principios de la década de los sesentas, las computadoras se volvían cada vez más asequibles y menos costosas. Muchos investigadores en Estados Unidos, se sintieron atraídos por el potencial de estas máquinas[1]. Una de dichas personas fue el ingeniero y a la vez sicólogo, John H. Holland. Él, además de su interés por las computadoras, se concentró en otra materia: el estudio de los mecanismos observados en la naturaleza concernientes a la adaptación de las especies. Quiso entonces Holland aplicar dichos mecanismos en la creación de sistemas artificiales (sistemas simulados por computadora)[2,3]. Estas investigaciones llevaron al surgimiento de una metodología denominada *los algoritmos genéticos*. Dichos algoritmos se pueden considerar entonces, como la adaptación de teorías evolutivas a procedimientos artificiales para la resolución de problemas. Es por esta razón que en los algoritmos genéticos (o AGs) se encuentran vocablos tales como *cromosomas*, *poblaciones* o *mutaciones*.

Años después con su tesis doctoral, **Adaptation in Natural and Artificial Systems** (1975) y con la labor de sus estudiantes en la Universidad de Michigan, Holland dio origen a la teoría formal y al empleo de los algoritmos genéticos[4]. Campos como la economía, la investigación de operaciones, la medicina, la inteligencia artificial y las ingenierías, se vieron beneficiados por esta nueva metodología[†]. Es interesante además saber que la ingeniería civil no demoró en darle un espacio de acción a los AGs (1981 - 1983)[5].

A pesar de todas las aplicaciones reales que se le están dando en el mundo, en Colombia su uso es aún incipiente (ver Figura 1). Existe además una ausencia de material de referencia básico sobre el tema[6], sobretodo enfocado a la ingeniería civil. Es por esta razón, que se vio conveniente realizar el presente trabajo de grado *Herramientas para la Implementación de Algoritmos Genéticos en Ingeniería Civil con Énfasis en Hidroinformática*. Se busca así, incentivar posteriores investigaciones de ingeniería civil en nuestro país. Para ello, este trabajo dará no sólo un marco teórico adecuado, sino un software ilustrativo para el estudio de un caso en Colombia relacionado con la hidroinformática.

---

[1] DE JONG. 1999. p. 1.

[2] KARR and FREEMAN. 1999. p. 6

[3] GOLDBERG. 1989. p. 1.

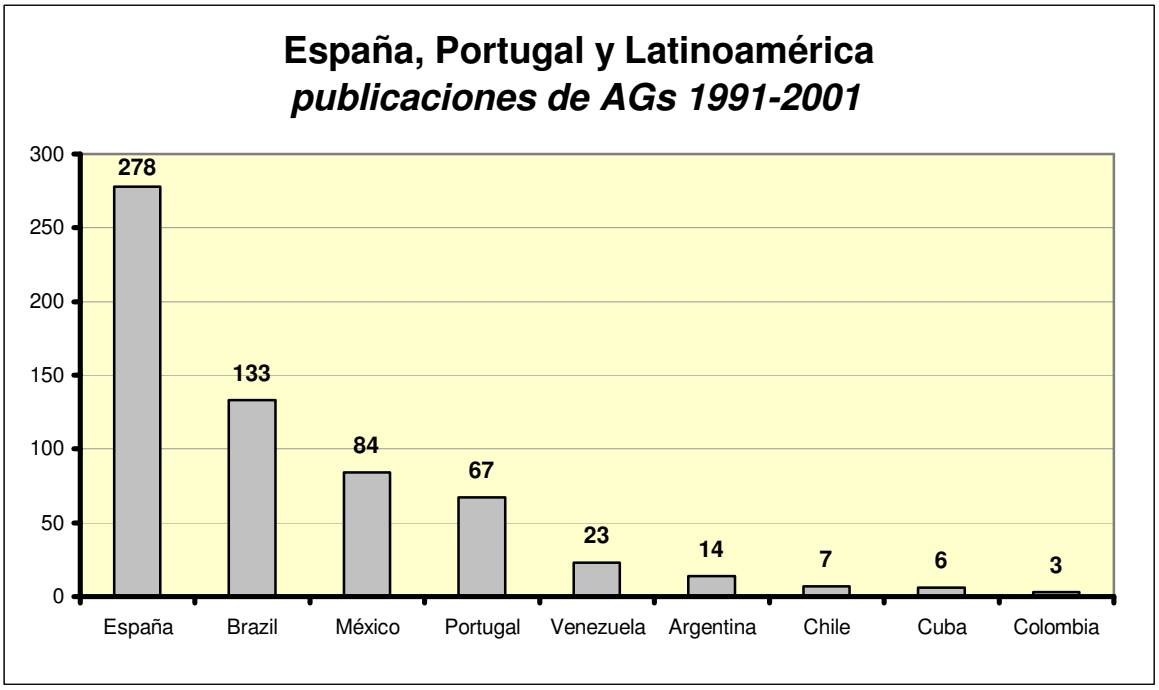
[4] GOLDBERG. 1989. p. 92.

[†] Una forma rápida de enterarse de todas estas aplicaciones es a través de la siguiente base de datos disponible en línea. ALANDER. 2003. p. 114.

[5] GOLDBERG. 1989. p. 126.

[6] MARTÍNEZ y ROJAS. 1999. p. 6.

Figura 1. *Contribuciones de AGs hechas por España, Portugal y Latinoamérica*[†]*.*



FORMULACIÓN DEL PROBLEMA

Ante la escasa familiaridad de la ingeniería civil colombiana con los algoritmos genéticos, cómo estimular entonces su implementación? ¿Cómo mostrar ante los ingenieros civiles las grandes ventajas que sí se están aprovechando en otros países?

[†] Las contribuciones hacen referencia a libros, partes de colecciones, artículos, ponencias, reportes, tesis de doctorado y tesis de maestría. La base de datos que se empleó como referencia reconoce una carencia de información entre los años 1999 y 2001. ALANDER. 2002. p. 5, 7.

OBJETIVOS

OBJETIVO GENERAL

Proporcionar herramientas para la implementación de metodologías de algoritmos genéticos en la ingeniería civil, particularmente en el área de la hidroinformática.

OBJETIVOS ESPECÍFICOS

- Proporcionar un marco conceptual para la utilización de los algoritmos genéticos en la ingeniería civil.

- Desarrollar e implementar un aplicativo computacional de algoritmos genéticos en el contexto de la hidroinformática.

INTRODUCCIÓN

Una revisión preliminar permite concluir que los algoritmos genéticos (AGs) no constituyen un tema familiar para la ingeniería civil en Colombia[†]. Con miras a cambiar este panorama, el presente proyecto busca que, mediante una clara ilustración de los AGs y mediante la realización de un programa computacional aplicado a la realidad colombiana, se suministren así herramientas suficientes para incentivar la investigación en este campo. Por lo tanto, el presente proyecto contendrá tanto un componente teórico como un componente práctico.

En la Figura 2 se indica cómo se irá introduciendo al lector en el tema desde una perspectiva global (*los sistemas inteligentes*) hasta un punto particular (la calibración de un modelo hidrológico mediante un software). Ya que la línea allí planteada (1-2-3-4-5-6-7-8) exige abarcar una temática muy extensa, el presente proyecto sólo se concentrará en los temas marcados de amarillo, es decir, los algoritmos genéticos en sí, sus aplicaciones en la hidroinformática y el desarrollo del aplicativo computacional.

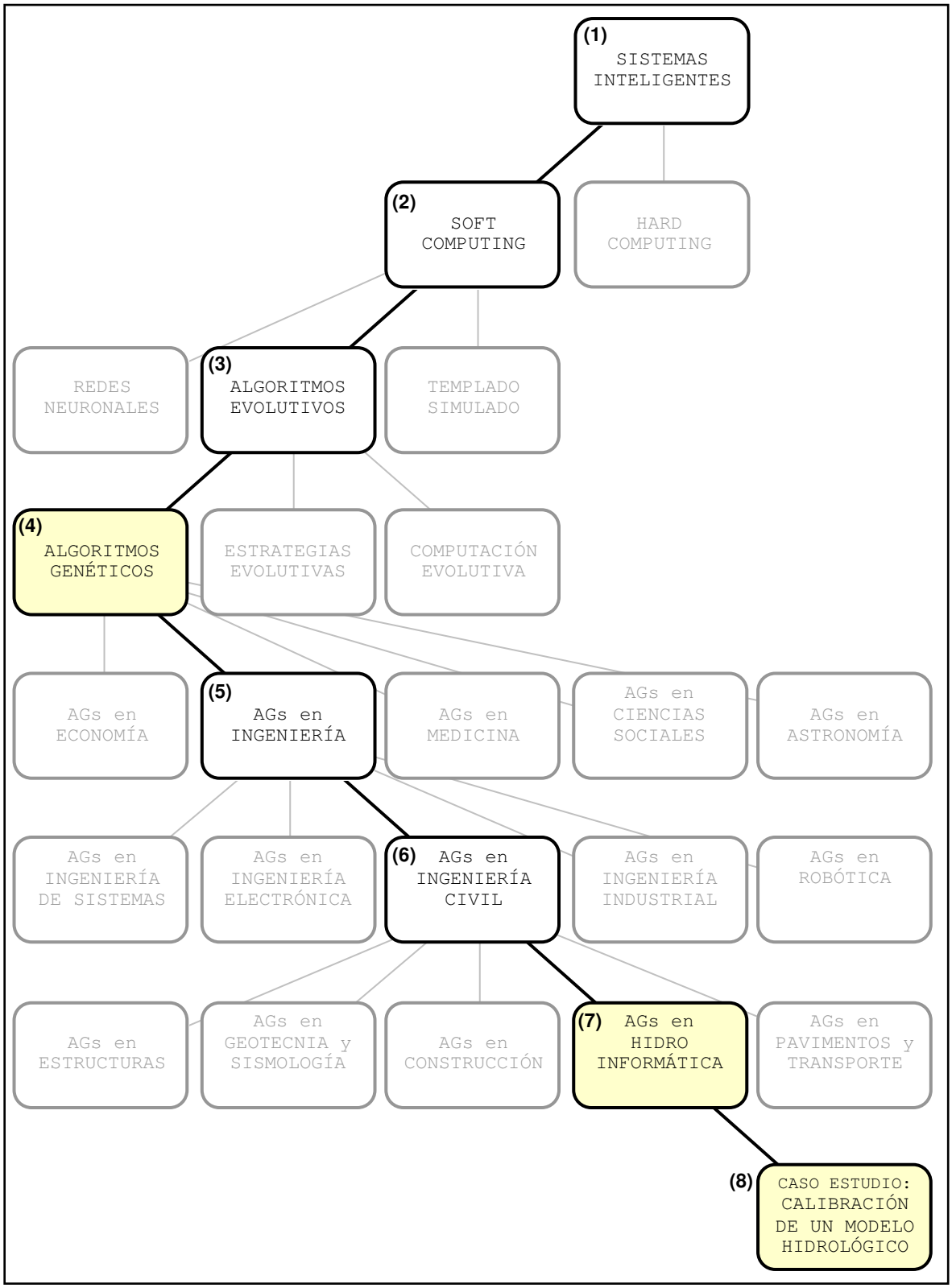
El primer capítulo ...1. Algoritmos Genéticos... comienza por identificar a los AGs como una herramienta más de los denominados sistemas inteligentes (...1.1. ¿Quiénes son?...). Luego explica formalmente qué son los AGs (...1.2. ¿Qué son?...) y cómo surgieron hasta llegar a extenderse su uso en las diferentes ciencias e ingenierías (...1.3. ¿Cómo surgieron?...). Después trata con mayor rigor sobre el funcionamiento de los AGs (...1.4.¿Funcionan?....) y finalmente, expone las áreas en donde los AGs se están empleando, especialmente en lo relacionado a la ingeniería civil y a la hidroinformática.

En el segundo capítulo ...2. Caso Estudio... se expone cómo se aplicaron los algoritmos genéticos en la calibración de un modelo lluvia-escorrentía para dos cuencas localizadas en la sabana de Bogotá (Subachoque y Curibital). Para ello, se desarrolló un aplicativo computacional. En la primera parte (...2.1. Descripción teórica del problema...) se explica en qué consiste la calibración de modelos desde un punto de vista general. En ...2.2. Descripción del caso estudio... se entra a la situación concreta de lo que se va a analizar. Previo a la descripción del programa, se resalta en ...2.3. Aspectos previos al desarrollo del programa..., todos los detalles que se deberán tener en cuenta para así responder a la pregunta, qué tipo de software se desea desarrollar. El capítulo finaliza con los apartados ...2.4. Descripción del programa..., ...2.5. Ejecutción..., …2.6. Resultados… y ...2.7. Análisis de los resultados....

Por último se dan unas conclusiones y recomendaciones acerca de lo expuesto en la parte teórica y acerca de los resultados obtenidos en el caso estudio. Se optó por incluir en dos anexos el código del programa y sus resultados. Así, se estimulará la mejor comprensión de los algoritmos genéticos y se podrá satisfacer a aquellos que se interesen por el tema.

[†] Por ejemplo, los documentos que representan a Colombia en la Figura 1 no son de ingeniería civil.

Figura 2. *Esquema conceptual que seguirá el presente proyecto.*



1. ALGORITMOS GENÉTICOS

El presente capítulo constituye el componente teórico del proyecto. Aquí no sólo se explicará qué son los algoritmos genéticos, sino que se dará también una perspectiva histórica y se expondrán las diversas aplicaciones que están teniendo en la ingeniería civil y en especial, en la hidroinformática.

1.1. ¿QUIÉNES SON? *UNA PERSPECTIVA DESDE LA INTELIGENCIA ARTIFICIAL*

Algunos ingenieros todavía no habrán oído hablar de algoritmos genéticos (AGs). Otros sí. Pero casi siempre, junto a este tema también se oye hablar de *redes neuronales*, *lógica difusa*, *sistemas expertos*, etc. Bajo todos estos términos se esconde una ciencia, la *inteligencia artificial* (IA). Para aquellos escépticos que piensan que la inteligencia artificial no tiene relevancia alguna en la ingeniería civil, la presente sección permitirá al menos, sospechar lo contrario.

Los algoritmos genéticos, de cuyas bondades se hablará en otras secciones, son de hecho una de las tantas herramientas computacionales (llámense *sistemas inteligentes*) que le ha legado la inteligencia artificial a científicos e ingenieros de otras áreas. Aunque no fueron precisamente investigadores en IA los que inventaron la teoría de los AGs (ni mucho menos los que ejercen la ingeniería civil), vale la pena detenerse por un momento para comprender varios aspectos de esta ciencia: ¿Qué quiere decir que un algoritmo genético sea un sistema inteligente? ¿Dónde se ubican los AGs dentro de este conjunto de herramientas computacionales? La inteligencia artificial es así, un buen punto de partida para comprender quiénes son los algoritmos genéticos.

El término "inteligencia artificial" fue propuesto inicialmente por el matemático John McCarthy en 1955[7] y fue adoptado luego en el verano de 1956 tras una convención que él organizó en el Darmouth College de los Estados Unidos[8]. Aunque desde la segunda guerra mundial, ya se habían comenzado a plantear las primeras ideas y los primeros trabajos, fue hasta dicha convención que se reconoció formalmente el nacimiento de esta nueva ciencia y se acordó bautizarla con el nombre de inteligencia artificial.

Aunque decir cuándo se originó la IA es tal vez fácil, definirla no lo es. No existe todavía un consenso entre los expertos al respecto. Sin embargo, todos ellos al definirla recaen de alguna u otra forma en uno de los siguientes cuatro conceptos.[9]

[7] McCARTHY *et al*. 1955.

[8] RUSSELL and NORVIG. 1995. p. 17.

[9] RUSSELL and NORVIG. 1995. p. 5.

IA es la ciencia que busca crear sistemas que...

(1) piensen como humanos.
(2) actúen como humanos.
(3) piensen racionalmente.
(4) actúen racionalmente.

Estos cuatro conceptos o filosofías, dieron origen a dos corrientes diferentes: la *IA humana* y la *IA ajena*[10]. En general, las dos corrientes se proponen crear sistemas que actúen o piensen racionalmente. Sin embargo la primera exige además, que estas propiedades se logren tal como lo hace el ser humano.

Con estas metas, la inteligencia artificial a través de sus investigaciones comenzó a alimentarse de diferentes filosofías (materialismo, empirismo, positivismo lógico, etc) y de diferentes ciencias (matemáticas, lógica, sicología, informática, lingüística, etc)[11]. Este proceso la enriqueció convirtiéndola entonces en una ciencia multidisciplinaria. Prueba de ello, es el hecho de algunos de los primeros investigadores en IA procedían de diferentes contextos: Warren McCulloch†, era neurofisiólogo, Bruce Buchanan y Joshua Lederberg eran filósofo y genetista respectivamente‡.

Hoy sesenta años después, la inteligencia artificial ha hecho avances, creando sistemas novedosos, interesantes, complejos en algunos casos, pero que todavía no llegan a esa meta exigente de generar una actuación o un pensamiento verdaderamente racional[12].

No obstante, los sistemas que se han logrado hasta el momento, han llegado a un grado tal de sofisticación que ahora, científicos e ingenieros de otras áreas se han visto interesados en sacarles provecho para sus respectivas disciplinas. Estos sistemas son en general *herramientas computacionales* o *software*¶, que algunas personas se atreven a denominar *sistemas inteligentes*[13] (ver Figura 3). En realidad, este despertar de las otras ciencias por la IA no es reciente. El empleo de los sistemas inteligentes en la industria por ejemplo, ya se ha venido dando en los países desarrollados desde 1980[14].

Los *sistemas expertos*, las *redes neuronales*, los *agentes* y los *algoritmos evolutivos* son ejemplos de sistemas inteligentes, los cuales están dotados de competencias que les

[10] COPELAND. 1996. p. 54, 97-98, 136, 312-314.

[11] RUSSELL and NORVIG. 1995. p. 8-16.

† Es considerado el primer autor de un trabajo en IA. (1943). Véase RUSSELL and NORVIG. 1995. p. 16.

‡ Fueron creadores del reconocido programa DENDRAL (1969). Véase RUSSELL and NORVIG. 1995. p. 22.

[12] HOPGOOD. 2001. p. 4.

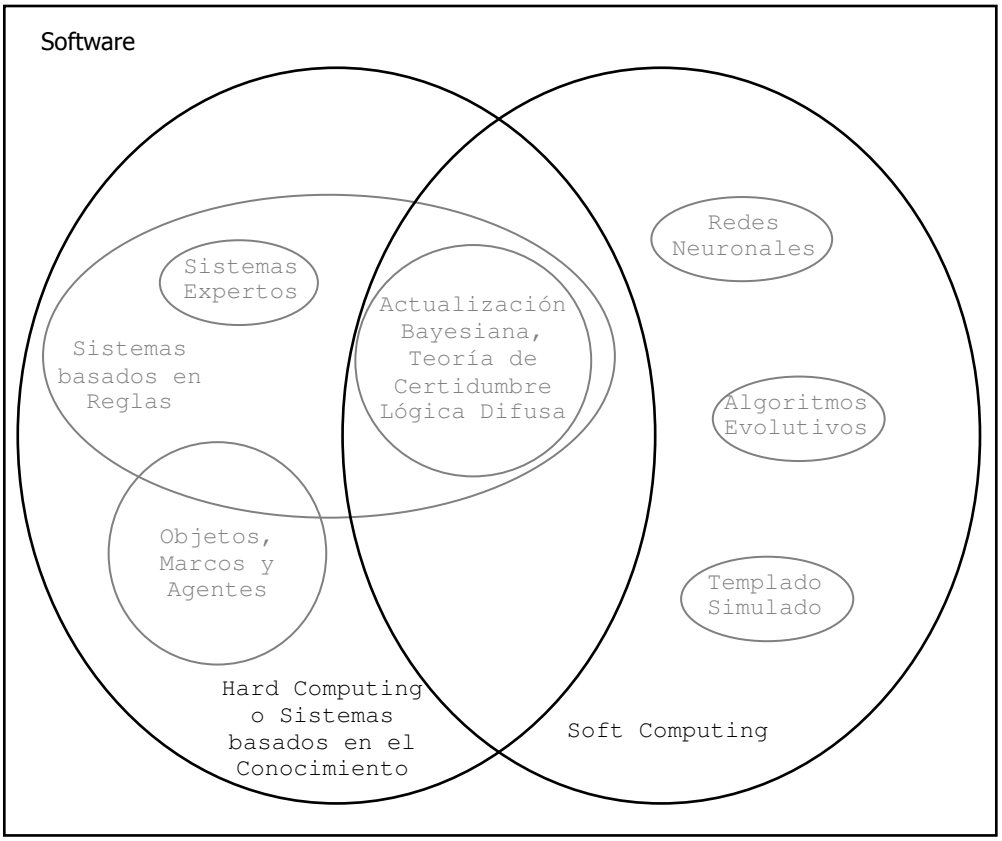
¶ Llamar a estas herramientas "software" es una manera bastante aproximada de caracterizarlas. Sin embargo, algunas de ellas como por ejemplo los *agentes,* pueden involucrar además de un software, un hardware. Véase RUSSELL and NORVIG. 1995. p. 35 –36.

[13] HOPGOOD. p. 4.

[14] RUSSELL and NORVIG. 1995. p. 24.

permiten enfrentar situaciones que involucren algún tipo de racionalidad, lógica o matemática. Hopgood los divide en dos grandes grupos: el **hard computing** o *sistemas basados en el conocimiento* y el **soft computing**. Según él, el primer grupo incluye a aquellos programas computacionales que basan su éxito en la posesión y aprovechamiento de determinados conocimientos, que pueden ser en algún momento dados a conocer al ser humano. Por otro lado en el **soft computing**, los buenos resultados que ha arrojado este software y el entendimiento que se tiene de él, permite concluir que también maneja conocimientos pero que no pueden ser interpretados por el humano. Esta imposibilidad para poder hacer tangible dicho conocimiento hace que estos programas adquieran el nombre de **soft** (suave). En cambio el **hard computing** indica que su conocimiento sí se puede materializar en un código interpretable, en algo **hard** (duro) [†§].

Figura 3. *Categorías de Sistemas Inteligentes, según Hopgood*[15].



[†] Autores como Hopgood le denominan también al **soft computing**, *inteligencia computacional*. Aunque este término es más manejable en la lengua española, para efectos del presente trabajo se optó por no emplearlo pues puede generar confusiones con otros autores.

[§] Más adelante, en …1.2.3. Características de los AGs…, se podrá comprender mejor la idea de qué es poseer un conocimiento intangible o **soft**.

[15] HOPGOOD. 2001. p. 16.

Hoy, el empleo de los sistemas inteligentes se encuentra bien extendido. Los sistemas expertos, son por ejemplo una de las herramientas que más éxito ha reportado en áreas como la ingeniería, el comercio y la medicina. Debido a su importancia, valdría la pena mencionar aquí algunas de sus aplicaciones reales y potenciales[†]:

- Ingeniería: Evaluación de prospecciones con el fin de hallar yacimientos minerales (PROSPECTOR)[16]; Estimación de los parámetros de comportamiento de cuencas hidrográficas a partir de sus características geológicas y morfológicas (HYDRO)[17] Asesoría en la evaluación de estructuras[18]; Asesoría en la construcción pilotes in-situ (DS^2)[19]; Caracterización de residuos radioactivos[20]; Asesoría en el compostaje de residuos sólidos (BESTCOMP)[21], etc.

- Medicina[22]: Evaluación de riesgo y detección de cáncer (CaDet); Asesoría en el manejo de dolores de pecho (ACORN, Thorask); Diagnóstico a partir de desórdenes cutáneos (DermaDex); Monitoreo de posibles reacciones inversas de un paciente hacia una droga (ADE); Diagnóstico de enfermedades cardíacas (HDP), etc.

- Administración, contabilidad y auditoría[23]: Análisis de riesgos; Planeación de auditorías; Planeación internacional de impuestos; Detección de robo de servicios; Búsqueda de fraudes; etc.

Otras de las herramientas con gran éxito son los algoritmos evolutivos. La Figura 4 muestra como fueron apareciendo a través de la historia[§]. Como se puede ver allí, la denominación de algoritmos evolutivos hace referencia a cinco clases de sistemas[25]: *estrategias evolutivas, programación evolutiva, algoritmos genéticos*, *programación genética* y *sistemas clasificadores*[Δ].

[†] Normalmente, cada vez que se diseña un sistema experto, se le suele colocar un nombre propio. Para los ejemplos siguientes, dichos nombres están marcados en paréntesis.

[16] CASTILLO y ÁLVAREZ. 1989. p. 28.

[17] CASTILLO y ÁLVAREZ. 1989. p. 28.

[18] BRANDON. 2000.

[19] FISHER, O'NEIL and CONTRERAS. 1995.

[20] HODGES *et al*. 2001.

[21] JAYAWARDHANA *et al*. 2003.

[22] FEDERHOFER.

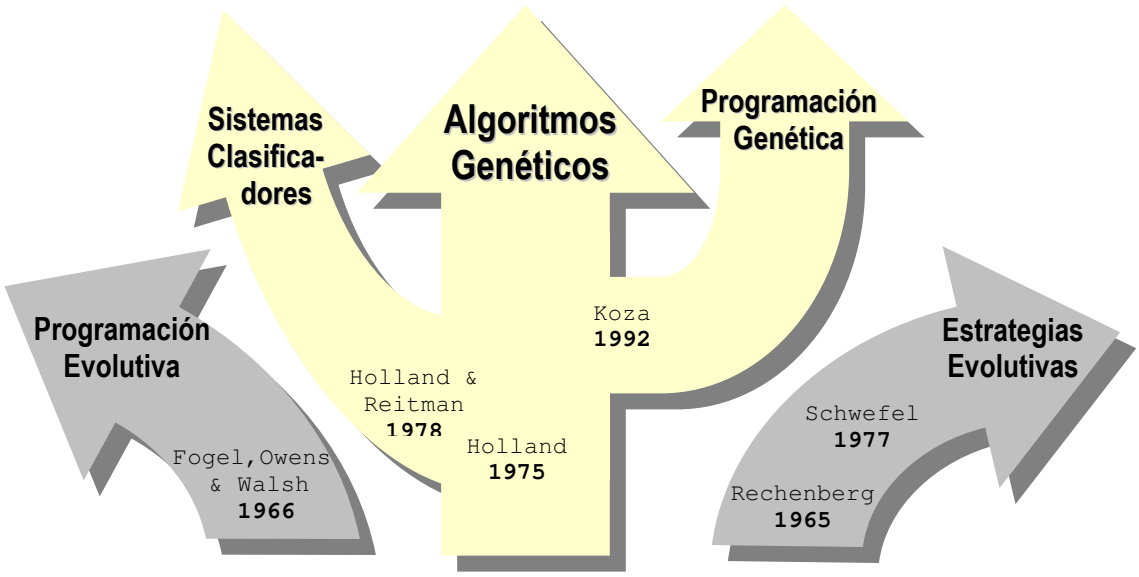
[23] SÁNCHEZ TOMÁS. 1998.

[§] Las fechas indicadas en la figura 4, no representan la verdadera cronología de cómo fueron apareciendo las diferentes metodologías, sino el orden de aparición de las primeras obras que mejor las dieron a conocer. Por ejemplo, la primera publicación en programación evolutiva fue un artículo escrito por Lawrence J. Fogel en 1962, pero hasta 1966 se publicó el primer libro. Asimismo, cuando en 1975 John H. Holland escribió el principal libro de algoritmos genéticos hasta la fecha, llevaba para entonces más de diez años escribiendo artículos al respecto.

[25] DUMITRESCU *et al*. 2000. p. 5.

[Δ] De Jong y Spears reportan que existen aún más tipos de algoritmos evolutivos. Véase SPEARS *et al*. 1993. p. 5.

Figura 4. *Desarrollo de los algoritmos evolutivos.*



1966 FOGEL, L.J., A.J. OWENS & M.J. WALSH. *Artificial Intelligence Through Simulated Evolution*. John Wiley & Sons. Nueva York, EE.UU.

1965 RECHENBERG, I. Cybernetic Solution Path of an Experimental Problem. Trad. B.F. Toms. Traducción No. 1122 de la Royal Aircraft Establishment. Inglaterra.

1975 HOLLAND, J.H. *Adaptation in Natural and Artificial Systems*. Universidad de Michigan. Ann Harbor, EE.UU.

1977 SCHWEFEL, H.P. *Evolutionsstrategie und Numerische Optimierung*. Disertación Doctoral. Universidad Técnica de Berlín. Berlín, Alemania.

1978 HOLLAND, J.H. & J.S. REITMAN. *Cognitive Systems Based on Adaptive Algorithms*. Pattern Directed Inference Systems. Eds. D.A. Waterman & F. Hayes-Roths. Pág. 313-329. Academic Press. Nueva York, EE.UU.

1992 KOZA, J.R. *Genetic Programming*. MIT Press. Cambridge, Massachusetts, EE.UU.

En un principio, se trataba de tres enfoques totalmente diferentes[26]. Por un lado estaban las estrategias evolutivas, las cuales se crearon para resolver el problema específico del diseño óptimo de alas en aeronaves. Se desarrollaron principalmente en Alemania[27]. Por otro lado se creó la programación evolutiva para desarrollar sistemas con la inteligencia para predecir un estado de algún tipo de ambiente[28]. Y la tercera herramienta, los algoritmos genéticos, surgen del interés de su creador John H. Holland por diseñar e implementar sistemas robustos con características de adaptación a ambientes cambiantes

[26] DE JONG and SPEARS. 1993. p. 1.

[27] GOLDBERG. 1989. p. 104, 105.

[28] DUMITRESCU *et al.* 2000. p. 7.

y con incertidumbre[29]. Así, los algoritmos genéticos no se desarrollaron *a priori* para desarrollar alguna clase especial de problemas. Posteriormente, los sistemas clasificadores y la programación genética nacieron como variaciones de los algoritmos genéticos.

A pesar de sus diferencias, todas estas herramientas describen procesos muy similares. Es más, existe actualmente un gran interés por combinarlos de alguna u otra manera para así crear mejores herramientas[30]. Los algoritmos evolutivos pueden definirse de manera general como sistemas que emplean procesos inspirados en la evolución natural para la resolución de problemas de búsqueda[31,32]. Estos problemas incluyen:

- Búsqueda de un número o números que maximicen o minimicen una función (optimización numérica).
- Búsqueda del orden adecuado de unos elementos para mejor satisfacer una condición (optimización combinatoria).
- Búsqueda de unas reglas que permitan comprender patrones (aprendizaje de máquina).
- Búsqueda de estrategias a seguir en juegos (y por extensión en situaciones reales)[33].
- Búsqueda de unas *líneas de código* que generen un programa para resolver un problema (este es principalmente el objetivo de la programación genética).

Los procesos inspirados en la evolución son principalmente los de reproducción y la selección natural. Se ha observado cómo en la naturaleza existen poblaciones de individuos que buscan sobrevivir. Los más aptos que sí lo logran son los que también logran una mayor cantidad de descendencia. De manera análoga, en los algoritmos evolutivos se parte de poblaciones (grupos de posibles soluciones). Aquellos que tengan un mejor desempeño, se les permitirá entonces sobrevivir para que luego se puedan reproducir sexual o asexualmente, generando así hijos (nuevas soluciones candidatas)[†].

Como es de esperar, las aplicaciones reales que se le están dando a los algoritmos evolutivos involucran en muchos casos el empleo algoritmos genéticos. Más ahora cuando se está explorando la combinación de los algoritmos evolutivos entre sí y la combinación con otros sistemas inteligentes[34]. He aquí una breve muestra de todas las aplicaciones que están teniendo los algoritmos evolutivos en el área de la ingeniería[¶]:

---

[29] DE JONG. 1999. p. 1.

[30] DE JONG and SPEARS. 1993. p. 1.

[31] KARR and FREEMAN. 1999. p. 2.

[32] HOPGOOD. 2001. p. 173.

[33] GOLDBERG. 1989. P.

[†] Estos procesos de mutación, reproducción y entrecruzamiento serán mejor explicados en la siguiente sección ...1.2. ¿QUÉ SON?....

[34] A estas combinaciones se les denomina *sistemas híbridos*. Véase HOPGOOD. 2001. Pág. 2.

[¶] Para la siguiente lista, se escogieron ejemplos donde no se empleasen AGs. Las aplicaciones de los AGs se podrán ver con detalle en …1.5. ¿DÓNDE APLICARLOS?....

- Diseño de componentes aerodinámicos[35] (DaimlerChrysler AG) – estrategias evolutivas.
- Optimización de cronogramas de semáforos[36] (Ministerio Holandés de Tráfico) – estrategias evolutivas.
- Optimización de redes viales[37] – estrategias evolutivas.
- Optimización en la programación del mantenimiento y rehabilitaciones de los pavimentos[38] – programación evolutiva.
- Procesamiento de imágenes satelitales[39] – sistemas clasificadores.
- Control de gasoductos[40] – sistemas clasificadores.
- Diseño de redes de comunicaciones[41] – algoritmo evolutivo.

Todo lo mencionado en la presente sección, permite ahora afirmar con firmeza, que los algoritmos genéticos son una de las tantas herramientas de la inteligencia artificial que están teniendo gran aplicación tanto en la ingeniería como en otros campos.

1.2. ¿QUÉ SON?

Definirlos no es fácil ya que actualmente se emplean diferentes estilos de algoritmos genéticos. No obstante, entre todos estos resalta el empleado por Holland en 1975 para explicar sus fundamentos teóricos, el cual es conocido como *algoritmo genético simple*[42] (**simple genetic algorithm**)[†]. A continuación se dará una definición que intenta ser comprehensiva de todas estas formas. Luego se explicará en detalle el algoritmo genético simple, lo cual servirá de base para poder mencionar las características de los AGs en general, el tipo de problemas a los que se pueden enfrentar y las diferentes formas que pueden adquirir.

1.2.1 Definición. Los algoritmos genéticos son una secuencia de pasos que:
- a partir de un conjunto de posibles soluciones a un determinado problema
- los codifica en cadenas de caracteres (llámense *cromosomas*)
- y los manipula iterativamente mediante funciones que simulan los procesos de selección natural, reproducción y mutación encontrados en la naturaleza

[35] NUTECH SOLUTIONS INC. DaimlerChrysler Aerospace AG.

[36] NUTECH SOLUTIONS INC. The Dutch Ministry of Traffic.

[37] SCHWEITZER *et al.* 1998.

[38] NUNOO. 2001.

[39] PALM. 2000.

[40] GOLDBERG. 1989. p. 289.

[41] NUTECH SOLUTIONS INC. A major communication network provider.

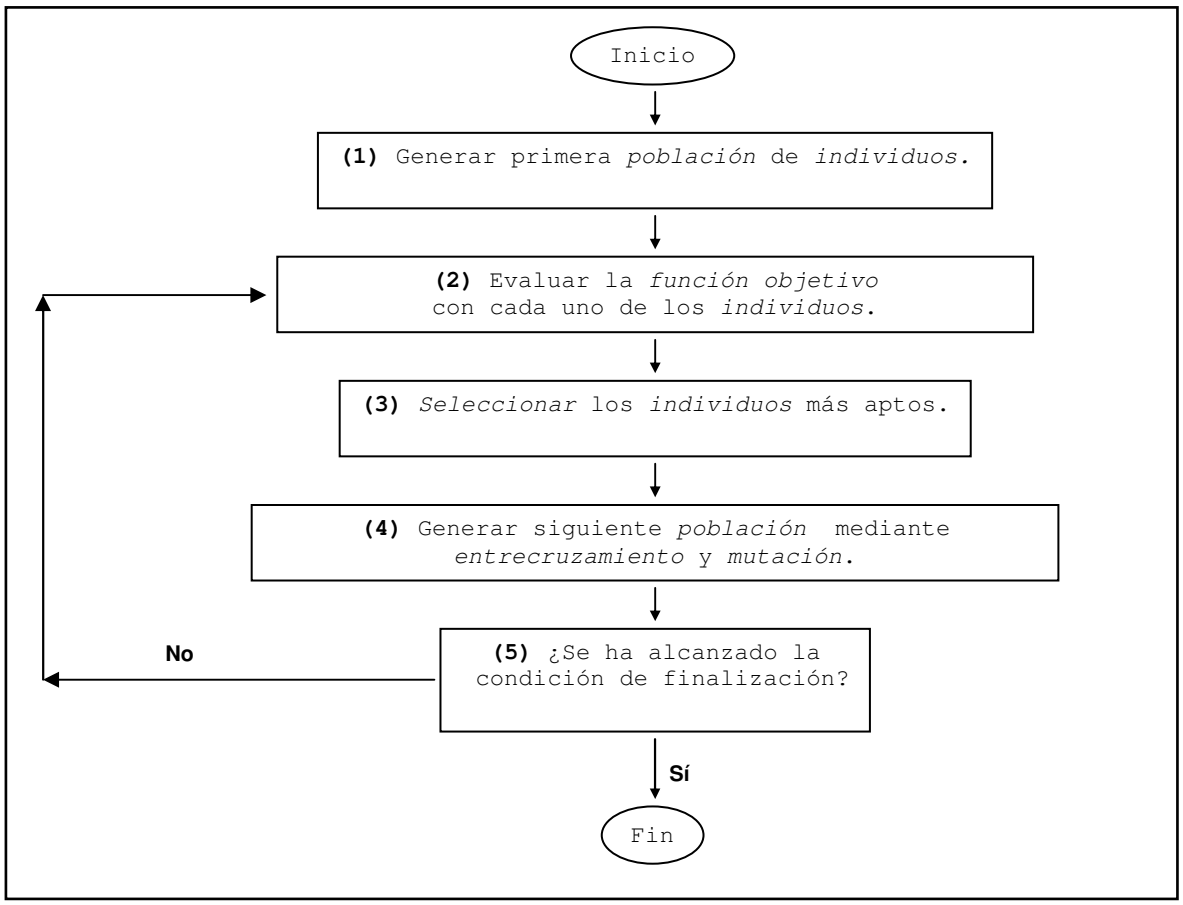
[42] GOLDBERG. 1989. p. 10.

[†] A partir de este momento algunas de las palabras claves, cuando aparezcan por vez primera, tendrán al lado su traducción en inglés. Esto se hace con el fin de facilitar la posterior consulta en otros textos.

- para generar así la solución que mejor satisfaga al problema.

La definición no habla de procesos computacionales. Sin embargo, la necesidad de realizar manipulaciones iterativas (o cálculos tediosos) con las cadenas, implica obligatoriamente programar en algún lenguaje computacional. Ahora bien, vale la pena hacerse las siguientes preguntas: (1) ¿qué tipos de problemas se podrán resolver? (2) ¿Cómo se realiza tal codificación con cromosomas? (3) ¿Cómo serán las funciones que simulan la naturaleza? La siguiente sección las responde para el caso del algoritmo genético simple.

Figura 5. *Diagrama de flujo de un algoritmo genético simple*[43]*.*



1.2.2 El algoritmo genético simple. Denominado también *algoritmo genético estándar* (**canonic genetic algorithm**)[44], pareciera estar diseñado exclusivamente para problemas de optimización matemática[†]. La función a maximizar se denomina *función objetivo*

[43] Este esquema está basado en LIONG, CHAN and SHREERAM. 1995. Fig. 1.

[44] DUMITRESCU *et al.* 2000. p. 28.

[†] En …1.2.4. Problemas a ser solucionados por los AGs… se podrá intuir que el AG simple sirve también para otros tipos de problemas.

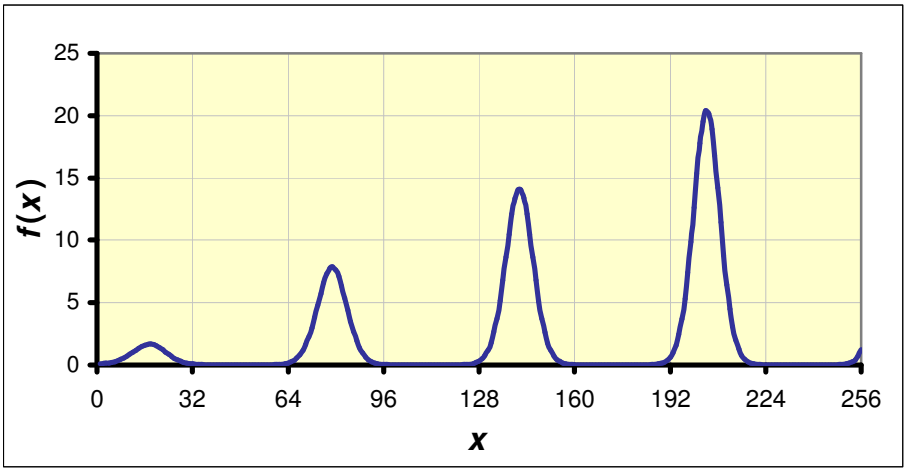
(**objective function**[†]). La secuencia de pasos se describe en la Figura 5. En ella se aprecian vocablos novedosos traídos de la biología tales como, *población* (**population**), *individuos* (**individuals**), *aptitud* (**fitness**), *entrecruzamiento* (**crossover**) y *mutación* (**mutation**). Estos términos y los pasos se explicarán a continuación mediante el planteamiento de un ejemplo.

Supóngase que se requiere resolver el siguiente problema:

Encuentre el valor de x *tal que se maximice la función:* $f(x) = \frac{1}{10} x^{sen\frac{x}{10}}$

Dicha función se puede observar en la Figura 6. Se aprecia que se trata de una *función multimodal*, es decir, tiene varios picos falsos[‡] lo cual podría engañar a cualquier método de resolución que se emplee (en este caso, al AG).

Figura 6. *Función* $f(x) = \frac{1}{10} x^{sen\frac{x}{10}}$



Antes de aplicar el algoritmo genético simple, se debe primero asegurar de haberse respondido a las siguientes cuatro cuestiones:

1. ¿Cuál es la función objetivo?

   $$f(x) = \frac{1}{10} x^{sen\frac{x}{10}}$$

2. ¿Qué se requiere hacer? ¿maximizarla o minimizarla?
   Maximizarla.

3. ¿Cuál es el espacio de búsqueda?
   Un número entre 0 y 255 (podría ser en otro ejemplo, una pareja de números en donde el primero esté entre –5 y 20, y el segundo entre –10 y 5000).

[†] Originalmente Holland en su obra de 1975, no mencionaba el término **objective function** sino **payoff**.

[‡] Esta definición de función multimodal, se tomó de GOLDBERG. 1989. p. 9.

4. ¿Cuál es la precisión requerida de la solución?
   Para este ejemplo, sólo se trabajará con números enteros (en otro ejemplo, podría ser 0,5 o 0,2235).

Ahora se procederá a seguir los pasos indicados en la Figura 5.

Primer Paso. Plantear al azar una población de individuos. Sean por ejemplo, los siguientes:

Solución (*Fenotipo*)	Cromosoma (*Genotipo*)
185	1 0 1 1 1 0 0 1
214	1 1 0 1 0 1 1 0
236	1 1 1 0 1 1 0 0
221	1 1 0 1 1 1 0 1
23	0 0 0 1 0 1 1 1
14	0 0 0 0 1 1 1 0

Cada uno de los números propuestos es un *individuo*. El grupo de individuos se denomina *población* (**population**). Cada individuo consta en realidad de dos partes: el número en sí o *fenotipo* (**phenotype**), y su cromosoma (**chromosome**) o *genotipo* (**genotype**)[Δ]. En un algoritmo genético simple, un cromosoma es la misma solución pero escrito como un número binario. Cada cuadro con su correspondiente 0 o 1, constituye un gen (**gene**) dentro del cromosoma.

Segundo Paso. Evaluar la función objetivo con cada uno de los individuos. Se obtiene entonces lo siguiente:

#	**Cromosoma**	x	$f(x)$
1	10111001	185	0,0167
2	11000100	196	3,6579
3	11101100	236	0,0004
4	11010111	215	1,2591
5	00010111	23	1,0362
6	00001110	14	1,3473

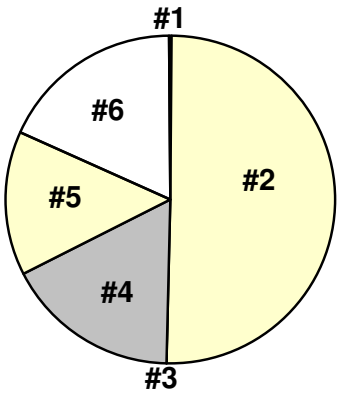
Tercer Paso. Seleccionar los individuos más aptos. De los valores obtenidos, se observa que el segundo y el último son los que mejor cumplen con la tarea que se les ha propuesto: maximizar a $f(x)$. ¿Pero se deberá simplemente ignorar los demás individuos?

El algoritmo genético simple recurre aquí al método de *selección por ruleta* (**roulette wheel selection**). A cada individuo se le asigna un espacio dentro de una ruleta de

[Δ] Tal como en la naturaleza el fenotipo es por ejemplo, el color de ojos de una persona, y el genotipo correspondiente serían los genes que le permiten tener dicha característica.

acuerdo a su desempeño o *aptitud* (**fitness**). Para el ejemplo, las probabilidades de selección (o pesos en la ruleta) serían las siguientes:

#	Cromosoma	x	$f(x)$	Peso en la Ruleta $f(x) / \Sigma f(x)$
1	10111001	14	0,0167	0,0023
2	11000100	215	3,6579	0,4999
3	11101100	196	0,0004	0,0001
4	11010111	196	1,2591	0,1721
5	00010111	23	1,0362	0,1416
6	00001110	196	1,3473	0,1841
		$\Sigma f(x)$	7,3177	



Se hace girar la "ruleta" y supóngase que se obtienen los siguientes resultados.

- En el **primer** giro de la ruleta, se seleccionó al cromosoma **#6**.
- En el **segundo** giro de la ruleta, se seleccionó al cromosoma **#4**.
- En el **tercero** giro de la ruleta, se seleccionó al cromosoma **#2**.
- En el **cuarto** giro de la ruleta, se seleccionó al cromosoma **#2**.
- En el **cinco** giro de la ruleta, se seleccionó al cromosoma **#5**.
- En el **seis** giro de la ruleta, se seleccionó al cromosoma **#2**.

