UNIVERSIDAD CENTRAL "MARTA ABREU" DE LAS VILLAS
FACULTAD DE MATEMÁTICA, FÍSICA Y COMPUTACIÓN
Departamento de Matemática

Determinación de las secuencias de aminoácidos de longitud N de máxima complejidad mediante el empleo de Algoritmos Genéticos.

Tesis presentada en opción al título académico de:

Master en Matemática Aplicada

Autora: **Lic. Oristela Cuéllar Jústiz**

Tutores: **Dr. C. Vicente Molina Padrón**
Dr. C. Robersy Sánchez Rodríguez

2007

Agradecimientos

A mis tutores Vicente Molina y Robersy Sánchez por su apoyo en todo momento

A mis hermanas

A mis amigas y amigos

Al Grupo de Bioinformática

A todos los que me han ayudado en estos años

A todos los que han contribuido a la conclusión de este trabajo

Agradecimiento especial

A mi hija Ana Beatriz por su apoyo, paciencia y comprensión

Resumen

En esta tesis se exponen los resultados obtenidos en la determinación de las secuencias de aminoácidos de máxima complejidad a partir de diferentes funciones estimadoras potenciales, tomando como referencia la complejidad de Kolmogorov. Las proteínas son cadenas de aminoácidos de tamaño *n* formadas por 20 aminoácidos proteicos. Para estos aminoácidos se conoce la probabilidad que tiene cada uno de aparecer en cualquier proteína después de cada uno de los restantes; así como la probabilidad de ser el primero en cualquier cadena. Se presentan tres problemas de optimización combinatoria, sobre el conjunto de estas cadenas, que se modelan por distintas vías; por la naturaleza de estos problemas, resulta muy costoso computacionalmente la aplicación de métodos exactos para buscar su solución, por este motivo se utilizan heurísticas modernas, particularmente, Algoritmos Genéticos (AG). En el trabajo se ofrecen los aspectos más importantes de esta heurística y su implementación en el Mathematica 5.0. Se implementan tres variantes para el AG clásico (AGC) y se introduce un nuevo operador, dando lugar a un nuevo algoritmo, el cual ha sido denominado Algoritmo Genético con Transposón (AGT). La mejor variante del AG clásico se determinó mediante pruebas numéricas y luego se comparó con el AGT. Los resultados evidencian la efectividad del nuevo operador propuesto, al mejorar de manera significativa, la eficiencia del algoritmo. Como consecuencia del trabajo, se obtiene un algoritmo que permite evaluar el potencial de las funciones objetivo en la estimación de la complejidad de la secuencia de aminoácidos que conforman las proteínas.

Abstract

In this thesis the results obtained in the determination of the sequences of amino acids of Maximum Complexity are exposed. Complexity has been evaluated from different potential estimator functions, taking like reference the complexity of Kolmogorov. The proteins are chains of amino acids of so large *n* formed by 20 proteic amino acids. For these amino acids the probability to appear in any protein after each one of the rest has been determined; as well as the probability of being first in any chain. Three optimization problems of combinatory, on the set of these chains, that are modeled by different routes; by the nature of these problems, the application turns out computationally very expensive from exact methods to look for its solution. For this reason, a modern heuristic are used, particularly, Genetic Algorithms (AG). The implementation of the heuristic was done using Mathematica 5.0 software. Three variants for classic AG (AGC) are implemented and a new operator was introduced, giving rise to a new algorithm, which has been denominated Genetic Algorithm with Transposón (AGT). The best variant of the classic AG was determined by means of numerical tests and it was compared itself with the AGT. The results demonstrate the effectiveness of the new proposed operator, when improving of significant way, the efficiency of the algorithm. As a result of the work, an algorithm is obtained that allows to evaluate the potential of the functions objective in the estimation of the complexity of the sequence of amino acids that conform proteins.

Tabla de Contenidos

Introducción

La aplicación de las Matemáticas para conocer las leyes del mundo real y aprovecharlas en la práctica social es posible a través de la elaboración de los modelos matemáticos. Los modelos nos permiten llevar la investigación de un objeto real a la solución de un problema matemático en el que se pueden emplear múltiples herramientas de esta ciencia, conjuntamente con las potencialidades de los ordenadores electrónicos. No se puede pensar en la aplicación de métodos matemáticos sin la formalización del objeto de estudio, a través de la descripción de los rasgos y propiedades más importantes con ayuda de los conceptos matemáticos.

Efectivamente, con frecuencia se tropieza con problemas matemáticos cuya solución no se consigue obtener en forma de una fórmula que ligue las magnitudes buscadas con las prefijadas, al hablar de semejantes problemas se dice que no se resuelven de forma explícita. Para hallar su solución es necesario definir un proceso infinito que converja a la solución buscada y probar que, realizando un cierto número finito de pasos, se obtiene la solución aproximada del problema y que, cuanto mayor sea el número de pasos, menor ha de ser el error que se comete al sustituir la solución exacta por la aproximada. Este proceso esta ligado a la realización de cálculos de acuerdo a un riguroso sistema de reglas, que se prefijan en función del carácter del proceso y que reciben el nombre de *algoritmo*. Este enfoque para la solución de los problemas era conocido desde épocas tempranas, pero era poco utilizado debido a la excesiva laboriosidad de los cálculos.

Por otro lado, la complejidad de los procesos reales y el escaso desarrollo de algunas teorías matemáticas, conjuntamente con la complejidad de los cálculos, condicionaban la necesidad de hacer grandes simplificaciones en los modelos, los cuales perdían su validez, es decir su potencialidad de representar con precisión a los objetos reales. Esto despertaba la desconfianza y el escepticismo hacia la utilización de los modelos matemáticos.

Cada día se requiere mayor habilidad en simulación y modelación tanto en la enseñanza como en la práctica de las ciencias. La modelación se ha vuelto imprescindible para estudiar una serie de sistemas complejos, desde las ciencias sociales hasta las ciencias básicas. Como

muchas veces es imposible o resulta económicamente inviable hacer experimentos en sistemas del mundo real, se realiza la modelación con el fin de imitar el mundo real, resolver ecuaciones complejas y experimentar en el mismo. Se emplean los modelos para realizar predicción de efectos del sistema en condiciones extremas o de cambios propios del mismo, así se puede evaluar por ejemplo, el efecto de la interferencia humana en el ambiente y las políticas y medidas previas a su implementación. Además los modelos son herramientas únicas en el análisis de diferentes escenarios.

La modelación matemática es una de las herramientas que se utilizan hoy en día para el estudio de problemas en medicina y biología [CAM94],[2] fisiología, bioquímica, farmacocinética; sus objetivos primordiales son describir, explicar y predecir fenómenos y procesos en dichas áreas. El modelo matemático nos permite representar un problema médico o biológico de una manera objetiva en que se definen una serie de relaciones matemáticas entre las mediciones cuantitativas (del problema) y sus propiedades [9], [20]. El uso integrado de diferentes áreas de la ciencia está entregando cada vez más y mejores técnicas a la investigación biotecnológica [20]. Así las matemáticas modernas están arremetiendo violentamente en un mundo que antes era privativo de la experimentación, a través de la modelación de los sistemas biológicos [15]. La evolución de las ciencias biomédicas [20] ha sido vertiginosa en los últimos veinte años, y los cambios no se detienen. Hoy, gracias al conocimiento obtenido a través de los diferentes proyectos de secuenciación, se están realizando diferentes aproximaciones para comprender los sistemas biológicos, en los que cada vez las matemáticas juegan un papel más importante, permitiendo a los investigadores avanzar mucho más deprisa en sus campos de estudio [MUR01].

Nuestro trabajo consiste en la modelación y búsqueda de la solución a un grupo de problemas planteados por el Grupo de Investigaciones en Bioinformática de la Universidad Central “Marta Abreu” de Las Villas. Estos problemas están vinculados a los excelentes resultados en la aplicación del Álgebra al Genoma Humano [SAN03], [SANGRA04], [SANMORGRA05] y esta investigación da continuidad a los trabajos del grupo.

El uso principal de cualquier teoría está en sus predicciones. Los modelos son distinguidos principalmente por los diversos experimentos que sugieren y, por supuesto, cómo se

relacionan de cerca con el fenómeno biológico real. No se reportan en la literatura antecedentes de modelación y solución de problemas de optimización en cadenas de aminoácidos, por tanto la temática es novedosa.

Las proteínas están conformadas por cadenas de aminoácidos de longitud *N* a las cuales se les llama cadenas polipeptídicas. En cada posición de la cadena puede encontrarse uno cualquiera de los veinte aminoácidos que conforman las proteínas naturales.

Problema de investigación

Es preciso hallar la(s) secuencia (s) de aminoácidos que maximiza(n) las funciones estimadoras potenciales de la complejidad de las proteínas (EPCP).

En el problema que proponemos resolver las cadenas polipeptídicas se forman estocásticamente a partir de una matriz de transición de probabilidades P, un vector de probabilidad del estado inicial p_0 y una probabilidad estacionaria dada por un vector p. Las funciones EPCP dependen de las probabilidades mencionadas, lo cual da lugar a varios Problemas de Optimización Combinatoria sobre el conjunto de todas las cadenas posibles. Para la solución de estos problemas se trabajó en una primera etapa en su modelación y después en su solución.

Hasta el presente, en la literatura no se conoce un estimador de la complejidad de las proteínas. El paradigma que se aplica con frecuencia es la complejidad definida por Kolmogorov, la cual nos brinda una medida de la aleatoriedad de las secuencias de proteínas y de su no redundancia. Sin embargo, esta definición no es fácilmente tratable en la práctica con las proteínas. Es por ello que se acude a las funciones EPCP y se toma como referencia la definición de Kolmogorov. Una secuencia de aminoácidos que maximaza una función EPCP estará más "cercana" al paradigma de Kolmogorov en la medida que sea menos redundante.

Hipótesis de investigación

Es posible establecer algoritmos exactos o heurísticas para determinar la(s) secuencia(s) de aminoácidos de longitud N de máxima complejidad, utilizando para este fin diferentes funciones estimadoras potenciales de la complejidad.

Objetivo

Modelar y resolver problemas de optimización con funciones estimadoras potenciales de la complejidad de cadenas de aminoácidos.

Objetivos específicos:

1. Elaborar diferentes modelos para los problemas de optimización.
2. Valorar la complejidad de los algoritmos de solución de los problemas.
3. Elaborar un algoritmo que permita determinar la cadena de aminoácidos de máxima complejidad a partir de diferentes funciones que potencialmente puedan medirla.

El tercer objetivo específico es un equivalente práctico a:

3´. Elaborar un algoritmo que permita evaluar diferentes funciones EPCP tomando como referencia las consecuencias de la definición de complejidad de Kolmogorov.

Tareas de investigación

1. Formulación del problema, recolección de los datos necesarios.
2. Elaboración de los modelos de optimización posibles.
3. Análisis de distintos algoritmos de solución de los problemas.
4. Obtención de la solución del problema a partir de uno de los algoritmos analizados.
5. Análisis de la complejidad de los algoritmos de solución de los problemas.
6. Interpretación de los resultados junto con los especialistas que propusieron el problema y análisis de posibles modificaciones de los modelos.
7. Resolución del problema utilizando los modelos modificados.
8. Validación de los resultados
9. Análisis sobre si las funciones propuestas son estimadoras potenciales de la complejidad o no en el sentido de Kolmogorov.

Novedad científica:

a. La modelación del problema de investigación por distintas vías, lo que abre nuevas posibilidades de aplicación de las matemáticas a las investigaciones que realiza el grupo sobre el Genoma Humano e, incluso, proporciona el mecanismo para generalizar estos resultados a otras investigaciones.
b. La elaboración de un algoritmo que permite determinar la cadena de aminoácidos de máxima complejidad utilizando diferentes funciones, estimadoras potenciales de la misma.
c. La introducción de un nuevo operador para los AG.

La tesis está estructurada en: Introducción, tres Capítulos, Conclusiones y Recomendaciones, Referencias Bibliográficas y Anexos.

En el primer capítulo se presenta el marco teórico de la investigación, los problemas biológicos a resolver, los modelos de optimización, la complejidad de los algoritmos y problemas de optimización, así como una explicación detallada de las heurísticas, metaheurísticas en optimización y, en específico, de los AG que es la que se implementa en la tesis.

En el segundo capítulo se destaca la importancia de la modelación matemática y se exponen los modelos de optimización elaborados para los problemas propuestos.

El tercer capítulo se dedica a la implementación de los algoritmos AG en el Matemática 5.0, la presentación de un nuevo operador, un AG modificado, al que le hemos llamado Algoritmo Genético con Transposón (AGT), el análisis de la complejidad temporal de este algoritmo, las pruebas numéricas y los análisis estadísticos. Por último, se concluye el capítulo con el análisis de las funciones utilizadas para estimar la complejidad de las proteínas (cadenas de aminoácidos (aa)) en el sentido de Kolmogorov.

1 Marco teórico

Desde la antigüedad se ha utilizado la modelación en las investigaciones abarcando paulatinamente nuevas esferas de los conocimientos científicos. En el siglo XX, el método de modelación obtuvo grandes éxitos y reconocimientos en todas las ramas de la ciencia contemporánea como son el diseño técnico, la construcción, la arquitectura, la astronomía, la física clásica, la química, la biología y las ciencias económicas. Sin embargo, la metodología de la modelación durante mucho tiempo se ha desarrollado en forma independiente dentro de algunas ciencias y no existe un sistema único de definiciones y terminología.

El *modelo* es un objeto material o mentalmente representado, que en el proceso de investigación, sustituye al objeto original, de manera que el estudio de dicho modelo permite obtener nuevos conocimientos sobre el objeto.

Modelación es el proceso de construcción, estudio y aplicación de los modelos, la peculiaridad fundamental de la modelación es el método de estudio condicionado de los objetos sustitutos. El modelo es el instrumento para el estudio del objeto y se ubica entre el investigador y el objeto original.

La necesidad de utilizar el método de modelación surge cuando es imposible investigar muchos objetos o problemas relacionado con ellos o cuando la investigación requiere grandes gastos de tiempo y recursos.

Es evidente que el modelo pierde su sentido en el caso de ser igual al objeto original (puesto que no es un modelo) y en el caso en que todas las características esenciales difieren considerablemente del objeto original. Por esta razón, cualquier modelo sustituye al objeto original, solo dentro de marcos muy limitados. Se deduce que para un objeto de investigación pueden elaborarse varios modelos especializados, que concentren la atención sobre algunas

propiedades determinadas de dicho objeto o propiedades que lo caractericen con diferente grado de detalle.

La modelación es un proceso cíclico, lo cual significa que después del primer ciclo puede realizarse un segundo, tercero y otros. En este proceso los conocimientos sobre el objeto de investigación se amplían y precisan y el modelo inicial se perfecciona, las deficiencias detectadas después del primer ciclo de modelación, y los errores en la construcción del modelo se pueden corregir en el desarrollo de los ciclos siguientes; de este modo, en la metodología de la modelación existen grandes posibilidades de auto perfeccionamiento.

Al desarrollar el modelo, se recomienda empezar con una versión muy sencilla y moverse en una forma evolutiva, hacia modelos más elaborados que reflejen, lo mejor posible, la complejidad del problema real. Este proceso de enriquecimiento del modelo continúa solo mientras permanezca manejable.

1.1 Planteamiento del problema

La aparición de la computación contribuyó a acelerar el desarrollo de las Matemáticas Aplicadas Modernas. En la solución de los problemas prácticos utilizando la modelación matemática y el experimento de cálculo surgen nuevos y difíciles problemas, relacionados con el estudio de los modelos y con la elaboración de algoritmos de cálculo eficaces que estimulan, a su vez, el desarrollo de las Matemáticas Aplicadas [TIJKOS87].

También el reclamo por la utilización de esta metodología planteado por ciencias como la Biología, exige la aplicación de nuevos modelos y, durante la elaboración de los mismos, surge un gran número de problemas matemáticos, de gran complejidad y dificultad, ya que muchos de esos modelos son no lineales, no estacionarios o están afectados por elementos aleatorios y no estacionarios, lo cual incide también en la aparición y desarrollo de nuevas ramas de las Matemáticas Aplicadas Modernas.

Sin ninguna duda, el siglo pasado perteneció a las Ciencias Exactas y en especial a la Física, tal y como pone de manifiesto el descubrimiento de las dos principales teorías de la Física Moderna: la Teoría General de la Relatividad y la Mecánica Cuántica. Ambas culminaron un largo camino iniciado en el siglo XVII con la aparición en 1686 de una de las obras cumbres del pensamiento Universal: los *Principia* de Newton, que eliminó lo sobrenatural de la naturaleza. De hecho, actualmente la Física permite describir con gran precisión un sinnúmero de fenómenos naturales, desde el comportamiento de un átomo hasta el movimiento de toda una galaxia. Sin embargo, este conocimiento no basta ni para explicar el mecanismo de funcionamiento de otra de las grandes revoluciones científicas: *el origen y evolución de las especies*; ni tampoco los nuevos problemas derivados de la secuenciación del ADN, el famoso proyecto Genoma -quizá el descubrimiento más importante de los últimos años-. Es por ello que muchos científicos afirman que el siglo XXI es el siglo de la Biología.

La Biología intenta dar respuestas a los fenómenos vitales. Al constituir el ADN la base genética de la vida [CRA68], es natural buscar explicaciones a nivel del mismo, ya que este conocimiento molecular puede dar la clave de muchos fenómenos que hoy entendemos a niveles menos profundos. Conocida la secuencia del ADN, una de las principales tareas de la Biología consiste en determinar su funcionamiento. Por otro lado, la resolución de estos nuevos problemas requiere no sólo del conocimiento previo alcanzado en las Ciencias Básicas -la Física, la Química, la Matemática, la Biología o la Medicina-, sino también de la interacción entre ellas -que ha dado lugar a la Biofísica, la Bioquímica o la Bioinformática, por citar algunas-. En todo este proceso las Matemáticas juegan un papel fundamental pues, al ser éstas el lenguaje de la Ciencia, cabe esperar que su interacción con la Biología, la Medicina o la Genética, por un lado, ayude a entender el funcionamiento del ADN y por otro, que aquellas aporten una nueva serie de retos matemáticos.

1.1.1 Los aminoácidos y las proteínas.

Las cadenas de aminoácidos conforman las proteínas. La unión de aminoácidos da lugar a la formación de péptidos que se denominan dipéptidos, tripéptidos, tetrapéptidos, pentapéptidos, octapéptidos o polipéptidos, si en su formación intervienen, 2, 3, 4, 5, 8 o un número superior. La unión de polipéptidos entre sí da lugar a la formación de proteínas.

Los aminoácidos [EPS66],[GRA74] son los componentes básicos de las proteínas y para cada una la secuencia de los aminoácidos es diferente. El número de secuencias posibles es tan grande que eso explica la gran cantidad de proteínas existentes.

Cada organismo vivo sintetiza sus propias proteínas a partir de los aminoácidos. Las plantas superiores sintetizan todos los aminoácidos necesarios. Hay que destacar que los animales carecen de esa capacidad. Cada especie animal puede sintetizar sólo algunos aminoácidos que necesita y, por lo tanto, depende de la dieta para incorporar aquellos aminoácidos que debe sintetizar para formar proteínas. Esos aminoácidos se les considera esenciales, no porque sean los únicos necesarios para la vida de la especie, sino porque deben estar incluidos en la dieta. Cada especie, tiene su grupo de aminoácidos esenciales propios.

En los animales superiores, las proteínas son los compuestos orgánicos más abundantes, pues representan alrededor del 50% del peso seco de los tejidos. Desde el punto de vista funcional, su papel es fundamental. No existe proceso biológico alguno que no dependa de la presencia y/o actividad de este tipo de sustancias. Las proteínas cumplen diferentes funciones: enzimas, hormonas, trasportadores, los anticuerpos, receptores de muchas células, etc.

Las proteínas son moléculas poliméricas de enorme tamaño; pertenecen a la categoría de macromoléculas, constituidas por gran número de unidades estructurales que forman largas cadenas. Las proteínas resultan de la unión de moléculas de aminoácidos por medio de enlaces peptídicos.

La estructura de las proteínas es muy compleja. La estructura primaria se refiere al número e identidad de los aminoácidos que componen la molécula y al ordenamiento o secuencia de esas unidades en la cadena polipeptídica. La unión peptídica sólo permite formar estructuras lineales; por ello, las cadenas no presentan ramificaciones. A medida que la longitud de las cadenas va aumentando, y en función de las condiciones fisicoquímicas del medio, se conforma la estructura secundaria, que es la disposición espacial regular, repetitiva, que puede adoptar la cadena polipeptídica, generalmente mantenida por enlaces de hidrógeno.

1.1.2 Problemas a resolver

La cuestión de cuándo la "complejidad" de una secuencia es mayor o menor al ser comparada con otra, puede ser tratada adoptando la definición de complejidad dada por Kolmogorov. Formalmente la complejidad de Kolmogorov de una cadena *c* es definida (dentro de los límites de cadenas "largas") como la longitud $|p|$ (en bits) del programa más corto *p* que genera *c*, cuando se ejecuta en una máquina universal de Turín *T*:

$$K(s) = Min\{|p| \mid s = C_T(p)\}$$

donde $C_T(p)$ es el resultado de ejecutar el programa *p* en la máquina *T* [Vol85] [ADA00] [YOK02].

Esta definición provee una medida de la regularidad o redundancia de una cadena de símbolos. De manera "cruda" se dice que una cadena es "regular" si el algoritmo necesario para reproducirla en una máquina de Turín es más corto (con la longitud medida en bits) que la cadena misma. Un ejemplo simple es la cadena 010101010101010101…

El programa mínimo que permite escribir esta cadena en una máquina de Turín solo requiere del patrón 01 y las instrucciones "repite y escribe". La complejidad determinada de tal modo es equivalente a la sucesión aleatoria de elementos [VOL85] [ADA00]. Una secuencia de símbolos aleatoria posee el mayor grado de complejidad según la definición de Kolmogorov.
Desde el punto de vista biológico esta definición no nos satisface y tiene ciertas limitaciones. Las secuencias biológicas son "altamente organizadas", lo cual significa que, aunque no son aleatorias, requieren de un algoritmo muy largo para ser generadas.
Gregory Chaitin [CHA98] ha probado que no se puede determinar cuando una sucesión de símbolos dada ha sido generada por un proceso estocástico o por uno altamente organizado [YOK05]. Como consecuencia, es imposible determinar cuando una secuencia es aleatoria o altamente organizada.

Luego, solo podemos referirnos a funciones estimadoras potenciales de la complejidad y la definición de Kolmogorov es tomada en la tesis como un punto de referencia que nos indica en que medida nos acercamos o nos alejamos de este paradigma de complejidad.

Como se señala en el epígrafe anterior, todas las proteínas utilizadas por los seres vivos están formadas por los mismos 20 aminoácidos (aa), por lo que podemos considerar las proteínas como cadenas de aa de tamaño n. Para cada uno de estos aa, se ha determinado la probabilidad que tiene de aparecer en cualquier proteína después de cada uno de los 20 existentes; así como la probabilidad de ser el primero en cualquier cadena.

El grupo de Bioinformática de la UCLV ha obtenido resultados relevantes en la aplicación de diversas ramas de la Matemática a la solución de diferentes problemas relacionados con el código genético [SAN03], [SANGRA04], [SAN05], [SANMORGRA05]. De estos trabajos se derivan algunas situaciones que requieren el desarrollo de modelos de optimización en las proteínas, los que pueden ser formulados de la manera siguiente:

1.1.2.1 Problema 1

Para expresar el contenido semántico de la información de la proteína el punto de partida es la estructura del Álgebra de Boole del código genético recientemente revelada. Se considera la secuencia de aminoácidos como *un mensaje*, en el cual cada símbolo (aminoácido) es generado según su probabilidad de deducción por una fuente discreta de información, representada por una cadena de Markov de primer orden.

Sean, la función $i:\{1,2,\cdots,n\}\to\{1,2,\cdots,20\}$ y la sucesión $X = aa_{i(1)}\, aa_{i(2)} \ldots aa_{i(n)}$ que representa una cadena de aminoácidos de tamaño n. Esto significa que $aa_{i(k)}$ representa uno cualquiera de los 20 *aa* ocupando la posición k dentro de la cadena.

Se define el ***valor de la información*** de la proteína como:

$$V(x) = -\log_4 p(x) \qquad \text{donde} \quad p(x) = \prod_{k=1}^{n} p\left(aa_{i(k)} / aa_{i(k-1)}\right)$$

- $p(aa_h / aa_j)$ representa la probabilidad de que el aminoácido ***h*** aparezca inmediatamente después del aminoácido ***j*** en una cadena,
- $p(aa_{i(1)} / aa_{i(0)})$ representa la probabilidad de que el aminoácido ***i(1)*** ocupe la primera posición en una cadena.

La selección de una base para el logaritmo corresponde a la selección de una unidad para medir el valor de la información. Cuando se utiliza la base 4, las unidades que resultan se pueden llamar tetra-dígitos, o más brevemente los *teds*.

Sea $m_{i(k)}$ la masa molecular del aminoácido que ocupa la posición ***k*** dentro de la cadena, entonces la masa molecular de una proteína de tamaño ***n***, podemos calcularla [SAN03] en base a:

$$M(x) = \sum_{k=1}^{n} m_{aa_{i(k)}} - (n-1) * 0.01802 \qquad \textbf{(1.1)}$$

Se desea optimizar la función que representa el ***valor de la información de la proteína por unidad de masa*****:**

$$r(x) = \frac{V(x)}{M(x)}$$

Sustituyendo estas expresiones en la función ***r (x)*** se obtiene que

$$r(x) = \frac{-\log_4 \left[\prod_{k=1}^{n} p\left(aa_{i(k)} / aa_{i(k-1)}\right) \right]}{\sum_{k=1}^{n} m_{aa_{i(k)}} - (n-1) * 0.01802} \qquad \textbf{(1.2)}$$

Si se aplican las propiedades de los logaritmos se obtiene otra expresión para la fórmula anterior:

$$r(x) = \frac{-\sum_{k=1}^{n} \log_4 p\left(aa_{i(k)} / aa_{i(k-1)}\right)}{\sum_{k=1}^{n} m_{aa_{i(k)}} - (n-1) * 0.01802} \qquad \textbf{(1.3)}$$

Como se desea maximizar el valor de la información de la proteína por unidad de masa, es necesario hallar la cadena ***x*** que maximice ***r* (*x*)**.

1.1.2.2 Problema 2

Para los 20 *aa* que forman las proteínas, se ha obtenido un vector de probabilidades absolutas [SAN03, SAN04] $\boldsymbol{P = [p_1, p_2, \ldots, p_{20}]}$, donde p_j ; $j = \overline{1,20}$ es la probabilidad de que el aminoácido ***j*** aparezca en cualquier posición dentro de la cadena.

Sean, ***i*** y ***x*,** respectivamente, la función y la sucesión definidas en el epígrafe anterior. La función $\boldsymbol{C_2(x)}$ representa la ***complejidad de la cadena*** y está definida por la expresión:

$\boldsymbol{C_2(x) = \frac{\sigma(x)}{\overline{V}_{aa}}}$, donde

- σ representa la desviación Standard, y se obtiene mediante la siguiente expresión:

$$\boldsymbol{\sigma^2(x) = \frac{1}{n}\left[\sum_{k=1}^{n}\left(-\log_4 p_{i(k)}\right)^2 - \frac{\left(\sum_{k=1}^{n} -\log_4 p_{i(k)}\right)^2}{n}\right]} \qquad \textbf{(1.4)}$$

- $\overline{V}_{aa}$ representa la media, y se calcula a través de:

$$\overline{V}_{aa} = -\frac{1}{n}\sum_{k=1}^{n} \log_4 p_{i(k)} \qquad \textbf{(1.5)}$$

Realizando transformaciones obtenemos:

$$\left[C_2(x)\right]^2 = n\left[\frac{\sum_{k=1}^{n}\left(-\log_4 p_{i(k)}\right)^2}{\left(\sum_{k=1}^{n} -\log_4 p_{i(k)}\right)^2} - \frac{1}{n}\right] \qquad \textbf{(1.6)}$$

Se desea encontrar ***x***, o sea, la secuencia de *aa* de tamaño ***n***, que maximice la complejidad de la cadena. Es evidente que, la cadena ***x*** que maximiza $\left[C_2(x)\right]^2$ también maximiza $C_2(x)$.

1.1.2.3 Problema 3

Sean ***u,*** el vector correspondiente a cada una de las cadenas *x*, P*,la matriz de probabilidades de orden 20 x 21 del problema 1

$$u(x) = \left(-\log_4 p\left(aa_{i(1)}\right), \ldots, -\log_4 p\left(aa_{i(k)}/aa_{i(k-1)}\right), \ldots, -\log_4 p\left(aa_{i(n)}/aa_{i(n-1)}\right)\right)$$

El valor de la información de la proteína, definido por las fórmulas **(1.2)** y **(1.3)**, también puede expresarse como:

$V(u(x)) = \sum_{k=1}^{n} u_k$ donde u_k representa la ***k***- ésima componente del vector ***u***

La masa molecular $M(x)$ de una proteína de tamaño ***n***, se calcula mediante la fórmula (1.1). Por tanto, se desea optimizar una función que represente el ***valor de la información de la proteína por unidad de masa,*** dada por:

$$r\left(u(x)\right)=\frac{V\left(u\left(x\right)\right)}{M\left(x\right)}$$

Sustituyendo estas expresiones en la función ***r* (*u*(*x*))** se obtiene que

$$r\left(u\left(x\right)\right)=\frac{\sum_{k=1}^{n}u_{k}}{\sum_{k=1}^{n}m_{aa_{i(k)}}-(n-1)*0.01802} \qquad \textbf{(1.7)}$$

De la misma manera, se define la función $\boldsymbol{C_2}$ **(*u*(*x*))** [1]que representa la ***complejidad de la cadena***:

$$C_2\left(u(x)\right)=\frac{\sigma(u)}{\overline{V}}$$ donde

- la desviación Standard **σ** se obtiene de:

$$\sigma^2\left(u(x)\right)=\frac{1}{n}\left[\sum_{k=1}^{n}(u_k)^2-\frac{\left(\sum_{k=1}^{n}u_k\right)^2}{n}\right] \qquad \textbf{(1.8)}$$

- la media de los $\boldsymbol{u_k}$, se calcula a través de:

$$\overline{V}=\frac{1}{n}\sum_{k=1}^{n}u_k \qquad \textbf{(1.9)}$$

Por analogía con la expresión **(1.6)**, obtenemos

$$\left[C_2\left(u(x)\right)\right]^2=n\left[\frac{\sum_{k=1}^{n}(u_k)^2}{\left(\sum_{k=1}^{n}u_k\right)^2}-\frac{1}{n}\right]$$

[1] Ver Problema 2

Si multiplicamos las razones *valor de la información de la proteína por unidad de masa* y *la desviación Standard entre la media de los* u_k, y teniendo en cuenta que $V\left(u(x)\right)=n\overline{V}$, se tiene que

$$\frac{n\overline{V}}{M(x)}\cdot\frac{\sigma(u(x))}{\overline{V}}=\frac{n\,\sigma(u(x))}{M(x)}$$

Se puede entonces, plantear el problema de hallar el vector $u(x)$ para el cual el valor de la razón es máximo. Expresemos esta razón mediante una función:

$$f\left(u(x)\right)=\frac{n\,\sigma(u(x))}{M(x)}$$

Sustituyendo los valores de $M(x)$ y de la desviación Standard:

$$\sigma(u(x))=\frac{1}{n}\sqrt{\left[\,n\sum_{k=1}^{n}(u_k)^2-\left(\sum_{k=1}^{n}u_k\right)^2\right]}$$

en la expresión anterior, se obtiene que

$$f\left(u(x)\right)=\frac{\sqrt{n\sum_{k=1}^{n}(u_k)^2-\left(\sum_{k=1}^{n}u_k\right)^2}}{\sum_{k=1}^{n}m_{aa_{i(k)}}-(n-1)*0,01802} \qquad \textbf{(1.10)}$$

Se desea encontrar el vector $u(x)$ que maximice la función f. Esta función representa la energía media de la proteína por unidad de masa.

Los problemas planteados son problemas de optimización combinatoria, los cardinales de los espacios de búsqueda para los problemas 1 y 3 son $|S|=20^n$ y para el problema 2 $|S|=20^n / n!$ Los espacios de búsqueda crecen rápidamente con el tamaño del problema, las funciones de evaluación son complejas, por estas razones aplicar métodos exactos para su solución es muy costoso desde el punto de vista computacional.

1.2 Problemas de optimización combinatoria.

En décadas pasadas, la comunidad científica asistió al nacimiento de la disciplina conocida como *Ciencias de la Computación* que, siendo inicialmente una rama de la Matemática Aplicada, encontró su propio espacio de investigación y se definió posteriormente como una nueva área de la ciencia. Esta disciplina experimentó un vertiginoso ascenso desde su nacimiento, convirtiéndose en la actualidad en una de las áreas con mayor actividad y desarrollo. Una de las ramas de mayor importancia y crecimiento en la actualidad es también el conjunto de actividades conocidas como *Investigación de Operaciones* que, por su impacto y resultados concretos en la industria y en otros ámbitos, se ha transformado en uno de los pilares de la Matemática Aplicada y la Ciencia de la Computación. Dentro de la Investigación de Operaciones, la *Optimización Combinatoria* es una de las actividades más importantes.

La Investigación de Operaciones es la aplicación, por grupos interdisciplinarios, del método científico a problemas relacionados con el control de las organizaciones o sistemas, a fin de que se produzcan soluciones que mejor sirvan a los objetivos de toda la organización. Los recursos y actividades dentro de una organización pueden conducirse de diversas maneras, y es tarea de la Investigación de Operaciones determinar la mejor forma de hacerlo.

Las “organizaciones” a las que hace referencia la definición anterior pueden interpretarse como un sistema con componentes e interacciones (controlables o no) entre las mismas. Puede tratarse de empresas comerciales, organizaciones militares, gubernamentales o sanitarias, con objetivos distintos en cada caso. Los objetivos de la organización se refieren a la eficiencia y efectividad con que las diferentes componentes del mismo pueden controlarse y/o modificarse. Por ejemplo, en una empresa comercial el enfoque estará puesto en la reducción de los costos, el mejor aprovechamiento de los recursos existentes o la mejora de los beneficios, mientras

que las organizaciones militares o sanitarias tienen habitualmente otros objetivos. Sin embargo, es importante notar que en todos los casos las técnicas de la Investigación de Operaciones son aplicables y, de hecho, se han logrado importantes mejoras en todas estas actividades gracias a la participación de esta disciplina.

La *Optimización Combinatoria* [18] es un área dentro de la Investigación de Operaciones, que se encarga de buscar la mejor solución en problemas *discretos* (es decir, en los que participa una cantidad finita de elementos). La planificación de actividades industriales, la organización del recorrido de vehículos, la organización de actividades y la búsqueda de esquemas de producción, entre otras, son posibles gracias a la Optimización Combinatoria.

