INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

"USO DE MODELOS DEL SISTEMA INMUNE PARA MANEJO RESTRICCIONES CON ALGORITMOS GENÉTICOS"

T E S I S

QUE PARA OBTENER EL GRADO DE :

DOCTOR EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A :

RICARDO PARAMONT HERNÁNDEZ GARCÍA

ASESOR: DR. ÁNGEL FERNANDO KURI MORALES
COASESOR: DR. JESÚS GUILLERMO FIGUEROA NAZUNO

MÉXICO, D.F. SEPTIEMBRE DE 2004

Reconocimiento a la DGAPA de la UNAM.

El autor de este trabajo agradece la beca recibida por parte del Programa de Apoyo a la Superación del Personal Académico (PASPA) de la Dirección General de Asuntos del Personal Académico (DGAPA) de la Universidad Nacional Autónoma de México, para llevar a cabo esta investigación.

Reconocimiento a la FES Cuautitlán de la UNAM.
El autor agradece la comisión académica para llevar a cabo este trabajo de investigación, concedida por la Facultad de Estudios Superiores Cuautitlán de la Universidad Nacional Autónoma de México.

A mis padres.

Índice

Lista de Figuras

Lista de Gráficas

Lista de Tablas

Glosario.

Afinidad. Es la fuerza con la que se une un receptor o anticuerpo con un epitope. Para que se desencadene la respuesta inmune debe existir gran afinidad entre el receptor y el epitope.

Algoritmos Evolutivos. Así se denomina a todos aquellos algoritmos que se inspiran en la evolución de los seres vivos para resolver algún problema, casi siempre de optimización. Especialmente aquéllos que pertenecen a la Programación Evolutiva, Estrategias Evolutivas o Algoritmos Genéticos.

Algoritmos Genéticos. Es un conjunto de algoritmos que utilizan generalmente cadenas finitas de símbolos sobre algún alfabeto finito para representar los puntos del espacio de búsqueda de un problema. A estas cadenas se les acostumbra llamar *cromosomas* o *individuos*. Utilizan *operadores genéticos* que utilizan sobre una lista de cadenas a la que se llama *población*, para llevar a cabo la búsqueda del óptimo. Los operadores más usados son: selección, cruzamiento y mutación. Cada cadena es evaluada por medio de una función de aptitud para poder seleccionarla o descartarla.

Anticuerpo. Es una inmunoglobulina libre, es decir, que no está unida a la membrana celular de un linfocito T.

Aptitud. Es una calificación que se le da a un individuo perteneciente a alguna población de individuos de un algoritmo evolutivo. Esta calificación por lo general es un número en los reales positivos y es tal que dados dos individuos el mejor de ellos tiene una calificación mayor que el otro.

Apoptosis. Es la muerte programada de las células. Cuando el proceso de apoptosis se presenta la célula se destruye quedando sus partes almacenada en vejiguitas que posteriormente son devoradas por los fagocitos. De esta forma no se dañan las células vecinas a ella lo que no ocurriría si el contenido de la célula se dejara escapar, ya que el contenido incluye algunas encimas proteolíticas que dañarían a las otras células.

Bloques Constructores. Son aquellos esquemas que subsisten a lo largo de muchas generaciones del un algoritmo genético y que al yuxtaponerse (concatenarse) entre sí forman la solución global (el óptimo global) del problema.

Células de Memoria. Son las células que permanecen después del ataque de algún patógeno. Las hay del tipo B y del tipo T. En sus membranas celulares poseen receptores con las regiones variables que identificaron al patógeno. Circulan por el organismo en concentraciones no muy grandes pero suficientes para desencadenar respuesta inmune si el mismo patógeno u otro con los mismos o similares epitopes de él, penetra nuevamente al organismo.

Células Plasmáticas. Son verdaderas fábricas de anticuerpos (aproximadamente 2000 por segundo cada célula) que surgen de la diferenciación de un linfocito B nuevo cuyos receptores han embonado con un epitope.

Célula Presentadora de Antígeno (APC). Son células que fagocitan a los patógenos, los destruyen en su interior y algunas de sus partes características las presentan engarzadas en una molécula llamada Complejo de Histocompatibilidad Mayor tipo II (MHC II por sus siglas en inglés) sobre su membrana celular para que sean identificados por un tipo de linfocitos T.

Correlación Espuria. Se presenta cuando el algoritmo genético no identifica bien un bloque constructor y conserva a lo largo de algunas generaciones al bloque constructor más algún o algunos otros símbolos junto con él . Esto provoca que no puedan yuxtaponerse a este bloque los demás bloques constructores. El algorimo genético requiere de más generaciones para "darse cuenta" que el bloque constructor no incluye los símbolos extra que viene acarreando.

Dimensionalidad. Es el número de variables independientes de una función.

Elitismo Total. Este término se usa para indicar que para un algoritmo evolutivo para el que se use una población de padres de tamaño μ, se conserven siempre los mejores μ indivi-duos obtenidos durante la búsqueda. Esta forma de proceder puede provocar que algunas veces la población P^{i+1} sea una copia de la población anterior P^i porque ninguno de los hijos de la población P^i resultó ser mejor que el peor de los individuos de esa población.

Epitope. Es una molécula a la que se une un receptor de un linfocito B, un linfocito T o un anticuerpo. Esa molécula puede pertenecer a un patógeno o a una parte del mismo organismo (epitope propio). Los epitopes de los linfocitos B no son, por lo general, los mismos que aquéllos de los linfocitos T.

Esquemas. En los algoritmos genéticos el espacio de búsqueda se representa como una lista finita de puntos. Cada uno de estos puntos se representa como una cadena finita de símbolos sobre un alfabeto finito de caracteres. Un esquema es un conjunto de puntos que presentan un patrón en particular. Para representar los esquemas se usan cadenas de la misma longitud que aquéllas para representar los puntos en el espacio de búsqueda. Estas cadenas se escriben con el mismo alfabeto más un metasímbolo extra. A tal representación se le acostumbra llamar también esquema.

Espacio de Búsqueda. Es la parte del dominio de una función donde el método de optimización lleva a cabo su búsqueda del óptimo.

Fagocito. Es un tipo de glóbulo blanco que introduce en su interior a los patógenos que encuentra. En su interior los destruye.

Función Engañosa. Es aquel tipo de función que impide que se formen algunos bloques constructores indispensables para obtener, por yuxtaposición con otros, el óptimo global de la función en el espacio de búsqueda.

Función Multimodal. Es aquella función que no es unimodal.

Función Unimodal. Es aquella función f: $R^n \rightarrow R$ tal que para toda pareja de puntos $\boldsymbol{x}_1$ y $\boldsymbol{x}_2$ de su dominio se cumple que $f(\lambda \boldsymbol{x}_1 + (1 - \lambda)\boldsymbol{x}_2) \leq \max\{f(\boldsymbol{x}_1), f(\boldsymbol{x}_2)\}$ para $\lambda \in (0, 1)$.

Inmunidad Celular (o Inmunidad Mediada por Células). Son los mecanismos que se desencadenan en un animal vertebrado para defender al organismo de patógenos que penetren al interior de sus células. Los principales actores en esta inmunidad son los linfocitos T. Las células presentadoras fagocitan a patógenos que penetran al organismo. Los destruyen en su interior y presentan algunos pedazos selectos en sus superficies. Estos pedazos selectos constituyen epitopes de las células T y van engarzados (como una joya en un anillo) en una proteína llamada Complejo de Histocompatibilidad Mayor tipo II(MHC II por sus siglas en inglés). Las células T que pasan pueden probar si sus receptores embonan en al complejo epitope-MHC II. Las células infectadas por virus que penetran a ellas presentan parte de los virus que a ellas han entrado engarzados en una proteína llamada Complejo de Histocompatibilidad tipo I (MHC I). Aquellas células T que embonaron bien con los complejos epitope-MHC II reconocen al complejo epitope-MHC I y destruyen a la célula que lo presenta, destruyendo con ella al patógeno en su interior.

Inmunidad Humoral. Es el conjunto de mecanismos que presenta un animal vertebrado para defenderse de los patógenos disueltos en los líquidos de su organismo. Los linfocitos B cuyos receptores embonan con algún epitope del patógeno producen una gran cantidad de copias de sí mismos que luego se diferencian en células de memoria y en células plasmáticas. Éstas últimas producen unas gran cantidad de anticuerpos que circulan por los líquidos del organismo.

Inmunidad Innata. También llamada primaria. Es la inmunidad no especializada. Como su nombre lo indica, ya la tienen los organismos al nacer. Se refiere al las barreras que presenta el organismo para defenderse de los patógenos. Tales barreras son: la piel, la fagocitosis, las mucosas, barreras químicas y fisiológicas. La piel es una barrera física a los patógenos externos. La fagocitosis se refiere la destrucción de patógenos que llevan a cabo las células llamadas fagocitos, que son diferentes tipos de glóbulos blancos. Introducen en su interior a los patógenos, allí los destruyen por medio de encimas proteolíticas. Las mucosas como la piel, también son barreras físicas para los patógenos externos. Las barreras químicas se refieren al uso de substancias donde los patógenos no pueden subsistir, como es el caso del ácido clorhídrico en el estómago. Las barreras fisiológicas se refieren a mecanismos del organismo tales como el aumento de la temperatura corporal para matar a algunos patógenos que no soportan estos aumentos.

Inmunoglobulina. Es una proteína que puede constituir un receptor superficial de un linfocito B o un anticuerpo. Tiene forma de “Y”, las puntas superiores de la “Y” constituyen la región variable de la inmunoglobulina. Hay una gran variedad de puntas distintas cada una de ellas puede embonar en un epitope distinto.

Linfocitos. Son las células del sistema inmune. Estas células son los actores principales de la inmunidad adquirida (también llamada *secundaria* o *adaptable*). Se dividen principalment en linfocitos B y linfocitos T. Los linfocitos B son los principales actores de la inmunidad humoral y los T de la inmunidad celular (en inglés *cell-mediated immunity*)Tienen estructuras unidas a sus membranas celulares y que se proyectan hacia el exterior del linfocito. En la puntas de estos receptores están las llamadas regiones variables. Son llamadas así porque hay una gran diversidad (miles de millones) de ellas, que embonan en una gran diversidad de epitopes.

Manejo de restricciones. Son los procedimientos que se aplican durante la optimización pa-ra procurar que la búsqueda se lleve a cabo en puntos de la región factible.

Opsonosis. Es el fenómeno mediante el cual los fagocitos incrementan grandemente (de 3000 a 4500 veces) su capacidad de identificar y destruir patógenos. Se presenta cuando los epitopes de un patógeno se han unido con un anticuerpo, la parte del anticuerpo que apunta hacia la parte exterior del patógeno atrae a los fagocitos que se presentan para destruirlo.

Óptimo Global. Si se maximiza, es el punto al que le corresponde el mayor valor de la función objetivo dentro del espacio de búsqueda. Si se minimiza, es el punto al que le corresponde el menor valor de la función objetivo dentro del espacio de búsqueda.

Óptimo Local. Sea *V* el conjunto de puntos dentro de espacio de búsqueda que rodean a un punto *p* también dentro del espacio de búsqueda. Se dice que *p* es un *máximo local* si no existe otro punto en *V* al que le corresponda un valor mayor que aquél que le corresponde a *p*. Se dice que *p* es un *mínimo local* si no existe otro punto en *V* al que le corresponda un valor menor que aquél que le corresponde a *p*.

Optimización. Búsqueda el óptimo de una función dentro del espacio de búsqueda.

Óptimo restringido. Es el óptimo de la función objetivo respetando las restricciones.

Óptimo no restringido. Es el óptimo de la función objetivo sin restricciones, o sin respetarlas si las hay.

Patógenos. Agentes capaces de causar daño a un organismo. Pueden ser bacterias, hongos, protozoarios, virus, substancias tóxicas naturales o artificiales. El mismo organismo puede producir a veces patógenos, por ejemplo, tumores malignos.

Punto Extremo. En una función derivable, es aquel punto donde la derivada de la función en todas las direcciones vale cero.

Población Intermedia. En un algoritmo genético, es aquella población donde se copian los mejores individuos conforme se van seleccionando de otra población. A los individuos de esa misma población se les aplica el cruzamiento y la mutación y la población intermedia pasa a ser la población de siguiente del proceso de evolución.

Respuesta inmune. Término con el que se denota la puesta en marcha de los mecanismos de defensa del organismo cuando entra a él un patógeno.

Región Factible. Es aquélla parte del dominio de la función donde se satisfacen las restric-ciones del problema.

Sistema Inmune. Es el conjunto de células y mecanismos encargados de defender a un orga-nismo vivo de los patógenos, marcándolos y destruyéndolos.

Sistemas Inmunes Artificiales. Son algoritmos que han sido inspirados por algunas caracte-rísticas del sistema inmune de los seres vivos.

Sistema Inmunitario. Véase *Sistema Inmune*.

Vasconcelos. Cruzamiento determinista de una población. Para aplicar el *Vasconcelos* primero debe ordenarse la población de tal manera que el individuo más apto sea el primero y el menos apto el último. El primer individuo de cruza con el último, el segundo con el penúltimo y así sucesivamente. La razón para usar este tipo de cruzamiento es mantener la diversidad genética.

Abstract

Three evolutionary algorithms for solving instances of the non linear programming problem were designed and implemented. Some ideas taken from the field of the immune systems were adapted to include them into the algorithms. These ideas are principally models of some mechanisms of the most evolved of the natural immune systems, the natural immune system of jawed vertebrates (NIS), mainly the human's and the murine's. These mechanisms were the *somatic recombination* to build the genes of the receptors' variable regions of B lymphocytes, and the *affinity maturation* of the B lymphocytes' receptors.

It was not pretended to simulate or imitate those mechanisms of the NIS, but to use only the parts of their models that resulted to be useful for performing the constrained numerical optimization.

Binary strings were used to represent the points in the search space. This kind of representation, the use of lower and upper bounds for each point's coordinate and of the crossover genetic operator suggest classifying these evolutionary algorithms as genetic algorithms.

The three algorithms were tested with a suite of problems proposed and used by Michalewicz [Mic95], and used by Koziel and Michalewicz [KM99], and Runnarson and Yao [RY00], among other investigators. Some of the problems in this suite have feasible regions whose relative size is smaller than 10^{-5}. Four experiments for each of the algortithms, each one with a different set of parameters' values, some of them taken at random, were carried out. Thirty independent runs for each problem in the suite were performed. The three algorithms could find at least a point in the feasible region using 500,000 or less function evaluations for every of the runs. One of the algorithms could find consistently the best point for each of the thirteen problems, other could find consistently the best points of twelve of the problems and the last could find the consistently the best point in eleven out of the thirteen problems, using 500,000 or less functions evaluations.

Resumen

Se diseñaron e implementaron tres algoritmos evolutivos para resolver instancias del problema de programación no lineal. Algunas ideas tomadas del campo de los sistemas inmunes fueron adaptadas para incluirlas en los algoritmos. Estas ideas son principal-mente modelos de algunos mecanismos del más evolucionado de los sistemas inmunes naturales, el sistema inmune natural de los vertebrados con mandíbula (SIN), principalmente el de los seres humanos y el de los ratones. Estos mecanismos fueron la *recombinación somática* para construir los genes de las regiones variables de los receptores de los linfocitos B, y la *maduración de la afinidad* de los receptores de los linfocitos B.

No se pretendió simular ni imitar esos mecanismos del SIN, sino usar sólo las partes de sus modelos que resultaron ser útiles para llevar a cabo la optimización numérica restringida.

Se usaron cadenas binarias para representar lo puntos en el espacio de búsqueda. Este tipo de representación, el uso de límites superiores e inferiores para cada coordenada de los puntos y el uso del operador de cruzamiento, son razones para clasificar estos algoritmos evolutivos como algoritmos genéticos.

Estos tres algoritmos fueron probados con una colección de problemas propuestos por MIchalewicz [Mic95], y usados por Koziel y Michalewicz [KM99] y Runarsson y Yao [RY00], entre otros investigadores. Algunos de los problemas en esta colección tienen regiones factibles cuyos tamaños relativos son menores a 10^{-5}. Cuatro experimentos para cada uno de los tres algoritmos, cada uno con un conjunto diferente de valores para los parámetros, algunos tomados al azar, fueron realizados. Se realizaron treinta corridas independientes para cada problema de la colección. Los tres algoritmos pudieron encontrar por lo menos un punto en la región factible usando 500,000 o menos evaluaciones de la función para cada una de sus corridas. Uno de los algoritmos pudo encontrar consistentemente el mejor punto para cada uno de los trece problemas, otro pudo encontrar consistentemente el mejor de los puntos para doce de los problemas, y el último pudo encontrar consistentemente el mejor punto para once de los trece problemas, usando 500,000 o menos evaluaciones de cada función.

Capítulo I
Introducción

El problema de optimizar una función f: $R^n \rightarrow R$ sujeta a un conjunto de restricciones que dan por resultado que dentro del dominio de la función se forme una región factible (donde se satisfacen todas las restricciones) y otra no factible (donde una o más de las restricciones no se cumplen) se presenta en múltiples áreas del conocimiento ([BSS93]p.2). Al hecho de usar estrategias para llevar a cabo la optimización procurando que su resultado final sea el óptimo dentro de la región factible, se le llama *manejo de restricciones* [Coe99].

I.1 Planteamiento formal del problema.

Una manera formal de plantear este problema es ([Mic96]p. 121):

Optimizar $f(\boldsymbol{x}) \in R$, (I.1)

donde:

$\boldsymbol{x} \in S = [li_1, ls_1] \times \ldots \times [li_n, ls_n]$ (I.2)

sujeta a:

$g_i(\boldsymbol{x}) \leq 0$, para $i = 1, \ldots, p$ (I.3)

$h_i(\boldsymbol{x}) = 0$, para $i = p+1, \ldots, m$ (I.4)

En (I.3) se muestran las restricciones de desigualdad y en (I.4) las de igualdad. S (I.2) es llamado el espacio de búsqueda. li_i, ls_i representan los límites inferior y superior de la componente x_i de $\boldsymbol{x}$. La región factible, F, está en función de las restricciones:

$F = \{\boldsymbol{x} \mid g_i(\boldsymbol{x}) \leq 0, \text{ para } i = 1, \ldots, p \text{ y } h(x) = 0 \text{ para } i = p+1, \ldots, m\}$ (I.5)

El término *optimizar* en (1.1) se refiere a encontrar el óptimo:

$\boldsymbol{x}^* \in F \cap S$, que es tal que $f(\boldsymbol{x}^*) \succeq f(\boldsymbol{x})$ para $\boldsymbol{x} \in F \cap S$ (I.6)

$$\text{donde } f(\boldsymbol{x}^*) \succeq f(\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}^*) \geq f(\boldsymbol{x}) \text{ si se está maximizando} \\ f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}) \text{ si se está minimizando} \end{cases} \quad \text{(I.7)}$$

A este problema se le llama problema de programación no lineal (NLPP, *non linear programming problem*). No existe un algoritmo determinista (además de revisar en algún orden todos los puntos en S[1]) que garantice que se encontrará el óptimo para cualquier caso ([KM99]).