Por lo tanto, la selección queda conformada de la siguiente manera:

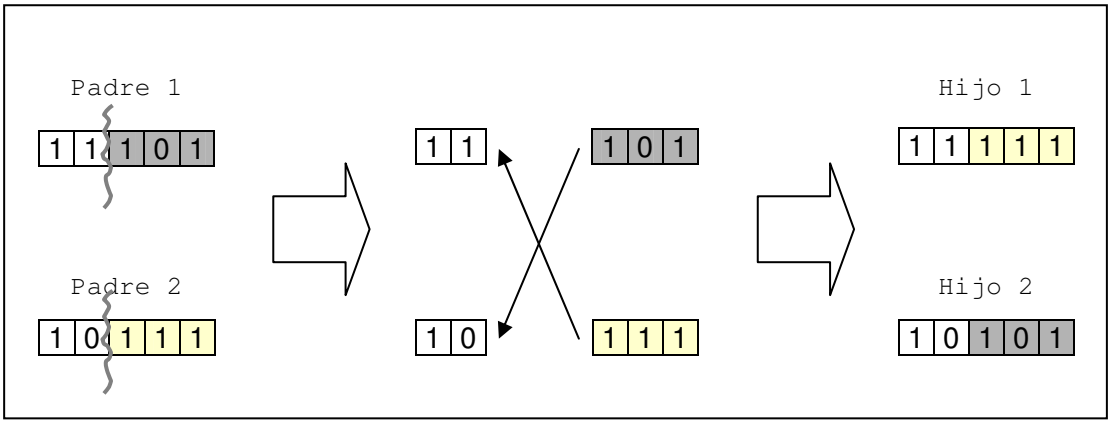
	Número (*Fenotipo*)	Cromosoma (*Genotipo*)
#6	14	0 0 0 0 1 1 1 0
#4	215	1 1 0 1 0 1 1 1
#2	196	1 1 0 0 0 1 0 0
#2	196	1 1 0 0 0 1 0 0
#5	23	0 0 0 1 0 1 1 1
#2	196	1 1 0 0 0 1 0 0

Como se verá a continuación, los cromosomas #6, #4, #2 y #5 serán los padres de una nueva generación de individuos.

*Cuarto Paso: Generar una nueva población mediante entrecruzamiento (**crossover**) y mutaciones (**mutation**)*. Se toman entonces los padres, se asignan parejas y se produce el *entrecruzamiento*[†]. La Figura 7 ilustra cómo se realiza esta operación:

[†] En el presente trabajo se empleará el término "entrecruzamiento". Existen documentos en español que emplean términos como "recombinación" y "apareamiento".

***Figura 7**. Operador de Entrecruzamiento.*



En el ejemplo, al aplicar el operador de entrecruzamiento se obtienen los siguientes resultados:

	Padres		Hijos	
	Número	Cromosoma	Número	Cromosoma
#6	14	0 0 0 0 1 1 1 0	15	0 0 0 0 1 1 1 1
#4	215	1 1 0 1 0 1 1 1	214	1 1 0 1 0 1 1 0
#2	196	1 1 0 0 0 1 0 0	196	1 1 0 0 0 1 0 0
#2	196	1 1 0 0 0 1 0 0	196	1 1 0 0 0 1 0 0
#5	23	0 0 0 1 0 1 1 1	20	0 0 0 1 0 1 0 0
#2	196	1 1 0 0 0 1 0 0	199	1 1 0 0 0 1 1 1

Obsérvese que en cada pareja se escoge un punto de ruptura (una posición al azar) a partir de la cual se hace el entrecruzamiento. Mediante este operador, se da luz a una nueva generación. Es importante aclarar lo siguiente. El algoritmo genético simple, tal como lo planteó Holland, establece un parámetro p_c para todo el AG, el cual indica la probabilidad de que una pareja de padres sí realice entrecruzamiento. En el ejemplo anterior, se está trabajando con un p_c igual a 1, pues todos los padres realizaron entrecruzamiento. Si se hubiese trabajado con un p_c = 0,33, se estimaría que dos padres no sufriesen esta operación.

Adicionalmente se aplica el operador *mutación*, que consiste en recorrer cada gen de los hijos y de acuerdo a una probabilidad (de pequeño valor) se cambia el carácter. Dicha probabilidad p_m se establece también inicialmente como un parámetro del AG. En el ejemplo, supóngase que se decidió emplear una probabilidad de mutación p_m = 0,03. Esto "podría" hacer cambiar la generación de la siguiente manera:

Antes de la Mutación		Después de la Mutación (Nueva Población Definitiva)	
Número (*Fenotipo*)	Cromosoma (*Genotipo*)	Cromosoma (*Genotipo*)	Número (*Fenotipo*)
15	0 0 0 0 1 1 1 1	0 0 0 0 1 1 1 1	15
214	1 1 0 1 0 1 1 0	1 1 0 1 0 1 1 0	214
196	1 1 0 0 0 1 0 0	1 1 1 0 0 1 0 0	228
196	1 1 0 0 0 1 0 0	1 1 0 0 0 1 0 0	196
20	0 0 0 1 0 1 0 0	0 0 0 1 0 0 0 0	16
199	1 1 0 0 0 1 1 1	1 1 0 0 0 1 1 1	199

Se observa ahora que los individuos de la nueva generación, al evaluarse en la función objetivo presentan los siguientes nuevos valores:

#	**Cromosoma**	x	$f(x)$
1	00001111	15	1,4899
2	11010110	214	1,9896
3	11100100	228	0,0020
4	11000100	196	3,6579
5	00010000	19	1,6221
6	11000111	199	9,8761

Comparando esta población con la población inicial, se observa que hubo un mejoramiento en la aptitud de los individuos. Se pasó de tener un x = 196 tal que $f(x)$ = 3,6579, a un x = 199 tal que $f(x)$ = 9,8761. Se está convergiendo así hacia la solución del problema.

Quinto Paso: ¿Se ha alcanzado la condición de finalización? Generalmente, de acuerdo a la complejidad del problema se establece previamente el número de ciclos que se desea recorrer, de manera que la condición de finalización ocurre cuando se cumpla con dicho número de ciclos.

En el ejemplo anterior, se podría haber establecido previamente que se realizarían cinco ciclos, así que lo que seguiría a realizar sería repetir cuatro veces más el segundo, el tercero y el cuarto paso tal como se indica en la Figura 5. Finalmente, se revisa cuál ha sido el individuo de mejor desempeño en la última generación. Otra condición de finalización puede consistir en repetir ciclos hasta observar que no haya ninguna mejoría en el desempeño del mejor cromosoma.

Los anteriores pasos se pueden organizar en tablas para poder hacerlo más claro. En la Figura 8, se muestra un ejemplo de cómo un algoritmo genético se encargó de resolver el problema anterior.

Figura 8. *Resolución del problema "maximizar la función:* $f(x) = \frac{1}{10} x^{sen\frac{x}{10}}$ *donde x ∈ [0, 255]" mediante un AG simple.*

Peso en la ruleta expresado en %

Posición en donde ocurrió la ruptura para el entrecruzamiento

Una vez procreado el nuevo cromosoma, número de sus genes que sufrieron mutación

GENERACIÓN 0							GENERACIÓN 1							GENERACIÓN 2						
#	Cromosoma	x	F(x)	Peso%	Padres	XP	NM	Cromosoma	x	F(x)	Peso%	Padres	XP	NM	Cromosoma	x	F(x)	Peso%
1	11111011	251	0.084	9.30	3	3	1	10011000	152	1.151	11.38	2	2	0	10001000	136	6.81	14.84
2	00100011	35	0.029	3.20	3	3	2	10001000	136	6.809	67.30	2	2	0	10001000	136	6.81	14.84
3	10011010	154	0.463	51.31	3	3	6	10011010	154	0.4603	4.55	2	2	0	10001000	136	6.81	14.84
4	01011101	93	0.176	19.59	3	3	6	10011010	154	0.4603	4.55	2	2	2	01001000	72	2.98	6.49
5	10100101	165	0.003	0.29	3	3	5	10011110	158	0.0628	0.62	2	1	1	10001100	140	13.4	29.13
6	01100000	96	0.045	5.03	3	3	5	00011010	26	0.5363	5.30	1	2	0	10011000	152	1.15	2.51
7	10100011	163	0.006	0.65	4	3	1	01011101	93	0.1758	1.74	2	1	0	10001000	136	6.81	14.84
8	11011100	220	0.095	10.63	3	4	1	10011010	154	0.4603	4.55	1	2	0	10011000	152	1.15	2.51

Población inicial

GENERACIÓN 3							GENERACIÓN 4							GENERACIÓN 5						
Padres	XP	NM	Cromosoma	x	F(x)	Peso%	Padres	XP	NM	Cromosoma	x	F(x)	Peso%	Padres	XP	NM	Cromosoma	x	F(x)	
2	2	2	0	10001000	136	6.81	10.04	5	8	4	0	10001100	140	13.4	15.20	4	3	3	0	10001101	141	14.1
2	2	2	0	10001000	136	6.81	10.04	8	5	4	1	10001000	136	6.81	7.74	3	4	3	0	10001100	140	13.4
5	7	6	0	10001000	136	6.81	10.04	4	4	7	1	10001101	141	14.2	15.98	4	1	5	2	10001001	137	8.63
7	5	6	0	10001100	140	13.4	19.72	4	4	7	0	10001100	140	13.4	15.20	1	4	5	1	11001100	204	20.4
8	5	4	0	10001100	140	13.4	19.72	8	5	8	0	10001100	140	13.4	15.20	5	6	7	1	10001000	136	6.81
5	8	4	1	10011010	154	0.46	0.68	5	8	8	0	10001100	140	13.4	15.20	6	5	7	0	10001100	140	13.4
7	5	8	0	10001000	136	6.81	10.04	5	1	2	0	10001000	136	6.81	7.74	7	3	1	0	10001000	136	6.81
5	7	8	0	10001100	140	13.4	19.72	1	5	2	1	10001000	136	6.81	7.74	3	7	1	0	10001101	141	14.2

Solución

Esto se hizo mediante un programa escrito en Matlab, que fue preliminar al que se desarrolló para el caso estudio que se trabajó en el presente proyecto[¶].

Para dicho ejemplo se trabajó con una probabilidad de entrecruzamiento p_c = 0,9 y una probabilidad de mutación p_m = 0,05. Ya que que p_c no fue igual a 1,0, se puede observar que los cromosomas 6 y 7 de la generación 1 fueron idénticos a sus padres, pues no hubo entrecruzamiento. Cuando no actúa el operador de entrecruzamiento se dice que simplemente actuó otro operador llamado *reproducción*[45].

A raíz del surgimiento de diferentes tipos de problemas y de la curiosidad de los estudiosos en el tema, se ha experimentado con muchas alternativas a la propuesta de Holland. Así, se ha ensayado con otras formas de entrecruzamiento, otras formas de selección de los mejores individuos, otras codificaciones. Esto ha hecho, que no sólo se haya extendido sus aplicaciones, sino que hoy se cuente con un mayor conocimiento de las ventajas y desventajas de hacer dichos cambios. Estas variaciones se explicarán en detalle en …1.2.5. Elementos de diseño en un AG….

1.2.3 Características de los AGs. Habiendo ya observado cómo funciona un algoritmo genético simple, se podrá ahora comprender algunas de las características más relevantes que poseen los AGs en general. Información interesante se obtiene cuando se les compara con otros métodos de optimización matemática. Golberg[46] cita las siguientes cuatro propiedades:

- *Los AGs funcionan simplemente gracias a los valores que va arrojando la función objetivo. Aquí no se requiere ningún otro tipo de información (como valores de derivadas o propiedades particulares de la función).*
  Esta característica abre las puertas para que puedan ser aplicados a un sin número de funciones. No importa qué función objetivo se esté empleando, lo único que importa es que arroje valores. Métodos tradicionales de optimización basados en el cálculo requieren que la función objetivo sea derivable y/o contínua. El *método símplex* requiere que la función objetivo sea lineal. Otros métodos *enumerativos* como la *programación dinámica* requieren que el espacio de búsqueda (el número de posibles soluciones) no sea muy grande ni mucho menos infinito.

- *Los AGs realizan la búsqueda de la solución óptima a partir de una población de puntos (es decir, de una población de soluciones o individuos). No lo hacen a partir de un solo punto.*
  Esta característica es importante pues ayuda a que el AG no se estanque en máximos o mínimos locales (como los picos falsos que se observan en la función de la Figura 6). El método convencional basado en el cálculo denominado *ascenso*

[¶] Se invita aquí al lector, a observar detenidamente la Figura 8, pues éste es el único ejemplo que se expone de manera completa y gráfica sobre el desarrollo de un AG simple. Además, su comprensión facilitará la lectura de la sección …1.4. ¿FUNCIONAN?...

[45] FALKENAUER. 1998. p. 40.

[46] GOLDBERG. 1989. p. 7.

de máxima pendiente (**hill climbing**) corre ese riesgo. Éste comienza por ensayar con un punto. Luego busca a su alrededor en donde haya un mayor gradiente y así, escoge su siguiente punto de búsqueda. Si la función tiene varios picos locales entonces corren el riesgo de no detectar el pico global[47].

- *Los AGs involucran pasos probabilísticos, no determinísticos.*
  La selección por ruleta, el punto de ruptura para el entrecruzamiento, la probabilidad p_c de entrecruzamiento y probabilidad p_m de mutación, todos estos pasos involucran aleatoriedad. Aunque esto haría sospechar que los AGs no sean más que una simple búsqueda al azar y sin dirección, en realidad no son así. Los AGs basan su búsqueda en la selección de los más aptos.

- *Los AGs manipulan las soluciones, pero de una manera indirecta.*
  Por eso se dice que los AGs son "ciegos", pues independientemente del problema que se les proponga, ellos simplemente se limitan a manipular cadenas de caracteres[48]. Esta característica junto con las tres anteriores, hacen de los algoritmos genéticos una herramienta *robusta,* es decir, de gran aplicación[§].

Vale la pena ahora mencionar por qué se considera que los algoritmos genéticos son un tipo de **soft computing**. Según lo dicho en …1.1. ¿QUIÉNES SON?..., el **soft computing** maneja unos conocimientos intangibles. Pues bien, antes de analizar detenidamente el funcionamiento de los AGs en …1.4. ¿FUNCIONAN?..., por el momento bastará con dar una explicación más intuitiva. Se podría considerar que cada cromosoma posee una o varias ideas sobre lo que puede ser la solución óptima. Las buenas ideas estarán contenidas en los mejores cromosomas y así, tendrán la posibilidad de ser seleccionadas para luego combinarse y formar mejores ideas o mejores soluciones. Es como un coctel en donde las personas con buenas ideas (los expertos) se ponen a charlar y construyen mejores ideas[†]. Las ideas que el AG maneja, constituyen un conocimiento abstracto o intangible que no puede ser interpretado como frases lógicas o reglas[Δ].

Ahora se pasará ahora a conocer más en detalle los individuos y sus cromosomas. La Figura 9 resume parte de la terminología más importante.

El individuo está compuesto por un genotipo (el cromosoma) y un fenotipo (la solución). En la naturaleza, todo organismo posee *unos* cromosomas que lo hacen único. En los AGs todo individuo posee *un* cromosoma que lo hace único, que lo diferencia de los demás. Un cromosoma es una cadena de genes. En la naturaleza, un gen está compuesto por un grupo de subunidades denominados nucleótidos. "Cada gen incluye

---

[47] GOLDBERG. 1989. p. 3, 4.

[48] GOLDBERG. 1999. p. 9.

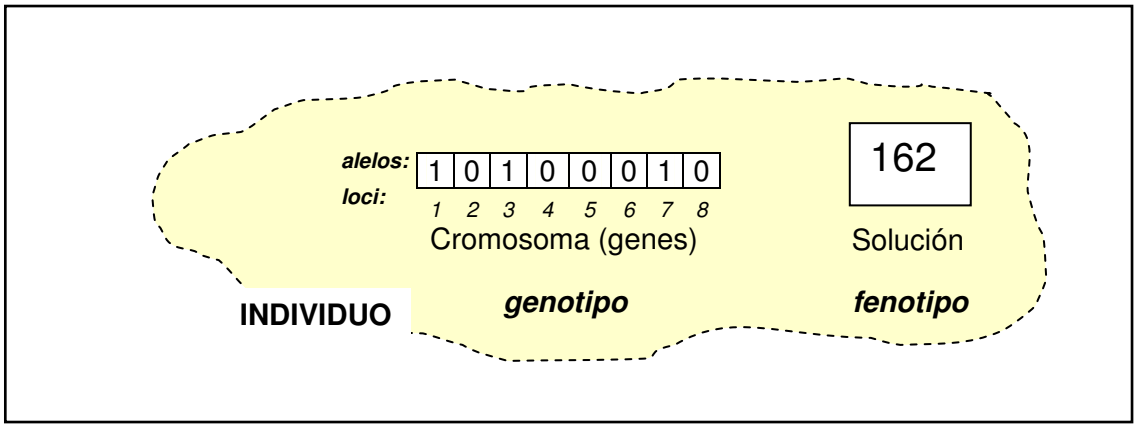
[§] Al inicio del capítulo …1.4. ¿FUNCIONAN?... se mostrarán todas las formas de funciones para las cuales se pueden aplicar los AGs.

[†] Estas analogías con las ideas y con el coctel, fue tomado de GOLDBERG. 1999. p. 14.

[Δ] El autor considera que las ideas toman la forma de *esquemas.* De estos se hablará en …1.4.1. Esquemas….

muchos nucleótidos"[49]. En los AGs, un gen es mucho más sencillo, está compuesto por un carácter, por un valor. En un gen se distinguen dos elementos: su valor o *alelo* (**allele**) y su posición o *locus* (plural, *loci*). Usualmente, los caracteres o alelos más empleados son ceros y unos; y usualmente el cromosoma suele ser la misma solución pero escrita en notación binaria. En …1.2.5.1. La codificación… se hablará sobre otras formas de codificación.

Figura 9. *El individuo en un algoritmo genético.*



1.2.4 Problemas a ser solucionados por los AGs. Gran parte de la investigación en AGs se ha dedicado a la optimización de funciones[50]. Tal como se vio en la explicación del algoritmo genético simple, esto consiste en la búsqueda de un número o números que maximicen o minimicen una función. En otros casos lo importante no es encontrar los números más óptimos sino el orden apropiado de números (u otro tipo de elementos) que mejor satisfisfagan una función (optimización combinatoria). Aquí entra por ejemplo el problema de cómo distribuir apropiadamente unos objetos para así minimizar los espacios empleados (**bin packing**).

Sin importar qué tan creativo sea un problema, si se desea que sea resuelto mediante AGs, éste deberá contener una función objetivo que indique cuantitativamente qué tan óptima es cada una de las soluciones que se van encontrando. Así, dentro de este esquema (búsqueda de soluciones que satisfagan una función objetivo) las aplicaciones se extienden a otros problemas, tales como:

- Optimización multi-objetivo[51], en donde se debe buscar la solución que mejor satisfaga varias funciones objetivo.

- Problemas en donde lo más importante no es encontrar la solución óptima sino un conjunto de soluciones óptimas. Esto sirve como soporte para toma de decisiones¶.

[49] MICROSOFT ENCARTA 2000. *Gen.*

[50] BEASLEY, BULL and MARTIN. 1993. p. 13.

[51] GOLDBERG. 1989. p. 197.

- Problemas en donde la función objetivo no es una expresión matemática sino que genera un resultado a partir de complejos procesos computacionales, como por ejemplo, evaluación de algún diseño o identificación óptima de imágenes.

- Problemas en donde la función objetivo no es una expresión matemática convencional, sino que está dada por lo que diga una persona[†] o por los movimientos que adopte una persona[‡].

- Desde un punto más biológico, simular procesos evolutivos de la naturaleza[‡] mediante algoritmos genéticos.

1.2.5 Elementos de diseño en un AG. A continuación se definirán ciertos elementos importantes dentro del funcionamiento de un algoritmo genético. Son estos elementos, los que hacen que un algoritmo genético varíe de un problema a otro. Estas explicaciones permitirán comprender mejor lo realizado para el caso estudio del presente proyecto.

2.1.5.1 *La codificación*. Determinar cómo un cromosoma va a representar a una solución es el primer elemento relevante en el diseño de un algoritmo genético. No todos los cromosomas tienen que ser representaciones binarias de un número real. Sin embargo, esta forma de codificar (la misma que se empleó en …1.2.2. El algoritmo genético simple…) servirá de base para comprender como se diseña una codificación.

Convertir un número en notación binaria, en términos matemáticos, se puede expresar de la siguiente manera.

Cadena		**Número *Positivo Entero***
σ	$\rightarrow$	$x(\sigma)$
$a_1,a_2,\ldots,a_L$	$\rightarrow$	$a_1 \cdot 2^{L-1} + a_2 \cdot 2^{L-2} + \cdots + a_{L-1} \cdot 2^1 + a_L \cdot 2^0$

en donde,

a_i puede ser igual a 0 o 1.

L es la longitud de la cadena.

$\sigma_{min} =$ 0, 0, 0,…,0

$x_{min}(\sigma_{min}) = 0 \cdot 2^{L-1} + 0 \cdot 2^{L-2} + \cdots + 0 \cdot 2^1 + 0 \cdot 2^0 = 0$

$\sigma_{max} =$ 1, 1, 1,…, 1

$x_{max}(\sigma_{max}) = 1 \cdot 2^{L-1} + 1 \cdot 2^{L-2} + \cdots + 1 \cdot 2^1 + 1 \cdot 2^0 = 2^L - 1.$

[¶] De estos problemas se hablará más adelante, cuando se haga referencia a las aplicaciones dentro de la hidroinformática, en …1.5. ¿DÓNDE APLICARLOS?...

[†] En BEASLEY, BULL and MARTIN. 1993. p. 13, se cita un ejemplo no convencional de identificación de caras de criminales. En él, es una persona (el testigo) quien tiene que evaluar que tan parecida es la cara dibujada por el AG con la del criminal real.

[‡] Varios problemas relacionados con estrategias de juego encajan dentro de esta situación. Un ejemplo es el juego del dilema del prisionero citado en Goldberg, 1989. p. 149.

[‡] Rosenberg en 1967 y Weinberg en 1970 comenzaron a emplear los AGs para estos fines. Ver GOLDBERG. 1989. p. 93-95, 98-99.

Una primera alternativa que surge es emplear una codificación similar pero empleando tres caracteres (0, 1, 2) o k caracteres (0, 1, 2, ..., k-1). En dicho caso, matemáticamente se expresaría de la siguiente manera:

Cadena		**Número *Positivo Entero***
σ	$\rightarrow$	$x(\sigma)$
$a_1,a_2,\ldots,a_L$	$\rightarrow$	$a_1 \cdot k^{L-1} + a_2 \cdot k^{L-2} + \cdots + a_{L-1} \cdot k^1 + a_L \cdot k^0$

en donde,

k indica el número de caracteres que se están empleando
a_i es un valor entre 0 y k-1.
L es la longitud de la cadena.

$\sigma_{min} =$ 0, 0, 0,…,0
$x_{min}(\sigma_{min}) = 0 \cdot k^{L-1} + 0 \cdot k^{L-2} + \cdots + 0 \cdot k^1 + 0 \cdot k^0 = 0$
$\sigma_{max} =$ k-1, k-1, k-1,…, k-1
$x_{max}(\sigma_{max}) = (k\text{-}1) \cdot k^{L-1} + (k\text{-}1) \cdot k^{L-2} + \cdots + (k\text{-}1) \cdot k^1 + (k\text{-}1) \cdot k^0 = 2^L\text{-}1.$

He aquí algunos ejemplos con diferentes valores de k y de L.

k	L	σ	$x(\sigma)$	σ_{max}	$x_{max}(\sigma)$
2	3	1,1,0	$1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 06$	1,1,1	$2^3 - 1 = 07$
2	4	1,1,0,1	$1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 13$	1,1,1,1	$2^4 - 1 = 15$
2	5	1,0,0,0,1	$1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 17$	1,1,1,1,1	$2^5 - 1 = 31$
3	3	2,1,0	$2 \cdot 3^2 + 1 \cdot 3^1 + 0 \cdot 3^0 = 21$	2,2,2	$3^3 - 1 = 26$
4	3	3,0,2	$3 \cdot 4^2 + 0 \cdot 4^1 + 2 \cdot 4^0 = 50$	3,3,3	$4^3 - 1 = 63$

Ahora, muchos problemas requieren soluciones compuestas de números reales (sobretodo en los de optimización matemática). Los ejemplos mostrados hasta el momento sólo permiten representar números enteros y además, positivos. La siguiente forma de codificar variables reales es lo que Goldberg denomina *codificación de punto fijo* (**fixed-point coding**)[52]. Aunque Goldberg lo menciona brevemente en su obra, a continuación se hará una más amplia ilustración del tema†.

Mientras la codificación a números enteros se hacía mediante una función, en la codificación de números reales, simplemente se le aplica otra función a la variable entera. Esto en términos matemáticos se expresa de la siguiente forma:

Cadena		**Número $\in$ Z^+**		**Número $\in$ R**
σ	$\rightarrow$	$x(\sigma)$	$\rightarrow$	$w(x) = w_{\min} + \frac{w_{\max} - w_{\min}}{x_{\max} - x_{\min}} \cdot (x - x_{\min})$

Al aplicar la función w, simplemente se está haciendo un reescalamiento. Las siguientes figuras (10, 11, 12, 13) pretenden explicar gráficamente, cómo es que cualquier número entero positivo, puede ser reescalado en un número real.

[52] GOLDBERG. 1989. p. 82.

† Es conveniente ilustrar bien este tema, pues así se explica cómo se realizó el aplicativo computacional.

Figura 10. *Reescalamiento de un rango de números enteros* $[x_{min}, x_{max}]$ *positivos a un rango de números positivos* $[w_{min}, w_{max}]$.

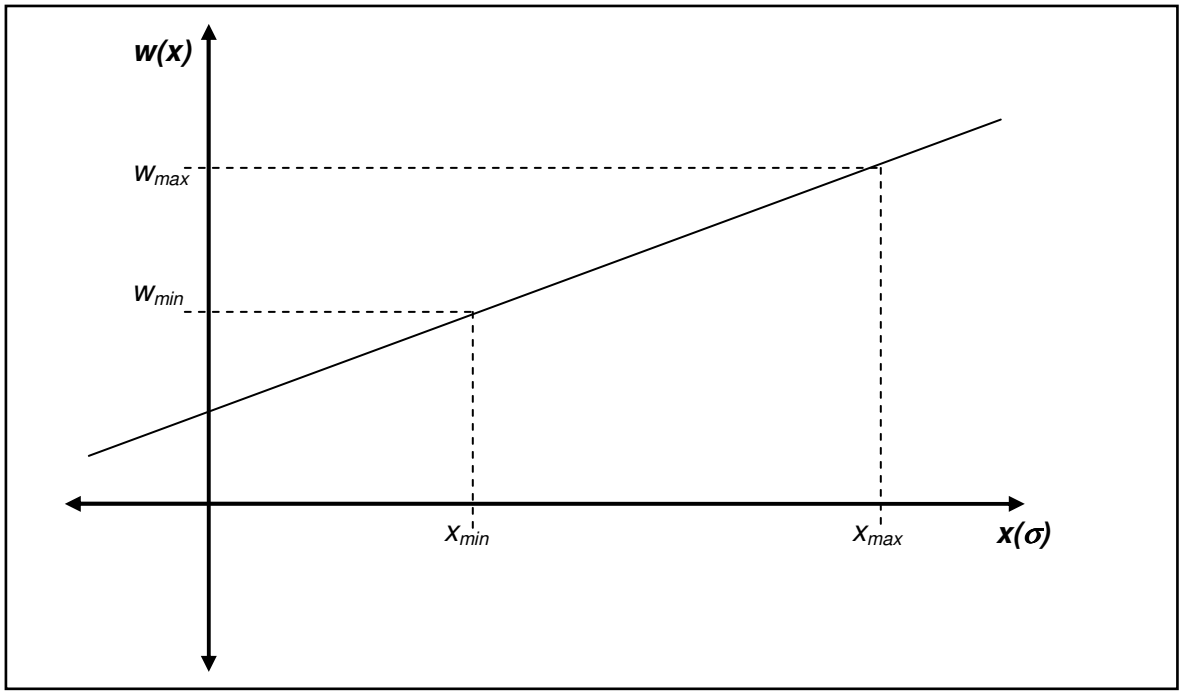


Figura 11. *Reescalamiento de un rango de números enteros* $[x_{min}, x_{max}]$ *positivos a un rango de números positivos y negativos* $[w_{min}, w_{max}]$.

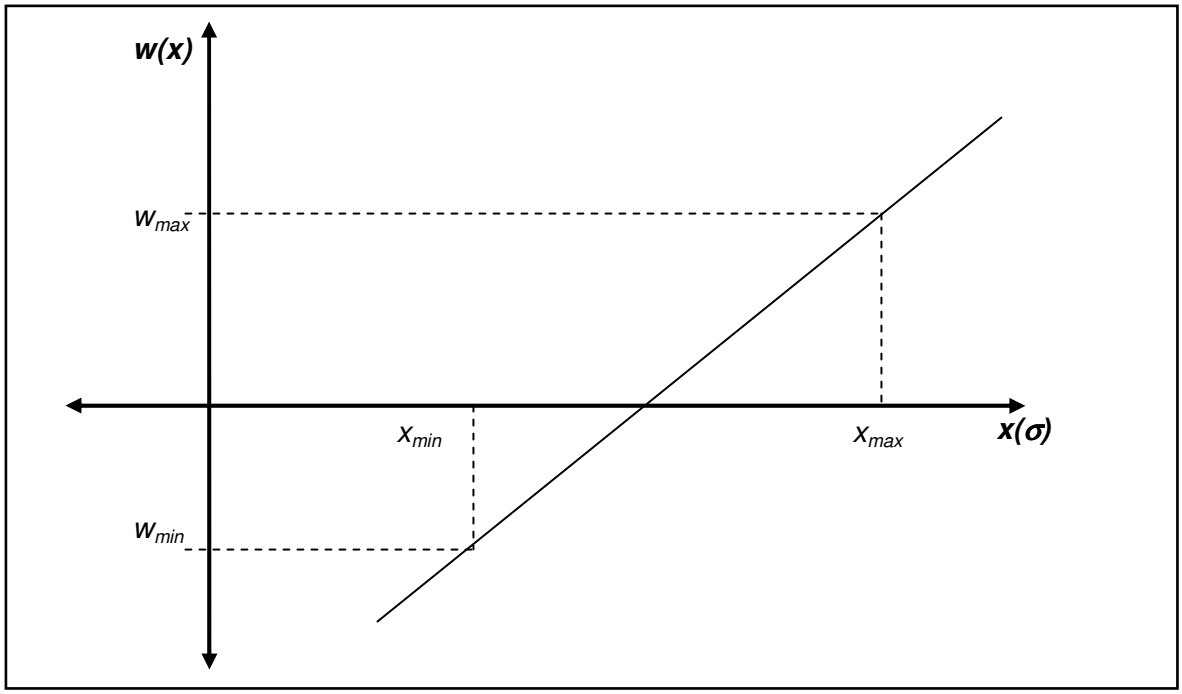


Figura 12. *Reescalamiento de un rango de números enteros* $[x_{min}, x_{max}]$ *positivos a un rango de números no enteros positivos [2, 3].*

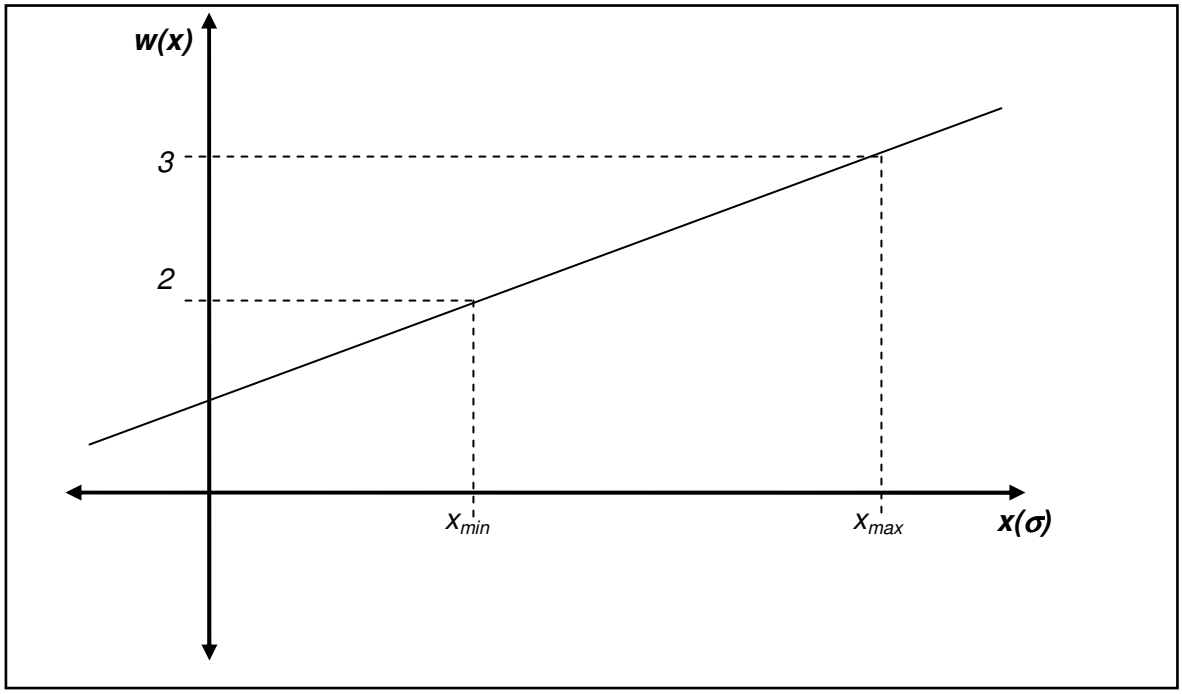
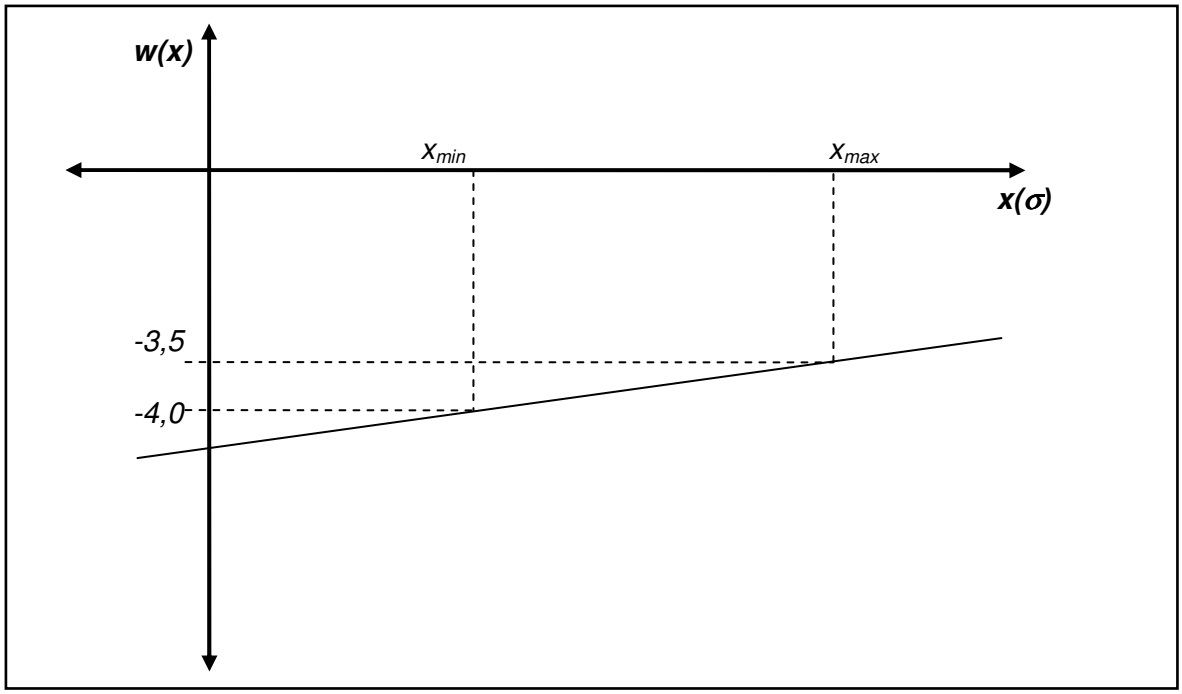


Figura 13. *Reescalamiento de un rango de números enteros* $[x_{min}, x_{max}]$ *positivos a un rango de números no enteros positivos y negativos [-3,5, -4,0].*



Téngase en cuenta que el x_{min} y x_{max} están dados por la función $x(\sigma)$, es decir, dichos valores son siempre igual a 0 y k^L-1 respectivamente. Además, $x(\sigma)$ es siempre un número entero positivo.

En la Figura 11 se observa que el simple reescalamiento lineal permite representar números negativos. Las figuras 12 y 13 permiten también intuir que, el reescalamiento lineal permite representar números que no sean enteros.

A partir de la misma ecuación de $w(x)$, se puede deducir con qué precisión quedan las variables reales.

$$w(x) = w_{\min} + \frac{w_{\max} - w_{\min}}{k^L - 1} \cdot x$$

$$w(x) = w_{\min} + \pi \cdot x$$

Ya que x es siempre un entero, entonces π viene a ser la precisión.

La fórmula de $w(x)$ también se podría reescribir en términos de la cadena así:

$$w(a_1, a_2, ..., a_L) = w_{\min} + \frac{w_{\max} - w_{\min}}{k^L - 1} \cdot \left(a_1 \cdot k^{L-1} + a_2 \cdot k^{L-2} + \cdots + a_L \cdot k^0\right)$$

Teniendo en cuenta lo dicho hasta el momento, se puede decir ahora cómo diseñar un cromosoma que codifique variables reales. Se debe seguir los siguientes tres pasos:

1. Decidir el valor de k: Normalmente se adopta un código binario, es decir, un k igual a 2.

2. Definir el w_{min} y el w_{max}: ¿Cuáles son el máximo y el mínimo valor que podrá asumir la variable real?

3. Calcular el L.: Definido ya lo anterior, se procede a determinar la longitud que tendrán los cromosomas. Esta se calcula de la expresión que a continuación se demuestra:

$$\pi = \frac{w_{\max} - w_{\min}}{k^L - 1} \qquad \textit{por definición de } \pi$$

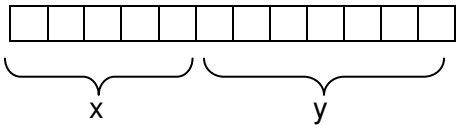
$$k^L - 1 = \frac{w_{\max} - w_{\min}}{\pi}$$

$$L = \log_k\left(\frac{w_{\max} - w_{\min}}{\pi} - 1\right)$$

$$L = \frac{\ln\left(\frac{w_{\max} - w_{\min}}{\pi} - 1\right)}{\ln(k)} \qquad (\textbf{\textit{Eq. 1}})$$

Otro aspecto a tener en cuenta en la codificación, es cuando se tiene un problema cuya solución está compuesta de varios números. Es decir, cómo se codifica cuando la función objetivo fuese como la siguiente: $f(x_1, x_2, x_3, x_4) = 5\,x_1 + 4\,x_2 + 4\,x_3 + 4\,x_4$

Esta situación se presenta en el caso estudio. Para ello, se concatena los diferentes cromosomas en uno sólo. Por ejemplo, supóngase que se desea maximizar la función $f(x, y) = x^2 - \cos y$. Si después de hacer los cálculos correspondientes, se determinó que x requiere una cadena de longitud $L_1 = 5$ y y con una cadena de longitud $L_2 = 12$, entonces se trabajará con cadenas de longitud $L = 5 + 12$ así:



Existe una pregunta abierta sobre la mejor forma de codificar las soluciones. Mientras Golberg afirma que se debe escoger siempre el alfabeto más pequeño posible (favoreciendo así la codificación binaria)[53], otros favorecen alfabetos más extensos como la codificación directa con números reales[54]. Pero siempre, la codificación dependerá del tipo de problema que se esté atacando†. Por ejemplo, en problemas de optimización combinatoria se sugieren codificaciones muy diferentes a la binaria[55].

2.1.5.2 *La selección*. En un algoritmo genético simple, la selección se hace por ruleta. En dicho procedimiento, cada cromosoma adquiere una probabilidad de ser seleccionado dependiendo del valor que adquiere al ser evaluado por la función objetivo. Por esta razón también recibe el nombre de selección proporcional (**proportional selection**). En este caso, la función objetivo $f(x)$ hace también las veces de función de aptitud $g(x)$ (**fitness function**). Sin embargo, existen muchas razones para que no coincidan estas dos funciones. Estos casos se presentan cuando:

- El problema consiste en minimizar en vez de maximizar la función objetivo. Goldberg[56] sugiere emplear una funcion de aptitud así: $g(x) = C\max - f(x)$. Sin embargo en este caso no existe un criterio único para seleccionar la constante $Cmax$.

- La función objetivo puede arrojar números negativos. Goldberg[57] vuelve sugerir la aplicación de una constante así: $g(x) = f(x) + Cmin$. Pero también surge una ambigüedad para escoger a $Cmin$.

- La función objetivo puede otorgar valores muy altos a los cromosomas más aptos, o valores muy parejos. Cuando un algoritmo genético se encuentra en las primeras

[53] GOLDBERG. 1989. p. 80.

[54] DUMITRESCU *et al.* 1989. p. 210.

† Al respecto se hablará con más detalle en …1.4.5. Epistasis, funciones engañosas y otros fenómenos….

[55] FALKENAUER. 1998. p. 97.

[56] GOLDBERG. 1989. p. 76.

[57] GOLDBERG. 1989. p. 76.

generaciones y la función otorga valores muy altos, el AG puede entonces seleccionar muchas veces a unos cuantos cromosomas que parecieran ser muy aptos, pero que pueden causar una convergencia prematura a una respuesta no deseada. Igualmente, después de muchas generaciones, los cromosomas en la población tienden a tener aptitudes muy parecidas y por lo tanto, no es conveniente que la función objetivo arroje valores muy parejos. Goldberg[58] propone en este caso un tipo función de aptitud $g(x)$ dinámica que varía a través de las generaciones.

- Ya que la selección por ruleta es al azar, se corre el riesgo de que efectivamente no se seleccionen los mejores cromosomas.

Ante estas complicaciones que puede presentar la simple selección por ruleta, se están empleando además otros esquemas de selección como los que se presentan a continuación[59]:

- Selección determinística (**deterministic selection**). Se hallan las aptitudes mediante la función de aptitud $g(x)$ y se calculan los pesos. Sin embargo, estos pesos no se emplean para actuar como probabilidades de selección, sino para seleccionar de manera determinística los mejores cromosomas en cantidades proporcionales a su peso. Este método sólo busca impedir que no se seleccionen los mejores candidatos. Muchas veces este método hace que se converja prematuramente a una solución sub-óptima, pues no se le permite al AG explorar soluciones que aunque de baja aptitud, podrían ayudar a converger a la verdadera solución.

- Ranking (**ranking**). La selección solo tiene en cuenta el puesto en aptitud que tiene un individuo con respecto al resto de la población. Por lo tanto, la función de aptitud se define así:

$$g(x_i) = \frac{g_{\max} - g_{\min}}{n} \cdot puesto\left(f(x_i), \sum_{i=1}^{n} f(x_i) \right)$$

en donde n es el tamaño de la población y en donde la función $puesto()$ arroja la posición del cromosoma entre 1 y n. De esta manera, se obtienen los pesos para una posterior selección por ruleta. Esto resuelve varios de los problemas pero es muy ajeno a los diversos valores que puede arrojar la función objetivo. En palabras de Falkenauer, "no se siente natural", ya que ignora los valores propios de $f(x)$, es decir, se ignora el medio ambiente.

- Normalización (**normalization**). Los valores de la función objetivo $f(x)$, simplemente se reescalan. Así, se impide obtener valores negativos y de paso, en el caso de un problema de minimización, se convierten los valores bajos en valores altos y viceversa. La función de aptitud $g(x)$ queda entonces dada por

[58] GOLDBERG. 1989. p. 76-79.

[59] FALKENAUER. 1998. p. 46-50

$$g(x) = g_{\min} + \frac{g_{\max} - g_{\min}}{f_{\max} - f_{\min}} \cdot [f(x) - f_{\min}]$$

Esta fue la metodología que se adoptó para el programa que se desarrolló en el caso estudio. En dicho caso, el f_{max} (f_{min}) correspondió al máximo (mínimo) valor de la función objetivo que se presentaba en la población. Se estableció además que g_{min} fuese igual a 10 y g_{max} fuese igual a 0, ya que se trataba de un problema que requería minimizar una función objetivo (en caso de tratarse de una maximización, entonces g_{min} sería igual a 0 y g_{max} sería igual a 10).

He aquí como sería dicho reescalamiento, pero con una función objetivo diferente a la del caso estudio:

Tabla 1. *Obtención de pesos mediante selección con normalización (g_{min} = 0, g_{max} = 10)*

x	$f(x) = x^3$		$g(x)$ *Aptitud*	*Peso*
25	15 625	$10 + \frac{0-10}{(15625)-(-32768)} \cdot [15625-(-32768)] =$	0,00	0,00
23	12 167	$10 + \frac{0-10}{(15625)-(-32768)} \cdot [12167-(-32768)] =$	0,71	0,04
-15	-3 375	$10 + \frac{0-10}{(15625)-(-32768)} \cdot [-3375-(-32768)] =$	4,01	0,22
-10	-1 000	$10 + \frac{0-10}{(15625)-(-32768)} \cdot [-1000-(-32768)] =$	3,44	0,19
-32	-32 768	$10 + \frac{0-10}{(15625)-(-32768)} \cdot [-32768-(-32768)] =$	10,00	0,55
		$\sum g(x) =$	18,16	

No obstante, este método presenta la incertidumbre al no haber un criterio fuerte para escoger los valores de g_{min} y g_{max}. Por ejemplo, si para la anterior tabla se hubiese escogido reescalar entre 10 y 20, se obtendría una distribución de los pesos muy diferente.

Tabla 2. *Obtención de pesos mediante selección con normalización (g_{min}=0, g_{max}=20).*

x	$f(x) = x^3$		$g(x)$ *Aptitud*	*Peso*
25	15 625	$20+\frac{10-20}{(15625)-(-32768)}\cdot[15625-(-32768)]=$	10	0,15
23	12 167	$20+\frac{10-20}{(15625)-(-32768)}\cdot[12167-(-32768)]=$	10,71	0,16
-15	-3 375	$20+\frac{10-20}{(15625)-(-32768)}\cdot[-3375-(-32768)]=$	13,93	0,20
-10	-1 000	$20+\frac{10-20}{(15625)-(-32768)}\cdot[-1000-(-32768)]=$	13,44	0,20
-32	-32 768	$20+\frac{10-20}{(15625)-(-32768)}\cdot[-32768-(-32768)]=$	20,00	0,29
		$\sum g(x)=$	68,08	

Compárese la Tabla 2 con la Tabla 1. Se observa que en este caso, se obtuvieron unos pesos con valores más parejos. Así, la selección con normalización deja también sus dudas de si es o no el procedimiento más natural y adecuado.