Resolver un Problema de Optimización Combinatoria (POC) es encontrar la mejor (optimal) solución entre un conjunto finito o infinito numerable de soluciones alternativas. Por el hecho de trabajar con estos conjuntos, que son los mismos que se estudian en el Análisis Combinatorio, esta rama de la Optimización (Programación Matemática) se denomina Optimización Combinatoria (OC). Esta es la denominación que aparece con más frecuencia en la literatura actual, anteriormente se han utilizado otras denominaciones, como son: Programación en Enteros (PE), que se refiere al hecho de que el conjunto de soluciones es un subconjunto de Z^n; Programación Discreta (PD)**,** que se refiere a la estructura discreta de estos conjuntos, es decir que son conjuntos de puntos aislados, según la topología usual en R^n.

Formalicemos la definición de POC [PAPSTE98], [MAI95]

> Un ***POC*** es o un problema de minimización -o un problema de maximización- sobre un conjunto finito, o infinito numerable, de soluciones alternativas, y está especificado por un conjunto de instancias del problema.
>
> Una ***instancia de un POC*** puede ser formalizada como un par **(*S, f*)** donde el *espacio de soluciones* ***S*** denota el conjunto finito, o infinito numerable, de todas las soluciones posibles y la *función de costo* ***f*** es una aplicación definida como $\boldsymbol{f : S \rightarrow R}$

En el caso de la minimización, el problema consiste en encontrar una solución $i_{opt} \in S$ que satisface $f(i_{opt}) \leq f(i)$, para toda $i \in S$

En el caso de la maximización, i_{opt} satisface $f(i_{opt}) \geq f(i)$, para toda $i \in S$

Esta solución i_{opt} es llamada *solución globalmente óptima, - minimal* o *maximal-*, o simplemente *óptimo, - mínimo* o *máximo*.

$f_{opt} = f(i_{opt})$ denota el costo óptimo, y S_{opt} es el conjunto de soluciones optímales.

Una cuidadosa distinción entre un problema y una instancia de un problema, se hace para diferenciar entre una simple "entrada de datos" -para la cual una solución puede ser obtenida sin ambigüedades- de un conjunto de datos de entrada -los cuales son generalmente generados del mismo modo.

Un ejemplo clásico de un problema de optimización combinatoria es el *Problema del viajero vendedor*, conocido por sus siglas en inglés (TSP).

La optimización combinatoria esta relacionada con la investigación de operaciones, la teoría de algoritmos, la teoría de la complejidad computacional, la inteligencia artificial e ingeniería de software, etc. Los algoritmos de optimización combinatoria resuelven instancias de problemas que se creen ser difíciles en general, explorando el espacio de soluciones (usualmente grande) para estas instancias. Los algoritmos de optimización combinatoria logran esto reduciendo el tamaño efectivo del espacio y explorando el espacio de búsqueda eficientemente.

Una característica recurrente en los POC es el hecho de que son muy "fáciles" de entender y de enunciar, pero generalmente son "difíciles" de resolver. Podría pensarse que la solución de un POC se restringe únicamente a buscar de manera exhaustiva el valor máximo o mínimo en un conjunto finito de posibilidades y que usando una computadora veloz, el problema carecería de interés matemático, sin pensar por un momento, en el tamaño de este conjunto.

Mediante el estudio de la teoría de la complejidad computacional es posible comprender la importancia de la optimización combinatoria. Los algoritmos de optimización combinatoria se relacionan comúnmente con problemas NP-hard (difíciles). Dichos problemas, en general, no son resueltos eficientemente; sin embargo, varias aproximaciones de la teoría de la complejidad sugieren que ciertas instancias ("pequeñas" instancias) de estos problemas pueden ser resueltas.

1.2.1 Métodos exactos y aproximados

En la solución de problemas de optimización existen dos opciones de métodos de solución apropiados: métodos exactos y métodos aproximados. Los primeros buscan optimalidad a riesgo de mucho tiempo de cómputo (posiblemente impracticable). Los segundos buscan rápida obtención de la solución a riesgo de suboptimalidad, es decir, de alcanzar una *buena solución* aunque no sea la óptima.

Dentro de los métodos exactos están todos los métodos de solución de problemas de programación lineal, los métodos de planos cortantes, los de enumeración o las técnicas de programación dinámica.

- *Métodos de planos cortantes*: Se transforma el problema de manera que sean utilizables los métodos de solución de los Problemas de Optimización Continuos, su nombre viene de la idea geométrica que lo sustenta, y que consiste en ir añadiendo restricciones al problema de modo que el conjunto de soluciones óptimas permanezca invariable.
- *Métodos de enumeración*: Utilizan las características discretas del conjunto de soluciones posibles para lograr, mediante la enumeración de algunas soluciones, recorrer todo el conjunto de soluciones factibles del problema. La idea central de este método consiste en la sustitución del conjunto de todas las soluciones por un subconjunto de ellas sobre la base de eliminar aquellas soluciones que con toda seguridad no ofrecen óptimo, estrechando el dominio de las variantes posibles. Se pueden mencionar métodos como: Ramas y Cotas (algoritmo de Lang & Doig, 1960), algoritmos de Balas, etc.

Estos métodos exactos de optimización requieren, por regla general, usar cálculos muy

voluminosos, y necesitan mucho espacio de memoria en su implementación computacional; además suelen ser excesivamente lentos para grandes problemas [RIB 85]. Por esto tiene gran importancia práctica la creación de métodos sencillos que aseguran la obtención de soluciones bastante próximas al óptimo. Es aquí donde entran los métodos de aproximación, combinados con ciertos elementos estocásticos y enfoques dinámicos. Entre estos están los métodos de búsqueda local y los algoritmos aleatorios.

Los algoritmos de búsqueda local presuponen la definición de vecindades y se basan en una mejora, paso a paso, mediante el análisis de las vecindades. Estos algoritmos terminan en un óptimo local y, a menos que la vecindad sea exacta, generalmente no se conoce cuán cercana está esta solución local de la óptima global. La calidad de la solución óptima local obtenida, generalmente depende de la solución inicial y en muchos problemas no hay un método disponible para seleccionar una solución inicial apropiada.

Los problemas de Optimización Combinatoria aparecen de forma natural en procesos de fabricación, logística, localización de servicios, planificación de actividades, etc. Se caracterizan por querer optimizar uno o varios objetivos sobre un gran número de soluciones que crece de forma exponencial en función del tamaño del problema. Es necesario para el decisor disponer de algoritmos eficientes que le ayuden en sus decisiones. Los métodos exactos únicamente proporcionan la solución óptima en tiempo razonable para unos pocos problemas combinatorios, mientras que para la mayoría de ellos, debemos recurrir a métodos aproximados que proporcionan un abanico de soluciones cercanas a la óptima. Cualquier problema combinatorio exigirá un análisis previo y, en función de su tamaño y complejidad, la selección y codificación de la técnica más adecuada.

1.2.2 Análisis de la complejidad de algoritmos

Primeramente, tomemos algunas definiciones relacionadas con los objetivos de nuestro trabajo. [PAPSTE98], [19], [21], [22], [SEGVER04].

Algoritmo es un secuencia de instrucciones precisas que resuelven cualquier instancia de un problema rigurosamente definido. Por su parte, *problema no decidible* es un problema bien

definido para el cual no existe un algoritmo. Por ejemplo, el conocido *problema del Halting* (abortar): Dado un programa de computación con su entrada, ¿terminará este alguna vez? Turing probó que no existe algoritmo que resuelva correctamente todas las instancias de este problema. Nos interesan los *problemas decidibles* o sea problemas para los cuales sí existen algoritmos que lo resuelvan.

Se le llama m*edida de realización de un algoritmo*, al tiempo que demora en entregar una solución final. Este tiempo depende del número de pasos elementales, de las operaciones aritméticas, comparaciones, instrucciones ramificadas, etc. Se asume que cada uno requiere una unidad de tiempo. [PAPSTE98]. La medida de realización de un algoritmo puede diferir de una instancia a otra de un mismo problema.

La entrada de un algoritmo [PAPSTE98] se codifica como una secuencia de símbolos de un alfabeto. Se entiende por *tamaño de la entrada de un algoritmo* la longitud de la secuencia que codifica la entrada del mismo.

La *Complejidad* de un algoritmo nos da una medida de los recursos (tiempo, memoria) que requiere su ejecución en función del tamaño de los datos de entrada. Para su estudio, las situaciones que se consideran son el análisis del desempeño del algoritmo en el caso peor, caso mejor y el caso promedio. La complejidad es una medida independiente de la arquitectura del computador [4], [8].

Para la determinación de la complejidad temporal de un algoritmo es necesario:

- Determinar las operaciones e instrucciones que requieren un gasto significativo de tiempo.
- Establecer un estimado del gasto de tiempo para cada una de las operaciones e instrucciones significativas.
- Determinar la frecuencia con que se ejecutan estas operaciones e instrucciones para las instancias.

Notación asintótica. Es una notación matemática que permite analizar el comportamiento de las funciones en el *límite*, o sea, su *tasa de crecimiento*. Se aplica a funciones de tiempo de ejecución, o de espacio de memoria, de los algoritmos en base al tamaño de la entrada: $f(n):\mathbf{N}\rightarrow\Re^{+}$. La notación asintótica independiza el análisis de los algoritmos de condiciones específicas de su implementación, lenguaje de programación, compilador, plataforma de ejecución, equipo de cómputo, etc.

Sean $\Re+$ el conjunto de reales estrictamente positivos, $\Re^*$.el conjunto de reales no negativos y N. el conjunto de números naturales y sea f: $\mathbf{N} \rightarrow \Re^*$ una función arbitraria. Se define *el orden de f(n)* como $O(f(n)) = \{t : N \rightarrow \Re^* \mid \exists c \in \Re^{+}, \exists n_o \in N \mid t(n) \leq cf(n), \forall n \geq n_o \}$

O(f(n)) representa el conjunto de todas las funciones t(n) acotadas superiormente por un múltiplo real positivo de f(n), siempre que n sea suficientemente grande (mayor que algún umbral n_o). Asimismo, se dice que una función g es del orden de f(n) cuando $\boldsymbol{g \in O(f(n))}$. El conjunto O (f(n)) define un orden de complejidad.

Sean f y g dos funciones definidas de $\mathbf{N} \rightarrow \Re^*$. Se dice que el orden de crecimiento de g es menor o igual que el de f, que se denota por g = O(f), si existe una constante $k > 0$ tal que $g(n) \leq k\ f(n)$, para todo $n \in N$ (excepto a lo sumo un número finito de valores).

En base a la notación asíntótica se demuestra la siguiente jerarquía de órdenes de complejidad

$$\boldsymbol{O(1) \subset O(log\ n) \subset O(n) \subset O(n\ log\ n) \subset O(n^2) \subset \cdots \subset O(n^k) \subset \cdots \subset O(2^n) \subset O(n!)}$$

Se consideran *razonables* en la práctica las complejidades siguientes:

$$\boldsymbol{O(1) \subset O(logn) \subset O(n) \subset O(n\,logn) \subset O(n^2)}$$

Y se consideran complejidades *tratables* las siguientes:

$$\boldsymbol{O(1) \subset O(logn) \subset O(n) \subset O(n\,logn) \subset O(n^2) \subset \cdots \subset O(n^k)}$$

Un *algoritmo de tiempo polinómico* es uno cuya complejidad temporal, en el caso peor, está acotada superiormente por un polinomio (en el tamaño de la entrada).

En la actualidad la comunidad científica ha aceptado como acuerdo general [PAPSTE 98] que un algoritmo es prácticamente útil para resolver un problema, solo si su complejidad crece *polinomialmente* con respecto al tamaño del problema.

Los algoritmos polinomiales son eventualmente más eficientes que los exponenciales, al crecer el tamaño del problema, su complejidad puede ser mejor. Los algoritmos polinomiales hacen mejor uso de los avances tecnológicos.

Las complejidades de orden superior a las polinómicas, se consideran *intratables*, por tanto podemos decir que los problemas de complejidad $O(n^k)$ son el límite de lo tratable hasta el momento.

La notación asintótica cumple una serie de propiedades que son de gran utilidad para el análisis de los algoritmos.

1. $O(f(n) + g(n)) = O(\max\{ f(n), g(n)\})$ **(**Regla de la suma)
2. $O(f(n))\mathbf{.}O(g(n))= O(f(n)\mathbf{.}g(n))$. **(**Regla del producto)
3. $O(c\ f(n)\ ) = O(\ f(n)\ )$
4. $O(\ f(n)\ ) = O(\ g(n)\ )$ si, y solo si, $f(n) \in O(\ g(n)\ )$ y $g(n) \in O(\ f(n)\ )$
5. $O(\ f(n)\ ) \subset O(\ g(n)\ )$ si, y solo si, $f(n) \in O(\ g(n)\ )$ pero $g(n) \notin O(\ f(n)\ )$
6. Si $f(n) \in O(\ g(n)\ )$ y $g(n) \in O(\ h(n)\ )$ entonces $f(n) \in O(\ h(n)\ )$
7. Si $\lim_{(n\to\infty)} f(n)/g(n) \in \Re^+$ entonces $O(\ f(n)\ ) = O(\ g(n)\ )$

Un importante hecho en el campo de la OC, obtenido a finales de la década de los 60, es la conjetura, aún no verificada, de que existe una clase de POC, de tal complejidad que cualquier algoritmo, que resuelva cada instancia de este problema, requiere un esfuerzo computacional que crece superpolinomialmente con respecto al tamaño de la instancia del problema. Durante los 70s, el desarrollo teórico de la Ciencia de la Computación ha permitido una formulación rigurosa de dicha conjetura. El resultado es la Teoría de la NP-Completitud, sobre la cual hay

numerosos trabajos científicos publicados; entre los trabajos más recientes podemos citar [GER06]

1.3 Heurísticas y Metaheurísticas.

Ha habido un incremento progresivo en la complejidad y tipos de problemas que deben solucionarse a través de métodos computacionales. Sin embargo, a pesar del rápido progreso tecnológico, la resolución de muchos problemas excede la capacidad tecnológica disponible. Muchos problemas del "mundo-real" pueden reducirse en su esencia a formulaciones abstractas bien conocidas para las cuales el número de soluciones potenciales crece exponencialmente con el número de variables consideradas. A pesar de que, para algunos de estos problemas, existen métodos exactos cuya complejidad escala linealmente (o al menos polinomialmente) con el número de variables, es ampliamente aceptado que para muchos tipos de problemas, no existen tales algoritmos. En consecuencia, a pesar del incremento gradual en el poder de cómputo, a partir de un cierto tamaño de los problemas, debemos abandonar la búsqueda de soluciones óptimas (comprobables) y encontrar otros métodos para encontrar buenas soluciones.

Más allá de los métodos exactos, encontramos una clase de métodos de búsqueda conocidos como heurísticas [11] o métodos de búsqueda adaptativos, que pueden considerarse como un conjunto de reglas para decidir qué solución potencial del espacio de búsqueda debe ser subsiguientemente generada y probada en el proceso de búsqueda.

Los métodos heurísticos, metaheurísticas o sencillamente heurísticas (este término deriva de la palabra griega *heuriskein* que significa encontrar o descubrir) se usan en el ámbito de la optimización para describir una clase de algoritmos de resolución de problemas.

La existencia de una gran cantidad y variedad de problemas difíciles, que aparecen en la práctica y que necesitan ser resueltos de forma eficiente, impulsó el desarrollo de procedimientos eficientes para encontrar buenas soluciones aunque no fueran óptimas. Estos métodos, en los que la rapidez del proceso es tan importante como la calidad de la solución obtenida, se denominan heurísticos o aproximados.

Se recogen en la literatura definiciones diferentes de algoritmo heurístico, entre las que puede destacarse la siguiente: un *método heurístico* es un procedimiento para resolver un problema de optimización bien definido mediante una aproximación intuitiva, en la que la estructura del problema se utiliza de forma inteligente para obtener una buena solución. -[BELMOR03], [11]

La concepción más común en Inteligencia Artificial [GCI04] es interpretar que heurístico es el calificativo apropiado para los procedimientos que, empleando conocimiento acerca de un problema y de las técnicas aplicables, tratan de aportar soluciones (o acercarse a ellas) usando una cantidad de recursos (generalmente tiempo) razonable. En un problema de optimización, aparte de las condiciones que deben cumplir las soluciones factibles del problema, se busca la que es óptima según algún criterio de comparación entre ellas. En Investigación de Operaciones, el término heurístico se aplica a un procedimiento de resolución de problemas de optimización con una concepción diferente. Se califica de heurístico a un procedimiento para el que se tiene un alto grado de confianza en que encuentra soluciones de alta calidad con un costo computacional razonable, aunque no se garantice su optimalidad o su factibilidad, e incluso, en algunos casos, no se llegue a establecer lo cerca que se está de dicha situación. Se usa el calificativo heurístico en contraposición a exacto, que se aplica a los procedimientos a los que se les exige que la solución aportada sea óptima o factible. En resumen, los métodos exactos proporcionan una solución óptima del problema mientras que los métodos heurísticos se limitan a proporcionar una buena solución no necesariamente óptima. Lógicamente, el tiempo invertido por un método exacto para encontrar la solución óptima de un problema difícil, si es que existe tal método, es de un orden de magnitud muy superior al del heurístico (pudiendo llegar a ser tan grande en muchos casos, que sea inaplicable).

Aunque hemos mencionado el caso de la resolución de un problema difícil, existen otras razones para utilizar métodos heurísticos, entre las que podemos destacar [11]:

- El problema es de una naturaleza tal que no se conoce ningún método exacto para su resolución o aunque exista un método exacto para resolver el problema, su uso es computacionalmente muy costoso.
- El método heurístico es más flexible que un método exacto; permitiendo, por ejemplo,

la incorporación de condiciones de difícil modelación.

- El método heurístico se utiliza como parte de un procedimiento global que garantiza el óptimo de un problema. Existen dos posibilidades:
    - El método heurístico proporciona una buena solución inicial de partida.
    - El método heurístico participa en un paso intermedio del procedimiento.

Unas heurísticas para resolver un problema de optimización pueden ser más generales o específicas que otras [GCI 04],[16]. Los métodos heurísticos específicos deben ser diseñados para cada problema, utilizando toda la información disponible sobre el mismo y el análisis de los modelos elaborados. Las heuristicas específicas bien diseñadas suelen tener un rendimiento significativamente superior al de las heurísticas generales. Las heurísticas más generales, presentan otro tipo de ventajas, como la sencillez, adaptabilidad y robustez de los procedimientos. Sin embargo, las heurísticas generales emanadas de las metaheurísticas pueden mejorar su rendimiento utilizando recursos computacionales y estrategias inteligentes.

El termino metaheurísticas se obtiene de anteponer a heurística el sufijo meta que significa "más allá" o "a un nivel superior". Las concepciones actuales [GCI04] de lo que es una metaheurística están basadas en las diferentes interpretaciones de lo que es una forma inteligente de resolver un problema. Las metaheurísticas son estrategias inteligentes para diseñar o mejorar procedimientos heurísticos muy generales con un alto rendimiento. El termino metaheurística apareció por primera vez en un artículo sobre búsqueda tabú de Fred Gloveren [GLO86]. A partir de entonces han surgido muchos procedimientos para resolver ciertos problemas que, al ampliar su campo de aplicación, han adoptado la denominación de metaheurísticas.

Las Hiperheurísticas [GCI04] pueden considerarse como herramientas o metaheurísticas de alto nivel que permiten controlar adaptativamente varias heurísticas o metaheurísticas de bajo nivel de conocimiento del problema, de forma que aplicando solo heurísticas fáciles de usar se pueden conseguir soluciones equiparables a las obtenidas por heurísticas de alto grado de conocimiento del problema.

En la actualidad existe un gran desarrollo y crecimiento de las metaheurísticas. Dentro de los

procedimientos relativamente consolidados y que han probado su eficacia sobre una colección significativa de problemas están:

- Búsqueda Tabú,
- Recocido Simulado
- Métodos Evolutivos, incluyendo los Algoritmos Genéticos y la Búsqueda Dispersa.
- Métodos de colonias de hormigas

Estos métodos permiten resolver, entre otros, problemas de optimización combinatoria. En el epígrafe siguiente se abordan los algoritmos genéticos.

1.4 Algoritmos Genéticos

Los algoritmos genéticos (AG) constituyen una estrategia de búsqueda de propósito general que permiten resolver problemas de naturaleza combinatoria. Son una parte de la computación evolutiva que a su vez es una creciente rama de la Inteligencia Artificial. [6], [7], [13]. Los AG mantienen una variedad de posibles soluciones para el problema a considerar y usan los principios de la evolución natural: adaptación y supervivencia del más adaptado para formar la siguiente generación de posibles soluciones.

En los AG cada individuo de cada generación recibe un calificativo, una medida de su grado de adaptación (*fitness*), un número real no negativo que es mayor en la medida que sea mejor la solución propuesta por dicho individuo. Este número permite distinguir las buenas soluciones de las que no lo son. Como a cada individuo se le asigna una y sola una calificación se está hablando de una función, que recibe el nombre de *función de adaptación*.

Estas soluciones [12], [14], [17], se construyen utilizando algunos operadores reproductivos, tradicionalmente el *operador de cruzamiento (cruce)* y el *operador de mutación*. Inicialmente, se trata de combinar las soluciones con mayor *fitness* de los individuos de la población para obtener así, a partir de dos soluciones, otras dos nuevas soluciones.

Aunque los AG se han aplicado a una amplia variedad de problemas, se ha demostrado que no son mejores que cualquier otro método de búsqueda de solución (incluido el método aleatorio)

si no se les proporciona alguna información inicial del problema. Esencialmente, los elementos del algoritmo deben ser cuidadosamente elegidos para cumplir las características de la solución del problema.

Los algoritmos evolutivos basan parte de sus buenos resultados en el balance entre una eficiente exploración y una eficiente explotación cuando se resuelve un problema difícil. La exploración se refiere a la capacidad de mostrar diferentes partes del espacio de búsqueda en la población del algoritmo, mientras la explotación se refiere a la capacidad de refinamiento y combinación de las soluciones subóptimas. La exploración es útil para evitar estancarse en óptimos locales mientras que la explotación se usa para obtener el óptimo global una vez que se ha aproximado a él lo suficiente.

En las etapas iniciales de la búsqueda, un algoritmo genético debe mostrar una gran diversidad, mientras que al final la diversidad debe disminuir para conseguir una solución.

Para tratar de mejorar la velocidad de convergencia del algoritmo genético se puede utilizar la llamada *Presión Selectiva*, que es aquella que se ejerce cuando bien en el proceso de selección o en el proceso de aceptación se utiliza un método basado en el *fitness* como puede serlo el de la ruleta o el torneo.

Los AG modelan el proceso de evolución como una sucesión de frecuentes cambios en los ***genes***, con soluciones análogas a ***cromosomas.*** El espacio de soluciones posibles es explorado aplicando transformaciones a éstas soluciones candidatas tal y como se observa en los organismos vivientes: ***cruce, inversión, mutación.***

Los AG constituyen el paradigma más completo de la computación evolutiva al resumir de modo natural todas las ideas fundamentales de dicho enfoque. Son muy flexibles ya que pueden adoptar con facilidad nuevas ideas, generales o específicas, que surjan dentro del campo de la computación evolutiva. Se pueden hibridar fácilmente con otros paradigmas y enfoques, aunque no tengan ninguna relación con la computación evolutiva.

1.4.1 Operadores de un Algoritmo Genético

Antes de poner en práctica un AG es necesario decidir qué operadores genéticos se van a utilizar. Esta decisión depende en gran medida de la estrategia de codificación. Analizaremos solamente tres de ellos el operador de selección, el operador de cruzamiento y el de mutación que son los que se implementan en la tesis. Aunque en la mayoría de sus aplicaciones los AG utilizan además de la selección, solamente el cruce y la mutación, muchos otros operadores y las estrategias para aplicarlos se han explorado en la literatura [MIT96].Éstos incluyen la inversión de genes y varios operadores para preservar diversidad en la población

Operador de selección

Como ya se ha visto los individuos se seleccionan para reproducirse, ahora bien el problema está en cómo seleccionar. De acuerdo con la teoría de la evolución de Darwin, sólo los mejores individuos se reproducen. Basándose en esto, existen varios métodos que son utilizados por los algoritmos genéticos [7]:

- Selección por la *Regla de la Ruleta***.** Los padres se seleccionan de acuerdo a su *fitness*. Los individuos mejores (con mayor *fitness*) son los que tienen mayores posibilidades de ser elegidos. Intuitivamente el proceso construye una ruleta o una "tarta" en la que cada uno de las porciones representa a un individuo. La porción de tarta que le toca a cada individuo es proporcional a su *fitness*. Así los individuos buenos se llevarán las mayores porciones y al revés ocurrirá con los peores.

Ahora, al igual que en un casino, se lanza a la ruleta una canica. En el lugar que pare dicha canica, será un lugar ocupado por un cromosoma que será elegido. Resulta claro que los individuos con mayor *fitness* son los que más a menudo son elegidos.
Existe un algoritmo para realizar este proceso:

1. [SumaTotal] Calcular la suma total acumulada de los *fitness* de todos los individuos de la población actual.
2. [Elegir un número aleatorio r] Generar un número aleatorio entre 0 y la SumaTotal.

3. [Recorrer] Recorrer la población acumulando nuevamente los *fitness*. Cuando la suma que se lleve sea mayor o igual a **r** seleccionamos el individuo donde se vaya recorriendo.

Selección por *Ranking*. El anterior tipo de selección funciona mal cuando existan grandes diferencias entre los *fitness* de los individuos de la población [7]. Por ejemplo, si un cromosoma ocupa el 90% de la ruleta el resto de los cromosomas tienen muy pocas posibilidades de ser elegidos. La selección por *ranking* da solución a este problema. Los individuos son ordenados de acuerdo a su *ranking* de *fitness*. De esta manera, si tenemos *n* cromosomas, el individuo con peor *fitness* se le asignará un 1 y el que tenga el mejor *fitness* se le asignará la *n*. Este tipo de selección le da a todos los cromosomas la oportunidad de ser seleccionados, sin embargo, puede hacer que el genético converja lentamente a la solución, ya que los mejores individuos no se diferencian apenas de los peores.

- Selección por *Torneo K/L*. Consiste en seleccionar K individuos de la población aleatoriamente y de estos K individuos se seleccionan los L que tengan mejor *fitness*.

Este proceso se repite todas las veces necesarias hasta formar la nueva población. Este es uno de los métodos de selección más utilizados actualmente. Se utiliza también en algunos algoritmos en el momento de la aceptación.

Operador de cruzamiento.

Los operadores de cruzamiento crean una nueva generación de individuos de descendientes pidiendo información a sus ancestros.

Los cruzamientos más utilizados son -[KURGAL98]

1. *Cruzamiento en un punto***.** Se elige un punto de corte y se intercambian los segmentos análogos de las dos cadenas.
2. *Cruzamiento en dos puntos*. Se eligen dos puntos de corte y se intercambian los segmentos medios de ambas cadenas. En otras palabras, se copian los genes del primer

padre comprendidos entre los dos puntos de cruce y se rellenan los que faltan con los del segundo padre considerando la cadena de genes como cíclica.

3. *Cruzamiento uniforme*. Para cada posición de bit de una cadena a generar se elige aleatoriamente el bit de la misma posición de alguna de las cadenas generadoras. Se escoge aleatoriamente si el gen i-ésimo del hijo se toma del primer o del segundo padre.

Operador de mutación

El operador de mutación modifica los genes con probabilidad determinada.

Los tipos de mutaciones más usadas [7]

1. *Inversión de genes***.** Se seleccionan genes aleatoriamente y se invierte su valor. Se utiliza en representaciones de bits, cambiando ceros por unos, o viceversa.
2. *Cambio de orden***.** Se seleccionan dos genes aleatoriamente y se intercambian sus posiciones.
3. *Modificación de genes***.** Se realizan pequeñas modificaciones en los genes

1.4.2 Parámetros de los AG

Los parámetros de un AG son [7], [17]

1. Porcentaje de Cruce (Pc). Indica con qué frecuencia se cruzarán los individuos. Si éste es 0%, los hijos serán como los padres y sólo serán alterados por la mutación. Si éste es 100% todos los individuos nuevos serán creados mediante cruce de los padres de la generación previa. Cuanto más se crucen los individuos se supone que los hijos serán mejores. Sin embargo, es recomendable, que algunos individuos pasen sin modificar a la siguiente generación.

2. Porcentaje de Mutación (Pm). Probabilidad con la cual los individuos serán mutados. La mutación trata de impedir que la búsqueda del AG caiga en extremos locales, por eso es conveniente que ocurra de vez en cuando. Tampoco es bueno que la mutación ocurra

continuamente, ya que la búsqueda puede pasar de ser "inteligente" a búsqueda aleatoria.

3. Tamaño de la Población (tam-pob). Cantidad individuos en cada una de las generaciones. Si el tamaño de la población es muy bajo, el algoritmo genético tiene pocas posibilidades de evolucionar ya que los hijos se parecerán mucho a sus padres. Si el tamaño es muy grande se llega a un punto en el que los resultados no mejoran por mucho que se incremente el tamaño de la población. Lo ideal es establecer un límite adecuado del tamaño de la población, en función del problema y de la codificación usada.

4. Número de Generaciones (nro-gen). Con el paso de las generaciones la población del genético evolucionará obteniendo cada vez mejores individuos. Conviene fijar un número de generaciones adecuado para conseguir el resultado deseado.

5. Tamaño del Individuo (tam-ind). Dependerá del número de elementos que constituyan una solución. Varía con el problema a resolver.

La ventaja de los AG radica en su paralelismo. Un algoritmo genético viaja por el espacio de búsqueda utilizando varios individuos, lo que hace que les resulte más difícil quedarse estancados en óptimos locales que a otros métodos.

La desventaja que presentan los algoritmos genéticos es el tiempo de computación. Es importante tener en cuenta que los algoritmos genéticos no son métodos completos, es decir, no se puede asegurar su convergencia a la mejor solución.

1.4.3 Estructura de los AG.

Un AG tiene, en general la siguiente estructura [7]:

Leer Parámetros; {Pc, Pm, tam-pob, nro-gen, tam-ind}
Generar la población inicial P (0);
Evaluar P (0);
Mientras no última-generación hacer
Generar una nueva población P (t) a partir de la población anterior aplicando los operadores de Selección, Cruce y Mutación;

Evaluar P (t);

*Fin mientras***;**

Devuelve el mejor individuo de la última población;

Fin.

1.4.4 Fundamentos matemáticos de los AG

Los AG son mecanismos de búsqueda que operan sobre un conjunto posiblemente muy grande pero finito de códigos. El AG opera sobre genotipos que se deberán mapear al espacio de soluciones. El dominio del problema se codifica para obtener estructuras manejables que pueden ser manipuladas por el AG. Cada una de estas estructuras constituye el equivalente al genotipo de un individuo en términos biológicos, el elemento del dominio al que le corresponde ese genotipo es el análogo al fenotipo. Para codificar los elementos del dominio se utiliza con frecuencia un alfabeto binario (0 y 1).

El AG recibe como entrada una población de códigos y a partir de ella generará nuevas poblaciones donde algunos códigos desaparecerán mientras otros, que se "mapean" en mejores posibles soluciones aparecen con más frecuencia hasta que se encuentra una satisfactoria o hasta que se cumpla alguna otra condición de terminación.

En los AG cada individuo de cada generación recibe una calificación, una medida de su grado de adaptación (*fitness*), un número real no negativo que es mayor en la medida que sea mejor la solución propuesta por dicho individuo. Este número permite distinguir las buenas soluciones de las que no lo son .Como a cada individuo se le asigna una y sola una calificación se está hablando de una función, que recibe el nombre de función de adaptación.

Evaluar está función es lo que más tiempo consume en la ejecución de un AG en la mayoría de los casos.

Esquemas

Los *esquemas* son cadenas binarias (cromosomas) compuestas de tres símbolos {0,1,*}. Consideremos un conjunto de cadenas que poseen valores comunes en ciertas posiciones [KURGAL98]. Para construir la cadena que denota este conjunto, basta colocar en las posiciones donde las cadenas coinciden el valor explicito que tienen y en las posiciones donde los valores no coinciden se coloca un *.

Por ejemplo, 1*0* denota el conjunto {1000; 1001; 1100; 1101}

Propiedades de los esquemas

- Orden del esquema: es el número de posiciones con valor explicito de un esquema H o el número de posiciones fijas en un esquema y se denota por O(H).
- Longitud de definición: σ(H) es la distancia entre la primera y la última posición explicita, la distancia entre la primera y la última posición fija de un esquema.

El orden es una medida de especificidad y la longitud de definición es una medida de la probabilidad de sobrevivencia de un esquema al aplicar *crossover* (cruzamiento).

Los esquemas y sus propiedades proporcionan los medios básicos para analizar el efecto neto de la reproducción y los operadores genéticos en los bloques de construcción de la población (esquemas cortos y de alto grado de adaptación).

Algoritmo Genético Simple (SGA)

Aunque existen en la actualidad varias propuestas de cómo seleccionar individuos y de cruzarlos, las propuestas de Holland constituyen aún hoy la base de muchos desarrollos teóricos y prácticos. Goldberg retomó el algoritmo de Holland y lo popularizó llamándolo Algoritmo Genético simple (SGA)

El SGA se caracteriza por tener:

- Tamaño de la población fijo en todas las generaciones.
- Selección proporcional (de rueda de la ruleta).

- Cruzamiento de un punto. La probabilidad de un cruzamiento se mantiene fija para todas las generaciones y todas las parejas.
- Mutación uniforme. La probabilidad de mutación permanece fija para todas las generaciones y todas los loci [2] de los individuos.

Teorema fundamental de los AG (Teorema del esquema)

El modelo matemático más usual para describir el comportamiento de los AG está basado en el teorema del esquema .Este teorema fue planteado inicialmente por Holland [HOL75] para el SGA.