I.2 Objetivos. Los objetivos de este trabajo de investigación son:

1) Usar abstracciones de ideas surgidas en el estudio del sistema inmune para aplicarlas en el diseño de algoritmos genéticos para resolver el problema de programación no lineal.

2) Programar los algoritmos diseñados en algún lenguaje procedimental y probar su desempeño con colecciones de funciones reportadas en la literatura y que contengan algunos casos con regiones factibles de tamaño relativo pequeño (menor a 10^{-5}), para analizar el desempeño de los algoritmos en su búsqueda del óptimos globales restringidos de las funciones.

[1] Esta forma de resolver el problema es casi siempre inaceptable, cuando no imposible, debido al enorme tamaño de los espacios de búsqueda.

I.3 Hipótesis. La hipótesis de la que aquí se parte es que se pueden establecer similitudes aunque sean débiles, entre los problemas que resuelven algunos mecanismos del sistema inmune y el problema de programación no lineal, incluyendo casos con regiones factibles pequeñas; lo que permitirá el resolver estos problemas desarrollando algoritmos genéticos que incluyan rutinas basadas en ideas sobre estos mecanismos.

Comentarios. Nótese que el objetivo (1) no es simular ni imitar el comportamiento del sistema inmune natural de los animales superiores – los vertebrados con mandíbula – (SIN), sino el tomar ideas que han surgido de su estudio y tratar de adaptarlas para que ayuden en la solución del problema descrito (el NLPP) al incluirlas durante el desarrollo de algoritmos genéticos diseñados para resolver ese problema. En el objetivo (2) se habla además de tratar de resolver problemas con regiones factibles pequeñas y se especifica qué se considera aquí una región factible pequeña: toda aquélla cuyo tamaño relativo sea igual o menor a 10^{-5}. La hipótesis es que algunas ideas tomadas del estudio de los sistemas inmunes pueden ayudar a resolver instancias del NLPP incluyendo algunas con regiones factibles pequeñas. Este punto es importante, existen algoritmos propuestos [HY96] inspirados en mecanismos del sistema inmune que buscan y encuentran óptimos restringidos en regiones factibles cuyos tamaños relativos son mayores a 10^{-5}**.**

Los objetivos de un **m**odelo computacional del **s**istema **i**nmune[2] diseñado para resolver el problema de programación no lineal para el caso de regiones **f**actibles **p**equeñas (MSIFP), pueden ser distintos y hasta contraponerse con los del SIN. Por ejemplo, se puede establecer que uno de los principales objetivos del SIN es:
a) *mantener protegido contra una amplia gama de antígenos* (de patógenos, por supuesto) *al cuerpo al que defiende y del que forma parte*.

Para cumplir con este objetivo, una de las estrategias que usa el SIN es el hacer uso de la *reactividad cruzada*. Ésta consiste en que un receptor del SIN puede identificar no sólo a uno, sino a un conjunto de antígenos. El SIN debe usar la reactividad cruzada porque el número de receptores distintos que pueden estar presentes a la vez en el cuerpo de un individuo es limitado. Es lógico pensar que esta estrategia la fue adquiriendo el SIN durante la evolución genética de la especie de la que forma parte el individuo. En cambio, un MSIFP que hace evolucionar una población de receptores, tiene por objetivo:

b) *hacer evolucionar a los receptores para que progresivamente vayan reconociendo antígenos de microbios cada vez más agresivos hasta encontrar al más letal*.

En esta metáfora biológica los antígenos (de patógenos) son los puntos en la región factible y los antígenos de los microbios más letales corresponden a los óptimos globales[3] en la región factible. Es obvio que las presiones de selección que se deben ejercer sobre las poblaciones del MSIFP, o al menos algunas de ellas, actúan en direcciones distintas a las de aquellas presiones sufridas por las poblaciones del SIN. Por ejemplo, aquellas presiones de selección que hagan que los receptores no reconozcan a *lo propio* (los antígenos del propio organismo)

[2] Al lector no familiarizado con algunos términos usados en el área de los sistemas inmunes le será muy útil el leer el punto III.2 en el capítulo III para continuar con los puntos que siguen.

[3] Los o el punto al que le corresponde el mayor de los máximos o el menor de los mínimos de la función, dependiendo de si el problema de optimización es de maximización o minimización.

son útiles para la evolución de ambos tipos de receptores (los del SIN y los del MSIFP), sin embargo, las presiones de selección que hacen que un receptor sea afín a varios antígenos ayudarán a que el objetivo (a) del SIN se cumpla, pero no necesariamente a que se cumpla el objetivo (b) del MSIP. Para hacer que se cumpla este objetivo, se debe dar un mayor valor de aptitud a los receptores que reconozcan a los antígenos más letales (puntos que se acerquen más al óptimo) sin importar que tantos antígenos distintos reconozcan.

I.4 El contenido de la tesis.

En el capítulo II se habla sobre los antecedentes de algoritmos genéticos y el manejo de restricciones para resolver el problema de programación no lineal con algoritmos evolutivos. En el capítulo III se presentan los mecanismos del sistema inmune natural y se habla sobre modelos computacionales del sistema inmune. En el capítulo IV se describen a detalle los algoritmos genéticos propuestos en este trabajo de investigación. En el capítulo V se describen los experimentos realizados para probarlos usando una colección de funciones. En el capítulo VI se hace el análisis de los resultados obtenidos y se escriben las conclusiones. En el apéndice A se muestran las funciones de la colección usada en el capítulo V para probar los algoritmos, se muestran también los mejores puntos conocidos de cada función. El apéndice B muestra los mejores puntos obtenidos por los algoritmos propuestos. En el apéndice C se habla sobre otros problemas atacados con los algoritmos propuestos en este trabajo. Se muestran resultados a detalle para algunos de ellos. El apéndice D habla sobre algunos métodos tradicionales para resolver el problema de programación no lineal para funciones unimodales y por lo general derivables.

Capítulo II
Antecedentes

Los algoritmos genéticos (GAs, *genetic algorithms*) son métodos heurísticos de optimización de propósito general. Han sido usados para resolver problemas de *optimización sin restricciones*[1]. Es necesario hacerles modificaciones cuando se debe llevar a cabo una *optimización con restricciones*. Los GAs pueden optimizar funciones que presentan dificultades para su optimización. Algunas de las causas por la que es difícil optimizar una función son: tener alta dimensionalidad, presentar discontinuidades, tener múltiples máximos y mínimos (multimodalidad). Es por esto que se intuye que el esfuerzo que se invierta en realizar dichas modificaciones para resolver problemas de optimización con restricciones será redituable la mayoría de las veces. En el capítulo I se ha descrito el problema con restricciones para el que han sido diseñados los algoritmos de este trabajo, el NLPP.

II.1 La evolución darwiniana de las especies.

Los GAs se basan en la teoría de la evolución de las especies de Darwin, quien la publicó en su libro *The Origin of Species by Means of Natural Selection, or the Preservation of Favored Races in the Struggle for Life* [Dar60], cuya primera edición es de 1859. En ella se establece que las especies de seres vivos no permanecen inmutables, sino que sufren pequeños cambios de generación en generación. Son esos pequeños cambios los que hacen diferentes a algunos individuos del resto de sus congéneres. Si esas pequeñas diferencias hacen que sus poseedores obtengan alguna ventaja que les permita subsistir mejor y llegar a tener mayor descendencia, poco a poco, a lo largo de las generaciones, la especie se modificará en el sentido que estará integrada por cada vez más individuos que posean tales diferencias con respecto a la especie original. Por el contrario, aquellos cambios que impidan a sus poseedores subsistir mejor y tener suficiente descendencia, no prosperarán, y a lo largo de las generaciones, el número de individuos que posean tales diferencias será cada vez menor hasta que no quede ninguno. Este fenómeno de supervivencia de las características que hacen a los individuos de una población más aptos y la eliminación a través de las generaciones de aquellas características que hacen a los individuos menos aptos, fue llamado *selección natural* por Darwin.

Se puede entonces visualizar a una población de seres vivos como una generadora de múltiples opciones distintas entre sí por pequeñas diferencias, y la selección natural es la guía que dirige el cambio paulatino de la población a través de las generaciones[2], premiando con más hijos a los individuos que poseen las características que los hacen mejor adaptados a su entorno, y castigando con pocos hijos a los individuos menos adaptados.

En las poblaciones que se reproducen sexualmente, los hijos no son copias exactas de los padres. Entonces, en este tipo de poblaciones, simplemente por mezclar el material genético

[1] Los GAs fueron creados por J. Holland (los llamo "planes genéticos"), en su libro "Adaptation in Natural and Artificial Systems" [Hol75] describe a detalle qué son y propone una explicación de cómo funcionan, pero no habla de cómo optimizar problemas con restricciones. De hecho, no habla de usarlos específicamente como métodos de optimización.

[2] Ha habido y sigue habiendo discusión sobre la fuerza conductora de la evolución. Hay autores (Kimura (1968), King y Jokes (1969), Kimura (1968, 1977, 1979, 1983), Kimura y Otha (1973)) que cuestionan que sea la selección natural la fuerza conductora de la evolución. "Una conclusión importante de la teoría neutral es que la diversidad genética es en gran medida causada por la deriva genética aleatoria, implicando que la diversidad genética observada en las poblaciones es un fenómeno transitorio: las mutaciones son introducidas al azar y se fijan o se pierden únicamente debido a fuerzas estocásticas". El párrafo entrecomillado ha sido copiado de [Neu01], las menciones a las referencias también han sido copiadas de allí .

de los padres, los hijos ya presentarán variaciones con respecto a ellos. Se puede concluir que la reproducción sexual en sí misma ya es una fuente de variedad. Por supuesto, entre más diferentes sean entre sí los padres, y por tanto sus materiales genéticos, mayor diferencia habrá de los hijos con respecto a ellos. Por el contrario, entre más parecidos sean entre sí los padres más se parecerán sus hijos a ellos.

En el caso de individuos que no se reproducen sexualmente, los hijos son, por lo general, copias exactas del padre. Sin embargo, a veces se producen hijos distintos a él. Esto es debido a que los procesos para copiar la información del padre a los hijos presenta defectos y la copia no es exacta ([Bro99]p.2,3). También se da el caso de que el material genético cambie antes de la reproducción debido a agentes tales como substancias químicas ([Bro99]p.2,3). A las alteraciones que sufre el material genético, antes o durante la reproducción, se les da el nombre de *mutaciones*. También en la reproducción sexual hay mutaciones.

Resumiendo, las fuentes de las variaciones de los individuos son la combinación del material genético de los miembros de una población y la mutación de la información recibida por los hijos de los padres. La selección dirige la evolución de una población eligiendo para subsistir a sus individuos más aptos.

El sistema inmune de los animales vertebrados sufre este tipo de evolución a dos niveles. A nivel genético porque el sistema inmune, al ser parte integral de cada individuo, evoluciona con la especie; y a nivel somático, porque dentro de cada individuo aspectos tales como la maduración de la afinidad de los linfocitos B con la consiguiente multiplicación (expansión monoclonal) de aquéllos que embonan con gran afinidad con los antígenos y la muerte inducida (apoptosis) de los que no, se asemeja a la lucha por la supervivencia de los individuos de diversas especies. Feitas y Rocha [FR00] sostienen que "el conflicto de los intereses de supervivencia entre los diferentes tipos de células dan lugar a patrones de interacciones que imitan el comportamiento de sistemas ecológicos complejos".

II.2 Los algoritmos genéticos.

Los algoritmos genéticos se basan en la Teoría de la Evolución de las Especies de Darwin para resolver problemas de optimización. A continuación se muestra la estructura de un algoritmo genético tradicional (esta forma de expresar un GA tradicional se inspira en la figura 4.1 de [Mic96]p.62):

1 **inicio**
2 Obtener de manera aleatoria una población $P^0 = (p_1, p_2, \ldots, p_N)$;
3 Obtener la aptitud de cada uno de los miembros de P^0;
4 Guardar al individuo más apto en el registro *MejorIndividuo*;
5 $t = 0$;
6 **mientras**(no se cumpla criterio de terminación)
7 {
8 $P^{1/2}$ = *Selección* con reemplazo y de manera aleatoria favoreciendo a los más aptos de los *N* individuos de la población P^t;
9 $P^{1/2'}$ = Formación de manera aleatoria de parejas de la población $P^{1/2}$, *cruzamiento* de los miembros de cada pareja entre sí con una probabilidad p_c;

10 P^{t+1} = *Mutación* de cada una de las partes que constituye a cada elemento de la población $P^{1/2'}$ con una probabilidad p_m;

11 $t = t + 1$;

12 Obtener la aptitud de cada uno de los miembros de P^t;

13 Si alguno de los individuos de P^t es mejor que el individuo en *MejorIndividuo*, ese individuo debe almacenarse en *MejorIndividuo*;

14 }

15 **mostrar** al individuo almacenado en *MejorIndividuo*;

16 **fin.**

Los elementos de una población en los algoritmos genéticos son representaciones de posibles soluciones al problema de optimización. La bondad de cada una de estas posibles soluciones se mide por medio de la función de aptitud: a mayor aptitud, mejor es la solución a dicho problema.

Tales representaciones son cadenas de símbolos en los algoritmos genéticos tradicionales. Dichas cadenas son de tamaño finito y están escritas con caracteres de un alfabeto también finito. Se acostumbra usar alfabetos de dos símbolos, entonces las cadenas escritas con estos alfabetos son binarias.

En la línea 6 se dice que mientas no se cumpla un criterio de terminación se repite el ciclo que va de la línea 7 a la línea 14. A cada repetición de este ciclo se le llama *generación*. Dicho criterio de terminación puede ser el que el número de generaciones, medido por el contador *t*, alcance un valor previamente fijado por el usuario. O podría ser que se analizara los elementos de la población y cuando todos o un porcentaje de ellos sean muy parecidos entre sí, salir del ciclo. Se puede establecer otros criterios de terminación.

En la línea 8 se indica que se forma una población intermedia, $P^{1/2}$, copiando con reemplazo los elementos de la población P^i hasta que haya *N* elementos en $P^{1/2}$. Copiar con reemplazo significa que se puede copiar más de una vez a cualquiera de los elementos de P^i. Favoreciendo a los más aptos significa que la probabilidad de copiar en $P^{1/2}$ a un elemento o individuo cualquiera en P^i debe ser mayor a la probabilidad de copiar a otro menos apto que él.

En la línea 9 se indica que se debe formar parejas de cadenas de manera aleatoria. Por cada pareja se debe obtener un número aleatorio $r \in [0,1]$ de distribución uniforme; si $r \leq p_c$, deben combinarse o cruzarse entre sí las cadenas que forman la pareja. El término cruzamiento o combinación se refiere a intercambiar fragmentos de cada una de las cadenas pertenecientes a las mismas posiciones en cada una de ellas. La probabilidad de cruzamiento es fijada por el usuario; su valor es alto, por lo general en el intervalo [0.6, 0.95] ([Bäc96] p. 114).

En la línea 10 se habla de aplicar el operador de mutación. Cuando la representación es con cadenas binarias pertenecientes al conjunto $\{0,1\}^l$ (cadenas escritas con el alfabeto {0, 1} y de longitud *l*), esto significa el someter al bit *j-ésimo* de la *i-ésima* cadena de la población $P^{1/2'}$ a cambiar su contenido con una probabilidad $p_m^{(i,j)}$ por el complemento de este contenido. Por

ejemplo, si el contenido de un bit es 0 se cambiará por 1. Por cada bit se debe obtener un número aleatorio $r^{(i,j)} \in [0,1]$ de distribución uniforme, si $r^{(i,j)} \leq p_m^{(i,j)}$, se cambia (muta) el contenido del bit en cuestión por su complemento. Las probabilidades de mutación (o la forma de calcularlas) $p_m^{(i,j)}$ deben ser fijadas por el usuario. Cuando las probabilidades de mutación son iguales entre sí, la mutación es llamada *mutación uniforme*. Estas probabilidades por lo general están en el intervalo [0.001, 0.01] ([Bäc96]p.113).

II.2.1 El Algoritmo Genético Simple.

El algoritmo genético simple (SGA, *simple genetic algorithm*) es aquel algoritmo genético de la forma mostrada en la sección II.2 y que utiliza : 1) selección proporcional a la aptitud, 2) cruzamiento en un punto y 3) mutación uniforme ([Gol89]pp. 10-14). Selección proporcional a la aptitud significa que la probabilidad de que un individuo sea copiado a la población intermedia es directamente proporcional a su aptitud; por ejemplo, si la aptitud de un individuo *a* es el doble de la de un individuo *b*, entonces la probabilidad de que *a* sea copiado en la población intermedia es del doble de la del individuo *b*. Cruzamiento en un punto significa que dadas dos cadenas $a = a_1 a_2 \ldots a_l$ y $b = b_1 b_2 \ldots b_l$ se obtiene un número aleatorio $r \in \{2, 3, \ldots, l\}$ y se intercambian entre las dos cadenas los símbolos que van de las posiciones *r* a la *l*, dando por resultado dos cadenas hijas: $a_1 a_2 \ldots a_{r-1} b_r b_{r+1} \ldots b_l$ y $b_1 b_2 \ldots b_{r-1} a_r a_{r+1} \ldots a_l$. Si las subcadenas $a_r \ldots a_l$ y $b_r \ldots b_l$ tienen por lo menos un símbolo distinto entre ellas, y las cadenas $a_1 \ldots a_{r-1}$ y $b_1 \ldots b_{r-1}$ son distintas entre sí, entonces las cadenas hijas serán distintas a las cadenas padres. El concepto de *mutación uniforme* se ha descrito en el último párrafo de la sección II.3.

Como se indica en las líneas 4, 13 y 15 del algoritmo general mostrado en II.1.1, se debe hacer un seguimiento del mejor individuo encontrado a lo largo de la corrida. Es necesario hacer tal seguimiento cuando se aplique el SGA (o cualquier otro GA que no use el *elitismo*, que se discutirá más abajo), ya que el mejor individuo encontrado hasta una generación *t* dada puede perderse al aplicarse los operadores de mutación y cruzamiento o podría incluso no ser seleccionado para ser copiado ni una sola vez en la población intermedia, ya que la selección no es determinista y por tanto existe una probabilidad, aunque baja, de que esto ocurra (para evitar esto último se puede aplicar el *muestreo estocástico universal* [Mit96]pp. 166,167, con él se garantiza que el mejor individuo será siempre seleccionado).

En pocas palabras, el mejor individuo puede perderse de una generación a otra, por lo que es necesario llevar un registro del mejor individuo.

II.2.2 El elitismo.

Consiste en preservar al mejor individuo de una población P^t poniendo directamente una copia de él en la población P^{t+1}. Si se aplica el elitismo en cada generación del GA, la última población siempre contendrá al mejor individuo encontrado a lo largo de todo el proceso.

II.2.3 El teorema del esquema.