- Torneo (**Tournament**). Ante los anteriores inconvenientes, surge esta novedosa estrategia que busca ser natural y fácil de implementar. Aquí, se simula la competencia que se da naturalmente entre los individuos por su supervivencia. Parece superar todos los inconvenientes que presenta la selección proporcional[†] (ver Figura 14).

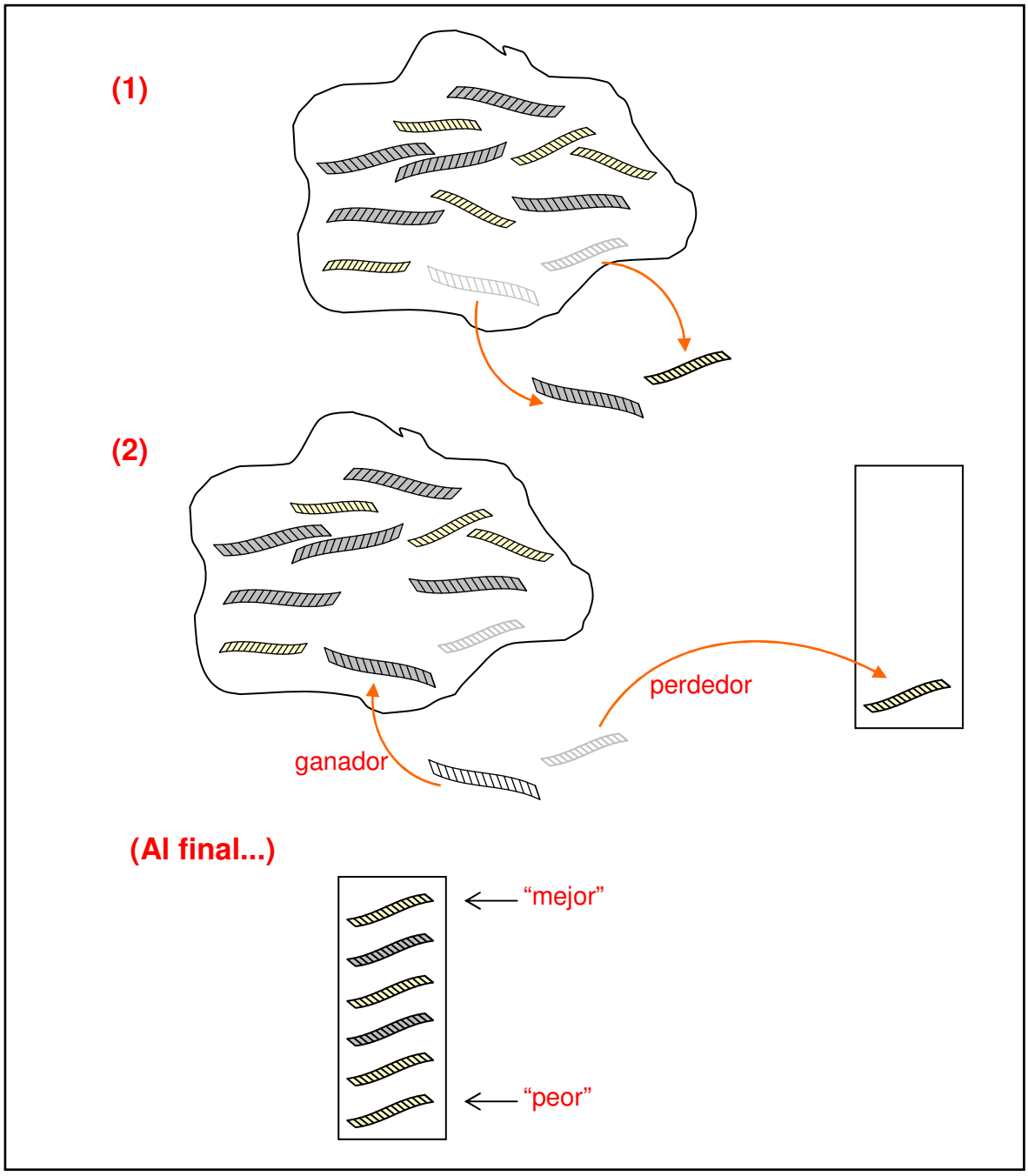
  La estrategia consiste en agrupar inicialmente todos los individuos de la población en un estanque. Se escoge al azar dos individuos de la población y se les compara entre sí de acuerdo a la función objetivo $f(x)$. El mejor vuelve al estanque y el peor pasa al fondo de una pila. Luego se vuelve a sacar al azar otra pareja de individuos y se vuelven a comparar. Se repite este proceso hasta que ya no quede ningún cromosoma en el estanque. Luego la pila contendrá en el fondo, el "peor" cromosoma y en la parte superior el "mejor". Como la selección del estanque se hizo probabilísticamente, ya no se requiere hacer una selección por ruleta.

  En este momento, se obtiene una pila con todos los cromosomas organizados en una jerarquía. Sin embargo aún queda por resolver quiénes serán padres y quiénes no. Pues existen varias maneras. Una es seleccionar la mitad hacia arriba para que se reproduzcan entre sí y llenen la mitad hacia abajo. Cualquier otra forma debe simplemente darle mayor prioridad a los cromosomas de arriba para que sean padres.

.

[†] No obstante, al igual que los otros métodos de selección, aun no se ha demostrado su efectividad desde la teoría.

Figura 14. *Selección por torneo.*



2.1.5.3 *Los operadores*. Una vez se hayan seleccionado los cromosomas que serán padres, se procede a hacer los entrecruzamientos y las mutaciones. Los AGs, a diferencia de otros algoritmos evolutivos, tienen predilección por realizar entrecruzamientos como forma principal de construir nuevas soluciones. Y las mutaciones en cambio, se realizan como un operador secundario. De ahí que éstas se suelen aplicar con probabilidades p_m por debajo de 0,05. No obstante, la mutación es tan importante como el entrecruzamiento

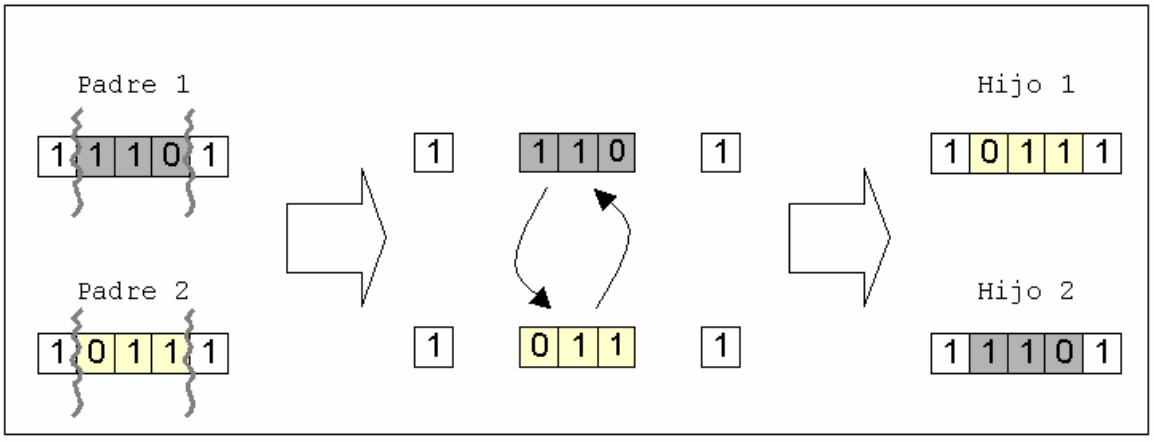
pues se encarga de explorar nuevos espacios de búsqueda ayudando así a que el AG no se estanque en una solución sub-óptima (en un pico local).

Aunque el algoritmo genético simple boga por respetar mucho los anteriores dos operadores se ha experimentado mucho, tanto modificando la forma como se aplican estos operadores, como el agregar otros operadores.

La primera modificación puede ser simplemente olvidarse de la probabilidad de entrecruzamiento p_c, es decir, fijarla en 1,0. Otra variación es emplear el entrecruzamiento de dos puntos (**double-point crossover**) o el entrecruzamiento uniforme (**uniform crossover**).

Para el programa desarrollado en el caso estudio, se empleó un entrecruzamiento de dos puntos. Sus ventajas sólo se podrán comprender en ...1.4.1. Esquemas, observación 4....

Figura 15. *Operador de entrecruzamiento de dos puntos.*



2.1.5.4 *La muerte y la supervivencia*. Después de seleccionar los padres y generar a los hijos, se procede a determinar quiénes de la anterior generación morirán. En el AG simple, los n hijos de la nueva generación reemplazan a todos los n individuos de la generación anterior. Esto es lo que se conoce como procedimientos estrictamente generacionales (**strictly generational procedures**)[60]. Otros procedimientos permiten traslapo (**overlap**). En estos casos, se establece la brecha generacional (**generation gap**) la cual hace referencia al porcentaje de la población que será reemplazada por la nueva generación[61].

Otra forma de proceder diferente al AG simple, es la de adoptar algún tipo de *elitismo* (**elitism**)[62], que consiste en conservar de alguna forma los mejores individuos, sin dejar que desaparezcan por algún tipo entrecruzamiento o mutación.

[60] DUMITRESCU. 2000. p. 27.

[61] DUMITRESCU. 2000. p. 27.

[62] GOLDBERG. 1989. p. 115.

En el programa que se desarrolló se aplicó un elitismo que consistía en reemplazar el peor individuo de una generación, con el mejor cromosoma de la generación anterior. Con esto se garantizaba que el AG conservase la mejor solución a través de las generaciones hasta que se encontrara uno mejor. El elitismo corre el riesgo de hacer converger prematuramente al AG, pues puede impedirlo de explorar otros espacios de búsqueda igualmente prometedores.

Hasta este punto, ya se explicó de manera amplia lo que es un AG y se dio una introducción hacia las diferentes versiones que puede tomar. Se dieron también algunas luces sobre cómo se desarrolló el programa computacional para el caso estudio del presente proyecto. No obstante, todas las características de dicho programa se expondrán luego en ... 2.4. DESCRIPCIÓN DEL PROGRAMA .... Ahora, antes de responder a la pregunta cómo funcionan, se dejará de un lado las explicaciones rigurosas y se hará un pequeño paréntesis para ver cómo los AGs fue abriendo su espacio en la historia.

1.3. ¿CÓMO SURGIERON? *UNA BREVE HISTORIA*

Antes de desarrollarse las primeras ideas sobre algoritmos genéticos, existieron trabajos previos relacionados con los temas de *aprendizaje de máquina* (**machine learning**) y *optimización numérica* (**numerical optimization**)[63,64]. Algunos de sus investigadores pioneros fueron Turing (1950), Box (1957), Friedberg (1958), Martin y Cockermann (1960), Bledsoe (1961) y Bremmerman (1962).

Existió también otra línea previa de investigación dedicada a la simulación de sistemas genéticos por computadora, hecha principalmente por biólogos[65,66,67] como Barricelli (1957) y Fraser (1957).

Holland publicó sus primeras ideas sobre algoritmos genéticos entre 1962 y 1965[68]. Era la década de los sesentas y existía en ese entonces, un creciente interés por emplear las computadoras, cada vez más asequibles, como herramientas científicas para la simulación[69]. Bajo este clima, Holland se interesó por la idea de[70] "diseñar e implementar sistemas artificiales altamente adaptativos capaces de enfrentarse a inciertos y cambiantes ambientes". La literatura ampliamente resalta el enfoque tan abierto que

[63] KARR and FREEMAN. 1999. p. 5.

[64] DUMITRESCU *et al.* 2000. p. 6.

[65] KARR and FREEMAN. 1999. p. 5.

[66] DUMITRESCU *et al.* 2000. p. 7.

[67] GOLDBERG. 1989. p. 89.

[68] GOLDBERG. 1989. p. 90 – 92.

[69] DE JONG. 1999. p. 1.

[70] DE JONG. 1999. p. 1.

tenían sus investigaciones, es decir, el hecho de que no estaba interesado en resolver algún tipo de problema específico[71,72,73] (lo cual le dio a los AGs propiedades de robustez). En aquel tiempo, dictaba clases de *sistemas adaptativos* en la universidad de Michigan, en las cuales sus estudiantes aplicaban reproducciones, entrecruzamientos y mutaciones a sus algoritmos, tal como se hace hoy día[74]. A partir de 1968, Holland desarrolló sus primeros análisis sobre *esquemas*[¶] y finalmente en 1975, publica su obra maestra **Adaptation in Natural and Artificial Systems**. Así, este ingeniero eléctrico y sicólogo, es ampliamente acreditado como el creador de los algoritmos genéticos[75,76].

La publicación de su obra en 1975 no tuvo el gran impacto que él esperaba. Se le criticaba por no pertenecer realmente a la inteligencia artificial, o por motivar a simular un proceso que a la naturaleza le había llevado billones de años. Al principio, sólo sus estudiantes y colegas empleaban su libro en sus investigaciones[77]. Sin embargo, cinco años después, Holland comenzó a observar interés por su obra. Esto se debió en parte, al cambio de enfoque que estaba presentando la inteligencia artificial al dedicarse más hacia los problemas de aprendizaje[78]. Por otro lado, comenzaron a aparecer estudios comparativos de los AGs en áreas que van desde el diseño integrado de circuitos hasta el diseño de portafolios de acciones y turbinas de aeronaves[79].

Según De Jong[80], la década de los setentas se empeñó por tratar de responder a dos preguntas: (1) cómo podrían emplearse los AGs para resolver problemas y (2) cómo caracterizar el comportamiento de los AGs. Así que durante esta década hubo una preocupación por adquirir más conocimientos a través de la investigación empírica. Luego[81], la década de los ochentas vio como dominaron las aplicaciones relacionadas con la optimización matemática y en los noventas, surge un interés por mejor comprender cuándo y cuándo no, funcionan adecuadamente los AGs (sobretodo, ante la gran variedad de modificaciones que se les estaba haciendo).

---

[71] DE JONG. 1999. p. 1.

[72] GOLDBERG. 1989. p. 90.

[73] KARR and FREEMAN. 1999. p. 6.

[74] GOLDBERG. 1989. p. 92.

[¶] Los "esquemas" constituyen la teoría estándar de los AGs. Se verán más adelante en …1.4.1. Esquemas….

[75] KARR and FREEMAN. 1999. p. 6.

[76] KARR and FREEMAN. 1999. p. 6.

[77] HOLLAND. 1992. p. ix.

[78] HOLLAND. 1992. p. ix.

[79] HOLLAND. 1992. p. ix.

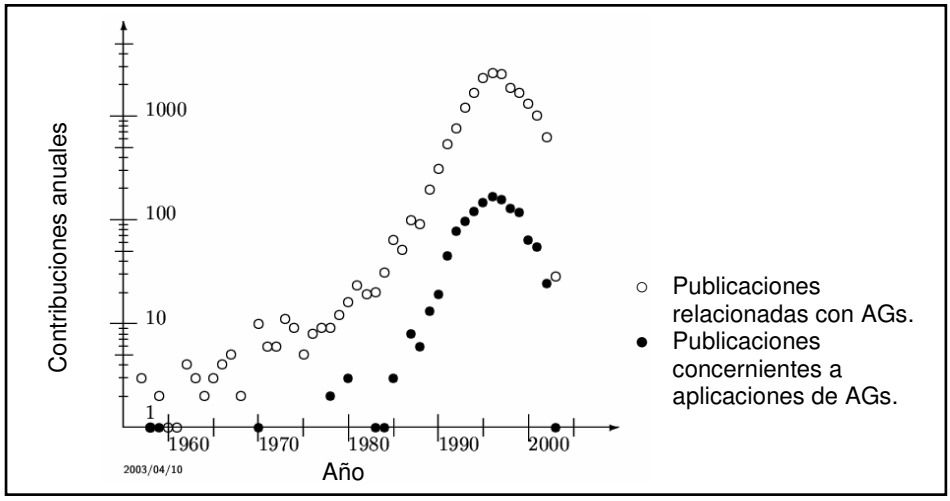
[80] DE JONG. 1999. p. 2.

[81] DE JONG. 1999. p. 3.

Existieron otras dos obras que también tuvieron gran impacto[82]. La publicación en 1989 de **Genetic Algorithms in Search, Optimization, and Machine Learning** de David E. Goldberg, presentaba a los AGs como una herramienta para resolver problemas, exponía varios ejemplos en donde investigadores hacían aplicaciones en muy diversas áreas. Además presentaba explicaciones teóricas claras y concisas. Así, Goldberg aceleró la aplicación de los AGs en el mundo real. Es interesante anotar que él, además de haber sido alumno de Holland, era ingeniero civil.

Por otro lado, Dave Davis publicó en 1991 **The Handbook of Genetic Algorithms**, el cual además de dar una breve fundamentación del tema, exponía capítulos de investigadores que habían aplicado con éxito los algoritmos genéticos en varios campos. Sirvió así de testimonio sobre la efectividad y robustez de la herramienta.

Figura 16*. Publicaciones anuales relacionadas con algoritmos genéticos 1958-2003 (Nota: los datos están aún incompletos para el período entre1998 y 2003)*[83]



El interés por los algoritmos genéticos está definitivamente creciendo[84,85]. Aparte del texto de Goldberg ya se han publicado otros más†. El número de publicaciones relacionadas con el tema ha crecido exponencialmente durante las décadas[86], afianzándose así como un sistema inteligente de gran aplicación.

---

[82] KARR and FREEMAN. 2000. p. 6.

[83] ALANDER. 2003. p. 6. Fig. 3.1.

[84] DE JONG. 1999. p. 1.

[85] KARR and FREEMAN. 2000. p. 7.

† Según KARR and FREEMAN. 2000. p. 7, 13. estos incluyen: DAVIDOR, Y. Genetic algorithms and robotics: a heuristic strategy for optimization. Singapore : World Scientific Publishing. 1991 ; RAWLINGS, G.J.E. Foundations of genetic algorithms. San Mateo, CA, USA : Morgan Kaufman Publishers. 1991. ; MICHALEWICZ, Z. Genetic algorithms + data structures = evolution programs. Belin : Springer-Verlag. 1992.

[86] KARR and FREEMAN. 2000. p. 7.

1.4. ¿FUNCIONAN? *UNA REVISIÓN A LOS FUNDAMENTOS TEÓRICOS*

La anterior sección da testimonio del éxito que han tenido los algoritmos genéticos en diferentes tipos de problemas, pero también se alcanza a ver que existe una constante investigación indagando sobre su verdadera eficacia. Pues bien, hay que ser claro en este tema.

Los AGs pueden ser aplicados a cualquier problema de optimización matemática[†]. No obstante, si la función objetivo no presenta alta complejidad entonces es muy probable que un método tradicional basado en el cálculo supere al algoritmo genético. Una función es compleja cuando no es contínua, no es derivable, es multimodal, tiene ruido y es multivariable[§]. Estos tipos de funciones se pueden observar en la Figura 17. En palabras de algunos investigadores[87], "los AGs no garantizan encontrar la solución óptima global a un problema, pero son buenos en encontrar soluciones aceptablemente buenas en un tiempo aceptablemente rápido". Es decir, se sacrifica la eficacia (llegar al óptimo global) por la eficiencia (rapidez de la solución) y aplicabilidad.

La Figura 15 permite intuir que los problemas complejos son situaciones más generales. Así, es de sospechar que son estos casos los que con mayor frecuencia se presentan en la naturaleza y por ende en la ingeniería. Goldberg[88] hace la crítica de cómo muchos teóricos de hoy, aun se obstinan en aceptar "aquel legado de los grandes matemáticos del siglo XVIII y XIX quienes pintaban [en sus elegantes ejercicios] un mundo limpio con funciones cuadráticas, restricciones ideales y derivadas siempre presentes".

Existen otros métodos diferentes al cálculo tales como los *enumerativos*. Estos consisten en ir recorriendo el espacio de búsqueda hasta encontrar la solución. Uno de estos métodos es la *programación dinámica*. Fallan cuando tienen que enfrentarse a espacios de búsqueda bastante grandes[89].

[†] En lo que resta de la presente sección sólo se hablará del funcionamiento de los AGs a la luz del problema de optimización matemática. En la sección …1.2.4. Problemas a ser solucionados por los AGs…, se vio que a partir de este tipo de problema, se pueden comprender las demás aplicaciones que tienen los AGs.

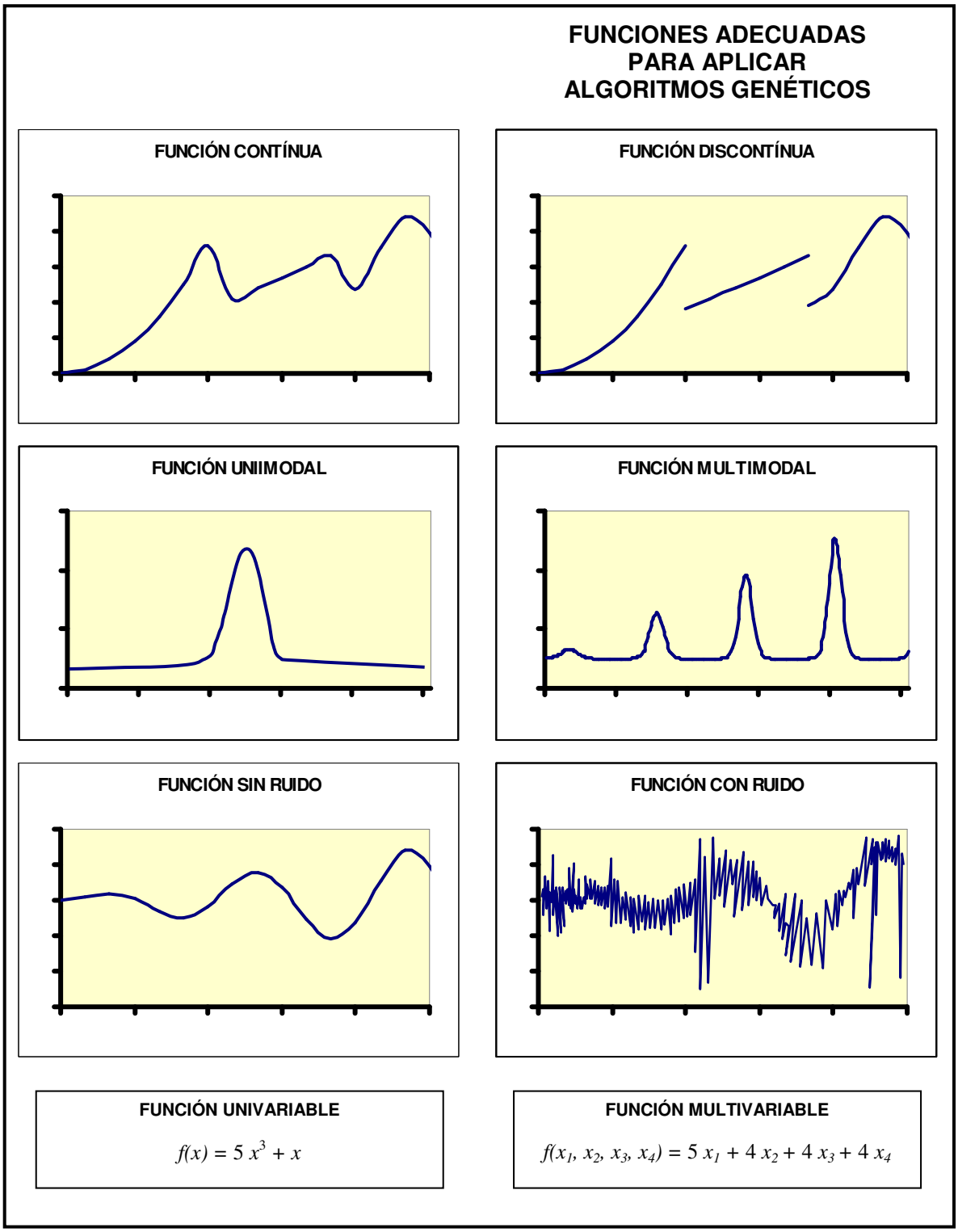
[§] Aquí se tomó arbitrariamente el término "complejo" para hacer referencia a dicho tipo de funciones.

[87] BEASLEY, BULL and MARTIN. 1993. p. 2.

[88] GOLDBERG. 1989. p. 3.

[89] GOLDBERG. 1989. p. 5.

Figura 17. *Tipos de funciones.*



La pregunta que sigue ahora es la siguiente. ¿Si se tiene un problema que no puede ser resuelto mediante el cálculo o mediante un método enumerativo, entonces podrá ser resuelto por un algoritmo genético (así sea sacrificando un poco de eficacia por eficiencia)? La respuesta no es "obviamente que sí". Para responderla, los investigadores han tenido que entrar a estudiar de una manera rigurosa el funcionamiento de los algoritmos genéticos. Estos estudios han permitido concluir que existen fenómenos como el de la *epistasis* y el de las *funciones engañosas* que le dificultan la labor a los AGs. Pero será estudiando estos fenómenos y el verdadero funcionamiento lo que permitirá adecuar los AGs para que sean totalmente exitosos. Lo que resta de la siguiente sección, tendrá por objeto ilustrar la teoría que busca explicar cómo funcionan los algoritmos genéticos.

Existen dos líneas. La primera es la teoría estándar de los AGs[†] propuesta por Holland[‡] y luego más ampliamente expuesta por De Jong, Golberg, Greffenstette, entre otros[§]. A pesar de recibir varias críticas (por hacer simplificaciones no válidas[90], por llegar a conclusiones erradas[91,92] o por sólo explicar al algoritmo genético simple), lo expuesto por Holland constituyó la primera explicación matemática[93], es aún ampliamente expuesta en la literatura y es el punto de partida para la comprensión de los AGs[94]. Por otro lado, existe la segunda línea de hipótesis que busca valerse de procesos markovianos[Δ]. Ésta no está limitada a explicar el algoritmo genético simple. De todas maneras, la literatura sugiere que la teoría es aún insuficiente para poder comprender el funcionamiento de los AGs, sobretodo ante las variaciones que se le hacen en la práctica[95,96].

El presente trabajo optó por ilustrar la teoría clásica de los AGs. Si el lector no se encuentra interesado en comprenderla, se recomienda leer simplemente el apartado …1.4.1. Esquemas… y observar con detenimiento la Figura 22. En caso contrario, tendrá

[†] Así la denomina Falkenauer en FALKENAUER. 1998. p. 58.

[‡] …en su clásica obra Adaptation in Natural and Artificial Systems. 1975.

[§] En WILSON. 1996. p. 5., se habla de cómo De Jong hace una buena discusión sobre las asunciones y derivaciones de lo expuesto por Holland. En FALKENAUER. 1998. p. 58-77 se afirma varias veces el aporte que Goldberg hace al explicar nuevamente lo expuesto por Holland pero de una manera mucho más comprensible. Finalmente en STILLWELL. 2001. p. 18., se expone cómo Grefenstette hace una explicación formal de la hipótesis de los bloques constructores.

[90] BEASLEY, BULL and MARTIN. 1993. p. 8.

[91] MACREADY and WOLPERT. 1996. p. 28.

[92] FALKENAUER. 1998. p. 76.

[93] BEASLEY, BULL, and MARTIN. 1993. p. 7.

[94] BANZHAF, Wolfgang *et al*. Genetic programming an introduction: on the automatic evolution of computer programs and its applications. 1998. p. 146, citado por STILLWELL. 2001. p. 18.

[Δ] Este segundo tema está fuera del alcance del presente trabajo, así que sólo se recomendará leer a GOLDBERG and SEGREST. 1987; DE JONG, SPEARS and GORDON. 1995; CHAKRABORTY and CHAKRABORTY. 1997; y PROCOPIO. 2001.

[95] DE JONG and SPEARS. 1995. p. 1, 2.

[96] PROCOPIO. 2001. p. 1.

la oportunidad de apreciar lo que aquí se hizo para exponerla de una manera clara y actualizada.

Antes de hacer todo el planteamiento, se debe primero comprender el concepto de esquemas y segundo, el problema análogo del que se valió Holland para explicar su teoría: el problema del tragamonedas de dos palancas (**the two-armed bandit problem**)[¶].

1.4.1 Esquemas. Toda la teoría clásica se basa en el concepto de esquema (**schema**, y su plural, **schemata**). Holland observó que para poder entender cómo funciona un algoritmo genético, más que ponerse a analizar los cromosomas, hay que analizar los esquemas que se esconden tras ellos. El autor considera aquí que los esquemas son de cierta forma esas ideas, buenas o malas, de las cuales se habló en …1.2.3. Características de los AGs…[†].

Un esquema es una plantilla que hace referencia a un conjunto de cromosomas que tienen valores idénticos en ciertos loci. Un esquema está compuesto por los mismos caracteres del código que se esté empleando y además, un carácter comodín * (**don't care symbol**). Por ejemplo, en el caso de un algoritmo genético que emplease un código binario y cromosomas de longitud *l* igual a 4, el siguiente esquema

1	*	0	*

haría referencia al siguiente conjunto de cromosomas.

1	0	0	0

1	0	0	1

1	1	0	0

1	1	0	1

Cada 0 y cada 1 en el esquema indica uno de los loci que debe contener dichos valores fijos. Cada carácter comodín * indica que en dicho locus puede ir un 1 o un 0.

Holland le definió dos características a los esquemas, un *orden* (**order**) y una *longitud definidora* (**defining length**). El orden es el número de valores fijos (ceros y unos) dentro del esquema. La longitud definidora (o simplemente *longitud*) es la distancia entre los dos valores fijos extremos en el esquema, o más precisamente, es la resta entre las posiciones de los valores fijos extremos. Al orden se le designa con la letra o y a la longitud con la letra δ. He aquí unos ejemplos.

[¶] Las siguientes explicaciones se tomaron de las siguietes dos obras: GOLDBERG. 1989. y FALKENAUER. 1998. Para la sección …1.4.2. El problema del tragamonedas de dos palancas…, se tomaron comentarios adicionales de MACREADY AND WOLPERT. 1996. y WILSON. 1996. El tema de la epistasis se basó primordialmente en BEASLEY, BULL and MARTIN. Part 2. 1993, y en menor grado en FALKENAUER. 1998 Finalmente, la amalgama de críticas citadas en …1.4.6. Críticas a la teoría estándar… se tomaron de diferentes autores, tal como se indica en los respectivos pié de página.

[†] Aunque el autor no encontró esta afirmación escrita textualmente, intuye que es la opinión de muchos autores.

Esquema	Orden o	Longitud δ
0 * 0 * * * 1 *	3	7 – 1 = 6
* 0 * 0 * * * 1 *	3	7 – 1 = 6
1 * * *	1	1 – 1 = 0
* * * * * * * *	0	0
* 1 1 0 * * 0 0	5	8 – 2 = 6
1 0 1 0 0	5	5 – 1 = 4

En lo que resta del presente documento, decir que un esquema sea largo o corto dependerá del valor de δ y no de l. Así, el esquema ***0*1******** es más corto que el esquema 0***1**. Como dato interesante, es de mencionar que en la literatura, a los esquemas se les suele llamar también *hiperplanos* (**hyperplanes**)[‡].

Holland y los que le siguieron, observaron en los esquemas propiedades que permiten comenzar a explicar lo que realmente pasa en los AGs. El autor vio conveniente ordenar estas ideas e ilustrarlas claramente mediante las siguientes cuatro "observaciones"[†]. Estas observaciones le permitirán luego al lector, comprender la teoría general de los AGs con mayor claridad.

1. Observación: *En un cromosoma, se pueden distinguir muchos esquemas a la vez.*

   Por ejemplo, en el siguiente cromosoma,

1	0

   existen a la vez cuatro esquemas que son:

*	*		*	0		1	*		1	0

   En general, en un cromosoma de longitud l existen $2 \cdot 2 \cdot 2 \cdot \ldots \cdot 2 = 2^l$ esquemas a la vez. En el ejemplo anterior, el cromosoma 10 contiene simultáneamente $2^2 = 4$ esquemas.

   Ahora, si se tiene una población de n cromosomas pueden existir hasta $n \cdot 2^l$ esquemas a la vez. Así, detrás de n cromosomas se esconde en realidad mucha más información. En otras palabras, detrás de los cromosomas se enconden muchas ideas, sean buenas o malas.

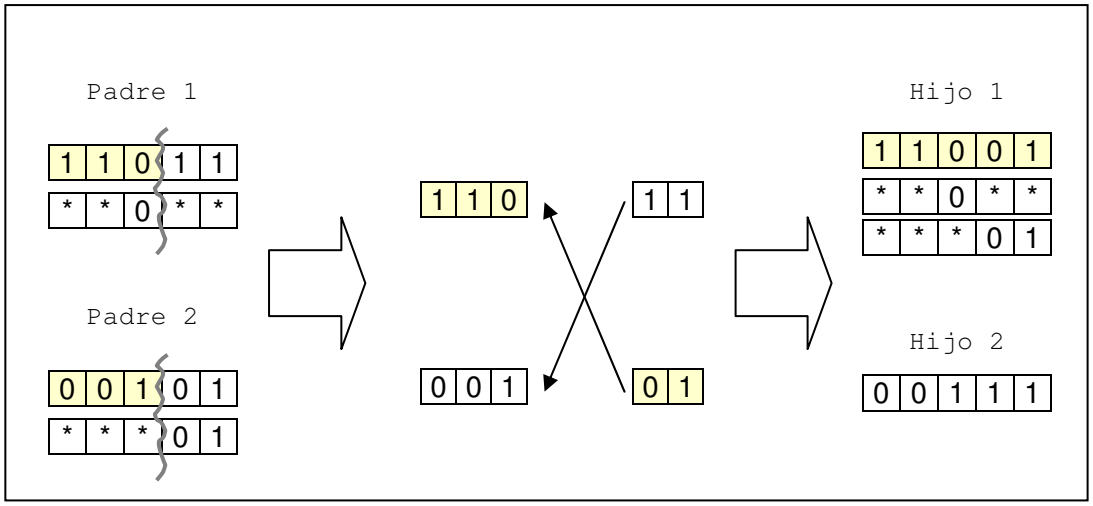
[‡] En realidad, esto es más que un dato interesante. Es una analogía que se hace con la geometría que busca una explicación más clara del concepto de esquema. No obstante, esto no es algo importante para el enfoque que se adoptó en el presente documento. Véase más en FALKENAUER. 1998. p. 63. o en GOLDBERG. 1989. p. 53.

[†] Se les denominó así "observaciones" pues no involucran ni supuestos, ni teorías ni hipótesis. Son afirmaciones que parten de la simple observación.

2. Observación: *Durante la operación de entrecruzamiento, los hijos no sólo heredan genes sino también esquemas de sus padres. Esta herencia permite además, construir a partir de esquemas cortos, esquemas más largos.*

   En la Figura 18 se analiza lo que pasa con los esquemas **0** y ***01. El hijo 1 no sólo pasa a contener estos dos esquemas, sino que también pasa a contener el esquema **001 (el cual no poseía ninguno de sus padres).

Figura 18. *Transmisión de esquemas de padres a hijos.*



Así como hubo una herencia de los esquemas escogidos en este ejemplo, simultáneamente se realizan otras herencias de esquemas. Pero algunas herencias fracasan. En la Figura 18 se puede observar también lo que le sucedió al esquema 1**1* en el padre 1. Éste no logró transmitirse a ninguno de los dos hijos. Simultáneamente el esquema 1***1 en el padre 1, aunque pareciera que se iba a perder con el entrecruzamiento, logró sobrevivir y ser transmitido al hijo 1. Sin duda, no es fácil poder predecir toda esta transmisión, muerte y construcción de esquemas en paralelo. Pero en las siguientes dos observaciones, se busca dilucidar algo más con respecto a esta manipulación de esquemas.

3. Observación: *Los esquemas de menor longitud δ tienden a conservarse más a través de las generaciones que los esquemas de mayor δ.*

   Sea por ejemplo el siguiente cromosoma

alelos:	1	0	1	0	0	0	1	0
loci:	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*

   el cual contiene (entre los 2^8= 256), a los siguientes tres esquemas

Esquema 1	*	0	0	*	*	*	1	*
Esquema 2	*	0	*	*	*	*	1	*
Esquema 3	*	*	*	0	0	*	*	*

Si el cromosoma sufriese un entrecruzamiento, existen siete posiciones donde se podría dar la ruptura: de la 2 hasta la 8[†]. Si la ruptura ocurriese en el locus 3, es muy probable que los esquemas 1 y 2 se dañen mientras que el esquema 3 se conservaría. El esquema 3 se dañará si la ruptura ocurre en el locus 5, mientras que los esquemas 1 y 2 se dañarán si las rupturas ocurren en los loci 3, 4, 5, 6 o 7. De aquí se puede observar que los esquemas de longitud δ, tienen δ posiciones críticas en el momento de hacer un entrecruzamiento. Ahora, teniendo en cuenta que el número de posiciones donde puede ocurrir la ruptura en un cromosoma de longitud l es l - 1, entonces:

la probabilidad de que un esquema se pierda tras un entrecruzamiento en el cromosoma que lo contiene, es de $\frac{\delta}{l-1}$.

Así, se puede concluir que cuando un cromosoma sufre entrecruzamiento, los esquemas de mayor longitud δ tendrán una mayor probabilidad de perderse. La probabilidad que aquí se acabó de determinar no tiene en cuenta los casos afortunados como el que ocurrió en la Figura 18, en donde el esquema 1***1 del padre 1, aunque sufrió una ruptura logró sobrevivir y ser transmitido al hijo 1. Por lo tanto, se debe replantear lo dicho anteriormente así:

la probabilidad de que un esquema se pierda tras un entrecruzamiento en el cromosoma que lo contiene, es mayor que $\frac{\delta}{l-1}$[¶].

La anterior observación permite hacer un pequeño paréntesis para comentar acerca del operador de entrecruzamiento de dos puntos (Figura 15). En él, se escogen al azar dos puntos de ruptura, uno inicial y uno final. Si el final queda a la derecha del inicial, el entrecruzamiento se realiza como en la Figura 15. Si el final queda a la izquierda del inicial, entonces el cromosoma se considera como circular y entonces los padres intercambian los genes que quedan en sus extremos. Este operador permite que ningún esquema pueda tener un δ mayor a l/2, disminuyendo así la probabilidad de ruptura de los "buenos" equemas[Δ]". Es por esta razón que se empleó dicho operador en el caso estudio del capítulo 2.

[†] Aclaración. Si se dice que el cromosoma 11101 va a sufrir una ruptura en el locus 3, quiere decir que se separará en los siguientes dos segmentos: 11 y 101.

[¶] Se recomienda al lector que tenga presente esta última frase para la sección ...1.4.4. El teorema fundamental de los AGs...

[Δ] Más adelante en …1.4.3. La teoría estándar de los AGs…, se definirá qué quiere que un esquema sea "bueno" o sea "malo". O, qué quiere decir que un cromosoma tenga una idea buena o una idea mala.

4. Observación: *Los esquemas de menor orden o tienden a conservarse más a través de las generaciones.*

   Se seguirá otra vez el cromosoma y los tres esquemas planteados en la observación anterior. Observe el lector el esquema 3. Si algún gen en el cromosoma llegara a mutar, no pasaría nada si esto se realizara en los loci 1, 2, 3, 6, 7 u 8. En cambio, si ocurriese una mutación en el locus 4 del cromosoma, el esquema 3 desaparecería de él. Por otro lado, el esquema 1 es más susceptible a ser afectado por una mutación por contener más valores fijos, es decir, por ser de un mayor orden o.

   Ahora, en la sección ...1.2. ¿QUÉ SON?... se decía que al diseñar un algoritmo genético normalmente se establece un parámetro p_m, el cual indica la probabilidad de que un gen mute. Por lo tanto, la probabilidad de que el gen de la posición 2 del cromosoma cambie tras una mutación es p_m. Asimismo, la probabilidad de que el gen de el locus 7 mute es p_m. Teniendo en cuenta que estos dos eventos son independientes (es decir, el que haya una mutación en la posición 2 no tiene nada que ver con que ocurra una mutación en la posición 7), entonces la probabilidad de que el esquema 2 se vea afectado por una mutación es $p_m \cdot p_m = p_m^2$. Así, se puede concluir que en general:

   la probabilidad de que un esquema se pierda por mutaciones en el cromosoma que lo contiene, es de $p_m^{\,o}$.‡

En conclusión, *tras la manipulación de cromosomas existe también una manipulación (selección, formación y destrucción) de esquemas, en donde aquellos esquemas cortos y de bajo orden, tienen mayor probabilidad de sobrevivir ante los efectos del entrecruzamiento y la mutación.*

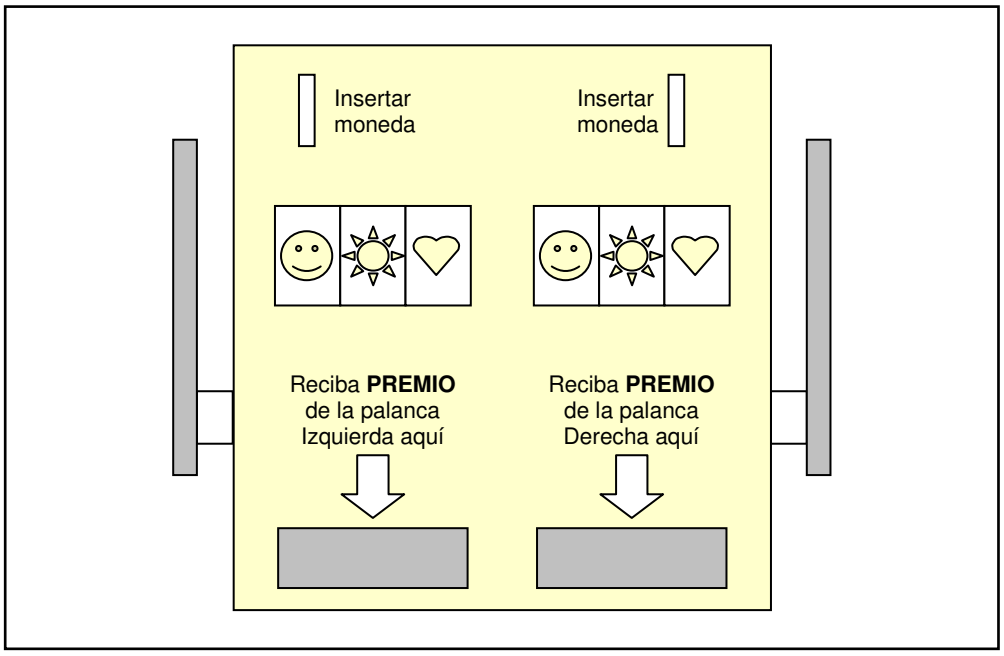
Ahora, antes de continuar con el planteamiento general de la teoría clásica, es necesario entender el siguiente problema análogo, el cual dará luces sobre lo que sucede (con los esquemas) en los AGs.

1.4.2 El problema del tragamonedas de dos palancas. Supóngase que uno se encontrase frente a una curiosa máquina como la de la Figura 19.

Es una máquina similar a los tragamonedas que hay en los casinos pero en realidad, se trata de dos tragamonedas a la vez, una en el lado izquierdo y otra en el lado derecho. Al insertar una moneda en el lado izquierdo y luego jalar la palanca izquierda, existe la posibilidad de obtener el premio en el lado izquierdo. De igual manera, al insertar una moneda en el lado derecho y jalar la palanca derecha, se puede obtener el premio en el lado derecho. Por convención, cada accionar de la palanca se le denomina en este problema un *ensayo* (**trial**).

‡ Una vez más se le pide al lector que tenga en cuenta esta segunda afirmación para la sección ...1.4.4. El teorema fundamental de los AGs...

Figura 19. *El tragamonedas de dos palancas.*



Ahora bien, además de saberse que se trata simplemente de una máquina con dos tragamonedas, existe una segunda peculiaridad: uno de los dos otorga el premio con mayor frecuencia que el otro. En otras palabras, existe una mayor probabilidad de obtener el premio jalando una palanca que la otra. Para ser más concretos, supóngase que uno de las tragamonedas otorga el premio con una probabilidad de 0,7 y el otro con una probabilidad de 0,3. Al primero se le llamará acá el tragamonedas bueno (con la palanca buena) y al segundo, el tragamonedas malo (con la palanca mala).

Finalmente, no se sabe cuál es el tragamonedas que tiene la mayor probabilidad de otorgar el gran premio[†].

La pregunta que se formula en el problema es la siguiente: ¿cómo se debe jugar esta máquina para así maximizar las ganancias? La mejor forma de jugar sería solamente ensayar la máquina buena pero para saber cuál es cuál, hay que jugar las dos. La Figura 19 busca resumir el problema.

Una estrategia[97,98] es dividir el juego en dos partes. En la primera, se jala n veces la palanca izquierda y n veces la palanca derecha. Luego se compara cuál da mejores resultados y se determina entonces cuál es la palanca buena. Así, con esta información

[†] Por su claridad, se siguió aquí el ejemplo expuesto en FALKENAUER. 1998. p. 59. No obstante, la segunda peculiaridad del tragamonedas normalmente se expone de la siguiente manera: se sabe que uno de los tragamonedas otorga premios de diferentes cantidades con un valor esperado μ_1 y una varianza σ_1, mientras que el otro con un μ_2 y σ_2, pero no se sabe a qué tragamonedas corresponde cada una de estas formas de premiar.