El teorema del esquema analiza el efecto de los operadores (selección, cruzamiento y mutación) sobre los esquemas. El teorema tiene varias expresiones todas equivalentes. El teorema que analizamos es más general que en [MIT 96] y [GOL89]

Supongamos que en una población de tamaño *N* en la generación *t* de un AG .se tienen *k* representantes de un esquema H y que $f_1, f_2, f_3, \cdots, f_k$ son los valores de la función de adaptación para los k representantes del esquema [KURGAL98].

$$P_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$$

La probabilidad de seleccionar algún representante del esquema H es

$$P_H = \frac{f_1}{\sum_{j=1}^{N} f_j} + \frac{f_2}{\sum_{j=1}^{N} f_j} + \cdots + \frac{f_k}{\sum_{j=1}^{N} f_j} = \frac{f_1 + f_2 + \cdots + f_k}{\sum_{j=1}^{N} f_j} \qquad (1.11)$$

Se supone que cada selección es un evento independiente

[2] Ver Anexo 8

Sean $\overline{f}(H)$ el valor de adaptación promedio de los representantes del esquema H y m (H , t) el número de representantes del esquema en la población de generación t (i.e m (H , t) = k) Por definición

$$\overline{f}(H) = \frac{f_1 + f_2 + \cdots + f_k}{m(H,t)}$$

Sea $\overline{f}$ el valor promedio de adaptación de la población de generación *t*

$$\overline{f} = \frac{\sum_{j=1}^{N} f_j}{N} \text{ de donde } N\overline{f} = \sum_{j=1}^{N} f_j$$

Entonces, si se seleccionan N individuos de la población de generación *t* para la generación siguiente, el valor esperado de individuos seleccionados del esquema H es:

$$m(H, t+1) = N\, m(H,t) \frac{\overline{f}(H)}{\sum_{j=1}^{N} f_j} = m(H,t) \frac{\overline{f}(H)}{\overline{f}} \qquad (1.12)$$

Estas expresiones son válidas para el tipo de selección proporcional

Cruzamiento

Supóngase ahora que se aplica algún tipo de cruzamiento con cierta probabilidad p_c sobre los individuos previamente seleccionados. El valor esperado de cadenas representantes del esquema H que han sido seleccionadas y a las que no se les aplica el cruzamiento es

$$(1 - p_c)\, m(H,t) \frac{\overline{f}(H)}{\overline{f}} \qquad (1.13)$$

Sea p_r la probabilidad de ruptura del esquema H bajo el tipo de cruzamiento que este siendo utilizado. Esta probabilidad depende del tipo de cruzamiento del orden del esquema $O(H)$ y

de la longitud de definición $\delta(H)$.El valor esperado del número de cadenas representantes de H que fueron seleccionadas y permanecen en el esquema después del cruzamiento es

$$p_c\left(m(H,t)\frac{\overline{f}(H)}{\overline{f}}(1-p_r)\right) \qquad (1.14)$$

Si consideramos aquellas cadenas que originalmente estaban fuera del esquema y que después de cruzarlas generaron representantes de él .Sea *g* el número de estas cadenas. La fórmula (1.14) quedaría

$$p_c\left(m(H,t)\frac{\overline{f}(H)}{\overline{f}}(1-p_r)+g\right) \qquad (1.15)$$

Resumiendo (1.13) y (1.15) el valor esperado del número de representantes del esquema H tras haber efectuado selección y cruzamiento es

$$m(H,t+sel+cro) = (1-p_c)m(H,t)\frac{\overline{f}(H)}{\overline{f}} + \\ + \quad p_c\left(m(H,t)\frac{\overline{f}(H)}{\overline{f}}(1-p_r)+g\right)$$

Excluyendo *g* y extrayendo factor común *m* (H, t) se obtiene la expresión (1.16)

$$m(H,t+sel+cro) \geq m(H,t)\frac{\overline{f}(H)}{\overline{f}}\left[(1-p_c) + p_c(1-p_r)\right] \qquad (1.16)$$

Mutación

Supongamos que se aplica una mutación, con probabilidad p_m, que tiene el efecto de invertir un bit (cambiar 1 por un 0 o viceversa).

Para que una cadena representante del esquema H permanezca en el tras una mutación debe ocurrir que ninguno de los bits definidos del esquema sea invertido. La probabilidad de que un bit no sea invertido por una mutación es $1-p_m$, así que la probabilidad de que ninguno de los bits definido sea invertido , suponiendo que el invertir cada bit es un evento independiente es :

$$\mu\left(p_m\right)=\left(1-p_m\right)^{0(H)} \qquad (1.17)$$

Añadiendo (1.17) a la expresión (1.16), se tiene

El valor esperado del número de representantes del esquema H tras haber efectuado selección, cruzamiento y mutación en la generación siguiente es:

$$m\left(H,t+1\right) \geq m\left(H,t\right)\frac{\overline{f}\left(H\right)}{\overline{f}}\left(1-p_c\,p_r\right)\left(1-p_m\right)^{O(H)} \qquad (1.18)$$

A esta expresión se le conoce como el teorema del esquema, existen versiones de este teorema, todas equivalentes. [KURGAL98]

Si se divide todo por el tamaño de la población, todo quedaría expresado en términos de probabilidad:

$$p\left(H,t+1\right) \geq p\left(H,t\right)\frac{\overline{f}\left(H\right)}{\overline{f}}\left(1-p_c\,p_r\right)\left(1-p_m\right)^{O(H)} \qquad (1.19)$$

En el teorema del esquema los esquemas más pequeños (de longitud definida corta respecto a la longitud de la cadena) sobreviven más fácilmente que los esquemas grandes ante el operador de cruzamiento, además la selección favorece esquemas con una alta calificación (de alto *fitness*).Este par de indicios lleva a pensar que las soluciones encontradas por un AG se construyen a lo largo de generaciones mezclando esquemas cortos y de alto grado de adaptación, a estos se les denomina bloques constructivos y a esta suposición la hipótesis de los bloques constructivos.

1.5 Conclusiones del capítulo.

En este capítulo han sido abordados los aspectos teóricos fundamentales que sirven de sustento a la tesis: los problemas de optimización combinatoria, los métodos exactos y aproximados de solución de los problemas de optimización y el análisis de complejidad de los algoritmos de solución de los problemas.

Dentro de los métodos de solución de los modelos se abordan los métodos exactos, los métodos aproximados, las heurísticas modernas. Se hace un estudio detallado de una de estas heurísticas los Algoritmos Genéticos, la estructura de los AG, sus operadores, parámetros, así como los fundamentos matemáticos de los mismos.

Es conveniente destacar que los AG se han utilizado en la resolución de problemas complicados, como los problemas NP-duros, para el aprendizaje automático y también para programas de evolución sencillos. Los algoritmos genéticos son bastante sencillos de implementar. Una vez que se ha implementado un algoritmo genético básico, tan sólo hay que implementar el nuevo cromosoma para resolver otro problema. Si se utiliza la misma codificación sólo se tendrá que programar la nueva función de *fitness*.

Los AG pueden resultar más lentos que otros métodos, si se les deja evolucionar hasta que alcancen la solución óptima; pero, teniendo en cuenta que se puede finalizar su ejecución cuando se desee y obtener una solución bastante buena y que los ordenadores cada vez son más rápidos, esto no supone una gran desventaja. Es importante tener en cuenta que estos algoritmos no son métodos completos, es decir, no se puede asegurar su convergencia a la mejor solución.

2 Modelos de optimización de los problemas

Ya hemos señalado en el capitulo 1 que la aparición de la computación determina el desarrollo de las Matemáticas Aplicadas Modernas, pero no es posible confiar solamente en el aumento de la velocidad del supercomputador electrónico, es necesario perfeccionar la tríada *modelo – algoritmo - programa,* o sea el núcleo intelectual del experimento de cálculo. En la solución de los problemas prácticos utilizando la modelación matemática y el experimento de cálculo surgen nuevos y difíciles problemas, relacionados con el estudio de los modelos y con la elaboración de algoritmos de cálculo eficaces que estimulan el desarrollo de las Matemáticas Aplicadas.

Una vez formulados los problemas de optimización a los que les damos solución en la tesis es necesario pasar a la siguiente etapa que consiste en reformularlos de manera conveniente para su análisis. La forma convencional en que la Programación Matemática hace esto es construyendo un modelo matemático que represente la esencia del problema a resolver.

Los modelos, o representaciones idealizadas, son una parte integral de la vida diaria. Los modelos juegan un papel muy importante en la ciencia y los negocios, como lo hacen patente los modelos del átomo y de las estructuras genéticas, las ecuaciones matemáticas que describen las leyes físicas del movimiento o las reacciones químicas, las gráficas, los organigramas y los sistemas contables en la industria. Estos modelos son invaluables, ya que extraen la esencia de la materia de estudio, muestran sus interrelaciones y facilitan el análisis.

Los modelos matemáticos [HILLIE97] también son representaciones idealizadas, pero están expresados en términos de símbolos y expresiones matemáticas. Así, si se pueden tomar *n* decisiones cuantificables relacionadas unas con otras, estas se representan como *variables de decisión* para las que se deben determinar los valores respectivos. La medida de desempeño adecuada se expresa, entonces, como una función matemática de estas variables de decisión. Esta función se llama *función objetivo*. También se expresan matemáticamente todas las

limitaciones que se puedan imponer sobre los valores de las variables de decisión, casi siempre en forma de ecuaciones o desigualdades. Tales expresiones matemáticas de las limitaciones, con frecuencia, reciben el nombre de *restricciones*. Las constantes en las restricciones y en la función objetivo se llaman parámetros del modelo. El modelo matemático puede expresarse, entonces, como el problema de elegir los valores de las variables de decisión de manera que se minimice o maximice la función objetivo, sujeta a las restricciones dadas.

Aun cuando se hable de "el modelo matemático de un problema", los problemas reales por lo general no tienen "un solo modelo correcto". Es posible que se puedan desarrollar varios modelos completamente diferentes para ayudar a analizar el mismo problema.

Un paso crucial en la formulación de un modelo de IO es la construcción de la función objetivo. Esto requiere desarrollar una medida cuantitativa de la efectividad relativa a cada objetivo del que toma las decisiones, identificado cuando se estaba definiendo el problema. Si el estudio contempla más de un objetivo, es necesario transformar y combinar las medidas respectivas en una medida compuesta de efectividad llamada medida global de efectividad.

Es inevitable que la primera versión de un modelo matemático grande tenga muchas fallas. Puede que algunos factores o interrelaciones relevantes no se incorporaron al modelo y algunos parámetros no se estimaron correctamente. Antes de usar el modelo, este debe probarse exhaustivamente para intentar identificar y corregir todas las fallas que se pueda.

2.1 Modelación de los problemas formulados

2.1.1 Modelo Combinatorio del Problema 1

Este problema puede ser modelado usando conceptos de la Teoría Combinatoria y como un problema de Programación No Lineal en Enteros.

Representación: Cada posible solución es una variación con repetición, que puede ser codificada como una función de la forma:

$$x = \left(aa_{i(1)}, aa_{i(2)}, \ldots, aa_{i(n)}\right) \quad con \quad aa_{i(k)} \in \left\{aa_1, \ldots, aa_{20}\right\} \quad k = \overline{1,n}$$

Espacio de búsqueda: *S* es el conjunto de variaciones de un conjunto de 20 elementos, siendo su cardinal **|S| = 20 n**

Objetivo: Buscar la variación ***x*** que maximiza la información que contiene la cadena por unidad de masa, o sea maximizar la función ***r(x).***

Función de Evaluación:

$$r(x) = \frac{-\sum_{k=1}^{n} \log_4 p\left(aa_{i(k)} / aa_{j(k-1)}\right)}{\sum_{k=1}^{n} m_{aa_{i(k)}} - (n-1) * 0.01802} \qquad (2.1)$$

Para este modelo podemos aplicar métodos exactos de enumeración como el de Ramas y Cotas. Agregando al modelo una estructura de vecindades, se pueden aplicar métodos de búsqueda local, como son el escalador de colinas, el escalador de colinas estocástico, la búsqueda tabú y el recocido simulado. A partir de este modelo combinatorio se pueden implementar también los algoritmos evolucionarios, que parten de un conjunto de elementos de S que van mejorando a través de lazos sucesivos de evaluación selección y variación. Estos son también enfoques metaheurísticos.

2.1.2 Modelo de programación no lineal del problema 1

Si representamos cada variación con repetición por un tensor $\lfloor \partial_{k\,i\,j} \rfloor$, donde

$$\partial_{k\,i\,j} = \begin{cases} 1 & \text{si el aminoácido } i \text{ está en la posición } \boldsymbol{k} \text{ y en la posición } \boldsymbol{k}-1 \\ & \text{está el aminoácido } j;\ k = \overline{2,n}\ ;\ j = \overline{1,20}\ ;\ i = \overline{1,20} \\ 1 & \text{si el aminoácido } \boldsymbol{i} \text{ está en la posición } 1;\ k = 1 \text{ y } j = 0 \\ 0 & e.o.c \end{cases}$$

de la expresión (2.1) se obtiene que la función a maximizar es

$$r(x) = \frac{-\sum_{i=1}^{20}\sum_{j=0}^{20}\sum_{k=1}^{n} \log_4 P\left(aa_{ik} / aa_{jk-1}\right)\partial_{kij}}{\sum_{i=1}^{20}\sum_{j=0}^{20}\sum_{k=1}^{n} m_{aa_i}\partial_{kij} - (n-1)\,0.01802} \qquad (2.2)$$

siendo

$P\left(aa_{ik} / aa_{jk-1}\right)$ la probabilidad de que el aminoácido *i* esté en la posición *k*, estando el aminoácido *j* en la posición anterior.

$P\left(aa_{i1} / aa_{j0}\right)$ la probabilidad de que el aminoácido *i* se encuentre en la primera posición de la cadena.

Deben cumplirse las siguientes restricciones:

- En cada posición *k* hay un, y solo un, aminoácido:

$$\sum_{i=1}^{20}\sum_{j=0}^{20}\partial_{kij} = 1 \quad \forall k = \overline{1,n}$$

- Si en la posición *k*+1 está el aminoácido α antecedido del aminoácido *i*, necesariamente en la posición *k* está el aminoácido *i* :

$$Si \;\; \exists \alpha : \partial_{k+1,\alpha,i} = 1 \;\Rightarrow\; \exists \beta : \partial_{k,i,\beta} = 1 \quad ; \forall k = \overline{1,n-1}$$

2.1.3 Modelo combinatorio del Problema 2

Este problema puede ser modelado usando conceptos de la Teoría Combinatoria y como un problema de Programación No Lineal en Enteros.

Representación: Cada posible solución es una combinación que puede ser codificada como una función de

$$x = \left(aa_{i(1)}, aa_{i(2)}, \ldots, aa_{i(n)}\right) \quad \text{con} \quad aa_{i(k)} \in \left\{aa_1, \ldots, aa_{20}\right\} \quad k = \overline{1,n}$$

Espacio de búsqueda: ***S*** es el conjunto de n combinaciones de un conjunto de 20 elementos, siendo su cardinal $|S| = \frac{20^{\,n}}{n!}$

Objetivo: Buscar la combinación $\boldsymbol{x}$ que maximiza la complejidad de la cadena.

Función de Evaluación:

$$\left[C_2(X)\right]^2 = n\left[\frac{\sum_{k=1}^{n}\left(-\log_4 p_{i(k)}\right)^2}{\left(\sum_{k=1}^{n}-\log_4 p_{i(k)}\right)^2} - \frac{1}{n}\right] \qquad (2.3)$$

Para este modelo podemos aplicar métodos exactos de enumeración como el de Ramas y Cotas, donde la cota superior del número de operaciones elementales es del orden del cardinal del espacio de búsqueda.

Al igual que para el Problema 1, agregando al modelo una estructura de vecindades, se pueden aplicar métodos de búsqueda local, como son el escalador de colinas, el escalador de colinas estocástico, la búsqueda tabú y el recocido simulado. Estas técnicas son realmente meta heurísticas que permiten obtener con bajo costo computacional soluciones suboptimales con una cierta calidad, a partir de ir mejorando un elemento inicial de S; pero que no garantizan la aproximación a la óptima. A partir de este modelo combinatorio se pueden implementar también los algoritmos evolutivos.

2.1.4 Modelo de Programación no lineal del problema 2

Variables de decisión: ∂_i ; $i = \overline{1, 20}$. Representa el número de veces que aparece el ***aa$_i$*** en la cadena.

Objetivo: Maximizar la función

$$[C_2(X)]^2 = n \left[\frac{\sum_{i=1}^{20} \left(-\log_4 p_i \partial_i \right)^2}{\left(\sum_{i=1}^{20} -\log_4 p_i \partial_i \right)^2} - \frac{1}{n} \right] \quad (2.4)$$

Sujeto a las restricciones:

$$\sum_{i=1}^{20} \partial_i = n$$

$\partial_i \geq 0$, enteras. $i = \overline{1..20}$

2.1.5 Modelo combinatorio del Problema 3

Este problema fue el último planteado por los especialistas, la investigación se encontraba en una fase en la que ya se habia analizado que era muy costoso desde el punto de vista computacional la aplicación de métodos exactos y por eso solo se elaboró para este problema el modelo combinatorio que es a partir del cual se implementan los algoritmos geneticos.

Representación: Cada posible solución es una variación que puede ser codificada como una función de

$$x = \left(aa_{i(1)}, aa_{i(2)}, \ldots, aa_{i(n)} \right) \quad con \quad aa_{i(k)} \in \left\{ aa_1, \ldots, aa_{20} \right\} \quad k = \overline{1, n}$$

Espacio de búsqueda: *S* es el conjunto de variaciones de un conjunto de 20 elementos, su cardinal es $|S| = 20^n$

Objetivo: Buscar el vector ***u*** correspondiente a la variación ***x*** que maximiza la siguiente

Función de Evaluación:

$$f\left(u(x)\right) = \frac{n\,\sigma\left(u(x)\right)}{M(x)}$$

que puede ser expresada como

$$f\left(u(x)\right) = \frac{\sqrt{n\sum_{k=1}^{n}\left(u_k\right)^2 - \left(\sum_{k=1}^{n} u_k\right)^2}}{\sum_{k=1}^{n} m_{a\,a_{i(k)}} - (n-1) * 0,01802} \qquad (2.5)$$

Para este modelo se pueden aplicar también métodos exactos de enumeración como el de Ramas y Cotas, métodos de búsqueda local, como son el escalador de colinas, el escalador de colinas estocástico, la búsqueda tabú y el recocido simulado y algoritmos evolutivos como los algoritmos genéticos[3].

2.2 Conclusiones del capítulo

La modelación matemática es una ciencia aplicada que se ocupa de los métodos para la formalización de los procesos y fenómenos. La posibilidad de construir modelos matemáticos de los objetos reales, está dada por el nivel de desarrollo alcanzado en las diferentes teorías matemáticas, así como también por el grado de madurez de los conocimientos sobre su objeto de estudio, alcanzado en las diferentes ciencias particulares.

El modelo matemático está siempre basado en una cierta simplificación, e idealización, no es idéntico al objeto, sino su reflejo aproximado. Sin embargo, gracias a la sustitución de un objeto real por el modelo correspondiente, aparece la posibilidad de formular su estudio como un problema matemático y poder utilizar las herramientas universales de la matemática, que no dependen de la naturaleza concreta del objeto.

[3] Ver epígrafe 1.4

Un número excesivo de simplificaciones puede conllevar a que los resultados que obtengamos con el modelo no constituyan una buena aproximación de la realidad. No debemos perder de vista que, las soluciones obtenidas con el modelo matemático son soluciones para éste y serán, por tanto, una buena aproximación de la realidad, en tanto el modelo lo sea.

También debemos tener en cuenta que un determinado objeto real, puede ser modelado con diferentes objetos matemáticos, es decir que podemos obtener para él diferentes modelos matemáticos. Cada modelo tiene asociado sus técnicas propias de trabajo. Por tanto, nos parece una buena idea que al tratar de resolver un problema complejo, analicemos todas las variantes de modelación a nuestro alcance y valoremos la eficiencia de las técnicas de solución, para el problema particular que nos ocupa.

En el capítulo 1 se abordó, de manera general, la complejidad de los algoritmos; en el Capítulo 3 se presenta la complejidad del algoritmo utilizado para resolver los problemas propuestos.

3 Aplicación de los Algoritmos Genéticos y análisis de los resultados

Ya señalábamos en el Capítulo 2 que los problemas a resolver son difíciles, por eso utilizamos heurísticas, en especial AG, que se presentan como una solución alternativa para este tipo de problema.

Un AG es una técnica de programación que imita a la evolución biológica como estrategia para resolver problemas. Mientras el poder de la evolución gana reconocimiento cada vez más generalizado, los algoritmos genéticos se utilizan para abordar una amplia variedad de problemas en un conjunto de campos sumamente diverso demostrando su capacidad y su potencial.

Los AG han encontrado aplicaciones en esferas como la Acústica, la Astronomía y la Astrofísica; en Química; en Ingeniería Eléctrica; en los mercados financieros; en el diseño de redes neuronales que pueden jugar a las damas; en Geofísica; en la Ingeniería de materiales; en las Matemáticas y la Algoritmia; en el Ejército y cumplimiento de la ley; en Biología Molecular en el diseño de un algoritmo que permite identificar el dominio transmembrana de una proteína; en el reconocimiento de patrones y explotación de datos; en Robótica; en el diseño de rutas y horarios; en Ingeniería de sistemas. Muchos de los problemas de optimización planteados en la Biología molecular son problemas que han sido resueltos utilizando AG.

Al trabajar los AG utilizamos los siguientes términos:

- genes - cromosomas (individuos)
- poblaciones (conjunto de cromosomas)
- operadores (selección, cruzamiento, mutación)
- la función de adaptación (Fitness)

En el presente problema se desea encontrar la secuencia de aminoácidos de longitud N que maximiza una función estimadora potencial de la complejidad (función EPCP) de dicha secuencia.

El procedimiento general:

1. inicializa y evalúa la población
2. si la solución encontrada es aceptable, entonces es la solución de salida y acaba.
3. se seleccionan los individuos
4. aplicar el operador de cruzamiento
5. aplicar el operador de mutación
6. evaluar, reemplazar u volver al paso 2.

Se utilizó la representación usual de los aminoácidos mediante símbolos de una letra para organizarlos alfabéticamente. Este orden nos permite, a su vez, codificar cada aminoácido por su índice $\{aa_1, aa_2, \ldots, aa_{20}\} = \{1,2,..,20\}$. De esta manera cada secuencia de aminoácido corresponderá a una lista de números enteros entre 1 y 20. Estas listas corresponden a los fenotipos de nuestro algoritmo y nos facilitan la evaluación de la función fitness.

Los genotipos (cromosomas) empleados en el algoritmo genético se obtienen de la representación binaria de las listas de los fenotipos. Es decir, cada índice se representa mediante un número de 5 dígitos en este sistema. De manera que si $\overline{a_0 a_1 a_2 a_3 a_4}\ (a_i \in \{0,1\})$ es la representación binaria del número entero $x \in \{1,..,20\}$, entonces: $x = \sum_{i=0}^{4} a_i 2^{4-i}$

Si $a_0 = a_1 = a_2 = a_3 = a_4 = 1$ entonces $\overline{a_0 a_1 a_2 a_3 a_4} = 31$. Como son solo 20 aminoácidos se aplica una congruencia módulo 20 y al resultado se le suma 1 porque no hay ningún aminoácido con índice 0.

3.1 Algoritmo Genético Clásico (AGC)

Datos de entrada: para los problemas a resolver.

P*: Matriz de probabilidades [20x20],

p: vector de probabilidades absolutas [20],

Pc: probabilidad de cruce,

Pm: probabilidades de mutación,

tam_pob: tamaño de la población (cantidad de cadenas),

tam_ind: tamaño de cada individuo (longitud de la cadena),

nro_gen: número de generaciones.

f(x): Función *fitness* a evaluar.

3.1.1 Implementación de los operadores

Operador de Selección

Este operador (selectParent) selecciona con distribución de probabilidad (probs) un número entero (índice) entre 1 y el tamaño de la población (tam_pob). El valor de la variable de entrada es la distribución de probabilidades **probs** (equivalente a la ruleta)

```
selectParent[probs_] := Module[
    {cumsums, rand, index},

    cumsums = FoldList[Plus, First[probs], Rest[probs]]; (* suma acumulada de las probabilidaes *)

    rand = Random[]; (* genera un número real en el intervalo [0, 1] *)

    index = Flatten[Position[cumsums,
                First[Select[cumsums, # >= rand &, 1]]]]; (* seleciona el índice *)
    Return[First[index]]
];(* Final del módulo *)
```

Operador de entrecruzamiento

Se utiliza el *cruzamiento en un punto*: dados dos genotipos padres, se elige al azar una posición y se realiza el entrecruzamiento entre las partes en que quedan divididas ambas cadenas.

El módulo siguiente retorna los dos hijos, de acuerdo con el *crossover* en un punto.

```
cross[parent1_, parent2_]:= Module[
   {cut, leftPart, rightPart, child},

   cut = Random[Integer, {1, strLength-1}];  (* Seleciona aleatoriamente un punto entre 1 y la
longitud de la cadena menos 1  *)

   leftPart1 = Take[parent1, cut]; (* Toma la parte izquierda del primer padre *)

   rightPart1 = Take[parent2, -(strLength - cut)];  (* Toma la parte derecha del segundo padre *)

   leftPart2 = Take[parent2, cut]; (* Toma la parte izquierda del segundo padre *)

   rightPart2 = Take[parent1, -(strLength - cut)];  (* Toma la parte derecha del primer padre *)

   child1 = Join[leftPart1, rightPart1]; (* Primer hijo *)
   child2 = Join[leftPart2,rightPart2]; (* Segundo hijo *)
   Return[{child1,child2}];
]; (* Final del módulo *)
```

Operador de mutación

Dado un individuo de la población, el operador de mutación introduce mutaciones en el cromosoma con probabilidad **p** de mutación en un punto simple. Para cada dígito binario de la cadena, si el número aleatorio generado entre cero y uno es menor que **p** entonces se cambia por su complementario, en caso contrario se mantiene.

```
Mutate[child_,p_]:=
    Function[X,If[Random[]< p,Complement[{0,1},{X}][[1]],X]]/@child;
```

El AGC se implementó para tres variantes diferentes:

0. El mejor hijo, de dos descendientes, reemplaza al peor padre.
1. El mejor hijo reemplaza el peor cromosoma de la población.
2. Si el mejor de los hijos es mejor que el mínimo de la población, reemplaza al mínimo.

3.1.2 Algoritmo

Cada población que se genera está formada por un conjunto de cadenas de aminoácidos, enumerados del 1 al 20. Esta cadena es la que se utiliza para evaluar la función objetivo. De cada cadena se tiene una réplica, pero en código binario, a las que se aplican los diferentes operadores para obtener nuevos individuos.

1. //Generar la población inicial **P (0)**.

a) generar **tam_pob** individuos de (5***tam_ind**) bits.

b) convertir **tam_pob** cadenas binarias en decimales.

c) evaluar **f** para cada cadena decimal.

2. //Construir las nuevas generaciones **P(j)**.

Hacer para **j=1** hasta **nro_gen-1**:

//Construir los individuos i que formarán la población **P(j)**

2.1 Hacer para i=0 hasta **tam_pob** – 1:

2.1.1 //Obtener un nuevo individuo:

a) aplicar el operador de selección para obtener dos padres,

b) cruzar los dos padres con probabilidad **Pc** para formar dos hijos.,

c) mutar cada bit en los hijos con probabilidad **Pm**

d) seleccionar el mejor de los hijos

2.1.2 Reemplazar el nuevo individuo en lugar del peor de la población, si es mejor que él.

2.1.3 Reemplazar el nuevo individuo en lugar del peor de la población, si es mejor que él

2.2 Actualizar los valores del mejor individuo generado y los valores mejor, peor y medio de la función en la generación **P(j)**.

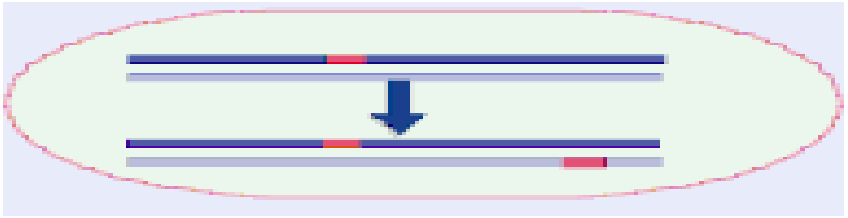
3. Imprimir los resultados y terminar.[4]

3.2 Algoritmo Genético con Transposón (AGT)

Durante el desarrollo de la investigación surgieron las siguientes interrogantes:

- ¿Cómo podemos mejorar el comportamiento del AGC?
- ¿Podemos, al incorporar un nuevo operador al AGC, mejorar su eficiencia?

Desde el punto de vista biológico un transposón es una secuencia de ADN capaz de copiarse a si misma e insertarse en una nueva posición en el genoma, sin que exista ninguna relación entre la secuencia y el lugar geométrico donde se inserta. Son secuencias discretas móviles; se transportan por si mismas a otras localizaciones, generando en el nuevo sitio copias al azar.



Los transposones pueden transportar secuencias de un genoma a otro, proporcionando la fuente principal de las mutaciones.

[4] En el Anexo 5 se presentan algunos ejemplos de corridas del AGC

3.2.1 Operador Transposón

Aunque se ha utilizado para este nuevo operador el nombre de *transposón* que desde el punto de vista biológico tiene el significado a que hacíamos referencia, el operador introducido es en realidad un operador de recombinación genética.

En biología, la *recombinación genética* es el proceso mediante el cual la información genética se redistribuye. Es gracias a ella que se ha dado la diversidad en la evolución al crear diversidad en los alelos de distintos genes, o redistribuyendo una mutación negativa que se da en un gen a otro donde puede que sea positiva. Existen varios tipos de recombinación genética, en las células eucariotas, se da la recombinación homóloga, la cual sucede durante la profase I de la meiosis y se simula en los AGC mediante el operador crossover. Otro tipo de recombinación es el que tiene lugar en el proceso de maduración de los linfocitos durante la respuesta del sistema inmune a la invasión de organismos patógenos. En esta recombinación, diferente a la homóloga, esencialmente ocurre una redistribución del material genético disponible en los cromosomas a fin de lograr un mayor Fitness en el reconocimiento de los patógenos invasores por los anticuerpos. Es un proceso de evolución molecular que tiene lugar diariamente en nuestros cuerpos y que básicamente optimiza (maximiza) la afinidad de los anticuerpos por los antígenos (moléculas presentes en los patógenos). Con el propósito de simular este tipo de recombinación se construyó el operador de transposón. [LEW04]

El procedimiento de cómputo para la puesta en práctica de este nuevo operador es:

1. seleccionar aleatoriamente un cromosoma de la población en curso.
2. seleccionar aleatoriamente un fragmento del cromosoma de longitud `strLength`/5.
3. seleccionar cuatro fragmentos consecutivos del cromosoma de longitud `strLength`/5.
4. obtener una subpoblación temporal con todas las permutaciones posibles de las partes seleccionadas
5. seleccionar de todas las permutaciones la que tiene el mejor cromosoma.
6. Sustituir al peor cromosoma de la generación en curso por el mejor cromosoma resultante de la transposición si su fitness es superior.

A continuación se presenta la implementación de este operador en el Mathematica v. 5.0

```
Transposon[genotype_] := Module[{R, NewStringArray, NewFitnessArray},

  (* Número aleatorio para realizar la partición *)

  R = Random[Integer, {1, strLength - strLength/5}];

  (* Particiones *)

  a1 = Take[genotype, {R, R + strLength/5 - 1}];
  a2 = Take[genotype, {strLength/5 + 1, 2 strLength/5}];
  a3 = Take[genotype, {2 strLength/5 + 1, 3 strLength/5}];
  a4 = Take[genotype, {3 strLength/5 + 1, 4 strLength/5}];
  a5 = Take[genotype, {1, strLength/5}];

(* Formación de una población de cromosomas a partir
    de las permutaciones de las particiones *)

  NewStringArray = Map[Function[X, Flatten[X]], Transposables];
  NewFitnessArray = Map[Function[lista, Fitness1[Phenotype[lista]]], NewStringArray];

  fitnessImmigrants = Max[NewFitnessArray];
  ClearAll[a1, a2, a3, a4, a5];

  (* La salida es un cromosoma inmigrante y su fitness *)

  {NewStringArray[[Position[NewFitnessArray, fitnessImmigrants][[1, 1]]]],
   fitnessImmigrants}];
```

3.2.2 Algoritmo

A continuación se describe el algoritmo general de búsqueda de la cadena óptima, al incorporar el operador transposón. [5]

0. //Inicializar la lista de permutaciones para el operador Transposón.
   Obtener las 120 permutaciones de una lista de 5 elementos.
1. //Generar la población inicial **P(0)**
   a) generar **tam_pob** individuos de (6***tam_ind**) bits.
   b) convertir **tam_pob** cadenas binarias en decimales.
   c) evaluar **f** para cada cadena decimal.
2. //Construir las nuevas generaciones **P(j)**.
   Hacer para j=1 hasta **nro_gen**-1:
   2.1 //Construir los individuos **i** que formarán la población **P(j)**
   Hacer para **i** = 0 hasta **tam_pob** – 1:
   2.1.1 //Obtener un nuevo individuo:
   a) aplicar el operador de selección para obtener dos padres,
   b) cruzar los dos padres con probabilidad **Pc** para formar dos hijos,
   c) mutar cada bit en los hijos con probabilidad **Pm**,
   d) evaluar la función **f** y seleccionar el mejor de los hijos.
   2.1.2 Reemplazar el nuevo individuo en lugar del peor de la población, si es mejor que él.
   2.1.3 //Aplicar el operador Transposón.
   a) seleccionar al azar un individuo de la población,
   b) seleccionar al azar un fragmento del individuo de longitud **tam_ind**/5
   c) seleccionar los primeros 4 fragmentos de longitud **tam_ind/**5,
   d) asignar los 5 fragmentos a la lista de permutaciones,
   e) evaluar la función en la subpoblación generada y seleccionar el mejor individuo.
   2.1.4 Reemplazar el nuevo individuo en lugar del peor de la población, si es mejor que él.

[5] La implementación del algoritmo se muestra en el Anexo 10 y los resultados en el Anexo 6

2.2 Actualizar los valores del mejor individuo generado y los valores mejor, peor y medio de la función en la generación **P(j)**.

3. Imprimir los resultados y terminar.

Salida: Mejor individuo generado en todo el proceso y valores mejor, peor y medio de la función en cada generación.

3.2.3 Análisis de la complejidad temporal del algoritmo.

Para el análisis de la complejidad temporal del algoritmo debemos trabajar con los dos parámetros que miden el tamaño de la instancia del problema que estamos resolviendo: **tam_pob** y **tam_ind**, así como con el parámetro que define la condición de parada del algoritmo: **nro_gen**. Para los problemas que se resuelven en esta investigación, queda establecido que **tam_pob** << **tam_ind.** Analicemos cada uno de los pasos del algoritmo:

0. El proceso de inicialización, aunque tiene que obtener las 120 permutaciones de los cinco fragmentos generados, es constante (**O(1)**), ya que no depende de ninguno de los tres parámetros.

1. La generación de la población inicial (en binario y en decimal), así como obtener el valor de la función para cada individuo, dependen del tamaño de la cadena y del número de individuos de la población, es decir **O (tam_pob * tam_ind).**

2. La construcción de las nuevas generaciones es un proceso que depende, en primer lugar, de cuántas generaciones se quieran generar (**nro_gen**), pero cada uno de sus cálculos depende de los dos parámetros que definen el tamaño del problema. Analicemos cada una de sus partes por separado.