Holland, el creador de los algoritmos genéticos, proporciona una explicación de por qué funciona el SGA ([Hol92] Cap.4 *Schemata*). Propone que en las cadenas que representan las soluciones, ciertos patrones, a los que él llama esquemas, tienden a proliferar o a desaparecer

en la población de cadenas según pertenezcan a cadenas aptas o no y según sea su grado de dispersión en la cadena. Dice que las cadenas que presentan un patrón dado pertenecen al esquema que describe dicho patrón. Así, un esquema puede interpretarse como un subconjunto de cadenas que comparten una característica particular. Por ejemplo, sean dos cadenas de longitud $l = 10$ escritas con el alfabeto {0, 1}:

Cadena 1 1 0 0 1 1 1 0 1 0 0
posición 1 2 3 4 5 6 7 8 9 10

Cadena 2 1 0 1 1 0 0 0 1 0 1
posición 1 2 3 4 5 6 7 8 9 10

Algunos de los esquemas o conjuntos de cadenas a los que pertenecen estas dos son:
conjunto de todas las cadenas en $\{0, 1\}^{l=10}$ que
a) empiezan con 1,
b) tienen el valor 0 en la segunda posición y 1 en la cuarta,
c) tienen el valor 1 en la posición 4,
d) tienen el valor 1 en la primera posición, 0 en la segunda posición, 0 en la séptima posición, 1 en la octava y 0 en la novena,
e) tienen el valor 1 en la primera posición y 0 en la novena posición,
f) tienen el valor 1 en la primera posición, 1 en la cuarta y 0 en la novena.

Estos esquemas o conjuntos pueden expresarse con cadenas de símbolos de la misma longitud ($l = 10$), escritas con el alfabeto: {0, 1, *}. Donde * es en realidad un metasímbolo, significa que en la posición donde se encuentre puede ir un 0 o un 1. Así el conjunto del inciso (a) se representa:

1 * * * * * * * * *
1 2 3 4 5 6 7 8 9 10
y los de los demás incisos:
b)
* 0 * 1 * * * * * *
1 2 3 4 5 6 7 8 9 10
c)
* * * 1 * * * * * *
1 2 3 4 5 6 7 8 9 10
d)
1 0 * * * * 0 1 1 *
1 2 3 4 5 6 7 8 9 10
e)
1 * * * * * * * 0 *
1 2 3 4 5 6 7 8 9 10
f)
1 * * 1 * * * * 0 *
1 2 3 4 5 6 7 8 9 10

Cada una de estas cadenas representan un esquema. Nótese que se le acostumbra llamar *esquema* tanto al conjunto de cadenas como a la representación de dicho conjunto.
En la desigualdad (II.1) se estima el número de cadenas de un esquema que estarán presentes en la siguiente población de un GA ([Gold89]pp. 32-33, [Whi94]):

$$m(H,t+1) \geq m(H,t)\frac{\varphi(H)}{\bar{\varphi}} \cdot (1 - p_c \frac{\delta(H)}{l-1}) \cdot (1-p_m)^{o(H)} \qquad \text{(II.1)}$$

donde

H se usa para denominar a un esquema,

$m(H,t)$ es el número de cadenas en la población número *t* que pertenecen al esquema H,

$\varphi(H)$ es la aptitud promedio de las cadenas de la población número *t* que pertenecen al esquema *H*,

$\bar{\varphi}$ es la aptitud promedio de todas las cadenas de la población número *t*,

p_c es la probabilidad de cruzamiento,

p_m es la probabilidad de mutación,

$\delta(H)$ es la longitud definitoria del esquema *H*. Se define como la resta de la posición del símbolo *definido* más a la derecha en la representación del esquema menos la posición del símbolo *definido* más a la izquierda en la representación del esquema. La posición de los símbolos se cuenta de izquierda a derecha de la representación del esquema.
Ejemplo 1:
Sea *H* un esquema para cadenas binarias de longitud $l = 10$:

H =	*	0	0	*	1	*	1	0	1	*
Posición	1	2	3	4	5	6	7	8	9	10

Símbolo *definido* en la representación del esquema es un símbolo diferente del metasímbolo: *. La longitud definitoria de este esquema es: $\delta(H) = 9-2 = 7$.
Ejemplo 2:

H =	1	*	*	*	1	*	0	*	*	0
Posición	1	2	3	4	5	6	7	8	9	10

$\delta(H) = 10-1 = 9$.
Ejemplo 3:

H =	*	*	*	*	*	*	0	*	*	*
Posición	1	2	3	4	5	6	7	8	9	10

$\delta(H) = 7-7 = 0$.

$o(H)$ es el orden de un esquema. El orden de un esquema se define como el número de símbolos definidos que contiene. El orden del esquema del ejemplo 1 es $o(H) = 6$, el del ejemplo 2 es $o(H) = 4$, y el del ejemplo 3 es $o(H) = 1$.

Obsérvese la desigualdad II.1, el número de cadenas pertenecientes al esquema *H* en la siguiente población, $m(H,t+1)$, será mayor al de la población presente, $m(H,t)$, si la relación $\frac{\varphi(H)}{\bar{\varphi}}$ es mayor que uno , y si el factor $(1 - p_c \frac{\delta(H)}{l-1})$ y el factor $(1-p_m)^{o(H)}$ tienden a parecerse a uno.

El término $p_c \cdot \delta(H)/(l-1)$, representa la probabilidad de destrucción del esquema *H* debido a los cruzamientos entre las cadenas. Obsérvese que entre mayor sea la longitud definitoria, mayor será la posibilidad de que el esquema sea destruido. Tómese el caso del ejemplo 2, en él la longitud definitoria tiene el máximo valor posible: $p_c \cdot \delta(H)/(l-1) = p_c \cdot 9/9 = p_c$. Como la probabilidad de cruzamiento tiene valores altos, el factor $(1 - p_c \frac{\delta(H)}{l-1})$ tenderá a parecerse a cero. Ahora tómese el caso del ejemplo 3, en él la longitud definitoria tiene su mínimo valor posible: cero; entonces el factor en cuestión valdrá uno.

El factor $(1-p_m)^{o(H)}$ representa la probabilidad de que subsista el esquema H al operador de mutación. El término $(1-p_m)$ es siempre menor que uno, por lo que entre mayor sea $o(H)$, más tenderá a parecerse a cero este factor.

De lo anterior se concluye que para que el número de cadenas de un esquema sea mayor en la población inmediata siguiente se debe cumplir que:
a) La aptitud promedio de las cadenas que pertenezcan al esquema y estén en la población sea mayor que el promedio de la aptitud de todas las cadenas de la población.
b) Que la longitud definitoria del esquema sea pequeña.
c) Que el esquema tenga pocos símbolos definidos.
A un esquema que cumpla con estas condiciones se le llama *bloque constructor*. Las conclusiones obtenidas arriba reciben el nombre de *teorema del esquema*.

II.2.4 La hipótesis de los bloques constructores.
Esta hipótesis consiste en suponer que es la yuxtaposición de los bloques constructores es el mecanismo por el cual se va acercando el SGA al óptimo ([Mic96]p.53). Hay dos impedimentos para que esto ocurra: la existencia de *funciones engañosas* ([Kur99]p.181-184) y la *correlación espuria* ([Mit96]p.131,132).

II.2.5 Las funciones engañosas.
Estas son las funciones que deben ser optimizadas y que desalientan la formación de bloques constructores asignándoles una baja aptitud a las cadenas que los contienen, esto provoca que proliferen bloques que aparentemente son mejores pero cuya yuxtaposición no forma la cadena óptima.

Ejemplo. Sea la función:

Cadena binaria b	$\varphi(b)$
000	5
001	2
010	2

011	0
100	2
101	0
110	0
111	10

En esta función el óptimo está en 111. Se favorece que aparezca un solo 1 en las cadenas, pero se desalienta la formación de cadenas con dos 1s porque a todas ellas se les da una calificación menor a la de aquellas cadenas que solamente tienen un 1. Es más ventajoso para una cadena que tiene un solo 1 el cambiarlo por un 0, pero de esta manera se aleja del óptimo global.

Una forma de disminuir lo engañoso de una función cuando se hace optimización numérica es codificar las coordenadas de cada punto en el espacio de búsqueda en código Gray. El código Gray tiene la característica de que dos números enteros sucesivos cualesquiera representados con él, difieren entre sí únicamente por un bit ([Nas77]p. 86). Por ejemplo, en código binario pesado el número 7 se representa: 0111 y el número 8: 1000. Nótese que aunque los dos enteros son vecinos el uno del otro, difieren entre sí en cuatro bits. En código Gray se representan: 0100 y 1100 respectivamente, con esta forma de representarlos difieren entre sí únicamente por un bit. Si para una función dada la solución representada con el número 8 es mejor que aquélla representada con el número 7 o viceversa, será más fácil para un GA saltar de una solución a otra si cada punto del espacio de búsqueda se representan en código Gray.

II.2.6 La correlación espuria.

Aunque la función que se pretenda optimizar no sea engañosa, existe otro problema que obstaculiza el que un GA se acerque al óptimo: la *correlación espuria*. Melanie Mitchell la estudió junto con otros investigadores y la describe en ([Mit96] pp. 131-133). Ésta consiste en que al GA le cuesta trabajo el "darse cuenta" cual el es *exactamente* el esquema que hace que una cadena tenga una buena aptitud. Entonces ocurre que el esquema tiende a proliferar de generación en generación pero "arrastra" algunos bits que no son parte de él. Esto trae por consecuencia que los bits que están de más no permitan que se yuxtapongan los bloques constructores entre sí. Se tiene que gastar más generaciones para que el GA "se dé cuenta" de esta situación.

La misma Mitchell da recomendaciones sobre cómo evitarla ([Mit96]pp. 137-138):
1) Se debe evitar que todas o la mayoría de las cadenas de una población tengan fijo un mismo valor en una posición dada, para ello el tamaño de la población y la mutación deben ser lo suficientemente grandes, y la selección debe ser lo suficientemente lenta.
2) Para ir preservando los esquemas útiles (bloques constructores) la selección debe ser lo suficientemente fuerte pero también lo suficientemente lenta para prevenir que se presente en gran medida la correlación espuria.
3) El cruzamiento debe ser rápido, el tiempo debe gastarse en encontrar los buenos esquemas.

Reconoce que estos mecanismos no son mutuamente compatibles (por ejemplo, el preservar esquemas se contrapone con la probabilidad de mutación grande) y que por ello deben balancearse cuidadosamente.

II.2.7 El SGA tiene pobre desempeño en problemas complejos.

Hay casos en los que el SGA no se comporta bien en la búsqueda del óptimo. Por ejemplo, en [LY00] se comenta "el algoritmo genético canónico estándar (SGA) que involucra los operadores de selección, reproducción, cruzamiento y mutación tiende a quedar corto al desempeño que se desearía de un algoritmo de búsqueda. Para resolver este problema , es común el mezclar técnicas basadas en los GAs con procedimientos deterministas de búsqueda local.". En [PGL02] se menciona "los GAs trabajan muy bien únicamente para problemas donde los bloques constructores se localizan de manera concentrada en las cadenas que representan las soluciones. En problemas donde los bloques constructores se distribuyen a lo largo de las soluciones los GAs simples experimentan un desempeño muy pobre. Es por eso que ha crecido el interés en métodos que aprenden la estructura de un problema sobre la marcha y utilizan esta información para asegurar un apropiado mezclado y crecimiento de los bloques constructores.".

Estos comentarios coinciden con los resultados obtenidos por Mitchell y colaboradores cuando diseñaron funciones pensadas para que el SGA las resolviera fácilmente, a tales funciones las llamaron "caminos reales". Para su sorpresa el Escalador por Mutación Aleatoria (RMHC por sus siglas en inglés, *random mutation hill climber*) llegó al óptimo de uno de los caminos reales con mucho menos iteraciones (aproximadamente un décimo) de las requeridas por una variación del SGA. Las sugerencias que Mitchell propone para mejorar el desempeño de los GAs son las que se presentan en II.3.1.5.

Es por esta razón (el pobre desempeño comprobado en la práctica del SGA) que en este trabajo se utilizaron GAs que difieren en varios puntos de él (ver capítulo IV).

II.2.8 Los parámetros que requiere un GA.

Hay varios tipos de GAs. El tipo de un GA depende, entre otras cosas, de los operadores de selección, cruzamiento y mutación que use (ver una taxonomía de los GAs en [Kur99]pp. 160-163). Dependiendo del tipo de GA se deben fijar ciertos parámetros antes de usarlo, por ejemplo, en el SGA, como en casi todos los GAs se debe fijar : 1) el tamaño de la población, 2) la probabilidad de cruzamiento, 3) la probabilidad de mutación y , si el problema es de una naturaleza tal que lo requiera, 4) el tamaño de las cadenas binarias, como es el caso de problemas de optimización numérica, en los que dependerá de la precisión que se quiera usar (a mayor precisión mayor longitud de las cadenas).

Es importante recalcar este punto ya que muchas veces el desempeño de un GA (convergerá o no al óptimo y en cuantas iteraciones) depende, además de su tipo, de los valores de estos parámetros. Las modificaciones que son llevadas a cabo en un GA para que optimice funciones restringidas involucran, por lo general, la definición de más parámetros, cuyos valores deberán también afinarse para obtener buenos resultados.

II.3 La evolución lamarckiana.

Lamarck postuló la hipótesis de que la evolución de las especies se debía a que las características o habilidades adquiridas por los padres durante su vida eran transmitidas a los hijos ([Lam1809]pp. 15-16, por ejemplo; de hecho en múltiples partes de su obra, Lamarck habla sobre la transmisión a través de las generaciones de las características adquiridas a lo largo de la vida de los individuos). Por ejemplo, según esta teoría, si un herbívoro estira el cuello cotidianamente para poder alcanzar las hojas de un árbol, sus hijos nacerán con el

cuello un poco más largo que los hijos de otros herbívoros que no tienen tal comportamiento. Si esta nueva generación tiene ese mismo hábito, los cuellos de sus hijos serán más largos. Si este comportamiento se repite durante varias generaciones, habrá para entonces animales con cuellos muy largos. Esto explicaría la existencia de las jirafas ([Lam1809]pp. 257-258).

Se ha comprobado que no es así como evolucionan las especies, sino como lo postuló Darwin (ver II.1). Ya desde el siglo XIX había argumentos fuertes en contra de esta teoría de la evolución (en 1893 August Weissmann arguye que no hay forma en que la información adquirida por las células somáticas sea transferida a las células germinales), de manera tal que ya desde finales de ese siglo, la teoría de Lamarck de la evolución (postulada a inicios del siglo XIX) ya estaba desacreditada.

Si bien la teoría de Lamarck desde el punto de vista biológico ya no es aceptada, su uso en programas de cómputo tiene buen éxito. Así que cuando el objetivo que se persigue al diseñar un algoritmo no es el imitar la evolución de alguna especie biológica, el uso de las ideas de la teoría de Lamarck resulta provechoso.

II.3.1 La escalada de montes por mutación aleatoria (RMHC) se puede interpretar como evolución lamarckiana.

La forma en la que Lamarck propone que se lleva a cabo la evolución de los seres vivos, en especial los animales, es más directa en el sentido que se enfoca en la modificación de una parte de la estructura del cuerpo del animal en base al uso que éste haga de ella. Como, según él, las características se heredan de generación en generación, si un animal tiende a utilizar con menos frecuencia una parte de su cuerpo, esa parte se irá atrofiando hasta después de muchas generaciones terminará por desaparecer. Si, por el contrario, la usa con más frecuencia, tenderá a persistir a lo largo de las generaciones y además se especializará cada vez más para servir mejor en las tareas para las que se use.

Es el que un individuo pueda transmitir la habilidad adquirida (y ventajosa para la subsistencia) durante su existencia a sus descendientes, lo que caracteriza a la evolución lamarckiana. El RMHC se asemeja a ella: un individuo (una cadena de símbolos) experimenta si un pequeño cambio a su estructura lo hace más apto, en caso afirmativo, transmite ese cambio a su hijo (la misma cadena con ese cambio), de lo contrario experimenta otro pequeño cambio en su estructura y se repite el ciclo. Se termina después de un número predeterminado de pequeños cambios.

II.4 El efecto de Baldwin.

La evoluciones darwiniana y la lamarckiana son dos extremos de la forma de ver cómo afecta el aprendizaje a la evolución de las especies. La evolución darwiniana se basa en la selección de los individuos más aptos. Como se considera que todas las características que hacen a los individuos más aptos[3] están codificadas en los genes, la evolución darwiniana sólo toma en cuenta la evolución de éstos para explicar la evolución de las especies (*i.e.*, no toma en cuenta lo aprendido por los individuos durante su vida). En el otro extremo se encuentra la evolución

[3] De hecho, considera que todas las características del fenotipo del individuo están codificadas en los genes.

lamarckiana, que, en pocas palabras, sostiene que lo aprendido se transmite a los hijos implícitamente (esto equivale a sostener que de alguna forma lo aprendido se codifica en los genes y se transmite a los hijos).
Ya se ha comentado arriba que la evolución lamarckiana ya estaba desacreditada desde finales del siglo XIX, a diferencia de la darwiniana que para ese tiempo ya era bien aceptada. Entonces una teoría que explicara cómo el aprendizaje podría afectar la evolución de las especies debería sustentarse en argumentos distintos a los de la teoría de Lamarck. Baldwin [Bal1896] propone una teoría sobre cómo el aprendizaje puede afectar la evolución de las especies dentro del marco de la teoría de la evolución de Darwin. A esta teoría que propone Baldwin se le acostumbra llamar el *efecto Baldwin*. En pocas palabras, lo que Baldwin sostiene es que una habilidad *aprendida* por los individuo de una especie durante su vida y que les ayude en la lucha por la subsistencia, después de varias generaciones puede terminar por aparecer codificada en el material genético de los individuos de esa especie debido a que la evolución genética terminó por *encontrar* esa característica genética que se traduce en la característica fenotípica favorable. Obsérvese que no es el paso implícito a la siguiente generación de lo aprendido por los padres lo que hace que aparezca en el genoma de los descendientes la buena característica, sino la *compra de tiempo*, por llamarla de alguna manera, para permitir que la evolución genética encuentre por sí misma esa característica. Esta *compra de tiempo* la propicia la habilidad aprendida, ya que permite a los individuos que la poseen tener una mayor probabilidad de subsistir y así tener una mayor descendencia sobre los que no la tienen.

Turney [Tur96] hace varias observaciones interesantes sobre el efecto Baldwin. Hace notar que no siempre el aprendizaje es superior a las habilidades heredadas, como lo es el instinto, a diferencia de lo que los seres humanos tienden a creer. Él sostiene que más bien es una cuestión de contrabalanceo, por ejemplo, el aprendizaje requiere de varios dispositivos, como lo son sensores finos, dispositivos para procesar la información y efectores precisos. Los dispositivos para procesar la información (cerebros) consumen energía (por ejemplo, el cerebro del humano tiene un gasto de energía como el de un foco de 100 watts), el aprendizaje requiere de tiempo, en el que el individuo puede ser vulnerable en lo que adquiere el suficiente dominio de la habilidad; además se requiere tener contacto con la fuente de la que se adquirirá el conocimiento. Pero es muy útil si el entrono sufre cambios rápidos. Si el entorno permanece estático, entonces el instinto será una opción más barata y eficiente.

French y Messenger [FM94] realizaron experimentos en los que estudiaron la aparición del efecto de Baldwin. Concluyeron que no siempre se presenta este efecto, depende de la *plasticidad fenotípica* del individuo (la capacidad que tiene un individuo de modificar su fenotipo), el grado en el que se ve incrementada su aptitud al adquirir la habilidad, la facilidad de adquirirla.