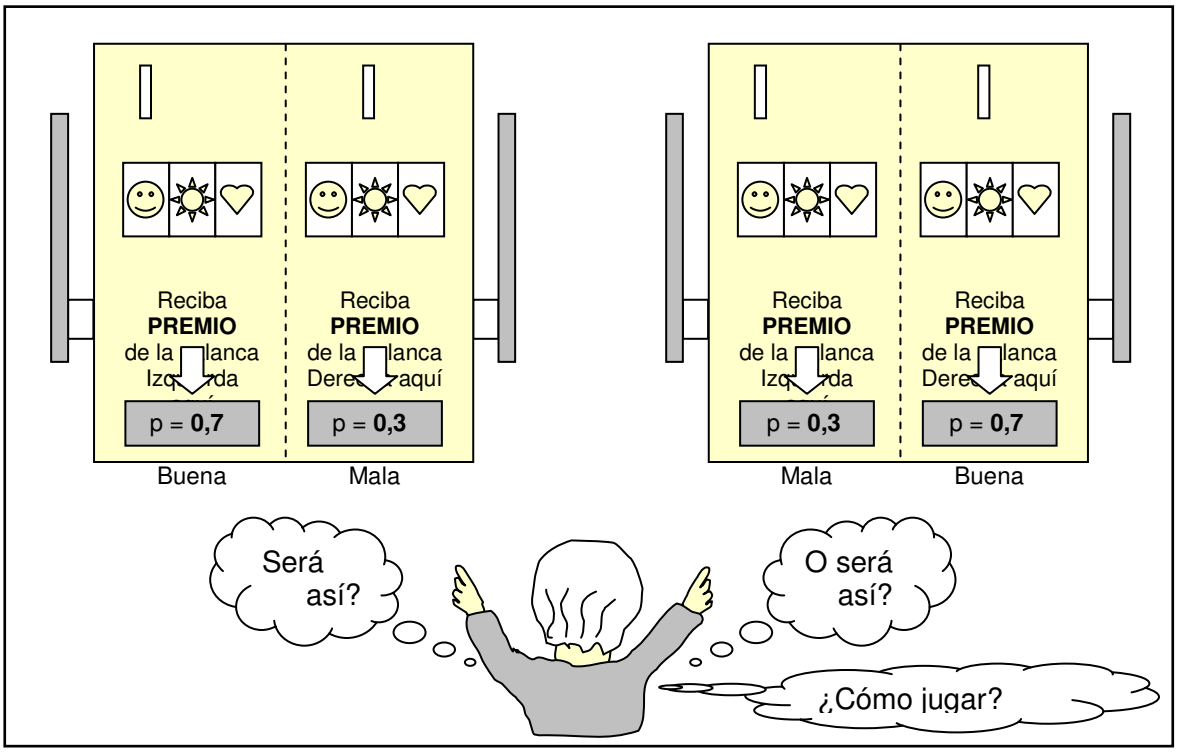
[97] GOLDERG. 1989. p. 36-37.

[98] FALKENAUER. 1998. p. 61-62.

se entra a la segunda parte del juego en donde sólo se jala la palanca buena. A la primera parte se le suele denominar etapa de *exploración* y a la segunda, etapa de *explotación.*

Pero en la anterior estrategia se observa una dicotomía entre las dos etapas. Para la exploración, lo ideal es hacer un número infinito de ensayos con cada palanca para así garantizar la obtención de las probabilidades correctas. Y para la explotación, lo ideal es jalar un número infinito de veces sólo la palanca buena para así maximizar las ganancias. Cada ensayo de la palanca mala en la exploración genera pérdida de dinero (por la moneda que hay que insertar y por el dinero que se deja de ganar). Surge entonces la siguiente pregunta: ¿qué tan larga deberá ser la etapa de exploración?

Figura 20. *El problema del tragamonedas de dos palancas.*



Falkenauer[99] invita aquí a considerar lo siguiente antes de entrar a comprender la respuesta que da Holland:

- La división entre una etapa de exploración y una de explotación no es fácil de determinar, hasta que uno se da uno cuenta que esta división es en realidad artificiosa.
- Efectivamente, en cada ensayo además de poderse recolectar dinero, se está también recolectando información.
- Por lo tanto, debe existir una mejor forma de jugar, aparte de hacer una división entre explorar y explotar. Una forma es jugar ambas palancas, en donde aquella que mejor vaya resultando se juegue más que la otra.

[99] FALKENAUER. 1998. p. 61-62.

Viendo la solución bajo este enfoque, en donde la idea es ensayar más veces la palanca buena que la mala a medida que se va jugando, la respuesta de Holland indicará entonces qué tantas veces más se deberá jugar la buena[Δ].

Tras una rigurosa disertación, Holland responde de la siguiente manera:

- Llámese N al número de ensayos que se van a hacer en total durante el juego.
- Llámese n^* al número de ensayos que se le otorgará a la palanca mala (en donde $n^* << N$).
- Llámese $N - n^*$, al número de ensayos que se le otorgará a la palanca buena.
- El número $N - n^*$ deberá ser igual a

$$N - n^* \cong N \cong \sqrt{8\pi \cdot b^4 \cdot \ln N^2} \cdot e^{n^*/2b^2} \qquad (\textbf{\textit{Eq. 2}})$$

en donde,

b = $\sigma_1/(\mu_1 - \mu_2)$
σ_1 ≡ varianza de la palanca buena
μ_1 ≡ valor esperado de la palanca buena
μ_2 ≡ valor esperado de la palanca mala.

En términos generales, Holland está diciendo que:
el número de ensayos otorgados a la palanca buena deberá crecer como una función exponencial con respecto al número de ensayos otorgados a la mala[100,101].

Continuando con las consideraciones hechas por Falkenauer, la estrategia de juego sería en términos generales una en donde a medida que se juega una y otra palanca, cada vez se otorgarán más ensayos a la buena que a la mala de forma tal que describan una función exponencial[†]. Hay que aclarar sin embargo, que la respuesta de Holland no resuelve el problema directamente[102]. Aquí se ha considerado el enfoque dado por Falkenauer (y probablemente por De Jong[103]). Pero la ecuación Eq. 2 en sí no responde a

---

[Δ] FALKENAUER. 1998. habla de cómo la respuesta de Holland está enmarcada bajo este enfoque de solución. Él parece estar siguiendo la misma línea de De Jong (citada en WILSON. 1996. p. 5). En cambio, otros autores como MACREADY and WOLPERT. 1996. p. 28, 30, aunque concuerdan con que Holland sí indica qué tantas veces ensayar la buena con respecto a la otra, piensan que Holland tiende a mantener el enfoque de dividir el juego en dos etapas.

[100] FALKENAUER. 1998. p. 62.

[101] GOLDBERG. 1989. p. 38.

[†] El lector se podrá estar haciendo las siguientes preguntas: ¿Qué quiere decir esto exactamente en la práctica? ¿Según lo anterior, cuáles son los pasos precisos a seguir? Falkenauer da una muy breve mención al respecto en FALKENAUER. 1998. p. 63. Pareciera como si siguiera la estrategia explicada por De Jong en su obra **An Análisis of the behavior of a class of genetic adaptive systems** (1975) citada por WILSON. 1996. p. 5. Así, será comprendiendo este documento, lo que logrará dar más claridad sobre la estrategia (pero aún así, seguiría siendo un complemento de De Jong a lo que propuso Holland). Para la realización del presente proyecto, no se logró contar con dicha obra. No obstante, saber los detalles de la puesta en práctica de la estrategia no es lo relevante. Lo que sí es importante es conocer la relación exponencial que debe existir entre N y N-n^*.

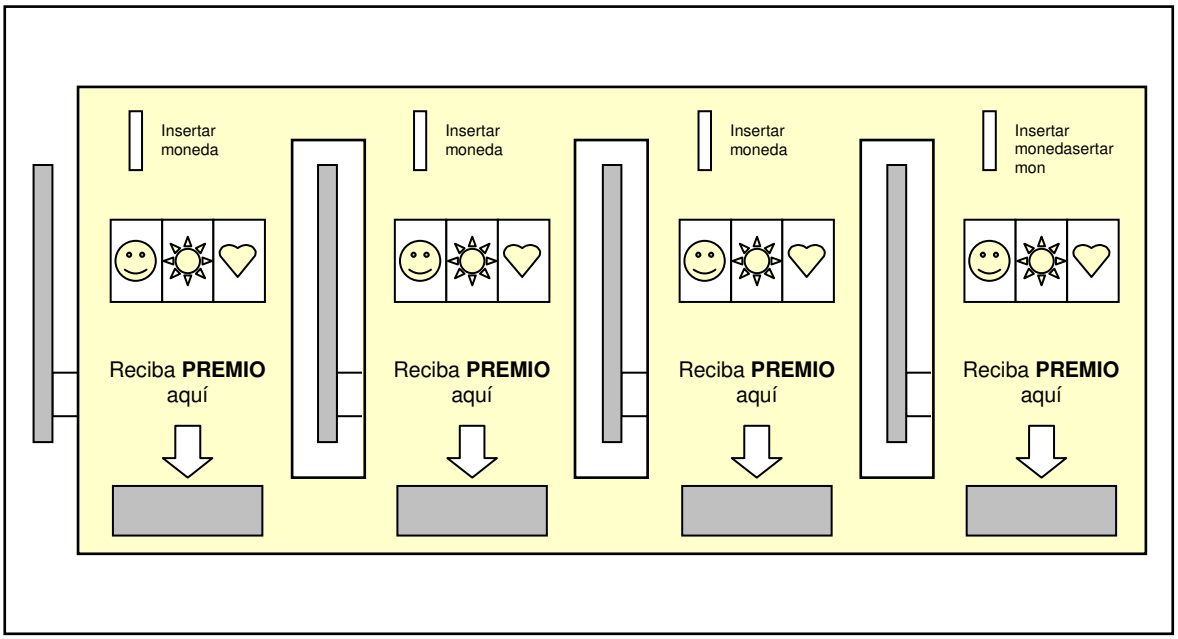
[102] WILSON. 1996. p. 5.

[103] DE JONG. An analysis of the bevahior of a class of genetic adaptive systems. 1975, citado por WILSON. 1996. p. 5.

las siguientes dos cuestiones: (1) cómo determinar cuál es la buena (para así jugarla más veces) y (2) cómo distribuir (precisamente) estos ensayos a lo largo del juego. Autores como MacReady y Wolpert[104], creen que Holland consideraba su respuesta, como algo a implementarse bajo la estrategia de dividir el juego en dos etapas.

La respuesta de Holland es de todas maneras importante pues establece el requisito que debe cumplir toda propuesta de estrategia[105].

Figura 21. *El tragamonedas de k palancas.*



Finalmente, Holland consideró una extensión del problema anterior, el tragamonedas de k palancas (ver Figura 21). Se trata de la misma situación pero con una máquina de varios tragamonedas a la vez.

La solución a la que llega Holland es similar a la del problema anterior: *se le deben otorgar un número de ensayos que aumenten de manera exponencial a las mejores palancas observadas*[106].

Esta conclusión, junto con las observaciones y la conclusión expuesta al final de ...1.5.1. Esquemas... servirán para comprender ahora, la teoría estándar de los AGs.

1.4.3 La teoría estándar de los AGs. Esta teoría busca comprender el funcionamiento del algoritmo genético simple, considerándolo como la base de cualquier otro AG. Consta

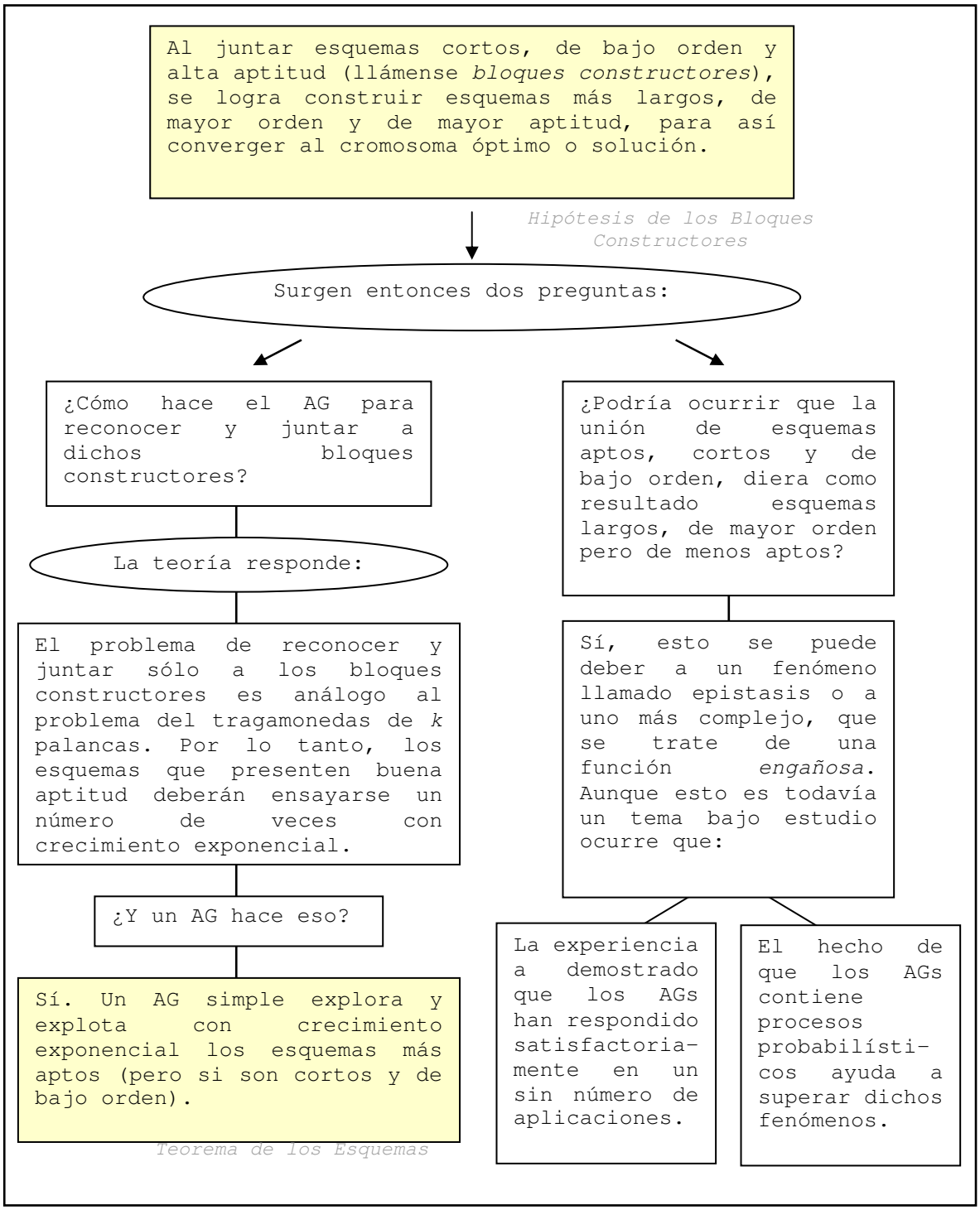
[104] MACREADY and WOLPERT. 1996. p. 28, 30

[105] GOLDBERG. 1989. p. 38.

[106] GOLDERG. 1989. p. 39.

básicamente de dos partes: el *teorema de los esquemas* (**schema theorem**) y la *hipótesis de los bloques constructores* (**building block hypothesis**). La Figura 22 ilustra cómo estas ideas se complementan para poder explicar cómo funciona el AG estándar.

Figura 22. *Teoría estándar de los algoritmos genéticos.*



Ya se vio que tras la manipulación de cromosomas, existe simultáneamente una manipulación de esquemas, bajo la cual esquemas más largos se pueden formar a partir otros más cortos (…1.4.1. Esquemas…, segunda observación). Pues bien, la hipótesis de los bloques constructores, considerando al cromosoma óptimo como un esquema tan largo como l, sugiere que éste se construye tras juntar esquemas más cortos, de bajo orden pero que sean los más aptos.. A este tipo de esquemas se les denomina *bloques constructores* (**building blocks**). Golberg[107] cita la siguiente analogía al respecto: "Tal como un niño crea magníficos fuertes [murallas] a través del arreglo de simples bloques de madera, así también un algoritmo genético busca un desempeño cercanamente óptimo a través de la yuxtaposición de esquemas cortos, de bajo orden y de alto desempeño, es decir, de bloques constructores".

Hasta el momento no se ha dicho precisamente qué quiere decir que un esquema sea más apto que otro. La *aptitud $f(H)$ de un esquema H* se define como el promedio de las aptitudes $f(x_i)$ de los cromosomas x_i que contienen a dicho H en una determinada población[†]. La Tabla 3 presenta un ejemplo en una determinada población, en donde la función objetivo es $f(x) = (1/10) \cdot x^{sen(x/10)}$.

Tabla 3. *Aptitud del esquema ***01100 bajo la función* $f(x) = (1/10) \cdot x^{sen\frac{x}{10}}$

Cromosoma	x	$f(x)$
0 1 0 1 0 0 0 0	80	7,6355
0 1 0 0 1 1 0 0	76	6,6142
1 0 0 1 0 0 0 1	145	10,4870
1 1 0 0 0 0 1 1	195	2,4362
1 1 0 1 1 1 1 1	223	0,0193
0 0 0 0 1 1 0 0	12	1,0135

f(***01100) = (6,6142 + 1,0135) /2 =	7,6277

Surge entonces la primera pregunta: ¿cómo encuentra el AG dichos bloques constructores para así aprovecharlos en la construcción de la solución? La respuesta escrita en palabras de Falkenauer[108] es la siguiente: "queremos ahorrar tiempo computacional [ya no dinero] en buscar prioritariamente los esquemas de bajo orden y alta aptitud [bloques constructores], pero para averiguar cuáles son, tenemos que gastar ese mismo tiempo computacional que queremos ahorrar." Por lo tanto, dicho problema

[107] GOLDBERG. 1989. p. 41.

[†] La literatura reciente registra el concepto de *aptitud estática* (**static fitness**), el cuál se utiliza sobretodo cuando se está analizando el fenómento de las funciones engañosas. La aptitud estática tiene en cuenta las aptitudes de todos los cromosomas que presentan una determinada población. No obstante, este concepto no será relevante para el alcance del presente documento. El lector interesado puede consultar en GREFENSTETTE. Deception considered harmful. 1993, citado por STILLWELL. 2001. p. 26.

[108] FALKENAUER. 1998. p. 68.

donde se plantea la dicotomía de exploración vs. explotación es análogo al problema del tragamonedas de k palancas. La pregunta ya no es *cómo ir ensayando las diferentes palancas para terminar explotando sólo las mejores*, sino a partir de unos cromosomas, *cómo ir seleccionando los diferentes esquemas para terminar explotando (combinando) sólo los más aptos.* La explotación de las mejores palancas garantizará la maximización de las ganancias, mientras que la explotación de los esquemas más aptos garantizará la construcción del cromosoma más óptimo.

Obsérvese por ejemplo la siguiente población.

Número (*Fenotipo*)	Cromosoma (*Genotipo*)
96	0 1 1 0 0 0 0 0
108	0 1 1 0 1 1 0 0
145	1 0 0 1 0 0 0 1
209	1 1 0 1 0 0 0 1
243	1 1 1 1 0 0 1 1
63	0 0 1 1 1 1 1 1
14	0 0 0 0 1 1 1 0
78	0 1 0 0 1 1 1 0

Tras esta población, se esconde una competencia de esquemas (ver Tabla 1).

Tabla 4. *Un juego de k esquemas escondido bajo una población de cromosomas.*

Número (*Fenotipo*)	Cromosoma (*Genotipo*)	Esquema (o *Palanca?*)
96	0 1 1 0 0 0 0 0	* * 1 0 * * * *
108	0 1 1 0 1 1 0 0	
145	1 0 0 1 0 0 0 1	* * 0 1 * * * *
209	1 1 0 1 0 0 0 1	
243	1 1 1 1 0 0 1 1	* * 1 1 * * * *
63	0 0 1 1 1 1 1 1	
14	0 0 0 0 1 1 1 0	* * 0 0 * * * *
78	0 1 0 0 1 1 1 0	

Jalar una vez una de las palancas es sólo una muestra del comportamiento de dicha palanca. Evaluar un cromosoma (evaluarle su aptitud $f(x)$) de un determinado esquema es sólo una muestra de la aptitud de dicho esquema†. Por eso, varias veces tendrán que ser jaladas las diferentes palancas para saber cuál es cuál, y por eso, varios cromosomas por

† Obsérvese por ejemplo que el esquema **00**** involucra a 2·2·2·2·2·2 = 64 cromosomas.

cada esquema tendrán que ser evaluados para saber cuál es cuál. Ahora, para poder concluir firmemente que el problema del tragamonedas de k palancas es análogo al de los esquemas, es necesario finalmente aclarar las siguientes cuestiones:

- *¿Por qué en la Tabla 4, se planteó una población en donde existen exactamente dos cromosomas por esquema?* Simplemente se está describiendo una situación ideal. En la práctica habrán m cromosomas para un esquema y p cromosomas para otro, y habrán otros esquemas que desafortunadamente no tendrán representación alguna.

- *¿Por qué se hace énfasis en que los esquemas aptos sean cortos y de bajo orden? ¿No se podría pensar en que esquemas aptos pero "medianos", no tan cortos como los de la Tabla 4, también se puedan ensamblar en la solución óptima?* En la tercera y cuarta observación de …1.4.1. Esquemas…, se vio que el AG tiende a conservar más a través de las generaciones, aquellos esquemas que sean cortos y de bajo orden. Por lo tanto, la hipótesis de los bloques constructores no espera que pueda haber una manipulación contínua de esquemas medianos pues rápidamente podrán ser destruídos por los efectos de entrecruzamiento y mutación. No obstante, este hecho no es del todo infortunado pues tiende a simplificar el problema. Esto se explica al observarse que un esquema de orden o compite contra 2^o -1 esquemas más. Así se puede apreciar en la Tabla 4, en donde el esquema **00**** de orden 2 sólo tiene que competir contra 2^2 -1 = 3 esquemas. Se trata entonces al caso análogo de sólo tener un tragamonedas de 4 palancas. En cambio, un esquema como **0*0*10 debería competir contra 2^4 -1 = 15, es decir, es como tener un tragamonedas de 16 palancas[Δ].

- *¿Por qué sólo se está considerando (en la Tabla 4) esquemas de la forma **XX****? *¿No existe acaso otra competencia por determinar otros esquemas que permitan saber qué valores deberán ir en los otros genes?* Pues bien, en realidad *un AG está compuesto de varios juegos a la vez*: un juego por cada grupo de esquemas que compiten entre sí. Así, paralelo al juego descrito en la Tabla 4, existen otros como el descrito en la Tabla 5. De acuerdo a la primera observación hecha en …1.4.1. Esquemas…, cada vez que la función objetivo evalúa a un cromosoma, está tomando una muestra de aptitud a 2^l esquemas a la vez. Y así, un algoritmo genético se enfrenta a 2^l juegos en paralelo[¶]. Esta resolución simúltanea (conocida en la teoría como *paralelismo intrínseco*, **intrinsic parallelism**)[±], se mencionará luego en …1.4.6. Críticas a la teoría estándar…. Por

---

[Δ] Aunque esta ventaja es resaltada en FALKENAUER. 1998. p. 67, allí también se advierte sobre el hecho de que un esquema corto representa más cromosomas que un cromosoma largo. No obstante, todo esto lo hace más similar al problema del tragamonedas de k palancas, en donde por cada palanca, existe un número infinito de ensayos que se le pueden realizar.

[¶] En realidad, el AG no logra jugarlos todos de la misma manera. Los efectos del entrecruzamiento y la mutación tienden a dañar los esquemas más largos o de mayor orden (esto se vió en la tercera y cuarta observación de la sección …1.4.1. Esquemas). Un análisis más detenido se puede ver en GOLDBERG. 1989. p. 39-41.

[±] Siguiendo la recomendación hecha en FALKENAUER. 1998. p. 70. se advierte aquí que este término no se debe confundir con *paralelismo implícito* (**implicit parallelism**), el cual consiste, no en la ejecución simultánea de varios juegos, sino en la manipulación simultánea de esquemas. El número esquemas que se manipulan en paralelo ha sido calculado en GOLDBERG. 1989. p. 40.

el momento se asumirá que está bien tomar en análisis, un solo juego para entender el funcionamiento del algoritmo genético.

Tabla 5. *Un AG visto como varios juegos de tragamonedas de k palancas.*

(Aquí k es igual a 8)		Juego 1	Juego 2
Fenotipo	Cromosoma	Esquema	Esquema
96	0 1 1 0 0 0 0 0	* * 1 0 * * * *	* * * * * * 0 0
108	0 1 1 0 1 1 0 0		
145	1 0 0 1 0 0 0 1	* * 0 1 * * * *	* * * * * * 0 1
209	1 1 0 1 0 0 0 1		
243	1 1 1 1 0 0 1 1	* * 1 1 * * * *	* * * * * * 1 1
63	0 0 1 1 1 1 1 1		
14	0 0 0 0 1 1 1 0	* * 0 0 * * * *	* * * * * * 1 0
78	0 1 0 0 1 1 1 0		

¿Qué implica entonces que la analogía se cumpla? La respuesta se expone a continuación en detalle.

Considerando que,

- el problema del tragamonedas de k palancas es análogo al de la exploración y explotación de los mejores esquemas, y que
- Holland demostró que la estrategia a emplear en el problema del tragamonedas es aquella en donde se ensaya con crecimiento exponencial las mejores palancas,

entonces,

- si se llegara a aprobar que un algoritmo genético logra emplear dicha estrategia en su búsqueda y aprovechamiento de los mejores esquemas, se podrá garantizar que el AG es capaz de hacerlo de la manera más óptima.

Pues bien, Holland logró probar lo anterior mediante un teorema, al cual se le conoce como el *teorema de los esquemas* o *el teorema fundamental* (**fundamental theorem**). Este establece que *durante la ejecución de un AG simple, el número de instancias de un esquema apto, corto y de bajo orden, crece de manera exponencial a través de las generaciones*. La siguiente expresión simplificada expresa dicho teorema en términos matemáticos[¶]:

$$m(H,t) \geq m(H,0) \cdot a^t \qquad (\textbf{\textit{Eq. 3}})$$

en donde,

t es la generación que se esté analizando.

[¶] Esta expresión se muestra aquí para que sea claro observar que se trata de una función exponencial. No obstante, la verdadera expresión que se menciona en la literatura se ilustrará luego en …1.4.4. El teorema fundamental de los AGs….

$m(H, t)$ es el número de cromosomas que contienen al esquema H en la generación t. O en otras palabras, es el número de ejemplares del esquema H.

$m(H, t)$ es el número de ejemplares del esquema H al principio del AG.

a es un valor proporcional a la aptitud relativa del esquema $f(H)/\Sigma f$, pero que decrece con la longitud δ y el orden o.

Si la aptitud $f(H)$ es grande, o el o y el δ son pequeños, entonces a será mayor que uno y por lo tanto, el esquema H aumentará su número de instancias en cada generación de manera exponencial. En cambio, si *H tiene bajo desempeño, es largo o de alto orden,* entonces el número de ejemplares del esquema H decrecerá exponencialmente. En realidad ocurre que a no es una constante, pues tanto $f(H)$ como Σf dependen de la población que se tenga en determinada generación. No obstante, se considera que esto en general no impide que se siga describiendo una función exponencial[Δ]. El significado de la variable a, la razón por la cual la expresión Eq. 3 es una desigualdad y la demostración del teorema, se explicarán más adelante en…1.4.4. El teorema fundamental de los AGs….

En resumen, la teoría estándar sugiere que un AG simple aumenta de manera exponencial el número de instancias de los bloques constructores (teorema de los esquemas), hasta construir la solución más óptima (hipótesis de los bloques constructores). El crecimiento exponencial de los bloques constructores durante la ejecución de los AGs es la estrategia más óptima (esto está respaldado por la analogía que existe con el problema del tragamonedas). ¿Pero (siguiendo el lado derecho de la Figura 22), no podrá ocurrir que bloques constructores al unirse formen esquemas más largos, de mayor orden pero de menor desempeño? ¿Y por consiguiente, que la unión de bloques constructores no forme la solución más óptima? Estas incógnitas se analizarán más adelante en …1.4.5. Epistasis, funciones engañosas y otros fenómenos….

1.4.4 El teorema fundamental de los AGs. Antes de pasar a analizar los obstáculos con los que se puede encontrar los AGs, he aquí el teorema de los esquemas con mayor detenimiento.

La versión simplificada del teorema es la siguiente:

$$m(H,t) \geq m(H,0) \cdot a^t \qquad (\textbf{\textit{Eq. 3}})$$

Pues bien, ahora se determinará a qué es igual a, o en otras palabras, cómo es la expresión completa del teorema.

El teorema busca determinar cómo crece el número de ejemplares a través de las generaciones. Por lo tanto, para poder llegar hasta la expresión Eq. 3, la primera pregunta que se debe responder es cómo varía dicho número de una generación a la siguiente.

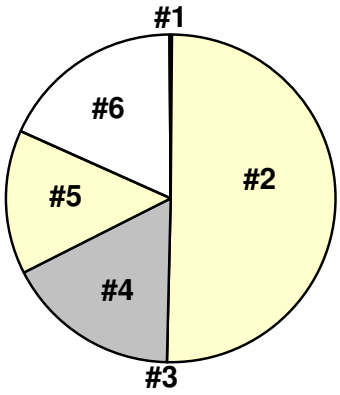
Llámese $m(H, t)$ el número de ejemplares del esquema H en la generación t y $m(H, t+1)$ el número de ejemplares del mismo H en la siguiente generación $t + 1$. Obsérvese que en un

[Δ] En GOLDBERG. 1989. p. 30, se asume también un valor constante llamado c (pero diferente a la variable a que aquí se describe).

algoritmo genético simple, cada cromosoma pasa de una generación a la siguiente tras tres etapas: selección, entrecruzamiento y mutaciones.

Considérese primero la selección. Se recurrirá una vez más a una de las gráficas que se emplearon para explicar este proceso.

#	Cromosoma	x	$f(x)$	Peso en la Ruleta $f(x) / \Sigma f(x)$
1	10111001	14	0,0167	0,0023
2	11000100	215	3,6579	0,4999
3	11101100	196	0,0004	0,0001
4	11010111	196	1,2591	0,1721
5	00010111	23	1,0362	0,1416
6	00001110	196	1,3473	0,1841
		$\Sigma f(x)$	7,3177	



El peso en la ruleta de un cromosoma, es en realidad la probabilidad de que dicho cromosoma sea seleccionado. Por lo tanto, se espera que en n giros de ruleta, un cromosoma sea seleccionado $n \cdot \left[f(x_i) \Big/ \sum_{i=1}^{n} f(x_i) \right]$ veces.

Por consiguiente, si un esquema tiene $m(H, t)$ instancias en la generación t, entonces se espera que para la siguiente generación t+1, tendrá:

$$m(H,t+1) = n \cdot \frac{f(x_1)}{\sum_{i=1}^{n} f(x_i)} + \cdots + n \cdot \frac{f(x_{m(H,t)})}{\sum_{i=1}^{n} f(x_i)}$$

Es decir,

$$m(H,t+1) = n \cdot \frac{f(x_1)+\cdots+f(x_{m(H,t)})}{\sum_{i=1}^{n} f(x_i)} = \frac{f(x_1)+\cdots+f(x_{m(H,t)})}{\overline{f}} = \frac{m(H,t) \cdot \frac{f(x_1)+\cdots+f(x_{m(H,t)})}{m(H,t)}}{\overline{f}}$$

$$m(H,t+1) = m(H,t)\frac{f(H)}{\overline{f}} \qquad (\textbf{\textit{Eq. 4}})$$

Se observa que el valor $m(H, t+1)$ debido al evento de la selección es igual a la cantidad inicial multiplicado por la probabilidad de ser seleccionado. Ahora, autores como Goldberg[109] parten de la sola expresión Eq. 4 para determinar cuál es la variable a que aquí se está buscando para Eq. 3. He aquí cómo lo hace. Asumiendo que la aptitud $f(H)$ se mantuviese siempre igual de superior con respecto a la población (es decir que $f(H)/\overline{f}$ se mantuviese constante a lo largo de las generaciones), luego a partir de la

[109] GOLDBERG. 1989. p. 30.

transición de la generación 0 a la 1, se puede deducir la expresión que muestra cómo los esquemas más aptos crecen exponencialmente, así:.

$$m(H,1) = m(H,0)\cdot\left(\frac{f(H)}{\overline{f}}\right)$$

$$m(H,2) = m(H,1)\cdot\left(\frac{f(H)}{\overline{f}}\right) = m(H,0)\cdot\left(\frac{f(H)}{\overline{f}}\right)\cdot\left(\frac{f(H)}{\overline{f}}\right) = m(H,0)\cdot\left(\frac{f(H)}{\overline{f}}\right)^2$$

$$m(H,3) = m(H,2)\cdot\left(\frac{f(H)}{\overline{f}}\right) = m\left(\frac{f(H)}{\overline{f}}\right)\cdot\left(\frac{f(H)}{\overline{f}}\right)^2\cdot(1+c) = m(H,0)\cdot\left(\frac{f(H)}{\overline{f}}\right)^3$$

…

$$m(H,t) = m(H,0)\cdot\left(\frac{f(H)}{\overline{f}}\right)^t \qquad (\textbf{\textit{Eq.5}})$$

Es decir, la expresión Eq. 5 indica que: si no se consideraran los efectos del entrecruzamiento y de la mutación, se concluiría que el AG simple otorga un crecimiento exponencial a los esquemas más aptos (siendo $a = f(H)/\overline{f}$). La pregunta que sigue es ¿qué pasa cuando el entrecruzamiento entra en juego? ¿Sí seguiran creciendo los esquemas más aptos

Pues bien, una vez un cromosoma es seleccionado, tiene que pasar por el evento de si efectivamente será mandado a entrecruzarse o no, con una probabilidad p_c. Si efectivamente es escogido para el entrecruzamiento, existirá entonces una probabilidad de $\frac{\delta}{l-1}$ de perderse. (Esta probabilidad ya se calculó al final de la tercera observación en …1.4.1. Esquemas…). Por lo tanto‡, un esquema sobrevirá a un entrecruzamiento con una probabilidad de $p_c \cdot \frac{\delta(H)}{l-1}$ y se perderá con una probabilidad de $1-\left(p_c \cdot \frac{\delta(H)}{l-1}\right)$.

Después del entrecruzamiento, ocurren las mutaciones. La probabilidad de que un gen no mute será 1 - $p_m(H)$, y de que ningún gen (de valor fijo) mute en un esquema será (1 - $p_m(H)$)^o. Esto concuerda con lo dicho al final de la cuarta observación en …1.4.1. Esquemas….

Ahora, teniendo en cuenta a los tres eventos (selección, entrecruzamiento y mutación) el número $m(H, t+1)$ de instancias de un esquema será†:

$$m(H,t+1) \geq m(H,t)\cdot\left(\frac{f(H)}{\overline{f}}\right)\cdot\left(1-p_c\frac{\delta(H)}{l-1}\right)\cdot(1-p_m(H))^o \qquad (\textbf{\textit{Eq. 6}})$$

‡ Se está siguiendo aquí la regla que dice que la probabilidad de que dos eventos E y F ocurran a la vez es $P(E\cap F) = P(E)\cdot P(E\,|\,F)$. En este caso $P(E) = p_c$ y $P(E\,|\,F) = \frac{\delta}{l-1}$. (Ver más en GOLDBERG. 1989. p. 321. o en WALPOLE, MYERS y MYERS. 1999. p. 38).

† Se siguió nuevamente la regla de probabilidad indicada en la nota anterior.

Para mejor entender los efectos del entrecruzamiento y la mutación, se suelen hacer las siguientes simplificaciones. Teniendo en cuenta que la probabilidad de mutación es usualmente pequeña entonces el término (1 -$p_m(H)$)° se puede aproximar a $o \cdot p_m(H)$. Multiplicando esta subexpresión con $1 - p_c \frac{\delta}{l-1}$ e ignorando los valores pequeños, se obtiene finalmente la expresión bajo la cual se acostumbra enunciar el teorema de los esquemas:

$$m(H,t+1) \geq m(H,t) \cdot \left(\frac{f(H)}{\overline{f}} \right) \cdot \left(1 - p_c \frac{\delta(H)}{l-1} - o \cdot p_m(H) \right) \qquad (\textbf{\textit{Eq. 7}})$$

El signo de desigualdad aparece ya que el valor de la probabilidad de supervivencia durante el entrecruzamiento es un mínimo. Se observa así, que el teorema de los esquemas tal como se expone en la expresión Eq. 7 establece que los esquemas más aptos, cortos y de bajo orden (es decir, los bloques constructores), tendrán mayor probabilidad de sobrevivir. En cuanto a la pregunta de si crecen exponencialmente, a Eq. 7 se le puede hacer un tratamiento como el que se hizo para obtener la expresión Eq. 5., obteniéndose entonces lo siguiente:

$$m(H,t) \geq m(H,0) \cdot a^t \qquad (\textbf{\textit{Eq. 3}})$$

en donde,

$$a = \left(\frac{f(H)}{\overline{f}} \right) \cdot \left(1 - p_c \frac{\delta(H)}{l-1} - o \cdot p_m(H) \right) \qquad (\textbf{\textit{Eq. 8}})$$

1.4.5 Epistasis, funciones engañosas y otros fenómenos. La hipótesis de los bloques constructores se cumple si[110]:

- Genes que se relacionan tienen sus loci cercanos. Dicho en otras palabras, que aquellos esquemas relevantes para la construcción de la solución (grupos de genes que se relacionan entre sí), tengan longitudes cortas.
- Existe poca interacción entre los genes.

Es fácil intuir por qué al no cumplirse la primera condición se dificultaría la construcción de la solución. La necesidad de la segunda se analiza a continuación.

La hipótesis de los bloques constructores asume tras de sí que cada bloque constructor que conforma al cromosoma más óptimo, contribuye con un determinado nivel de aptitud. Es decir, se asume que en cierto grado, la aptitud de la solución es la sumatoria de las aptitudes de sus bloques constructores. Pues bien, si los genes tienen un gran grado de interacción entre sí, ocurrirá que la contribución de un gen afectaría (aumentando o reduciendo) la contribución de otro gen. Esta interacción entre los genes se ha observado también en la naturaleza y se denomina *epistasis* (epistasis). La naturaleza presenta varios casos.

[110] BEASLEY, BULL and MARTIN. 1993. Parte 1. p. 7.

Por ejemplo, el hombre tiene unos genes que le deben determinar su color de ojos y otros genes que determinan su color de cabello. Es de esperarse que si tiene unos genes para ojos verdes y otros genes para cabello café, dé como resultado un fenotipo de ojos verdes y cabello café. Sin embargo sucede que si llega a tener unos genes para ojos azules, independientemente de los genes para el cabello, cuando nazca el niño, éste siempre tendrá el cabello mono[§]. Existe aquí una interferencia entre el aporte de los genes de los ojos y los del cabello.

Los casos de epistasis en la naturaleza son de varias formas. Otro ejemplo se presenta en los murciélagos. Para ellos, tener unos genes que les permitan emitir un chillido especial es de por sí insignificante para su supervivencia. No obstante, el haber desarrollado (con otros genes) el buen sentido del oído, ha hecho que los primeros no sólo contribuyan a la generación de chillidos, sino que sean claves para crear el sentido de la orientación a través del eco, vital para su supervivencia[¶].

Y así, existen otros casos en donde el efecto de ciertos genes se ve enmascarado, inactivado o estimulado por los alelos (valores) de otros genes. Pasando ahora al contexto de los algoritmos genéticos, Beasley, Bull y Martin[111] definieron formalmente la epistasis, estableciendo tres grados así:

- Nivel 0 – Cero interacción. *Un particular cambio en un gen, siempre produce el mismo cambio en la aptitud del cromosoma (es decir, sin importar qué alelos tengan los otros genes).*
  Aquí simplemente se está haciendo referencia a cuando la epistasis es nula. Un ejemplo se presentaría en la siguiente función trivial:

  $f(x)$ = número de 1s que posee el cromosoma x
  $f(00110) = 2$
  $f(11111) = 5$

  En esta función, la solución óptima (la que generará el mayor valor de $f(x)$) surge como el resultado de la contribución individual de cada gen (o esquema de orden 1) apto. La aptitud del cromosoma es proporcional a la aptitud de cada gen.

- *Nivel 1 – Interacción media. Un particular cambio en un gen siempre produce un cambio en la aptitud aumentándola o disminuyéndola a cero (dependiendo de los valores en otros genes)*
  Este es el grado de epistasis al que usualmente se enfrenta un AG. Sin embargo, según Davis[112], si sólo se tuviese hasta este grado, otros métodos de optimización podrán desempeñarse mejor que el algoritmo genético. He aquí otra función trivial que busca mostrar esta situación:

[§] Este ejemplo fue tomado de FALKENAUER. 1998. p. 28.

[¶] Ejemplo tomado de BEASLEY, BULL and MARTIN. 1993. Parte 1. p. 7.

[111] BEASLEY, BULL and MARTIN. 1993. Parte 2. p. 4.

[112] DAVIS. Bit climbing, representational bias and test suite design. 1991, citado por BEASLEY, BULL and MARTIN. 1993. Parte 1. p. 5.

$$f(x)=\begin{cases} 1 & si \quad todos\ los\ alelos\ son\ igual\ a\ 1 \\ 0 & si \quad algún\ gen\ no\ es\ igual\ a\ 1 \end{cases}$$

$$f(11111)=1$$

$$f(01111)=0$$

En el segundo ejemplo se observa como el primer gen puede causar que toda la aptitud del cromosoma sea nula, cancelando el efecto de los demás genes.

- *Nivel 2 – Epistasis (verdadera). Un particular cambio en un gen produce un cambio en la aptitud, haciéndola variar de magnitud y de signo, dependiendo de los valores en otros genes.*

Al presentarse el fenómeno de la epistasis, puede ocurrir lo que se temía en la Figura 22: "¿Podría ocurrir que la unión de esquemas aptos, cortos y de bajo orden, diera como resultado esquemas largos, de mayor orden pero de menos aptos?" La respuesta es sí, debido a la epistasis. Se requiere entonces que el grado de interacción entre genes no sea alto, para que no dificulte la labor de los AGs. No obstante, son los problemas que contienen epistasis los que verdaderamente interesan resolver, pues si el grado de interacción es muy bajo, entonces otros métodos presentarían mejor desempeño[113,114].

Existe un segundo fenómeno relacionado con el anterior. Es la existencia de los *problemas* o *funciones engañosas* (**deceptive problems**). En estos casos existe epistasis y algo más: la solución contiene esquemas que al no ser evaluados conjuntamente, muestran una aptitud inferior a los esquemas no contenidos en dicha solución. De esta manera ocurrirá que a través de las generaciones, esquemas que no forman parte de la solución aumentarán exponencialmente, eliminando así a los verdaderos bloques constructores. Esto hace del problema difícil para el algoritmo genético. Se dice además que un problema es *completamente engañoso* (**fully deceptive**) si[115] "todos los esquemas de bajo orden que conforman una solución subóptima son mejores que otros esquemas competentes". Bajo una solución subóptima, el AG pareciera estar condenado a no encontrar la solución óptima.

A pesar de que los anteriores dos fenómenos constituyen serios problemas para el buen desempeño de los AGs, la literatura es muchas veces al respecto. Estudios hechos por Grefensette[116] han mostrado que no siempre los problemas engañosos constituyen un problema para los AGs. Otros autores como Beasley, Bull y Martin[117] afirman que los algoritmos genéticos logran desempeñarse satisfactoriamente bajo altos niveles de epistasis. Está además, el sin número de publicaciones que hablan de la efectividad que

[113] DAVIS, L. D. Handbook of Genetic Algorithms. 1991, citado enr BEASLEY, BULL and MARTIN. Part 2. 1993. p. 5.

[114] FALKENAUER. 1998. p. 76.

[115] DEB and GOLDBERG. Analyzing deception in trap functions. 1991, citado en BEASLEY, BULL and MARTIN. Part 2. 1993. p. 5.

[116] GREFENSTETTE. Deception considered harmful. 1993, citado en BEASLEY, BULL and MARTIN. Part 2. 1993. p. 5.

[117] BEASLEY, BULL and MARTIN. Part 2. 1993. p. 5.

tienen los algoritmos genéticos en toda una gama de aplicaciones (parte de estas publicaciones se expondran en …1.5. ¿DÓNDE APLICARLOS?).

El fenómeno de la epistasis ha sido objeto de bastante estudio. Será analizándolo bien que se podrá sugerir alternativas que permitan el buen desarrollo de un AG. Una alternativa es la de no limitarse a la codificación binaria, sino emplear otras formas dependiendo del tipo de problema[118]. Existen por ejemplo, codificaciones en donde los alelos pueden adquirir números reales[119]. Goldberg recomienda sin embargo, que la codificación siempre deberá emplear el más pequeño *alfabeto* posible, entendiéndose el término alfabeto como el conjunto de caracteres que pueden ir dentro de cada gen (así, el alfabeto de la codificación binaria es {0, 1}). Por otro lado, Beasley, Bull y Martin[120] proponen otra forma de codificar denominada *codificación expansiva* (**expansive coding**).

Otra alternativa para reducir la epistasis es modificando el AG en sí (y desarrollando una teoría que sustente su buen funcionamiento)[121]. Estas modificaciones pueden ser simplemente emplear operadores más novedosos como la inversión, el diploidismo, el entrecruzamiento uniforme, etc. Mucas veces el empleo de otros operadores obedece al cambio que se hizo en la codificación. Por ejemplo, Falkenauer[122] propone una codificación propia para problemas de agrupamiento (**grouping problems**), y por ello requirió definir otros operadores, creando así un género de AGs nuevo denominado *algoritmo genético para grupos* (**grouping genetic algorithm**). Están también los *algoritmos genéticos desarreglados* (**messy genetic algorithms**) introducidos por Goldberg, Korb y Deb[123], los cuales proponen otras codificaciones y otros operadores (pero se alejan bastante del AG simple).

Otros fenómenos que obstaculizan o pueden obstaculizar el buen desempeño de los AGs son la *preadaptación*[124] (**preadaptation**), la *deriva genética*[125,126] (**genetic drift**) y la sospecha que existe sobre si la función objetivo mide verdaderamente la aptitud de los cromosomas[127,128].

---

[118] BEASLEY, BULL and MARTIN. Part 2. 1993. p. 5.

[119] DUMITRESCU *et al.* 2000. p. 187.

[120] BEASLEY, BULL and MARTIN. Reducing epistasis in combinatorial problems by expansive coding. 1993, citado por BEASLEY, BULL and MARTIN. Part 1. 1993. p. 6.

[121] BEASLEY, BULL and MARTIN. Part 2. 1993. p. 5.

[122] FALKENAUER. 1998. p. 77, 97.

[123] DUMITRESCU *et al.* 2000. p. 232.

[124] STILLWELL. 2001. p. 27.

[125] BEASELY, BULL and MARTIN. Part 1. p. 8.

[126] DUMITRESCU *et al.* 2000. p. 37.

[127] GOLDBERG. 1989. p. 76.

[128] BEASELY, BULL and MARTIN. Part 1. p. 8.

1.4.6 Críticas a la teoría estándar. La teoría tradicional ha recibido varias críticas sobre su validez. He aquí un breve resumen de algunas de ellas.

- Wolpert y Macready[129] consideran que existen errores matermáticos en la forma como Holland halló la solución al problema del tragamonedas de 2 palancas. También le critican no haber considerado otros tipos de estrategias para resolver dicho problema.

- Grefenstette[130] demostró mediante contraejemplos, que la hipótesis de los bloques constructores no es una explicación válida de cómo el AG converge hacia la solucón óptima.

- Aunque es ventajoso el hecho de que un AG procese en paralelo varios juegos tal como se indica en la Tabla 5 (propiedad conocida como *paralelismo intrínseco*), la teoría estándar asume que dichos juegos son independientes[131]. En realidad esto no es así, y por lo tanto pierde validez la analogía que se ha hecho entre el problema del tragamonedas de k palancas y el de los esquemas.