2.1 Construir una población requiere obtener **tam_pob** individuos, pero a cada uno de ellos es necesario:

- aplicar los diferentes operadores definidos en el paso 2.1.1. La aplicación de estos operadores es independiente entre si y cada uno de ellos depende, indistintamente, del

tamaño de la cadena o del tamaño de la población. Asumiendo que el **max {tam_pob; tam_ind} = tam_ind**, observamos que el paso 2.1.1 tiene una complejidad del **O (tam_ind)**.

- compararlo con el peor de la población para reemplazarlo, en caso necesario. Esto tiene un costo constante, es decir un **O (1)**.
- aplicar el operador Transposón. Al igual que en el paso 2.1.1, aquí se desarrollan una serie de pasos independientes entre sí y que dependen, indistintamente, del tamaño de la cadena o del tamaño de la población, obteniéndose una complejidad del **O (tam_ind)**.
- nuevamente, compararlo con el peor de la población para reemplazarlo, en caso necesario, tiene un costo constante.

Obtenemos, por tanto, que construir una población tiene un costo del **O (tam_pob*tam_ind)**.

2.2 Actualizar los valores de la función en la población generada depende de su tamaño, o sea **O (tam_pob)**.

Esto nos conduce a que construir todas las generaciones depende de los tres atributos, es decir su costo es del **O (nro_gen*tam_pob*tam_ind)**.

3 Imprimir los resultados no es costoso, ya que los valores de salida se han ido actualizando para cada población generada, obteniendo un costo del **O (tam_pob).**

El paso **2** es el que define la complejidad del algoritmo en su conjunto, por lo que obtenemos un orden cúbico, dependiendo de los tres parámetros analizados, es decir, **O(nro_gen*tam_pob*tam_ind)**.

3.3 Pruebas numéricas y análisis estadísticos de los resultados.

Las pruebas numéricas se realizaron para los problemas 1 y 3 y se realizaron en dos etapas:

1. Determinar, de las tres variantes del AGC propuestas, cuál es la que proporciona las mejores soluciones para los problemas formulados.
2. Comparar la mejor variante del AGC con el AGT.

En la primera etapa, para determinar la mejor variante de las tres propuestas[6], se realizaron 360 corridas para cada problema, es decir, en total se hicieron 720 corridas del AGC. De ellas, se realizaron 40 corridas para cada variante, manteniendo fijo el tamaño de la población en 30 individuos, el número de generaciones en 2000, pero variando la longitud de las cadenas de aminoácidos (20, 50 y 100). Como se trabajó con 5 bits para la representación binaria de los aminoácidos, se generaron cadenas de 100, 250 y 500 dígitos binarios, respectivamente.

Los resultados de las pruebas numéricas realizadas en esta etapa se muestran en el Anexo 2. Cada tabla se identifica con:

- El número del Problema
- Longitud de la cadena binaria. strLength = (100, 250 ó 500)
- Tamaño de la Población. Popsize = 30;
- Probabilidad de mutación para cada hijo. Prob mut = 0.053 ó 0.033
- Variante implementada.

En las tablas siguientes se presenta un análisis estadístico de los resultados obtenidos en esta etapa. Se destacan en negritas los mejores resultados. Las variables utilizadas son:

Ngopt - número de la generación en la que se alcanza el óptimo para cada variante del algoritmo (0, 1, 2), para cadenas de longitud (20, 50, 100)

Val.fun- valor de la función *fitness* para cada variante del algoritmo (0, 1, 2), para cadenas de longitud (20, 50, 100)

Como se puede apreciar en la Tabla 3.1, el análisis descriptivo de los datos nos sugiere que la mejor variante resultó ser aquella en la que el mejor cromosoma hijo reemplaza al peor cromosoma de la población. Esta situación se visualiza claramente en la Figura 3.1. En particular observemos que medida aumenta el tamaño de las cadenas de aminoácidos consideradas, para un número fijo de generaciones, disminuyen los valores de la función. Esto nos indica que a fin de lograr mejores resultados debemos incrementar el número de generaciones, lo cual implica un incremento del costo computacional del algoritmo.

[6] Ver epígrafe 3.1

Tabla 3.1. Estadísticos descriptivos del problema 1.

	Media	Desviación standart	Valor mínimo	Valor máximo
Ngopt.0.20	1273.00	548.36	155.00	1999.00
Ngopt.1.20	1250.53	389.18	376.00	1941.00
Ngopt.2.20	1444.20	387.97	645.00	1969.00
Val.fun.0.20	**43.63**	**0.83**	**41.98**	**45.86**
Val.fun.1.20	47.29	0.76	45.60	48.37
Val.fun.2.20	46.38	0.92	44.12	48.15
Ngopt.0.50	1105.15	588.03	79.00	1964.00
Ngopt.1.50	1813.35	146.55	1304.00	1992.00
Ngopt.2.50	1750.73	225.64	1179.00	1989.00
Val.fun.0.50	**35.79**	**0.35**	**35.15**	**36.70**
Val.fun.1.50	41.79	0.85	40.21	44.05
Val.fun.2.50	40.62	0.68	39.39	42.22
Ngopt.0.100	1103.50	522.64	240.00	1976.00
Ngopt.1.100	1741.73	238.63	1102.00	1997.00
Ngopt.2.100	1767.33	183.13	1295.00	1996.00
Val.fun.0.100	**33.09**	**0.22**	**32.54**	**33.63**
Val.fun.1.100	36.69	0.39	35.79	37.99
Val.fun.2.100	35.97	0.31	35.48	36.81

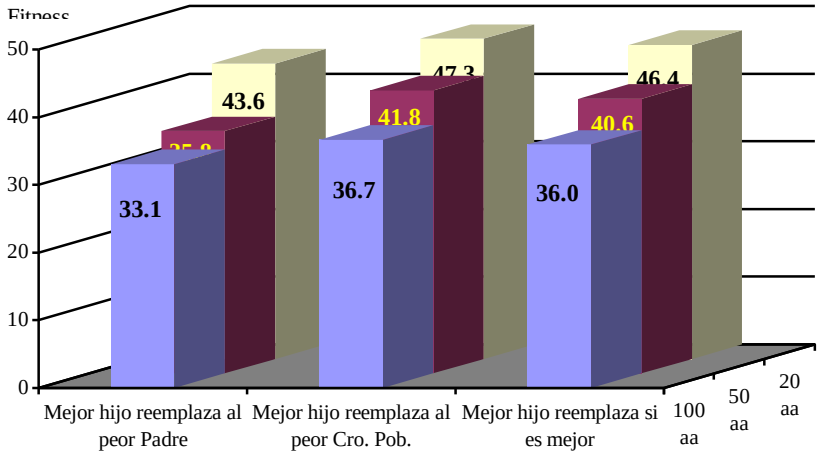


Figura 3.1. Se muestran los resultados del AGC en sus tres variantes para diferentes longitudes de cadenas de aminoácidos (aa).

Una situación análoga al problema anterior se presenta en el problema 3. En la Tabla 3.2 se muestran los estadísticos descriptivos de estos resultados y en la Figura 3.2 se visualiza la situación.

Tabla 3.2. Estadísticos descriptivos del problema 3

	Media	Desviación standart	Valor mínimo	Valor máximo
Ngopt.0.20	1106.03	549.71	171.00	1971.00
Ngopt.1.20	1042.93	539.66	156.00	1998.00
Ngopt.2.20	**1073.23**	**498.75**	**180.00**	**1996.00**
Val.fun.0.20	14.85	0.44	14.03	15.93
Val.fun.1.20	16.05	0.35	15.16	16.70
Val.fun.2.20	15.74	0.47	14.67	16.53
Ngopt.0.50	1042.20	527.34	189.00	1920.00
Ngopt.1.50	1795.50	179.89	1334.00	1999.00
Ngopt.2.50	**1744.13**	**266.93**	**1091.00**	**1987.00**
Val.fun.0.50	10.63	0.26	10.30	11.69
Val.fun.1.50	14.37	0.39	13.41	15.12
Val.fun.2.50	13.74	0.36	12.94	14.61
Ngopt.0.100	1286.40	539.23	251.00	1995.00
Ngopt.1.100	1729.33	205.07	1263.00	1998.00
Ngopt.2.100	**1780.55**	**177.11**	**1391.00**	**1993.00**
Val.fun.0.100	9.12	0.82	8.44	11.12
Val.fun.1.100	11.37	0.24	11.00	12.14
Val.fun.2.100	10.85	0.27	10.33	11.46

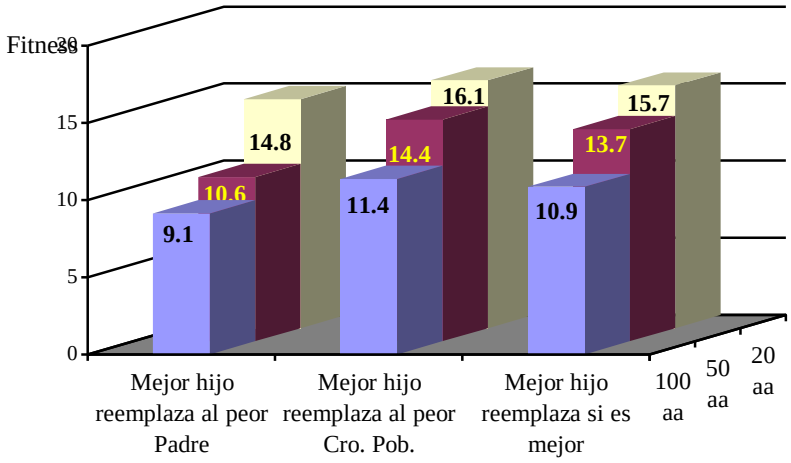


Figura 3.2. Resultados de la primera etapa para el Problema 3.

En la segunda etapa se compara esta mejor variante del AGC con el AGT, para probar la eficiencia del nuevo operador propuesto en la obtención de las soluciones de los problemas formulados. En esta etapa se trabajó con cadenas de aminoácidos de longitud 50, se fijó el tamaño de la población en 40 individuos y se varió solamente el número de generaciones,

realizándose corridas para 100, 1000 y 2000 generaciones. Se realizaron 40 corridas para cada algoritmo y para cada número de generaciones prefijado. Al igual que en la etapa anterior los resultados fueron recogidos en tablas[7] que, a diferencia de las tablas de la etapa anterior, también se recoge el tiempo de corrida.

A partir de estos resultados se realizaron con el SPSS v. 13.0 distintas analisis descriptivos de las variables, cuyos resultados se resumen en las tablas siguientes.

Tabla 3.3 Resultados estadísticos de la comparación de la mejor variante del AGC y el AGT para el Problema 1.

		Media	Desviación standart	Valor mínimo	Valor máximo
Para 100 generaciones	Ngopt.AGC.	88.10	9.48	61.00	99.00
	Ngopt.AGT.	91.90	11.68	34.00	96.00
	Ti.corr.AGC.	**19.15**	**0.64**	**18.67**	**21.45**
	Ti.corr.AGT.	249.81	10.49	233.33	286.17
	Val.fun.AGC.	37.07	0.73	35.74	39.33
	Val.fun.AGT.	47.49	1.25	44.68	49.08
Para 1000 generaciones	Ngopt. AGC.	914.38	101.99	486.00	998.00
	Ngopt. AGT.	156.28	69.94	76.00	413.00
	Ti.corr. AGC.	**193.30**	**6.85**	**188.55**	**223.64**
	Ti.corr. AGT.	2779.85	74.54	1689.84	3159.69
	Val.fun. AGC.	41.01	0.77	39.45	42.92
	Val.fun. AGT.	49.01	0.05	48.87	49.08

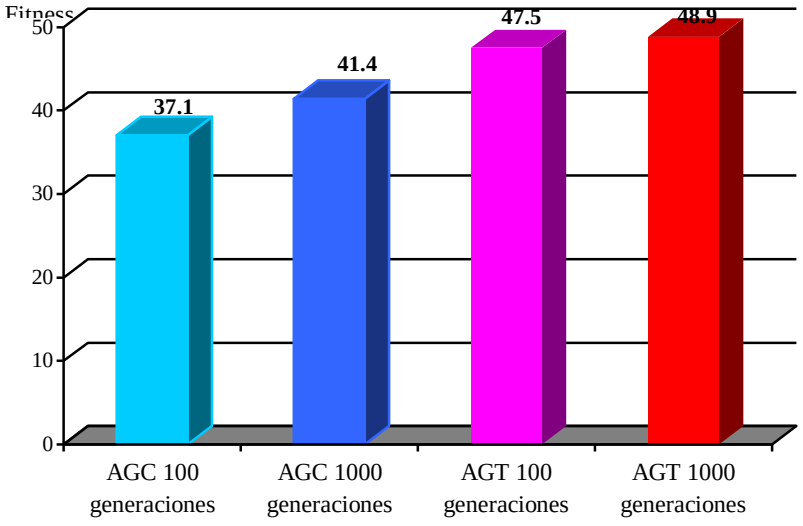


Figura 3.3. Comparación de la mejor variante del AGC con el AGT para el Problema 1.

[7] Ver Anexos 3 y 4

La Figura 3.3 muestra la comparación de la mejor variante del AGC con el AGT para el Problema 1.Observe que las diferencias entre los valores de la media de la función son bien marcadas

La tabla 3.4 muestra los resultados de la comparación de la mejor variante del AGC con el AGT para el Problema 3.

Tabla 3.4. Resultados estadísticos de la comparación de la mejor variante del AGC y el AGT para el Problema 3

		Media	Desviación standart	Valor mínimo	Valor máximo
Para 100 generaciones	Ngopt.AGC.	89.13	7.92	61.00	99.00
	Ngopt.AGT.	93.73	8.89	49.00	99.00
	Ti.corr.AGC.	22.28	0.50	21.39	23.34
	Ti.corr.AGT.	304.52	13.19	281.17	338.06
	Val.fun.AGC.	11.30	0.39	10.68	12.18
	Val.fun.AGT.	**14.72**	**1.19**	**10.99**	**16.25**
Para 1000 generaciones	Ngopt. AGC.	871.15	103.60	634.00	996.00
	Ngopt. AGT.	590.48	340.85	73.00	974.00
	Ti.corr. AGC.	215.39	6.26	209.58	242.83
	Ti.corr. AGT.	3276.74	206.77	3035.61	3557.59
	Val.fun. AGC.	14.12	0.35	13.32	14.93
	Val.fun. AGT.	**15.15**	**1.24**	**12.75**	**16.71**

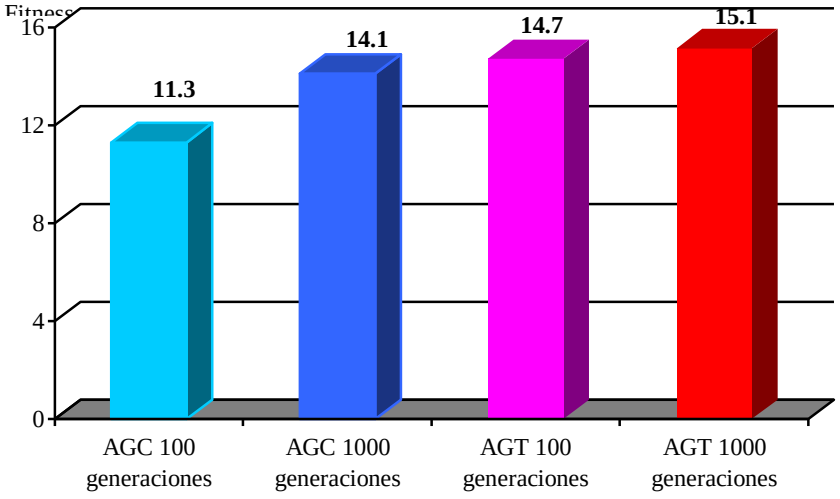


Figura 3.4. Comparación de la mejor variante del AGC con el AGT para el Problema 3.

A partir del análisis descriptivo de los resultados pueden destacarse los aspectos siguientes:

- A medida que aumenta el número de generaciones aumenta también la media de los valores de la función.
- Los valores de la función son más altos en el AGT, tanto para 100 como para mil generaciones.
- Cuando se aplica el AGT, a partir de 100 generaciones, se obtienen resultados significativamente superiores a los del AGC con 1000 generaciones.
- En el AGT es menor la diferencia entre los valores máximos y mínimos de la función.
- El AGT mejora considerablemente la eficiencia del AG.

Las cadenas de 50 *aa* de máxima complejidad obtenidas son las siguientes:

Problema 1.

Solución	Valor de la función
GPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGP	48.9873
GPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGA	49.0803
AGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPG	49.0803
GPGPGPGAGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGPGA	48.8654

Problema 3

Solución	Valor de la función
GGGGGGGGGGHAAHAAGGGGGGGGGGHAAHAAGHAAHAAGGGGGGGGGGGGGHAAHAA	16.2538
GGGGGGGGGGGGGGGGGGGPGGGGGPAAYAAYAAYAAYAAYAAAGGGGGGGGGGGGG	16.5146
GGGGGGGGGGGGGGGGGGGPGGGPAAYAAYAAYAAYAAYAAGGGGGGGGGGGGGGG	16.7101

3.3.1 Análisis estadísticos realizados

Las hipótesis derivadas del análisis descriptivo fueron verificadas estadísticamente. Con el objetivo de verificar si las muestras de valores de la funciones fitness provenían de una población con distribución normal, se aplicaron la prueba de normalidad de Kolmogorov Smirnov con la correción de Lilliefors y la prueba de Shapiro-Wilk. Debido a que se rechazó la normalidad de las variables analizadas en las pruebas anteriores, para la verificación de las diferencias entre muestras originadas por las variantes de AGs se aplicarón las pruebas no paramétricas de Kruskall-Wallis y de Mann-Whitney. En la tesis se recogen los resultados de los test para el problema 3 solamente ya que los resultados son similares a los obtenidos para el problema 1.

Las variantes que se compararon en el análisis fueron codificadas como:

Grupo 0: Variante cero del AGC: El mejor hijo, de dos descendientes, reemplaza al peor padre

Grupo 1: Variante uno: El mejor hijo reemplaza el peor cromosoma de la población

Grupo 2: Variante dos: Si el mejor de los hijos es mejor que el mínimo de la población, reemplaza al mínimo.

Estos grupo se compararon en cuanto a:

$\mathbf{X_{1j}}$: Número de la generación en que se alcanza el óptimo para cadenas de 20 aminoácidos

X_{2j}: Valor de la función para cadenas de 20 aminoácidos

$\mathbf{X_{3j}}$: Número de la generación en que se alcanza el óptimo para cadenas de 50 aminoácidos

X_{4j}: Valor de la función para cadenas de 50 aminoácidos

$\mathbf{X_{5j}}$: Número de la generación en que se alcanza el óptimo para cadenas de 100 aminoácidos

X_{6j}: Valor de la función para cadenas de 100 aminoácidos

donde el índice j recorre los grupos 0, 1 y 2.

Grupos (0 y 2)

- El test Kruskal-Wallis arrojó que existen diferencias altamente significativas entre los grupos anteriores.

A continuación para cada *i* (de la variable X_{ij}, i = 1,..6 y j = 0, 1, 2) se realizaron pruebas de Mann-Whitney par a par entre los *j* grupos considerados.

Comparación de los pares de grupos 0-2 y 0-1

- Se aprecia que al comparar los resultados de los grupos 0 y 2 y los grupos 0 y 1, que las diferencias son altamente significativas para todas las variables, excepto para el número de la generación para la cual se alcanza el óptimo para cadenas de longitud 20 (variables X_{1j}).

Comparación de los grupos 1 y 2

- Las diferencias son altamente significativas solo para los valores de la función (variables X_{2j}, X_{4j} y X_{6j}).

Los análisis estadísticos confirmaron que para los problemas analizados la mejor variante corresponde al grupo 1, o sea, la variante donde el mejor hijo reemplaza el peor cromosoma de la población. Esta variante fue seleccionada para ser comparada con el AGT.

Comparación de la mejor variante del AGC con el AGT

Los resultados de la mejor variante del AGC se compararon con el AGT utilizando la prueba Mann-Whitney. El análisis estadístico reveló diferencias altamente significativas para todas las variables consideradas. En otras palabras, el AGT mejoró considerablemente los valores de las funciones EPCP y satisface las expectativas de esta investigación. Este algoritmo nos permite contar con una herramienta para evaluar el potencial de las funciones analizadas en cuanto a su capacidad de estimar la complejidad. Por ejemplo, la solución dada para el problema 1 nos revela que la secuencia que maximiza la función EPCP correspondiente es: (1, 6, 13, 6, 13, 6, 13,…) la cual corresponde a la secuencia de aminoácidos AGPGPGPGP…. Desde el punto de vista de la definición de complejidad de Kolmogorov esta función tiene una complejidad muy baja, sin embargo desde el punto de vista físicoquímico su complejidad es muy alta, pues se debe requerir mucha energía para generar tan solo una secuencias de 20 aminoácidos. La razón estriba en que cada aminoácido Prolina (P) al ser introducido como nuevo eslabón en la cadena provoca un giro con fuertes impedimentos estéricos haciendo más difícil la introdución del siguiente aminoácido. Es decir, si la complejidad de la secuencia se mide sobre la base de la definición de Kolmogorov entonces la función EPCP del problema 1 no es un buen estimador, mientras que si medimos la complejidad sobre la base de los

requerimientos energéticos para sintetizar la secuencia, entonces la función mencionada es un estimador potencial de la misma.

3.4 Conclusiones del capítulo.

- La mejor variante del AGC implementada fue aquella en la que el mejor hijo reemplaza el peor cromosoma de la población.
- El nuevo operador introducido para el Algoritmo Genetico en las pruebas numéricas realizadas probó su eficacia al aportar el Algoritmo Genetico con Transposon mejores soluciones e incluso sin necesidad de elevar considerablemente el número de generaciones.
- El Algoritmo Genetico con Transposon constituye una herramienta que nos permite evaluar la capacidad en la estimación de la complejidad de las secuencias de aminoácidos que poseen las funciones estimadoras potenciales de la complejidad.

Conclusiones

1. Se elaboraron diferentes modelos de optimización para los problemas planteados: modelo combinatorio y, modelo de programacion no lineal.La implementación del Algoritmo Genético se realizó a partir del modelo combinatorio.
2. Se introdujo un nuevo operador para los AG que mejora considerablemente la eficiencia del algoritmo.Las soluciones obtenidas con el AGT son significativamente superiores a las obtenidas con el AGC en cualesquiera de sus variantes.
3. Se valoró la complejidad de los algoritmos de solución de los problemas. La complejidad AGT resultó ser de orden cúbico.
4. Se elaboró un algoritmo que permite determinar la cadena de aminoácidos de máxima complejidad para diferentes funciones estimadoras potenciales de la misma.

Recomendaciones

1. Validar los resultados obtenidos utilizando otras heurísticas modernas como Recocido Simulado, Métodos de colonias de hormigas u otras….
2. En el capítulo 1 se analiza el efecto de los distintos operadores sobre los esquemas por eso proponemos analizar el efecto del nuevo operador propuesto en el teorema del esquema.
3. Utilizar el AGT en la búsqueda de otras funciones EPCP.

Referencias bibliográficas

[ADA00]-Adami, C. y Cerf, N.J. Physical complexity of symbolic sequences. Physica D, 137, 62–69. 2000.

[BELMOR03]-Belén Melián, José A. Moreno Pérez, J.Marcos Moreno Vega Metaheuristics: A global view. Departamento de Estadística, I.O. y Computación. Centro Superior de Informática. Universidad de La Laguna.2003

[CAM94]- Campollo RO. Modelos matemáticos en medicina y biología. Bases teóricas y fundamentos. Rev Invest Clin 1994; 46(4): 307-3

[CHA98]-Chaitin, G. J. The limits of mathematics. A Course on Information Theory and the Limits of Formal Reasoning. Springer-Verlag. New York, 1998.

[CRA68]- Crack , F. H .C . (1968). The origin of the genetic code. J . Mol. Biol. 38, 367-379.

[EPS66]-Epstein, C. J. (1966). Role of the Amino-acid "code" and selection for conformation in the evolution of proteins. Nature 210, 25-28.

[GCI04]- Julio Brito Santana, Clara Campos Rodriguez, Félix C. Garcia López, Miguel Garcia Torres, Belén Melían Batista, Jose A. Moreno Perez, J. Marcos Moreno Vega. Grupo de Computación Inteligente. Universidad de La Laguna. Metaheurísticas una revisión actualizada gci@ull.es http://webpages.ull.es/users/gci/ Documento de Trabajo n 02/2004

[GER06]- Gerhard Post. The class NP. March 22, 2006

[GOL89]- David Goldberg , Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley , 1989.

[GRA74]- Grantham, R. (1974). Amino acid difference formula to help Explain Protein Evolution. Science 185. 862-864.

[HILLIE97]-Frederick S. Hillier y Gerald J. Lieberman, "Introducción a la Investigación de Operaciones". Sexta edición. McGraw-Hill. 1997.

[HIN97] - Hing, C. Rosina, "Conferencias sobre la modelación matemática"(primera y segunda parte). Universidad EAFIT, Colombia, 1997.

[HOL75]- Holland John H. Adaptation in natural and Artificial Systems, Ann Arbor. The University of Michigan.1975.

[KURGAL98]-Algoritmos Genéticos. Ángel Kuri Morales, José Galaviz Casas. Instituto Politécnico Nacional. Centro de Investigación en Computación.

[LEW04]- Benjamín Lewin. Genes VIII. Published by Pearson Prentice Hall. Pearson Education .2004.

[MAI95]- Mainegra Hing Marisela. Tesis en opción del titulo Licenciado en Ciencias de la Computación. UCLV 1995.

[MIT96]- Melani Mitchel, An Introducction to Genetic Algorithms. MIT Press 1996.

[MUR01]- J.D.Murray.Interdisciplinary applied mathematics.Mathematical Biology.

I.An Introduction.Springer.Third Edition .2001

II.Spatial Models and Biomedical Aplications

[PAPSTE98]-Christos H. Papadimitriou, Kenneth Steiglitz Combinatorial Optimization: Algorithms y Complexity.1998.

[RIB85]- Ribnikov, K.; "Análisis Combinatorio." Editorial Mir. 1985.

[SÁN03]- Sánchez Rodríguez Robersy. Estudio del orden en el Código Genético mediante la aplicación de métodos estadísticos y algebraicos. Tesis de Maestría. Facultad Matemática-Física y Computación. UCLV. Santa Clara. Cuba. (2003)

[SANGRA04]- R. Sánchez, Ricardo Grau. A genetic code Boolean structure. II. The genetic information system as a Boolean information system. Bulletin of Mathematical Biology. Artículo en prensa. doi:10.1016/j.bulm.2004.12.004, pdf de la copia corregida disponible en: http://dx.doi.org/10.1016/j.bulm.2004.12.004

[SÁN05]- Sánchez R., Grau R. A genetic code Boolean structure. II. The genetic information system as a Boolean information system. Bull. Math. Biol. 67, 1017-1029, 2005.

[SÁNGRA05]- Sánchez R., Morgado E., Grau R. A genetic code boolean structure I. The meaning of boolean deductions, Bull. .Math. Biol., 67, 1–14, 2005.

[SANMORGRA05]-R Sánchez, E. Morgado, R. Grau. A genetic code Boolean structure. I. The meaning of Boolean deductions. Bulletin of Mathematical Biology 67 (2005) 1–14.

[SEGVER04]- Clara Segura, Alberto Verdejo. Análisis de la complejidad de algoritmos y problemas. Dpto. de Sistemas Informáticos y Programación. Universidad Complutense de Madrid.2004.

[TIJKOS87]- Tijonov, A. y Kostomarov, D., "Conferencias de introducción a las matemáticas aplicadas". Editorial MIR, Moscú 1987.

[VOL85]-Volkenshtein, M.V. Biofísica. Editorial MIR, Moscú, 1985.

[YOC02]-Yockey, H.P. Fundamentals of life. Information theory, evolution and the origin of life. Éditions scientifiques et médicales Elsevier SAS. Chapter II.10, 2002.

[YOC05]-Yockey H. Information Theory, evolution and origin of life. Cambridge University Press, 2005

Referencias Web

[1]-http://tierra.ciens.ucv.ve/postfismed/pfmpub/t98v1pp/t98v1pp5.html

[2]-http://www.bioplanet.net/magazine/2003.htm

[3]-http://www.eticayempresa.com/congreso3/27_1_4.

[4]-http://es.wikipedia.org/wiki/Teor%C3%ADa_de_la_complejidad_algoritmica

[5]-http://w3.mor.itesm.mx/~jfrausto/Papers/Sanvice/Enc99/Red_Cen_art2.doc

[6]-http://the-geek.org/docs/algen

[7]-http://www14.uniovi.es/ia/Genetico-TSP/Introduccion.htm

[8]-http://es.wikipedia.org/wiki/Complejidad_computacional

[9]-http://euler.us.es/~renato/adn/

[10]-http://www.moebio.uchile.cl/10/pena.htm Complejidad de

[11]-**http://www.metaheuristics.org/index.php?main=3&sub=34**

[12]-http://en.wikipedia.org/wiki/Genetic_algorithm

[13]-http://cs.felk.cvut.cz/~xobitko/ga/ Introduction to Genetic Algorithms with Java

[14]-http://www.aaai.org/aitopics/html/genalg.html

[15]-http: // www.biopsicología.net/

[16]-http://webpages.ull.es/users/gci/o

[17]-http://tigre.aragon.unam.mx/geneticos/titulo.html .

[18]-http://es.wikipedia.org/wiki/Optimizaci%C3%B3n_combinatoria

[19]-http://www.fdi.ucm.es/profesor/rdelvado/Docencia/Docencia2005-2006/Los%20Or%C3%ADgenes%20de%20La%20Inform%C3%A1tica%20en%20la%20Historia%20de%20las%20Matem%C3%A1ticas/complejidad.pdf

[20]-http://www.biologia-en-internet.com/default.asp?Id=4&Fb=2

[21]- http://www.cs.princeton.edu/ introcs/75turing/

[22]- http://physics.kenyon.edu/coolphys/thrmcmp/newcomp.htm

[23]- Glossary of Genetics Terms ; http://www.bis.med.jhmi.edu/Dan/DOE/prim6.html

Anexos

Anexo I. Datos de entrada del algoritmo

Tabla A1.1. Matriz de probabilidades condicionales.

	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
A	0.115	0.020	0.040	0.046	0.029	0.072	0.023	0.043	0.029	0.072	0.015	0.029	0.072	0.020	0.052	0.107	0.087	0.087	0.013	0.029
C	0.042	0.071	0.036	0.030	0.047	0.114	0.035	0.032	0.018	0.092	0.010	0.024	0.046	0.023	0.112	0.094	0.030	0.066	0.034	0.047
D	0.081	0.035	0.070	0.058	0.023	0.111	0.035	0.031	0.035	0.047	0.009	0.047	0.047	0.023	0.070	0.093	0.058	0.064	0.016	0.047
E	0.083	0.026	0.052	0.088	0.015	0.129	0.021	0.027	0.052	0.052	0.014	0.031	0.036	0.031	0.088	0.077	0.052	0.072	0.024	0.031
F	0.059	0.046	0.023	0.018	0.069	0.065	0.028	0.047	0.012	0.137	0.012	0.018	0.080	0.016	0.074	0.115	0.047	0.082	0.019	0.034
G	0.055	0.042	0.042	0.055	0.024	0.182	0.015	0.037	0.031	0.064	0.018	0.024	0.022	0.018	0.112	0.074	0.033	0.099	0.031	0.024
H	0.042	0.030	0.031	0.021	0.025	0.036	0.087	0.034	0.031	0.085	0.007	0.052	0.123	0.046	0.102	0.092	0.073	0.026	0.010	0.046
I	0.057	0.021	0.021	0.020	0.031	0.065	0.025	0.089	0.026	0.115	0.025	0.031	0.077	0.019	0.097	0.086	0.089	0.081	0.010	0.015
K	0.058	0.017	0.035	0.058	0.012	0.081	0.035	0.040	0.070	0.058	0.019	0.047	0.064	0.047	0.116	0.076	0.081	0.047	0.016	0.023
L	0.046	0.029	0.015	0.019	0.043	0.054	0.030	0.056	0.019	0.164	0.019	0.019	0.102	0.027	0.100	0.094	0.059	0.067	0.020	0.021
M	0.057	0.018	0.018	0.031	0.022	0.088	0.015	0.070	0.035	0.110	0.044	0.022	0.055	0.022	0.119	0.073	0.075	0.097	0.020	0.011
N	0.057	0.023	0.046	0.034	0.017	0.063	0.057	0.046	0.046	0.057	0.011	0.069	0.080	0.034	0.092	0.092	0.092	0.040	0.009	0.034
P	0.055	0.017	0.018	0.016	0.030	0.022	0.052	0.044	0.024	0.122	0.011	0.031	0.178	0.037	0.069	0.107	0.100	0.033	0.008	0.026
Q	0.044	0.024	0.025	0.038	0.018	0.050	0.055	0.031	0.050	0.090	0.013	0.038	0.104	0.068	0.123	0.079	0.069	0.031	0.020	0.030
R	0.033	0.035	0.022	0.032	0.023	0.095	0.036	0.047	0.037	0.100	0.020	0.030	0.058	0.036	0.166	0.080	0.046	0.054	0.026	0.023
S	0.070	0.030	0.030	0.028	0.037	0.065	0.033	0.043	0.025	0.097	0.013	0.030	0.092	0.024	0.082	0.128	0.066	0.059	0.015	0.033
T	0.079	0.013	0.026	0.026	0.021	0.039	0.037	0.061	0.037	0.084	0.018	0.042	0.118	0.029	0.066	0.092	0.131	0.053	0.007	0.021
V	0.080	0.029	0.029	0.037	0.037	0.120	0.013	0.056	0.021	0.096	0.023	0.019	0.040	0.013	0.077	0.082	0.053	0.133	0.021	0.019
W	0.042	0.053	0.027	0.044	0.030	0.135	0.019	0.025	0.027	0.103	0.017	0.015	0.032	0.030	0.133	0.076	0.025	0.076	0.059	0.030
Y	0.063	0.049	0.050	0.038	0.036	0.069	0.055	0.025	0.025	0.072	0.006	0.038	0.073	0.030	0.079	0.110	0.050	0.044	0.020	0.068

Vector de probabilidades absolutas

A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
0.062	0.030	0.031	0.034	0.030	0.081	0.034	0.046	0.031	0.096	0.016	0.031	0.080	0.028	0.096	0.093	0.068	0.067	0.019	0.028

Vectores de probabilidades absolutas y condicionales de la ruleta

```
AbsoluteSum = Drop[FoldList[Plus, 0, ProbabilidadAbsoluta], 1]

ConditionalSum = Map[Function[lista, Drop[FoldList[Plus, 0, lista], 1]], ProbabilidadCondicional];
```

Índice de los aminoácidos y sus masas en Kg/mol

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
0.089	0.121	0.133	0.147	0.165	0.075	0.155	0.131	0.146	0.131	0.149	0.132	0.115	0.146	0.174	0.105	0.119	0.117	0.204	0.181

Anexo 2. Pruebas numéricas para el AGC. Problema 1.