El cerebro no es la única parte del cuerpo de un animal superior que aprende, también lo hace el sistema inmune ya que algunos de sus componentes sufren una evolución somática, así que el efecto de Baldwin se puede presentar también con respecto a él. Hightower et al. [HFP96] han estudiado este efecto en simulaciones computacionales del sistema inmune, concluyeron que sólo bajo ciertas circunstancias se presenta. Perelson *et al.* [PHF96], también reportan la aparición de este efecto para ciertas condiciones durante su experimentación con un programa diseñado por ellos que simula al sistema inmune.

II.5 Manejo de restricciones con algoritmos genéticos.
El término manejo de restricciones se refiere a cómo debe un método de optimización (en este caso los GAs) proceder para optimizar una función cuyo espacio de búsqueda tiene puntos factibles y no factibles, es decir, $S \not\subset F \cap S$, para encontrar el mejor punto en $F \cap S$.

II.5.1 Antecedentes de manejo de restricciones[4].
A menos que se especifique otra cosa, en esta subsección se supondrá que se desea resolver un problema de minimización todas las restricciones deben estar enunciadas como (ec. I.3) y (ec. I.4) y por comodidad a todas se les llamará ρ y se denominará $\mathcal{R}$ al conjunto de todas las restricciones : $\mathcal{R} = \{\rho_1, \rho_2, ..., \rho_m\}$.

Existen varios acercamientos sobre cómo manejar las restricciones con GAs, [Coe99] las clasifica en:

- Uso de funciones de penalización.
- Mantenimiento de una población factible por representaciones y operadores genéticos especiales.
- Separación de objetivos y restricciones.
- Métodos híbridos.
- Nuevas estrategias

A continuación se mostrarán algunos ejemplos.

II.5.1.1 Uso de funciones de penalización.
Pena de muerte.
Esta es la forma más sencilla de manejo de restricciones, si una solución potencial no cae en la región factible simplemente es desechada y se vuelve a obtener otra. Esto sólo es eficiente si la relación $F \cap S / S$ es grande, de lo contrario la búsqueda tiende a degenerar en una búsqueda aleatoria.

Satisfacción de las restricciones por su número.
Kuri y Villegas [KV98] proponen el usar una función de aptitud que tome en cuenta el número de restricciones satisfechas :

$$\phi(\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}) & \text{si } \boldsymbol{x} \text{ es factible} \\ K - \sum_{i=1}^{s} \frac{K}{m} & \text{si } \boldsymbol{x} \text{ no es factible} \end{cases}$$

donde
K es un número positivo muy grande
s es el número de restricciones satisfechas.

Para poder aplicar este método es necesario conocer puntos factibles desde el principio.

[4] Para la elaboración de esta subsección ha sido de gran utilidad el artículo de Coello "A Survey of Constraint Handling Techniques used with Evolutionary Algorithms" [Coe99].

Penalización estática y dinámica.
Consiste en convertir el problema restringido en uno no restringido agregando un término de penalización a la función que se desea minimizar :

$$\phi(\boldsymbol{x}) = f(\boldsymbol{x}) + P(\boldsymbol{x})$$

donde

$$P(\boldsymbol{x}) = \begin{cases} >0 & \text{si } \boldsymbol{x} \text{ no es factible} \\ 0 & \text{si } \boldsymbol{x} \text{ es factible} \end{cases}$$

$P(\boldsymbol{x})$ es el término de penalización, hay investigadores que lo han propuesto estático (que no cambia conforme se avanza en la aplicación del GA) y otros que lo han propuesto dinámico.

El problema se convierte ahora en: minimizar la función $\phi(\boldsymbol{x})$ que no está sujeta a restricciones. La idea es "castigar" a los puntos que no estén en la región factible para que el operador de selección los retire de la población y se vaya favoreciendo la creación de poblaciones en la región factible.

Un método de penalización estático es el propuesto por Homaifar *et al.*[HQL94] :

$$P(x) = \begin{cases} 0 & si\ \boldsymbol{x} \in F \\ \sum\limits_{\substack{\rho_i \in R \\ \rho_i\ violada}} \rho_i^2(\boldsymbol{x}) \cdot Q_{i,j} & si\ \boldsymbol{x} \notin F \end{cases}$$

donde
$Q_{i,j}$ es el factor de penalización, este factor depende del nivel j de penalización. Esto implica que se deben proponer l_i+1 límites para l_i niveles de penalización además de l_i valores para Q, un total de $2 \cdot l_i + 1$ parámetros para la *i-ésima* restricción. En total habrá entonces $m \cdot (2 \cdot l_i + 1)$ parámetros. Pone Coello [Coe99] un ejemplo : si se tiene un problema de seis variables y se fijan dos niveles de penalización por cada una entonces deberán fijarse 30 parámetros. Son demasiados parámetros los que pide este método para el manejo de las restricciones.

Joines y Houck [JH94] proponen un método de penalización dinámica :

$$P(\boldsymbol{x}, t) = (C \cdot t)^{\alpha} \sum_{\substack{\rho_i \in R \\ \rho_i \text{ violada}}} \left|\rho_i(\boldsymbol{x})\right|^{\beta}$$

donde
t es el número de iteración
C, α, β son constantes definidas por el usuario. Coello comenta (*op. cit.*) que la calidad de las soluciones encontradas es muy sensible a cambios en los valores de los parámetros. Agrega que Michalewicz reporta que valores propuestos por Joines y Houck. (C= 0.5, β=1, o 2) casi siempre provoca convergencia prematura en otros ejemplos, además la técnica o converge a un punto no factible o converge a uno factible pero lejos del óptimo.

Otra forma de proponer penalizaciones dinámicas es usando el recocido simulado. Se propone una función de temperatura T que depende del número de iteraciones: $T = T(t)$, la temperatura

disminuye conforme aumenta el número de iteraciones realizadas. Carlson *et al.* [CSBA] proponen la función ϕ siguiente para un problema de **maximización**:

$$\phi(\boldsymbol{x}) = e^{-\frac{M}{T}} \cdot f(\boldsymbol{x})$$

donde
M es una medida del grado de violación de las restricciones, vale cero si ninguna es violada.
T es una función de la temperatura cuyo orden es $T = O\left(\frac{1}{\sqrt{t}}\right)$

La idea es iniciar el método con una temperatura alta, así el exponente del factor exponencial tiende a cero y este factor tenderá a 1 por abajo. De esta forma, al inicio de la aplicación del método la $\phi(\boldsymbol{x})$ será muy parecida a *f*(x) aunque $\boldsymbol{x}$ viole algunas restricciones, pero cuando se hayan realizado más iteraciones, el factor exponencial tendrá un valor cada vez más pequeño cuando no se cumpla alguna restricción. En otras palabras, la penalización se incrementará con el número de iteraciones.

El éxito de este método depende de la temperatura inicial y de la forma de la función de enfriamiento.

II.5.1.2 Mantenimiento de una población factible por representaciones y operadores genéticos especiales.
Decodificadores.
Koziel y Michalewicz [KM99] proponen un método para resolver el problema general de programación no lineal. Trabajan con representación en los reales. Proponen usar un decodificador que imponga funciones de correspondencia entre la región factible en *S* $F \cap S \subset R^n$, y un rectángulo *n* dimensional : $[-1, 1]^n$. Los mismos investigadores indican las condiciones que deben ser satisfechas por tal decodificador : 1) para cada solución $s \in F$ debe haber una solución codificada *d*, 2) cada solución codificada *d* corresponde a una solución factible *s*, y 3) todas las soluciones en *F* deben ser representadas por el mismo número de codificaciones *d*. Agregan que es razonable pedir otras dos condiciones más : 4) que la transformación entre los dos dominios sea computacionalmente rápida y 5) que tenga la característica de *localidad*, en el sentido de que pequeños cambios en la solución codificada correspondan a pequeños cambios en la solución misma. Proponen el uso de funciones de correspondencia entre los dominios de las funciones y el rectángulo *n* dimensional para funciones con regiones factibles convexas y no convexas. El dominio original de la función y el rectángulo *n* dimensional son conjuntos homomorfos, con la ventaja de que el rectángulo *n* dimensional sólo contiene soluciones factibles, por lo que se puede usar cualquier AG para llevar a cabo la búsqueda del óptimo.

El método requiere que se afine un parámetro que depende de cada problema y requiere hacer muchos cálculos por lo que requiere de mucho tiempo de computadora.

II.5.1.3 Separación de objetivos y restricciones.
Superioridad de puntos factibles.
Esta es otra forma de atacar el problema. Powell y Skolnick [PS93] proponen definir :

$$\phi(x) = \begin{cases} E(f(\boldsymbol{x})) & \text{si } \boldsymbol{x} \text{ es factible} \\ 1 + r \sum_{\substack{\rho_i \in R \\ \rho_i \text{ violada}}} \rho_i(\boldsymbol{x}) & \text{si } \boldsymbol{x} \text{ no es factible} \end{cases}$$

donde
E es una función que escala a f en el intervalo abierto $(-\infty, 1)$,
r es un parámetro que se debe afinar, si se usa el ordenamiento por rango[5] en la población es irrelevante el usarlo.
La idea es que los puntos factibles siempre tendrán menores valores (que son mejores para un problema de minimización) que los puntos de la región no factible. Si además se usa el ordenamiento por rango se mejoran las diferencias de escala de la función y las restricciones, además de permitir que los puntos no factibles colaboren también con las poblaciones en la búsqueda del óptimo.

Mäkinen, Miettinen y Mäkelä proponen otro método de superioridad de los puntos factibles pero, a diferencia del propuesto por Powell y Skolnick, está orientado a la forma de dar mejor calificación a los elementos en la región factible de cada población. Su propuesta es adicionar otra función a la función ya penalizada :

$$\phi(\boldsymbol{x}) = f(\boldsymbol{x}) + r \sum_{\substack{\rho_i \in R \\ \rho_i \text{ violada}}} \rho_i(\boldsymbol{x}) + \theta(\boldsymbol{x})$$

donde

$$\theta(x) = \begin{cases} 0 & si\ P \cap \mathcal{F} = 0\ o\ x \in \mathcal{F} \\ \alpha & de\ otra\ manera \end{cases}$$

P es una población.

$$\alpha = max\left[0, \max_{y \in P \cap \mathcal{F}} f(y) - \min_{z \in P - \mathcal{F}} \left(f(z) + r \cdot \sum_{\substack{\rho_i \in \mathcal{R} \\ \rho_\nu \text{ violada}}} \rho_i(z) \right) \right]$$

Para que estos métodos trabajen necesitan que en sus poblaciones existan puntos en la región factible F.

Coevolución.
Paredis [Par94] propone atacar el problema del manejo de restricciones usando la coevolución, Coello [Coe99] lo describe más o menos de la siguiente manera : formar dos poblaciones, una constituida por todas las restricciones que se deben satisfacer y otra por un conjunto de soluciones potenciales (soluciones que pueden estar o no dentro de la región factible). Un individuo de la primera población que tenga un alto valor de aptitud representa una restricción

[5] El ordenamiento por rango consiste en ordenar con base en su aptitud a todos los elementos de una población y asignarles otros valores de aptitud en base a esta ordenación. Por ejemplo, al mejor elemento se le puede asignar una aptitud de 2 y al peor una aptitud de 1, a los elementos intermedios se le pueden asignar valores proporcionales a la posición que ocupen entre estos dos valores.

que no es cumplida por un gran número de soluciones potenciales y un individuo de la segunda población con un alto valor de aptitud representa una solución que satisface muchas restricciones. La forma en la que se obtienen las aptitudes de ambas poblaciones es tomando muestras de ambas y evaluando a cada individuo de ambas muestras. Los resultados de cada individuo se registran. El proceso se repite varias veces, la aptitud de cada individuo de cada población depende de la historia de estos registros. De esta forma se identifica tanto a la restricción más difícil de cumplir como al mejor individuo. El objetivo es hacer que el proceso de búsqueda se concentre en encontrar soluciones que vayan satisfaciendo a la restricción más difícil en turno. Por supuesto, es la población de soluciones potenciales la que se somete al proceso evolutivo.

El método se estanca para aquellos casos en los que la dificultad para satisfacer las restricciones es grande para todas ellas.

Memoria conductista.
Este método es propuesto por Schoenauer y Xanthakis [SX93]. Consiste en tomar en cuenta una sola de las restricciones a la vez y hacer evolucionar a la población hasta que una fracción de ella, el umbral de cambio, satisfaga a esa restricción. Esa misma población se usa como población inicial para iniciar un proceso de evolución que fuerce a la población a satisfacer una segunda restricción, aquellas soluciones potenciales producida que no satisfagan alguna o las dos restricciones son eliminadas. Se continúa hasta que la fracción fijada de la población (el umbral de cambio) satisfaga ambas restricciones. Esta nueva población servirá de población inicial para iniciar un proceso de evolución para satisfacer la tercera restricción. El proceso se repite hasta obtener una población cuyo umbral de cambio satisfaga a todas las restricciones.

El orden en el que son satisfechas las restricciones puede afectar el resultado.

II.5.1.4 Métodos híbridos.
Multiplicadores de Lagrange.
Adeli y Cheng [AC94] propusieron un método basado en la minimización secuencial del método de Lagrange. Usa una función de aptitud de la forma:

$$\phi(\boldsymbol{x}) = f(\boldsymbol{x}) + \frac{1}{2}\sum_{j=1}^{m}\gamma_j\{[\rho_j(\boldsymbol{x}) + \mu_j]^+\}^2$$

donde $\gamma_j > 0$, μ_j es un parámetro asociado con la *j-ésima* restricción y *m* es el número de restricciones.

$$[\rho_j(\boldsymbol{x}) + \mu_j]^+ = \max[0,\ \rho_j(\boldsymbol{x}) + \mu_j]$$

μ_j se define en términos de la máxima violación de su correspondiente restricción registrada previamente y es escalada usando un parámetro $\beta > 1$ que es definido por el usuario. γ_j se incrementa multiplicando β (que se mantiene constante durante todo el proceso) por el valor previo de γ_j, asegurando así que la penalización se incrementa en cada generación

II.5.1.5 Nuevas estrategias.

Sistema inmune.

Este acercamiento imita lejanamente el comportamiento del sistema inmune específico de los animales superiores. El sistema inmune es capaz de generar unas moléculas llamadas anticuerpos que "embonan" en partes de las estructuras de entes extraños, patógenos, que penetran en el organismo anfitrión defendido por él; de esta manera son marcados para su posterior destrucción. Hajela y Yoo [HY96] proponen un algoritmo evolutivo consistente en dos algoritmos genéticos uno anidado en el otro. El algoritmo genético más externo se encarga de llevar a cabo la optimización y el anidado se encarga de hacer evolucionar una población de anticuerpos para que "identifiquen" (en este contexto significa que se parezcan a) una población de antígenos. Los antígenos son puntos dentro de la región factible, los anticuerpos son puntos fuera de esa región. Este algoritmo genético anidado toma muestras de ambas poblaciones y asigna valores de aptitud a los anticuerpos con base en su similitud con los de la muestra de la población de antígenos; entre más parecidos mayor aptitud. El algoritmo genético hace evolucionar a la población resultante de anticuerpos. La población de antígenos es actualizada usando los mejores antígenos (anticuerpos que caen dentro de la región factible) obtenidos por el GA externo y se vuelve a aplicar el GA anidado. El proceso se repite hasta que se cumpla la condición de terminación.

Este método requiere que desde el principio se conozcan puntos dentro de la región factible.

Hajela y Yoo [HY96] han utilizado cadenas binarias para representar los anticuerpos y los antígenos. Cruz [Cru00] ha utilizado el algoritmo de Hajela y Yoo usando dígitos decimales para la representación de los anticuerpos y antígenos, y complementos a 10 para el operador de mutación, ha obtenido buenos resultados.

II.5.1.6 Comentarios.

En las subsecciones anteriores se ha querido dar un panorama muy general de las formas en las que se ha atacado el problema de la optimización numérica de funciones restringidas con algoritmos evolutivos.

Los autores que estudian la optimización de funciones restringidas aplican algunas reglas heurísticas que han aprendido, por ejemplo, para el caso de penalización una de estas heurísticas dice[6]: *las penalizaciones que están en función de las distancias a la región factible funcionan mejor que aquellas que dependen únicamente del número de restricciones violadas*. Sin embargo, tales reglas no siempre funcionan:

> Kuri y Gutiérrez [KG01] compararon los métodos de: k) satisfacción de las restricciones por su número de Kuri y Villegas, h) de Homaifar, j) de Joines y Houck, s) de Schoenauer y Xantakis y p) de Powell y Skolnick, todos ellos vistos arriba. Para llevar a cabo la comparación usaron una colección de 25 funciones que iban de relativamente fáciles de resolver a muy difíciles. Hicieron varias corridas para justificar estadísticamente sus resultados. Cada vez que probaban un método escogían una función al azar. El número de generaciones que corría el GA lo fijaron previamente. Hicieron algunos experimentos cada uno con diferentes valores de los parámetros del GA para ver si esas variaciones en los parámetros afectaban

[6] [RPLH89], citado por Coello en [Coe99]

apreciablemente al desempeño de los métodos. Concluyeron que dicha variaciones no afectaron grandemente el desempeño de éstos. A continuación se enuncian los métodos del mejor al peor según los resultados que obtuvieron: k, p, s, j y h.

En este caso resultó ser mejor el método que sólo toma en cuenta el número de restricciones violadas y no su magnitud, lo que va en contra de la heurística[7]. Esto puede indicar que la efectividad de un método depende de los problemas a los que se aplique. Para ciertos problemas es necesario el tomar en cuenta no sólo el número sino también la magnitud de la violación de las restricciones, mientras que para otros el tomar en cuenta sólo el número de restricciones no cumplidas es mejor.

Lo que se podría concluir de esto es que es necesario el probar un algoritmo para manejo de restricciones con problemas del tipo para los que se pretende que el algoritmo trabaje.

II.6 Sistemas inmunes artificiales.

El estudio del sistema inmune para usarlo como paradigma para procesos computacionales ha tomado gran auge en los últimos años [Das98b].

El uso de sistemas inmunes artificiales es una disciplina nueva. No ha alcanzado, como otras disciplinas inspiradas en sistemas biológicos, un estado en el que los investigadores en un área trabajen sobre algoritmos bien establecidos y se dediquen a hacerles modificaciones, para hacerlos funcionar mejor, como es el caso de las redes neuronales y los algoritmos genéticos ([DT02]p.4).

II.6.1 Antecedentes del uso de los (SIAs) para optimización numérica.

Fukuda *et al.* [FMT98] utilizan el concepto de recombinación somática y mutación para llevar a cabo la optimización de funciones numéricas *no* restringidas.

Bersini y Varela, según reportan de Castro y Timmis en su libro ([DT02]p.124) usan un modelo para describir la sensibilidad de los componentes del sistema inmune y una función metadinámica para el reclutamiento de nuevos componentes y la eliminación de elementos inútiles, para la optimización de funciones numéricas.

Los mismos autores dan como único antecedente de usos de modelos del sistema inmune para optimización de funciones numéricas restringidas el trabajo de Hajela y Yoo ([DT02]p.125); ya se ha mencionado aquí el trabajo de Cruz [Cru00], quien usa una modificación del algoritmo de Hajela y Yoo para hacer optimización numérica restringida.