- El teorema de los esquemas sólo considera selección proporcional o por ruleta. No explica por qué en la práctica, otros métodos de selección han mostrado mejor desempeño[132] (violando así la conclusión que se tenía de que el AG simple explora y explota los bloques constructores de la manera más óptima posible).

Lo anterior pemite concluir que si de verdad se quiere responder a la pregunta planteada al principio del capítulo, "¿FUNCIONAN [los AGs]?", se deberá decidir entre (1) continuar estudiando las fallas que contiene y los problemas que enfrenta la teoria estándar, o (2) desecharla y emplear un enfoque como el de los procesos markovianos. A pesar de las críticas anteriormente expuestas, autores como Falkenauer[133] aún creen que bajo ciertas condiciones, todavía no bien conocidas, los algoritmos genéticos sí siguen la teoría estándar.

1.5. ¿DÓNDE APLICARLOS? *UN ENFOQUE HACIA LA INGENIERÍA CIVIL*

Como se mencionó al inicio de la sección ...1.4. ¿FUNCIONAN?..., los AGs llaman la atención por poder enfrentarse a funciones multimodales, con grandes espacios de búsqueda, con ruido, etc. Afortunadamente para los AGs, este tipo de problemas se presentan muy a menudo en muchas ciencias e ingenierías.

La ingeniería civil es por tanto, candidata a ser beneficiada por esta herramienta. Actualmente ya existe una amplia literatura que da prueba del provecho que se le está dando. Las Tablas 6 al 9 presentan algunos ejemplos:

---

[129] MACREADY and WOLPERT. 1996. p. 26.

[130] GREFENSTETTE. Deception considered harmful. 1993, citado por STILLWELL. 2001. p. 21.

[131] FALKENAUER. 1998. p. 76.

[132] STILLWELL, 2001. p. 17.

[133] FALKENAUER. 1998. p. 59.

Tabla 6. *Aplicaciones de AGs en el diseño de estructuras.*

DISEÑO DE ESTRUCTURAS	
HADI and SCHMIDT. [2000] KOUMOUSIS and ARSENIS. 1998.	Diseño óptimo de miembros en concreto reforzado.
CHEN and RAJAN. [1998]	Los AGs como herramienta automática de diseño estructural.
SOH and YANG. 1998. WOODWARD. 2001.	Diseño óptimo de puentes.
SUDARSHAN. 2000.	Diseño de armaduras.

Tabla 7. *Aplicaciones de AGs en la planeación y ejecución de la construcción.*

PLANEACIÓN Y EJECUCIÓN DE LA CONSTRUCCIÓN	
NASSAR. 2001.	Toma de decisiones para la compra y venta de equipos.
FENG, LIU and BURNS. [1997]	Programación de actividades de construcción.
O'BRIEN and FISHER. 1997. MAHACHI. [2001]	Programación de actividades y asignación de espacios en la planeación de proyectos.
HAIDAR *et al*. 1999.	Selección óptima de equipos y maquinaria para las actividades de construcción.
NATSUAKI *et al*. 1995.	Obtención de la secuencia óptima de actividades en la construcción de puentes.
ROTSHTEIN. 2001	Diagnóstico de las causas de grietas en edificios.

Tabla 8. *Aplicaciones de AGs en la geotecnia y la sismología.*

GEOTECNIA Y SISMOLOGÍA	
JIN and WANG. 2001.	Medición de la rugosidad del terreno y la humedad del suelo.
PACHEPSKY and TIMLIN. 1996.	Estimación de la conductividad hidráulica de un suelo cuando las condiciones de frontera son desconocidas.
NISHIMURA, YONEDA and MORISAWA. 1999.	Diseño óptimo de la localización de muestreos en la evaluación de la contaminación de un suelo.
AURNHAMMER, TÖNNIES and GUERICKE.	Correlación de horizontes a través de fallas.
VAN DITZHUIJZEN. 2001.	Parametrización geológica de un modelo de un reservorio.

Tabla 9. *Aplicaciones de AGs en ingeniería de transporte y pavimentos.*

INGENIERÍA DE TRANSPORTE Y PAVIMENTOS	
CHAN, FWA and HOQUE. 2000. NUNOO. 2001.	Programación del mantenimiento de pavimentos.
CHEU *et al.* 1998.	Calibración de modelos de tránsito.
CHAKROBORTY, DEB and SRINIVAS. 1998.	Programación óptima de tránsito.
JONG, JHA and SCHONFELD. 2000.	Diseño preliminar de vías.

Otro campo para la aplicación de los algoritmos genéticos es la hidroinformática. Aunque se ha asumido en el presente trabajo qué quiere decir esta rama de la ingeniería, vale la pena definirlo aquí. La hidroinformática es el aprovechamiento de las últimas herramientas computacionales (sistemas inteligentes, tecnologías de comunicación, digitalización de imágenes, etc) en el planeamiento y administración del medio acuático y de las estructuras de ingeniería requeridas. La hidroinformática posee varias características así: (1) muchas veces involucra la simulación y análisis a través de modelos computacionales de los sistemas reales, sean naturales o artificiales; (2) requiere de la buena integración de la ingeniería ambiental, civil, de sistemas, electrónica, etc; y (3) analiza información proveniente de áreas externas a la ingeniería como la economía, la ecología y las ciencias sociales.

La literatura registra muchas formas de aplicar los AGs en la hidroinformática. Aunque en ella se aprecia un interés por ampliar cada vez más el alcance de los algoritmos genéticos, casi siempre se aplican como alguna de las siguientes formas:

1. *Calibración de modelos*. La hidroinformática se apoya mucho en la modelación. Cada modelo tiene que ser "ajustado" (o calibrado) para poder ser confiable y así justificar su empleo. Dicha calibración consiste en la búsqueda de ciertos parámetros (valores propios del modelo) que permitan calcular (y simular) los resultados correctamente a partir de la información suministrada. Dicha búsqueda es muchas veces el problema adecuado para ser abarcado por un algoritmo genético. Un ejemplo es precisamente el que se expone en el caso estudio del presente trabajo. (En la sección ...2.1. DESCRIPCIÓN TEÓRICA DEL PROBLEMA... se explicará con detalle la calibración). Otros ejemplos son:
   - Calibración de modelos hidrológicos.
     LIONG, CHAN and SHREERAM. 1995
     EL HARROUNI, OUAZAR and CHENG. 1998
     LIONG, KHU, CHAN. 1999

   - Calibración de redes de tubería mediante la estimación de las rugosidades y detección de fugas:
     VITKOVSKY, SIMPSON and LAMBERT. 2000

2. *Entrenamiento de redes neuronales.* Las redes neuronales son, al igual que los algoritmos genéticos, un tipo de sistema inteligente. Consituyen una alternativa novedosa para modelar sistemas en donde existe bastante incertidumbre sobre su funcionamiento. Estos novedosos modelos también requieren ser calibrados. Así, los AGs también cumplen acá la función de calibrar. Vale la pena anotar que las redes neuronales están teniendo un rápido crecimiento dentro de la hidroinformática, ampliándole también el campo de aplicación a los algoritmos genéticos. He aquí algunos tipos de modelos de redes neuronales donde se estén empleando AGs:
   - Remediación de aguas subterráneas:
     YAN and MINSKER. 2003

   - Tratamiento de aguas residuales industriales:
     CHEN, CHANG and SHIEH. 2001.

3. *Soporte para la toma de decisiones mediante la búsqueda de alternativas.* Cuando se está planeando una actividad (desarrollo de una región, disposición de desechos, etc), un modelo permite simular un posible escenario de acuerdo a las decisiones que se tomen (es decir, a los valores que se le asignen a sus variables). Si los resultados de dichos modelos se logran cuantificar, se pueden entonces integrar con un algoritmo genético que permita proponer la alternativa más óptima o un grupo de alternativas óptimas para una posterior toma de decisiones. He aquí algunos ejemplos:
   - Obtención de alternativas para lograr el desarrollo sostenible de una región:
     CAI, MCKINNEY and LASDON. [2001]

   - Distribución óptima de sistemas de contención contra inundaciones:
     YEH and LABADIE. 1997.

   - Modificaciones de cauces según las necesidades de la zona.
     GOOSENS, VAN DEN BOOGAARD and DOUBEN. 2000.

- Remediación de aguas subterráneas considerando tiempos de operación, número de pozos, tasas de bombeo, etc.
  GÜMRAH *et al*. 2000

- Selección económica y segura de tecnologías para el tratamiento o disposición de contaminantes en cuerpos de agua.
  VÁSQUEZ *et al*. 2000.

- Optimización de embalses para la obtención de energía, suministro de agua y reducción de costos.
  SHARIF and WARDLAW. 1998.

4. *Diseños óptimos.* Este problema es similar al anterior pero aplicado a estructuras. Aquí se busca determinar las propiedades que generen el acueducto, el alcantarillado o la presa más económica (y que cumpla con requisitos de presión, energía, etc). He aquí algunos ejemplos:
   - Diseño de redes de tubería mediante la mejor selección de diámetros:
     LIPPAI, HEANEY and LAGUNA. 1999.
     SOLOMATINE. 1999.

   - Distribución óptima de válvulas en una red para reducir riesgos por fugas.
     REIS, PORTO nad CAHUDHRY. 1997.

   - Optimización (o rehabilitación) de redes mediante cambios de diámetros y adición, remoción, limpieza o alineación de tubos:
     LIPPAI, HEANEY and LAGUNA. 1999.
     SOLOMATINE. 1999.
     ČISTÝ. 2000.
     HALHAL *et al.* 1997.
     MURPHY and SIMPSON. 1992.

   - Diseño de determinados componentes de presas y selección preliminar de alternativas (sitios, tipos de túneles de presión, lugares de extracción de material, tipos de presas, etc). Esta búsqueda es importante en la parte inicial del proyecto, cuando no se cuenta con información completa.
     PARMEE. 1998.

5. *Otras aplicaciones.*
   - Programación óptima de bombeos en sistemas de abastecimiento de agua:
     SCHAETZEN, SAVIC and WALTERS. 1998.

   - Diseño óptimo de toma de aforos en una red, previos a su calibración:
     MEIER and BARKDOLL. 2000.

   - Determinación de redes urbanas de drenaje a partir de información incompleta:
     BLANPAIN *et al*. [1999]

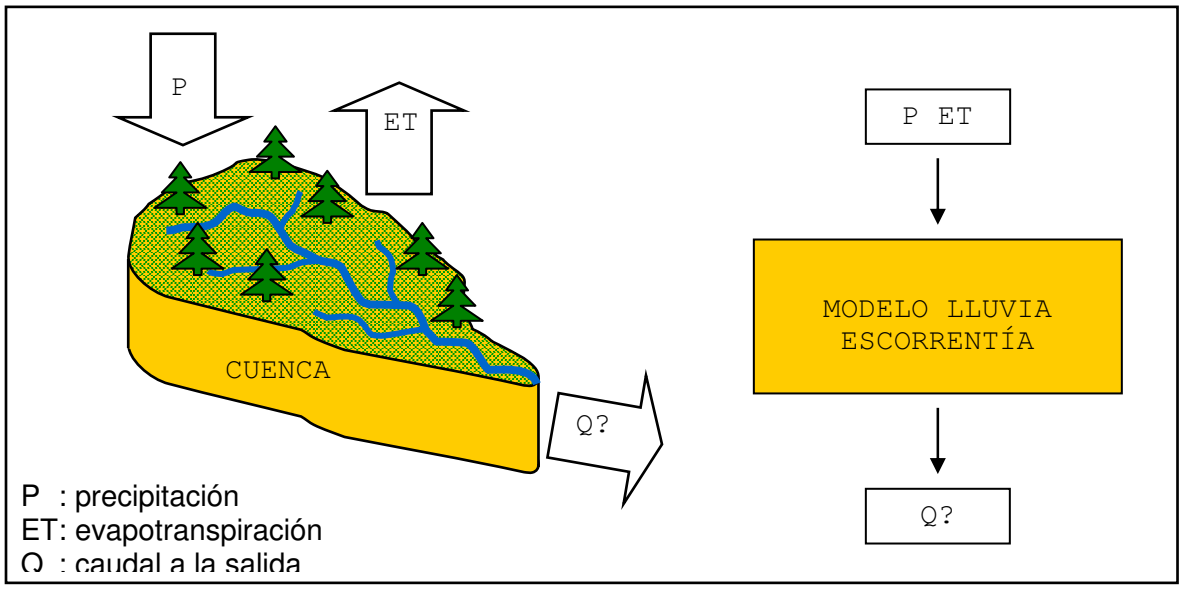
2. CASO ESTUDIO: CALIBRACIÓN DE UN MODELO HIDROLÓGICO

El presente capítulo constituye el componente aplicativo del proyecto. Se pretende dar un testimonio tangible de la posibilidad de aplicar la metodología de los algoritmos genéticos a casos reales en Colombia. Como se verá a continuación, se empleó un AG para lograr la calibración de un modelo lluvia-escorrentía (más precisamente, el modelo de Thomas) aplicado a dos subcuencas ubicadas en la sabana de Bogotá. Esto dio como resultado la elaboración de un programa computacional (el AG en sí) y la obtención de unos resultados que indican si se logró o no la calibración. Estos resultados son luego analizados. No obstante, previo a toda esta exposición, se explicará la teoría que explica el problema que aquí se está atacando: la calibración de modelos hidrológicos.

2.1 DESCRIPCIÓN TEÓRICA DEL PROBLEMA

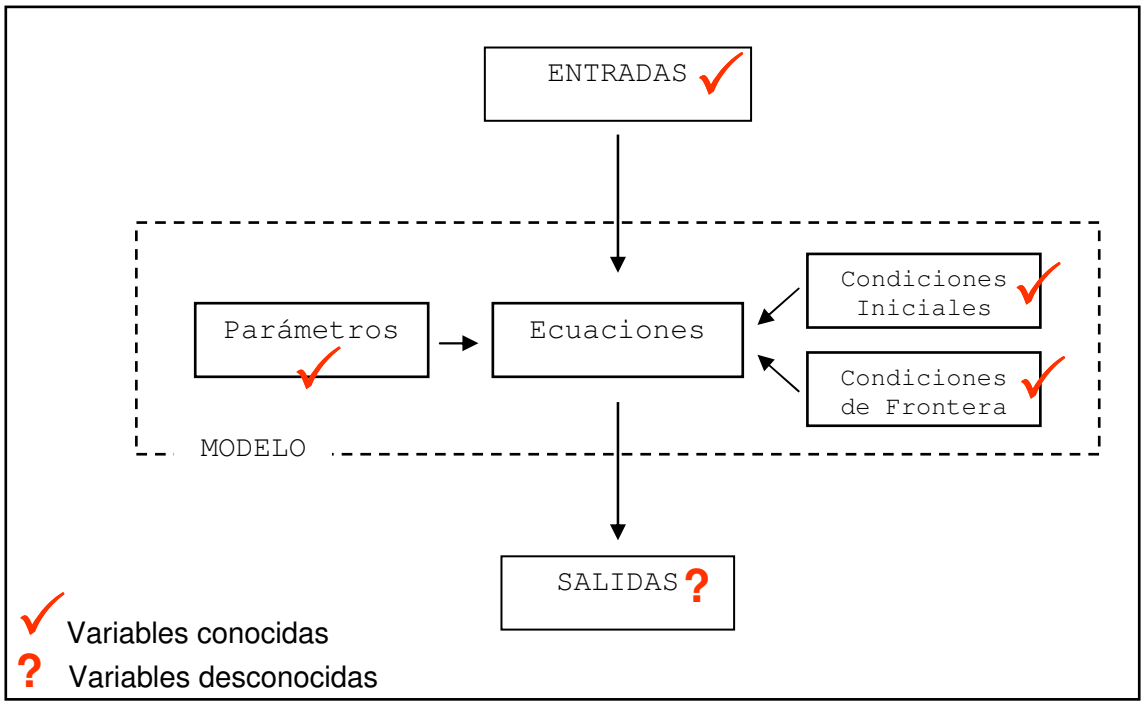
Gran parte de la hidrología está dedicada a la predicción del comportamiento de una cuenca ante los diversos factores. Estos comportamientos se pueden observar en los caudales de sus ríos, en la recarga de sus acuíferos, en la erosión de sus suelos, etc. Y entre estos comportamientos, poder predecir los caudales que salen de sus sistema de drenaje constituye una de los grandes herramientas con que cuenta el hombre para enfrentar problemas tales como abastecimiento de agua, control de inundaciones, dimensionamiento de estructuras hidráulicas, irrigación de cultivos, etc. Esta predicción de caudales se hace, una vez se conoce qué precipitación (y qué evapotranspiración) ha habido sobre la cuenca, a través de modelos lluvia-escorrentía.

Figura 23. *Representación de una cuenca mediante un modelo lluvia-escorrentía.*



La Figura 23 muestra cómo el sistema artificial (llámese modelo lluvia-escorrentía) busca poder simular el sistema natural (llámese cuenca) para así poder predecir el caudal a partir del conocimiento de la precipitación y la evapotranspiración en un determinado período de tiempo.

Figura 24. *Componentes de un modelo hidrológico*[†]*.*



La Figura 24 pemite observar lo que existe en el interior un modelo lluvia escorrentía. Éste estaría compuesto por[134]:

- Parámetros: Representaciones numéricas de las características de la cuenca tales como: geometría, zonas de impermeabilidad, coeficientes de precolación, etc. Según Sorooshian y Gupta[135], estos pueden ser físicos (medibles en campo) o de proceso (no se pueden medir directamente en campo).

- Condiciones Iniciales: Representaciones numéricas de las características de la cuenca. Sin embargo éstas, a diferencia de las representadas por los parámetros, son muy variables en el tiempo, y por lo tanto se requiere conocer únicamente sus valores en el estado previo al ingreso de las entradas (de las precipitaciones).

- Condiciones de Frontera: Representaciones numéricas de la interacción de la cuenca con cuencas vecinas (u otros sistemas vecinos). Muchas veces se asume que la cuenca es totalmente independiente de sus vecinas, así que estos valores

[†] Basado en la Figura 1.6. de SINGH. 1995. p. 7

[134] SINGH. 1995. fig. 1.6. p. 4.

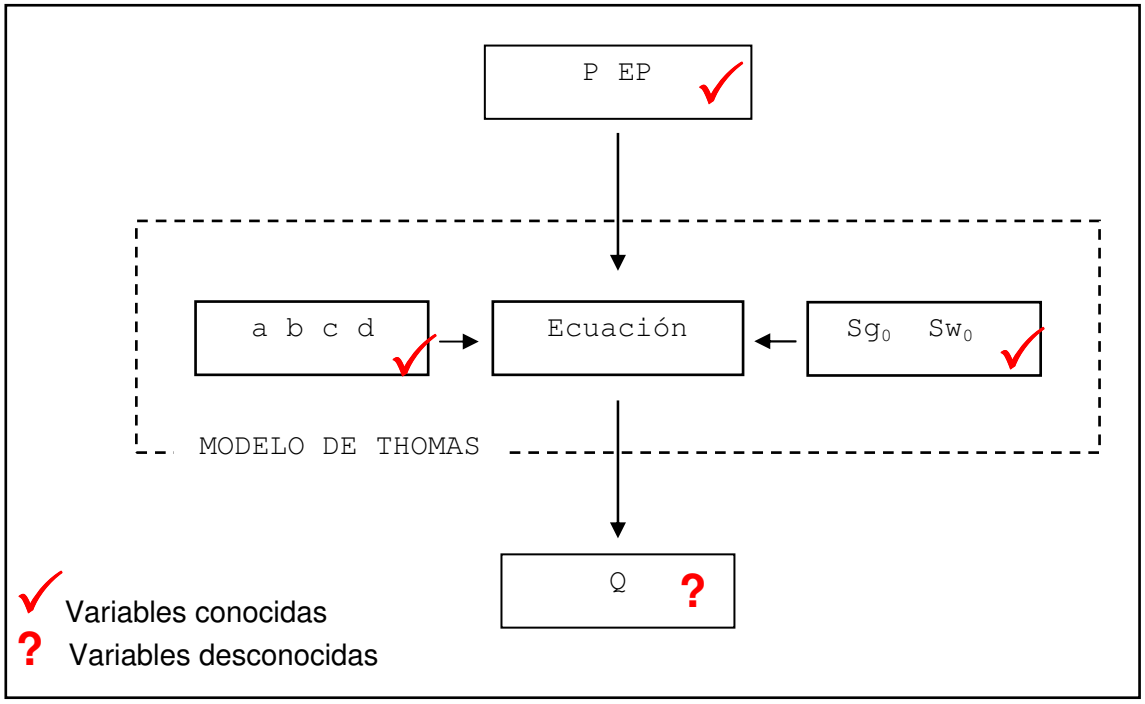
[135] SOROOSHIAN and GUPTA. 1995. p. 24.

no entran en el modelo. Por lo tanto, en lo que resta del presente documento, no se hará más referencia a estos valores.

- Ecuaciones: Expresiones matemáticas que calculan las salidas a partir de las entradas y de los parámetros, condiciones iniciales y condiciones de frontera.

Uno de los modelos lluvia-escorrentía que posee la hidrología es el modelo de Thomas. Este es precisamente el que se empleó para la resolución del caso estudio. En él se definen cuatro parámetros denominados a, b, c, d y dos valores de condición inicial denominados Sw_0 y Sg_0. Dicho modelo se presenta en la Figura 25.

Figura 25. *Variables en la calibración del modelo de Thomas.*



Los parámetros del modelo de Thomas reflejan las siguientes características de la cuenca[†]:

a tendencia de que ocurra escorrentía antes de que el suelo se encuentre completamente saturado.

b límite superior de la suma de la evapotranspiración y el contenido de humedad del suelo.

c fracción de escorrentía proveniente del agua subterránea.

d valor recíproco de el tiempo de residencia del agua subterránea.

Las condiciones iniciales hacen referencia a[‡]:

[†] Definiciones textuales tomadas de ALLEY. 1984. p. 1139.

[‡] Definiciones tomadas de SERRANO. 1997. p. 29.

Sw_0 Contenido inicial de humedad en el suelo
Sg_0 Almacenamiento inicial de agua subterránea

La ecuación que permite calcular (para un período específico) el caudal a partir de la precipitación P, la evapotranspiración ET y las variables a, b, c, d, Sw_0 y Sg_0 es una:

$$Q = \left\{ (1-c) \cdot \left[P + Sw_0 - \frac{P + Sw_0 + b}{2a} \right] + \sqrt{\left(\frac{P + Sw_0 + b}{2a} \right)^2 - \frac{(P + Sw_0) \cdot b}{a}} \right\} + \frac{d}{d+1} \left\{ c \cdot \left[P + Sw_0 - \frac{P + Sw_0 + b}{2a} + \sqrt{\left(\frac{P + Sw_0 + b}{2a} \right)^2 - \frac{(P + Sw_0) \cdot b}{a}} \right] + Sg_0 \right\} \quad (\textbf{\textit{Eq. 9}})$$

No obstante, dicho cálculo se suele discriminar en una serie de pequeñas ecuaciones que van arrojando además otros datos aparte de los caudales. Se sigue entonces la siguiente secuencia hasta calcular finalmente el caudal.

1. Cálculo del agua disponible

$$W = P + Sw_0 \quad (\textbf{\textit{Eq. 10}})$$

2. Cálculo de la variable Y

$$Y = \frac{W + b}{2a} - \sqrt{\left(\frac{W + b}{2a} \right)^2 - \frac{W \cdot b}{a}} \quad (\textbf{\textit{Eq. 11}})$$

3. Cálculo del contenido de humedad del suelo

$$Sw = Y \cdot e^{-EP/b} \quad (\textbf{\textit{Eq. 12}}$$

4. Cálculo de la escorrentía directa

$$Ro = (1-c) \cdot (W - Y) \quad (\textbf{\textit{Eq. 13}})$$

5. Cálculo de la recarga de agua subterránea

$$Rg = c \cdot (W - Y) \quad (\textbf{\textit{Eq. 14}})$$

6. Cálculo del almacenamiento de agua subterránea

$$Sg = \frac{Rg + Sg_0}{d+1} \quad (\textbf{\textit{Eq. 15}})$$

7. Cálculo del caudal subterráneo

$$Qg = d \cdot Sg \quad (\textbf{\textit{Eq. 16}})$$

8. Cálculo del caudal a la salida de la cuenca

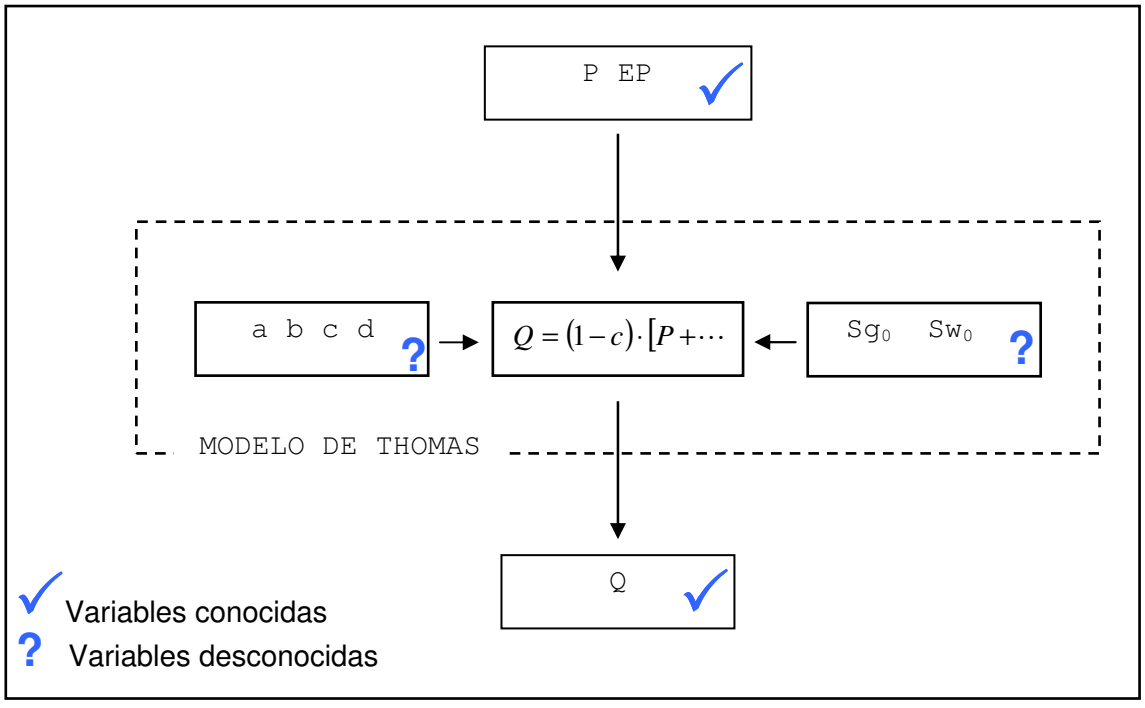
$$Q = Ro + Qg \quad (\textbf{\textit{Eq. 17}})$$

Ahora, para poder alcanzar con éxito la habilidad de predecir caudales se requiere enfrentarse a dos problemas[136]: (1) seleccionar el tipo de modelo que mejor represente el funcionamiento real de la cuenca que se va a analizar, y (2) determinar los parámetros y condiciones iniciales reales de la cuenca. Los resultados que arroje el modelo sólo podrán ser confiables en la medida en que los dos problemas se logren superar adecuadamente.

Por lo tanto, asumiendo que se logró escoger con éxito el modelo (sea por el buen conocimiento de personas expertas, sea por el buen desempeño que ha presentado dicho tipo modelo, etc) se debe garantizar poder conocer bien los parámetros y las condiciones iniciales. La determinación de los parámetros de proceso presentan el problema de no poderse medir directamente en campo. Asimismo, aunque los parámetros físicos aunque sí se pueden medir en campo, requieren de ajustes debido a la posible inexactitud de dichas medidas. El proceso mediante el cual se estiman los parámetros se denomina calibración. Si además no se cuenta con medidas de condiciones iniciales, la calibración podrá incluir también la medición de dichas variables.

Las Figuras 24 y 25 muestran gráficamente cuáles son las variables que se deben conocer para enfrentar el problema de "predecir caudales". Compárese dichas variables con las indicadas en la Figura 26, cuando se trata de enfrentar el problema "calibrar el modelo".

Figura 26. *Variables en el problema de la calibración del modelo de Thomas.*



[136] SOROOSHIAN and GUPTA. 1995. p. 23.

Las entradas (precipitaciones y evapotranspiraciones) y las salidas (caudales) son datos obtenidos de registros históricos. Estos datos se les conoce también como *precipitaciones observadas*, *evapotranspiraciones observadas* y *caudales observados*.

La calibración del modelo (en el caso del de Thomas) se logra cuando se lleguen a establecer los valores de a, b, c, d, Sw_0 y Sg_0 tales que al ingresar la precipitación observada P y la evapotranspiración observada EP, se obtenga un caudal $Qsim$ igual al caudal observado $Qobs$.

Para lograr una buena calibración, no se trabaja con un solo valor de P, de EP y de Q en un período de tiempo específico. Se trabaja con varios valores correspondientes a varios períodos de tiempo. Así, en realidad la calibración se logra cuando se logre reducir a cero algunas de las siguientes funciones (llámense *funciones objetivo*)[§]:

$$F(\theta) = \frac{1}{n} \cdot \sqrt{\sum_{t=1}^{n} \left[Qobs_i - Qsim_i(\theta)\right]^2} \qquad (\textbf{\textit{Eq. 18}})$$

$$F(\theta) = \frac{1}{n} \cdot \sum_{t=1}^{n} \left|Qobs_i - Qsim_i(\theta)\right| \qquad (\textbf{\textit{Eq. 19}})$$

$$F(\theta) = \max\left|Qobs_i - Qsim_i(\theta)\right| \qquad (\textbf{\textit{Eq. 20}})$$

$$F(\theta) = \frac{1}{n} \cdot \sum_{t=1}^{n} \left[Qobs_i - Qsim_i(\theta)\right] \qquad (\textbf{\textit{Eq. 21}})$$

en donde

$Qobs_i$ = caudal medido (obtenido de los archivos)

$Qsim_i(\theta)$ = caudal calculado por el modelo

θ = conjunto (vector) de los parámetros y condiciones iniciales empleados por el modelo

n = número de caudales

En resumen, el problema de la calibración del modelo de Thomas consiste encontrar los valores de a, b, c, d, Sw_0 y Sg_0, tales la función objetivo escogida sea igual a cero. Para el presente estudio se escogió como función objetivo la indicada en Eq. 18, a la cual se le denomina *norma cuadrática normalizada*.

2.2 DESCRIPCIÓN DEL CASO ESTUDIO

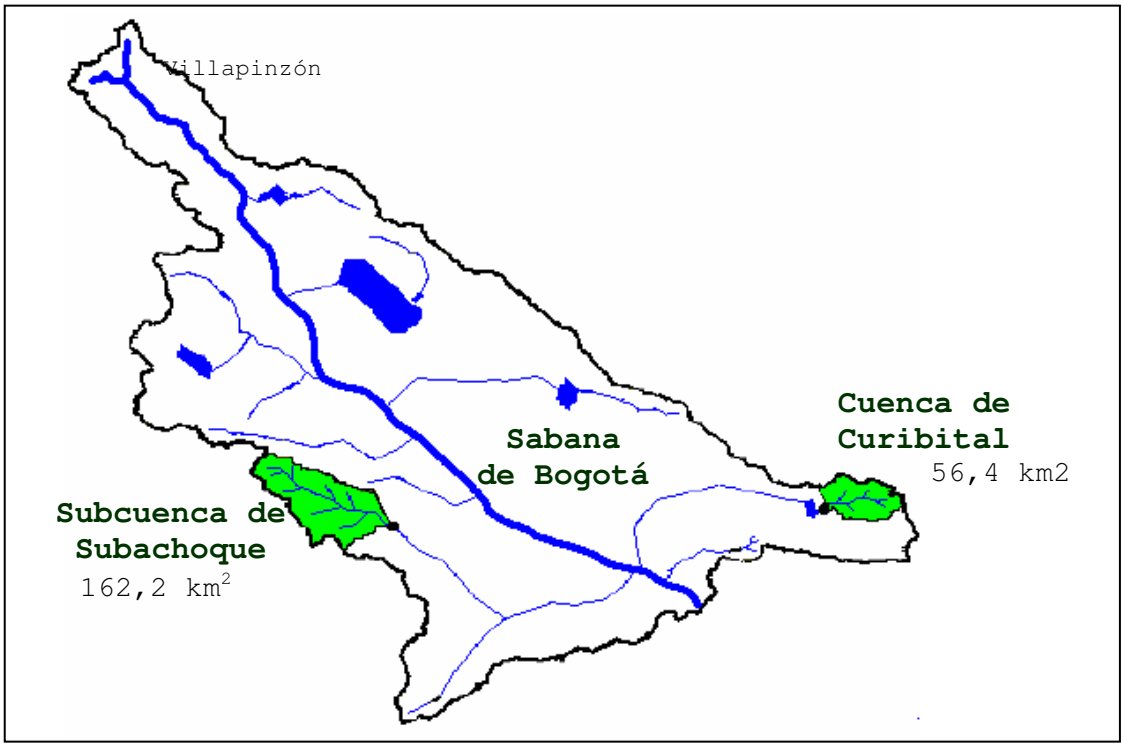
Se requiere calibrar el modelo de Thomas aplicados a dos cuencas: una ubicada en la parte alta del río Subachoque y otra ubicada en la parte alta del río Tunjuelo (cuenca de Curibital). Dichas cuencas se encuentran en la sabana de Bogotá. La Figura 27 describe sus ubicaciones geográficas y las áreas que cubren.

[§] Ver más en SOROOSHIAN, GUPTA and BASTIDAS. 1998. p. 16.

Al final de la sección ...2.1. DESCRIPCIÓN TEÓRICA DEL PROBLEMA... que implica decir que los dos cuencas del estudio se van a calibrar. Ahora la pregunta es cómo resolver el problema.

La solución que se adoptó para el presente caso estudio fue la de emplear un algoritmo genético el cual, como se expuso en el anterior capítulo, tiene la capacidad de resolver problemas de minimización de funciones. Si se observa que la función que se desea aquí minimizar (Eq. 18) es una sumatoria de varias ecuaciones como la Eq. 9, se puede pensar *a priori* que no es ni unimodal ni describe una superficie suave. Además, se está hablando aquí de seis incógnitas a determinar, generando así un espacio de búsqueda muy grande[†]. Así, los algoritmos genéticos con su gran robustez y buena reputación para enfrentar funciones de las que no se tiene buen conocimiento, constituyen una herramienta *ad hoc* para la resolución del presente problema.

Figura 27. *Zona de estudio.*



Se obtuvieron las precipitaciones, evapotranspiraciones y caudales mensuales para los años 1997, 1998 y 1999. A partir de estos datos mensuales, se promediaron entre sí aquellos que pertenecían al mismo mes, obteniéndose así sólo doce valores de cada variable. Estos datos se tomaron de las mediciones hechas por Caro y Flechas[137,†], los cuales se pueden observar en la Tabla 10a y 10b.

[†] Más adelante se observará que este espacio de búsqueda está compuesto por 3,6 x 10^{16} posibles soluciones.

[137] CARO CAMARGO y FLECHAS PARRA. 2002.

2.3 ASPECTOS PREVIOS AL DESARROLLO DEL PROGRAMA

Para la elaboración del programa que aplica algoritmos genéticos para la calibración del modelo, se tuvieron que tener en cuenta varios aspectos importantes. Estos incluyen, la clara definición de la función objetivo a minimizar, la definición del espacio de búsqueda (y así establecer que codificación se adoptará en el AG), y el estilo de software que se requiere desarrollar (qué lenguaje se empleará, qué resultados arrojará, que pasos intermedios se desean mostrar).

Tablas 10a y 10b. *Precipitaciones, evapotranspiraciones y caudales históricos*

a **Subcuenca de Subachoque**

Área 162.2 km²

Datos promedios de los años 1997-1999

mes	P[mm]	ET[mm]	Q[m3/s]	Q[mm]
ene	40.62	49.4	0.26	4.22
feb	27.17	35.94	0.58	9.35
mar	92.96	38.87	0.60	9.64
abr	53.4	55.03	0.74	11.76
may	70.94	47.59	0.58	9.25
jun	51.8	36.45	0.53	8.52
jul	42.85	45.7	0.40	6.31
ago	35.29	35.06	0.34	5.42
sep	77.56	35.92	0.47	7.49
oct	114.7	55.21	1.78	28.46
nov	89.93	55.73	1.11	17.79
dic	48.55	46.54	0.41	6.57
Suma	746	537	8	125

b **Cuenca de Curibital**

Área 56.4 km²

Datos promedios de los años 1997-1999

mes	P[mm]	ET[mm]	Q[m3/s]	Q[mm]
ene	47.78	66.43	0.26	11.86
feb	86.98	65.02	0.53	24.22
mar	76.08	70.87	0.22	10.25
abr	139.15	70.46	1.19	54.83
may	229.01	71.06	2.35	107.86
jun	210.03	76.89	2.19	100.83
jul	243.56	77.01	2.88	132.59
ago	173.34	79.33	2.00	91.69
sep	121.47	71.57	1.01	46.23
oct	146.5	69.77	1.40	64.39
nov	105.42	63.62	0.77	35.53
dic	52.29	67.88	0.47	21.46
Suma	1632	850	15	702

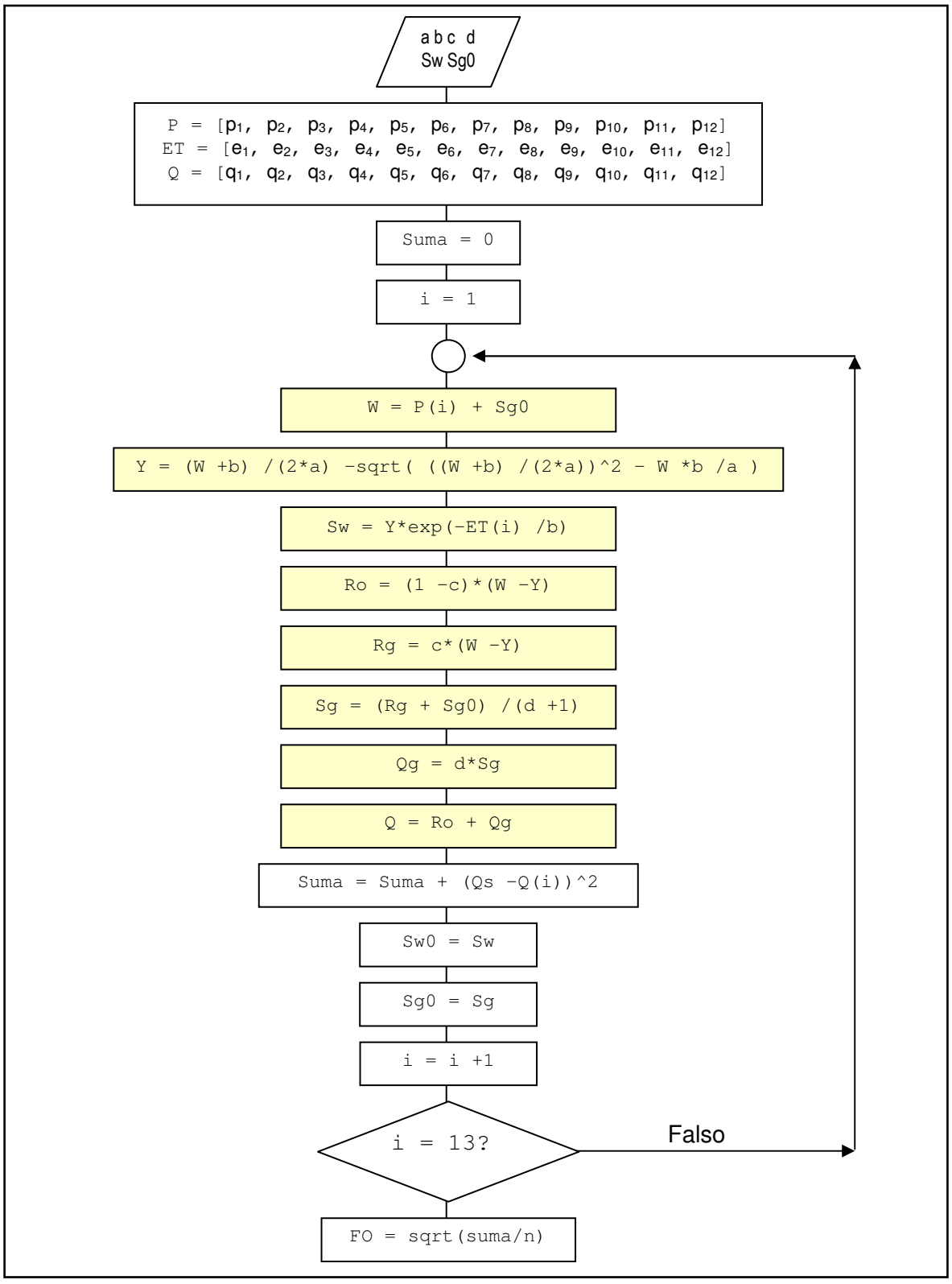
2.3.1. Función objetivo. Se requiere minimizar la función

$$FO = \sqrt{\frac{1}{n} \cdot \sum_{mes=1}^{12} \left(Q_{obs\ en\ el\ mes\ i} - Q_{sim\ para\ el\ mes\ i} \right)^2} \qquad (\textbf{\textit{Eq. 18a}})$$

en donde $Q_{obs\ en\ el\ mes\ i}$ está determinado por la ecuación Eq. 9. Esto involucra entonces un proceso iterativo. Por consiguiente, es pertinente precisar cuál será la función objetivo a minimizar.

† Aunque los caudales simplemente fueron datos que ellos tomaron de la Corporación Autónoma Regional de Cundinamarca, las precipitaciones requieren ser estimadas a partir de datos puntuales, muchas veces no ubicados sobre las cuencas en estudio. Asimismo, las evapotranspiraciones se obtienen indirectamente a partir de otras variables medidas en campo.

Figura 28. La función objetivo vista como una subrutina.



Optando además por calcular los caudales, no directamente mediante Eq. 9, sino a través de las diferentes expresiones Eq. 10 a Eq. 17, la función objetivo quedó finalmente organizada como se indica en la Figura 28. El programa computacional deberá entonces contener una subrutina que siga los pasos indicados en dicha figura.

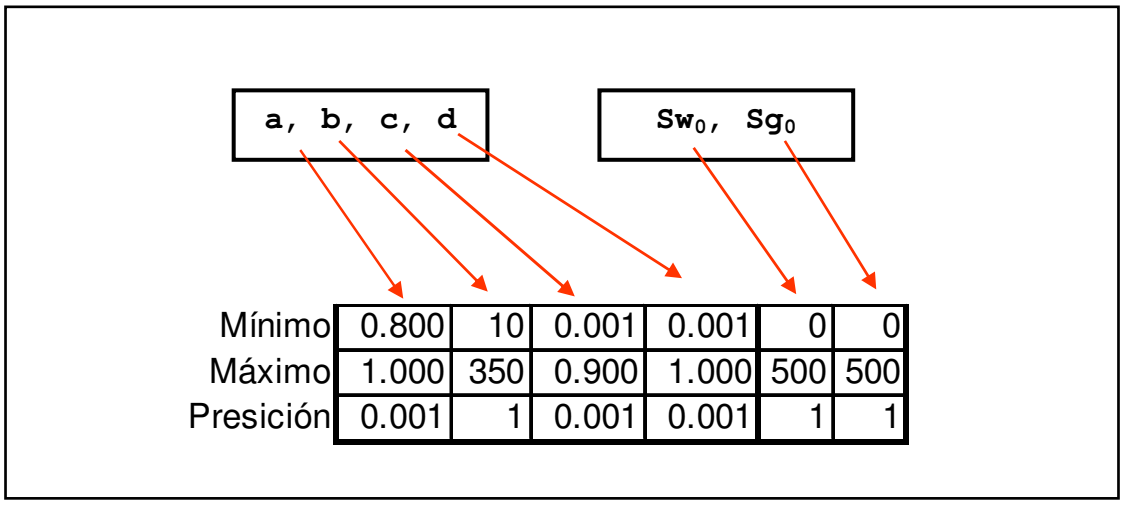
El cálculo de la función objetivo comienza por obtener externamente a, b, c, d, Sw_0 y Sg_0. La función tiene dentro de sí, los valores de las precipitaciones, las evapotranspiraciones y los caudales. Como estos datos no los pide la función externamente, se deberá emplear dos subrutinas diferentes: una para calibrar la subcuenca de Subachoque con los datos de la Tabla 10a y otra para calibrar la cuenca de Curibital con los datos de la Tabla 10b. En amarillo se indica los pasos para calcular cada caudal (y de paso, cada Sw y cada Sg).

Es importante observar que, a pesar de todos los cálculos que involucra la función objetivo, la Figura 28 muestra claramente que cada vez que se llama a esta función, ingresan cinco valores a, b, c, d, Sw_0 y Sg_0 y sólo sale uno: FO (que es el mismo valor de la ecuación Eq. 18a).

2.3.2. Espacio de búsqueda y su codificación.

Trabajos anteriores como el de Alley[138], sugieren rangos entre los cuales pueden variar los parámetros. Ampliando dichos rangos, se determinó que los espacios de búsqueda para cada variable (y las precisiones requeridas) son los indicados en la Figura 29.

Figura 29. *Rangos de búsqueda y precisión para cada unos de los parámetros.*



	a	b	c	d	Sw_0	Sg_0
Mínimo	0.800	10	0.001	0.001	0	0
Máximo	1.000	350	0.900	1.000	500	500
Presición	0.001	1	0.001	0.001	1	1

Sabiendo ya cuál es el espacio de búsqueda, se procede a diseñar los cromosomas. Siguiendo los pasos indicados en ...1.2.1. La codificación... se decide entonces que se empleará un código binario. La longitud de cada subcadena dentro del cromosoma se determina mediante la ecuación Eq. 1. Las longitudes resultantes son las siguientes:

[138] ALLEY. 1984. p. 1142. Tabla 2.