Parámetros generales

strLength = 250; (* Longitud de la cadena *)

Popsize = 30; (* Tamaño de la Población *)

Probabilidad de mutación: 0.053 y 0.033

Tabla A2.1. El mejor hijo, de dos descendientes, reemplaza al peor padre. Resultados de 40 corridas del algoritmo.

Corr [a]	Gen [b]	Cadena óptima	Val f [c]
1	275	{12, 6, 13, 2, 9, 1, 6, 17, 2, 10, 17, 12, 1, 13, 1, 17, 1, 6, 1, 6, 12, 4, 13, 6, 17, 6, 1, 9, 6, 13, 6, 2, 1, 9, 1, 12, 6, 1, 7, 10, 16, 9, 17, 2, 1, 12, 6, 13, 6, 2}	35.4245
2	425	{6, 7, 2, 13, 3, 6, 13, 6, 13, 6, 6, 3, 5, 13, 6, 7, 6, 2, 13, 6, 13, 6, 13, 1, 2, 12, 11, 6, 10, 6, 3, 18, 2, 5, 12, 16, 9, 1, 6, 18, 2, 4, 2, 12, 6, 11, 1, 13, 6, 2}	35.3466
3	952	{11, 6, 17, 2, 1, 9, 1, 6, 13, 2, 1, 3, 10, 6, 17, 6, 11, 1, 2, 1, 6, 6, 13, 1, 11, 1, 8, 6, 11, 2, 9, 14, 17, 6, 13, 2, 11, 9, 2, 11, 9, 6, 3, 2, 11, 13, 6, 13, 6, 13},	36.1858
4	1576	{6, 13, 1, 2, 11, 1, 2, 11, 6, 13, 6, 13, 6, 2, 2, 7, 8, 1, 1, 12, 1, 4, 13, 9, 8, 6, 7, 1, 5, 11, 6, 17, 6, 13, 12, 1, 2, 6, 13, 6, 13, 18, 6, 5, 2, 3, 1, 2, 1, 6}	35.7392
5	1032	{2, 1, 6, 12, 6, 1, 2, 14, 6, 17, 17, 6, 11, 16, 11, 3, 6, 7, 1, 11, 1, 7, 1, 12, 1, 6, 16, 1, 9, 6, 13, 6, 13, 2, 6, 3, 2, 12, 2, 11, 6, 13, 6, 13, 6, 17, 1, 16, 6, 6}	36.5898
6	1913	{1, 1, 8, 13, 6, 10, 6, 13, 4, 9, 1, 14, 1, 6, 13, 6, 13, 8, 7, 6, 1, 8, 3, 6, 3, 2, 13, 1, 9, 10, 11, 6, 6, 13, 6, 11, 6, 11, 1, 2, 1, 11, 13, 6, 13, 6, 17, 6, 17, 6}	36.2069
7	683	{2, 9, 2, 17, 1, 2, 1, 2, 17, 6, 1, 6, 1, 6, 12, 6, 14, 8, 1, 7, 12, 6, 5, 8, 2, 8, 1, 11, 2, 1, 6, 13, 20, 12, 2, 6, 1, 2, 17, 11, 2, 9, 2, 11, 2, 3, 12, 6, 13, 6}	35.4569
8	839	{13, 6, 13, 1, 8, 6, 11, 12, 6, 12, 11, 2, 1, 12, 1, 2, 8, 8, 1, 5, 17, 2, 1, 1, 6, 13, 2, 6, 13, 1, 13, 1, 4, 11, 6, 2, 1, 6, 13, 2, 13, 2, 16, 8, 6, 17, 6, 13, 2, 6}	35.8157
9	180	{2, 6, 1, 2, 6, 2, 9, 10, 1, 6, 13, 2, 9, 6, 1, 6, 11, 2, 9, 6, 7, 6, 2, 8, 2, 6, 16, 6, 12, 6, 9, 6, 13, 2, 17, 18, 7, 14, 6, 12, 6, 13, 10, 3, 11, 6, 17, 6, 17, 6}	35.7813
10	1939	{1, 1, 6, 13, 18, 2, 6, 7, 3, 6, 13, 1, 2, 11, 6, 13, 4, 12, 12, 18, 13, 6, 6, 1, 2, 7, 6, 9, 2, 3, 5, 9, 1, 10, 2, 3, 6, 9, 2, 6, 7, 6, 7, 6, 9, 6, 13, 6, 13, 6}	35.4073
11	582	{6, 11, 2, 11, 6, 6, 7, 6, 9, 12, 9, 6, 1, 11, 2, 1, 2, 4, 2, 1, 13, 6, 13, 6, 13, 6, 10, 9, 2, 1, 6, 10, 9, 6, 9, 1, 9, 12, 1, 6, 13, 2, 11, 2, 12, 11, 12, 6, 1, 8}	35.6831

Tabla A2.1. (Continuación)

Corr [a]	Gen [b]	Cadena óptima	Val f [c]
12	1597	{1, 2, 1, 7, 6, 10, 9, 8, 1, 11, 2, 17, 6, 13, 11, 13, 1, 2, 5, 1, 6, 9, 6, 14, 6, 14, 6, 6, 17, 6, 8, 1, 6, 17, 6, 10, 6, 13, 2, 6, 1, 6, 2, 4, 12, 2, 12, 2, 6, 1}	35.699
13	326	{10, 3, 13, 6, 11, 6, 14, 6, 10, 3, 6, 6, 17, 6, 13, 11, 12, 1, 14, 6, 1, 2, 6, 11, 3, 6, 1, 6, 12, 17, 3, 13, 6, 13, 6, 17, 6, 17, 6, 3, 11, 1, 2, 7, 6, 1, 2, 6, 1, 10}	36.4373
14	754	{6, 13, 6, 1, 6, 16, 6, 13, 6, 14, 1, 20, 11, 6, 13, 2, 1, 6, 16, 7, 8, 6, 1, 10, 12, 11, 3, 6, 1, 8, 1, 16, 6, 1, 7, 2, 8, 3, 10, 6, 1, 6, 9, 2, 3, 16, 6, 17, 6, 17}	35.8663
15	951	{9, 2, 13, 8, 8, 6, 1, 6, 12, 2, 12, 6, 1, 3, 6, 7, 18, 12, 2, 1, 11, 6, 11, 6, 16, 11, 1, 4, 2, 17, 6, 12, 11, 6, 13, 3, 1, 2, 11, 1, 11, 1, 6, 6, 13, 6, 1, 2, 10, 6}	35.4252
16	544	{1, 6, 7, 2, 1, 2, 17, 6, 13, 5, 17, 2, 6, 5, 17, 14, 6, 1, 7, 11, 6, 1, 1, 2, 14, 4, 11, 2, 11, 17, 2, 7, 2, 1, 2, 17, 2, 11, 1, 6, 11, 1, 2, 1, 2, 6, 7, 6, 13, 6}	35.6538
17	162	**{7, 18, 6, 13, 6, 7, 6, 13, 1, 8, 1, 2, 1, 7, 6, 13, 6, 1, 11, 1, 2, 1, 17, 5, 6, 13, 6, 13, 14, 6, 3, 13, 1, 6, 1, 3, 1, 2, 10, 5, 11, 6, 17, 6, 1, 6, 13, 6, 16, 11}**	**36.6963**
18	1795	{1, 17, 18, 2, 1, 6, 6, 1, 5, 11, 13, 2, 3, 3, 14, 6, 13, 6, 1, 7, 6, 2, 1, 6, 13, 11, 6, 1, 17, 6, 10, 2, 2, 6, 13, 6, 9, 6, 11, 6, 13, 6, 1, 9, 10, 11, 6, 1, 10, 6}	35.2709
19	1330	{1, 2, 11, 2, 13, 6, 13, 6, 14, 4, 8, 1, 12, 6, 1, 1, 8, 6, 14, 6, 8, 6, 11, 2, 1, 6, 8, 4, 1, 6, 1, 18, 7, 2, 1, 10, 1, 11, 1, 2, 1, 6, 7, 2, 2, 8, 2, 6, 11, 6}	35.5828
20	1129	{6, 1, 1, 11, 13, 6, 13, 10, 1, 6, 1, 11, 1, 12, 1, 6, 8, 10, 6, 13, 6, 1, 6, 11, 6, 13, 6, 6, 1, 13, 6, 5, 6, 1, 9, 3, 4, 8, 14, 3, 18, 12, 6, 1, 9, 6, 1, 6, 13, 6}	36.2382
21	1814	{17, 1, 6, 13, 6, 13, 6, 11, 18, 9, 9, 18, 13, 1, 2, 12, 1, 6, 5, 14, 2, 11, 20, 6, 8, 3, 1, 6, 1, 2, 11, 16, 3, 1, 6, 1, 6, 13, 2, 13, 6, 13, 6, 16, 14, 11, 2, 17, 6, 13}	36.0776
22	1092	{4, 13, 6, 1, 2, 6, 13, 2, 1, 17, 6, 17, 5, 6, 12, 13, 6, 6, 17, 2, 1, 2, 8, 6, 8, 9, 7, 2, 3, 13, 3, 16, 1, 16, 11, 6, 4, 13, 6, 1, 2, 11, 6, 13, 6, 17, 6, 17, 6, 8}	35.9047
23	1471	{7, 2, 11, 6, 11, 2, 17, 6, 1, 6, 3, 6, 6, 5, 1, 6, 2, 9, 7, 6, 6, 1, 6, 12, 19, 1, 2, 13, 6, 14, 6, 13, 6, 13, 2, 2, 1, 11, 2, 9, 17, 1, 6, 1, 7, 6, 3, 1, 2, 1}	35.6219
24	1823	{6, 13, 6, 1, 9, 18, 6, 12, 6, 13, 3, 5, 9, 13, 6, 5, 6, 1, 17, 6, 13, 6, 13, 12, 6, 2, 3, 1, 13, 4, 8, 9, 11, 2, 1, 6, 14, 6, 12, 1, 8, 18, 7, 6, 11, 6, 6, 1, 6, 6}	35.6014
25	944	{6, 9, 6, 14, 6, 1, 12, 7, 17, 2, 12, 6, 1, 9, 1, 6, 11, 13, 6, 1, 2, 6, 7, 11, 4, 1, 2, 8, 13, 2, 1, 6, 13, 2, 1, 6, 13, 12, 11, 6, 10, 12, 6, 9, 2, 11, 6, 11, 6, 13}	35.4737
26	664	{13, 2, 6, 12, 6, 13, 6, 1, 5, 12, 2, 2, 1, 2, 1, 12, 6, 8, 9, 13, 6, 13, 6, 8, 6, 13, 2, 4, 17, 6, 2, 2, 17, 2, 8, 6, 13, 6, 14, 3, 11, 11, 16, 2, 17, 2, 1, 6, 2, 11}	36.1186

Tabla A2.1. (Continuación)

Corr [a]	Gen [b]	Cadena óptima	Val f [c]
27	1011	{1, 6, 6, 17, 4, 11, 7, 1, 8, 6, 13, 2, 8, 2, 1, 6, 13, 6, 17, 3, 1, 12, 6, 6, 1, 6, 1, 11, 2, 13, 1, 2, 6, 7, 1, 12, 6, 14, 6, 8, 14, 1, 3, 1, 18, 6, 17, 2, 2, 13}	35.5807
28	542	{9, 2, 1, 6, 7, 11, 8, 6, 14, 9, 2, 17, 2, 9, 6, 17, 2, 17, 6, 8, 2, 12, 2, 11, 6, 1, 14, 1, 12, 1, 6, 1, 6, 8, 7, 6, 9, 1, 2, 14, 6, 2, 1, 2, 13, 3, 16, 6, 7, 6}	35.6352
29	1438	{6, 13, 6, 13, 5, 4, 6, 17, 6, 2, 13, 6, 7, 6, 13, 6, 13, 13, 11, 2, 2, 1, 1, 13, 6, 13, 6, 5, 3, 5, 4, 2, 12, 6, 7, 6, 12, 11, 14, 2, 6, 8, 6, 1, 2, 1, 1, 12, 6, 16}	36.1642
30	1830	{11, 6, 1, 8, 2, 3, 2, 1, 1, 6, 6, 1, 2, 16, 6, 5, 9, 2, 11, 3, 6, 12, 11, 1, 6, 2, 17, 6, 1, 6, 1, 6, 6, 7, 14, 2, 1, 6, 10, 2, 17, 6, 7, 6, 16, 6, 13, 14, 2, 1}	35.6127
31	1191	{1, 2, 7, 6, 13, 13, 11, 6, 13, 6, 13, 6, 13, 6, 1, 18, 14, 1, 6, 3, 6, 4, 3, 13, 8, 1, 13, 6, 3, 6, 6, 7, 6, 13, 13, 2, 13, 6, 1, 6, 12, 1, 6, 16, 2, 9, 2, 1, 9, 10}	35.7971
32	79	{5, 1, 13, 18, 13, 6, 13, 6, 13, 6, 14, 8, 1, 1, 1, 7, 6, 1, 6, 1, 1, 6, 1, 7, 11, 1, 12, 1, 6, 12, 6, 1, 6, 11, 6, 1, 11, 7, 6, 1, 11, 2, 6, 3, 11, 2, 15, 10, 1, 18}	35.4617
33	1824	{6, 12, 1, 1, 6, 10, 12, 1, 17, 13, 12, 11, 9, 6, 13, 6, 8, 1, 6, 10, 11, 2, 1, 2, 1, 6, 9, 18, 6, 17, 6, 17, 6, 1, 6, 2, 1, 6, 13, 6, 13, 6, 13, 4, 11, 2, 5, 1, 6, 18}	36.0673
34	1964	{2, 6, 8, 6, 13, 1, 8, 14, 6, 1, 2, 10, 3, 10, 12, 16, 10, 11, 2, 1, 6, 8, 1, 2, 6, 1, 6, 13, 6, 7, 2, 3, 14, 6, 6, 16, 6, 13, 6, 13, 6, 6, 17, 1, 12, 8, 1, 6, 1, 17}	35.6508
35	1458	{6, 7, 8, 6, 2, 9, 1, 6, 13, 6, 11, 3, 18, 7, 6, 17, 6, 6, 13, 6, 9, 1, 2, 12, 6, 11, 14, 14, 6, 13, 6, 13, 6, 1, 18, 2, 16, 7, 16, 3, 9, 2, 17, 6, 17, 4, 8, 13, 6, 6}	35.6586
36	1903	{17, 6, 2, 1, 6, 9, 1, 12, 6, 1, 1, 2, 6, 17, 6, 6, 2, 17, 18, 2, 2, 6, 17, 1, 11, 3, 2, 13, 1, 1, 18, 13, 6, 13, 6, 11, 7, 6, 12, 2, 1, 8, 16, 1, 13, 2, 11, 2, 6, 18}	35.1478
37	1248	{1, 6, 1, 6, 1, 2, 11, 12, 1, 6, 12, 14, 6, 14, 1, 2, 12, 2, 17, 6, 2, 6, 6, 1, 1, 12, 8, 14, 6, 18, 7, 6, 8, 6, 7, 10, 1, 6, 1, 6, 8, 6, 13, 8, 6, 1, 2, 6, 2, 6}	35.8983
38	1849	{6, 1, 8, 6, 11, 1, 6, 9, 6, 13, 1, 7, 18, 13, 6, 1, 2, 16, 6, 11, 13, 6, 13, 1, 2, 14, 18, 16, 5, 12, 8, 14, 1, 2, 12, 6, 13, 6, 1, 14, 6, 11, 8, 6, 13, 2, 6, 1, 1, 12}	35.9343
39	575	{1, 6, 12, 2, 9, 6, 18, 16, 13, 6, 12, 8, 1, 6, 8, 1, 1, 1, 2, 17, 6, 4, 2, 17, 6, 7, 1, 2, 9, 2, 1, 8, 6, 1, 2, 1, 6, 1, 2, 11, 6, 13, 3, 1, 13, 1, 6, 2, 11, 2}	36.0059
40	502	{3, 10, 1, 2, 2, 1, 11, 18, 3, 6, 13, 6, 1, 6, 1, 13, 11, 1, 6, 16, 5, 4, 10, 2, 8, 6, 14, 18, 1, 2, 6, 5, 6, 13, 6, 13, 6, 1, 6, 1, 8, 1, 4, 6, 7, 1, 6, 13, 6, 2}	35.6928

[a] Número de la corrida. [b] Numero de la generación en la que se alcanza el óptimo. [c] Valor de la función para la cadena óptima

Tabla A2.2. El mejor hijo reemplaza el peor cromosoma de la población. Resultados de 40 corridas del algoritmo

Corr [a]	Gen [b]	Cadena óptima	Val f [c]
1	1725	{6, 13, 6, 13, 6, 12, 1, 2, 11, 6, 13, 6, 1, 6, 8, 6, 13, 6, 13, 6, 7, 6, 13, 1, 6, 17, 6, 13, 6, 13, 9, 6, 1, 6, 13, 6, 1, 12, 6, 13, 6, 13, 6, 12, 6, 18, 12, 6, 1, 6}	42.0759
2	1727	{12, 6, 1, 6, 13, 6, 13, 6, 6, 2, 6, 1, 6, 12, 6, 1, 6, 2, 1, 12, 6, 13, 6, 1, 2, 1, 1, 13, 6, 10, 16, 2, 1, 7, 1, 2, 1, 6, 13, 6, 13, 6, 13, 6, 1, 2, 17, 6, 13, 6}	40.7757
3	1771	{6, 17, 6, 13, 6, 13, 6, 13, 6, 1, 12, 6, 13, 6, 13, 6, 1, 6, 1, 6, 17, 6, 12, 6, 7, 6, 13, 6, 12, 6, 13, 6, 1, 11, 6, 1, 6, 1, 14, 1, 6, 1, 6, 6, 6, 1, 6, 1, 6, 1}	42.4048
4	1832	{13, 6, 13, 6, 13, 6, 13, 2, 6, 1, 6, 2, 17, 6, 13, 6, 13, 6, 1, 6, 9, 6, 2, 1, 6, 6, 13, 6, 13, 6, 13, 6, 2, 6, 6, 13, 6, 1, 13, 6, 1, 2, 1, 6, 14, 6, 12, 6, 1, 6}	42.0735
5	1519	{6, 13, 6, 12, 1, 6, 13, 1, 6, 7, 6, 13, 6, 1, 13, 6, 13, 6, 12, 6, 1, 6, 6, 6, 13, 6, 6, 1, 6, 13, 6, 1, 11, 6, 13, 6, 1, 6, 1, 6, 8, 6, 1, 6, 6, 1, 13, 6, 13, 6}	42.1239
6	1928	{13, 6, 13, 6, 13, 6, 1, 7, 1, 1, 6, 13, 6, 13, 6, 1, 16, 6, 13, 6, 1, 6, 13, 6, 1, 2, 11, 3, 13, 6, 1, 2, 8, 6, 13, 6, 1, 6, 1, 6, 1, 2, 1, 2, 1, 2, 1, 6, 13, 6}	42.3866
7	1859	{1, 6, 1, 6, 13, 6, 13, 6, 13, 3, 11, 12, 6, 13, 6, 13, 1, 6, 7, 6, 1, 6, 1, 5, 7, 6, 1, 2, 1, 6, 16, 11, 2, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 1, 12, 2, 1, 6, 5}	40.6573
8	1946	{1, 6, 13, 6, 13, 6, 2, 1, 6, 1, 12, 6, 1, 13, 6, 13, 1, 6, 8, 6, 1, 6, 2, 9, 6, 17, 2, 6, 13, 6, 1, 12, 6, 13, 6, 13, 2, 1, 6, 13, 6, 1, 2, 6, 13, 6, 13, 6, 7, 6}	41.69
9	1872	{6, 1, 6, 1, 2, 1, 6, 17, 1, 6, 13, 17, 6, 13, 6, 13, 6, 13, 2, 6, 13, 6, 13, 6, 1, 6, 13, 3, 6, 12, 6, 1, 6, 13, 6, 1, 13, 6, 7, 18, 1, 6, 16, 6, 13, 1, 6, 17, 6, 6}	41.1385
10	1785	{6, 1, 6, 13, 6, 13, 6, 13, 6, 18, 3, 13, 6, 17, 6, 13, 6, 3, 13, 6, 13, 6, 13, 1, 6, 12, 6, 1, 6, 17, 2, 13, 6, 1, 16, 6, 13, 6, 1, 1, 6, 13, 3, 6, 13, 6, 13, 6, 12, 6}	42.0259
11	1718	{6, 1, 6, 1, 2, 6, 17, 6, 6, 13, 6, 16, 1, 6, 1, 2, 1, 2, 12, 6, 13, 6, 13, 6, 13, 6, 17, 6, 16, 6, 17, 6, 1, 6, 17, 6, 1, 17, 2, 1, 1, 2, 1, 6, 6, 1, 12, 6, 13, 6}	41.1503
12	1883	{6, 13, 1, 6, 13, 6, 13, 1, 6, 1, 6, 13, 6, 13, 6, 1, 6, 1, 6, 6, 1, 6, 17, 6, 1, 2, 13, 6, 17, 2, 1, 2, 12, 5, 1, 6, 12, 1, 6, 6, 13, 6, 1, 6, 7, 6, 1, 6, 17, 6}	41.3802
13	1770	{6, 12, 6, 17, 2, 6, 1, 2, 6, 1, 2, 1, 6, 13, 6, 13, 6, 1, 6, 13, 3, 12, 2, 6, 13, 6, 6, 12, 6, 13, 6, 11, 12, 6, 13, 6, 13, 6, 3, 6, 1, 6, 13, 6, 17, 6, 6, 13, 6, 1}	41.222
14	1646	{13, 6, 13, 6, 12, 1, 6, 13, 6, 6, 1, 6, 2, 12, 6, 1, 6, 13, 6, 13, 6, 6, 13, 6, 2, 1, 6, 13, 6, 6, 1, 13, 6, 1, 8, 6, 13, 6, 13, 6, 6, 13, 6, 13, 1, 6, 13, 6, 2, 1}	42.602
15	1792	{16, 5, 6, 6, 13, 6, 13, 1, 6, 13, 6, 14, 6, 13, 6, 6, 12, 6, 13, 6, 13, 6, 1, 6, 10, 6, 1, 6, 6, 8, 2, 1, 6, 13, 6, 1, 6, 12, 6, 1, 1, 6, 13, 6, 13, 6, 7, 6, 13, 6}	41.4415

Tabla A2.2. (Continuación)

Corr [a]	Gen [b]	Cadena óptima	Val f [c]
16	1922	{6, 13, 6, 12, 1, 6, 1, 6, 13, 6, 1, 6, 13, 6, 12, 2, 1, 2, 1, 2, 6, 6, 13, 6, 1, 6, 12, 2, 1, 1, 6, 1, 6, 1, 6, 12, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 6, 13, 6, 1}	43.0691
17	1742	{6, 17, 6, 13, 6, 13, 6, 1, 1, 6, 1, 7, 6, 13, 6, 7, 1, 6, 17, 6, 13, 6, 17, 6, 1, 1, 6, 13, 6, 1, 2, 17, 6, 1, 6, 6, 1, 6, 17, 6, 1, 2, 6, 16, 6, 6, 13, 6, 17, 6}	41.9773
18	1901	{13, 6, 13, 6, 1, 6, 13, 6, 13, 6, 13, 6, 17, 6, 13, 6, 17, 6, 13, 2, 6, 13, 6, 1, 6, 13, 6, 2, 1, 6, 13, 6, 1, 1, 6, 10, 12, 6, 12, 13, 6, 12, 6, 12, 6, 2, 17, 6, 13, 6}	42.4509
19	1647	{6, 17, 6, 12, 6, 13, 6, 13, 6, 12, 1, 2, 1, 6, 13, 6, 1, 2, 1, 13, 6, 1, 7, 6, 13, 6, 13, 6, 13, 6, 8, 6, 2, 1, 6, 12, 1, 6, 13, 6, 12, 2, 1, 2, 6, 17, 2, 12, 6, 1}	41.4464
20	1959	{6, 13, 6, 1, 2, 11, 16, 6, 13, 6, 8, 6, 13, 11, 1, 6, 17, 6, 1, 6, 13, 6, 13, 6, 1, 6, 6, 13, 6, 13, 6, 1, 6, 13, 6, 1, 1, 6, 13, 6, 2, 1, 6, 1, 2, 6, 17, 6, 13, 6}	42.8426
21	1804	{6, 13, 6, 17, 6, 13, 6, 1, 6, 1, 6, 1, 6, 13, 6, 6, 17, 6, 13, 6, 6, 12, 13, 6, 1, 6, 1, 6, 1, 13, 6, 1, 2, 9, 6, 1, 17, 6, 1, 6, 1, 6, 6, 13, 6, 6, 1, 6, 1, 2}	41.7648
22	1963	{1, 2, 3, 1, 6, 13, 6, 1, 12, 6, 13, 6, 13, 6, 1, 6, 13, 1, 2, 1, 11, 6, 17, 6, 13, 1, 1, 13, 6, 9, 1, 12, 1, 2, 11, 2, 17, 3, 6, 13, 6, 10, 1, 6, 13, 6, 13, 6, 13, 6}	40.2103
23	1779	{13, 6, 1, 6, 1, 6, 1, 1, 6, 17, 6, 1, 12, 6, 1, 6, 1, 6, 13, 6, 6, 13, 6, 1, 5, 6, 2, 13, 6, 13, 6, 13, 6, 17, 6, 12, 1, 6, 13, 6, 17, 2, 8, 2, 6, 12, 6, 17, 6, 1}	40.9054
24	1722	{17, 2, 1, 6, 13, 6, 13, 6, 1, 12, 6, 1, 6, 13, 6, 1, 6, 1, 6, 6, 13, 6, 16, 6, 13, 6, 13, 6, 1, 6, 13, 6, 13, 6, 1, 2, 1, 6, 13, 6, 1, 2, 17, 2, 1, 6, 1, 6, 13, 6}	43.9386
25	1948	{8, 2, 17, 6, 6, 1, 2, 1, 6, 13, 6, 13, 6, 6, 13, 6, 1, 6, 13, 1, 1, 6, 1, 6, 3, 1, 6, 1, 6, 13, 6, 13, 6, 1, 6, 13, 1, 11, 1, 6, 6, 1, 6, 6, 13, 6, 1, 11, 2, 1}	41.0449
26	1304	{6, 1, 13, 6, 17, 6, 13, 6, 13, 6, 16, 1, 6, 12, 6, 1, 3, 6, 13, 1, 6, 17, 6, 6, 12, 6, 1, 6, 13, 6, 13, 6, 17, 6, 1, 6, 17, 6, 13, 6, 1, 2, 6, 13, 6, 13, 6, 1, 6, 1}	42.509
27	1992	{6, 7, 6, 17, 6, 1, 2, 6, 12, 6, 13, 6, 1, 6, 1, 6, 17, 6, 1, 1, 6, 1, 6, 1, 6, 6, 12, 6, 17, 6, 13, 6, 1, 1, 6, 6, 13, 6, 1, 2, 12, 1, 6, 13, 6, 17, 2, 12, 6, 1}	41.0055
28	1972	**{6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 6, 1, 13, 3, 1, 6, 13, 6, 1, 6, 13, 6, 13, 6, 13, 6, 6, 13, 1, 6, 1, 11, 1, 6, 1, 6, 13, 6, 1, 6, 1, 12, 6, 6, 13, 6}**	**44.0496**
29	1771	{13, 6, 1, 6, 1, 6, 17, 6, 13, 6, 13, 6, 13, 6, 9, 6, 13, 6, 17, 6, 2, 17, 6, 6, 17, 6, 1, 11, 2, 7, 1, 6, 13, 2, 6, 7, 6, 1, 6, 13, 6, 17, 6, 1, 11, 7, 6, 13, 6, 1}	41.0262
30	1924	{1, 6, 13, 1, 6, 13, 6, 17, 6, 12, 1, 2, 1, 6, 13, 1, 6, 1, 2, 1, 2, 1, 6, 1, 6, 11, 2, 1, 13, 6, 6, 17, 6, 1, 2, 1, 2, 1, 6, 6, 13, 6, 13, 6, 1, 13, 6, 13, 6, 13}	41.6923

Tabla A2.2. (Continuación)

Corr [a]	Gen [b]	Cadena óptima	Val f [c]
31	1938	{6, 13, 6, 17, 6, 12, 1, 6, 17, 6, 13, 6, 16, 6, 16, 6, 12, 1, 6, 13, 6, 6, 14, 2, 1, 2, 1, 6, 17, 2, 6, 13, 6, 6, 13, 6, 13, 6, 1, 6, 6, 1, 1, 6, 7, 6, 13, 6, 17, 6}	41.5446
32	1876	{2, 13, 6, 13, 6, 13, 6, 13, 6, 14, 6, 1, 6, 1, 1, 6, 1, 13, 6, 17, 6, 17, 6, 1, 6, 13, 6, 13, 6, 6, 7, 6, 6, 13, 6, 13, 6, 16, 2, 1, 6, 1, 16, 6, 9, 6, 6, 1, 2, 1}	41.5994
33	1992	{2, 17, 6, 13, 6, 8, 13, 6, 12, 6, 6, 1, 6, 6, 1, 2, 1, 6, 1, 6, 13, 6, 17, 2, 1, 6, 13, 6, 7, 6, 1, 12, 6, 6, 1, 3, 17, 6, 17, 6, 13, 6, 13, 6, 13, 6, 6, 17, 6, 1}	41.0994
34	1963	{6, 6, 13, 6, 1, 6, 13, 6, 2, 1, 6, 13, 6, 13, 6, 6, 17, 11, 1, 6, 12, 6, 13, 6, 6, 1, 1, 6, 13, 6, 13, 6, 8, 2, 1, 6, 13, 6, 1, 6, 12, 8, 6, 13, 6, 3, 6, 1, 6, 2}	41.2568
35	1984	{13, 6, 13, 6, 13, 6, 5, 6, 1, 6, 1, 6, 1, 12, 6, 13, 6, 14, 1, 6, 1, 17, 2, 1, 6, 13, 6, 1, 14, 6, 13, 6, 13, 6, 12, 6, 1, 6, 13, 6, 11, 1, 6, 13, 2, 6, 13, 6, 8, 6}	41.6592
36	1814	{6, 1, 1, 6, 13, 6, 1, 16, 6, 17, 6, 13, 6, 13, 6, 2, 1, 1, 6, 1, 6, 17, 6, 6, 6, 13, 6, 12, 6, 8, 13, 6, 1, 13, 6, 7, 6, 13, 6, 13, 6, 1, 6, 6, 17, 6, 12, 6, 1, 1}	41.1248
37	1581	{6, 13, 6, 1, 2, 6, 1, 1, 6, 13, 6, 12, 6, 1, 14, 6, 11, 1, 6, 1, 6, 6, 12, 6, 13, 6, 13, 6, 8, 6, 1, 6, 17, 6, 13, 6, 13, 6, 1, 2, 6, 13, 6, 1, 1, 6, 13, 6, 2, 1}	41.9453
38	1894	{1, 6, 13, 1, 6, 13, 6, 1, 11, 1, 2, 6, 13, 6, 17, 6, 1, 13, 6, 2, 1, 6, 1, 16, 6, 13, 6, 13, 6, 2, 12, 6, 13, 6, 11, 1, 6, 17, 6, 6, 1, 2, 6, 1, 6, 1, 2, 1, 6, 1}	41.0551
39	1658	{2, 6, 13, 6, 13, 6, 13, 6, 13, 2, 1, 6, 6, 13, 6, 17, 6, 1, 6, 13, 6, 1, 6, 6, 13, 6, 13, 6, 1, 6, 17, 13, 1, 6, 1, 2, 1, 13, 6, 13, 6, 13, 6, 6, 13, 2, 1, 6, 1, 6}	43.1001
40	1711	{6, 13, 6, 13, 6, 6, 13, 6, 1, 2, 13, 4, 2, 1, 1, 6, 13, 6, 1, 6, 13, 6, 16, 6, 1, 6, 13, 6, 13, 6, 1, 6, 12, 6, 7, 6, 13, 6, 2, 1, 6, 1, 8, 6, 1, 2, 6, 13, 6, 1}	42.2326

[a] Número de la corrida. [b] Numero de la generación en la que se alcanza el óptimo. [c] Valor de la función para la cadena óptima

Tabla A2.3. Si el mejor de los hijos es mejor que el mínimo de la población, reemplaza al mínimo. Resultados de 40 corridas del algoritmo.