[7] Sobre este caso, el que esto escribe opina que sería conveniente repetir los experimentos fijando un número mayor de generaciones. Un número lo suficientemente grande como para que al menos uno de los métodos logre encontrar un punto en región factible con un valor cercano al óptimo restringido para cada una de las funciones. El número de generaciones fijado por los autores fue de 100, un número que, si la región factible es difícil de encontrar, puede no permitir el encontrar un punto en ella, y lo que se estará calificando será la capacidad que tiene el algoritmo de satisfacer sólo algunas de las restricciones y no su capacidad para entrar en región factible y buscar el óptimo dentro de ella. Esta misma inquietud la manifiestan los autores en el inciso (c) de sus conclusiones [KG01].

Capítulo III
Sistema Inmune

De los sistemas inmunes que existen en la naturaleza parece ser que es el de los vertebrados con mandíbula el más evolucionado y complejo. Es este sistema, en especial el del ser humano y el de los ratones de laboratorio, el que los investigadores tratan de simular o del que toman ideas para tratar de resolver otros problemas. En este capítulo se hablará a grandes rasgos de ese sistema inmune, explicando sus principales mecanismos y la terminología usada para describir sus partes y su funcionamiento. Esto servirá para hablar sobre algunos de los modelos computacionales inspirados en él, ya que los desarrolladores de estos modelos toman ideas de ese sistema inmune y usan mucha de la misma terminología para describirlos.

III.1 Una clasificación.

Dos maneras de clasificar los modelos computacionales de sistemas inmunes podrían ser : I) según el modelo de sistema inmune natural en el que se basen y II) según el fin que se persiga con el modelo computacional.

A grandes rasgos, existen dos modelos que explican el funcionamiento de los sistemas inmunes naturales de los vertebrados con mandíbula (que son los sistemas inmunes más evolucionados [LAR99]): I.1) la red inmunitaria y I.2) la selección clonal. En lo general, ambos modelos toman en cuenta la existencia de los mismos integrantes del sistema inmune: los que forman parte del sistema inmune primario (el sistema no especializado, el que no "aprende") que son básicamente diferentes tipos de células (diferentes tipos de glóbulos blancos) que fagocitan a los patógenos, y los que forman parte del sistema inmune adaptable (el que sí "aprende", el que se puede especializar) que está constituido por linfocitos B y T (que son otros tipos de glóbulos blancos), y por células a las que ellos dan origen como lo son las células plasmáticas (que son "fabricas" de anticuerpos) y las células de memoria.

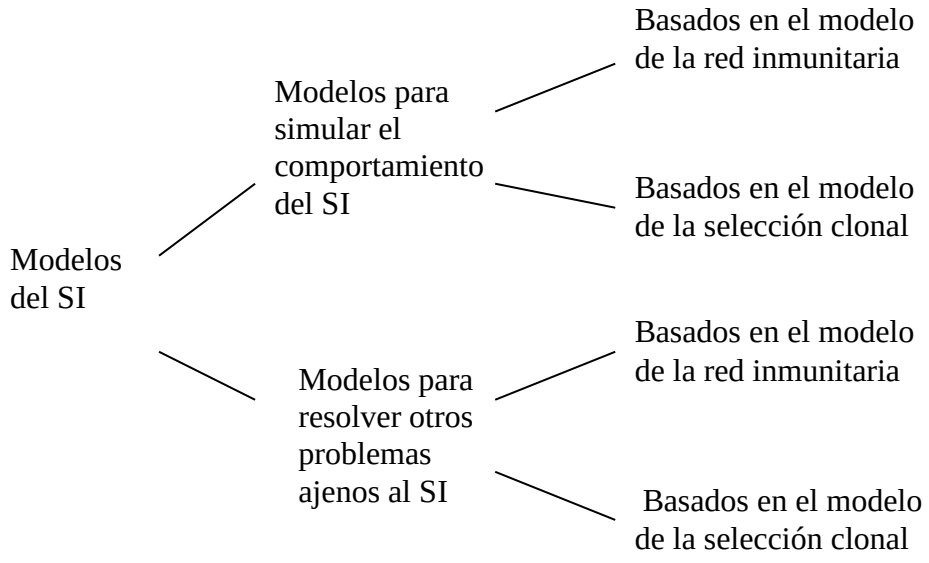


Figura III.1. Una clasificación de los modelos del sistema inmune.

Es sobre las relaciones que tienen las células del sistema inmune adaptable donde difieren ambos modelos. Tales diferencias se discutirán más adelante. Según el fin que se persiga al

elaborarlos, los modelos computacionales del sistema inmune pueden clasificarse en II.1) modelos para simular el comportamiento de los sistemas inmunes naturales, y II.2) modelos que se inspiran en el funcionamiento del sistema inmune natural (en mayor o menor grado) para resolver algún otro problema no relativo al sistema inmune.
Ambas clasificaciones pueden combinarse como se muestra en la figura III.1.

III.2 El sistema inmune natural como paradigma de los modelos computacionales del sistema inmune[1].

Introducción.

Para poder hablar sobre los modelos computacionales del sistema inmune ayuda mucho revisar, aunque sea de manera burda, los modelos que tratan de explicar cómo funciona el sistema inmune natural, ya que los investigadores en el área de cómputo los han estudiado ya sea para usarlos como fuente de inspiración para desarrollar algoritmos para resolver problemas ajenos al sistema inmune o para desarrollar modelos computacionales para simularlo. Otra razón para hablar sobre los modelos que tratan de explicar cómo funciona el sistema inmune natural, es que los investigadores que desarrollan modelos computacionales de sistemas inmunes toman muchos de los términos del área del sistema inmune natural para aplicarlos a la de los modelos computacionales. Los objetivos de esta sección son entonces 1) el explicar a grandes rasgos cuáles son y en que consisten algunos de los principales mecanismos usados por el sistema inmune natural para la defensa contra los patógenos y 2) introducir los términos usados en el área de los sistemas inmunes naturales, que en gran medida son usados en el área de los modelos computacionales del sistema inmune, aunque a veces no para designar exactamente a los mismos actores o las mismas situaciones, cuando éste sea el caso, se procurará hacer las aclaraciones pertinentes.

Hay quien afirma que el sistema inmune no tiene un objetivo determinado (Segel y Bar-Or, [SB99a]), pero agrega que el atribuirle un objetivo final resulta de utilidad para modelar su comportamiento. El principal objetivo que parece ser el más obvio atribuirle, es el de defender al organismo en el que reside y del que forma parte, de los patógenos tanto internos (*e.g.*, células cancerosas) como externos (*e.g.* bacterias, virus, protozoarios), procurando producirle el menor daño posible, ya que entre sus estrategias de defensa, existen algunas que pueden causar daños de diferentes magnitudes al propio organismo. Para cumplir este objetivo, el sistema inmune usa varios y complejos mecanismos. En un modelo computacional por lo general sólo se utilizan uno o unos cuantos de ellos.

III.2.1 El sistema inmune adaptable.

El funcionamiento del sistema inmune adaptable se acostumbra explicar dividiendo los procedimientos de defensa que usa contra los patógenos en dos partes: 1) *inmunidad humoral* y 2) *inmunidad con células de por medio* (*cell mediated immunity*). La primera comprende los mecanismos que usa el sistema inmune para *marcar* a los patógenos que se encuentren en los líquidos del organismo (para que posteriormente sean mucho más fáciles de identificar para su inmediata destrucción por parte de los glóbulos blancos que forman parte del sistema inmune primario). La segunda comprende los mecanismos que usa el sistema inmune para destruir a las células del propio organismo junto con los patógenos que han logrado penetrar a ellas. Los

[1] Para elaborar esta sección se han consultado varias referencias sobre sistemas inmunes naturales, principalmente [GKO00], [AL01], varios artículos de la 2ª edición de la enciclopedia de inmunología [DR98] y varios artículos de revistas de inmunología.

principales actores en la inmunidad humoral son los linfocitos B, y en la inmunidad con células de por medio son los linfocitos T. Ambos tipos de procedimientos han sido usados como inspiración en modelos computacionales del sistema inmune.

Tanto los linfocitos T como los B son generados dentro de la médula ósea, que es un órgano que se encuentra dentro de los huesos largos. Ambos tipos de células poseen estructuras alargadas que emergen de su superficie (membrana celular) y cuyo eje longitudinal está orientado desde la célula hacia su exterior. Cada linfocito tiene una gran cantidad de estas estructuras (alrededor de cien mil) todas ellas idénticas entre sí. Esas estructuras reciben el nombre de *receptores*. Los receptores son proteínas y los de los linfocitos B tiene forma de "Y", el extremo que tiene los dos brazos es el que se dirige hacia el exterior del linfocito. Estas proteínas que son los receptores de los linfocitos B reciben el nombre de *inmunoglobulinas*. Existe una gran cantidad de linfocitos B diferentes entre sí. Lo que hace distintos a dos linfocitos B entre sí es la superficie de las puntas de cada uno de los brazos de la "Y" (*regiones variables*). Justo en el extremo de las regiones variables de cada bazo existe una oquedad en la que *podrían* embonar las partes protuberantes (*epitopes*, llamados también *antígenos* o *regiones determinantes antigénicas*) de alguna molécula *de gran tamaño*. El grado en el que embona (es decir, el que tan bien embona) un epitope en la oquedad de cada una de las regiones variables de la inmunoglobulina, es llamado *afinidad* , entre mejor embonen, mayor será la afinidad entre el epitope y la inmunoglobulina. Existe un umbral de la afinidad por arriba del cual se activan los mecanismos del sistema inmune (a esta activación de los mecanismos del sistema inmune se le llama *respuesta inmune*). Es por eso que es importante que la afinidad epitope-inmunoglobulina sea grande. Entre mayor afinidad haya entre un epitope y una inmunoglobulina mejor será la respuesta inmune (contra un patógeno con ese epitope).

III.2.2 El uso indistinto del término antígeno y patógeno.

En muchos casos, cuado no hay lugar a confusión, se usa el término *antígeno* como sinónimo de *patógeno*, en otros, el término antígeno tiene su estricta acepción, que es aquella parte (de una gran molécula) que coincide en el receptor sin importar si es parte del propio organismo o de un patógeno.

III.2.3 La recombinación somática, una fuente de la variedad de los receptores de los linfocitos B y T.

Por la forma en la que se genera el material genético de los linfocitos B y T puede haber una gran variedad de ellos. La parte del material genético de los linfocitos que *expresa*[2] a la región variable de sus receptores, se forma tomando un conjunto de piezas de entre varias posibles, disponibles en el genoma del organismo y que sirven específicamente para esto (en el genoma del organismo hay *bibliotecas* de estas piezas). A esto se le llama *recombinación somática*.

III.2.4 La maduración de la afinidad, una fuente aún mayor de variedad de los linfocitos B. Selección positiva de los linfocitos B. Edición de receptores.

Algunos linfocitos B van a los nodos linfáticos o al baso, que son órganos del sistema inmune. Dentro de esos órganos existen unas estructuras llamadas *centros germinales*. En los centros

[2] El término *expresar* significa obtener una proteína a partir de su codificación en una parte del ácido desoxirribonucléico (DNA, *deoxí-ribo nucleic acid*) que se encuentra en el núcleo de la célula.

germinales hay unas células llamadas *células dendríticas foliculares* en cuya superficie hay pequeñas estructuras denominadas iccosomas (*iccosomes, immune-complex coated bodies*), en ellas se encuentran aglomerados una gran cantidad de epitopes de los patógenos que han irrumpido en el organismo. También dentro de los centros germinales ocurre el siguiente fenómeno: cada vez que se reproduce (asexualmente) un linfocito B hay una probabilidad muy cercana a uno de que mute la mínima parte de información (mutación puntual) que puede mutar en la parte del genoma del linfocito B que codifica a la región variable de los receptores. Así que es muy probable que el hijo de un linfocito B tenga una pequeña variación en la parte variable de sus receptores con respecto al padre. A este fenómeno se le llama *hipermutación somática*. Los linfocitos B son probados con los iccosomas en las células dendríticas foliculares. Si después de un tiempo los receptores de un linfocito B no reconocen con alta afinidad ninguno de lo epitopes presentados por los iccosomas, el linfocito B es destruido (se "suicida" por *apoptosis*). Si el linfocito B reconoce con gran afinidad uno de dichos epitopes, sufre una expansión monoclonal (ver siguiente sección). Esto es la *selección positiva* de los linfocitos B. Existen mecanismos todavía no muy claros, que hacen que el hijo varíe en mayor medida que con respecto a una sola mutación puntual, a esto se le llama *edición de receptores*.

III.2.5 Expansión monoclonal.

Cuando los receptores de un linfocito B embonan con alta afinidad con un epitope, se inicia la respuesta inmune. Primero el linfocito se divide en varias células idénticas entré sí, es decir, tienen todas ellas los mismos receptores. Estas células reciben el nombre de *clones* y al hecho de que el linfocito B produzca células similares a él e idénticas entre sí se le llama *expansión monoclonal*. Los clones a su vez dan origen a las *células plásmáticas* y a las *células de memoria*.

III.2.6 Los anticuerpos y equivalencia de los términos *anticuerpo* y *linfocito B* en los modelos computacionales.

Las células plasmáticas producen gran cantidad de inmunoglobulinas (las proteínas que son los receptores de los linfocitos B) y las liberan para que circulen por los líquidos del organismo. A estas inmunoglobulinas "libres" (es decir que no están unidas a la membrana celular de un linfocito B) se les llama *anticuerpos*. Los anticuerpos circulan por los líquidos del organismo. En la gran mayoría de los modelos computacionales del sistema inmune no es necesario hacer una diferencia entre los receptores de los linfocitos B y los anticuerpos, por lo que se usan de manera indistinta los términos *anticuerpo* y *linfocito B.*

III.2.7 La opsonosis.

Los anticuerpos que se adhieren a los epitopes provocan un aumento en la capacidad de destrucción que poseen los glóbulos blancos del sistema inmune primario. Un patógeno con anticuerpos pegados en su superficie tiene entre 3000 y 4500 veces mayor probabilidad de ser destruido que el mismo patógeno sin los anticuerpos. Al hecho de que se peguen los anticuerpos a los epitopes de una estructura se le llama *opsonosis*.

III.2.8 Las células de memoria y la reactividad cruzada.

Estas células poseen los mismos receptores que los linfocitos que les dieron origen. En el *modelo de selección clonal* se establece la hipótesis de que a diferencia de los linfocitos, las células de memoria tienen una muy larga vida (se supone que algunas viven dentro del

organismo hasta la muerte de éste) y se quedan circulando en él para protegerlo en caso de que el mismo patógeno u otro con un epitope muy similar a él, irrumpa de nuevo en el organismo. Si se diera este caso, la respuesta del sistema inmune sería mucho más rápida y no se presentarían, o se presentarían más ligeros, los síntomas de la enfermedad causada por el patógeno en cuestión. Es así como el modelo de selección clonal explica la adquisición de la inmunidad contra una enfermedad que presentan los individuos que con anterioridad la han sufrido. Es importante recalcar el hecho que aquí se ha mencionado de que un epitope parecido a otro puede ser reconocido por (embonar con similar afinidad con) la misma inmunoglobulina o receptor, a esto se le llama *reactividad cruzada*. Esta es una característica muy importante del sistema inmune natural, ella explica el hecho de que haya presente en el organismo sólo una pequeña fracción de todos los linfocitos *distintos entre sí* que pueden existir, y que a la vez se le brinde a este organismo una aceptable protección contra los patógenos. También explica el que funcionen muchos tipos de vacunas.

III.2.9 Enfermedades autoinmunes.

Hay que hacer notar que si 1) existe en el organismo un linfocito B con receptores altamente afines a un epitope propio y 2) ocurre que se embonan el receptor y ese epitope, entonces también se puede desencadenar una respuesta inmune que provocará la destrucción de la estructura (molécula, célula, tejido, etc.) en la que está dicho epitope, esto es ¡la parte del propio organismo a la que pertenece[3]! . A esto se le llama *enfermedad autoinmune*.

III.2.10 Selección negativa de los linfocitos B.

Los linfocitos B son generados y maduran (esto es, adquieren sus receptores) dentro de la médula ósea, allí mismo se prueba si sus receptores son afines a epitopes del propio organismo. En caso afirmativo, son destruidos dentro de la médula ósea, para evitar que salgan de ella y circulen por el organismo. A esto se le llama *selección negativa* de los linfocitos B

Este es uno de los mecanismos en los que se inspiran muchos trabajos en donde se plantean modelos computacionales de sistema inmune.

III.2.11 Observaciones que hasta aquí es importante recalcar sobre el modelo de selección clonal.

Hasta aquí hemos visto el planteamiento de que el sistema inmune es un sistema que permanece *estático,* es decir, que comienza a actuar hasta que un patógeno irrumpe en el organismo. También se ha planteado que después de un ataque de un patógeno que ha sido dominado por el sistema inmune, quedan células de memoria con gran tiempo de vida que servirán para repeler un nuevo ataque de ese o de otro con un epitope similar al de ese patógeno. Esto es lo que postula el *modelo de la selección clonal*.

III.2.12 El timo y la maduración de los linfocitos T.

Como los linfocitos B, los linfocitos T son generados dentro de la médula ósea, pero a diferencia de éstos, no maduran (es decir no adquieren sus receptores) dentro de ella, por lo que no son sometidos a ningún tipo de selección dentro de ese órgano. Los linfocitos T salen de la médula ósea y se dirigen hacia el timo, que es un órgano que se encuentra por detrás del

[3] Existe un mecanismo para que en el caso de que se haya escapado un linfocito B con estas características no se monte respuesta inmune. Tal mecanismo recibe el nombre de anergia. Sin embargo muchas veces tal mecanismo falla y sí se monta respuesta inmune contra el propio organismo.

esternón y que se va degenerando (disminuye en tamaño drásticamente y baja su actividad) con la edad. Es allí donde continúa el crecimiento de los linfocitos T y adquieren sus receptores.

III.2.13 Presentación del antígeno, el MHC clase I.

Para explicar cómo se seleccionan los linfocitos T ya con receptores es necesario explicar cómo invaden algunos patógenos a las células del organismo y la respuesta de éstas a tal invasión. A continuación se explican estas cuestiones. Existen patógenos que penetran dentro de las células del organismo evadiendo así la acción de los linfocitos B (cuando un patógeno está dentro de una célula del propio organismo no hay forma en que pueda haber contacto entre los receptores de los linfocitos B y los epitopes del patógeno), se multiplican dentro de la células (en muchos casos, hasta que son tantos que las hacen estallar) y luego salen de ellas y se diseminan para infectar más células. La célula infectada tiene un mecanismo para avisar al exterior de lo que está ocurriendo en su interior. Toma pequeños fragmentos de los patógenos, los engarza en la punta de una proteína (llamada MHC clase I, aquí se le llamará MHC-I) y luego saca a la proteína con el fragmento de patógeno engarzado para que quede pegada en la parte externa de su membrana celular con el extremo que tiene engarzado el patógeno apuntando en dirección contraria a la célula, es decir, hacia el exterior. Los linfocitos T tienen receptores que reconocen a la MHC-I. Al igual que los linfocitos B, estos receptores de los linfocitos T (TCRs, *T cell receptors*) son muy específicos. Si un TCR embona bien con la muestra de antígeno presentada por la MHC-I en la superficie de la célula, se desencadena la respuesta inmune que terminará por destruir a la célula invadida con substancias venenosas para ella (¡y para los tejidos celulares circundantes que estén o no infectados!).