Para a:

$$L = \frac{\ln\left(\frac{1{,}000 - 0{,}800}{0{,}001} - 1\right)}{\ln(2)} \approx 8$$

Para b:

$$L = \frac{\ln\left(\frac{350 - 1}{10} - 1\right)}{\ln(2)} \approx 9$$

Para c:

$$L = \frac{\ln\left(\frac{0{,}900 - 0{,}001}{0{,}001} - 1\right)}{\ln(2)} \approx 10$$

Para d:

$$L = \frac{\ln\left(\frac{1{,}000 - 0{,}001}{0{,}001} - 1\right)}{\ln(2)} \approx 10$$

Para Sw_0:

$$L = \frac{\ln\left(\frac{500 - 0}{1} - 1\right)}{\ln(2)} \approx 9$$

Para Sg_0:

$$L = \frac{\ln\left(\frac{500 - 0}{1} - 1\right)}{\ln(2)} \approx 9$$

Otro aspecto a tener en cuenta es el grado de sensibilidad de los parámetros. El trabajo de Caro y Flechas[139] sugiere que los parámetros organizados de mayor a menor sensibilidad son a, d, c y b. Esto influye en el diseño del cromosoma como se explicará a continuación.

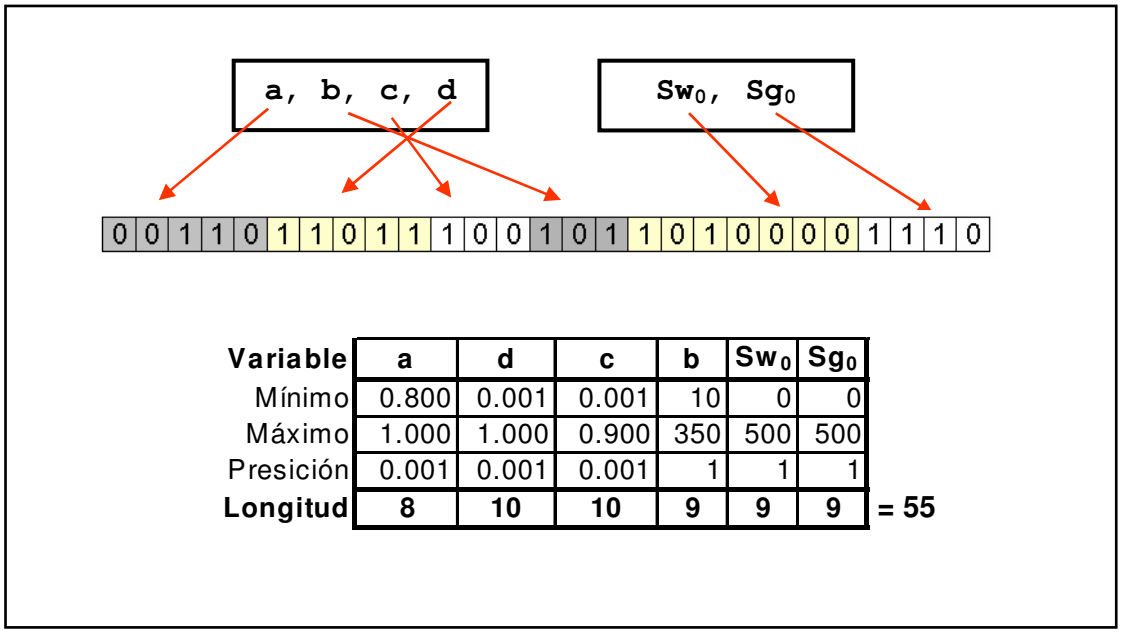
Se sabe que el operador de entrecruzamiento tiende a romper con mayor probabilidad aquellos esquemas de gran longitud δ. Por esta razón, se optó por aplicar el operador de entrecruzamiento de doble punto. Por otro lado, entre más sensible sea un parámetro más tiende a influir en la función objetivo. Esto hace pensar que los esquemas que contienen posiciones fijas sobre estas subcadenas tendrán mayor influencia sobre la función objetivo, es decir, serán los más importantes. Si se colocan las variables más sensibles distanciadas entre sí, los esquemas más importantes tendrán mayores longitudes δ y en consecuencia, tenderán a perderse más fácilmente tras el entrecruzamiento. Por lo tanto, se optó por ubicar juntos dentro del cromosoma a los parámetros más sensibles[¶]. El diseño final de los cromosomas se indica en la Figura 30.

Obsérvese ahora un dato interesante. Si cada cromosoma está compuesto por 55 genes, entonces quiere decir que el espacio de búsqueda es de 2^{55} = 3,6 x 10^{16} posibles soluciones. Esto constituye una razón más por la que se empleó la metodología de los AGs.

[139] CARO CAMARGO y FLECHAS PARRA. 2002.

[¶] Este análisis concuerda con la discusión que se hizo en …1.4.5. Epistasis, funciones engañosas y otros fenómenos…, en donde se dice al principio que para que la hipótesis de los bloques constructores sea válida, se requiere que los esquemas importantes sean cortos. De lo contrario, se puede dificultar la labor del AG.

Figura 30. *Conformación del tipo de cromosoma que se empleará.*



Variable	a	d	c	b	Sw_0	Sg_0	
Mínimo	0.800	0.001	0.001	10	0	0	
Máximo	1.000	1.000	0.900	350	500	500	
Presición	0.001	0.001	0.001	1	1	1	
Longitud	**8**	**10**	**10**	**9**	**9**	**9**	**= 55**

2.3.3. El estilo de programa requerido. Existen muchas formas de hacer un programa computacional. Pues bien, se optó por que el programa cumpla con los siguientes requisitos:

- Que sea escrito en un lenguaje de programación conocido y actual.

- Que prime la legibilidad sobre la eficacia y los novedosos entornos gráficos. Como se mencionó en los objetivos del presente proyecto, se desea acá dotar al lector con herramientas que le permitan implementar en un futuro los algoritmos genéticos. La idea es que el código de programación sea fácil de comprender y que las subrutinas que involucre sean fáciles de asociar entre sí. Si se optase por la línea de la eficacia, se recurriría a líneas de código que permiten procesos más rápidos pero que no son fáciles de comprender. Por otro lado, los novedosos entornos gráficos (con los que vienen muchos lenguajes de programación orientados a objetos) no permiten una clara exposición del programa, ya que no hay un orden secuencial de los comandos.

- Que permita hacer un seguimiento de cómo se desarrolla el AG, y que todos los resultados queden registrados en archivos para un futuro análisis.

- Que sea claro al pedir los datos del usuario.

En la siguiente sección se observará como se escribió el programa de tal forma que cumpliese con los anteriores requisitos.

2.4 DESCRIPCIÓN DEL PROGRAMA

Para la resolución del problema de la calibración de las dos cuencas, se diseñó un programa en lenguaje de MATLAB R12. La programación tuvo un enfoque estructural (y no orientado a objetos) lo que permite exponer todo su código de manera secuencial en el Anexo A. El programa, además de arrojar al final una solución que lograse la minimización de la función objetivo, registra en archivos los resultados de las diferentes generaciones así como estadísticas que permiten verificar todo el proceso.

El programa está compuesto por una subrutina principal llamada AG8, el cual hace llamadas a otras subrutinas. Se siguió prácticamente los pasos indicados en la Figura 5. Es decir, se sigue un algoritmo simple pero con las siguientes características especiales:

- La selección de los cromosomas más aptos se hizo mediante normalización[†].
- Los entrecruzamientos empleados son entrecruzamientos de dos puntos[§].
- Se aplicó elitismo tal como se describió en ...1.2.5.3. La muerte y la supervivencia....

La Figura 31 indica los mismos pasos de la Figura 5 pero pero indicando en cuadros amarillos los nombres de las subrutinas que ejecutaban dichos pasos. Los cuadros en azul simplemente indican pasos intermedios en los cuales se reportan en archivo o en pantalla, los resultados que se llevan hasta el momento.

El programa inicia con la subrutina `obtenerParametros`, el cual le solicita al usuario los siguientes valores:

- TPOB: Tamaño de la población.
- MAXGEN: Número de generaciones que realizará el AG.
- PX: Probabilidad de entrecruzamiento p_c.
- PMUT: Probabilidad de mutación p_m[Δ].
- LCAD: Longitud de cada una de las subcadenas que conformarán al cromosoma
- MINVAL Mínimo valor que puede ser representado por la cadena.
- MAXVAL Máximo valor que puede ser representado por la cadena.

Los tres últimos parámetros corresponden a los indicados en la Figura 30 como “longitud”, “mínimo” y “máximo”. En la subrutina `obtenerOtros` se le solicita al usuario que especifique cómo se llamarán los archivos donde se registrarán los resultados y que

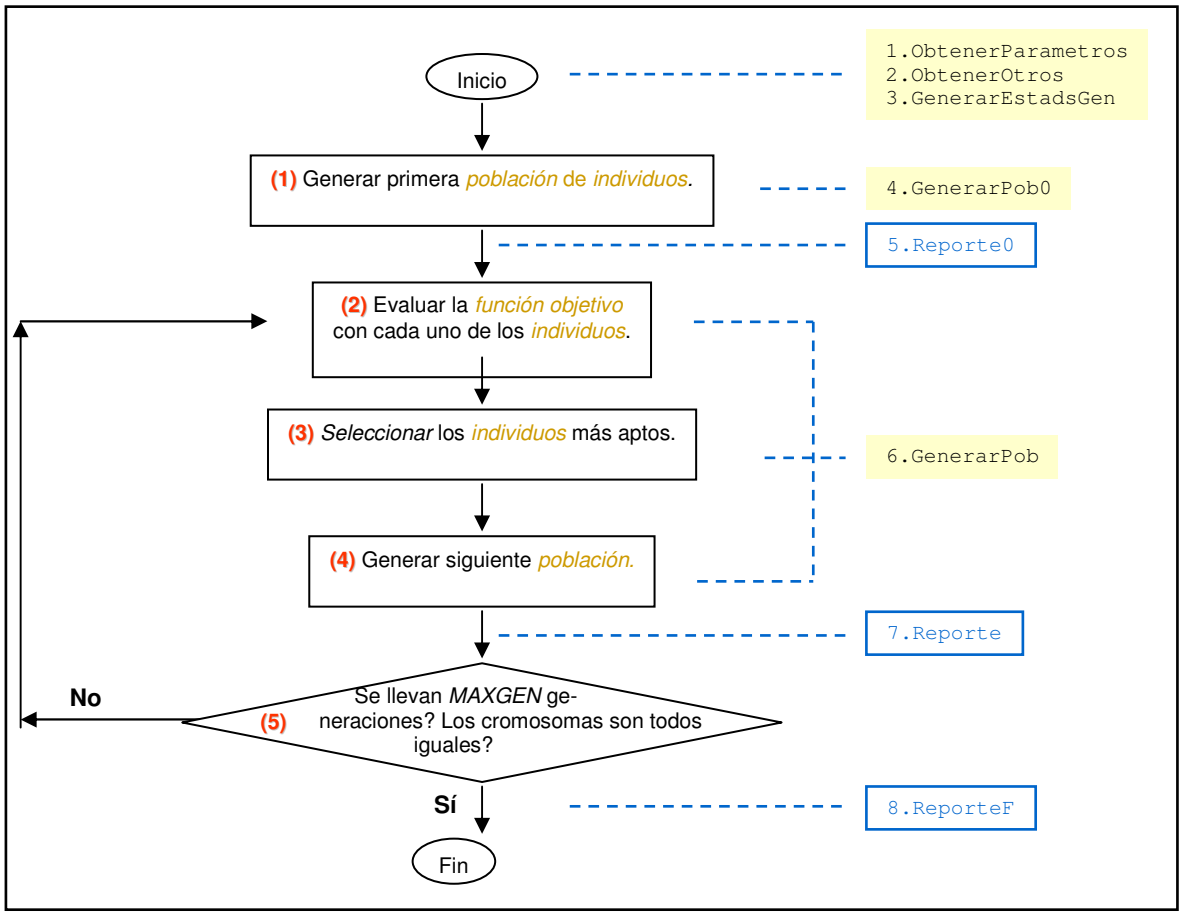
[†] Este método de selección se explicó en …1.2.5.2. La selección…

[§] Este operador se explicó al final de la tercera observación mencionada en …1.4.1. Esquemas…. También se ilustra en la Figura 15.

[Δ] Los parámetros p_c y p_m se explicaron en …1.2.2. El algoritmo genético simple….

indique él mismo la semilla que empleará el AG para la generación de números seudo aleatorios[†].

***Figura 31**. Diagrama de flujo del programa (AG8)*



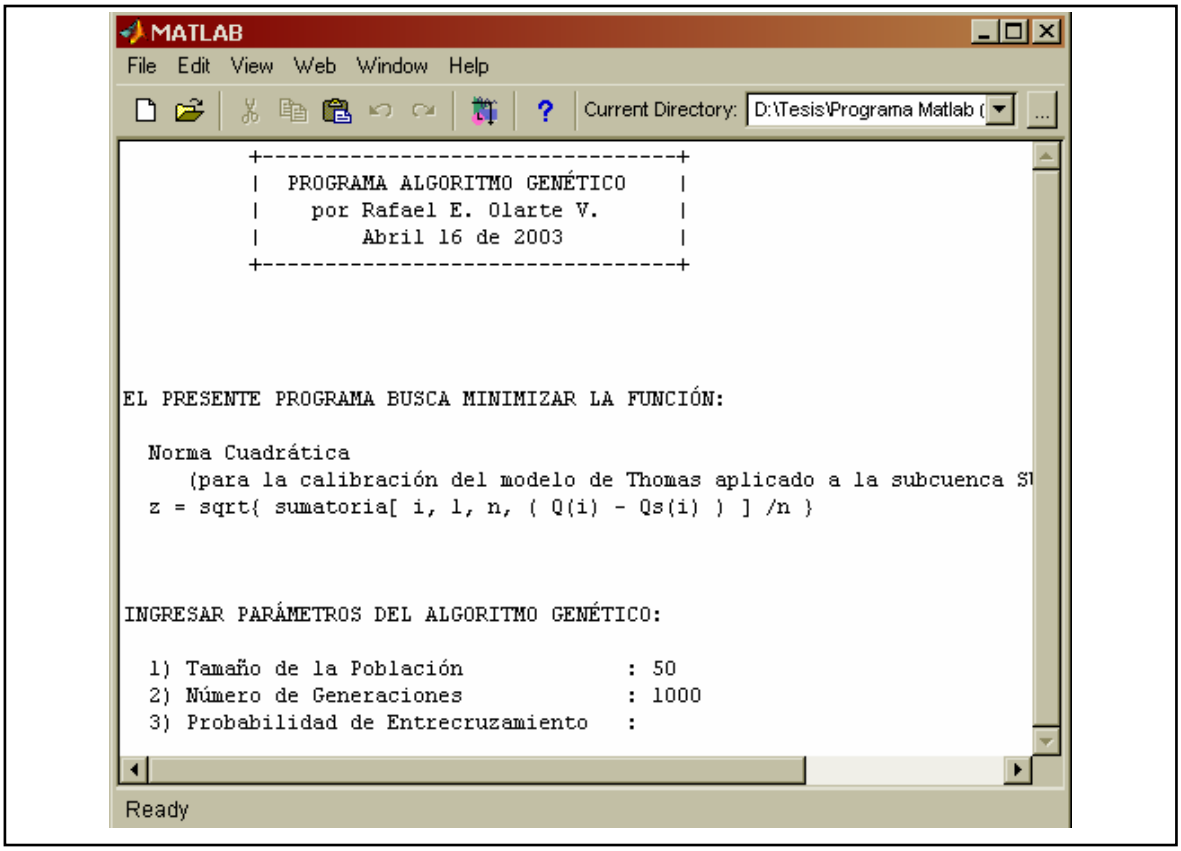
El AG finaliza cuando se haya llegado al número de generaciones indicado en MAXGEN o cuando todos los individuos en una generación sean iguales (es decir, cuando se convege a una sola solución). La Figura 32 muestra cómo es la pantalla inicial del programa.

La función `GenerarPob` realiza los pasos 2, 3 y 4. Aunque no se observa en la Figura 31, cuando en el paso 2 se requiere evaluar a la función objetivo, se llama a una función denominada `FunObj`, la cual sigue lo indicado en la Figura 28[¶].

[†] A pesar de que el AG involucra procesos aleatorios, el permitirle al usuario indicar la semilla hará que se puedan generar resultados reproducibles.

[¶] El lector notará sin embargo dos pequeñas diferencias entre lo indicado en la Figura 28 y el código de la subrutina FunObj. Primero, no se siguió un ciclo tipo "while" sino un ciclo tipo "for". Segundo, aunque los caudales observados si están escritos explícitamente en la función, primero hay que cambiar sus unidades de m^3/s a *mm* mensuales, recurriendo al valor del área de la cuenca. Esto se hizo así, para estandarizar la forma como se obtuvieron dichos valores y poder luego, comparar de manera precisa los resultados con el trabajo de CARO CAMARGO y FLECHAS PARRA. 2002.

***Figura 32**. Pantalla inicial del programa computacional.*



Otros aspectos característicos del programa, se pueden apreciar claramente en el mismo código que se expone en el Anexo A, ya que éste se escribió con suficientes comentarios que lo hacen autoexplicativo.

2.5 EJECUCIÓN

Se corrió el programa cinco veces para cada cuenca (es decir 10 ejecuciones en total). Esto es con el fin de ensayar diferentes semillas. Siempre se ingresaron los siguientes parámetros.

- TPOB = 50
- MAXGEN = 1000
- PX = 0.8
- PMUT = 0.01
- LCAD = [8 9 10 10 9 9]
- MAXVAL = [1 350 0.9 1 500 500]
- MINVAL = [0.8 10 0.001 0.001 0 0]

Cada vez que el programa era ejecutado, se creaba un archivo para cada generación, es decir 1000 archivos. Debido a que creación de archivos reducía la velocidad de ejecución, simplemente se desactivaba la subrutina `reporte`.

2.6 RESULTADOS

En cuanto a la subcuenca de Subachoque, los mejores resultados se obtuvieron con un valor de semilla igual a 49. En el Anexo B, se encuentran impresos los archivos con la primera y última generación para dicha semilla. Los mejores resultados para la cuenca de Curibital se obtuvieron con la semilla de valor 45. Estos no fueron impresos en aras de reducir el tamaño del presente trabajo.

El programa durante su proceso de búsqueda cada vez convergía a una solución más óptima. No obstante, como se puede observar en las Figuras 33 y 34, después de la generación 200 dicha optimización es mínima en comparación con las primeras generaciones. Se puede observar también en dichas figuras, los valores finales en la función objetivo.

Figura 33. *Desarrollo de la búsqueda del AG en la cuenca Curibital.*

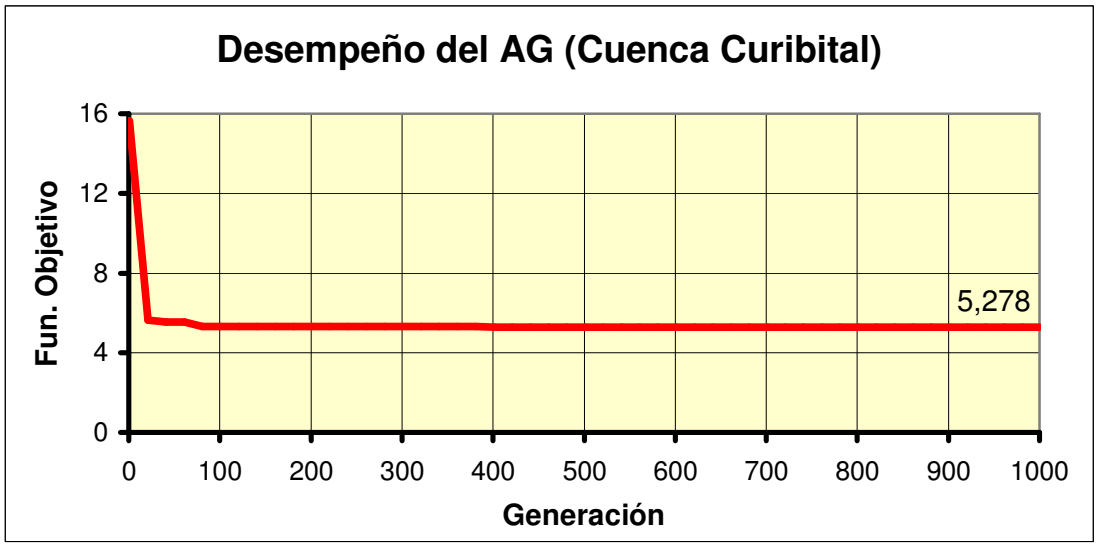
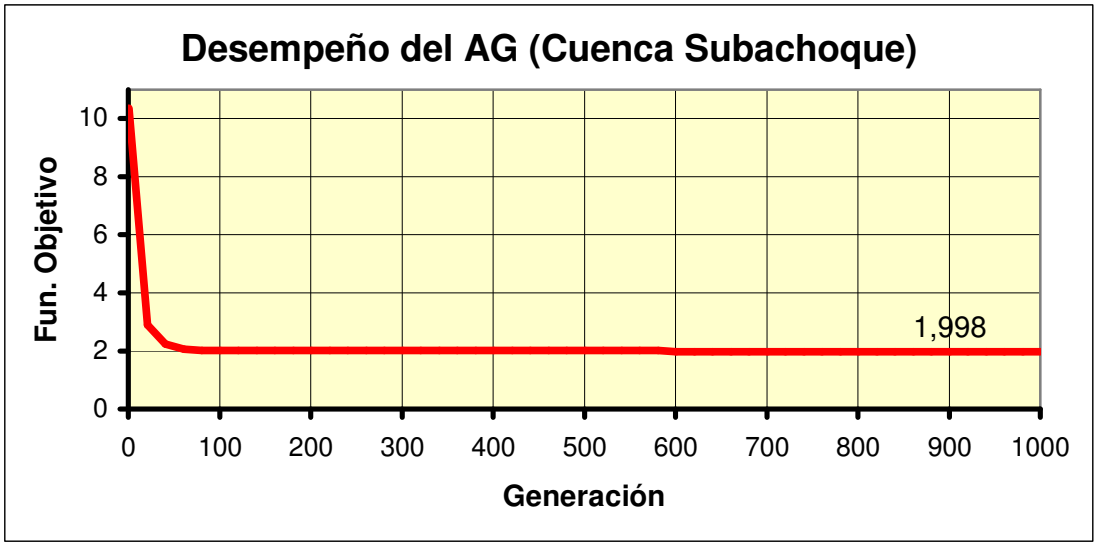


***Figura 34**. Desarrollo de la búsqueda del AG en la cuenca Subachoque.*



Los valores finales en la función objetivo, son el resultado de las siguientes soluciones:

***Tabla 11.** Soluciones obtenidas por el AG.*

	Cuenca de Curibital	**Subcuenca de Subachoque**
a	0,999216	1
b [mm]	2777,4755	348,6693
c	0,3525152	0,7506061
d	0,061545	0,017601
Sw_0 [mm]	244,6184	67,514677
Sg_0 [mm]	7,8277886	480,43053

Lo anterior, muestra lo que el programa calculó en sí. Ahora, a partir de las anteriores soluciones, se calculó de manera externa las Tablas 12 y 13, empleando las expresiones a partir de las ecuaciones Eq. 10 a Eq. 17. En ellas, los valores en las columnas verdes son los datos de entrada (caudales observados, precipitaciones y evapotranspiraciones en *mm*). Las casillas en amarillo indican la solución arrojada por el programa. Igualmente, la casilla `F. Obj.` indica el valor que se obtuvo en la función objetivo y la columna naranja indica los caudales simulados. A partir de estas tablas se grafican los caudales simulados vs. los observados de manera que si las dos líneas coinciden, se habrá logrado una perfecta calibración (ver Figuras 36 y 37).

Tabla 12. *Caudales observados vs. simulados en la cuenca de Curibital.*

a	b[mm]	c	d	Sw0[mm]	Sg0[mm]	F.Obj.
0.999216	277.4755	0.3525152	0.061545	244.6184	7.8277886	5.27810

mes	P[mm]	ETP[mm]	Sw[mm]	Ro[mm]	Rg[mm]	Sg[mm]	Qg[mm]	Qs[mm]	Qr[mm]
1	47.78	66.43	215.84	11.76	6.40	13.41	0.83	12.59	11.86
2	86.98	65.02	217.80	17.81	9.70	21.77	1.34	19.15	24.22
3	76.08	70.87	212.58	12.59	6.85	26.96	1.66	14.25	10.25
4	139.15	70.46	214.63	48.60	26.46	50.32	3.10	51.69	54.83
5	229.01	71.06	214.51	107.82	58.70	102.70	6.32	114.14	107.86
6	210.03	76.89	210.01	95.48	51.98	145.72	8.97	104.45	100.83
7	243.56	77.01	209.97	114.24	62.20	195.86	12.05	126.29	132.59
8	173.34	79.33	208.05	68.89	37.51	219.84	13.53	82.42	91.69
9	121.47	71.57	213.52	34.43	18.74	224.75	13.83	48.26	46.23
10	146.5	69.77	215.22	53.91	29.35	239.37	14.73	68.65	64.39
11	105.42	63.62	219.55	28.82	15.69	240.28	14.79	43.61	35.53
12	52.29	67.88	208.81	3.34	1.82	228.06	14.04	17.38	21.46
Suma	1632	850	2560	598	325	1709	105	703	702

Tabla 13. *Caudales observados vs. simulados en la subcuenca de Subachoque.*

a	b[mm]	c	d	Sw0[mm]	Sg0[mm]	F.Obj.
1	348.6693	0.7506061	0.017601	67.514677	480.43053	1.98833

mes	P[mm]	ETP[mm]	Sw[mm]	Ro[mm]	Rg[mm]	Sg[mm]	Qg[mm]	Qs[mm]	Qr[mm]
1	40.62	49.4	93.85	0.00	0.00	472.12	8.31	8.31	4.22
2	27.17	35.94	109.17	0.00	0.00	463.95	8.17	8.17	9.35
3	92.96	38.87	180.80	0.00	0.00	455.93	8.02	8.02	9.64
4	53.4	55.03	200.01	0.00	0.00	448.04	7.89	7.89	11.76
5	70.94	47.59	236.38	0.00	0.00	440.29	7.75	7.75	9.25
6	51.8	36.45	259.58	0.00	0.00	432.68	7.62	7.62	8.52
7	42.85	45.7	265.27	0.00	0.00	425.19	7.48	7.48	6.31
8	35.29	35.06	271.81	0.00	0.00	417.84	7.35	7.35	5.42
9	77.56	35.92	314.54	0.18	0.53	411.13	7.24	7.41	7.49
10	114.7	55.21	297.61	20.09	60.48	463.45	8.16	28.25	28.46
11	89.93	55.73	297.17	9.69	29.18	484.10	8.52	18.21	17.79
12	48.55	46.54	302.52	0.00	0.00	475.73	8.37	8.37	6.57
Suma	746	537	2829	30	90	5390	95	125	125

Figura 35. Comparación caudales simulados vs. observados en la cuenca Curibital.

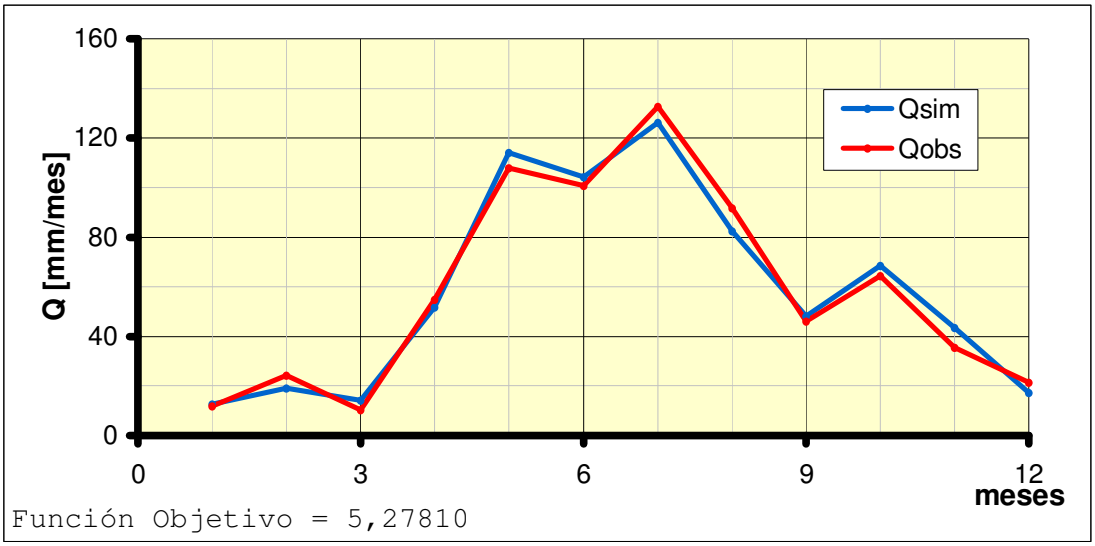
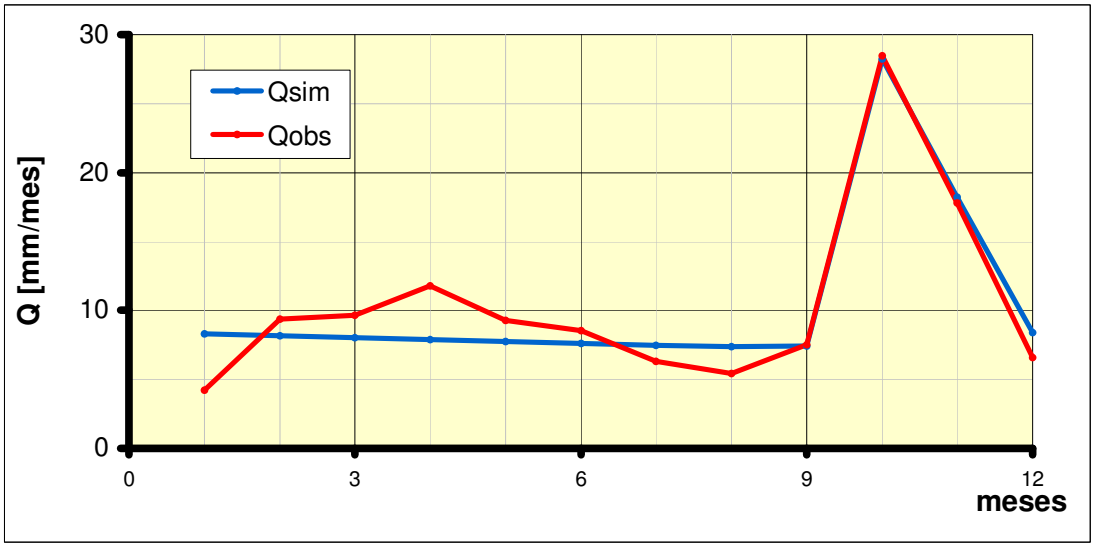


***Figura 36**. Comparación caudales simulados vs. observados en la subcuenca Subachoque.*



Aunque gráficamente se puede observar si se logró o no la calibración, esto también se podrá cuantificar calculando el coeficiente de correlación R^2. Entre más coincidan las líneas roja y azul, más se acercará este valor a 1. Para la cuenca de Curibital, dicho coeficiente fue igual a 0,99098 mientras que para la subcuenca de Subachoque, 0,95057.

2.7 ANÁLISIS DE LOS RESULTADOS

El primer aspecto que se debe tener en cuenta es que sólo el valor obtenido en la función objetivo, no basta para saber si se logró una buena calibración o no. Esto se puede observar fácilmente en las Figuras 35 y 36. Mientras el valor de la función objetivo fue menor en la subcuenca de Subachoque, su calibración no fue tan satisfactoria como la de Curibital. Se observa así, que el valor de la función objetivo (la norma cuadrática normalizada) está influida por los rangos de los caudales que está analizando. Entre más grande es el rango entre los caudales máximos y mínimos, más se le permite a la función objetivo para que distancie del valor cero.

En segundo lugar, los resultados aquí observados son similares pero ligeramente superiores a los obtenidos en el trabajo de Caro y Flechas[140,†].

Se obtuvo un dato curioso en la subcuenca de Subachoque. Aquí se arrojó un valor del parámetro a igual a 1. Esto quiere decir que la escorrentía se da muy fácilmente, mucho antes de que los suelos se comiencen a saturar.

El hecho de no haberse obtenido calibraciones perfectas no indica necesariamente que se deba reevaluar el algoritmo genético empleado. Como se observa a continuación, los AGs son sólo una de las tantas fuentes de error. Nótese que las tres primeras plantean la posibilidad misma de que el modelo sea incalibrable.

- Imprecisiones o malas tomas de datos de campo (precipitaciones, caudales, etc).

- Imprecisiones en el cálculo de las evapotransporaciones (esto no se debería a un error operativo sino a que existe conflictos entre las diferentes metodologías que se pueden emplear para estimar estos datos[‡]).

- Mala adopción del modelo hidrológico. Simplemente no se adoptó el modelo adecuado para dicha cuenca[¶].

- La función objetivo empleada no es la más recomendable. Se deben entonces emplear otras como las expresiones Eq. 19 a Eq. 21. O tal vez exista la necesidad de plantear toda la calibración con un enfoque multiobjetivo, es decir, que satisfaga varias funciones objetivo a la vez.

Lo interesante es que si el AG empleado fuese en sí deficiente, sería muy probable que no se hubiese logrado la calibración que se obtuvo en el modelo para la cuenca Curibital.

---

[140] CARO CAMARGO y FLECHAS PARRA. 2002.

[†] No obstante, a partir de simplemente obtener mejores soluciones que en el trabajo de Caro y Flechas, no basta para decir que los AGs son una mejor metodología a la empleada por ellos. Entrar a comparar las dos técnicas se sale del alcance del presente proyecto, ya que sería necesario entrar a considerar los tiempos empleados, los rendimientos de los computadores empleados, la cantidad de semillas evaluadas, etc.

[‡] Existen muchos métodos en hidrología para medir la evapotranspiración. Cada uno toma diferentes variables y asumen diferentes simplificaciones, arrojando diferentes resultados (en dónde sólo uno podrá ser el verdadero). Algunos de estos métodos son Turc, Penman, Thorntwaite, Christiansen, Hargreaves, Blaney y Criddle, etc. Véase más en MONSALVE. 1999. p. 161 – 188.

[¶] Al igual que con el cálculo de evapotranspiraciones, existen muchos modelos lluvia-escorrentía.

3. CONCLUSIONES Y RECOMENDACIONES

A partir del primer capítulo se concluye que:

1. Los AGs poseen características que los hacen altamente robustos para resolver problemas de optimización, toma de decisiones, diseño y otros usos en la ingeniería civil.

2. Cuando a los AGs se les compara con métodos tradicionales de optimización, se observa que en general, se sacrifica eficacia para ganar más eficiencia y robustez.

3. El extendido aprovechamiento que se le está dando a los algoritmos genéticos es real. No obstante, la ingeniería civil colombiana no ha todavía explotado esta útil herramienta.

4. Aún existe espacio dentro de la teoría de los AGs para continuar con la investigación. No solamente indagando sobre su efectividad sino también extendiendo el abanico de aplicaciones que se le puedan aún dar.

Tras la elaboración y ejecución del aplicativo computacional, se concluye que:

5. Los resultados obtenidos en la calibración del modelo aplicado a la subcuenca de Subachoque abren la especulación sobre las posibles fuentes de error. Estas se resumen así: imposibilidad misma de la calibración, la mala adopción de la función objetivo o la existencia de deficiencias en el algoritmo genético. De tratarse el tercer caso, otros operadores o codificaciones deberán intentarse y compararse con el presente trabajo. Se presume sin embargo, que si el AG empleado fuese en realidad deficiente, no debería poder entonces calibrar el modelo en la cuenca de Curibital.

6. Las soluciones obtenidas mostraron ser coherentes y ligeramente superiores a las obtenidas en trabajos anteriores. No obstante, para entrar a comparar sobre cuál de las metodologías fue mejor, se requiere analizar otros aspectos tales como los tiempos requeridos para la búsqueda, los computadores empleados, la ausencia o presencia de comandos secundarios que reduzcan la velocidad (como por ejemplo, registro de estadísticas), el número de semillas empleadas, etc.

BIBLIOGRAFÍA

ALANDER, Jarmo T. An indexed bibliography of Genetic algorithms: civil and structural engineering [en línea]. Report No. 94-1-CIVIL. Finlandia : University of Vaasa, Department of Information Technology and Production Economics, 2003. [*Citado* 2003-05-02] *Disponible en* ftp://ftp.uwasa.fi/cs/report94-1/gaCIVILbib.ps.Z

ALANDER, Jarmo T. An indexed bibliography of genetic algorithms in the Latin America, Portugal and Spain [en línea]. Report No. 94-1-LATIN. Finlandia : University of Vaasa, Department of Information Technology and Production Economics, 2003. [*Citado* 2003-05-02] *Disponible en* ftp://ftp.uwasa.fi/cs/report94-1/gaLATINbib.ps.Z

AURNHAMMER, Melanie, TÖNNIES, Klaus and GUERICKE, Otto von. Horizon correlation across faults guided by geological constraints (en línea). [*Citado* 2003-05-02] *Disponible en* http://isgwww.cs.uni-magdeburg.de/bv/pub/pdf/TR-ISGBV-02-02.pdf

BEASLEY, David, BULL, David R. and MARTIN, Ralph R. An overview of genetic algorithms: Part 1, Fundamentals [en línea]. En :University Computing : [Reino Unido], Inter-University Comité on Computing, 1993. p. 58-69. [*Citado 2003-06-06*] *Disponible en* www.indicart.com.ar/ga/OVER93_P.GZ

BEASLEY, David, BULL, David R. and MARTIN, Ralph R. An overview of genetic algorithms: Part 2, Research Topics [en línea]. En :University Computing : [Reino Unido], Inter-University Comité on Computing, 1993. p. 58-69. [*Citado 2003-06-06*]

BLANPAIN, Olivier, BOUSSEMART, F., LECOUTRE, C. and MERCHEZ, Sylvain. Genetic algorithms to determine the profile of urban drainage networks form incomplete data (abstract) [en línea]. [1999]. En : DHI Software Conference 1999. IAHR, AIRH. [*Citado* 2003-02-11] *Disponible en* http://www.dhi.dk/hic98/papers/full%20papers/047/abs_047.doc

BRANDON, J. A. The design and implementation of intelligent systems for structural integrity assessment [en línea]. En : International Journal of Systems Science. Vol. 31, No 11 (Nov 2000); p. 1505-1510. [*Citado* 2003-05-20] *Disponible en* http://search.epnet.com/ (autorización requerida). ISSN 00207721.

BROWN, Carol E.. Accounting expert systems applications [en línea]. En : BROWN, Carol. E. And O'LEARY, Daniel E.. Introduction to Artificial Intelligence and Expert Systems [Estados Unidos] [*Actualizado* 1994-12-18] [*Citado* 2003-05-20] *Disponible en* http://accounting.rutgers.edu/raw/aies/www.bus.orst.edu/faculty/brownc/es_tutor/acc_es.htm

CAI, Ximing, McKINNEY, Daene C. and LASDON, Leon S. A framework for sustainability analysis in water resources management and application to the Syr Darya basin [en línea]. [2001] [*Citado* 2003-40-18] *Disponible en* http://www.ce.utexas.edu/prof/mckinney/papers/aral/LTM-Final.pdf

CARO CAMARGO, Carlos Andrés y FLECHAS PARRA, Franklin Paolo. Análisis del balance hídrico de la sabana de Bogotá mediante el método de Thomas. Bogotá, 2002, 139 p. Trabajo de Grado (Ingeniero Civil). Pontificia Universidad Javeriana. Facultad de Ingeniería. Departamento de Ingeniería Civil.

CASTILLO, Enrique y ÁLVAREZ, Elena. Sistemas expertos : aprendizaje e incertidumbre. Madrid : Paraninfo, 1989. ISBN 84-283-1669-4.

CERROLAZA, Miguel y ANNICCHIARICO, William. Algoritmos de optimización estructural basados en simulación genética. Caracas : Universidad Central de Venezuela, Consejo de Desarrollo Científico y Humanístico, 1996. p. 17-66.

ČISTÝ, Milan. Advantages of using genetifc algorithms over deterministic methods in optimal design of the water networks rehabilitation [en línea]. En : Proceedings of ALGORITMY 2000. Conference on Scientific Computing. p. 293-300. [*Citado* 2003-1-10] *Disponible en:* http://www.emis.de/journals/AMUC/~contributed/algo2000/cisty.pdf

COPELAND, Jack. Inteligencia artificial : una introducción filosófica. Traductor: Julio César Armero San José. Madrid : Alianza 1996. 421 p. ISBN 84-206-2844-1.

CHAN, Weng Tat, FWA, T. F. and HOQUE, Kh. Zahidul. Constraint handling methods in pavement maintenance programming [en línea]. En : Transportation Research Part C 9. Pergamon, 2001. p. 175-190. [*Citado* 2003-1-10] *Disponible en:* http://www.cs.cinvestar.mx/~constraint/papers/chan01.pdf.gz.

CHAKRABORTY, Uday Kumar and CHAKRABORTY, Mandir. Analysis of selection algorithms: a markov chain approach. En: Evolutionary Computation, 1997. V. 4 No. 2 p. 133-167. [*Citado* 2002-11-11] *Disponible en* http://search.epnet.com/ (autorización requerida). ISSN 10636560.

CHAKROBORTY, Partha, KALYANMOY, Deb and SRINIVAS, B. Network-wide optimal scheduling of transit systems using genetic algorithms [en línea]. En : Computed-Aided Civil and Infrastructure Engineering. 1998. V. 13. p. 363-376. [*Citado* 2003-04-11] *Disponible en* http://search.epnet.com/ (autorización requerida) *Abstract disponible en* http://www.blackwell-synergy.com*links/doi/10.1111/0885-9507.00115/abs
ISSN 10939687.

CHEN, S-Y. and RAJAN, S. D. Using genetic algorithm as an automatic structural design tool [en línea]. Tempe, Arizona, Estados Unidos, Arizona State University, Department of Civil Engineering, [1998] [*Citado* 2002-11-15] *Disponible en* http://www.geocities.com/shenyeh_chen/paper/wcsmo3a.pdf

CHEN, W. C., CHANG, Ni-Bin and SHIEH, Wen K. Advanced hybrid fuzzy-neural controller for industrial wastewater treatment [en línea]. En : Journal of Environmental Engineering. nov, 2001. [*Citado* 2003-04-10] *Disponible en* http://search.epnet.com/ (autorización requerida) ISSN 07339372.

CHEN, Zhengxin. Computational intelligence for decision support. Boca Ratón, Florida, Estados Unidos : CRC Press, 2000. p. 380. ISBN 0-8493-1799-1.

CHEU, Ruey-Long, JIN, Xin, NG, Kim-Chwee, NG, Yew-Liam and SRINIVASAN, Dipti. Calibration of FRESIM for Singapore expressway using genetic algorithm [en línea]. En : Journal of Transportation Engineering. Nov-Dic, 1998. [*Citado* 2002-11-11] *Disponible en* http://search.epnet.com/ (autorización requerida). ISSN 0733947X.

DE JONG, Kenneth. Genetic algorithms: a 30 year perspective [en línea]. En : UNIVERSIDAD DE MICHIGAN. Festschrift Conference in Honor of John Holland. [Ann Arbor, Michigan, Estados Unidos]. 1999. [*Citado* 2002-11-11] *Disponible en* http://www.pscs.umich.edu/jhhfest/abstracts.html

DE JONG, Kenneth and SPEARS, William. On the state of evolutionary computation [en línea]. [En : Proceedings of the Fifth International Conference on Genetic Algorithms. 1993.] [*Citado* 2002-11-11] *Disponible en* http://navy.mil/~spears/papers/icga93.ps.gz

DE JONG, Kenneth and SPEARS, William M. Using Markov chains to analyze GAFOs. 1995. [*Citado en* 2003-05-04] *Disponible en:* *http://www.aic.nrl.navy.mil/~gordon/papers/foga94.pdf*

DE SCHAETZEN, W., SAVIC, Dragan and WALTERS, G. A. A genetic algorithm approach to pump scheduling in water supply systems (abstract) [en línea]. En : DHI Software Conference 1999. IAHR, AIRH. [*Citado* 2003-02-11] *Disponible en* http://www.dhi.dk/hic98/papers/full%20papers/346/abs_346.doc

DUMITRESCU, D., *et al*. Evolutionary computation. Boca Ratón, Florida, Estados Unidos : CRC Press, 2000. 386 p. ISBN 0-8493-0588-8.

EL HARROUNI, Khalid, OUAZAR, Driss and CHENG, A. H.-D. Gas for parameter estimation in contaminant transport by groundwater [en línea]. [*Citado* 2003-03-11] *Disponible en* http://kfki.baw.de/conferences/ICHE/1998-Cottbus/69.pdf

FALKENAUER, Emanuel. Genetic algorithms and grouping problems. Chichester, Reino Unido : John Wiley & Sons, 1998. 220 p. ISBN 0-471-97150-2.

FEDERHOFER, Judith. Medical expert systems : doctor's silent partners [en línea]. [*Citado* 2003-03-11] *Disponible en* http://www.computer.privateweb.at/judith/special_field3.htm

FENG, Chung-Wei, LIU, Lang and BURNS, Scott A. Using genetic algorithms to solve construction time-cost trade-off problems. En : Journal of Computing in Civil Engineering, V. 11. No. 3. (jul 1997) p. 184. [*Citado* 2003-05-20] *Disponible en* http://search.epnet.com/ (autorización requerida). *Abstract disponible en* http://www.wmich.edu/jcce/v11n3ab4.htm ISSN 08873801.

FISHER, Deborah J., O'NEIL, Michael W. and CONTRERAS, Jeffrey C.. DS^2: Drilled shaft decision support system [en línea]. En : Journal of Construction Engineering and Management. V. 121, No. 1 (mar 1995); p. 86-88. [*Citado* 2003-05-20] *Disponible en* http://search.epnet.com/ (autorización requerida). ISSN 07339364.

FOGEL, Lawrence J. Intelligence through simulated evolution: forty years of evolutionary programming. La Jolla, California, Estados Unidos : John Wiley & Sons, 1999. p. ix-7. ISBN 0-471-33250-X (alk. paper).

FORSYTH, Richard. The expert systems phenomenon. En : FORSYTH, Richard. Expert systems : principles and case studies. 2 *ed*. Londres : Chapman and Hall Computing 1989. ISBN 0-412-30460-0.

GOLDBERG, David E. Genetic algorithms in search, optimization, and machine learning. Reading, Massachusetts, Estados Unidos : Addison-Wesley, 1989. 412 p. ISBN 0-201-15767-5.

GOLDBERG, David E. and SEGREST, Phillip. Finite markov chain analysis of genetic algorithms. Tuscaloosa, Alabama : University of Alabama. En : Proceedings of the Second International Conference on Genetic Algorithms. 1987. p. 1-8.