Corr [a]	Gen [b]	Cadena óptima	Val f [c]
1	1819	{1, 6, 16, 6, 13, 6, 13, 6, 6, 13, 6, 7, 6, 17, 6, 6, 17, 6, 6, 1, 17, 6, 1, 2, 1, 6, 13, 6, 1, 2, 12, 1, 1, 6, 6, 6, 6, 11, 6, 17, 6, 17, 2, 6, 1, 2, 13, 1, 6, 1}	39.7222
2	1671	{7, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 6, 1, 6, 1, 6, 9, 13, 6, 13, 6, 13, 6, 16, 6, 1, 16, 2, 17, 2, 6, 7, 1, 6, 17, 6, 13, 6, 14, 6, 17, 6, 1, 11, 13, 6, 2, 1, 11, 6}	40.8906
3	1956	{1, 11, 1, 6, 1, 13, 6, 7, 6, 13, 6, 12, 6, 1, 6, 12, 6, 13, 2, 7, 6, 12, 1, 6, 13, 6, 13, 6, 6, 1, 13, 6, 5, 6, 12, 6, 13, 6, 2, 11, 2, 1, 6, 1, 11, 2, 17, 2, 11, 2}	39.3903
4	1979	{6, 13, 6, 17, 6, 13, 6, 1, 6, 1, 6, 12, 1, 2, 11, 2, 6, 12, 13, 6, 1, 6, 13, 1, 11, 2, 6, 1, 6, 10, 6, 2, 6, 1, 6, 13, 6, 7, 6, 2, 1, 6, 17, 6, 13, 6, 1, 2, 1, 12}	40.2222
5	1380	{1, 8, 6, 17, 6, 8, 6, 13, 6, 12, 6, 1, 14, 6, 13, 6, 1, 2, 11, 6, 17, 6, 7, 6, 17, 6, 8, 1, 6, 13, 6, 1, 6, 1, 2, 1, 14, 6, 1, 6, 6, 16, 6, 1, 6, 1, 2, 6, 1, 6}	40.1296
6	1625	{6, 7, 6, 16, 9, 6, 1, 6, 6, 12, 6, 1, 6, 1, 6, 13, 1, 6, 1, 12, 6, 1, 6, 1, 17, 1, 6, 17, 6, 11, 17, 2, 17, 6, 7, 6, 13, 6, 13, 6, 13, 6, 1, 6, 13, 6, 13, 6, 7, 6}	40.4689
7	1760	{6, 13, 6, 1, 1, 3, 13, 6, 17, 6, 13, 6, 1, 2, 1, 9, 6, 17, 6, 6, 13, 6, 10, 6, 13, 6, 13, 6, 12, 1, 2, 13, 6, 1, 6, 13, 6, 1, 2, 1, 6, 13, 6, 6, 10, 1, 6, 1, 6, 1}	41.4575
8	1615	{6, 7, 6, 13, 6, 8, 6, 9, 6, 13, 1, 6, 13, 6, 13, 1, 6, 13, 1, 2, 1, 8, 16, 6, 13, 6, 2, 1, 2, 1, 13, 6, 6, 13, 6, 14, 1, 6, 2, 11, 7, 6, 1, 12, 1, 6, 13, 6, 2, 1}	39.7732
9	1914	{2, 1, 11, 2, 6, 13, 1, 6, 1, 6, 1, 6, 13, 6, 6, 16, 6, 13, 6, 13, 6, 17, 6, 13, 6, 17, 6, 13, 8, 4, 1, 2, 1, 2, 1, 6, 6, 13, 11, 12, 6, 7, 6, 13, 6, 1, 6, 2, 1, 6}	40.4315
10	1973	{13, 6, 13, 6, 1, 10, 1, 2, 6, 1, 6, 13, 6, 13, 6, 13, 6, 1, 1, 1, 6, 17, 6, 11, 9, 6, 13, 6, 17, 11, 1, 2, 1, 6, 13, 6, 12, 2, 1, 6, 7, 6, 13, 6, 1, 6, 13, 1, 6, 1}	40.8932
11	1931	{14, 6, 1, 13, 6, 13, 6, 1, 12, 6, 13, 6, 1, 12, 6, 11, 6, 2, 13, 6, 13, 6, 6, 1, 1, 6, 2, 1, 2, 1, 6, 6, 1, 2, 17, 6, 17, 6, 13, 6, 17, 6, 13, 6, 13, 6, 1, 1, 3, 1}	40.62
12	1873	{1, 6, 13, 6, 13, 6, 12, 11, 1, 14, 6, 13, 1, 2, 1, 6, 1, 1, 16, 6, 12, 6, 1, 6, 1, 11, 1, 13, 6, 1, 6, 13, 6, 1, 6, 13, 6, 1, 2, 13, 2, 17, 6, 13, 6, 8, 6, 12, 5, 6}	40.1479
13	1510	{6, 8, 6, 1, 2, 17, 6, 13, 6, 1, 13, 6, 13, 6, 17, 1, 6, 1, 3, 1, 6, 6, 13, 6, 13, 6, 1, 13, 6, 12, 6, 1, 6, 1, 6, 16, 2, 6, 1, 6, 1, 2, 1, 6, 1, 1, 6, 17, 6, 1}	40.817
14	1907	{6, 13, 6, 13, 6, 7, 6, 13, 6, 1, 14, 6, 17, 6, 13, 6, 1, 6, 6, 17, 6, 1, 1, 6, 1, 6, 13, 6, 18, 1, 6, 17, 6, 13, 6, 17, 2, 1, 6, 6, 14, 6, 7, 6, 1, 10, 6, 1, 2, 1}	41.2939
15	1635	{6, 11, 13, 6, 13, 6, 17, 6, 1, 6, 1, 6, 13, 2, 6, 14, 6, 1, 2, 6, 13, 6, 13, 6, 1, 2, 1, 7, 6, 16, 11, 6, 1, 6, 1, 2, 9, 6, 1, 6, 1, 6, 13, 6, 13, 6, 13, 6, 1, 6}	41.6629

Tabla A2.3. (Continuación)

16	1680	{1, 13, 2, 1, 2, 1, 6, 1, 2, 11, 1, 6, 13, 1, 6, 1, 6, 1, 13, 6, 7, 6, 13, 6, 6, 17, 2, 6, 13, 11, 3, 10, 6, 1, 6, 13, 6, 6, 6, 13, 2, 11, 6, 6, 13, 6, 13, 6, 8, 1}	39.7018
17	1932	{6, 1, 6, 12, 2, 8, 6, 1, 3, 11, 12, 6, 17, 6, 13, 6, 12, 1, 6, 13, 6, 13, 6, 13, 6, 10, 1, 14, 2, 6, 1, 6, 1, 14, 6, 1, 2, 11, 1, 6, 13, 6, 13, 2, 1, 13, 6, 7, 13, 6}	39.6508
18	1376	{6, 17, 6, 13, 6, 1, 13, 6, 13, 6, 8, 6, 13, 6, 6, 1, 11, 2, 1, 6, 13, 6, 1, 6, 1, 2, 1, 6, 13, 1, 17, 2, 1, 11, 13, 6, 1, 6, 6, 14, 6, 1, 12, 6, 17, 6, 13, 6, 11, 6}	40.8778
19	1204	{6, 6, 1, 6, 16, 6, 11, 12, 6, 13, 2, 6, 13, 6, 1, 7, 13, 6, 13, 6, 17, 6, 1, 2, 1, 6, 7, 6, 7, 1, 6, 16, 1, 6, 12, 6, 13, 6, 13, 6, 13, 6, 1, 6, 3, 1, 6, 13, 6, 8}	40.2228
20	1777	{1, 6, 13, 6, 13, 6, 13, 6, 14, 6, 1, 2, 6, 13, 6, 13, 6, 12, 6, 9, 6, 13, 6, 13, 6, 1, 11, 1, 6, 9, 3, 6, 1, 6, 1, 1, 2, 1, 8, 17, 6, 13, 6, 6, 13, 6, 13, 6, 17, 2}	41.2905
21	1753	{1, 6, 13, 6, 1, 12, 6, 13, 6, 6, 13, 6, 13, 6, 11, 6, 13, 8, 1, 9, 6, 12, 2, 6, 13, 6, 1, 1, 6, 13, 6, 11, 6, 17, 6, 13, 6, 17, 6, 1, 2, 8, 14, 6, 13, 6, 8, 6, 17, 6}	40.5859
22	1946	{1, 6, 2, 1, 6, 12, 2, 13, 13, 6, 13, 6, 13, 6, 1, 6, 1, 6, 17, 6, 6, 13, 6, 13, 6, 6, 17, 6, 16, 6, 1, 6, 6, 1, 6, 1, 11, 13, 6, 1, 13, 6, 1, 12, 6, 13, 6, 1, 6, 1}	41.1079
23	1918	{13, 6, 1, 6, 7, 6, 17, 6, 2, 13, 6, 13, 6, 13, 6, 7, 6, 1, 6, 6, 7, 11, 6, 1, 13, 6, 13, 1, 2, 11, 6, 1, 6, 13, 6, 13, 11, 7, 6, 17, 6, 13, 2, 6, 13, 6, 13, 6, 17, 6}	41.1159
24	1688	{6, 1, 6, 13, 6, 17, 6, 12, 1, 6, 13, 6, 17, 3, 1, 11, 2, 1, 6, 13, 6, 17, 6, 17, 6, 1, 13, 6, 1, 12, 6, 1, 18, 1, 13, 6, 13, 6, 6, 17, 6, 13, 6, 13, 6, 13, 6, 1, 11, 13}	41.144
25	1754	{6, 13, 6, 2, 1, 6, 1, 6, 1, 12, 6, 12, 11, 6, 1, 2, 6, 13, 6, 13, 1, 2, 11, 6, 13, 6, 1, 12, 6, 1, 12, 5, 1, 2, 13, 6, 13, 6, 13, 6, 1, 6, 13, 6, 6, 13, 1, 6, 13, 6}	41.1335
26	1922	**{6, 8, 3, 1, 5, 6, 13, 6, 13, 6, 13, 6, 13, 6, 17, 2, 1, 2, 6, 6, 1, 6, 13, 6, 1, 6, 1, 6, 1, 6, 1, 6, 17, 6, 13, 6, 12, 6, 13, 6, 13, 6, 1, 6, 6, 13, 6, 1, 12, 1}**	**42.2194**
27	1989	{6, 1, 6, 11, 1, 6, 13, 1, 12, 1, 13, 6, 1, 6, 16, 6, 9, 6, 14, 1, 2, 11, 3, 6, 1, 6, 6, 1, 6, 1, 6, 6, 13, 6, 13, 6, 13, 6, 1, 6, 1, 2, 13, 6, 13, 6, 13, 6, 12, 6}	40.7219
28	1760	{6, 5, 6, 13, 6, 13, 6, 6, 1, 11, 12, 1, 16, 1, 6, 13, 6, 13, 6, 13, 6, 12, 17, 2, 1, 6, 13, 6, 13, 6, 11, 12, 2, 1, 6, 7, 6, 13, 11, 1, 6, 13, 6, 17, 13, 6, 13, 1, 6, 1}	40.46
29	1902	{6, 17, 6, 2, 13, 6, 1, 6, 13, 6, 13, 6, 12, 2, 1, 6, 14, 1, 6, 13, 6, 6, 12, 6, 13, 6, 13, 1, 6, 2, 13, 12, 6, 17, 6, 13, 1, 2, 1, 13, 6, 1, 2, 1, 1, 6, 13, 6, 1, 6}	41.1452
30	1894	{6, 6, 1, 13, 6, 13, 6, 11, 2, 12, 17, 6, 11, 12, 18, 6, 1, 2, 6, 13, 6, 17, 6, 6, 13, 6, 8, 6, 1, 2, 11, 1, 6, 1, 6, 1, 14, 2, 1, 2, 13, 6, 13, 6, 13, 6, 13, 6, 1, 6}	40.2335
31	1179	{1, 12, 6, 1, 6, 8, 1, 6, 1, 6, 1, 6, 7, 6, 8, 17, 6, 1, 6, 13, 6, 7, 6, 1, 16, 6, 13, 6, 8, 6, 1, 2, 6, 12, 6, 16, 1, 2, 1, 6, 13, 6, 12, 11, 6, 7, 6, 13, 6, 1}	39.6522

Tabla A2.3. (Continuación)

32	1894	{6, 1, 6, 1, 6, 13, 6, 6, 1, 6, 13, 6, 1, 2, 6, 17, 6, 1, 6, 6, 7, 6, 13, 1, 8, 1, 2, 1, 6, 13, 6, 13, 6, 1, 6, 9, 6, 13, 6, 1, 6, 14, 6, 10, 1, 6, 13, 6, 13, 1}	41.4745
33	1584	{6, 17, 6, 1, 13, 6, 16, 6, 13, 6, 1, 6, 13, 6, 17, 6, 7, 6, 12, 1, 2, 1, 6, 6, 13, 6, 13, 1, 6, 1, 6, 10, 6, 1, 2, 1, 6, 1, 11, 1, 6, 13, 6, 1, 6, 13, 6, 1, 6, 6}	41.7113
34	1209	{1, 6, 13, 6, 12, 13, 6, 1, 2, 12, 1, 6, 13, 6, 1, 6, 7, 6, 6, 10, 17, 6, 16, 6, 13, 6, 12, 6, 17, 6, 1, 6, 7, 6, 1, 12, 6, 12, 6, 1, 6, 1, 6, 16, 17, 6, 13, 6, 13, 6}	40.4235
35	1923	{6, 1, 13, 6, 13, 6, 1, 17, 6, 1, 2, 13, 6, 1, 2, 13, 6, 13, 6, 1, 6, 1, 11, 3, 13, 6, 13, 6, 12, 2, 12, 1, 2, 1, 6, 13, 6, 13, 2, 11, 2, 11, 6, 13, 6, 11, 16, 6, 11, 1}	40.4481
36	1648	{6, 6, 1, 6, 1, 6, 13, 6, 6, 13, 6, 1, 3, 13, 6, 1, 6, 13, 6, 13, 6, 1, 2, 6, 13, 6, 13, 6, 13, 1, 1, 6, 1, 16, 13, 6, 17, 6, 1, 16, 6, 12, 1, 3, 9, 2, 11, 1, 11, 2}	39.9934
37	1964	{1, 2, 1, 6, 14, 2, 1, 2, 1, 6, 7, 9, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 1, 2, 1, 6, 1, 18, 2, 1, 6, 6, 13, 6, 13, 6, 12, 1, 6, 8, 6, 17, 2, 11, 1, 6, 7, 6, 1, 6}	40.4645
38	1941	{13, 6, 13, 6, 12, 1, 2, 9, 1, 6, 12, 1, 2, 1, 6, 1, 1, 11, 2, 1, 11, 1, 6, 16, 2, 13, 6, 1, 6, 11, 9, 6, 17, 6, 13, 6, 13, 6, 7, 6, 6, 13, 6, 6, 13, 6, 6, 13, 6, 1}	39.8578
39	1896	{1, 10, 1, 2, 1, 6, 13, 6, 11, 1, 6, 6, 17, 6, 1, 1, 1, 6, 17, 6, 16, 1, 14, 1, 6, 13, 6, 17, 6, 12, 1, 6, 17, 6, 13, 6, 12, 2, 1, 7, 6, 11, 2, 6, 17, 6, 13, 6, 1, 6}	39.6606
40	1748	{1, 6, 13, 8, 1, 2, 1, 6, 1, 2, 1, 6, 13, 6, 13, 6, 4, 2, 1, 12, 6, 1, 6, 14, 6, 17, 6, 11, 2, 1, 6, 1, 6, 6, 13, 6, 13, 6, 13, 6, 7, 6, 12, 6, 13, 6, 13, 6, 17, 6}	41.4462

[a] Número de la corrida. [b] Numero de la generación en la que se alcanza el óptimo. [c] Valor de la función para la cadena óptima

Anexo 3. Pruebas numéricas para el Problema 1.Mejor variante del AGC.

Mejor variante del AGC: El mejor hijo reemplaza el peor cromosoma de la población

Parámetros generales

strLength = 250; (* Longitud de la cadena *)

Popsize = 40; (* Tamaño de la Población *)

Probabilidad de mutación: 0.053 y 0.033

Tabla A3.1. Para 100 generaciones. Resultados de 40 corridas del algoritmo.

Corr [a]	Gen [b]	Tiem [c]	Cadena óptima	Val f [d]
1	76	19. 188	{3, 11, 3, 6, 12, 6, 12, 5, 2, 12, 18, 7, 6, 12, 2, 8, 6, 6, 11, 1, 13, 6, 1, 1, 2, 13, 6, 17, 2, 1, 6, 13, 1, 6, 13, 17, 2, 13, 6, 8, 2, 17, 2, 11, 6, 17, 6, 11, 7, 6}	36.6375
2	96	20.219	{6, 1, 6, 1, 12, 2, 14, 6, 13, 6, 7, 6, 17, 1, 2, 1, 6, 13, 3, 10, 1, 3, 6, 13, 8, 2, 13, 1, 2, 8, 7, 4, 11, 13, 6, 7, 6, 13, 6, 13, 6, 1, 12, 1, 2, 12, 6, 1, 12, 1}	37.4403
3	77	20.218	{6, 8, 1, 13, 6, 1, 1, 2, 1, 12, 8, 2, 13, 6, 2, 1, 12, 2, 1, 1, 3, 13, 1, 2, 6, 1, 2, 1, 9, 6, 11, 1, 6, 3, 12, 12, 6, 13, 6, 16, 2, 12, 6, 13, 6, 6, 1, 2, 11, 9}	36.1835
4	75	19.704	{6, 13, 6, 12, 1, 11, 1, 2, 1, 2, 2, 1, 6, 5, 6, 14, 1, 6, 17, 6, 16, 2, 1, 6, 13, 6, 9, 1, 2, 13, 10, 6, 13, 6, 10, 3, 8, 6, 1, 10, 6, 13, 6, 6, 17, 6, 14, 1, 2, 3}	37.5223
5	70	18.812	{1, 6, 4, 14, 8, 17, 6, 1, 2, 6, 1, 2, 1, 6, 5, 9, 1, 6, 14, 6, 12, 16, 2, 12, 1, 10, 1, 6, 13, 6, 7, 1, 14, 8, 16, 12, 6, 13, 6, 2, 1, 6, 13, 6, 13, 18, 12, 6, 17, 6}	36.7822
6	96	18.703	{2, 12, 6, 6, 16, 14, 2, 6, 13, 6, 11, 3, 1, 2, 1, 11, 13, 2, 1, 2, 9, 2, 1, 13, 6, 13, 6, 10, 6, 1, 11, 6, 5, 1, 6, 7, 6, 13, 1, 1, 5, 18, 10, 1, 2, 17, 6, 1, 11, 6}	36.1513
7	97	18.735	{8, 16, 6, 13, 6, 13, 6, 1, 6, 7, 6, 13, 6, 1, 4, 13, 1, 6, 7, 6, 13, 6, 6, 6, 13, 17, 2, 12, 6, 1, 6, 13, 6, 6, 13, 6, 6, 13, 6, 13, 6, 12, 2, 1, 2, 6, 3, 10, 2, 8}	39.3344
8	96	18.703	{6, 13, 13, 1, 2, 1, 6, 13, 6, 11, 6, 17, 2, 8, 13, 6, 10, 2, 1, 6, 9, 6, 13, 6, 13, 6, 13, 6, 11, 16, 6, 12, 5, 1, 14, 1, 2, 1, 6, 14, 1, 6, 9, 6, 1, 14, 6, 13, 6, 7}	38.067
9	91	18.672	{2, 1, 2, 11, 1, 11, 1, 17, 6, 13, 6, 14, 1, 14, 6, 13, 6, 1, 6, 11, 3, 16, 2, 7, 17, 2, 12, 2, 1, 6, 1, 8, 6, 17, 11, 1, 17, 6, 12, 6, 13, 6, 2, 8, 12, 1, 6, 2, 7, 1}	36.4554
10	92	18.75	{2, 13, 6, 9, 2, 6, 6, 2, 1, 13, 6, 13, 6, 6, 8, 2, 9, 6, 17, 6, 13, 6, 17, 9, 6, 13, 12, 6, 13, 6, 1, 9, 6, 17, 6, 12, 2, 1, 5, 13, 10, 6, 13, 6, 1, 2, 11, 13, 6, 12},	37.4766
11	78	18.703	{1, 6, 11, 6, 13, 6, 1, 2, 9, 2, 1, 6, 13, 6, 1, 14, 2, 2, 17, 6, 2, 17, 1, 18, 13, 2, 6, 2, 1, 6, 6, 13, 6, 1, 2, 12, 6, 5, 13, 6, 2, 1, 17, 2, 5, 6, 1, 6, 5, 6}	36.9583

Tabla A3.1 (Continuación)

Corr [a]	Gen [b]	Tiem [c]	Cadena óptima	Val f [d]
12	90	18.703	{6, 17, 13, 6, 13, 2, 1, 6, 2, 1, 2, 1, 6, 1, 6, 1, 8, 2, 1, 6, 12, 1, 6, 6, 10, 1, 1, 2, 7, 2, 6, 13, 6, 1, 2, 17, 2, 11, 2, 1, 6, 1, 6, 4, 11, 12, 11, 16, 6, 11}	37.1149
13	96	18.688	{6, 17, 2, 1, 8, 6, 13, 6, 1, 6, 12, 6, 8, 11, 7, 6, 1, 1, 1, 6, 13, 6, 6, 17, 6, 16, 4, 1, 2, 18, 14, 1, 6, 13, 6, 9, 6, 13, 6, 20, 6, 8, 2, 12, 1, 2, 11, 3, 13, 6}	37.5181
14	98	18.719	{6, 1, 1, 6, 14, 1, 1, 11, 13, 6, 12, 6, 17, 6, 11, 1, 3, 2, 1, 6, 13, 6, 1, 8, 1, 12, 18, 1, 6, 14, 1, 16, 4, 8, 6, 13, 6, 3, 6, 3, 9, 6, 13, 6, 13, 1, 1, 13, 11, 16}	36.3369
15	92	18.672	{1, 2, 1, 7, 1, 13, 6, 4, 17, 1, 6, 13, 6, 17, 2, 6, 12, 6, 7, 1, 12, 6, 1, 13, 9, 6, 16, 6, 17, 2, 9, 5, 1, 2, 12, 1, 2, 9, 11, 13, 6, 13, 6, 1, 12, 1, 6, 1, 8, 2}	36.6295
16	95	18.687	{2, 8, 13, 6, 13, 6, 6, 17, 6, 13, 18, 7, 6, 10, 11, 1, 12, 18, 12, 6, 5, 3, 6, 13, 6, 8, 6, 1, 18, 12, 6, 7, 6, 13, 1, 6, 1, 2, 1, 10, 4, 7, 5, 1, 14, 1, 6, 6, 1, 6}	36.0682
17	92	18.672	{6, 12, 13, 6, 13, 3, 11, 1, 2, 11, 8, 1, 16, 1, 2, 13, 6, 3, 13, 6, 10, 6, 8, 6, 17, 6, 3, 6, 13, 6, 6, 13, 6, 1, 6, 17, 4, 2, 8, 9, 5, 1, 6, 13, 6, 13, 6, 13, 3, 6}	37.5566
19	88	18.781	{8, 6, 6, 17, 6, 2, 1, 6, 7, 11, 20, 6, 1, 6, 13, 2, 6, 13, 6, 11, 6, 1, 13, 6, 13, 6, 18, 10, 1, 10, 4, 1, 6, 13, 6, 13, 6, 18, 6, 17, 6, 6, 13, 6, 9, 6, 7, 6, 12, 6}	37.8192
19	93	18.735	{6, 17, 6, 5, 1, 6, 13, 3, 2, 6, 1, 12, 6, 16, 14, 1, 6, 2, 1, 6, 6, 6, 17, 6, 13, 1, 2, 2, 18, 2, 6, 13, 6, 14, 6, 2, 17, 7, 6, 6, 13, 12, 6, 13, 6, 17, 2, 6, 7, 6}	37.1134
20	79	18.781	{6, 17, 6, 17, 12, 6, 6, 6, 6, 1, 6, 1, 14, 6, 1, 8, 1, 3, 12, 2, 16, 6, 13, 2, 13, 8, 6, 2, 17, 2, 9, 1, 6, 14, 6, 7, 6, 13, 2, 1, 6, 7, 6, 1, 11, 2, 17, 2, 12, 6}	36.7139
21	89	18.719	{1, 6, 11, 1, 3, 7, 6, 10, 2, 11, 1, 3, 1, 1, 6, 1, 13, 6, 13, 6, 13, 6, 7, 18, 11, 1, 6, 1, 6, 16, 2, 14, 11, 6, 1, 2, 6, 13, 3, 11, 6, 13, 6, 1, 8, 6, 1, 6, 6, 2}	36.9395
22	93	18.796	{1, 6, 13, 6, 13, 6, 9, 13, 18, 12, 6, 11, 9, 6, 17, 6, 11, 2, 17, 2, 1, 13, 6, 1, 6, 1, 12, 2, 17, 6, 16, 11, 6, 12, 2, 18, 6, 7, 2, 1, 6, 9, 11, 6, 1, 1, 6, 13, 6, 13}	37.3584
23	99	18.922	{1, 7, 6, 17, 2, 11, 2, 11, 7, 13, 6, 13, 6, 17, 11, 2, 9, 2, 17, 10, 1, 6, 1, 7, 6, 6, 12, 2, 17, 6, 13, 2, 11, 1, 2, 12, 3, 17, 6, 1, 1, 2, 8, 6, 6, 12, 6, 13, 6, 17}	36.7484
24	97	18.953	{6, 13, 8, 6, 13, 1, 1, 6, 13, 6, 10, 4, 6, 1, 3, 1, 6, 1, 6, 17, 8, 6, 13, 6, 13, 6, 10, 4, 9, 13, 6, 13, 6, 1, 1, 2, 2, 17, 6, 1, 6, 13, 6, 13, 2, 6, 1, 2, 6, 1}	38.3098
25	61	20.422	{1, 11, 3, 6, 7, 6, 7, 6, 1, 2, 14, 7, 6, 1, 12, 3, 10, 6, 13, 6, 6, 13, 1, 2, 1, 2, 7, 1, 1, 2, 11, 13, 1, 8, 3, 2, 6, 1, 6, 12, 13, 6, 6, 11, 2, 1, 2, 8, 1, 12}	35.7389
26	92	19.188	{1, 1, 6, 12, 18, 6, 1, 17, 6, 13, 6, 12, 14, 2, 11, 1, 2, 1, 2, 6, 13, 17, 11, 2, 6, 13, 6, 7, 6, 2, 1, 6, 8, 2, 6, 1, 18, 7, 1, 13, 6, 17, 2, 1, 8, 6, 13, 6, 13, 6}	37.8171

Tabla A3.1 (Continuación)

Corr [a]	Gen [b]	Tiem [c]	Cadena óptima	Val f [d]
27	71	18.875	{6, 13, 6, 13, 2, 6, 8, 6, 9, 8, 6, 13, 12, 18, 17, 2, 6, 1, 6, 11, 3, 11, 2, 1, 6, 13, 2, 11, 1, 6, 3, 13, 6, 16, 2, 1, 2, 3, 1, 6, 13, 2, 6, 1, 12, 2, 6, 10, 12, 6}	36.9313
28	92	19.313	{1, 6, 13, 1, 7, 6, 11, 6, 13, 1, 6, 6, 8, 3, 2, 11, 6, 13, 9, 2, 6, 13, 6, 1, 2, 9, 2, 16, 9, 6, 1, 2, 8, 6, 13, 12, 1, 13, 6, 1, 6, 1, 6, 13, 6, 1, 13, 6, 1, 6}	38.1403
29	80	19.125	{12, 6, 8, 6, 13, 6, 13, 1, 12, 12, 10, 4, 8, 1, 2, 6, 1, 2, 1, 6, 1, 6, 1, 6, 10, 9, 6, 12, 13, 1, 6, 13, 1, 5, 1, 13, 6, 13, 1, 2, 12, 1, 6, 1, 8, 6, 1, 14, 12, 6}	36.279
30	86	19.219	{6, 7, 6, 13, 6, 7, 6, 1, 6, 2, 1, 6, 6, 6, 6, 13, 6, 16, 14, 2, 12, 6, 8, 2, 9, 2, 1, 2, 10, 8, 6, 6, 16, 9, 6, 7, 6, 5, 1, 10, 6, 13, 6, 13, 6, 6, 16, 6, 2, 11}	36.6847
31	90	20.234	{6, 1, 2, 9, 6, 16, 6, 13, 6, 13, 1, 2, 8, 1, 5, 6, 13, 6, 13, 2, 11, 18, 6, 17, 3, 6, 1, 2, 1, 6, 14, 1, 2, 17, 6, 1, 2, 12, 17, 1, 6, 13, 6, 5, 1, 12, 5, 6, 13, 6}	37.6293
32	95	19.188	{11, 6, 13, 6, 17, 6, 8, 1, 6, 12, 6, 1, 12, 4, 12, 3, 6, 7, 6, 2, 11, 2, 9, 6, 1, 2, 6, 17, 2, 6, 1, 1, 10, 1, 6, 7, 2, 13, 6, 1, 17, 6, 2, 6, 13, 2, 1, 3, 11, 1}	36.3383
33	84	18.718	{6, 1, 6, 13, 6, 11, 6, 9, 6, 13, 6, 10, 2, 11, 1, 2, 17, 2, 6, 13, 1, 10, 18, 1, 11, 1, 6, 1, 6, 6, 13, 6, 2, 17, 6, 12, 16, 6, 17, 12, 5, 13, 6, 12, 6, 13, 6, 17, 6, 1}	37.6643
34	84	18.766	{5, 1, 2, 1, 2, 1, 2, 8, 6, 13, 6, 14, 1, 2, 1, 6, 1, 11, 12, 6, 7, 6, 12, 2, 12, 6, 3, 13, 9, 6, 3, 13, 6, 12, 2, 1, 2, 12, 13, 2, 13, 6, 7, 2, 11, 3, 6, 1, 6, 1}	36.7965
35	95	19.5	{6, 8, 1, 2, 1, 6, 1, 9, 12, 6, 7, 6, 2, 13, 6, 11, 13, 4, 14, 6, 11, 6, 1, 6, 13, 2, 6, 8, 17, 11, 2, 12, 6, 12, 6, 13, 6, 12, 6, 12, 12, 18, 1, 6, 1, 6, 1, 2, 1, 8}	36.1967
36	93	20.328	{6, 13, 6, 2, 17, 6, 7, 1, 6, 7, 1, 14, 6, 7, 6, 13, 4, 11, 2, 12, 2, 1, 16, 18, 16, 6, 11, 6, 17, 6, 6, 1, 10, 11, 9, 2, 13, 14, 1, 2, 17, 6, 13, 6, 1, 11, 2, 1, 2, 1}	36.7141
37	99	21.453	{5, 6, 1, 6, 1, 2, 11, 6, 1, 13, 6, 7, 6, 1, 9, 2, 8, 6, 13, 2, 14, 6, 1, 2, 13, 1, 6, 7, 6, 6, 12, 2, 1, 5, 1, 6, 6, 13, 11, 7, 2, 17, 6, 13, 6, 13, 11, 1, 6, 5}	36.9425
38	97	19.328	{17, 6, 13, 6, 13, 11, 1, 13, 6, 13, 6, 1, 11, 1, 1, 12, 6, 13, 6, 13, 6, 17, 6, 9, 2, 1, 8, 18, 16, 10, 1, 11, 1, 14, 6, 12, 6, 7, 6, 2, 11, 3, 6, 10, 2, 12, 2, 1, 2, 1}	37.3477
39	89	19.11	{18, 7, 11, 17, 6, 13, 1, 13, 2, 13, 6, 1, 6, 13, 6, 13, 6, 13, 2, 11, 1, 6, 1, 2, 1, 6, 2, 9, 2, 12, 6, 1, 6, 17, 6, 11, 2, 3, 14, 1, 6, 13, 3, 6, 8, 12, 6, 12, 6, 10}	37.6547
40	71	18.953	{12, 6, 1, 11, 12, 5, 8, 2, 3, 6, 13, 6, 1, 6, 8, 1, 2, 11, 2, 1, 14, 6, 6, 2, 13, 6, 14, 11, 9, 18, 18, 13, 6, 13, 6, 17, 1, 11, 2, 13, 6, 13, 6, 13, 4, 1, 6, 17, 11, 2}	36.5889

[a] Número de la corrida. [b] Numero de la generación en la que se alcanza el óptimo. [c] Tiempo de corrida.
[d] .Valor de la función para la cadena óptima.

Tabla A3.2. Para 1000 generaciones. Resultados de 40 corridas del algoritmo.