III.2.14 Células presentadoras de antígeno, el MHC Clase II.

Existen células que fagocitan a los patógenos que acostumbran penetrar a las células del organismo. En su interior los destruyen y engarzan partes de estos patógenos sobre una proteína (llamada MHC clase II, aquí se le llamará MHC-II), luego esa proteína es mandada a la parte exterior de la membrana celular, donde queda pegada con el extremo en el que está engarzado el antígeno orientado hacia el exterior de la célula. Cuando el receptor de un linfocito T "reconoce" (embona con alta afinidad con el antígeno presentado por el MHC-II), ese linfocito T sufre una expansión clonal como en el caso de los linfocitos B. Estas células son llamadas *células presentadoras de antígeno* (APCs, *antigen presenting cells*). Como se ha visto, sirven de "reclutadoras" de linfocitos T, que son los que se encargan de patrullar en busca de células con los antígenos engarzados en los MHC-I. Por supuesto, a diferencia de las células infectadas, ellas no son destruidas (más bien son ellas las que destruyen a los patógenos que fagocitan para después presentar algunas de sus partes engarzadas en el MHC-II). Cabe aclarar que como resultado de la expansión clonal de un linfocito T se generan otras células, tales como los *linfocitos T auxiliadores* (T *helpers*) de tipo 1 o 2 (Th1 o Th2), *linfocitos T de memoria*, *linfocitos T citotóxicos*, etc.

III.2.15 Selección positiva y negativa de los linfocitos T en el timo.

Como se ha mencionado arriba, los linfocitos T adquieren sus receptores en el timo. Es allí mismo donde son sometidos a la selección positiva y negativa. Aquellos linfocitos T que no tienden a acercarse a los MHCs son descartados (destruidos), ya que no servirán para probar si sus receptores embonan en los antígenos engarzados en los MHC. Aquellos cuyos TCRs sí reconocen a los MHCs son selecciondos para la siguiente prueba. A esto se le llama *selección*

positiva. En el timo existen células llamadas *células dendríticas* que tiene la peculiaridad de presentar engarzados en MHCs antígenos propios (*i.e.*, del propio organismo), que es al que los linfocitos T deben defender (y del que forman parte). Aquellos linfocitos cuyos receptores se unen con gran fuerza a estos complejos antígeno-MHC presentados por las células dendríticas son también destruidos en el timo, para no dejarlos salir ya que podrían provocar una enfermedad autoinmune. A esto se le llama *selección negativa* de los linfocitos T. Un promedio de sólo el 2% de los linfocitos T que llegan al timo para madurar, pasan estas dos pruebas y salen del timo para defender al organismo.

III.2.16 Una red química de comunicaciones, las citocinas.

Los linfocitos secretan substancias químicas (*citocinas*) cada vez que hacen algo importante Las citocinas se difunden por los tejidos del organismo e informan a los demás linfocitos de lo que está ocurriendo. Por ejemplo, cada vez que el receptor de un linfocito embona con alta afinidad con un antígeno, secreta un conjunto de citocinas para que los otros linfocitos se estimulen. Como todos los linfocitos hacen lo mismo (secretar ese conjunto de citocinas), una alta concentración de ese conjunto de citocinas indica a todos ellos que la infección está en su apogeo y que deben estar muy "alerta" (la estimulación de los linfocitos es proporcional a las concentraciones de ese conjunto de citocinas).

III.2.17 El modelo de la red inmune.

Existe una forma alterna de explicar la inmunidad contra una enfermedad después de haberla sufrido, y la gran variedad de linfocitos B que se mantienen circulando por el organismo. La explicación la da la teoría de *la red inmune*. En ésta se sostiene que en la región variable de los receptores, a un costado del extremo de la oquedad donde embonan los epitopes, existen a su vez regiones que son epitopes para los receptores de otros linfocitos. La parte de la región variable donde se encuentran la oquedad donde embonan los epitopes recibe el nombre de *idiotope* y la parte de la región variable que es epitope para otros receptores recibe el nombre de *paratope*. En la figura III.2 se ilustra la relación entre diferentes linfocitos B. En ella se muestra como un epitope de un patógeno es reconocido por el receptor del linfocito B rotulado como B-1 en la figura. El paratope del linfocito B-1 es reconocido a su vez por el receptor del linfocito B rotulado como B-2. De esta manera, las concentraciones de los patógenos son controladas por los linfocitos B-1 y la concentración de éstos es regulada a su vez por los linfocitos B-2. Un tercer tipo de linfocito B se representa en la figura 1, el rotulado como B-3, este linfocito tiene un paratope igual al epitope del patógeno, por eso se le denomina *imagen interna* del patógeno. Este linfocito sirve para estimular la creación de linfocitos B-1 para que la inmunidad contra el patógeno persista aun cuando éste se haya eliminado totalmente. Nótese que los linfocitos se encuentran entonces en un *equilibrio metadinámico*. No es necesario que un patógeno perturbe sus concentraciones, ya que éstas, las concentraciones de los linfocitos, están variando continuamente al estarse regulando unas a otras. Esto explica la persistencia por largo tiempo de los linfocitos que surgen para atacar un patógeno, y con ello se explica la inmunidad contra la enfermedad después de haberla sufrido una vez. También explica la gran variedad de linfocitos que se encuentran presentes al mismo tiempo en el organismo.

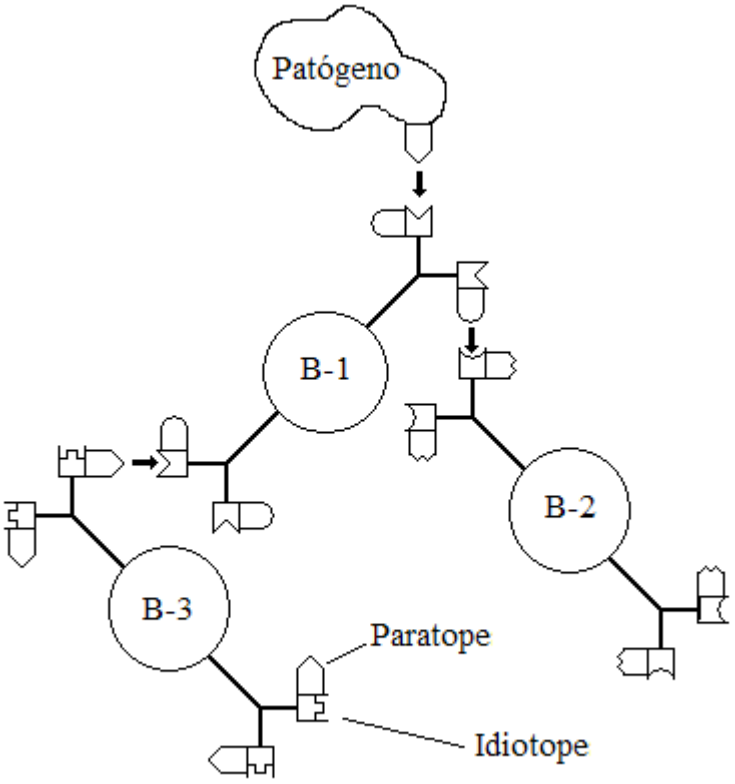


Figura III.2. Parte de una red inmune. El patógeno es reconocido por el linfocito B-1, este a su vez es reconocido por el linfocito B-2. El linfocito B-1 también reconoce al linfocito B-3. B-3 es la imagen interna del patógeno.

III.3 Modelos computacionales del sistema inmune.

El objetivo que se persigue aquí no es hacer una revisión exhaustiva de los modelos computacionales del sistema inmune (obra que por sí misma sería motivo de un extenso estudio aparte) sino mostrar varios ejemplos de los tipos de modelos según la clasificación mostrada en la figura III.1. Con esto se pretende dar una idea general de qué se hace en el área de modelos computacionales del sistema inmune, y principalmente dar una idea de cómo la inmunología suministra modelos expresados de diferentes formas (desde la expresión en lenguaje natural complementada con esquemas hasta expresiones matemáticas) que pueden dar buenas ideas para la solución de problemas fuera del ámbito de la imunología.

El término *modelo computacional* usado aquí se refiere indistintamente a modelos matemáticos del sistema inmune, desarrollados por diversos investigadores, que son resueltos usando métodos numéricos ejecutados por computadoras; y a modelos que no han sido planteados con expresiones matemáticas (al menos no en su totalidad) sino que han sido concebidos en forma de procedimientos para ser ejecutados por computadoras. Un ejemplo de los primeros podría ser un conjunto de ecuaciones diferenciales ordinarias que representen los cambios con respecto al tiempo de las concentraciones de diversas células del sistema inmune, de las citocinas que secretan y las substancias que perciben, así como de las concentraciones de los patógenos. Un ejemplo de los segundos podría ser un conjunto de procedimientos para

representar la evolución con un algoritmo evolutivo de células del sistema inmune para mejorar su respuesta al ataque de los patógenos.

A) Modelos para simular el comportamiento del sistema inmune

Como comentan Perelson y Weisbuch [PW97], uno de los objetivos de modelar en inmunología, es deducir las propiedades macroscópicas del sistema a partir de las propiedades e interacciones entre sus componentes elementales.

A.1) Modelos para simular el comportamiento del sistema inmune basados en el modelo de selección clonal.

El objetivo de estos modelos es simular el comportamiento del SI para explicar las causas de algún fenómeno relativo a él. Los que se revisarán en esta subsección están basados en el modelo de *selección clonal* (ver III.2).

El modelo de la retroalimentación difusa y la red de información difusa.

Se conocen muchos de los *efectores* (las entidades que realizan acciones: linfocitos, fagocitos, macrófagos, células presentadoras, etc.) del sistema inmune y muchas de las citocinas que secretan. Sin embargo, de la forma global en la que opera el sistema inmune sólo se conoce muy poco. Existen modelos que tratan de explicarlo. Segel y colaboradores han propuesto a lo largo de varios trabajos (Segel & Lev Bar-Or, 1999 [SL99]; Segel, 2001 a [Seg01]; Segel, 2001b [Seg01b]; Segel 2001c [Seg01c]; Segel 2001d [Seg01d]; Segel, 2002 [Seg02]; Bergmann, van Hemmen y Segel, 2000 [BHS00]) un conjunto de ideas que constituyen un modelo que trata de explicar el funcionamiento del sistema inmune en su conjunto. En los siguientes párrafos se explicará en que consiste este modelo.

Este modelo representa una de las principales características del sistema inmune: es un sistema constituido por elementos independientes, es decir, es un sistema distribuido que no posee un elemento central que lo coordine y controle. Así que cuando en esta sucinta descripción de este modelo se usen oraciones como "el sistema inmune recibe información sobre el estado del organismo" se debe entender que parte o todos los elementos que lo constituyen reciben esa información; o cuando se diga algo así como "el sistema inmune cambia la estrategia de defensa", se refiere a que el conjunto de efectores que lo constituyen "deciden" y participan en ese cambio de estrategia.

Segel propone que para modelar el funcionamiento del sistema inmune sirve el atribuirle metas que debe alcanzar[4]. Estas metas pueden obstruirse en mayor o menor grado unas a otras, y algunas veces pueden hasta contraponerse, como es el caso de las metas 1) destruir a los patógenos y 2) *sin causar daño* al organismo. Hay mecanismos por los que algunos efectores del sistema inmune secretan substancias tóxicas que destruyen tanto a las células infectadas como a las células vecinas a ellas, por lo que la segunda meta no podrá cumplirse. Esta meta debe replantearse: 2) *causando el menor daño* posible al organismo. También propone que la primera respuesta del sistema inmune es siempre *sesgada* hacia un mecanismo

[4] No es una visión teleológica del sistema, simplemente es una forma de expresar las observaciones que se han hecho sobre él. Por ejemplo, la aseveración "el sistema inmune defiende al organismo contra los patógenos", es una forma de expresar que según múltiples observaciones y experimentos, el sistema inmune tiende a destruir a los patógenos internos y externos que aparecen en el, provocando de manera muy eficiente que en muchas ocasiones las concentraciones de los patógenos se mantengan muy bajas o lleguen a cero dentro del organismo.

predeterminado de defensa dependiendo del tipo de patógeno que irrumpa en el organismo. Este sesgo ha sido adquirido por el organismo durante su proceso evolutivo. Sin embargo, mientras se realizan el ataque del patógeno y la consecuente defensa del sistema inmune, éste último recibe valiosa información sobre el estado del organismo y sobre el desarrollo del combate. El estado del organismo se refiere al grado de destrucción sufrido por éste debido a los patógenos y al propio sistema inmune (se mide la destrucción debida a estas dos causas cada una por separado). La información sobre el desarrollo del combate se refiere a la concentración de los patógenos y de los efectores del sistema inmune. Esta información constituye lo que Segel llama una *retroalimentación difusa* (*diffuse feedback*), la llama así porque la posición de la *referencia*[5] de este sistema no está muy clara. Con base en esta información el sistema inmune decide si la respuesta inicial está funcionando, en caso afirmativo, continúa usando ese mecanismo, en caso contrario, cambia de mecanismo de defensa. Por ejemplo, si ha penetrado un patógeno que se introduce al interior de las células, el sistema inmune "escogerá" la estrategia de combatirlo usando la respuesta de la inmunidad con *células de por medio* (ver III.2), pero si a lo largo del combate, recibe información de que son los linfocitos B los que están teniendo más encuentros con los patógenos, y que éstos están siendo destruidos con *opsonosis* (ver III.2) de por medio, el sistema inmune cambiará poco a poco su estrategia para favorecer la respuesta inmune humoral, favoreciendo la multiplicación de linfocitos B y de linfocitos Th2 (linfocitos *T helpers 2*, ayudan a estimular el funcionamiento de los linfocitos B).

Las citocinas que son secretadas por las diferentes células del sistema inmune (tanto del primario como del adaptable) más que una red de comandos (que es como tradicionalmente se le considera), constituye una *red de información difusa* (DIN, *diffuse information network*). La DIN suministra información a las células del sistema inmune y cada una de ellas "decide" cómo actuar con base en esa información. Para que la DIN funcione cada célula del sistema debe avisar lo que está haciendo. A esto lo llama Segel el principio de "realiza y avisa" (*the do moo principle*). La información que mandan las células no está constituida por una sola citocina, sino por grupos de ellas. La información va repetida varias veces en este mensaje, esto es para evitar que la comunicación sea interferida por los patógenos. Las células del sistema inmune tienen varios tipos de receptores para las citocinas. La relación *tipo de citosina – receptor* no es uno a uno, un receptor puede captar varios tipos de citocina y una citocina puede unirse a varios tipos de receptor.

El sistema inmune va siguiendo el cumplimiento de sus metas por medio de la información que le manda la DIN y por información sobre el estado del organismo y de los patógenos.

Por último, en esta sucinta descripción se explicará cómo son premiados los efectores que han actuado bien en este modelo de sistema inmune. Segel propone que "el premio" de la buena actuación (multiplicación y/o aumento en la estimulación de los efectores que colaboraron en hacer algo bueno) es repartido a *todos* los efectores del *lugar* del organismo donde se ha actuado bien. De esta manera siempre se garantizará que se premiará a los efectores

[5] O *set point*, es el valor que se fija en los controladores de los sistemas de ingeniería, para que la variable medida se acerque lo más posible a él, por ejemplo, la temperatura que se desea en un punto de un sistema se fija como valor de la referencia o *set point* del controlador. El controlador se encarga de mover los valores de las variables manipulables para que el valor de la temperatura se acerque al de la referencia.

responsables del buen funcionamiento[6]. Segel comenta que siguiendo esta práctica en todos los lugares donde se haga algo bien terminará por premiarse más a los responsables reales de la buena actuación[7].

En la figura III.3 se trata de esquematizar lo que postula este modelo. Segel expresa las relaciones entre las concentraciones de efectores, patógenos y las substancias químicas como ecuaciones diferenciales ordinarias. Por ejemplo, en las ecuaciones (III.1-4) se describe una modelo que proponen Segel y Lev Bar-Or, 1999 [SL99] para describir la destrucción de patógenos y el daño infligido al propio organismo.

$$\frac{dN}{dt} = sE - g_N N \qquad \text{(III.1)}$$

$$\frac{dP}{dt} = rP - aEPN \qquad \text{(III.2)}$$

$$\frac{dE}{dt} = E(\mu_P P - g_E) \qquad \text{(III.3)}$$

$$\delta = \frac{1}{T}\int_0^T (h_P P + h_N N)dt \qquad \text{(III.4)}$$

donde:

E es la concentración de células efectoras,

N es la concentración de una substancia venenosa para matar patógenos secretada por las células efectoras,

s es el coeficiente de producción de la substancia venenosa N por parte de las células efectoras E,

g_N es el grado de decaimiento de N (se supone que se va descomponiendo con el tiempo),

P es la concentración de los patógenos,

r es el coeficiente de velocidad de reproducción de los patógenos,

a es el coeficiente de velocidad de eliminación de los patógenos,

m_P es el coeficiente de producción de las células de los efectores,

g_E es el coeficiente de velocidad de muerte de los efectores,

δ es la velocidad de daño promedio infligido al organismo,

h_P es el coeficiente de velocidad de daño al organismo debido a los patógenos,

h_N es el coeficiente de velocidad de daño al organismo debido a la substancia N.

[6] Aunque también se premiará a aquellos que no tuvieron que ver con la buena actuación. Esto recuerda el caso de la correlación espuria, en la optimización con algoritmos genéticos, cuando esquemas que no son los responsables de la buena aptitud de una cadena binaria son también premiados junto con aquellos esquemas que sí lo son.

[7] Esto implica que la forma en la que se considere esté partido el organismo y el número de esas partes sea crucial para el buen funcionamiento de ese sistema de retroalimenación difusa. Segel pone un ejemplo en el que considera dos recipientes o partes para simular cómo ayuda una inmunoterapia a curar una alergia en Segel, 2002 [Seg02].