GOOSENS, Hans, VAN DEN BOOGAARD, Henk and DOUBEN, Klaas-Jan. Application of a hybrid genetic algorithm for inverse model use in river bank design of the river Rhine, the Netherlands [en línea]. En : DHI Software Conference 1999. IAHR, AIRH. [*Citado* 2003-02-11] *Disponible en* http://www.dhi.dk/hic98/papers/full%20papers/306/abs_306.doc

GÜMRAH, Fevzi, ERBAS, Demet, ÖZ, Bora and ALTINTAŞ. Waste water disposal: polluted aquifer cleanup optimization by using genetic algorithms [en línea]. En : Energy Sources. 2000. V. 22. p. 543-556. [*Citado* 2003-03-20] *Disponible en* http://search.epnet.com/ (autorización requerida). *Abstract disponible en* http://mustafa.ingentselect.com/vl=5129965/cl/=30/nw=1/rpsv/catchword/tandf/00908312/v22n6/s5/p543

HAIDAR, A. e*t al.* Genetic algorithms application and testing for equipment selection [en línea]. En : Journal of Construction Engineering and Management. (ene-feb-1999); p. 32. [*Citado* 2003-05-20] *Disponible en* http://search.epnet.com/ (autorización requerida). ISSN 07339364.

HADI, Mamad N. S. and SCHMIDT, Lewis C. Using genetic algorithms to find the optimum design of reinforced concrete beams [en línea]. [En: Australasian Matlab Users Conference 2000]. [*Citado* 2003-05-20] *Disponible en:* http://www.ceanet.com.au/mluserconf/papers/Hadi_Schmidt.pdf

HALHAL, D., WALTERS, G. A., OUAZAR, D. and SAVIC, D. A. Water network rehabilitation with a structured messy genetic algorithm (abstract) [en línea]. [*Citado* 2003-02-11] *Disponible en* http://www.lania.mx/~ccoello/EMOO/halhal97.ps.gz (el sitio podrá estar dañado, así que se recomienda buscar el título por google.com y luego de encontrar la obra, ver versión de texto)

HODGES, Julia *et al*. Assessing the performance of a waste characterization expert system [en línea]. En : Applied Artificial Intelligence. V. 15, No. 4 (Abr 2001); p. 385-387. [*Citado* 2003-05-20] *Disponible en* http://search.epnet.com/ (autorización requerida). ISSN 08839514.

HOLLAND, John H. Adaptation in Natural and Artificial Systemas. Cambridge, Massachusetts : MIT Press, 1992. Edición 2. 211 p. ISBN 0-262-08213-6

HOPGOOD, Adrian A. Intelligent Systems for Engineers and Scientists. Boca Ratón, Florida : CRC Press, 2001. Edición 2. 467 p.

JAYAWARDHANA, L. C. *et al*. BESTCOMP : expert system for Sri-Lankan solid waste composting (abstract) [en línea]. En :Expert systems with applications. V. 24, No. 3 (Abr 2003). [*Citado* 2003-05-20] *Disponible en* http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6V03-47C49DF-1&_user=10&_handle=W-WA-A-A-Y-MsSAYZW-UUA-AUCCADYEZZ-EUDDWDUW-Y-U&_fmt=summary&_coverDate=04%2F01%2F2003&_rdoc=4&_orig=browse&_srch=%23toc%235635%232003%23999759996%23398732!&_cdi=5635&view=c&_acct=C000050221&_version=1&_urlVersion=0&_userid=10&md5=76a98d48f6009b006b7e45bdffc15b7b

JIN, Y.-Q. and WANG, Y. A genetic algorithm to simultaneously retrieve land surface roughness and soil wetness [en línea]. En : International Journal of Remote Sensing. V. 22. No. 16. 2001. p. 3093-3099. [*Citado* 2003-05-20] *Disponible en* http://dandini.ingentaselect.com/vl=11019346/cl=19/nw=1/rspv/catchword/tandf/01431161/v22n16/53/p3093 ISSN 1366-5901.

JONG, Jyh-Cherng, JHA, Manoj K. and SCHONFELD, Paul. Preliminary highway design with genetic algorithms and geographic information systems [en línea]. En : Computer-Aided Civil and Infrastructure Engineering. Julio, 2000. V. 15. No. 4. p. 261-271 [*Citado* 2003-05-20] *Disponible en* http://www.engr.umd.edu/~mjha/papers/c_a.pdf *También disponible en* http://search.epnet.com/ (autorización requerida). ISSN 10939687.

KARR, Charles L. and FREEMAN, L. Michael. Industrial applications of genetic algorithms. Boca Ratón, Florida, Estados Unidos : CRC Press, 1999. ISBN 0-8493-9801-0.

KOUMOUSIS V. K. and ARSENIS, S. J. Genetic algorithms in optimal detailed design of reinforced concrete members [en línea]. En : Computer-Aided Civil and Infraestructure Engineering. 1998. V. 13. p. 43-52. [*Citado* 2003-05-20] *Disponible en* http://search.epnet.com/ (autorización requerida). *Abstract disponible en:* http://www.blackwell-synergy.com/links/doi/10.1111/0885-9507.00084/abs/ ISSN 10939687

LIONG, Shie-Yui, CHAN, Weng Tat and SHREERAM, Jaya. Peak-flow forecasting with GA and SWMM [en línea]. En : Journal of Hydraulic Engineering. (Ago 1995); p. 613-617. [*Citado* 2002-11-11] *Disponible en* http://search.epnet.com/ (autorización requerida). ISSN 07339429.

LIONG, Shie-Yui, KHU, Soon Thiam and CHAN, Weng Tat. Derivation of pareto front with accelerated convergence genetic algorithm, ACGA (asbstract) [en línea]. En : DHI Software Conference 1999. IAHR, AIRH. [*Citado* 2003-02-11] *Disponible en* http://www.dhi.dk/hic98/papers/full%20papers/038/abs_38.doc

LIPPAI, Istvan, HEANEY, James P. and LAGUNA, Manuel. Robust water system design with commercial intelligent search optimizers [en línea]. En : Journal of computing in civil engineering. Jul, 1999. [*Citado* 2003-02-11] *Disponible en* http://search.epnet.com/ (autorización requerida). ISSN 08873801.

MAHACHI, J. An integrated approach to critical time-space scheduling [en línea]. [En : International Conference: IT in Construction in Africa. CSIR : Mpumalanga, Sudáfrica, may 30 – jun 01, 2001.] [*Citado* 2003-05-02]. *Disponible en:* http://buildnet.csir.co.za/constructitafrica/authors/Papers/w78-063.pdf

MARTÍNEZ, José J. y ROJAS, Sergio A. Introducción a la informática evolutiva. Bogotá : Universidad Nacional de Colombia, Facultad de Ingeniería, Departamento de Sistemas, Octubre 1999.

McCARTHY J. *et al*. A proposal for the Dartmouth Summer Research Project on artificial intelligence [en línea]. 1955-08-31. [*Actualizado* 1996-04-03] [*Citado* 2003-03-02]. *Disponible en:* http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html

McREADY, William G. and WOLPERT, David H. On 2-armed gaussian bandits and optimisation [en línea]. Santa Fe, New Mexico, Estados Unidos: Santa Fe Institute. 1996-03-07. [*Citado* 2003-03-02]. *Disponible en:* http://www.santafe.edu/sfi/publications/Working-Papers/96-03-009.ps

MEIER, Roger W. and BARKDOLL, Brian D. Sampling design for network model calibration using genetic algorithms [en línea]. En : Journal of Water Ressources Planning and Management. jul-ago, 2000. *Disponible en* http://search.epnet.com/ (autorización requerida) *Abstract disponible en* http://home.olemiss.edu/~cvbark/samplingdesign.html ISSN 07339496.

MONSALVE SÁENZ, Germán. Hidrología en la ingeniería. 2 ed. Bogotá : Escuela Colombiana de Ingeniería, 1999. 383 p. ISBN 958-95742-1-1.

MURPHY, L. J. and SIMPSON, A. R. Genetic algorithms in pipe network optimisation (reporte) [en línea]. Research report No. R93. Australia : University of Adelaide, jun 1992. [*Citado* 2003-03-02]. *Disponible en:* http://www,civeng.adelaide.edu.au/research/reports/R93.pdf

NASSAR, Khaled. Managing construction equipment buy and sell decisions replacement: a simulation game. En : ASC Proceedings fo the 37th Annual Conference. Denver, Colorado, Estados Unidos : University of Denver, abril 2001. p. 187-798. [*Citado* 2003-05-20] *Disponible en* http://asceditor.unl.edu/archives/2001/nasser01b.htm

NATSUAKI *et al.* Application of genetic algorithm to bridge construction management (abstract) [en línea]. [En : Proceedings of the Third International Symposium on Uncertainty Modeling and Analysis and Annual Conference of the North American Fuzzy Information Processing Society. IEEE, : Los Alamitos, California, Estados Unidos] marzo 1995 p. 105 – 108. [*Citado* 2003-05-20] *Disponible en* http://www.computer.org/proceedings/isuma/7126/71260105abs.tm

NISHIMURA, Rumi, YONEDA, Minoru and MORISAWA, Shinsuke. Optimal design of sampling location for evaluating soil contamination using stochastic methods and genetic algorithms (abstract) [en línea]. En : Environmental Sanitary Engineering Resarch. 1999. V. 13. No. 2. p. 5-18. [*Citado* 2003-04-20] *Disponible en* http://koken-db.kogaku.kyoto-u.ac.jp/1999/B/GE/99GE05101.html

NUNOO, Charles N. Optimization of pavement maintenance and rehabilitation programming using shuffled complex evolution algorithm (abstract) [en línea]. Ph. D. Dissertation. 2001. [*Citado* 2003-01-16] *Disponible en* http://eic4.eng.fiu.edu/~cnunoo/resume/ -> *Abstract*

NUTECH SOLUTIONS, INC. A major communication network provider. [*Actualizado* 2001] [*Citado* 2003-05-01] *Disponible en* http://www.nutechsolutions.com/pdf/cs_network_design.pdf

NUTECH SOLUTIONS, INC. DaimlerChrysler Aerospace AG [*Actualizado* 2001] [*Citado* 2003-05-01] *Disponible en* http://www.nutechsolutions.com/pdf/cs_daimler_chrysler_engineering.pdf

NUTECH SOLUTIONS, INC. The Dutch Ministry of Traffic [*Actualizado* 2001] [*Citado* 2003-05-01] *Disponible en* http://www.nutechsolutions.com/pdf/cs_dutch_min_traffic.pdf

O'BRIEN, W.J., M.A. Fischer y B.H. Akinci. Importance of site conditions and capacity allocation for construction cost and performance : a case study [en línea]. [Estados Unidos] [*Actualizado* 1997-08-04] [*Citado* 2003-03-14]. *Disponible en:* http://web.bham.ac.uk/d.j.crook/lean/iglc5/obrien/obrien.htm

PACHEPSKY, Yakov and TIMLIN, Dennis. Using genetic algorithms to estimate soil hydraulic conductivity when boundary conditions are unkown [en línea]. [En : SSSA Annual Meetings. Indiana, Estados Unidos : Soil Science Society of America, nov 1996.] . [*Citado* 2003-02-10] *Disponible en* http://www.agr.okstate.edu/posters.96/pachepsky/pachepsky.htm

PALM, S.K., BANDYOPADHYAY, S. and MURTHY, C.A. Genetic classifiers for remotely sensed images: comparison with standard methods [en línea]. En : Int. J. Remote Sensing. Calcuta, India, 2001. V. 22. No. 13. p. 2545–2569. [*Citado* 2003-01-16] *Disponible en* http://www.isical.ac.in/~sanghami/pal_ijrs.pdf.gz

PARMEE, I. C. Genetic algorithms and hydropower system design [en línea]. En : Computer-Aided Civil and Infrastructure Engineering. ene 1998. V. 13. No. 1. p. 31. [*Citado* 2003-01-16] *Disponible en* http://www.blackwell-synergy.com/links/doi/10.1111/08885-9507.00083/abs/ *Abstract disponible en* http://search.epnet.com/ (autorización requerida) ISSN 10939687.

POOLE, David Lynton, MACKWORTH, Alan K. and GOEBEL, Randy. Computational intelligence : a logical approach. Nueva York : Oxford University, 1998. ISBN 0195102703

PROCOPIO, Michael J. Genetic algorithm analysis [en línea]. Gainesville, Florida, Estados Unidos, 2001, 45 p. Senior Design Project, Final Report. Universidad de Florida. Department of Computer and Information Science and Engineering. [*Citado en* 2003-05-04] *Disponible en:* http://www.mikeprocopio.com/publications/SeniorDesignFinalReport.pdf

ROTSHTEIN, Alexander. Diagnosing the cause of structural cracks in buildings (abstract) [en línea]. En : Evoweb, marzo 2001. [*Actualizado en* 2002-10-14] [*Citado en* 2003-05-04] *Disponible en:* http://evonet.dcs.napier.ac.uk/evoweb/resources/evolution_work/ework42.html

RUSSELL, Stuart J. and NORVIG, Peter. Artificial intelligence : a modern approach. Upper Saddle River, New Jersey, Estados Unidos : Prentice Hall, 1995. ISBN 0-13-103805-2.

SÁNCHEZ TOMÁS, Antonio. Expert systems applications in accounting. En : VASARHELYI, Miklos A. y KOGAN, Alexander. Artificial intelligence in accounting and auditing : towards new paradigms. Princeton, Nueva Jersey, Estados Unidos : Markus Wiener Publishers, 1998. v. 4, p. 7-28.

SCHWEITZER *et al*. Optimization of road networks using evolutionary strategies [en línea]. En : Evolutionary Computation. Massachusetts, Estados Unidos : MIT, 1998. v. , No. 4. p. 419-438. [Citado 2002-11-11] *Disponible en* http://www.first.gmd.de/persons/rose/PAPER/paper/opt-rev.ps.gz *También disponible en* http://summa.physik.hu-berlin.de/~frank/download/web-evcomp.ps.gz y http://summa.physik.hu-berlin.de/~frank/download/web-evcomp.zip

SERRANO, Sergio E. Hydrology for engineers, geologists and environmental professionals. Lexington, Kentucky : HydroScience, 1997. ISBN 0-9655643-9-8.

SINGH, Vijay P. Computer models of watershed hidrology. Highlands Ranch, Colorado, Estados Unidos : Water Resources Publications,1995. ISBN 0-918334-91-8.

SOH, Chee-Kiong and YANG, Jiaping. Optimal layout of bridge trusses by genetic algorithms [en línea]. En : Computer-Aided Civil and Infrastructure Engineering. 1998. V. 13. p. 247-254. [*Citado* 2003-05-20] *Disponible en* http://search.epnet.com/ (autorización requerida). ISSN 10939687.

SOLOMATINE, D. P. Random search methods in model calibration and pipe network design [en línea]. En : Water Industry Systems: Modelling and Optimization Applications. V. 2. SAVIC, D, WALTERS, G. editors. Baldock, Reino Unido : Research Studies Press, 1999.] [*Citado en* 2003-05-02] *Disponible en:* http://www.ihe.nl/hi/sol/papers/CCWI99-Glopt.pdf

SOROOSHIAN, Sorosoosh, GUPTA, Hoshi Vijai and BASTIDAS, Luis Alberto. Calibration of hydrologic models using multi-objectives and visualization techniques [en línea]. Tucson, Arizona, Estados Unidos, 1998, p. 1-23. Final Project on Project Proposal Number EAR-9418147. Universidad de Arizona. Department of Hidrology and Water Resources. [citado en 2002-11-11] *Disponible en:* http://salt.hwr.arizona.edu/eoshome/proj/nsf/finalreport.pdf

SOROOSHIAN, Sorosoosh and GUPTA, Hoshi Vijai. Model calibration. En : SINGH, Vijay P. Computer models of watershed hidrology. Highlands Ranch, Colorado, Estados Unidos : Water Resources Publications,1995. ISBN 0-918334-91-8.

SPEARS *et al*. An overview of evolutionary computation [en línea]. En :Proceedings of the 1993 European Conference on Machine Learning. Viena, Austria. [1993] [*Citado* 2002-11-11] *Disponible en* http://www.aic.nrl.navy.mil/~spears/papers/ecml93.ps.gz *También disponible en* http://www.cs.uwyo.edu/~wspears/overview/ecml93.all.html

STILLWELL, Michael. Studies in evolutionary algorithms [en línea]. [Australia] School of Computer Science and Software Engineering. Faculty of Information Technology. Monash University. 2001-11-13. [*Citado* 2003-03-14] *Disponible en:* http://beebo.org/thesis/stillwell-2001.pdf

SUDARSHAN, R. Genetic algorithms and application to the optimization of space trusses (bachelor thesis) [en línea]. India, Department of Civil Engineering, Indian Institute of Technology, Madras, mayo 2000. [*Citado* 2003-05-20] *Disponible en* http://wavelets.mit.edu/~darshan/pubs/btech_thesis.pdf

VAN DITZHUIJZEN. Geological parameterization of a reservoir model for history matching [en línea]. En : JPT Online, jul 2003. [*Citado* 2002-11-17] *Disponible en*: http://www.spe.org/spe/cda/views/jpt/jptMaster/0,1513,1648_2300_2474978_0,00.html

VASQUEZ, José A., MAIER, Holger R., LENCE, Barbara Jo., TOLSON, Bryan A. and FOSCHI, Ricardo O. Achieving water quality system reliability using genetic algorithms [en línea]. En : Journal of Environmental Engineering. D*isponible en:* http://search.epnet.com/ (autorización requerida) ISSN 07339372.

VITOVSKY, John P., SIMPSON, Angus R. and LAMBERT, Martin F. Leak detection and calibration using transients and genetic algorithms [en línea]. En : Journal of Water Resources Planning and Management. jul-ago, 2000. [*Citado* 2002-11-17] *Disponible en*: http://www.hydrosoft.co.kr/documents/Leak_GA.pdf *También disponible en:* http://search.epnet.com/ (autorización requerida). ISSN 07339496.

WILSON, Stewart W. Explore/exploit strategies in autonomy [en línea]. Cambridge Massachusetts, Estados Unidos : The Rowland Institute for Science. [*Citado* 2003-03-14]. *Disponible en:* http://world.std.com/~sw/ps/ee.ps.gz *También disponible en:* MAES, P. Et al. From animals to animats 4. Proceedings of the Fourth International conference on simulation of adaptive behaviour. Cambridge Massachusetts, Estados Unidos : 1996. The MIT Press/Bradford Books. p. 325. ISBN 0-262-63178-4

WOODWARD, R. J. Deliverable D3: review of existing procedures for optimization. (BRIME PL97-2220) [en línea]. Transport RTD. Program of the 4th Framework Program, European Commission, Enero 2001. [*Citado* 2003-05-20] *Disponible en* http://www.trl.co.uk/brime/D3.pdf

YAN, Shengquan and MINSKER, Barbara. A dynamic meta-model approach to genetic algorithm solution of a risk-based groundwater remediation design model [en línea]. [En : American Society of Civil Engineers (ASCE) Environmental & Water Resources Institute (EWRI) World Water & Environmental Resources Congress 2003 & Related Symposia, Filadelfia, Filadelfia, Estados Unidos, 2003] [*Citado* 2003-02-15] *Disponible en* http://cee.uiuc.edu/emsa/conference/smyan-2003-01.pdf

YEH, Chao-Hsien and LABADIE, John W. Multiobjective watershed-level planning of storm water detention systems [en línea]. En : Journal of water resources planning and management. Nov-dic, 1997. [*Citado* 2003-01-15] *Disponible en* http://search.epnet.com/ (autorización requerida) *Abstract disponible en* http://knight.fcu.edu.tw/~chyeh/publication/j97-2.htm ISSN 07339496.

WALPOLE, Ronald E., MYERS, Raymond H. and MYERS, Sharon L. Probabilidad y estadística para ingenieros. 6 *ed*. Traductor: Ricardo Cruz. México : Pearson Educación, 1999. 739 p. ISBN 970-17-064-6.

ANEXO A
CÓDIGO DE PROGRAMACIÓN DESARROLLADO PARA EL CASO ESTUDIO

El siguiente código está escrito en lenguaje de Matlab R12. Está dividido en cuatro archivos: AG8.m, funObj.m, funObjDescrip.m y varsGlobales.m. El primero contiene el programa principal. El segundo contiene la función objetivo (sólo se presentará el correspondiente a la subcuenca de Subachoque). El tercero contiene unas líneas descriptivas de la función objetivo. El cuarto contiene las variables globales del programa. Por tratarse de Matlab R12, las variables globales deben ir en un archivo aparte.

AG8.m

```
function AG8
% PROGRAMA ALGORITMO GENÉTICO
%         versión 8.0 - entrecruzamiento doble; elitismo; normalización
% por Rafael Ernesto Olarte Valdivieso
% Creado: Mayo 19 de 2003
%
%
% El presente programa emplea un algoritmo genético para maximizar o minimizar
una función
% objetivo:
%
%  1. La solución a buscar podrá contener varios números reales.
%
%  2. Método de selección
%      Ruleta con normalización.  Se evalúa cada cromosoma en la función
%      y se reescala este valor a un rango entre 0 y 10.  Se obtienen así valores
positivos
%      los cuales compondrán los diferentes pesos en la ruleta.
%      OJO:  En el reescalamiento, existe la siguiente correspondencia
%            0   -> mínimo valor en la generación que se encuentre el algoritmo.
%            10  -> máximo valor en la generación que se encuentre el algoritmo.
%
%  3. Operadores Genéticos
%     - Reproducción y Entrecruzamiento DOBLE (double-point crossover).  Luego,
el cromosoma
%       con el desempeño menos sobresaliente, es reemplazado por el mejor
cromosoma de la
%       generación anterior.
%     - Mutación
%
%  4. Todas las poblaciones contenien un número fijo de cromosomas, pero debe ser
una cantidad
%     par.
%
%  5. Antes de correr el programa, el usuario debe entrar a los archivos
"funObj.m" y
%     "funObjDescrip.m" para establecer la función objetivo, si desea maximizar o
minimizar
%     y 3 líneas de texto que describan a dicha función.
%
```

```
%  6. Dentro del programa, el usuario debe indicar 3 aspectos, así:
%     Parámetros del Algoritmo Genético. Son 7 variables con los siguientes
nombres:
%       - TPOB       tamaño de la población
%       - MAXGEN     número de generaciones
%       - PX         probabilidad de entrecruzamiento
%       - PMUT       probabilidad de mutación
%       - LCAD       longitud de cada una de las subcadenas que conformarán al
cromosoma
%       - MINVAL     mínimo valor que puede ser representado por la cadena.
%       - MAXVAL     máximo valor que puede ser representado por la cadena.
%
%     Semilla para la generación de números seudoaleatorios.
%
%     Nombre de los archivos donde se almacenarán los resultados.  Cada resultado
de cada
%     generación, se imprime en un archivo de texto diferente, localizados en el
mismo
%     directorio del programa.
%
%  7. Se llevan unas cuantas estadísticas.
%     - A nivel de cada generación: el desempeño máximo, el valor que genera
dicho desempeño
%       máximo, el desempeño mínimo, el valor que genera dicho desempeño mínimo,
el desempeño
%       promedio, el total de mutaciones y entrecruzamientos realizados.  Se
llama acá
%       "desempeño" al valor que se obtiene de evaluar un cromosoma en la función
objetivo.
%       Debido a que se aplicara elitismo, se requerirá también llevar los
genotipos que
%       generaron el máximo y el mínimo desempeño.
%     - A nivel de todo el AG: el máximo desempeño obtenido entre todas las
generaciones, el
%       valor que genera dicho desempeño máximo, el mínimo desempeño, el valor
que genera
%       dicho desempeño mínimo, el total mutaciones y el total de
entrecruzamientos.
%
%  8. El programa garantiza un formato de impresión legible cuando los cromosomas
no contengan
%     más de 33 genes.


%**VARIABLES
GLOBALES**************************************************************************
****

%  Se optó por declarar como global, los parámetros del AG y la descripción de la
función objetivo.
%  Cada función incluirá la llamada al archivo  (varGlobales.m)  donde se hace la
declaración de
%  dichas variables globales.
%
%   Parámetros del AG
%   - TPOB  : (REAL) Tamaño de la población
%   - MAXGEN: (ENTERO) Número que indica la cantidad de generaciones que se
producirán
%   - PX    : (REAL) Probabilidad de entrecruzamiento
%   - PMUT  : (REAL) Probabilidad de mutación de cada gen
%   - LCAD  : (VECTOR DE ENTEROS) Longitud de cada una de las subcadenas
```

```
%   - MINVAL: (VECTOR DE REALES) Valor real mínimo que puede ser representado por
la subcadena
%   - MAXVAL: (VECTOR DE REALES) Valor real máximo que puede ser representado por
la subcadena
%
%   Descripción de la función objetivo
%   - FTXT1, FTXT2, FTXT3: (STRING) Líneas que describen la función objetivo
%   - FIN                : (STRING) Podrá contener el valor 'MAXIMIZAR' o
'MINIMIZAR' indicando así
%                          lo que se quiere hacer con la función objetivo


%**VARIABLES
LOCALES*************************************************************************
***

%  - gen      : (ENTERO) Contador que indica cuál es la generación actual.
%
%  - cromosomas: (VECTOR DE ESTRUCTURAS) Cada estructura contiene los valores y
demás propiedades
%                correspondientes a cada cromosoma de la generación actual.
Estos valores son:
%                - genotipo : (ARREGLO DE VALORES LÓGICOS) Contiene toda la
cadena de genes.
%                - fenotipo : (VECTOR DE REALES)  Contiene los números reales
representados por el
%                             genotipo.
%                - padres   : (VECTOR DE ENTEROS)  Indica los 2 padres de donde
se nació el cromosoma.
%                - xPos     : (ENTERO) Posición en las cadenas de los padres
donde ocurrió el
%                             entrecruzamiento.
%                - nMut     : (ENTERO) Número de mutaciones que sufrió el
cromosoma, después del
%                             entrecruzamiento.
%                - desempeno: (REAL) Valor de la función objetivo cuando se le
aplica al valor del
%                             fenotipo.
%
%  - cromosomas0: (VECTOR DE ESTRUCTURAS) Cada estructura contiene los valores y
algunas propiedades
%                 correspondientes a cada cromosoma de la generación anterior.
Estos valores son:
%                 - genotipo : (ARREGLO DE VALORES LÓGICOS)
%                 - fenotipo : (VECTOR DE REALES)
%                 - peso     : (REAL) Porcentaje que indica el peso del cromosoma
en la ruleta.
%                 - seleccion: (REAL) Porcentaje que relaciona el número de veces
en que el cromosoma
%                              fue seleccionado para ser padre entre el número
total de selecciones.
%
%  - estadsGen: (ESTRUCTURA) Contiene estadísticas de la generación actual.  Las
variables dentro de
%               la estructura son:
%               - maxDesemp  : (REAL) Desempeño máximo observado en alguno de los
cromosomas
%               - fenConMaxD : (REAL) Fenotipo que generó el desempeño máximo
%               - genConMaxD : (VECTOR DE VALORES LÓGICOS) Genotipo que generó el
desempeño máximo
%               - minDesemp  : (REAL) Desempeño mínimo de los cromosomas
```

```
%               - fenConMinD : (REAL) Fenotipo que generó el desempeño mínimo
%               - genConMinD : (VECTOR DE VALORES LÓGICOS) Genotipo que generó el
desempeño mínimo
%               - medDesemp  : (REAL) Desempeño promedio de los cromosomas
%               - nX         : (ENTERO) Total de entrecruzamientos ocurridos en
el establecimiento de
%                              la generación actual.
%               - nMut       : (ENTERO) Total de mutaciones ocurridas en todos
los cromosomas de la
%                              generación actual.
%
%  - estadsAG: (ESTRUCTURA) Contiene estadísticas propias del AG.  La estructura
contiene las
%              siguientes variables.
%              - maxDesemp  : (REAL) Máximo desempeño en lo corrido del AG
%              - fenConMaxD : (REAL) Fenotipo que ha generado el desempeño máximo
en lo corrido del AG
%              - minDesemp  : (REAL) Mínimo desempeño en lo corrido del AG
%              - fenConMinD : (REAL) Fenotipo que ha generado el desempeño mínimo
en lo corrido del AG
%              - nX         : (ENTERO) Total de todos los entrecruzamientos
ocurridos en el algoritmo
%              - nMut       : (ENTERO) Total de todas las mutaciones ocurridas en
el algoritmo.
%
%  - semilla:  (ENTERO) Semilla para la generación de números seudoaleatorios.
%
%  - arch:  (VETOR DE CARACTERES) Contiene la parte inicial de cómo se llamarán
los archivos donde se
%           almacenarán las generaciones.
%
%  - ext:  (VECTOR DE CARACTERES) Contiene el nombre de la extensión de los
archivos donde se
%          almacenarán las generaciones.
%
%  - i, j:  (ENTEROs)  Otras variables auxiliares


%**FUNCIONES
ADICIONALES**************************************************************************
**

%  Funciones llamadas directamente por la función principal AG
%  01) obtenerParametros
%  02) obtenerOtros
%  03) generarPob0
%  04) generarPob
%  05) generarEstadsGen
%  06) generarEstadsAG
%  07) reporte0
%  08) reporte
%  09) reporteF
%
%  Funciones llamadas por las anteriores funciones
%  10) seleccionar
%  11) entrecruzar
%  12) mutar
%  13) decodificar
```

```
%**ARCHIVOS ENLAZADOS CON EL PRESENTE
PROGRAMA***********************************************

%  varsGlobales.m   :  Contiene las declaraciones de las variables globales.
%  funObjetivo.m    :  Contiene la función objetivo.
%  funObjetivoTxt.m :  Contiene una función que devuelve tres lineas de texto
indicando la
%                      descripción de la función.


%**PROGRAMA
PRINCIPAL**************************************************************************
***

clear

% Declaración de variables globales
varsGlobales;

% Obtención de los parámetros del AG: TPOB, MAXGEN, PX, PMUT, LCAD, MAXVAL,
MINVAL.
obtenerParametros;

% Obtención de la semilla para generar números seudoaleatorios, y
% especificaciones relacionadas con los archivos: sem, arch, ext
[semilla, arch, ext] = obtenerOtros;

% Crear la población inicial (Generación 0)
cromosomas = generarPob0(semilla);

% Generar estadísticas de la Generación 0.
estadsGen = generarEstadsGen(cromosomas);

% Generar primeras estadísticas del AG.
estadsAG.maxDesemp = estadsGen.maxDesemp;
estadsAG.fenConMaxD = estadsGen.fenConMaxD;
estadsAG.minDesemp = estadsGen.minDesemp;
estadsAG.fenConMinD = estadsGen.fenConMinD;
estadsAG.nX = estadsGen.nX;
estadsAG.nMut = estadsGen.nMut;

% Imprimir en pantalla y en un archivo, la generación 0 y sus estadísticas.
reporte0(cromosomas, estadsGen, arch, ext, semilla);

gen = 1;

while (gen <= MAXGEN) & (estadsGen.maxDesemp ~= estadsGen.minDesemp),

%  Establecer la generación actual como la generación anterior, y crear la nueva
generación.
   [cromosomas, cromosomas0] = generarPob(cromosomas, estadsGen.maxDesemp,
estadsGen.minDesemp, estadsGen);

%  Generar estadísticas de la generación actual y del AG.
   estadsGen = generarEstadsGen(cromosomas);
   estadsAG = generarEstadsAG(estadsAG, estadsGen);

%  Imprimir en un archivo: la generación anterior, la generación actual, las
estadísticas de la
%  generación actual y las estadísticas parciales del AG.
   reporte(gen, cromosomas, cromosomas0, estadsGen, estadsAG, arch, ext);
```

```
   gen = gen +1;

end

% Imprimir en pantalla, la última generación, junto con sus estadísticas y las de
todo el AG.
reporteF(gen -1, cromosomas, estadsGen, estadsAG, arch, ext);


%**FUNCIÓN (01) obtenerParametros
************************************************************
function obtenerParametros

% Declaración de las variables globales.
varsGlobales;

% Inicialización de las variables globales LCAD, MINVAL y MAXVAL.  Esto se hace
para impedir el
% conflicto que se genera cuando al cambiar a una función objetivo que requiere
menos argumentos,
% estos vectores no reducen su tamaño.
LCAD = [0];
MINVAL = [0];
MAXVAL = [0];

% Obtención de la descripción de la función a optimizar.
[FTXT1, FTXT2, FTXT3, FIN] = funObjDescrip;

% Borrar pantalla.
clc;

% Imprimir en pantalla el encabezado.
disp '          +---------------------------------+';
disp '          |  PROGRAMA ALGORITMO GENÉTICO    |';
disp '          |    por Rafael E. Olarte V.      |';
disp '          |        Abril 16 de 2003         |';
disp '          +---------------------------------+';
fprintf(1, '\n\n\n\n');

% Imprimir la función objetivo
fprintf(1, '%s%s%s\n\n', 'EL PRESENTE PROGRAMA BUSCA ', FIN,' LA FUNCIÓN:');
fprintf(1, '%s%s\n', '  ', FTXT1);
fprintf(1, '%s%s\n', '  ', FTXT2);
fprintf(1, '%s%s\n', '  ', FTXT3);

% Ingreso de los 4 primeros parámetros del AG
fprintf(1, '\n\n\n');
disp 'INGRESAR PARÁMETROS DEL ALGORITMO GENÉTICO:';
disp ' ';
TPOB   = input('  1) Tamaño de la Población             : ');
MAXGEN = input('  2) Número de Generaciones             : ');
PX     = input('  3) Probabilidad de Entrecruzamiento   : ');
PMUT   = input('  4) Probabilidad de Mutación           : ');

% Ingreso de los 3 últimos parámetros del AG
disp ' ';
disp '  5) Longitud de cada Subcadena         : ?????';
disp '  6) Valor Real Mínimo de cada Subcadena: ?????';
disp '  7) Valor Real Máximo de cada Sucadena : ?????';
disp '  ';
```

```
disp '          +-------------------------------------------------------+';
disp '          | Indique como desea determinar los parámetros 5, 6 y 7.   |';
disp '          |                                                       |';
disp '          | a) Ingresar directamente longitud, valor máximo y valor  |';
disp '          |    mínimo para cada subcadena.                        |';
disp '          | b) Ingresar longitud, valor mínimo y precisión para cada |';
disp '          |    subcadena.                                         |';
disp '          | c) Ingresar longitud, valor máximo y precisión para cada |';
disp '          |    subcadena.                                         |';
disp '          | d) Ingresar valor mínimo, valor máximo y precisión para  |';
disp '          |    cada subcadena.                                    |';
disp '          +-------------------------------------------------------+';
resp1 = input ('            (Indique letra correspondiente) >> ', 's');

switch resp1
   case {'a', 'A'}
       i = 1;
       resp2 = '1';

      while resp2 == '1',
         fprintf(1, '\n\n%s%d\n', '            Subcadena # ', i);
         LCAD(i)   = input('            Longitud de la subcadena       : ');
         MINVAL(i) = input('            Mínimo valor real a representar: ');
         MAXVAL(i) = input('            Máximo valor real a representar: ');
         presicion = (MAXVAL(i) -MINVAL(i)) / (2^LCAD(i) -1);
         fprintf(1, '%s%.10f\n\n', '            Presición de los valores reales:
', presicion);
         disp '            (Adicionar más subcadenas,    ingrese 1) ';
         resp2 =     input('            (NO adicionar más subcadenas, ingrese 2)
>> ', 's');
         i = i +1;
      end

   case {'b', 'B'}
      i = 1;
      resp2 = '1';

      while resp2 == '1',
         fprintf(1, '\n\n%s%d\n', '            Subcadena # ', i);
         LCAD(i)   = input('            Longitud de la subcadena       : ');
         MINVAL(i) = input('            Mínimo valor real a representar: ');
         presicion = input('            Presición de los valores reales: ');
         MAXVAL(i) = presicion*(2^LCAD(i) -1) +MINVAL(i);
         fprintf(1, '%s%f\n\n', '            Máximo valor real a representar: ',
MAXVAL(i));
         disp '            (Adicionar más subcadenas,    ingrese 1) ';
         resp2 =     input('            (NO adicionar más subcadenas, ingrese 2)
>> ', 's');
         i = i +1;
      end

   case {'c', 'C'}
      i = 1;
      resp2 = '1';

      while resp2 == '1',
         fprintf(1, '\n\n%s%d\n', '            Subcadena # ', i);
         LCAD(i)   = input('            Longitud de la subcadena       : ');
         MAXVAL(i) = input('            Máximo valor real a representar: ');
         presicion = input('            Presición de los valores reales: ');
         MINVAL(i) = MAXVAL(i) -presicion*(2^LCAD(i) -1);
```

```
         fprintf(1, '%s%f\n\n', '            Mínimo valor real a representar: ',
MINVAL(i));
         disp '            (Adicionar más subcadenas,    ingrese 1) ';
         resp2 =     input('            (NO adicionar más subcadenas, ingrese 2)
>> ', 's');
         i = i +1;
      end

   otherwise
      i = 1;
      resp2 = '1';

      while resp2 == '1',
         fprintf(1, '\n\n%s%d\n', '            Subcadena # ', i);
         presicion = input('            Presición de los valores reales: ');
         MINVAL(i) = input('            Mínimo valor real a representar: ');
         MAXVAL(i) = input('            Máximo valor real a representar: ');
         LCAD(i) = ceil( log2( (MAXVAL(i) - MINVAL(i))/presicion -1 ) );
         presicion = (MAXVAL(i) -MINVAL(i)) / (2^LCAD(i) -1);
         fprintf('%s%d\n', '            Longitud de la subcadena       : ',
LCAD(i));
         fprintf('%s%.10f\n\n', '            La presición cambia a          : ',
presicion);
         disp '            (Adicionar más subcadenas,    ingrese 1) ';
         resp2 =     input('            (NO adicionar más subcadenas, ingrese 2)
>> ', 's');
         i = i +1;
      end

end


%**FUNCIÓN (02) obtenerOtros
*****************************************************************
function [semilla, arch, ext] = obtenerOtros

% Declaración de las variables globales.
varsGlobales;

% Ingreso de la semilla para generar números seudoaleatorios.
fprintf(1, '\n\n\n');
disp 'INGRESAR SEMILLA PARA GENERAR NÚMEROS SEUDOALEATORIOS:';
disp ' ';
semilla = input('  Semilla (número entero positivo): ');

% Ingreso de las especificaciones de los archivos donde se almacenarán los
resultados.
arch = 'gen';
ext = 'txt';
aux = '0';
fprintf('\n\n\n');
disp 'INGRESAR ESPECIFICACIONES DE LOS ARCHIVOS:';
resp1 = '1';

while resp1 == '1',
   disp ' ';
   fprintf(1, '%s\n%s%s%s%s\n\n', '  Cada una de las generaciones se almacenará
en un archivo', '  de la forma:  ', arch, '000.', ext);
   fprintf(1, '  (Cambiar formato,    ingrese 1) \n');
   resp1 = input('  (NO cambiar formato, ingrese 2) >> ', 's');
```

```
   if resp1 == '1'
      fprintf(1, '\n');
      arch = input('  Raíz      (3 caracteres) : ', 's');
      ext = input('  Extensión (3 caracteres) : ', 's');
  end

end


%**FUNCIÓN (03) generarPob0
**********************************************************************
function cromos = generarPob0(semilla)

% Declaración de las variables globales.
varsGlobales;

% Inserción de la semilla para generar números seudoaleatorios
rand('state', semilla);

% Creación de las cadenas de genes.
for i = 1: TPOB,

   for j = 1: sum(LCAD),
      cromos(i).genotipo(j) = logical(round(rand));
   end

end

% Determinación de los fenotipos y desempeños.  Inicialización a 0 de los padres,
posición de
% entrecruzamiento y número de mutaciones  (en la generación 0, aun no tiene
sentido darles un valor).
for i = 1: TPOB,
   cromos(i).fenotipo = decodificar(cromos(i).genotipo);
   cromos(i).padres = [0 0];
   cromos(i).xPos = 0;
   cromos(i).nMut = 0;
   cromos(i).desempeno = funObj(cromos(i).fenotipo);
end


%**FUNCIÓN (04) generarPob
**********************************************************************
function [cromos, cromos0] = generarPob(cromosomas, maxD, minD, estadsGen)

% Declaración de las variables globales.
varsGlobales;


% Realizar ajustes dependiendo de si se quiere maximizar o minimizar la función
objetivo.
if sum(FIN == 'MINIMIZAR') == 9

%  Ajustes para cuando se estén calculando los pesos de la nueva generacion.
   aux = maxD;
   maxD = minD;
   minD = aux;

%  Ajustes para cuando se esté aplicando el elitismo.
   mejorDes = estadsGen.minDesemp;
   mejorFen = estadsGen.fenConMinD;
```

```
   mejorGen = estadsGen.genConMinD;
else

%  Ajustes para cuando se esté aplicando el elitismo.
   mejorDes = estadsGen.maxDesemp;
   mejorFen = estadsGen.fenConMaxD;
   mejorGen = estadsGen.genConMaxD;
end

% Almacenar la generación actual como la generación anterior.  Establecer además
los pesos
% en la ruleta. Estos pesos se almacenarán dentro de la estructura cromos0 y
dentro de un
% vector llamado pesos (para poder pasar dicho vector a la función seleccionar).
sumaPesos = 0;

for i = 1 : TPOB,
   cromos0(i).genotipo = cromosomas(i).genotipo;
   cromos0(i).fenotipo = cromosomas(i).fenotipo;
   cromos0(i).peso = 10* (cromosomas(i).desempeno -minD) / (maxD -minD);
   cromos0(i).seleccion = 0;                      % En este momento sólo se
inicializa en 0.

   sumaPesos = sumaPesos + cromos0(i).peso;
end

for i = 1 : TPOB,
   cromos0(i).peso = cromos0(i).peso *100 / sumaPesos;
   pesos(i) = cromos0(i).peso;
end

% Crear la variable cromos con la misma estructura de cromosomas
cromos = cromosomas;

% Crear cadenas de la nueva generación
for i = 1 : 2 : (TPOB -1)

%  Reproducción.  p1 y p2 serán dos enteros que indicarán la posición de los
padres en la
%  generación anterior.
   p1 = seleccionar(pesos);
   p2 = seleccionar(pesos);

%  Entrecruzamiento
   [cromos(i).genotipo, cromos(i+1).genotipo, cromos(i).xPos, cromos(i+1).xPos] =
entrecruzar(cromos0(p1).genotipo, cromos0(p2).genotipo);

%  Mutación
   [cromos(  i).genotipo, cromos(  i).nMut] = mutar(cromos(  i).genotipo);
   [cromos(i+1).genotipo, cromos(i+1).nMut] = mutar(cromos(i+1).genotipo);

%  Actualizar la cuenta cromos0.seleccion.
   cromos0(p1).seleccion = cromos0(p1).seleccion +1 *100 /TPOB;
   cromos0(p2).seleccion = cromos0(p2).seleccion +1 *100 /TPOB;

%  Determinar fenotipos de los dos nuevos cromosomas.
   cromos(i  ).fenotipo = decodificar(cromos(i  ).genotipo);
   cromos(i+1).fenotipo = decodificar(cromos(i+1).genotipo);

%  Determinar los padres de los dos nuevos cromosomas.
   cromos(  i).padres(1) = p1;
```

```
   cromos(  i).padres(2) = p2;
   cromos(i+1).padres(1) = p2;
   cromos(i+1).padres(2) = p1;

%  Determinación de los desempeños de la nueva generación,
%  pero llevando la cuenta de cuál es el peor.
   if i == 1
      peor = i;
      peorDes = funObj(cromos(i).fenotipo);
   end

   if sum(FIN == 'MAXIMIZAR') == 9
      cromos(i).desempeno = funObj(cromos(i).fenotipo);

      if cromos(i).desempeno < peorDes
         peor = i;
         peorDes = cromos(i).desempeno;
      end

      cromos(i+1).desempeno = funObj(cromos(i+1).fenotipo);

      if cromos(i+1).desempeno < peorDes
         peor = i +1;
         peorDes = cromos(i+1).desempeno;
      end

   else
      cromos(i).desempeno = funObj(cromos(i).fenotipo);

      if cromos(i).desempeno > peorDes
         peor = i;
         peorDes = cromos(i).desempeno;
      end

      cromos(i+1).desempeno = funObj(cromos(i+1).fenotipo);

      if cromos(i+1).desempeno > peorDes
         peor = i +1;
         peorDes = cromos(i+1).desempeno;
      end
   end

end

% Reemplazar el peor cromosoma con el mejor de la generación anterior.
cromos(peor).desempeno = mejorDes;
cromos(peor).fenotipo = mejorFen;
cromos(peor).genotipo = mejorGen;
cromos(peor).padres = [0 0];


%**FUNCIÓN (05) generarEstadsGen
************************************************************
function estads = generarEstadsGen(cromos)

% Declaración de las variables globales.
varsGlobales;

% Determinación del máximo y mínimo desempeños en la población, de los fenotipos
que han
```

```
% generado dichos desempeños y del # de entrecruzamientos y mutaciones ocurridos
en la población.
estads.maxDesemp = cromos(1).desempeno;
estads.fenConMaxD = cromos(1).fenotipo;
estads.minDesemp = cromos(1).desempeno;
estads.fenConMinD = cromos(1).fenotipo;
suma = cromos(1).desempeno;

% Las siguientes dos estadísticas no se imprimirán en los resultados, sino que
son
% simplemente tenidas en cuenta en la función "generarPob", cuando se aplica el
elitismo.
estads.genConMaxD = cromos(1).genotipo;
estads.genConMinD = cromos(1).genotipo;

if cromos(1).xPos > 0
   estads.nX = 1;
else
   estads.nX = 0;
end

estads.nMut = cromos(1).nMut;

for i = 2: TPOB,
   suma = suma + cromos(i).desempeno;

   if cromos(i).desempeno > estads.maxDesemp
      estads.maxDesemp = cromos(i).desempeno;
      estads.fenConMaxD = cromos(i).fenotipo;
      estads.genConMaxD = cromos(i).genotipo;
   end

   if cromos(i).desempeno < estads.minDesemp
      estads.minDesemp = cromos(i).desempeno;
      estads.fenConMinD = cromos(i).fenotipo;
      estads.genConMinD = cromos(i).genotipo;
   end

   if cromos(i).xPos > 0
      estads.nX = estads.nX + 1;
   end

   estads.nMut = estads.nMut +cromos(i).nMut;

end

% Determinación del promedio de desempeño en la población
estads.medDesemp = suma /TPOB;


%**FUNCIÓN (06) generarEstadsAG
*************************************************************
function estads = generarEstadsAG(estadsAG, estadsGen)

% Declaración de las variables globales.
varsGlobales;

% Determinación del desempeño máximo y mínimo en el AG.
if estadsGen.maxDesemp > estadsAG.maxDesemp
   estads.maxDesemp = estadsGen.maxDesemp;
   estads.fenConMaxD = estadsGen.fenConMaxD;
```

```
else
   estads.maxDesemp = estadsAG.maxDesemp;
   estads.fenConMaxD = estadsAG.fenConMaxD;
end

if estadsGen.minDesemp < estadsAG.minDesemp
   estads.minDesemp = estadsGen.minDesemp;
   estads.fenConMinD = estadsGen.fenConMinD;
else
   estads.minDesemp = estadsAG.minDesemp;
   estads.fenConMinD = estadsAG.fenConMinD;
end

% Obtención del # de entrecruzamientos y mutaciones realizadas hasta el momento.
estads.nX = estadsAG.nX + estadsGen.nX;
estads.nMut = estadsAG.nMut + estadsGen.nMut;


%**FUNCIÓN (07) reporte0
**********************************************************************
function reporte0(cromos, estads, arch, ext, sem)

% Declaración de las variables globales.
varsGlobales;

% Establecer el nombre del archivo y abrirlo.
archivo = [arch '000.' ext];
unit = fopen(archivo, 'w');

% Borrar pantalla
clc

% Mediante el mismo código, se imprimirán los datos de la población
% inicial en un archivo, y luego en pantalla.
for u = 1 : (unit -1) : unit

%  Impresión del encabezado.
   fprintf(u, '\n%s%s\n\n', 'Archivo: ', archivo);
   fprintf(u, 'ALGORITMO GENÉTICO - DATOS DE ENTRADA Y POBLACIÓN INICIAL');
   fprintf(u, '\n\n\n');
   fprintf(u, '%s%s%s\n', '  FUNCIÓN A ', FIN, ':');
   fprintf(u, '   %s\n   %s\n   %s\n\n', FTXT1, FTXT2, FTXT3);
   fprintf(u, '  +-------------------------------------------------------------
+\n');
   fprintf(u, '  |                     POBLACIÓN INICIAL                       
|\n');
   fprintf(u, '  +---+---------------------------------+----------+----------
+\n');
   fprintf(u, '  | # |           Cromosoma             |    X     |   F(X)   
|\n');
   fprintf(u, '  +---+---------------------------------+----------+----------
+\n');

%  Impresión de las líneas de la tabla. Para posibilitar la impresión de la
cadena, se
%  requiere convertirlo en una variable de tipo VECTOR DE VALORES LÓGICOS a
VECTOR DE CARACTERES.
   for i = 1 : TPOB

      for j = 1: length(cromos(i).genotipo),
         cadena(j)= num2str(cromos(i).genotipo(j));
```

```
      end

      fprintf(u, '%s%3d%c%33s%c%10.4g%c%10.4g%c\n', '  |', i, '|', cadena, '|',
cromos(i).fenotipo(1), '|', cromos(i).desempeno, '|');

      for j = 2 : length(LCAD)
         fprintf(u, '%s%10.4g%s\n', '  |   |                                |',
cromos(i).fenotipo(j),'|          |');
      end
   end

%  Impresión de la parte final de la tabla.
   fprintf(u, '  +---+--------------------------------+----------+----------
+\n\n\n');

%  Impresión de las estadísticas.
   fprintf(u, '  +-------------------------------------------------------------
+\n');
   fprintf(u, '  |          ESTADÍSTICAS DE LA POBLACIÓN INICIAL
|\n');
   fprintf(u, '  +-------------------------------------------------------------
+\n');
   fprintf(u, '%s%10.4g\n', '    Fpromedio : ', estads.medDesemp);
   fprintf(u, '%s%10.4g\n', '    Fmax(x1)  : ', estads.maxDesemp);
   fprintf(u, '%s%s\n', '    x1        : ', num2str(estads.fenConMaxD));
   fprintf(u, '%s%10.4g\n', '    Fmin(x2)  : ', estads.minDesemp);
   fprintf(u, '%s%s\n', '    x2        : ', num2str(estads.fenConMinD));
   fprintf(u, '  +-------------------------------------------------------------
+\n\n\n');


%  Impresión de los parámetros del AG.
   fprintf(u, '  +-------------------------------------------------------------
+\n');
   fprintf(u, '  |             PARÁMETROS DEL ALGORITMO GENÉTICO
|\n');
   fprintf(u, '  +-------------------------------------------------------------
+\n');
   fprintf(u, '%s%d\n', '    Tamaño de la Población     : ', TPOB);
   fprintf(u, '%s%d\n', '    Número de Generaciones     : ', MAXGEN);
   fprintf(u, '%s%f\n', '    Prob. de Entrecruzamiento  : ', PX);
   fprintf(u, '%s%f\n', '    Prob. de Mutación          : ', PMUT);
   fprintf(u, '%s%s\n', '    Longitud de las Subcadenas : ', num2str(LCAD));
   fprintf(u, '%s%s\n', '    Máximo Valor Real          : ', num2str(MAXVAL));
   fprintf(u, '%s%s\n', '    Mínimo Valor Real          : ', num2str(MINVAL));
   fprintf(u, '  +-------------------------------------------------------------
+\n\n');

   fprintf(u, '%s%d\n\n\n', '  Semilla = ', sem);

end

fclose(unit);
input('(Ingrese una letra para continuar) >> ');


%**FUNCIÓN (08) reporte
*********************************************************************
function reporte(gen, cromos, cromos0, estadsGen, estadsAG, arch, ext)

% Declaración de las variables globales.
```

```
varsGlobales;

% Establecer el nombre del archivo y abrirlo.
archivo = [arch num2str(gen) '.' ext];
u = fopen(archivo, 'w');

% Impresión del encabezado.
fprintf(u, '\n%s%s\n\n', 'Archivo: ', archivo);
fprintf(u, '%s%s', 'ALGORITMO GENÉTICO - ESTABLECIMIENTO DE LA GENERACIÓN ',
num2str(gen));
fprintf(u, '\n\n');
fprintf(u, '%s%s%s\n', '  FUNCIÓN A ', FIN, ':');
fprintf(u, '    %s\n    %s\n    %s\n\n', FTXT1, FTXT2, FTXT3);
fprintf(u, '  +---------------------------------------------------------------+----
-----------------------------------------------------------+\n');
fprintf(u, '%s%3s%s%3s%s\n', '  |                      GENERACIÓN ', num2str(gen-
1), '                          |                          GENERACIÓN ',
num2str(gen), '                              |');
fprintf(u, '  +---+---------------------------------+----------+------+-----+---
+---+--+--+---------------------------------+----------+----------+\n');
fprintf(u, '  | # |           Cromosoma             |     x    |Peso%%
|Sele%%|Padres |XP|NM|           Cromosoma             |     x    |    F(x)
|\n');
fprintf(u, '  +---+---------------------------------+----------+------+-----+---
+---+--+--+---------------------------------+----------+----------+\n');

%  Impresión de las líneas de la tabla. Para posibilitar la impresión de la
cadena, se
%  requiere convertirlo en una variable de tipo VECTOR DE VALORES LÓGICOS a
VECTOR DE CARACTERES
for i = 1 : TPOB,

   for j = 1: length(cromos(i).genotipo),
      cad0(j)= num2str(cromos0(i).genotipo(j));
      cad(j)= num2str(cromos(i).genotipo(j));
   end

   fprintf(u,
'%s%3d%c%33s%c%10.4g%c%6.3f%c%5.2f%c%3d%c%3d%c%2d%c%2d%c%33s%c%10.4g%c%10.4g%c\n'
, '  |', i, '|', cad0, '|', cromos0(i).fenotipo(1), '|', cromos0(i).peso, '|',
cromos0(i).seleccion, '|', cromos(i).padres(1), '|', cromos(i).padres(2), '|',
cromos(i).xPos, '|', cromos(i).nMut, '|', cad, '|', cromos(i).fenotipo(1), '|',
cromos(i).desempeno, '|');

   for j = 2 : length(LCAD),
      fprintf(u, '%s%10.4g%s%10.4g%s\n', '  |   |
|', cromos0(i).fenotipo(j), '|      |     |   |   |  |  |
|', cromos(i).fenotipo(j), '|          |');
   end

end

% Impresión de la parte final del cuadro.
fprintf(u, '  +---+---------------------------------+----------+------+-----+---
+---+--+--+---------------------------------+----------+----------+\n');
fprintf(u, '   Peso%%= Peso dentro de la ruleta que se empleará para seleccionar
los padres.\n');
fprintf(u, '   Sele%%= %% que indica el # de veces en que fue efectivamente
seleleccionado como padre.\n');
fprintf(u, '   XP= Posición donde hubo entrecruzamiento.  Si no hubo
entrecruzamiento, entonces XP= 0.\n');
```

```
fprintf(u, '   NM= Número de mutaciones.\n');

% Impresión de las estadísticas de la generación.
fprintf(u, '\n\n');
fprintf(u, '  +-------------------------------------------------------------+\n');
fprintf(u, '%s%3s%s\n', '  |           ESTADÍSTICAS DE LA GENERACIÓN ',
num2str(gen), '                |');
fprintf(u, '  +-------------------------------------------------------------+\n');
fprintf(u, '%s%10.4g\n', '    Fpromedio        : ', estadsGen.medDesemp);
fprintf(u, '%s%10.4g\n', '    Fmax(x1)         : ', estadsGen.maxDesemp);
fprintf(u, '%s%s\n', '    x1               : ', num2str(estadsGen.fenConMaxD));
fprintf(u, '%s%10.4g\n', '    Fmin(x2)         : ', estadsGen.minDesemp);
fprintf(u, '%s%s\n', '    x2               : ', num2str(estadsGen.fenConMinD));
fprintf(u, '%s%d\n', '    Entrecruzamientos: ', estadsGen.nX);
fprintf(u, '%s%d\n', '    Mutaciones       : ', estadsGen.nMut);
fprintf(u, '  +-------------------------------------------------------------+\n');

% Impresión de las estadísticas del AG.
fprintf(u, '\n\n  +-------------------------------------------------------------
+\n');
fprintf(u, '%s%3s%s\n', '  |           ESTADÍSTICAS DEL AG HASTA LA GENERACIÓN ',
num2str(gen), '      |');
fprintf(u, '  +-------------------------------------------------------------+\n');
fprintf(u, '%s%10.4g\n', '    Fmax(x1)         : ', estadsAG.maxDesemp);
fprintf(u, '%s%s\n', '    x1               : ', num2str(estadsAG.fenConMaxD));
fprintf(u, '%s%10.4g\n', '    Fmin(x2)         : ', estadsAG.minDesemp);
fprintf(u, '%s%s\n', '    x2               : ', num2str(estadsAG.fenConMinD));
fprintf(u, '%s%d\n', '    Entrecruzamientos: ', estadsAG.nX);
fprintf(u, '%s%d\n', '    Mutaciones       : ', estadsAG.nMut);
fprintf(u, '  +-------------------------------------------------------------+\n');

fclose(u);


%**FUNCIÓN (09) reporteF
**********************************************************************
function reporteF(gen, cromos, estadsGen, estadsAG, arch, ext)

% Declaración de las variables globales.
varsGlobales;

% Borrar pantalla
clc;

% Establecer el nombre del archivo.
archivo = [arch num2str(gen) '.' ext];

% Impresión del encabezado.
fprintf(1, '%s%s%s\n', 'Archivo: ', archivo, '   (Versión Simplificada)');
fprintf(1, '\n\n');
fprintf(1, '%s%d', 'ALGORITMO GENÉTICO - RESULTADOS FINALES');
fprintf(1, '\n\n');
fprintf(1, '%s%s%s\n', '  FUNCIÓN A ', FIN, ':');
fprintf(1, '    %s\n    %s\n    %s\n\n', FTXT1, FTXT2, FTXT3);
fprintf(1, '  +-------------------------------------------------------------+\n');
fprintf(1, '%s%3s%s\n', '  |                        GENERACIÓN ', num2str(gen), '
|');
fprintf(1, '  +---+---------------------------------+----------+----------+\n');
fprintf(1, '  | # |           Cromosoma             |     x    |   F(x)   |\n');
fprintf(1, '  +---+---------------------------------+----------+----------+\n');
```

```
%  Impresión de las líneas de la tabla. Para posibilitar la impresión de la
cadena, se
%  requiere convertirlo en una variable de tipo VECTOR DE VALORES LÓGICOS a
VECTOR DE VALORES
%  DE CARACTERES.
for i = 1 : TPOB,

   for j = 1: length(cromos(i).genotipo),
     cadena(j)= num2str(cromos(i).genotipo(j));
   end

   fprintf(1, '%s%3d%c%33s%c%10.4g%c%10.4g%c\n', '  |', i, '|', cadena, '|',
cromos(i).fenotipo(1), '|', cromos(i).desempeno, '|');

   for j = 2 : length(LCAD),
      fprintf(1, '%s%10.4g%s\n', '  |   |                                 |',
cromos(i).fenotipo(j), '|          |');
  end

end

% Impresión de la parte final del cuadro.
fprintf(1, '  +---+---------------------------------+----------+----------+');

% Impresión de las estadísticas de la generación.
fprintf(1, '\n\n\n');
fprintf(1, '  +-------------------------------------------------------------+\n');
fprintf(1, '%s%3s%s\n', '  |            ESTADÍSTICAS DE LA GENERACIÓN ',
num2str(gen), '               |');
fprintf(1, '  +-------------------------------------------------------------+\n');
fprintf(1, '%s%10.4g\n', '    Fpromedio        : ', estadsGen.medDesemp);
fprintf(1, '%s%10.4g\n', '    Fmax(x1)         : ', estadsGen.maxDesemp);
fprintf(1, '%s%s\n', '    x1               : ', num2str(estadsGen.fenConMaxD));
fprintf(1, '%s%10.4g\n', '    Fmin(x2)         : ', estadsGen.minDesemp);
fprintf(1, '%s%s\n', '    x2               : ', num2str(estadsGen.fenConMinD));
fprintf(1, '%s%d\n', '    Entrecruzamientos: ', estadsGen.nX);
fprintf(1, '%s%d\n', '    Mutaciones       : ', estadsGen.nMut);
fprintf(1, '  +-------------------------------------------------------------
+\n\n\n');

% Impresión de las estadísticas del AG.
fprintf(1, '\n\n');
fprintf(1, '  +-------------------------------------------------------------+\n');
fprintf(1, '%s%3s%s\n', '  |          ESTADÍSTICAS DEL AG HASTA LA GENERACIÓN ',
num2str(gen), '       |');
fprintf(1, '  +-------------------------------------------------------------+\n');
fprintf(1, '%s%10.4g\n', '    Fmax(x1)         : ', estadsAG.maxDesemp);
fprintf(1, '%s%s\n', '    x1               : ', num2str(estadsAG.fenConMaxD));
fprintf(1, '%s%10.4g\n', '    Fmin(x2)         : ', estadsAG.minDesemp);
fprintf(1, '%s%s\n', '    x2               : ', num2str(estadsAG.fenConMinD));
fprintf(1, '%s%d\n', '    Entrecruzamientos: ', estadsAG.nX);
fprintf(1, '%s%d\n', '    Mutaciones       : ', estadsAG.nMut);
fprintf(1, '  +-------------------------------------------------------------
+\n\n');


%**FUNCIÓN (10) seleccionar
*********************************************************************
function p = seleccionar(pesos)

% Declaración de las variables globales.
```

```
varsGlobales;

p = 1;
valRuleta = pesos(1);
valAzar = floor(rand *100);

while valRuleta < valAzar,
   p = p +1;
   valRuleta = valRuleta +pesos(p);
end


%**FUNCIÓN (11) entrecruzar
*****************************************************************
function [cadHijo1, cadHijo2, xPos1, xPos2] = entrecruzar(cadPadre1, cadPadre2)

% Declaración de las variables globales.
varsGlobales;

% Realizar entrecruzamiento dependiendo de la probabilidad PX.  En caso de que no
se realice
% el entrecruzamiento, xPos1 y xPos2 serán iguales a 0.
long = sum(LCAD);

if rand <= PX

%  Determinar los dos puntos de referencia para hacer el entrecruzamiento.  xPos1
no podrá ser
%  igual a xPos2.
   xPos1 = 1 + mod(floor(rand*10000), long);
   xPos2 = 1 + mod(floor(rand*10000), long);

   while xPos2 == xPos1,
      xPos2 = 1 + mod(floor(rand*10000), long);
   end

%  Proceder a hacer el entrecruzamiento dependiendo de si xPos1 es menor a que
xPos2 o no.
   if (xPos1 < xPos2)

      for i = 1 : (xPos1 -1),
         cadHijo1(i) = cadPadre1(i);
         cadHijo2(i) = cadPadre2(i);
      end

      for i = xPos1 : (xPos2 -1),
         cadHijo1(i) = cadPadre2(i);
         cadHijo2(i) = cadPadre1(i);
      end

      for i = xPos2 : long,
         cadHijo1(i) = cadPadre1(i);
         cadHijo2(i) = cadPadre2(i);
      end

   else

      for i = 1 : (xPos2 -1),
         cadHijo1(i) = cadPadre2(i);
         cadHijo2(i) = cadPadre1(i);
      end
```

```
      for i = xPos2 : (xPos1 -1),
         cadHijo1(i) = cadPadre1(i);
         cadHijo2(i) = cadPadre2(i);
      end

      for i = xPos1 : long,
         cadHijo1(i) = cadPadre2(i);
         cadHijo2(i) = cadPadre1(i);
      end

   end

else
   xPos1 = 0;
   xPos2 = 0;
   cadHijo1 = cadPadre1;
   cadHijo2 = cadPadre2;
end


%**FUNCIÓN (12) mutar
*************************************************************************
function [cadMutada, nMuts] = mutar(cadena)

% Declaración de las variables globales.
varsGlobales;

% Inicializar nMuts y cadMutada
nMuts = 0;
cadMutada = cadena;

% Proceder a hacer las mutaciones
for i = 1 : sum(LCAD),

    if rand <= PMUT
      cadMutada(i) = ~cadena(i);
      nMuts = nMuts +1;
    end

end


%**FUNCIÓN (13) decodificar
*********************************************************************
function vectReales = decodificar(genotipo)

% Declaración de las variables globales.
varsGlobales;


for i = 1: length(LCAD),

%  Separación de la subcadena
   inicio =  sum(LCAD( 1: (i -1) )) +1;
   finCad =  inicio +LCAD(i) -1;
   subcadena = genotipo(inicio:finCad);

% Conversión a un número entero positivo
   n = LCAD(i);
   entero = 0;
```

```
   for j = 1 : n,
      entero = entero + subcadena(j)*2^(n -j);
   end

%  Reescalamiento de acuerdo a los rangos indicados en los parámetros del AG:
MAXVAL Y MINVAL
   vectReales(i) = MINVAL(i) + (MAXVAL(i) -MINVAL(i)) *entero /(2^LCAD(i) -1);
end
```