Corr [a]	Gen [b]	Tiem [c]	Cadena óptima	Val f [d]
1	827	197	{1, 16, 1, 14, 6, 13, 6, 17, 6, 12, 6, 2, 6, 1, 6, 13, 6, 13, 6, 13, 1, 6, 13, 6, 17, 2, 11, 2, 1, 6, 1, 6, 13, 6, 6, 17, 6, 13, 6, 1, 2, 2, 6, 1, 6, 13, 6, 13, 1, 6}	41.5415
2	994	195	{6, 1, 13, 6, 17, 2, 8, 6, 14, 6, 1, 2, 11, 6, 17, 6, 13, 6, 16, 6, 6, 13, 3, 1, 6, 1, 2, 1, 6, 1, 6, 13, 6, 1, 6, 12, 6, 13, 6, 5, 1, 6, 13, 6, 13, 6, 12, 13, 6, 13}	40.5947
3	961	188	{6, 13, 2, 1, 6, 13, 6, 7, 1, 6, 14, 3, 6, 13, 6, 10, 6, 13, 6, 1, 2, 13, 6, 1, 6, 1, 6, 13, 6, 13, 6, 1, 16, 6, 2, 6, 13, 6, 13, 6, 6, 17, 2, 1, 6, 1, 6, 13, 6, 1}	42.0156
4	909	188	{1, 11, 6, 13, 6, 5, 13, 6, 2, 13, 6, 13, 6, 13, 6, 6, 13, 6, 12, 6, 1, 6, 6, 11, 17, 6, 1, 6, 17, 2, 1, 6, 1, 6, 1, 6, 1, 6, 13, 6, 13, 6, 12, 6, 1, 6, 13, 6, 13, 6}	41.8287
5	886	188	{2, 1, 6, 6, 1, 6, 13, 1, 6, 6, 13, 6, 10, 2, 6, 13, 6, 3, 6, 13, 6, 13, 6, 14, 6, 13, 6, 1, 12, 6, 13, 6, 17, 6, 2, 11, 6, 1, 6, 13, 14, 6, 13, 3, 1, 6, 13, 6, 13, 6}	40.8881
6	622	188	{8, 6, 11, 6, 13, 6, 1, 6, 12, 1, 14, 6, 13, 6, 7, 1, 6, 13, 6, 6, 17, 2, 1, 6, 1, 12, 1, 2, 1, 6, 16, 2, 8, 6, 6, 1, 2, 1, 6, 17, 6, 13, 6, 6, 13, 6, 13, 6, 13, 6}	40.7039
7	964	202	{13, 6, 11, 6, 6, 13, 6, 13, 6, 13, 1, 6, 8, 1, 6, 13, 6, 1, 11, 18, 13, 6, 13, 6, 17, 2, 6, 13, 6, 1, 1, 6, 13, 6, 13, 6, 13, 6, 6, 1, 2, 6, 13, 2, 10, 6, 13, 6, 1, 6}	41.5232
8	998	194	{13, 6, 12, 6, 13, 11, 6, 13, 6, 1, 6, 2, 13, 6, 13, 6, 13, 6, 13, 6, 1, 6, 1, 6, 1, 11, 1, 2, 1, 6, 2, 8, 6, 1, 6, 14, 2, 1, 6, 13, 2, 1, 13, 6, 1, 6, 13, 6, 13, 6}	41.7039
9	969	196	{1, 6, 1, 6, 12, 6, 6, 13, 6, 13, 6, 1, 6, 16, 1, 6, 1, 6, 13, 6, 6, 16, 6, 1, 6, 1, 2, 11, 2, 6, 1, 2, 1, 6, 17, 2, 6, 13, 6, 2, 11, 3, 6, 13, 6, 1, 1, 6, 17, 6}	40.7583
10	985	201	{3, 13, 6, 13, 6, 11, 17, 6, 13, 6, 7, 6, 11, 6, 13, 6, 1, 6, 13, 6, 13, 12, 6, 13, 6, 6, 13, 6, 13, 6, 1, 2, 1, 13, 6, 13, 6, 13, 6, 1, 6, 7, 6, 13, 6, 11, 6, 1, 6, 1}	42.4191
11	781	193	{6, 13, 6, 13, 6, 1, 1, 11, 6, 13, 6, 1, 6, 1, 6, 13, 6, 13, 6, 13, 6, 13, 1, 2, 1, 2, 6, 13, 6, 7, 6, 12, 3, 11, 6, 17, 6, 1, 12, 6, 1, 6, 13, 6, 1, 6, 17, 6, 13, 6}	42.4816
12	831	195	{6, 11, 13, 6, 17, 6, 14, 17, 6, 13, 6, 6, 13, 6, 2, 1, 6, 1, 6, 6, 12, 2, 1, 6, 13, 6, 12, 6, 1, 6, 17, 6, 13, 2, 1, 6, 1, 13, 6, 13, 6, 10, 1, 6, 1, 6, 13, 6, 13, 6}	41.4284
13	957	190	{6, 13, 6, 7, 1, 6, 17, 6, 6, 1, 6, 13, 6, 8, 1, 6, 12, 6, 1, 6, 13, 6, 12, 10, 6, 13, 6, 16, 6, 13, 6, 13, 6, 17, 6, 1, 6, 7, 6, 6, 13, 2, 1, 6, 13, 6, 13, 6, 13, 6}	42.4961
14	978	206	{6, 1, 6, 13, 6, 17, 1, 6, 1, 6, 13, 6, 13, 6, 1, 6, 6, 17, 1, 6, 13, 6, 13, 6, 10, 6, 13, 6, 1, 2, 1, 6, 13, 6, 1, 13, 6, 7, 6, 1, 6, 8, 6, 17, 6, 13, 2, 9, 1, 1}	41.7864

Tabla A3.2 (Continuación)

Cor [a]	Gen [b]	Tiem [c]	Cadena óptima	Val f [d]
15	928	198	{2, 1, 6, 13, 2, 6, 10, 1, 12, 2, 17, 6, 1, 6, 13, 6, 13, 6, 12, 6, 17, 6, 13, 6, 13, 6, 1, 6, 1, 6, 13, 6, 13, 6, 1, 8, 3, 1, 12, 2, 1, 2, 1, 6, 13, 6, 13, 6, 13, 6}	42.0335
16	938	203	{6, 1, 2, 1, 6, 1, 6, 13, 2, 1, 6, 13, 6, 17, 16, 6, 1, 11, 2, 11, 2, 1, 2, 1, 6, 17, 6, 1, 13, 6, 13, 6, 6, 17, 6, 6, 17, 6, 1, 2, 13, 6, 17, 6, 1, 6, 13, 6, 13, 6}	41.5791
17	968	223	{6, 13, 6, 6, 13, 6, 1, 6, 2, 6, 13, 6, 1, 7, 6, 13, 6, 1, 6, 13, 6, 14, 6, 8, 6, 17, 6, 13, 6, 1, 6, 1, 12, 1, 6, 13, 18, 13, 6, 1, 6, 13, 1, 6, 13, 6, 1, 6, 13, 6}	42.6973
18	953	193	{13, 6, 6, 17, 2, 13, 6, 7, 6, 13, 6, 6, 1, 2, 13, 1, 1, 6, 13, 6, 13, 2, 6, 1, 6, 1, 6, 3, 13, 6, 13, 6, 1, 6, 2, 1, 16, 6, 13, 6, 1, 6, 13, 6, 1, 6, 1, 8, 6, 9}	40.4543
19	996	188	{6, 13, 6, 17, 6, 17, 6, 1, 5, 18, 1, 4, 17, 6, 13, 6, 2, 12, 6, 13, 6, 13, 6, 3, 6, 7, 6, 13, 6, 13, 6, 13, 6, 1, 6, 13, 2, 1, 6, 6, 7, 6, 13, 1, 2, 1, 6, 13, 6, 1}	40.8082
20	972	190	{6, 6, 17, 6, 14, 1, 6, 17, 6, 13, 6, 13, 6, 8, 6, 13, 8, 6, 13, 6, 1, 2, 1, 1, 6, 13, 1, 6, 1, 6, 13, 6, 2, 1, 6, 2, 13, 6, 13, 6, 10, 6, 1, 6, 1, 2, 1, 6, 13, 6}	41.5889
21	982	189	{1, 1, 6, 13, 6, 17, 6, 1, 6, 6, 1, 6, 13, 3, 11, 5, 6, 1, 2, 1, 13, 6, 13, 6, 13, 11, 1, 6, 14, 6, 1, 6, 2, 1, 6, 12, 13, 6, 2, 6, 1, 6, 1, 2, 1, 6, 8, 6, 1, 2}	39.4474
22	810	193	{6, 13, 6, 1, 9, 6, 17, 6, 17, 6, 13, 6, 13, 6, 12, 6, 1, 1, 6, 1, 6, 7, 1, 12, 6, 13, 6, 12, 6, 6, 1, 1, 6, 1, 6, 2, 1, 8, 6, 13, 11, 2, 1, 2, 17, 6, 7, 6, 13, 6}	40.4076
23	961	194	{1, 6, 13, 6, 13, 6, 17, 6, 2, 9, 17, 6, 11, 12, 6, 12, 2, 1, 1, 6, 6, 1, 6, 13, 6, 13, 6, 1, 6, 13, 6, 6, 1, 6, 1, 6, 6, 1, 6, 11, 6, 7, 6, 1, 11, 1, 2, 1, 6, 1}	40.1583
24	486	195	{6, 8, 6, 1, 6, 13, 6, 12, 1, 2, 1, 8, 6, 1, 6, 1, 2, 13, 2, 12, 6, 13, 6, 13, 6, 1, 6, 10, 1, 6, 17, 16, 6, 1, 6, 13, 6, 13, 6, 9, 6, 2, 1, 6, 13, 6, 13, 6, 13, 6}	41.2912
25	842	190	{1, 14, 6, 17, 6, 1, 6, 13, 6, 13, 1, 6, 1, 6, 12, 1, 6, 1, 2, 11, 6, 6, 1, 6, 13, 3, 6, 1, 6, 13, 6, 13, 6, 1, 6, 13, 6, 13, 6, 6, 1, 10, 6, 6, 1, 6, 13, 6, 1, 9}	41.1825
26	932	188	{1, 6, 6, 17, 6, 2, 11, 12, 6, 1, 2, 1, 12, 16, 6, 17, 2, 1, 6, 17, 2, 6, 17, 6, 1, 2, 6, 13, 6, 13, 6, 13, 6, 16, 6, 13, 6, 13, 6, 7, 6, 7, 1, 6, 1, 6, 13, 6, 1, 16}	40.6556
27	934	188	{1, 6, 13, 6, 8, 6, 1, 6, 13, 6, 1, 6, 2, 13, 6, 1, 6, 1, 6, 3, 1, 6, 13, 6, 6, 18, 6, 2, 1, 6, 9, 1, 2, 17, 6, 2, 1, 6, 13, 6, 13, 6, 13, 6, 1, 6, 13, 6, 13, 6}	41.7572
28	877	189	{6, 13, 6, 13, 6, 1, 6, 13, 6, 13, 6, 2, 11, 1, 6, 1, 6, 1, 6, 2, 1, 6, 1, 6, 1, 13, 6, 13, 6, 1, 3, 13, 6, 13, 4, 6, 1, 6, 13, 6, 17, 18, 2, 11, 13, 6, 13, 6, 13, 6}	42.0426
29	935	189	{6, 13, 6, 13, 6, 1, 6, 13, 6, 13, 6, 1, 14, 6, 1, 1, 6, 6, 13, 6, 13, 6, 1, 10, 6, 14, 2, 1, 13, 6, 1, 6, 1, 3, 6, 13, 6, 1, 6, 13, 6, 6, 1, 6, 1, 6, 13, 6, 13, 6}	42.9243

Tabla A3.2 (Continuación)

Corr [a]	Gen [b]	Tiem [c]	Cadena óptima	Val f [d]
30	923	189	{1, 6, 6, 13, 6, 1, 6, 13, 6, 13, 6, 6, 6, 12, 2, 1, 8, 1, 6, 2, 13, 6, 13, 6, 13, 6, 17, 6, 8, 6, 13, 6, 13, 6, 6, 1, 6, 1, 6, 1, 11, 2, 11, 1, 2, 6, 1, 13, 6, 1}	41.2241
31	973	188	{6, 17, 6, 1, 6, 1, 6, 13, 1, 6, 1, 6, 16, 9, 8, 6, 13, 6, 13, 6, 6, 6, 1, 6, 1, 6, 7, 6, 1, 6, 13, 6, 6, 13, 6, 3, 6, 17, 2, 1, 13, 6, 1, 6, 13, 6, 1, 6, 17, 6}	41.264
32	855	189	{6, 6, 13, 6, 17, 6, 1, 2, 6, 13, 6, 1, 2, 18, 6, 1, 6, 13, 6, 1, 6, 12, 6, 13, 6, 13, 1, 6, 1, 11, 2, 16, 6, 2, 17, 6, 14, 1, 6, 1, 6, 13, 2, 6, 13, 6, 1, 13, 6, 1}	40.9188
33	963	189	{1, 6, 1, 6, 6, 13, 6, 13, 6, 8, 2, 1, 8, 1, 6, 17, 2, 1, 6, 13, 6, 1, 6, 17, 6, 12, 6, 13, 5, 6, 1, 6, 13, 6, 7, 6, 13, 6, 2, 13, 6, 14, 1, 2, 1, 6, 13, 6, 7, 6}	41.0196
34	947	189	{6,6,6,1,6,6,13,6,7,6,1,6,13,6,9,1,6,1,16,6,13,1,6,1,6,13,6,7,1,6,13,6,1,6,6,13,6,1,2,13,6,16,1,6,13,6,17,1,6,6}	41.2135
35	977	189	{1,2,1,6,8,2,1,4,12,1,6,13,6,1,6,13,6,8,1,6,13,6,13,1,6,11,6,1,6,13,6,6,13,6,1,13,6,2,1,6,11,6,1,2,6,13,6,1,16}	40.2783
36	867	188	{12,14,6,13,6,13,6,6,13,6,2,17,6,13,6,13,1,6,1,6,13,6,6,17,6,2,6,5,6,13,6,2,6,13,6,16,1,13,6,6,13,6,11,2,1,6,13,6,1,6}	41.1292
37	959	189	{12,6,13,6,1,6,13,6,1,2,1,18,11,6,1,6,1,6,17,6,16,6,1,2,6,13,6,1,9,1,12,6,13,6,1,17,2,1,6,13,6,17,2,6,13,6,13,6,13,6}	41.3816
38	975	189	{6,1,6,6,13,6,17,6,17,6,12,1,2,1,6,1,6,1,6,1,1,6,1,6,13,6,13,6,3,6,8,1,6,13,6,16,6,17,6,13,6,13,6,6,6,13,6,1,2,16}	42.43
39	968	189	{13,6,13,6,13,6,17,6,17,2,13,6,1,5,6,13,6,7,6,13,6,12,2,6,9,6,1,6,13,6,13,6,1,6,13,6,6,13,6,11,1,13,6,13,6,13,6,1,2,1}	42.3576
40	992	197	{1,6,1,10,1,6,1,6,6,13,6,13,6,7,6,13,6,8,6,1,2,3,11,1,6,1,6,13,6,13,6,12,6,10,6,6,7,6,1,2,1,6,13,6,13,6,7,6,13,6}	41.334

[a] Número de la corrida. [b] Numero de la generación en la que se alcanza el óptimo. [c] Tiempo de corrida. [d] .Valor de la función para la cadena óptima.

Tabla A3.3. Para 3000 generaciones. Resultados de 40 corridas del algoritmo.

Corr [a]	Gen [b]	Tiem [c]	Cadena óptima	Val f [d]
1	2923	568	{6,13,6,13,1,6,13,6,1,6,6,13,6,1,6,17,6,6,1,1,6,7,6,13,6,6,1,1,6,13,6,13,1,6,13,6,1,6,2,13,6,11,1,6,13,6,13,6,13,6}	43.071
2	2780	562	{6,12,6,6,1,6,12,6,6,13,6,13,6,1,6,13,6,16,6,13,1,6,1,6,13,6,13,6,2,1,6,12,6,1,6,13,6,1,6,3,6,1,6,13,1,2,1,6,13,6}	43.1596
3	2924	566	{5,1,6,2,13,6,6,13,6,13,6,13,6,12,1,2,1,6,3,6,8,6,1,6,1,6,1,13,6,13,6,13,6,13,6,17,6,13,6,1,6,1,6,13,6,13,6,17,6,1}	43.3129
4	2443	590	{6,13,6,1,6,13,6,6,1,6,1,6,13,11,6,13,6,12,6,16,6,13,6,13,6,13,6,12,6,6,1,6,1,6,1,6,1,6,1,6,6,13,1,6,13,6,13,6,13,6}	43.5385
5	2585	592	{6,13,6,13,6,6,13,6,13,6,1,6,6,6,13,6,1,6,13,1,12,1,6,6,1,2,1,6,13,6,9,6,13,6,1,6,1,6,6,13,6,13,6,1,6,1,6,13,6,1}	43.4009
6	2854	592	{1,6,17,6,6,1,6,1,2,11,1,6,1,13,6,13,6,13,6,1,6,13,6,1,11,2,1,3,1,6,13,6,13,6,13,1,6,13,6,13,6,1,1,6,13,6,12,6,13,6}	42.8409
7	2187	572	{6,17,6,13,6,13,6,13,6,13,6,13,6,12,6,13,6,10,1,6,6,17,6,6,1,2,11,1,6,13,6,6,1,7,6,13,6,13,6,1,6,13,1,6,13,6,13,6,13,6}	43.7028
8	2952	569	{13,6,1,6,17,6,6,13,6,1,6,1,6,13,6,1,2,1,6,13,2,12,1,6,1,6,13,6,13,6,7,6,1,6,6,1,6,6,13,6,1,6,14,6,13,18,6,13,6,1}	42.2356
9	2823	562	{1,6,13,2,6,13,6,17,6,13,6,11,8,6,13,6,1,6,13,6,14,1,6,13,6,13,6,17,6,13,6,1,6,1,6,6,13,6,1,6,13,6,12,6,2,1,6,13,6,1}	43.1725
10	2841	565	{13,6,13,6,13,6,1,10,1,6,16,1,6,1,6,1,6,13,1,6,1,6,13,6,1,1,6,13,6,13,1,6,13,6,1,1,6,1,6,1,6,1,6,17,6,13,6,13,6,1}	42.7912
11	2448	565	{13,6,13,6,13,6,1,11,16,6,1,2,1,6,13,6,6,13,6,6,13,6,1,2,1,1,2,1,1,2,1,6,1,14,6,11,2,17,6,13,6,1,1,6,13,6,13,1,6,13}	41.4806
12	2531	566	{1,6,7,6,17,6,1,6,13,6,13,6,2,1,2,1,12,6,8,1,6,17,6,7,6,13,6,6,13,6,2,13,6,13,6,17,6,6,1,6,17,6,6,13,6,13,6,6,13,6}	42.2
13	2476	597	{6,13,6,1,6,17,6,2,1,6,13,6,13,6,13,2,1,6,13,6,17,6,13,6,1,6,13,6,13,6,13,6,13,2,1,14,6,7,6,1,6,13,6,13,6,2,1,6,1,6}	44.1405
14	2866	566	{1,6,13,6,6,1,6,1,6,12,6,13,6,13,6,13,6,1,6,13,6,9,6,13,6,13,6,13,6,13,6,8,6,13,6,13,6,13,6,12,9,6,13,6,13,6,13,6,13,6}	44.8006
15	2917	563	{6,13,6,7,6,1,6,13,6,13,6,6,13,6,7,6,17,6,1,6,1,2,1,2,6,1,6,1,1,14,1,6,6,2,17,6,7,6,1,13,6,1,6,17,6,6,13,6,13,6}	41.708
16	2970	588	{6,2,1,6,13,6,17,2,1,1,6,13,6,13,6,13,6,1,6,14,1,6,13,6,1,6,17,6,1,6,1,6,13,6,6,10,6,1,6,13,6,6,13,6,6,13,6,13,2,6}	42.9349
17	2688	573	{2,6,13,6,13,6,13,6,13,6,16,6,13,6,6,12,1,6,13,1,6,1,6,1,6,13,6,1,6,13,2,6,17,6,1,2,6,13,6,1,12,1,6,1,6,13,6,13,6,1}	43.1029
18	2710	572	{6,1,1,6,17,6,6,13,6,13,6,6,13,6,6,1,13,6,13,6,17,2,18,12,6,13,6,13,6,13,6,12,1,1,6,13,6,13,6,13,6,6,6,13,6,13,6,1,6,13}	43.1984

Tabla A3.3(Continuación)

Corr [a]	Gen [b]	Tiem [c]	Cadena óptima	Val f [d]
19	2810	570	6,13,6,13,6,1,1,6,13,6,13,6,6,1,6,1,2,1,11,6,13,6,14,6,13,6,1,17,6,13,6,1,6,1,6,13,6,2,12,6,17,6,13,6,1,6,1,6,13,6}	43.3927
20	2741	569	{6,13,6,13,6,1,6,17,6,1,6,13,6,1,6,13,6,13,6,13,6,17,2,13,6,1,6,12,6,12,6,13,6,13,6,1,6,13,6,13,6,13,6,1,6,13,6,6,13,6}	45.4554
21	2284	607	{1,13,6,13,6,17,6,7,6,2,13,6,17,6,12,1,6,13,6,1,6,6,1,13,6,13,6,12,6,13,6,6,13,6,6,1,1,6,13,6,13,6,6,13,6,1,6,13,6,1}	42.849
22	2627	605	{6,13,6,1,6,13,6,1,2,6,16,1,6,13,6,2,1,6,1,6,13,2,18,1,6,13,6,13,6,13,6,6,13,6,1,12,6,1,6,13,6,2,1,6,13,6,1,2,1,6}	42.8287
23	2627	605	{6, 13, 6, 1, 6, 13, 6, 1, 2, 6, 16, 1, 6, 13, 6, 2, 1, 6, 1, 6, 13, 2, 18, 1, 6, 13, 6, 13, 6, 13, 6, 6, 13, 6, 1, 12, 6, 1, 6, 13, 6, 2, 1, 6, 13, 6, 1, 2, 1, 6}	42.8287
24	2954	596	{6, 12, 1, 6, 13, 6, 1, 3, 6, 12, 16, 6, 1, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 6, 12, 6, 12, 6, 17, 6, 1, 6, 6, 13, 6, 6, 13, 6, 13, 11, 4, 12, 6, 13, 6, 13, 6}	41.8774
25	2940	581	{6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 17, 6, 13, 5, 1, 6, 1, 2, 1, 2, 1, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 6, 1, 6, 1, 6, 1, 6, 13, 6, 13, 2, 1, 2, 17, 6, 1, 6, 1}	44.0604
26	2911	565	{6, 13, 1, 2, 1, 6, 1, 6, 6, 13, 6, 13, 6, 2, 1, 6, 6, 13, 6, 13, 6, 14, 1, 6, 11, 6, 1, 6, 13, 6, 7, 6, 17, 6, 13, 6, 13, 6, 2, 1, 6, 13, 6, 13, 1, 6, 13, 6, 13, 6}	43.3006
27	2620	567	{6, 1, 2, 6, 17, 6, 1, 6, 13, 6, 1, 1, 6, 1, 6, 6, 1, 6, 1, 6, 1, 6, 17, 6, 14, 6, 13, 6, 13, 6, 6, 1, 13, 6, 1, 6, 1, 6, 13, 6, 6, 13, 6, 13, 6, 7, 6, 13, 6, 12}	42.7143
28	2983	565	{6, 14, 1, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 6, 1, 6, 13, 6, 13, 6, 13, 6, 13, 6, 5, 6, 13, 6, 1, 6, 13, 6, 1, 5, 1, 6, 1, 6, 13, 6, 6, 13, 6, 13, 6, 13, 6}	44.8058
29	2919	566	{13, 6, 13, 6, 16, 1, 2, 1, 6, 6, 1, 6, 1, 6, 11, 1, 6, 1, 8, 6, 1, 14, 6, 13, 6, 1, 6, 1, 6, 1, 6, 1, 6, 6, 13, 6, 7, 6, 13, 6, 13, 6, 13, 6, 13, 6, 17, 2, 17, 6}	42.117
30	2965	580	{6, 13, 6, 17, 6, 1, 6, 1, 6, 13, 6, 6, 13, 6, 2, 1, 6, 13, 6, 6, 1, 3, 6, 1, 2, 1, 6, 1, 1, 6, 12, 1, 6, 1, 6, 1, 6, 11, 2, 1, 13, 6, 13, 6, 6, 13, 6, 13, 6, 13}	41.9871
31	2889	585	{1, 2, 1, 6, 13, 6, 8, 1, 6, 13, 6, 1, 6, 1, 6, 13, 6, 6, 13, 6, 13, 2, 1, 6, 12, 13, 6, 1, 13, 6, 1, 6, 13, 6, 13, 6, 6, 12, 6, 13, 6, 17, 6, 1, 6, 6, 13, 6, 13, 6}	43.244
32	1480	595	{2, 1, 6, 13, 6, 13, 6, 13, 6, 6, 13, 6, 1, 6, 13, 6, 6, 8, 6, 1, 6, 17, 6, 13, 6, 13, 6, 6, 1, 6, 8, 1, 6, 1, 13, 6, 13, 6, 1, 6, 6, 1, 6, 13, 6, 13, 6, 13, 6, 6}	43.7619
33	2749	564	{1, 6, 13, 6, 1, 6, 6, 13, 6, 13, 6, 7, 6, 6, 16, 6, 13, 6, 11, 2, 6, 13, 6, 13, 6, 13, 6, 1, 13, 6, 13, 6, 13, 6, 13, 1, 6, 13, 4, 17, 6, 13, 6, 13, 1, 6, 13, 6, 12, 6}	43.1617

Tabla A3.3 (Continuación)

Corr [a]	Gen [b]	Tiem [c]	Cadena óptima	Val f [d]
34	2487	570	{6, 13, 6, 17, 1, 16, 6, 13, 6, 6, 13, 6, 13, 6, 13, 6, 8, 6, 13, 6, 13, 6, 13, 6, 1, 6, 1, 6, 13, 6, 13, 6, 1, 6, 1, 2, 11, 1, 6, 13, 6, 1, 6, 1, 6, 1, 1, 6, 13, 6}	44.1578
35	2163	581	{1, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 7, 1, 6, 13, 6, 1, 6, 1, 2, 6, 13, 6, 13, 6, 1, 6, 1, 6, 13, 6, 6, 1, 8, 6, 17, 6, 1, 1, 6, 13, 2, 1, 6, 13, 6, 17, 2}	43.4338
36	2242	563	{6, 1, 6, 17, 6, 1, 6, 13, 6, 13, 6, 13, 1, 14, 3, 13, 6, 13, 6, 1, 6, 13, 6, 1, 6, 13, 6, 13, 6, 1, 2, 6, 13, 6, 1, 6, 13, 6, 6, 13, 6, 13, 6, 13, 6, 1, 6, 1, 6, 1}	44.4674
37	2916	574	{6, 13, 6, 17, 6, 13, 6, 17, 6, 1, 6, 12, 1, 6, 13, 1, 6, 13, 6, 13, 6, 16, 1, 6, 13, 6, 13, 6, 1, 6, 1, 6, 1, 1, 6, 13, 6, 1, 6, 13, 6, 1, 6, 13, 1, 6, 13, 6, 1, 6}	44.1486
38	2720	576	{2, 1, 6, 1, 8, 12, 6, 12, 2, 1, 13, 6, 1, 13, 6, 7, 6, 1, 6, 13, 6, 2, 11, 6, 13, 6, 13, 6, 13, 6, 6, 7, 6, 13, 6, 7, 6, 1, 6, 13, 6, 17, 6, 6, 1, 1, 6, 13, 6, 1}	41.2845
39	1888	577	{1, 6, 1, 6, 13, 6, 1, 6, 1, 2, 12, 6, 1, 6, 13, 6, 1, 6, 1, 12, 6, 12, 6, 13, 6, 17, 6, 17, 6, 13, 6, 11, 6, 17, 6, 1, 12, 6, 1, 6, 6, 13, 6, 1, 13, 6, 13, 6, 17, 6}	42.2966
40	2691	580	{1, 6, 13, 6, 13, 6, 7, 6, 13, 6, 13, 6, 17, 6, 12, 6, 17, 6, 13, 6, 13, 6, 17, 6, 1, 2, 1, 6, 13, 6, 6, 1, 8, 1, 2, 1, 6, 1, 1, 6, 13, 6, 1, 6, 13, 6, 13, 6, 7, 6}	43.518

[a] Número de la corrida. [b] Numero de la generación en la que se alcanza el óptimo. [c] Tiempo de corrida. [d] .Valor de la función para la cadena óptima.

Anexo 4. Pruebas numéricas para el AGT. Problema1.

En todos los casos los descendientes se introducen en la población solo si son mejores que el individuo menos ajustados

Parámetros generales

strLength = 300; (* Longitud de la cadena *)

Popsize = 40; (* Tamaño de la Población *)

Probabilidad de mutación: 0.053 y 0.033

Tabla A4.1. Para 100 generaciones. Resultados de 40 corridas del algoritmo.

Cor [a]	Gen [b]	Tiem [c]	Cadena óptima	Val f [d]
1	92	263	{13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 17, 6, 13, 6, 13, 6, 13, 6, 17, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 17, 6, 13, 6}	48.0776
2	99	251	{6, 13, 6, 13, 6, 17, 6, 13, 6, 17, 6, 13, 6, 13, 6, 17, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 17, 17, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6}	47.2657
3	96	251	{16, 12, 6, 13}	48.0123
4	94	235	{6, 13, 6, 13, 6, 13, 6, 1, 2, 1, 6, 13, 6, 1, 6, 1, 2, 1, 6, 13, 6, 13, 1, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 2, 1, 6, 13, 6, 1, 6, 13, 6, 13, 6, 13, 6, 13, 6}	46.4533
5	99	233	{6, 13, 6, 1, 6, 6, 13, 6, 13, 6, 1, 6, 13, 6, 1, 6, 6, 13, 6, 13, 6, 13, 6, 1, 1, 6, 13, 6, 1, 6, 1, 6, 13, 6, 13, 6, 1, 1, 6, 13, 6, 13, 6, 13, 6, 1, 1, 6, 13, 6}	45.6484
6	80	246	{6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 12, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6}	48.522
7	99	286	{6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13,6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6,13,6,13, 6, 13, 6,13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13}	48.9873
8	87	251	{1, 6, 13, 6, 1, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 6, 13, 6, 13, 6, 1, 6, 13, 6, 13, 6, 13, 6}	48.2027
9	98	252	{1, 6, 13, 6, 1, 6, 13, 6, 1, 6}	48.6475
10	34	262	{1, 11, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13,6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6,13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13}	48.202

Tabla A4.1 (Continuación)

Cor [a]	Gen [b]	Tiem [c]	Cadena óptima	Val f [d]
11	80	251	{2, 1, 2, 1, 13, 6}	48.1295
12	82	250	{6, 13, 6, 13}	48.9873
13	89	249	{1, 6, 13, 6, 13, 6, 1, 6, 13, 6, 13, 6, 13, 6, 1, 6, 13, 6, 13, 6, 13, 6, 1, 6, 13, 6, 13, 6, 1, 13, 6, 13, 6, 13, 6, 1, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1}	47.5493
14	81	251	{6, 13, 6, 13}	48.9873
15	99	236	{6, 1, 6, 13, 6, 13, 6, 1, 6, 13, 6, 17, 6, 1, 6, 13, 6, 1, 6, 13, 6, 17, 6, 1, 6, 13, 6, 1, 6, 13, 6, 13, 6, 1, 6, 13, 6, 1, 6, 12, 6, 13, 6, 13, 6, 1, 6, 13, 6, 1}	45.8574
16	98	242	{13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 3, 13, 6, 1, 2, 1, 2, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13}	46.895
17	99	258	{1, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 2, 1, 13, 6, 13, 14, 6, 1, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 2, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 2}	45.8598
19	90	252	{6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 6, 17, 6, 7, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6}	47.3936
19	98	253	{6, 13, 6, 13}	48.9873
20	68	262	{6, 13, 6, 13}	48.9873
21	99	257	{1, 6, 13, 6, 13, 6, 13, 6, 1, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 2, 1, 6, 1, 6, 13, 6, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 6, 13, 6, 13, 6, 13, 6, 13}	47.2028
22	96	255	{6, 13, 6, 13, 6, 13, 6, 13, 6, 2, 6, 13, 6,13, 6, 6,13, 6, 13, 6, 13, 6, 6, 13, 6, 13, 6, 1, 2, 6, 1, 2, 6, 13, 6, 13, 6, 13, 6, 2, 6, 13, 6, 13, 6, 6, 13, 6, 13, 6}	45.6972
23	92	248	{6, 13, 6, 1, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 6, 13, 6, 1}	47.9756

Tabla A4.1(Continuación)

Cor [a]	Gen [b]	Tiem [c]	Cadena óptima	Val f [d]
24	97	250	{9, 2, 1, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13,6,13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6,13, 6,13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6}	48.1899
25	96	248	{17, 6, 13, 6, 13, 6, 13, 6,13, 6,13, 6,13, 6,13, 6,13, 6, 13, 6, 13, 6,13,6,13, 6,13,6, 13, 6,13,6, 13, 6, 17, 6, 13, 6, 13, 6, 13, 6, 17, 6, 13, 6,13,6,13, 6}	48.2258
26	96	248	**{6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6,13, 6,13, 6,13, 6,13, 6, 13, 6, 13, 6,13, 6, 13, 6, 13, 6,13, 6,13,6,13, 6, 13, 6, 13, 6,13, 6, 13}**	**48.9873**
27	99	243	{6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 6, 13, 6, 13, 6, 13, 6, 13, 1, 1, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 7, 6, 13, 6,13,6,13, 6, 13, 6, 13, 6, 13}	47.4129
28	88	254	**{1, 6, 13, 6, 13, 6, 13, 6, 13, 6,13, 6, 13, 6, 3, 6, 13, 6, 13, 6, 13, 6, 13, 6,13, 6,13, 6,13, 6,13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6,13,6, 3, 6,13, 6,13, 6}**	**49.0803**
29	97	251	{1, 13, 6,13, 6, 14, 6, 13, 6, 13, 6, 13, 6,13,6,13, 6, 13, 6,13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6,13, 6, 13, 6,13, 6, 13,6,13, 6, 13, 6, 13, 6,13, 6,14, 6, 13}	47.5097
30	93	239	{6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6,13, 6, 1, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1,6,13,6,13,6, 13, 6, 13, 6,1, 6,13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1}	48.4267
31	95	243	**{6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6,13, 6, 3, 6,13, 6,13,6,13,6,13, 6, 13}**	**48.9873**
32	96	241	{1, 6, 13, 6, 13, 1, 6, 17, 6, 14, 1, 6, 17, 6, 11, 1, 6, 13, 6, 13, 6, 13, 6, 6, 1, 6, 1, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 6, 1, 6, 1, 6, 13, 6, 13, 6, 13, 6, 1, 6}	44.6806
33	99	235	{1, 6, 1, 13, 6, 1, 13, 6, 13, 6, 13, 6, 13, 6, 13, 1, 6, 1, 13, 6, 1, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 13, 6, 13, 6, 13, 1, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1}	45.5819
34	99	240	{6, 13, 6, 1, 6, 13, 6, 13, 6, 13, 1, 6, 13, 6, 13, 6, 13, 6, 13, 6, 14, 6, 13, 6, 1, 6, 13, 6, 13, 6,1,6,13, 6, 11, 6, 13, 6, 13, 6, 1, 6, 13, 6, 13, 6,13, 6, 13, 6}	46.6055
35	93	254	{6, 13, 6, 13, 6,13, 6,13, 6, 13, 1, 6, 13, 6, 6, 16, 13, 6, 13, 6, 13, 6,13, 6, 3, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6,13, 6, 13,6,13,6,13,6, 3,6, 13, 6}	47.6547
36	97	276	{6, 6, 13, 6, 6, 13, 6, 6, 13, 6, 13, 6, 13, 6, 6, 13, 6, 6, 13, 6, 13, 6, 6, 13, 6, 6, 13, 6, 13, 6, 6, 13, 6, 6, 13, 6, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 6, 13, 6, 6},	45.5691
37	96	248	{6, 1, 6, 13, 6, 1, 6, 13, 6, 8, 1, 6, 13, 6, 13, 6, 1, 6, 13, 6, 13, 6, 1, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 6, 13, 6, 13, 6, 13, 6, 1, 6, 13, 6, 13, 6}	47.0668

Tabla A4.1(Continuación)

Cor [a]	Gen [b]	Tiem [c]	Cadena óptima	Val *f* [d]
38	90	239	{6, 13, 6, 13, 6, 1, 2, 1, 6, 7, 6, 13, 6, 13, 6, 1, 2, 1, 6, 7, 6, 7, 6, 7, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 7, 6, 13, 6, 13, 6, 13, 6, 13, 6}	45.1991
39	97	241	{6, 1, 12, 6, 9, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 6, 13, 6, 6, 17, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1}	46.7226
40	99	244	{6, 1, 6, 13, 6, 1, 6, 13, 6, 1, 1, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 6, 13, 6, 13, 6, 1, 6, 13, 6, 1, 6}	47.3079

Número de la corrida. [b] Numero de la generación en la que se alcanza el óptimo. [c] Tiempo de corrida. [d] .Valor de la función para la cadena óptima.

Tabla A4.2. Para 1000 generaciones. Resultados de 40 corridas del algoritmo.

Cor [a]	Gen [b]	Tiem [c]	Cadena óptima	Val *f* [d]
1	185	2784	{6, 13, 6, 13}	48.9873
2	143	2754	{6, 13, 6, 13}	48.9873
3	158	2796	**{6, 13, 6, 1}**	**49.0803**
4	194	2909	{13, 6, 13, 6}	48.9873
5	168	2870	**{6, 13, 6, 1}**	**49.0803**
6	119	2378	{6, 13, 6, 13}	48.9873
7	338	2718	{6, 13, 6, 13}	48.9873
8	127	2761	{1, 6, 13, 6},	49.0803
9	92	2803	{6, 13, 6, 13},	48.9873
10	101	2791	{13, 6, 13, 6},	48.9873
11	117	2767	**{1, 6, 13, 6},**	**49.0803**

Tabla A4.2 (Continuación)

Cor [a]	Gen [b]	Tiem [c]	Cadena óptima	Val f [d]
12	301	2783	{6, 13, 6, 13, 6, 13, 6, 1, 6, 13, 6, 1}	48.8654
13	112	2721	**{1, 6, 13, 6}**	**49.0803**
14	221	2755	{13, 6, 13, 6}	48.9873
15	413	2822	{6, 13, 6, 13}	48.9873
16	129	2829	{13, 6, 13, 6}	48.9873
17	197	2766	**{1, 6, 13, 6}**	**49.0803**
18	224	2690	{6, 13, 6, 13}	48.9873
19	118	2748	{13, 6, 13, 6}	48.9873
20	108	2770	{6, 13, 6, 13}	48.9873
21	193	2789	{6, 13, 6, 13}	48.9873
22	118	2779	{13, 6, 13, 6}	48.9873

Tabla A4.2 (Continuación)

Cor [a]	Gen [b]	Tiem [c]	Cadena óptima	Val *f* [d]
23	105	2760	{6, 13, 6, 13}	48.9873
24	188	2696	**{1, 6, 13, 6}**	**49.0803**
25	113	2795	{6, 13, 6, 13}	48.9873
26	132	2769	**{1, 6, 13, 6}**	**49.0803**
27	178	2793	{6, 13, 6, 13}	48.9873
28	118	2762	{6, 13, 6, 13},	48.9873
29	103	2796	{13, 6, 13, 6},	48.9873
30	193	2711	{6, 13, 6, 13},	48.9873
31	77	2726	**{1, 6, 13, 6}**	**49.0803**
32	193	2727	{6, 13, 6, 13}	48.9873
33	159	2752	{6, 13, 6, 13}	48.9873

Tabla A4.2 (Continuación)

Cor [a]	Gen [b]	Tiem [c]	Cadena óptima	Val *f* [d]
34	76	2769	{13, 6, 13, 6}	48.9873
35	764	2791	**{6, 13, 6, 1}**	**49.0803**
36	143	2772	{6, 13, 6, 13}	48.9873
37	174	2737	{13, 6, 13, 6}	48.9873
38	123	2754	**{1, 6, 13, 6}**	**49.0803**
39	92	2760	{13, 6, 13, 6}	48.9873
40	112	2750	**{1, 6, 13, 6}**	**49.0803**

[a] Número de la corrida. [b] Numero de la generación en la que se alcanza el óptimo. [c] Tiempo de corrida.
[d] .Valor de la función para la cadena óptima.