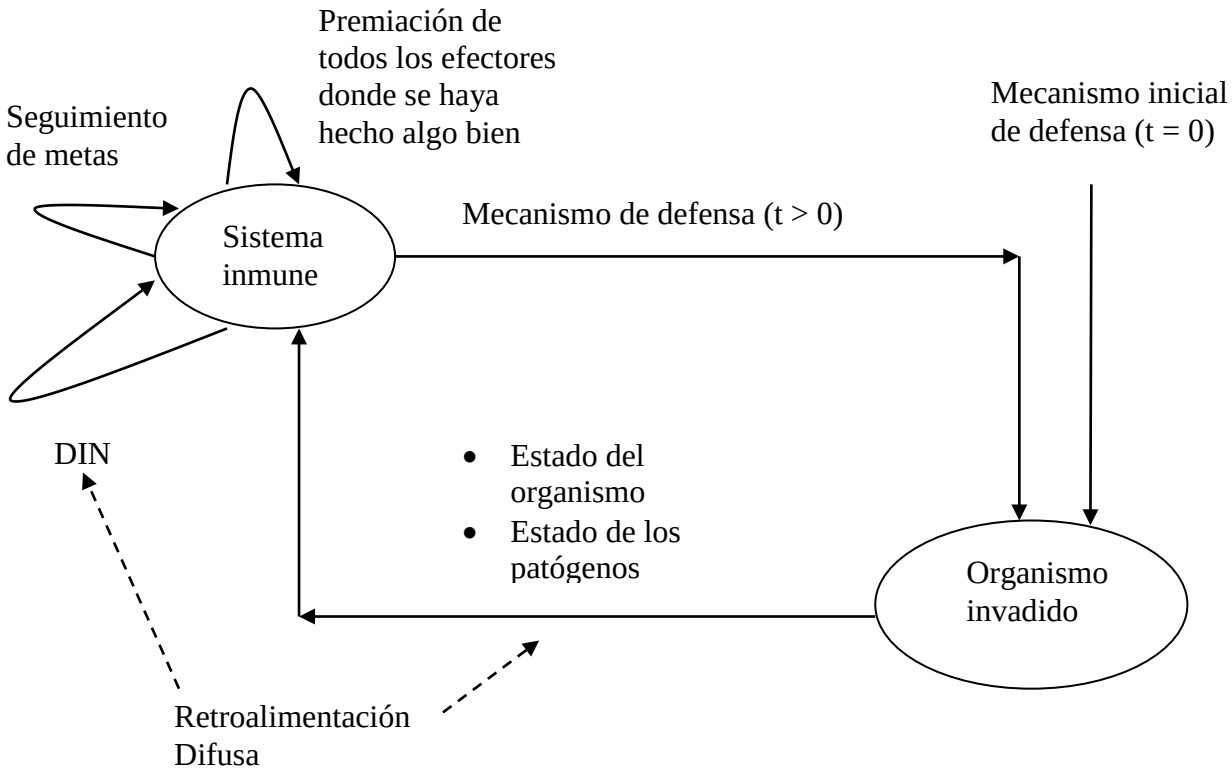


Figura III.3 Modelo de la Retroalimentación Difusa y la Red Difusa de Información. El sistema inmune (SI) responde inicialmente al ataque de un patógeno en el organismo con un mecanismo inicial que depende del tipo de patógeno. Con la información que recibe de la retrolimentación difusa (DIN), revisa cómo va el avance en el cumplimiento de sus metas y premia a *todos* los efectores que están en el lugar donde se ha hecho algo bien. Esto puede reafirmar el primer mecanismo que usó o ir cambiándolo con el tiempo.

Modelos para representar el comportamiento del sistema inmune auxiliado por medicamentos en un organismo infectado por retrovirus.

Perelson [Per02] muestra modelos para tratar de explicar los cambios de concentración con respecto al tiempo de células infectadas, de partículas virales, de linfocitos T, entre otros, en organismos infectados por el virus de inmunodeficiencia humana, virus de hepatitis B y virus de hepatitis C. Para hacer análisis cuantitativos, plantea sistemas de ecuaciones diferenciales ordinarias (del tipo mostrado en el anterior modelo). Los parámetros de esos modelos son obtenidos a partir de datos experimentales. Perelson alerta sobre el problema en la estimación de estos parámetros, "aunque los modelos y las técnicas de ajustes de datos permiten la estimación de los parámetros que caracterizan una infección viral, los modelos pueden estar mal, pueden ocurrir errores en el análisis, diferentes métodos de ajuste pueden dar diferentes respuestas y soluciones múltiples (mínimos locales) pueden confundir al método no lineal de los mínimos cuadrados o al método del máximo ajuste.". Este autor resalta que estos modelos han surgido en el ámbito de la investigación clínica y no el de la investigación básica en inmunología. Lo atribuye a la necesidad de lograr avances substanciales prácticos y a la disponibilidad de recursos en esa área. Estos modelos han resultado ser muy útiles.

Un modelo de la respuesta inmune humoral para estudiar efectos negativos de los recuerdos asociativos de la memoria del sistema inmune.

Smith y colaboradores [SFP99], estudian el hecho (ya documentado por lo menos desde el siglo XIX) de que para producir inmunidad contra una enfermedad grave se puede inocular en el organismo un patógeno similar al que la provoca (esto es, con al menos un epitope –ver III.2– igual o similar a uno del patógeno que provoca la enfermedad grave), pero que cause otra enfermedad mucho más benigna. El sistema inmune monta respuesta contra el patógeno de la enfermedad benigna, lo controla y genera células de memoria para "recordar" la forma del atacante. Si posteriormente el patógeno de la enfermedad grave irrumpe en el organismo, las células de memoria "recuerdan" el epitope al que ya se habían enfrentado e inmediatamente repelen al agresor antes de que cause graves daños. A este fenómeno se le conoce como *recuerdo asociativo* (*associative recall*). Así como este fenómeno es benéfico en algunos casos (como el que se acaba de describir) en otros no sólo no ayuda sino que perjudica en el combate contra los patógenos. Esto ocurre cuando no deja que una nueva vacuna genere anticuerpos contra una enfermedad. Supóngase que un individuo ha sufrido una enfermedad o ha sido vacunado contra una enfermedad producida por un patógeno *A*, luego se le pone una vacuna con epitopes *B* muy similares a los del patógeno *A*, para protegerlo contra un patógeno *C* que tiene epitopes muy parecidos a los de *B* pero ya no tan parecidos a los de *A*. Puede ocurrir que el recuerdo asociativo impida que el sistema inmune monte respuesta contra los epitopes de la vacuna (porque las células de memoria contra *A* inmediatamente eliminan a los epitopes de *B* inoculados al organismo) y por tanto no se generen células de memoria contra los epitopes de *B*, entonces se corre el riesgo de que si se presenta el patógeno *C* en el organismo, provoque la enfermedad si las células de memoria contra el patógeno *A* no reconocen a los epitopes de *C* por no ser lo suficientemente parecidos a éstos.

Para estudiar la aparición de este último fenómeno Smith y colaboradores [SFAP99] diseñaron un modelo que es una simplificación de la respuesta inmune humoral. Contiene células B, células plasmáticas, anticuerpos, células B de memoria y antígenos (ver sección anterior). Las células T auxiliadoras tipo 2 son simuladas de manera implícita ya que su efecto está disponible cada vez que se requiera. El modelo está *basado en agentes* (*agent based*), cada componente se representa como una entidad aparte (un agente). Los receptores y los anticuerpos se representan como cadenas de símbolos, al igual que los antígenos. Cuando los receptores de los linfocitos B se complementan con un antígeno, los linfocitos B se multiplican permitiendo que cada uno mute con respecto a su padre, simulando de esta manera la selección clonal y la hipermutación somática (ver sección anterior). Estos linfocitos resultantes de la multiplicación pueden diferenciarse, de manera determinada por el azar, en células plasmáticas o en linfocitos B de memoria. Las células plasmáticas producen anticuerpos que pueden, al azar, unirse con los antígenos, cuando un antígeno tiene un número mayor a un límite predeterminado de anticuerpos pegados, es removido del sistema, simulándose así la opsonosis (ver III.2). Los receptores de la célula B, los anticuerpos y los antígenos están constituidos por cadenas de 20 símbolos. Las cadenas están escritas con un alfabeto de cuatro símbolos. La afinidad se mide por el número de símbolos que se complementan entre dos cadenas (de receptor de linfocito B y antígeno o de anticuerpo y antígeno). En este modelo se dice que dos cadenas son afines si se complementan en 15 o más símbolos. Afinaron sus parámetros para que el programa respondiera como el sistema inmune de los vertebrados. Mediante pruebas comprobaron que cualitativamente sí respondía como se

esperaba. Después de esto llevaron a cabo varias corridas para estudiar el fenómeno de los efectos negativos del recuerdo asociativo. Los resultados se muestran en [SFAP99].

A.2) Modelos para simular el comportamiento del sistema inmune basados en modelos de la red inmune.

Estos modelos explican por medio de un equilibrio metadinámico entre anticuerpos y anti-anticuerpos la existencia de la memoria del sistema inmune así como la tolerancia, sin embargo, todavía no se ha podido desarrollar un modelo de este tipo que imite satisfactoriamente la gran cantidad de fenómenos relativos al sistema inmune.

Modelos de la red inmune.

Fue Jerne [Jer74] quien propuso la hipótesis de la red inmune. Según esta hipótesis, las regiones variables de los anticuerpos y de los receptores de los linfocitos se reconocen unas a otras, de tal manera que uno de estos elementos actúa tanto como antígeno de un elemento A como un anticuerpo de otro B. Otros investigadores han desarrollado modelos matemáticos de la red inmune para tratar de simular el comportamiento del sistema inmune natural. Un acercamiento interesante es el tratar de explicar la red inmune como dos conjuntos de efectores, sobre esto hacen un análisis Stewart y Carneiro [SC99]. Un conjunto de estos efectores constituye el sistema inmune central, CIS (*central immune system*), el otro el sistema inmune periférico, PIS (*peripheral immune system*), estos conjuntos son exclusivos entre sí. Los efectores que forman parte del CIS constituyen propiamente la red inmune y explican la memoria inmunológica y la tolerancia, los que forman parte del PIS no están en una red, son independientes y se encargan de reconocer a los antígenos, al hacerlo sufren expansiones clonales. Es al tratar de explicar la interfase entre el CIS y el PIS que tiene problemas este modelo. Cuando simulan con los modelos de diferentes autores el CIS tiende a ocupar un porcentaje demasiado grande de todos los efectores, dejando un porcentaje muy bajo para el PIS, lo cual no es lógico. Los modelos sobre de la red inmune sólo toman en cuenta las interacciones de los linfocitos B y en algunos casos de los anticuerpos. Stewart y Carneiro [SC99] proponen un modelo que toma en cuenta de manera explícita las interacciones de los linfocitos T con los B y obtienen resultados más acordes con el comportamiento observado de sistema inmune. Sin embargo estos autores reconocen que "la dificultad de establecer una relación productiva entre la teoría y la experimentación es una marca de todos los acercamientos al sistema inmune utilizando el modelo de la red, y es tal vez la razón por la cual, en el presente, la comunidad inmunológica ve a las redes idiotípicas con escepticismo".

Vemos pues que este acercamiento a la simulación del sistema inmune no da buenos resultados. Sin embargo, se dan casos de modelos biológicos que no son bien vistos desde el punto de vista de la biología, que sí dan buenos resultados en el campo de la computación, tal es el caso, por ejemplo, de la evolución lamarckiana (escaladores) y de los modelos de la red inmune. En la parte de modelos computacionales del sistema inmune para resolver problemas ajenos a él, se verá que se han usado modelos de la red inmune con buenos resultados para resolver problemas prácticos.

B) Modelos inspirados en el sistema inmune para resolver problemas ajenos a él.
La finalidad de este tipo de modelos no es simular el comportamiento del sistema inmune, sino basarse en algunos de sus mecanismos para aplicarlos en la solución de problemas en otras áreas distintas de la inmunología. A este tipo de algoritmos pertenecen los llamados sistemas inmunes artificiales (AIS, *artificial immune systems*) a continuación se hablará sobre ellos.

Sistemas Inmunes Artificiales
El tipo de algoritmos que reciben este nombre ha variado a lo largo de los pocos años que tiene de existir esta área del conocimiento. Por ejemplo en el libro editado por Dasgupta y publicado en 1999 [Das99] titulado "Sistemas Inmunes Artificiales y sus Aplicaciones" (*Artificial Immune Systems and their Applications*) aparecen varios algoritmos cuyo objetivo es el simular el funcionamiento del sistema inmune. Sin embargo en el libro de de Castro y Timmis publicado en 2002 [DT02] se hace énfasis en aclarar que "la pura simulación computacional del sistema inmune no califica como AIS". De Castro y Timmis proponen una definición para sistema inmune artificial ([DT02]p.58):

> "Los sistemas inmunes artificiales (AIS) son sistemas adaptables inspirados por la inmunología teórica y funciones inmunes observadas, principios y modelos, que son aplicados en la solución de problemas"

Aclaran estos autores ([DT02] p. 59) que "atribuir simplemente 'terminología inmunológica' a un sistema dado no es suficiente para caracterizarlo como un AIS. Por ejemplo, nombrar un conjunto de patrones de entrada antígenos y a otro conjunto de patrones como anticuerpos no lo califica para ser un AIS. Debe haber un mínimo de inmunología involucrado, tal como un modelo para hacer coincidencia de patrones, un principio inmune incorporado tal como algoritmo de selección clonal y/o selección negativa, o una red inmune". Por último, agregan ([DT02] p.59) que "Otro tipo de sistemas que pertenecen a la clase de sistemas inmunes artificiales pero que no fueron cubiertos en la definición 3.4 [la definición que aquí se ha reproducido unas cuantas líneas arriba] son los híbridos de metodologías de inteligencia computacional (*e.g.*, algoritmos genéticos, redes neuronales artificiales y sistemas difusos) con modelos teóricos de inmunología." Aquí se tomará en cuenta esta definición y estas aclaraciones para designar un algoritmo con el nombre de *sistema inmune artificial*.

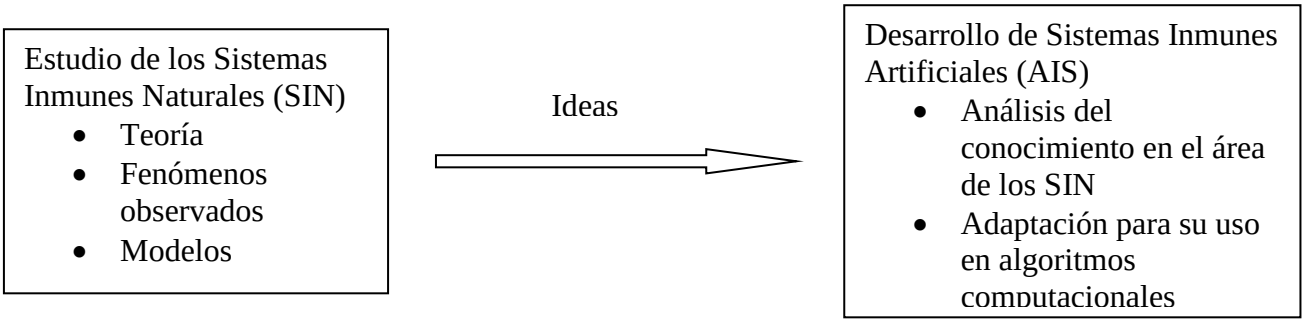


Figura III.3. Flujo de ideas SIN→AIS. El estudio de los sistemas inmunes naturales ayuda en el desarrollo de programas aportando ideas para el diseño de sistemas inmunes artificiales.

Enriquecimiento de la inmunología con el advenimiento de los sistemas inmunes artificiales.
No sólo el flujo de ideas va en la dirección del estudio de los sistemas inmunes naturales hacia el desarrollo de sistemas inmunes artificiales como podría esperarse (ver figura III.3).

El hecho de que varios investigadores en el área de cómputo estén estudiando el conocimiento generado y la información obtenida por los investigadores de los sistemas inmunes naturales, provoca que los primeros puedan desarrollar hipótesis y teorías que en algunos casos[8] ayuden a entender el funcionamiento de los sistemas inmunes naturales y ayuden a fijar rutas para nuevas investigaciones (ver figura III.4).

Un ejemplo en el que el flujo de ideas se da en ambas direcciones (AIS ↔SIN) es el artículo de Forrest y Hofmeyr, *Engineering an Immune System* [FH01], en el que se propone un sistema inmune artificial para proteger a una red de cómputo de programas no autorizados provenientes del exterior y de instrucciones no permitidas de software trabajando mal y sí autorizado en la red. Sobre él se hablará en el siguiente punto.

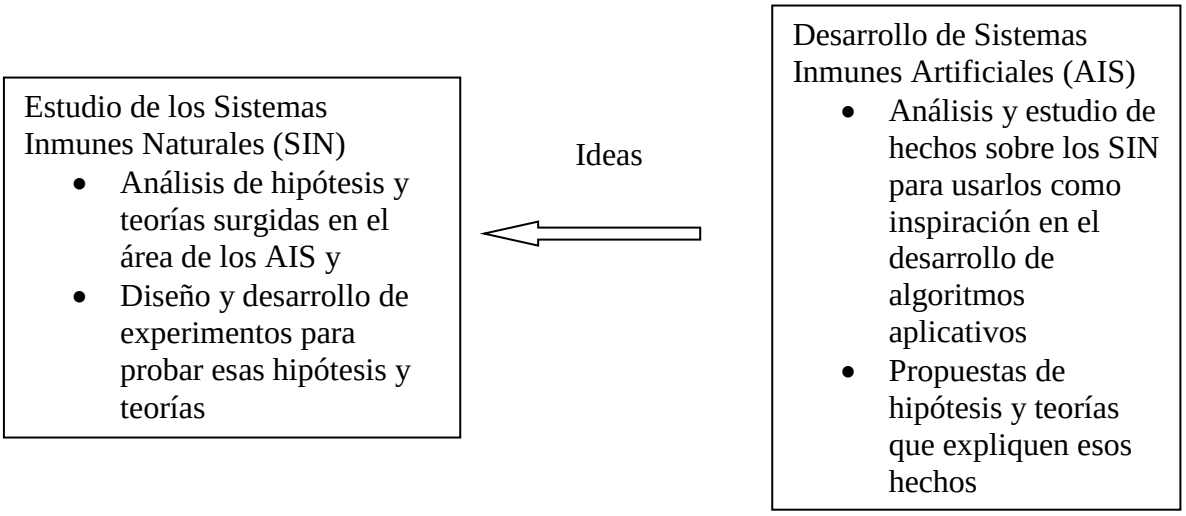


Figura III.4. Flujo de ideas AIS→SIN. El que los investigadores en el área de los AIS estudien los fenómenos que se presentan en los SIN hace que, eventualmente, puedan surgir nuevas hipótesis y teorías que ayuden explicar el funcionamiento de los SIN y sirvan para fijar nuevos rumbos en las investigaciones sobre los SIN.

B.1 Modelos de computadora basados en la selección clonal.

Un sistema inmune artificial para proteger una red de computadoras.
Forrest y Hofmeyr [FH01] proponen el desarrollar un sistema inmune artificial (AIS) para una red local de computadoras (LAN, *local area network*) con el objetivo de defenderla de intrusos y de programas internos que por mal funcionamiento traten de hacer operaciones no permitidas. Su trabajo puede verse como una aportación tanto a las ciencias de la computación

[8] Tal vez no muchos por cierto, debido a que los modelos usados en el área de los AIS son muy simplificados o sólo se inspiran muy de lejos en los SIN. Pero el hecho importante es que se ha incrementado el número de investigadores interesados en tratar de entender cómo y por qué los SIN funcionan como lo hacen

como a la inmunología, ya que ese AIS les servirá para cumplir el objetivo mencionado, pero además para observar que mecanismos debe aplicar ese AIS para cumplir su cometido y compararlo con los mecanismos que usa el sistema inmune natural (SIN) humano, y tratar de entender (y aportar nuevas hipótesis sobre) el porque de su constitución y su funcionamiento (*i.e.*, porque es como es y porque funciona como funciona).