funObj.m

```
function z = funObj(vector)
% Esta funcion obtiene la norma cuadrática de la función thomas


% Desglosamiento del vector en los diferentes parámetros del modelo.
a = vector(1);
d = vector(2);
c = vector(3);
b = vector(4);
Sw0 = vector(5);
Sg0 = vector(6);

% Ingresar aquí las precipitaciones (en mm)
P = [40.62 27.17 92.96 53.4 70.94 51.8 42.85 35.29 77.56 114.7 89.93 48.55];

% Ingresar aquí las evapotranspiraciones potenciales (en mm)
PE = [49.4 35.94 38.87 55.03 47.59 36.45 45.7 35.06 35.92 55.21 55.73 46.54];

% Ingresar aquí los caudales observados  (en m3/s)
QrM3s = [0.264 0.585 0.603 0.736 0.579 0.533 0.395 0.339 0.469 1.781 1.113
0.411];

% Ingresar el área
area = 162.2;


% Determinacion de variables auxiliares.
n = length(P);
suma = 0;

% Cálculo de la norma cuadrática

for i = 1 : n
   QrMm = 2592 *QrM3s(i) /area;
   [Qs Sw Sg] = thomas(P(i), PE(i), a, b, c, d, Sw0, Sg0);
   suma = suma + (Qs -QrMm)^2;
   Sw0 = Sw;
   Sg0 = Sg;
end

z = sqrt(suma/n);



function [Q, Sw, Sg] = thomas(P, EP, a, b, c, d, Sw0, Sg0)
% Aquí se calcula el caudal, el contenido de humedad en el suelo y almacenamiento
de agua subterránea.
```

```
W = P + Sw0;

Y = (W +b) /(2*a) -sqrt( ((W +b) /(2*a))^2 - W *b /a );

Sw = Y*exp(-EP /b);

Ro = (1 -c)*(W -Y);

Rg = c*(W -Y);

Sg = (Rg +Sg0) /(d +1);

Qg = d*Sg;

Q = Ro + Qg;
```

funObjDescrip.m

```
function [ftxt1, ftxt2, ftxt3, fin] = funObjDescrip

ftxt1 = 'Norma Cuadrática';
ftxt2 = '   (para la calibración del modelo de Thomas aplicado a la subcuenca
SUBACHOQUE - 1997-1999)';
ftxt3 = 'z = sqrt{ sumatoria[ i, 1, n, ( Q(i) - Qs(i) ) ] /n }';
fin = 'MINIMIZAR';
```

varsGlobales.m

```
global TPOB MAXGEN PX PMUT LCAD MINVAL MAXVAL

global FTXT1 FTXT2 FTXT3 FIN
```

ANEXO B
RESULTADOS CUENCA DE SUBACHOQUE

Archivo: sub000.txt

ALGORITMO GENÉTICO - DATOS DE ENTRADA Y POBLACIÓN INICIAL

FUNCIÓN A MINIMIZAR:
Norma Cuadrática
(para la calibración del modelo de Thomas aplicado a la subcuenca SUBACHOQUE - 1997-1999)
z = sqrt{ sumatoria[i, 1, n, (Q(i) - Qs(i))] /n }

POBLACIÓN INICIAL

#	Cromosoma	X	F(X)
1	011000111000000011111000000111100110100001101001110110 01	0.8776	64.13
		0.5039	
		0.791	
		282.8	
		50.88	
		462.8	
2	101000101110010110101101001010100011010001100010001 0111	0.9271	72.7
		0.8975	
		0.6355	
		226.9	
		274	
		273	
3	101011000001001001111111101100100001010010011011100 1011	0.9349	39.25
		0.07229	
		0.8965	

		53.91	
		287.7	
		449.1	
4	10101011000000100101110101111110110110001000111100000101	0.9341	10.36
		0.009789	
		0.4149	
		326	
		34.25	
		380.6	
5	00100011010111000100011010010111011000101100111110100000	0.8275	77.52
		0.3613	
		0.09327	
		167	
		175.1	
		407	
6	01011110001000101100101101001111000101011101100110010001	0.8737	57.75
		0.1367	
		0.1592	
		330.7	
		365.9	
		196.7	
7	00100011100100001010001010001101001000100000110100100 11	0.8275	37.31
		0.5654	
		0.4861	
		289.5	
		128.2	
		143.8	
8	10001110100100000010011010011000010110101100001001000 00	0.9114	57.4
		0.5635	
		0.5432	
		187.7	
		172.2	
		281.8	
9	10001001111100001111100011110100111100110111111110111000	0.9075	102.1
		0.9414	
		0.8016	
		115.1	
		218.2	

		430.5		
10	01110101000110001100011001101111011101101100101100111110	0.8918	76.09	
		0.09768		
		0.09064		
		338.7		
		424.7		
		405.1		
11	00110101101010010010000011000011111011101100011000010011	0.8416	71.93	
		0.6611		
		0.4615		
		91.84		
		347.4		
		18.59		
12	00100110111100101100011000011011010101000010000010000000	0.8298	50.4	
		0.9492		
		0.08624		
			250.9	
		258.3		
		62.62		
13	10000010001100001111001000000011000111100110100110111100	0.902	68.75	
		0.1914		
		0.704		
		75.87		
		401.2		
		215.3		
14	10100111000011001110001000100100011011101111110100100001	0.931	63.84	
		0.0508		
		0.4808		
		103.8		
		437.4		
		141.9		
15	01101100110011111111011001000011100010110011101100101010	0.8847	95.28	
		0.8125		
		0.7638		
		85.19		
		201.6		
		396.3		
16	11110000010001100100000100001000000101000000101001101101	0.9882	22.01	

		0.2754	
		0.01506	
		183.7	
		4.892	
		106.7	
17	10100101000111011111001110011000101100100011110001101111	0.9294	14.13
		0.1172	
		0.726	
		195	
		139.9	
		53.82	
18	11011111100111101010111101100111011111101011000010101110	0.9749	67.46
		0.6201	
		0.6671	
		169	
		418.8	
		84.15	
19	10110000101110000111101010111011101110111010100101010100	0.938	42.38
		0.7207	
		0.8262	
		259.5	
		229	
		166.3	
20	01110011011000100101100101101011110100011000001111011000	0.8902	66.24
		0.3848	
		0.3578	
		261.5	
		93.93	
		481.4	
21	00101111010110000010111011110110000110000011101101110010	0.8369	56.01
		0.3447	
		0.661	
		139.7	
		13.7	
		431.5	
22	10011010110001000110000100001110010000000010010111001110	0.9208	37.39
		0.7676	
		0.465	

		313.4	
		8.806	
		226	
23	011100001110101111010010010101011001011111010010100 1011	0.8878	154.3
		0.9219	
		0.2585	
		128.4	
		489.2	
		323.9	
24	111011110010011001000001001011010000101000001111100 0001	0.9875	41.36
		0.1504	
		0.01682	
		287.5	
		128.2	
		439.3	
25	011011100100001111101100000001111101011110010110111 1101	0.8863	94.76
		0.2656	
		0.6197	
		176.3	
		474.6	
		372.8	
26	001000111101001100101110111111011101001011110101011 1001	0.8275	48.03
		0.8252	
		0.661	
		304.1	
		184.9	
		181	
27	110001000111111101000011101110110110010001011110010 0110	0.9537	67.13
		0.4981	
		0.05285	
		252.2	
		273	
		287.7	
28	001100010100110101011010001110111010010101000100000 1000	0.8384	44.58
		0.3028	
		0.3692	
		257.5	
		329.7	

		7.828	
29	00110000001110100001001010000111010111101100111100100000	0.8376	107.5
		0.2276	
		0.2611	
		166.4	
		425.6	
		391.4	
30	00001110111100111100001101111101111110100000100001011101	0.811	31.96
		0.9531	
		0.04933	
		307.4	
		127.2	
		44.03	
31	01000001000111000111011100010110110010101111111100010000	0.851	37.03
		0.1113	
		0.7752	
		154.4	
		249.5	
		266.1	
32	00010011011111010101011011000010011111100010010010000111	0.8149	95.83
		0.4902	
		0.3209	
		62.56	
		384.5	
		69.47	
33	00010100110110011001100010011101101001101101101111110100	0.8157	136.4
		0.8506	
		0.3464	
		300.1	
		428.6	
		489.2	
34	11101100010010110111000010010101101011100111010111110011	0.9851	72.31
		0.2949	
		0.6838	
		130.4	
		404.1	
		237.8	
35	01001111001111011000110011010110000000110100010010000010	0.862	45.04

		0.2412	
		0.1812	
		137.7	
		204.5	
		64.58	
36	00100110100000110101001011100100101001001000100001011110	0.8298	65.15
		0.5137	
		0.2664	
		108.5	
		283.8	
		45.01	
37	01111111000111101111100010001111000011000010011001011111	0.8996	32.92
		0.1211	
		0.7954	
		330	
		259.3	
		296.5	
38	01110101010010011011100001100010010000001001001010001111	0.8918	45.2
		0.2881	
		0.7937	
		57.91	
		35.23	
		320	
39	01011110000010001111000011010001000101111000100101111111	0.8737	51.77
		0.03518	
		0.6873	
		32.62	
		471.6	
		186.9	
40	11100101000001100000011101100000001010000110100011110101	0.9796	43.01
		0.02444	
		0.1047	
		13.33	
		25.44	
		114.5	
41	01101000101011110010101011010000100011000010010101111101	0.8816	88.97
		0.6846	
		0.603	

		21.31	
		259.3	
		184.9	
42	1000111101000111100011011100000000111011100110010001000	0.9122	84.43
		0.2803	
		0.1943	
		14.66	
		225	
		133.1	
43	0110111001110001110100110001111100010100000111110110000	0.8863	77.24
		0.4453	
		0.269	
		330.7	
		257.3	
		422.7	
44	1111011111111111011011010011110000010110110110110000100	0.9937	105.8
		0.998	
		0.6364	
		266.8	
		428.6	
		379.6	
45	0100000100110101110111101110011011001001010010001000011	0.851	25.14
		0.211	
		0.4351	
		154.4	
		80.23	
		65.56	
46	0011110110010100010001011000101001010101010010110110000	0.8478	110.5
		0.5801	
		0.07833	
		229.6	
		330.7	
		422.7	
47	1010110100000110000011011001110100011000000011011100110	0.9357	17.37
		0.02444	
		0.1917	
		288.8	
		2.935	

		225	
48	10110111111011101100110111011011000100011010111000001 00	0.9435	48.96
		0.9336	
		0.1952	
		245.5	
		104.7	
		254.4	
49	11101000011010010100101010001111100100100111000010110 11	0.982	18.82
		0.4121	
		0.1486	
		341.4	
		152.6	
		89.04	
50	10011101001011010011011100100100111100100001010011110 10	0.9231	24.38
		0.1768	
		0.7761	
		115.1	
		130.1	
		119.4	

ESTADÍSTICAS DE LA POBLACIÓN INICIAL

Fpromedio :	61.32444					
Fmax(x1) :	154.3094					
x1 :	0.8878431	0.9218768	0.2584848	128.4344	489.2368	323.8748
Fmin(x2) :	10.36388					
x2 :	0.9341176	0.009788856	0.4149091	326.047	34.24658	380.6262

PARÁMETROS DEL ALGORITMO GENÉTICO

Tamaño de la Población : 50
Número de Generaciones : 1000

```
  Prob. de Entrecruzamiento  : 0.800000
  Prob. de Mutación          : 0.010000
  Longitud de las Subcadenas : 8 10 10  9  9  9
  Máximo Valor Real          : 1             1           0.9           350           500           500
  Mínimo Valor Real          : 0.8       0.001       0.001          10           0           0
+-------------------------------------------------------------------------------+

Semilla = 49
```

Archivo: sub1000.txt

ALGORITMO GENÉTICO - ESTABLECIMIENTO DE LA GENERACIÓN 1000

FUNCIÓN A MINIMIZAR:
Norma Cuadrática
(para la calibración del modelo de Thomas aplicado a la subcuenca SUBACHOQUE - 1997-1999)
z = sqrt{ sumatoria[i, 1, n, (Q(i) - Qs(i))] /n }

GENERACIÓN 999									GENERACIÓN 1000		
#	Cromosoma	x	Peso%	Sele%	Padres	XP	NM		Cromosoma	x	F(x)
1	11111111100000100011101010101111111110100100011110010011	1	2.128	4.00	49	45	10	1	11111111100000001001100100010011110100001000111111100001	1	3.879
		0.0176								0.004906	
		0.7506								0.7058	
		348.7								172.3	
		69.47								69.47	
		284.7								470.6	
2	11111110000000100011101011000111010101001000001111110011	0.9976	2.232	4.00	45	49	20	1	11111101000001101011010100001110101010010011011000011	0.9984	3.375
		0.0176								0.02639	
		0.7532								0.7462	
		322.1								322.1	
		63.6								76.32	
		488.3								378.7	
3	11111110000000110001100011011011010101011011111011111101	0.9976	0.903	0.00	14	37	28	0	11111111100000110001101000100111110011001000101111101011	1	3.522
		0.02444								0.02444	
		0.6996								0.7357	
		151.7								342	
		218.2								67.51	
		247.6								480.4	
4	11111101000001010011010000111111011010010001111111101000	0.9984	2.196	4.00	37	14	31	0	11111111100000101011101011001111010110001000111110000011	1	2.279
		0.02053								0.02151	
		0.7348								0.7541	
		338								322.7	
		69.47								69.47	
		477.5								378.7	
5	11111101000000001001101010111111111010100100010111101111	0.9984	1.839	4.00	4	4	0	0	11111101000001010011010000111111011010010001111111101000	0.9984	3.183
		0.004906								0.02053	
		0.7524								0.7348	
		343.3								338	
		67.51								69.47	
		468.7								477.5	
6	11111111100000110011101000101111111001100100010111110101	1	2.108	0.00	4	4	0	0	11111101000001010011010000111111011010010001111111101000	0.9984	3.183
		0.02541								0.02053	
		0.7365								0.7348	
		342								338	
		67.51								69.47	
		480.4								477.5	
7	11111111100000110011101000101111111011100100010111110101	1	2.114	2.00	30	42	49	0	11111111100000100011101010101111111110100100011111110101	1	1.989
		0.02541								0.0176	
		0.7365								0.7506	
		344.7								348.7	
		67.51								69.47	
		480.4								480.4	
8	11111111100000110111100000010011110000001011111010000001	1	2.064	0.00	42	30	45	1	11111111100000101001100011001111110001001001101010000011	1	5.162
		0.02737								0.02053	
		0.6777								0.6979	
		169.7								340.7	
		92.95								75.34	
		126.2								128.2	
9	11111110000000100011101011010110110001011100111011101011	0.9976	2.298	2.00	15	36	29	0	11111101000001001011010110001110101100010001011011100011	0.9984	2.737
		0.0176								0.01858	
		0.755								0.7532	
		298.1								322.7	
		226								67.51	
		229.9								363	

10	11111000000000100011101011010110110001011100111011010001	0.9945 0.0176 0.755 298.1 226 204.5	2.269	2.00	36	15	45	1	11111110000000100001111011010111010001011000101111000000	0.9992 0.01662 0.8675 319.4 192.8 438.4	3.814
11	11011110000000100001101011010110110001011100111011010000	0.9741 0.01662 0.755 298.1 226 203.5	2.064	0.00	33	44	21	0	11111101000000100011101010101111110100001000101111010001	0.9984 0.0176 0.7506 342.7 67.51 455	2.601
12	11111100000000100011101011010111010110001000111111000000	0.9976 0.0176 0.755 322.7 69.47 438.4	2.253	2.00	0	0	13	3	11111111000001000111010101011111111010010001011111101011	1 0.0176 0.7506 348.7 67.51 480.4	1.988
13	11111111000001000111010101011111111010010001011111101011	1 0.0176 0.7506 348.7 67.51 480.4	2.363	2.00	33	50	6	0	11111101000001010111011000111111011010010001011111101011	0.9984 0.02151 0.7629 338 67.51 480.4	3.326
14	11111111000001100011010001011111100110010001011111101011	1 0.02444 0.7365 342 67.51 480.4	2.150	6.00	50	33	38	0	11111111000001000111010101011111111010010101110100111101	1 0.0176 0.7506 348.7 85.13 153.6	5.415
15	11111101000001001011010110001110100010110001011011110011	0.9984 0.01858 0.7532 319.4 192.8 363	2.231	4.00	48	9	5	1	11111100000000100011101011010110110001011101001111101001	0.9976 0.0176 0.755 298.1 228 478.5	4.516
16	11111101000000001001101010000111011100001000001111101001	0.9984 0.004906 0.7462 326.7 63.6 478.5	1.877	0.00	9	48	43	0	11110100000000001111110101001110111000010001110111101011	0.9914 0.007836 0.8622 326.7 69.47 229.9	7.516
17	11111100000000110001100011011111011101001001110111111000	0.9976 0.02444 0.6996 327.4 77.3 242.7	2.223	2.00	45	13	0	1	11111101000000001001101010000111010101001000111110000011	0.9984 0.004906 0.7462 322.1 69.47 378.7	5.812
18	11111101000001111111010101011111101000010001011111010001	0.9984 0.03127 0.7506 342.7 67.51 455	1.852	2.00	13	45	0	2	11101111000001000111010101111111111010010001011111101011	0.9875 0.0176 0.7524 348.7 67.51 480.4	4.288
19	11111100000000100011101010101111111101001000001100100011	0.9976 0.0176 0.7506 348.7 63.6 284.7	2.111	2.00	7	2	8	0	11111110000000100011101011000111010101001000001111110011	0.9992 0.0176 0.7532 322.1 63.6 488.3	2.537
20	11111111000001000111010110011110101010010001111101110011	1 0.0176 0.7541 322.1 69.47 363	2.257	0.00	2	7	7	0	11111101000001100111010001011111101110010001011111101011	0.9984 0.02541 0.7365 344.7 67.51 480.4	4.484
21	11111101000001001111010010001110100010010001111111010011	0.9984 0.01955 0.7392	2.226	0.00	24	25	48	1	11111110000001000111111110011111111010010001111111101011	0.9992 0.0176 0.8947	4.388

		319.4										348.7			
		69.47										69.47			
		456.9										480.4			
22	11110100000001000111010110001110100010110001011011100000	0.9914	2.132	2.00	25	24	11	1	11110111000001000111010100011111111010110001011110100000	0.9937	4.09				
		0.0176								0.0176					
		0.7532								0.7471					
		319.4								348.7					
		192.8								192.8					
		360.1								454					
23	11111111100000100011101011000111010101001000001111110011	1	2.325	4.00	28	14	41	0	11111111100000110001101000101111110100001000101111101011	1	3.504				
		0.0176								0.02444					
		0.7532								0.7365					
		322.1								342.7					
		63.6								67.51					
		488.3								480.4					
24	11110111000001000111111111001111111110100100011111110100 00	0.9937	1.964	2.00	14	28	27	2	11111111100000100010101010111111001100100000111101000 1	1	11.39				
		0.0176								0.0176					
		0.8947								0.3007					
		348.7								342					
		69.47								63.6					
		454								455					
25	11111111100000100011101010001111111101001000101111101011	1	2.363	2.00	45	49	0	0	11111101000000001001101010000111010101001001111110000011	0.9984	5.765				
		0.0176								0.004906					
		0.7471								0.7462					
		348.7								322.1					
		67.51								77.3					
		480.4								378.7					
26	11011111100000110001100011001111110111001000100111011011	0.9749	1.575	0.00	49	45	0	0	11111111100000110101100000001001111010000100011111100001	1	8.859				
		0.02444								0.02639					
		0.6979								0.6777					
		344.7								172.3					
		66.54								69.47					
		464.8								470.6					
27	11111101000001111111101010101111111101001000101111101011	0.9984	1.779	0.00	47	2	22	0	11111111100000100011111011000110110010001000111111111011	1	3.564				
		0.03127								0.0176					
		0.7506								0.8657					
		348.7								298.8					
		67.51								69.47					
		480.4								496.1					
28	11111111100000100011101010101111111010000100010111110100 01	1	2.357	6.00	2	47	31	1	11111100000001000111010111100110101010010000011111110011	0.9976	4.96				
		0.0176								0.0176					
		0.7506								0.7585					
		342.7								151.7					
		67.51								63.6					
		455								488.3					
29	11111101000001001111010110001110101100110001011011100 11	0.9984	2.216	0.00	28	15	53	0	11111101000001000111010101011111110100001000101111010011	0.9984	2.603				
		0.01955								0.0176					
		0.7532								0.7506					
		322.7								342.7					
		192.8								67.51					
		363								456.9					
30	11111111100000101001100011001111110001001001111110000011	1	2.299	2.00	15	28	16	2	11111111100000110101101011000111110001011000101101110001	1	3.975				
		0.02053								0.02639					
		0.6979								0.7532					
		340.7								340.7					
		77.3								192.8					
		378.7								361.1					
31	10110111000001000111111111000111010101001001111110000000	0.9435	1.940	0.00	1	28	0	0	11111111100000100011101010101111111101001000111100100011	1	3.67				
		0.0176								0.0176					
		0.8938								0.7506					
		322.1								348.7					
		77.3								69.47					
		375.7								284.7					
32	11111101000001000111010110011111111010110001111111010011	0.9984	2.190	4.00	28	1	0	0	11111111100000100011101010101111111010000100010111101000 1	1	2.033				
		0.0176								0.0176					
		0.7541								0.7506					
		348.7								342.7					
		194.7								67.51					
		456.9								455					

33	1111111100000100011101010101111111101001000101111101011	1	2.363	6.00	17	32	32	1	111111001000011000110001101111111110101100011111010011	0.9976	58.13
		0.0176								0.5244	
		0.7506								0.6996	
		348.7								348.7	
		67.51								194.7	
		480.4								456.9	
34	111111010000010011110110001111101000101100010110111001 1	0.9984	2.220	0.00	32	17	8	2	111101010000010001110101100111101110100100101101111000	0.9922	4.011
		0.01955								0.0176	
		0.7629								0.7541	
		319.4								327.4	
		192.8								73.39	
		363								242.7	
35	1111111110000110110100000011111100100001010111010000001	1	0.000	0.00	19	12	19	0	111111000000001000111010110101110101100010001111111000000	0.9976	2.774
		0.5274								0.0176	
		0.2286								0.755	
		332								322.7	
		85.13								69.47	
		126.2								438.4	
36	1111111000000100001111011010111010110001000111111000000	0.9992	2.078	2.00	12	19	9	1	111111000000010001110101010111111110110100000110010001 1	0.9976	3.272
		0.01662								0.0176	
		0.8675								0.7506	
		322.7								348.7	
		69.47								314.1	
		438.4								284.7	
37	1111111100000101011101011000111010110001000111110000011	1	2.322	2.00	22	39	37	0	111101110000010101111101010111111000000100000101000001 1	0.9937	6.743
		0.02151								0.02151	
		0.7532								0.8631	
		322.7								340	
		69.47								63.6	
		378.7								128.2	
38	111110000000000010110001110111111011000100111111101011	0.9976	1.829	0.00	39	22	36	1	111101000000001000111010110001110110110110001011011100000	0.9914	3.636
		0.002953								0.0176	
		0.7014								0.7532	
		344								326	
		77.3								192.8	
		480.4								360.1	
39	1111011100000101011111010101111110010001000001010000011	0.9937	1.697	2.00	23	44	35	1	111110010000010001110101100011101010000100010111101000 1	0.9953	3.208
		0.02151								0.0176	
		0.8631								0.7532	
		341.4								321.4	
		63.6								67.51	
		128.2								455	
40	1111111100000110111100000001001111010000101011111111000 1	1	1.264	0.00	44	23	8	1	1111111100000111110101010101111110101001000001111110011	1	14.07
		0.02737								0.03127	
		0.6777								0.3007	
		172.3								343.3	
		85.13								63.6	
		486.3								488.3	
41	1111111000000100001101010011111110110001000111011110000	0.9992	2.027	0.00	10	14	34	2	1111111100000110011101000111110110011001000101111101011	1	4.755
		0.01662								0.02541	
		0.7488								0.7383	
		344								299.4	
		69.47								67.51	
		234.8								480.4	
42	1111111100000100011101010101111111101001000101111101011	1	2.363	4.00	14	10	27	2	111110000000001000111010110011110100000111001110110100 01	0.9945	2.789
		0.0176								0.0176	
		0.7506								0.7541	
		348.7								318.7	
		67.51								226	
		480.4								204.5	
43	1111111100000000011110101100011101011000100011111101010	1	1.784	0.00	42	5	48	0	111111010000000100110101011111111101001000101111011111	0.9984	5.807
		0.00393								0.004906	
		0.7532								0.7524	
		322.7								348.7	
		69.47								67.51	
		479.5								468.7	
44	1111110100000111111101010101111110100001000101111010001	0.9984	1.852	4.00	5	42	33	2	1111111100000100011101010101111110111001000101111101010	1	1.996
		0.03127								0.0176	
		0.7506								0.7506	

		342.7								344.7	
		67.51								67.51	
		455								479.5	
45	1111110100000000100110101000011101010100100111111110000011	0.9984	1.835	6.00	32	5	51	2	111111010000010100110101010111111110101100010111101111 1	0.9984	4.086
		0.004906								0.02053	
		0.7462								0.7506	
		322.1								348.7	
		77.3								192.8	
		378.7								468.7	
46	111111010000010000110101100011101110000100000111110100 1	0.9984	2.277	0.00	5	32	27	0	111111010000010001110101101111111010100100010111101001 1	0.9984	2.628
		0.01662								0.0176	
		0.7532								0.7559	
		326.7								343.3	
		63.6								67.51	
		478.5								456.9	
47	11111111000001000111110110100101100100010001111111110 11	1	2.030	2.00	23	33	53	1	1111111100001100011101010101111111101001000001111110011	1	11.02
		0.0176								0.04885	
		0.8675								0.7506	
		128.4								348.7	
		69.47								63.6	
		496.1								488.3	
48	1111010000000000111111010100111011100001000001111101001	0.9914	1.783	2.00	33	23	35	0	1111111100000100011101011000111010101001000101111101011	1	2.288
		0.007836								0.0176	
		0.8622								0.7532	
		326.7								322.1	
		63.6								67.51	
		478.5								480.4	
49	1111111100000110101100000001001111010000100011111111000 01	1	1.403	4.00	1	18	0	1	1111111100000100011101010101111111101001000111000100011	1	7.31
		0.02639								0.0176	
		0.6777								0.7506	
		172.3								348.7	
		69.47								69.47	
		470.6								34.25	
50	1111110100000101011101100011111101101001010111010011101	0.9984	1.971	2.00	18	1	0	0	1111110100000111111101010101111110100001000101111010001	0.9984	5.643
		0.02151								0.03127	
		0.7629								0.7506	
		338								342.7	
		85.13								67.51	
		153.6								455	

Peso%= Peso dentro de la ruleta que se empleará para seleccionar los padres.
Sele%= % que indica el # de veces en que fue efectivamente seleleccionado como padre.
XP= Posición donde hubo entrecruzamiento. Si no hubo entrecruzamiento, entonces XP= 0.
NM= Número de mutaciones.

```
+----------------------------------------------------------------------------------------------------------+
|                      ESTADÍSTICAS DE LA GENERACIÓN 1000                                                   |
+----------------------------------------------------------------------------------------------------------+
 Fpromedio        :         5.57148
 Fmax(x1)         :         58.1267
 x1               : 0.9976471     0.5244252     0.6996364      348.6693      194.7162      456.9472
 Fmin(x2)         :        1.988333
 x2               : 1    0.01760117     0.7506061      348.6693      67.51468      480.4305
 Entrecruzamientos: 40
 Mutaciones       : 35
+----------------------------------------------------------------------------------------------------------+
```

```
+----------------------------------------------------------------------------------------------------------+
|                      ESTADÍSTICAS DEL AG HASTA LA GENERACIÓN 1000                                         |
+----------------------------------------------------------------------------------------------------------+
 Fmax(x1)         :        154.3094
 x1               : 0.8878431     0.9218768     0.2584848      128.4344      489.2368      323.8748
 Fmin(x2)         :        1.988333
 x2               : 1    0.01760117     0.7506061      348.6693      67.51468      480.4305
 Entrecruzamientos: 39898
 Mutaciones       : 27643
+----------------------------------------------------------------------------------------------------------+
```