Tabla A4.3. Para 500 generaciones. Resultados de 32 corridas del algoritmo.

Cor [a]	Gen [b]	Tiem [c]	Cadena óptima	Val f [d]
1	218	1453	{6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 17, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13}	48.6834
2	136	1454	{6, 13, 6, 13}	48.9873
3	158	1419	**{6, 13, 6, 1}**	**49.0803**
4	359	1306	{6, 13, 6, 13}	48.9873
5	86	1468	{6, 13, 6, 13}	48.9873
6	161	1483	{6, 13, 6, 13}	48.9873
7	164	1362	{6, 13, 6, 13}	48.9873
8	148	1388	{6, 13, 6, 13}	48.9873
9	124	1355	{13, 6, 13, 6}	48.9873
10	493	1416	**{6, 13, 6, 1}**	**49.0803**
11	398	1270	{6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 6, 1, 6, 13, 6, 1, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1}	48.4267

Tabla A4.3.(Continuación)

Cor [a]	Gen [b]	Tiem [c]	Cadena óptima	Val f [d]
12	39	1369	{13, 6, 13, 6}	48.9873
13	111	1354	{6, 13, 6, 13}	48.9873
14	148	1372	{13, 6, 13, 6}	48.9873
15	115	1377	{6, 13, 6, 1}	49.0803
16	146	1369	{6, 13, 6, 13}	48.9873
17	119	1371	{6, 13, 6, 1}	49.0803
18	179	1374	{6, 13, 6, 13}	48.9873
19	165	1357	{6, 13, 6, 13}	48.9873
20	120	1368	{13, 6, 13, 6}	48.9873
21	61	1368	{1, 6, 13, 6}	49.0803
22	213	1350	{1, 6, 6, 13}	48.5319
23	91	1369	{2, 1, 6, 13}	48.5235

Tabla A4.3.(Continuación)

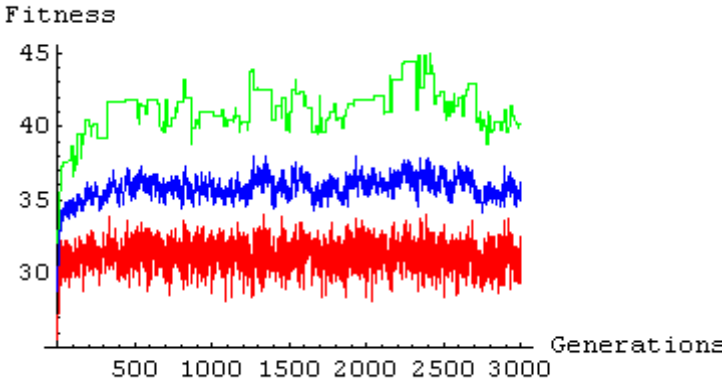
Cor [a]	Gen [b]	Tiem [c]	Cadena óptima	Val *f* [d]
24	175	1343	{6, 13, 6, 1, 6, 13, 6, 1}	48.8654
25	347	1345	{6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1}	48.8654
26	172	1380	{6, 13, 6, 1}	49.0803
27	110	1372	{6, 13, 6, 1}	49.0803
28	96	1566	{1, 6, 13, 6}	49.0803
29	197	1543	{13, 6, 13, 6}	48.9873
30	110	1474	{1, 6, 13, 6}	49.0803
31	94	1481	{1, 6, 13, 6},	49.0803
32	196	1492	{6, 13, 6, 1}	49.0803

[a] Número de la corrida. [b] Numero de la generación en la que se alcanza el óptimo. [c] Tiempo de corrida. [d] .Valor de la función para la cadena óptima.

Anexo 5. Ejemplos de salidas del AGC

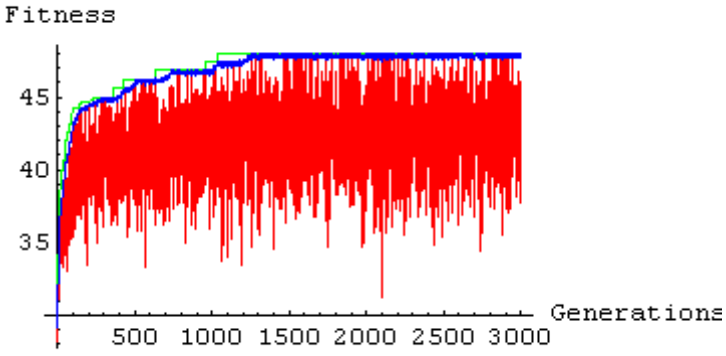
Problema 1.Cadenas de 20 aa

Variante cero: El mejor hijo, de dos descendientes, reemplaza al peor padre



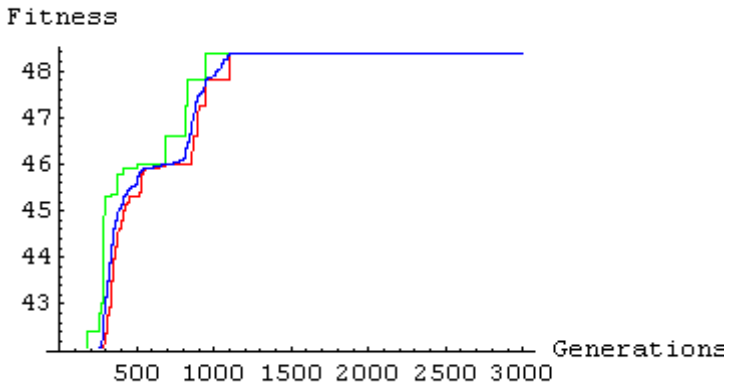
{The best fitness was obtain in generation number: 2418 with phenotype: {6, 13, 6, 13, 6, 1, 6, 13, 6, 1, 13, 6, 12, 6, 13, 6, 13, 6, 17, 6}, 44.9835}

Variante uno: El mejor hijo reemplaza el peor cromosoma de la población



{The best fitness was obtained in generation number: 1037 with phenotype: {6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 6, 13, 6, 13, 6, 13, 6, 13, 6}, 48.0211}

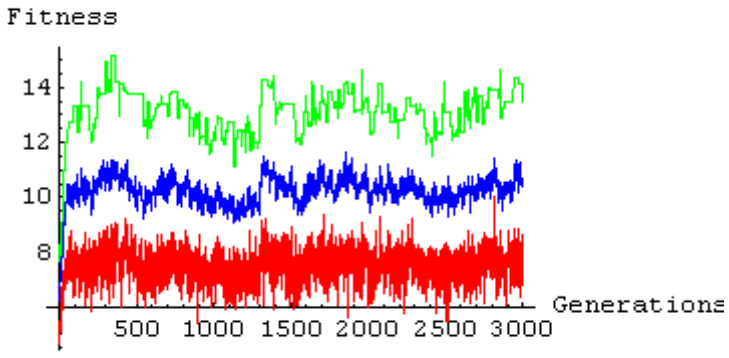
Variante dos: Si el mejor de los hijos es mejor que el mínimo de la población, reemplaza al mínimo



{The best fitness was obtained in generation number: 950 with phenotype: {6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 1}, 48.3672}

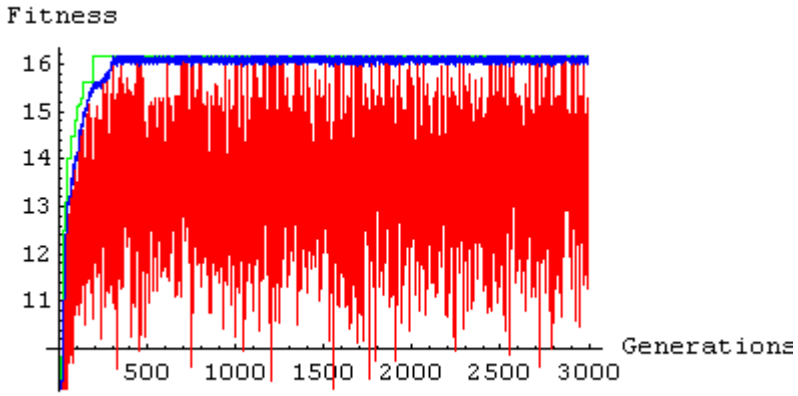
Problema 3.Cadenas de 20 aa.

Variante cero: El mejor hijo, de dos descendientes, reemplaza al peor padre



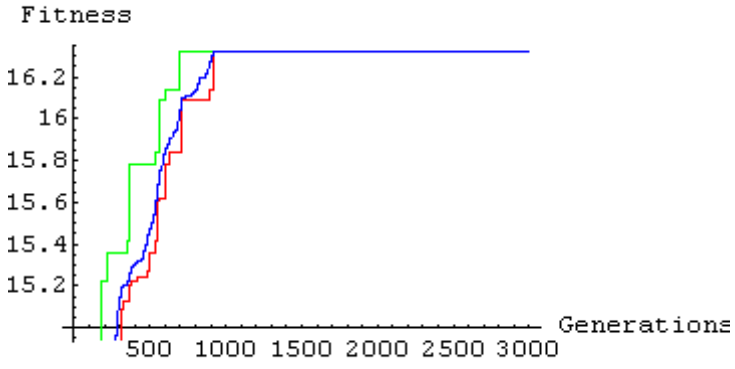
{The best fitness was obtained in generation number: 336 with phenotype: {3, 11, 2, 11, 6, 6, 6, 13, 2, 11, 2, 6, 6, 6, 6, 6, 6, 6, 6, 6}, 15.1771}

Variante uno: El mejor hijo reemplaza el peor cromosoma de la población



{The best fitness was obtained in generation number: 338 with phenotype: {6, 6, 6, 6, 6, 13, 6, 6, 6, 6, 6, 6, 13, 2, 11, 20, 11, 6, 6, 6}, 16.1731}

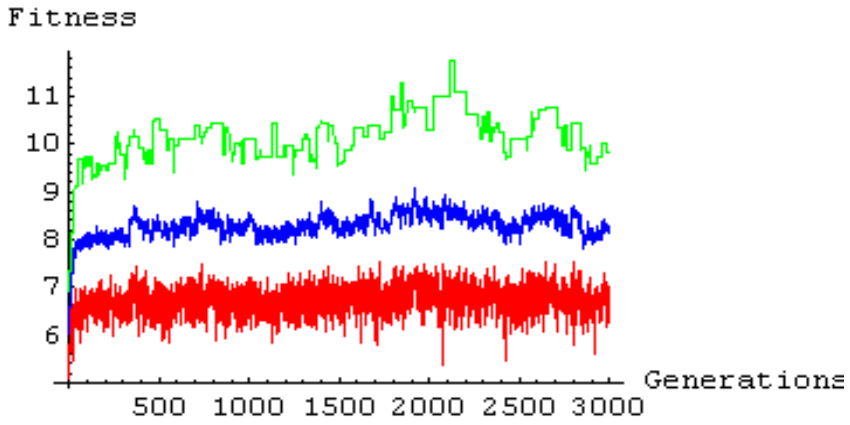
Variante dos: Si el mejor de los hijos es mejor que el mínimo de la población, reemplaza al mínimo



{The best fitness was obtained in generation number: 695 with phenotype: {6, 6, 6, 6, 6, 6, 6, 6, 7, 11, 20, 11, 6, 6, 6, 6, 7, 11, 6, 6}, 16.317}

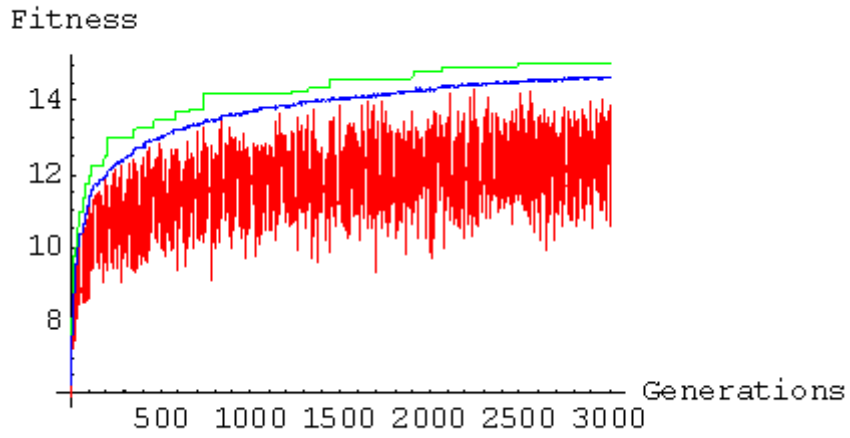
Problema 3.Cadenas de 50 aa.

Variante cero:



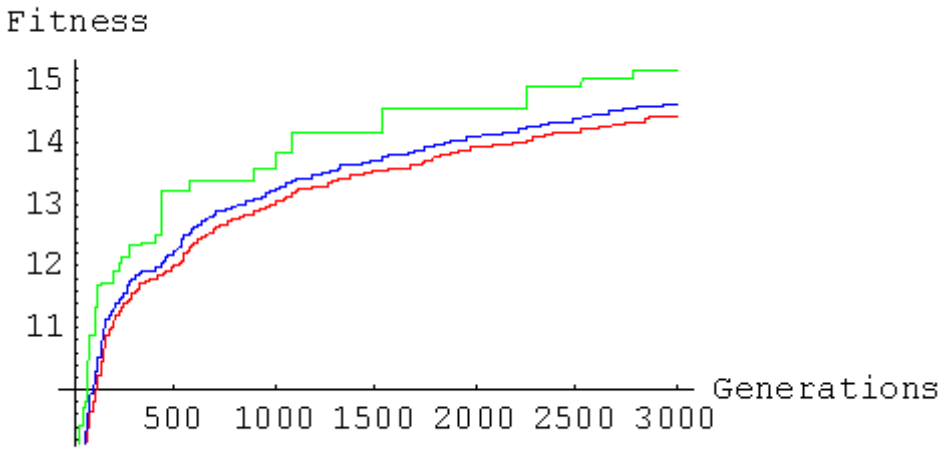
{The best fitness was obtained in generation number: 2119 with phenotype: {6, 6, 12, 16, 3, 14, 2, 11, 2, 11, 18, 12, 6, 6, 6, 6, 6, 7, 11, 6, 3, 17, 17, 19, 7, 6, 6, 6, 6, 13, 6, 6, 6, 6, 12, 11, 2, 6, 12, 16, 4, 11, 7, 3, 2, 11, 6, 6, 6, 6}, 11.7692}

Variante uno:



{The best fitness was obtained in generation number: 2488 with phenotype: {6, 6, 7, 11, 20, 11, 7, 1, 6, 6, 6, 6, 6, 11, 1, 6, 6, 6, 6, 6, 17, 19, 6, 6, 6, 6, 11, 6, 6, 6, 6, 6, 6, 6, 6, 7, 11, 6, 7, 11, 2, 13, 6, 6, 6, 6, 6, 6, 6, 6}, 15.0335}

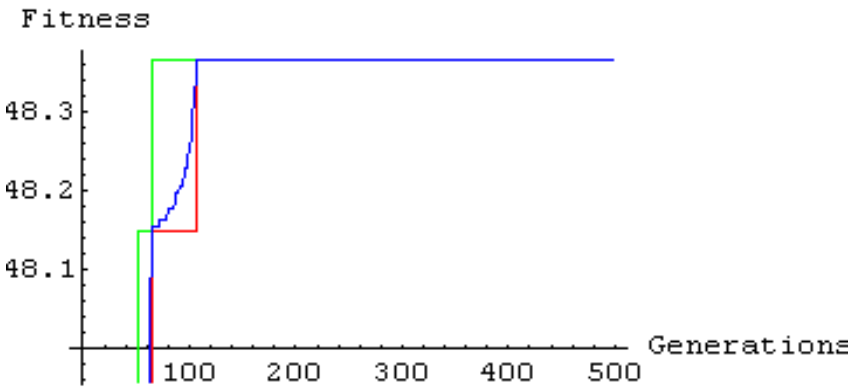
Variante dos:



{The best fitness was obtained in generation number: 2782 with phenotype: {6, 6, 6, 7, 11, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 2, 11, 2, 11, 20, 11, 7, 11, 20, 11, 2, 6, 6, 6, 6, 6, 7, 11, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 12, 6, 6, 9, 9, 6}, 15.1679}

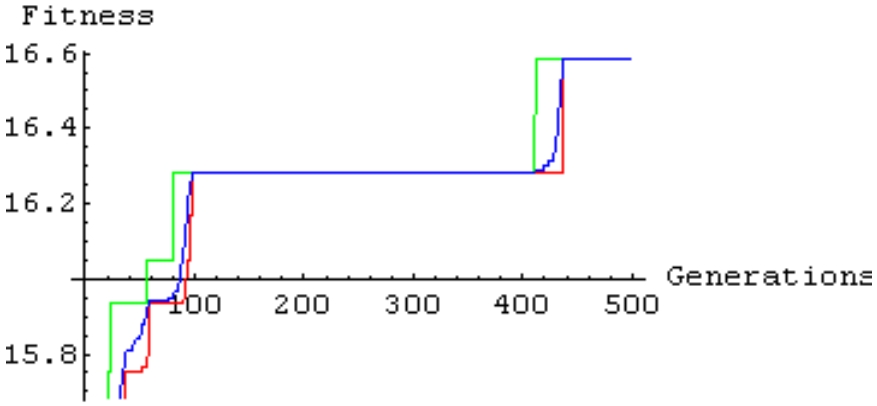
Anexo 6. Ejemplos de salidas del AGT

Problema 1.Cadenas de 20 aa



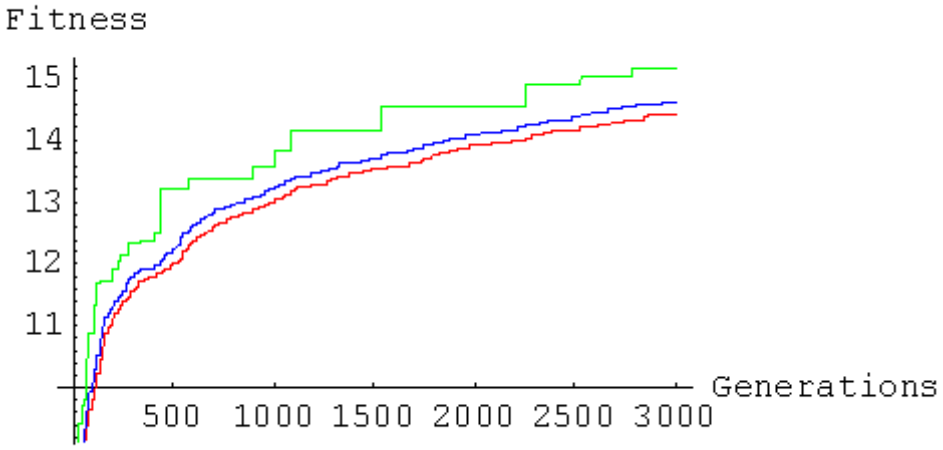
{The best fitness was obtained in generation number: 66 with phenotype: {1, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6, 13, 6}, 48.3672}

Problema 3. Cadenas de 16 aa.



{The best fitness was obtained in generation number: 412 with phenotype: {6, 6, 6, 6, 6, 6, 7, 11, 20, 11, 6, 6, 6, 6, 6, 6, 6, 6}, 16.5847}

Problema 3. Cadenas de 50 aa.



{The best fitness was obtain in generation number: 101 with phenotype: {6, 6, 6, 7, 11, 6, 7, 11, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 11, 6, 6, 6, 6, 6, 7, 11, 6, 6, 6, 7, 11, 6, 6, 7, 11, 6, 6, 6, 6, 6, 7, 11},16.0669}

Anexo 7. Pruebas estadísticos realizados para el Problema 3

Comparación de las variantes propuestas para el AGC.

Variantes que se comparan

Grupo 0: Variante cero del AGC: El mejor hijo, de dos descendientes, reemplaza al peor padre

Grupo 1: Variante uno: El mejor hijo reemplaza el peor cromosoma de la población

Grupo 2: Variante dos: Si el mejor de los hijos es mejor que el mínimo de la población, reemplaza al mínimo.

$\boldsymbol{X_{1j}}$: Número de la generación en que se alcanza el óptimo para cadenas de 20 aminoácidos

X_{2j}: Valor de la función para cadenas de 20 aminoácidos

$\boldsymbol{X_{3j}}$: Número de la generación en que se alcanza el óptimo para cadenas de 50 aminoácidos

X_{4j}: Valor de la función para cadenas de 50 aminoácidos

$\boldsymbol{X_{5j}}$: Número de la generación en que se alcanza el óptimo para cadenas de 100 aminoácidos

X_{6j}: Valor de la función para cadenas de 100 aminoácidos

donde el índice j recorre los grupos 0, 1 y 2.

Tabla A7.1. Comparaciones entre los pares de variables.

	$X_{10} - X_{12}$	$X_{20} - X_{22}$	$X_{30} - X_{32}$	$X_{40} - X_{42}$	$X_{50} - X_{52}$	$X_{60} - X_{62}$
Chi-Square	.199	69.296	49.663	94.766	17.958	89.155
Df	2	2	2	2	2	2
Asymp. Sig.	.905	.000	.000	.000	.000	.000

Se aprecia que al comprar los resultados de los grupos 0 y 2 las diferencias son altamente significativas para todas las variables excepto para el número de la generación para la cual se alcanza el óptimo para cadenas de longitud 20.

Tabla A7.2. Comparaciones entre los pares de variables

	X_{10}- X_{11}	X_{20}- X_{21}	X_{30}- X_{31}	X_{40}- X_{41}	X_{50}- X_{51}	X_{60}- X_{61}
Chi-Square	.196	54.511	39.606	59.260	10.736	58.376
Df	1	1	1	1	1	1
Asymp. Sig.	.658	.000	.000	.000	.001	.000

Tabla A7.3. Comparaciones entre los pares de variables

	X_{11}- X_{12}	X_{21}- X_{22}	X_{31}- X_{32}	X_{41}- X_{42}	X_{51}- X_{52}	X_{61}- X_{62}
Chi-Square	.078	9.553	.002	34.567	1.589	44.468
Df	1	1	1	1	1	1
Asymp. Sig.	.780	.002	.965	.000	.207	.000

Las diferencias son altamente significativas solo para los valores de la función.

Comparación de la mejor variante del AGC con el AGT

Test de Kruskal Wallis: Grupo 1 y 2

Grupo 1: Mejor variante del AGC

Grupo 2: AGT

Y_{1j}**:** Número de la generación en que se alcanza el óptimo para el AGC para 100 generaciones

Y_{2j}**:** Tiempo de corrida para .100 generaciones

Y_{3j}**: V**alor de la función fitness para la cadena óptima para 100generaciones

Y_{4j}**:** Número de la generación en que se alcanza el óptimo para el AGC para 1000 generaciones

Y_{5j}**: T**iempo de corrida para 1000 generaciones

Y_{6j}**:** Valor de la función fitness para la cadena óptima para 1000generaciones

donde el índice j recorre los grupos 0, 1 y 2.

Tabla A7.4. Comparaciones entre los pares de variables

	Y_{11}- Y_{12}	Y_{21}- Y_{22}	Y_{31}- X_{32}	Y_{41}- Y_{42}	Y_{51}- Y_{52}	Y_{61}- Y_{62}
Chi-Square	14.114	59.260	53.764	13.023	59.260	9.542
Df	1	1	1	1	1	1
Asymp. Sig.	.000	.000	.000	.000	.000	.002

Las diferencias son altamente significativas para todas las variables consideradas.

Anexo 8. Glosario de términos biológicos usados en la tesis

Los términos biológicos usados en la tesis son [23]

Alelo: (del griego *allelon* = "el uno al otro", recíprocamente): Formas alternativas de un gen, se hereda separadamente de cada padre (p. ej. en el locus para el color de ojos puede haber un alelo para ojos azules o uno para ojos negros). Uno o más estados alternativos de un gen.

Cromosomas (del griego *khroma* = color; *soma* = cuerpo): Estructuras del núcleo de la célula eucariota que consiste en moléculas de ADN (que contienen los genes) y proteínas (principalmente histonas).

Evolución (del latín *e-* = fuera; *volvere* = girar)**:** Cambio de los organismos por adaptación, variación, sobrerreproducción y reproducción/sobrevivencia diferencial, proceso al que Charles Darwin y Alfred Wallace se refirieron como selección natural.

Expresión: En genética, proceso por el cual la información codificada en los genes se convierte en estructuras operacionales presentes en la célula.

Fenotipo (del griego *phaineim* = mostrar, *typos* = imprimir, estampar): Características observables de un individuo. La expresión de la composición alélica para un determinado carácter bajo estudio (***Lo que se ve***).

Genes (del griego *genos* = nacimiento, raza; del latín genus = raza, origen): segmentos específicos de ADN que controlan las estructuras y funciones celulares; la unidad funcional de la herencia. Secuencia de bases de ADN que usualmente codifican para una secuencia polipeptídica de aminoácidos.

Genética: el estudio de la herencia de los caracteres

Genotipo: La totalidad de los alelos de un organismo.

Herencia (del latín *haerentia*= pertenencias, cosas vinculadas) Transmisión de características de padres a hijos.

Locus (del latín: lugar, plural *loci*): Posición que ocupa un determinado gen en un cromosoma

Meiosis (del griego *meio* = menor; *meiosis* = reducción): División celular en la cual la copia de los cromosomas es seguida por dos divisiones nucleares. Cada uno de los cuatro gametos resultantes recibe la mitad del número de cromosomas (número haploide) de la célula original.

Mitosis (del griego *mitos* = hebra, filamento): La división del núcleo y del material nuclear de una célula; se divide usualmente en cuatro etapas: profase, metafase, anafase, y telofase. La mitosis ocurre únicamente en eucariotas. El ADN de la célula se duplica en la interfase y se distribuye durante las fases de la mitosis en las dos células resultantes de la división.

Mutación (del latín *mutare* = cambiar): El cambio de un gen de una forma alélica a otra, cambio que resulta heredable.

Mutante: Organismo que lleva un gen que ha sufrido una mutación

Recesivo: Término que se aplica a un carácter (alelo) que solo se expresa cuando el segundo carácter (alelo) es igual.

Segregación: separación de los cromosomas durante la división celular.

Anexo 9. Módulo optimizador del AGC

Dado el número de ensayos a realizar, e módulo siguiente (AG) ejecuta el algoritmo para optimizar la función utilizando los operadores anteriormente definidos. Los valores máximo, medio y el mínimo de los *fitness* que se registran en cada generación son dados como salida. Finalmente, se da el cromosoma correspondiente al mejor *fitness* encontrado en todas las generaciones.

```
GA[Generations_] := Module[
   {TotalFitness, probs, index1, index2, parent1, parent2, child,
    bestindex,   solution, mins = {}, means = {}, maxs = {}},

  GeneStringArray = InitialPop;

  FitnessArray = Map[Function[lista, Fitness1[Phenotype[lista]]],
    GeneStringArray];

  MaxFitness = Max[FitnessArray];

  i = 1;
  While[Generations > i,

   (* Ruleta rusa: se obtiene un vector de probabilidades
      dividiendo cada fitness por la suma total de
      fitness *)

   TotalFitness = Plus @@ FitnessArray;
   probs = FitnessArray / TotalFitness;

   (* Aplicación de los operadores para obtener una
    nueva generación *)

   c = 1;
   While[c ≤ Popsize - 1,

    (* Seleciona dos
     padres: parent1 con probabilidad probs de acuedo
       a su fitness y parent2 con distribución uniforme
       en el intervalo [0,1] *)

    index1 = selectParent[probs]; (* Operador de selección *)
    index2 = Random[Integer, {1, Popsize}];
    parent1 = GeneStringArray[[index1]];
    parent2 = GeneStringArray[[index2]];
```

```
(* Entrecruzamiento, operador crossover *)

    children = cross[parent1, parent2];
    child1 = Mutate[children[[1]], 0.0533];
    child2 = Mutate[children[[2]], 0.0333];

    (* En encuentra el cromosoma con menor fitness
      de la última generación lo reemplaza por el
      mejor hijo *)

    children = {child1, child2};

    (* El mejor hijo *)

    If[Fitness1[Phenotype[child1]] > Fitness1[Phenotype[child2]],
     BestChild = 1, BestChild = 2]; (* Encuentra el mejor hijo *)

    (* El peor cromosoma *)

    MinFitnessIndex = Ordering[FitnessArray, 1][[1]];

    (* El mejor hijo reemplaza el peor cromosoma *)

    GeneStringArray[[MinFitnessIndex]] = children[[BestChild]];

    FitnessArray[[MinFitnessIndex]] =
      Fitness1[Phenotype[children[[BestChild]]]];

    c = c + 1];

  FitnessArray = Map[Function[lista, Fitness1[Phenotype[lista]]],
     GeneStringArray];

  (* Parámetros estadísticos de la nueva generación *)

  AppendTo[mins, Min[FitnessArray]];
  AppendTo[means, Mean[FitnessArray]];
  AppendTo[maxs, Max[FitnessArray]];
  If[MaxFitness < Max[FitnessArray], MaxFitness = Max[FitnessArray],
    Null];
  i++;
  ];
```

```
  BestPhenotype =
    Phenotype[
     GeneStringArray[[Position[FitnessArray, MaxFitness][[1, 1]]]]];

bestindex = Position[FitnessArray,
        Max[FitnessArray]][[1, 1]];
solution = {GeneStringArray[[bestindex]],
                    Phenotype[GeneStringArray[[bestindex]]],
     Max[FitnessArray]};
Return[{solution, mins, maxs, means, BestPhenotype}]
  ];
(* Final del Module *)
```

Anexo10. Módulo optimizador del AGT

Dado un número de generaciones el módulo siguiente(AG) ejecuta el algoritmo para optimizar la función utilizando los operadores anteriormente definidos. Los valores máximo, medio y el mínimo de los *fitness* que se registran en cada generación son dados como salida. Finalmente, se da el cromosoma correspondiente al mejor *fitness* encontrado en todas las generaciones.

```
GA[Generations_] := Module[
  {TotalFitness, probs, index1, index2, parent1, parent2, child,
   bestindex,  solution, mins = {}, means = {}, maxs = {}},

  GeneStringArray = InitialPop;

  FitnessArray = Map[Function[lista, Fitness1[Phenotype[lista]]],
    GeneStringArray];

  MaxFitness = Max[FitnessArray];

  i = 1;
  While[Generations > i,

   (* Ruleta rusa: se obtiene un vector de probabilidades
      dividiendo cada fitness por la suma total de
      fitness *)

   TotalFitness = Plus @@ FitnessArray;
   probs = FitnessArray / TotalFitness;

   (* Aplicación de los operadores para obtener una
    nueva generación *)

   c = 1;
   While[c ≤ Popsize - 2,

    (* Seleciona dos
     padres: parent1 con probabilidad probs de acuedo
       a su fitness y parent2 con distribución uniforme
       en el intervalo [0,1] *)

    index1 = selectParent[probs]; (* Operador de selección *)
    index2 = Random[Integer, {1, Popsize}];
    parent1 = GeneStringArray[[index1]];
    parent2 = GeneStringArray[[index2]];
```

```
(* Entrecruzamiento, operador crossover *)

children = cross[parent1, parent2];
child1 = Mutate[children[[1]], 0.0533];
child2 = Mutate[children[[2]], 0.0333];
children = {child1, child2};

(* Encuentra el mejor hijo *)

FitnessChildren = {Fitness1[Phenotype[child1]],
   Fitness1[Phenotype[child2]]};

If[FitnessChildren[[1]] > FitnessChildren[[2]], BestChild = 1;
  BestFitnesChild = FitnessChildren[[1]], BestChild = 2;
  BestFitnesChild = FitnessChildren[[2]]];

(* Si el fitness del mejor hijo es superior al menor
  fitness de la última generación lo reemplaza *)

MinFitnessIndex = Ordering[FitnessArray, 1][[1]];
If[FitnessArray[[MinFitnessIndex]] < BestFitnesChild,
  GeneStringArray[[MinFitnessIndex]] = children[[BestChild]]; h = 1;
  j = 0; FitnessArray[[MinFitnessIndex]] = BestFitnesChild, h = 0, j = 1];

(* Transposon *)

(* Seleción de un cromosoma por la ruleta *)

index = selectParent[probs];

(* Generación de un cromosoma inmigrante utilizando
  el operador Transposon *)

Immigrant = Transposon[GeneStringArray[[index]]];

MinFitness = Min[FitnessArray];
IndexMinFitness = Ordering[FitnessArray, 1][[1]];

(* El cromosoma inmigrante se acepta en la población
  si es mejor que el peor cromosoma de la generación
  presente *)

If[Immigrant[[2]] > MinFitness,
  GeneStringArray[[IndexMinFitness]] = Immigrant[[1]]; k = 1; j = 0;
  FitnessArray[[IndexMinFitness]] = Immigrant[[2]], k = 0; j = 1];

(* Final del Transposon *)
```

```
c = c + h + k + j];

    FitnessArray = Map[Function[lista, Fitness1[Phenotype[lista]]],
       GeneStringArray];

    AppendTo[mins, Min[FitnessArray]];
    AppendTo[means, Mean[FitnessArray]];
    AppendTo[maxs, Max[FitnessArray]];
    If[MaxFitness < Max[FitnessArray], MaxFitness = Max[FitnessArray],
      Null];
    i++;
    ];

  BestPhenotype =
    Phenotype[
     GeneStringArray[[Position[FitnessArray, MaxFitness][[1, 1]]]]];

  bestindex = Position[FitnessArray,
        Max[FitnessArray]][[1, 1]];
solution = {GeneStringArray[[bestindex]],
                    Phenotype[GeneStringArray[[bestindex]]],
     Max[FitnessArray]};
Return[{solution, mins, maxs, means, BestPhenotype}]
  ];
(* end Module *)
```