Su AIS detecta intrusos en la LAN por la forma en la que manejan la información. En esto se diferencia de los buscadores de virus, que revisan los archivos para detectar cambios de su tamaño. El SIN opera de manera similar a ese AIS propuesto, revisa las expresiones de los genes (proteínas), no los genes mismos para detectar patógenos.

Los detectores de su AIS (lo que vendría a ser en el SIN las células del sistema inmunitario y los anticuerpos) son cadenas binarias que se comparan contra los mensajes que se transmiten por la red. Los mensajes constan de tres partes, la computadora a la que va el mensaje, la computadora de la que proviene y la información transmitida. Por eso los mensajes reciben el nombre de tripletas. Los detectores se comparan con los mensajes y en base a reglas basadas en umbrales se decide si hay coincidencia entre ellos o no. Si hay coincidencia, significa que la tripleta contiene información no permitida en la red. Cuando se da este caso, se almacena este receptor de manera permanente para que sirva para detectar al mismo antígeno si se presenta otra vez. En esto el AIS propuesto se parece al modelo de la selección clonal, donde las células de memoria viven por mucho tiempo para crear la inmunidad contra la enfermedad (ver sección III.2). Los detectores trabajan como en el SIN, de manera independiente y distribuida. Los detectores deben entrenarse antes de ponerse a trabajar para que no hagan coincidencia con tripletas que contengan información permitida en la red (lo propio). Los autores se percataron que los detectores daba muchos falsos positivos (i.e., coincidían con tripletas permitidas muchas veces) por lo que tuvieron que modificar su primera propuesta, antes de actuar contra una tripleta con la que ha coincidido un detector, se pide la confirmación de un operador humano, si después de un tiempo (24 hrs.) no hay tal confirmación, no se actúa contra la tripleta. Esto es muy similar a la actuación de los linfocitos T auxiliadores (T *helpers*) de tipo 1 o 2, que estimulan respectivamente a los linfocitos B (para activar la respuesta humoral –ver sección III.2) o a los linfocitos T (para la respuesta con células de por medio –ver sección III.2), sin ellos la respuesta no se dispara. También se percataron que el aprendizaje de los detectores era muy lento, por lo que debían guardar en un lugar muestras de las cadenas de *lo propio* para después entrenar con ellas a los detectores para hacer más rápido su entrenamiento. Los autores mencionan que esto es muy similar a lo que ocurre en el timo donde hay muestras de lo propio y se hace la selección negativa de los linocitos T (ver III.2). El autor de estas líneas agrega que es lo mismo que ocurre con la selección negativa de los linfocitos B en la médula ósea (ver III.2). El sistema detector perfecto, debe tener un detector por cada antígeno, esto es, un detector por cada patrón distinto de *lo propio*. Esto implica la existencia de un número demasiado grande de detectores. Para resolver este problema, se debe usar la *generalización*, esto es que un receptor sea capaz de coincidir con diferente epitopes (ver *reactividad cruzada* en el punto III.2). Para que haya suficiente definición entre lo propio y lo ajeno la variedad de esos receptores debe ser todavía muy grande, es decir, debe haber una cantidad de distintos receptores muy grande. Una solución a este otro problema es que sólo una fracción de esos distintos receptores esté presente a la vez. Esto provoca que haya *agujeros*, en el sistema, es decir que no esté presente un receptor que podría detectar al patógeno en un momento dado. Los patógenos evolucionarían para explotar estos *agujeros*. La solución a este problema es ofrecer una

protección dinámica cambiando la muestra de receptores continuamente, esto se logra haciendo que los detectores tengan un tiempo limitado de vida y que su lugar sea ocupado por nuevos receptores.

Existe otro problema más de fondo con la *generalización*, al tener que aceptar que un detector coincida con varios antígenos, el sistema tendrá *agujeros* para todo aquel antígeno para el que todos los receptores que coincidan con él también coincidan con *lo propio*. Si la selección negativa funciona bien, ninguno de estos receptores debe ponerse a trabajar (de lo contrario se corre el riesgo de que se desencadene una enfermedad autoinmune, ver sección III.2), quedando un *agujero* para este antígeno. Una solución para esto es usar varias codificaciones a la vez, de manera tal que estos *agujeros* no coincidan para los mismos antígenos en las diferentes representaciones. Esto se logra en el AIS propuesto por los autores, fijando al azar un número de permutaciones distinto para cada receptor, cada vez que se vaya a comparar una tripleta con un receptor, se le deben aplicar las permutaciones correspondientes a la tripleta y después compararla con el receptor. Forrest y Hofmeyr [FH01] sugieren que lo mismo hace el SIN al usar el MHC (ver sección III.2) de diferentes tipos. Al embonar un antígeno del patógeno a una molécula de MHC para que luego el complejo antígeno-MHC sea reconocido por un receptor de un linfocito T, lo que hace el SIN es cambiar la codificación para tratar de resolver el problema de este tipo de agujeros.

Como comentan Forrest y Hofmeyr [FH01], el sistema inmune es tan complejo que algunos investigadores creen que nunca se llegará entender a detalle cual es la función de cada uno de sus componentes. Por otra parte, también hay autores que sostienen que la evolución del sistema inmune no tiene por objeto el optimizarlo, sino simplemente el obtener un sistema inmune que permita la supervivencia y un desempeño aceptable del individuo. Si es un sistema que no está optimizado, esto podría implicar que muchos de sus componentes y mecanismos son redundantes. Después de todo, existen animales con sistemas inmunes mucho más sencillos que parecen funcionar bien. A esto hay que agregar además que hay quienes opinan que la función del sistema inmune no está clara del todo. Hay quienes opian que su función principal es el distinguir entre *lo propio* y *lo ajeno* (*self* y *non-self*), otros creen que su función principal es advertir del peligro y otros más que su función es reparar daños sufridos por el organismo.

Con el AIS que desarrollaron los autores quedó claro que fue necesario usar muchos de los mecanismos (más de los que los autores suponían tendrían que usar al inicio de su investigación) usados por el SIN. Lo que da indicios de que los mecanismos usados por él son necesarios y no son redundantes. Además queda demostrado que muchas de sus estrategias son de defensa contra los patógenos, así que se puede concluir eso, que el SIN es un sistema de defensa. Con esto pueden concluir los autores que la visión teleológica de este sistema sí da frutos importantes en su estudio.

Modelos computacionales basados en la evolución de bibliotecas de anticuerpos.
Diferentes investigadores han explorado la recombinación somática de genes de bibliotecas que sirven para crear las partes variables de los receptores de los linfocitos B y T (ver punto III.2), por ejemplo, Perelson, Hightower & Forrest, 1996 [PHF96]; Hightower, Forrest & Perelson 1996 [HFP96]; Oprea & Forrest, 1998 [OF98]; Oprea & Forrest, 1999 [OF99]. Para llevar a cabo estos estudios han representado al sistema inmune de la siguiente manera: los anticuerpos (o receptores de los linfocitos B) son cadenas binarias de longitud L, los antígenos

de los patógenos son también cadenas binarias de longitud L, las bibliotecas con las que son construidos los anticuerpos son cadenas binarias de longitud $L \times A$, así que cada una de estas bibliotecas puede producir hasta A diferentes anticuerpos. Los primeros que representaron con cadenas binarias los anticuerpos y los antígenos fueron Farmer, Packard y Perelson, 1986 [FPP86]. El objetivo de estos investigadores es el estudiar al sistema inmune como un sistema de reconocimiento de patrones. Como lo hace resaltar Forrest *et al.*, 1993 [FSJP93], el sistema inmune es un muy buen reconocedor de patrones, ya que debe distinguir *lo ajeno* de *lo propio*, se calcula que el sistema inmune es capaz de identificar hasta 10^{16} [Inm78] antígenos distintos (propios y de patógenos), el cuerpo humano tiene alrededor de 10^5 proteínas distintas[9]. Esto significa de entrada que el sistema inmune puede distinguir un conjunto de 10^5 en un conjunto de 10^{16} elementos. Agregan estos autores que en el cuerpo humano hay presentes a la vez alrededor de 10^7 receptores distintos.

Los investigadores experimentan para ver que mecanismos son necesarios para obtener un *repertorio completo* de anticuerpos (este término se refiere a obtener un conjunto de anticuerpos tal que un antígeno de patógeno sea reconocido por lo menos por un anticuerpo) para diferentes conjuntos de antígenos.

Los algoritmos que usan estos investigadores se basan en tomar una muestra al azar de antígenos de patógenos de un conjunto que ellos definen de antemano y evalúan la aptitud de cada biblioteca tomando una muestra al azar de sus anticuerpos. Se compara cada anticuerpo con cada no de los antígenos de la muestra, algunos autores usan como medida la distancia de Hamming entre el anticuerpo y el antígeno como medida de su afinidad (a menor distancia mayor afinidad), otros, usan la complementariedad del anticuerpo con el antígeno, esto es la sumatoria de la operación XOR entre los bits de las dos cadenas, entre mayor valor tenga esta sumatoria mayor afinidad hay entre las moléculas. En base a la calificación obtenida por los anticuerpos se califica a la biblioteca.

Además de la recombinación somática, estos investigadores se han visto obligados a usar en sus modelos computacionales también la *mutación somática* (mejor conocida como *hipermutación somática*) para obtener mejores resultados. Éste es el proceso mediante el cual los linfocitos B al reproducirse (asexualmente) transmiten su código genético a sus descendientes con una mutación (*mutación puntual, point mutation*) en la parte del genoma que codifica a la región variable de sus receptores (ver sección III.2). Con esto se mejora la aptitud de los receptores (o anticuerpos). La hipermutación somática no altera los genes, así que es una evolución somática, es la evolución darviniana la que altera los genes, es decir esa es una evolución genética. Algunos de estos investigadores ([PHF96]) han reportado que se les ha presentado el efecto Baldwin en algunos de sus experimentos (ver II.5), con esto el aprendizaje obtenido durante la evolución somática (a veces) ayuda a la evolución genética.

Combinando estos dos mecanismos los autores han obtenido buenos resultados, sus repertorios de anticuerpos tienden a ser completos.

[9] Por supuesto, estos autores no están tomando en cuenta la producción por medio de bibliotecas de los receptores de los linfocitos que son ¡10^{16} proteínas distintas!, sino que usan para sus cálculos la vieja regla de un gen para cada proteína.

Hardware Tolerante a Fallas por Medio de Sistemas Inmunes Implementados con Hardware

Un grupo de investigadores (Bradley, Ortega-Sánchez & Tyrrel, 2000 [BOT00]; Bradley & Tyrrell, 2000 [BT00], [BT00b]; Bradley y Tyrrell, 2001 [BT01]; Bradley y Tyrrell, 2002 [BT02]; Canham & Tyrrell 2002) se han dedicado al estudio de la forma de diseñar hardware a prueba de fallas implementando con hardware un sistema inmune. Como ejemplo de hardware utilizan una máquina de estados finitos (FSM, *finite state machine*), aunque sostienen que podrían diseñarse sistemas inmunes implementados con hardware para otro tipo de máquinas [BT01]. Entre otras características, el hardware del sistema inmune se encuentra aparte del hardware de la FSM, este último sólo debe recibir como información cada salto que da la FSM de un estado a otro. Esto es una ventaja, ya que no se debe rediseñar la FSM. Previamente (fuera de línea) se han almacenado *muestras de buen funcionamiento* de de la FSM en memorias accesibles por su contenido (CAM, content acces memory). Estas *muestras de buen funcionamiento* son saltos *correctos* de un estado *correcto* de la FSM a otro de sus estados también *correcto*. Las fallas se presentan cuando ocurre un salto *incorrecto* de un estado *correcto* a otro estado *correcto* de la FSM (*i.e.*, los estados son parte de la FSM pero el salto de uno a otro es incorrecto ya sea porque se tomó como condición para dar ese salto el valor de una variable de entrada que en el diseño original no controla el salto, o porque se tomó en cuenta el valor de una variable que sí debería provocar el salto pero no con el valor que lo provocó) o cuando se salta a un estado *incorrecto* (*i.e.*, cuando se salta a un estado que no esta contemplado en el diseño de la FSM). La parte del hardware que vendría a ser las veces de los linfocitos no debe *reconocer* los saltos correctos, pero sí los incorrectos. El hardware que usan los autores consta de compuertas que pueden modificarse en línea (FPGA, *field programmable gate arrays*), una vez detectada la falla se debe proceder a corregirla reprogramando el arreglo de compuertas.

Este sistema inmune no requiere de software para funcionar, todo la hace a nivel de hardware. Los autores reportan que han usado FSM de pequeño tamaño y que por tanto con un número razonable de muestras de buen funcionamiento pueden resolver todas las fallas que se presentan, sin embargo, advierten que para FSM de mayor tamaño, ese tamaño de muestras podría crecer demasiado. A este concepto de diseñar hardware con un sistema inmune para detectar fallas le llaman *inmunotrónica* (*immunotronics* = electronics + immune system concepts). Otro concepto interesante que introducen es el de *embriónica* (*embrionics*) ([BT00], [CT02]), está fundamentado en los FPGAs. Se inspira en el crecimiento de una célula germinada (huevo germinado) de las especies de individuos pluricelulares que se reproducen sexualmente. La célula germinada se multiplica y así se va formando el embrión. Cada nueva célula es capaz, al menos en teoría, de realizar cualquiera de las funciones que se requieran, ya que tiene una copia completa del código de funciones del individuo (como las células naturales la tienen en el DNA en el núcleo celular). La función de cada nueva célula está determinada por las coordenadas de sus células vecinas y por sus propias coordenadas. Si se presenta una falla en alguna célula del arreglo, manda una señal a las otras células, se desactiva ella y a otras células vecinas a ella, y en base a sus coordenadas cada una de las células restantes se reprograman para realizar las tareas que las de las células desactivadas dejaron de hacer. Si hay un número suficiente de células, el sistema continúa trabajando realizando sus funciones con el sistema de células restantes con las tareas repartidas de otra manera. El sistema encargado de detectar las fallas es un sistema inmune implementado en hardware.

B.2 Modelos de computadora basados en modelos red inmunitaria

Como ya se ha comentado arriba, los modelos de las redes idiotípicas no tiene buena aceptación en la comunidad de inmunólogos que tratan de simular el comportamiento del sistema inmune, pero ha sido usado con éxito en la solución de problemas ajenos al sistema inmune.

Un Sistema Inmune para Escoger los Movimientos de un Robot Autónomo

Watanabe, Ishiguro y Uchikawa [WIU99] proponen usar una abstracción de una red idiotípica para dar autonomía de movimiento a un robot mientras cumple una tarea. Los antígenos son la información que en cada momento recibe el robot por medio de sus sensores. Estos antígenos afectan a los anticuerpos que están formados por 1) una instrucción del tipo *si <situación> entonces <acción>* y por 2) un apuntador a otro anticuerpo. A la primera parte del anticuerpo representa al *paratope* y la segunda al *idiotope*. Cada anticuerpo tiene almacenada además una medida que representa su concentración. Esta concentración es incrementada por los estímulos que recibe (embonar con alta afinidad con un antígeno o con un idiotope) y es disminuida por los estímulos inhibitorios que recibe (que otro anticuerpo coincida con alta afinidad con su idiotope, que se acabe su periodo de vida). La instrucción del anticuerpo con mayor concentración es la que se ejecuta. En sus experimentos, las relaciones entre los idiotopes y los paratopes fueron fijadas al principio de manera manual, más adelante diseñaron mecanismos para que dichas relaciones se establecieran de manera automática. El caso especial que ellos estudian es el hacer que un robot móvil autónomo recoja basura tirada alrededor y a diferentes distancias de una base donde debe recargar sus baterías cada vez que estén bajas de energía. También el bote de basura se encuentra en esa base, el objetivo es que el robot recoja la basura de manera autónoma (sin que nada externo a él le indique qué hacer) sin que se quede sin energía. Los autores usaron un pequeño robot en el que programaron este sistema y al que pusieron a recoger lo objetos identificados como basura. En su sistema el robot pierde energía por desplazarse, por chocar contra la basura o contra las paredes que delimitan el campo donde debe trabajar. También el robot debe evitar la sobrecargar sus baterías, esto es que vaya a cargarlas cuando tienen su carga alta. Los autores reportan que obtuvieron buenos resultados en esta tarea.

Un Sistema Inmune para Detectar Fraudes en Préstamos Hipotecarios

Hunt y colaboradores [HTCNK99], desarrollaron un sistema basado en una red inmune. Esta no es una red idiotípica, como en al caso anterior. Cada nodo de la red tiene un patrón de reconocimiento (una cadena de símbolos) y *m* arcos para unirse a otros tantos nodos de la red. La información introducida se compara con los patrones de reconocimiento de nodos escogidos al azar. Si el grado de similitud sobrepasa un umbral, el nodo se multiplica con ligeras mutaciones y queda unido al nodo que le dio origen, de esta manera se incrementa la posibilidad de que haya mayor coincidencia para posteriores entradas. Cuando el nodo tiene ocupados sus *m* arcos, se elimina el que tenga menor coincidencia con la entrada y en su lugar se introduce el nuevo nodo. De esta manera se van formando cúmulos en la red que identifican ciertos tipos de entrada. La información que recibe la red es sobre datos de clientes que piden una hipoteca, tal información incluye nombre, dirección, puesto en el trabajo y salario. Los autores comentan que hay dos tipos de fraudes principalmente, aquéllos que falsean información tal como su trabajo, puesto y sueldo para pedir el préstamo hipotecario (son personas que tienen la intensión de pagar) y aquellos que falsean información personal tal como el nombre (son personas que no tienen la intención de pagar). La red va a ir formando cúmulos de diferentes tipos de préstamos, una vez que se formen y se identifiquen aquellos

cúmulos correspondientes a solicitudes fraudulentas se podrá identificar las posteriores solicitudes de ese tipo al ver que se dirigen a esos cúmulos.

Capítulo IV
Los Algoritmos Propuestos

En el capítulo anterior se ha descrito a grandes rasgos el funcionamiento del sistema inmune natural de los vertebrados con mandíbula (SIN) y se ha hablado sobre modelos computacionales del sistema inmune inspirados en él. En este trabajo se proponen tres algoritmos para resolver instancias del NLPP (*non linear programming problem* –ver capítulo I), dos de ellos están inspirados en mecanismos del sistema inmune, otro más es un caso extremo de uno de ellos. Al primero se le ha llamado aquí algoritmo genético para la *Evolución de Bibliotecas Generadoras de Anticuerpos* (EBGA)[1], toma como principal elemento de inspiración la *recombinación somática* para la formación del gene de la región variable de los receptores de los linfocitos. El EGA, un algoritmo que es un híbrido de un algoritmo genético y un escalador, hace evolucionar las bibliotecas. El segundo algoritmo es un caso extremo del primero. Consiste en hacer evolucionar a la región variable de los receptores en lugar de hacer evolucionar a las bibliotecas de las que se forma el gene de esa región. El algoritmo que hace evolucionar a las regiones variables de los receptores es también el EGA, por esta razón a este algoritmo se le ha llamado el EGA-DR. El tercero, llamado algoritmo evolutivo basado en la evolución de anticuerpos y autoanticuerpos (EAA), toma como elementos principales de inspiración la **m**aduración de la **a**finidad con **e**dición de los **r**eceptores, por eso también se le llama aquí EAA-MAER. En este capítulo se describe cada uno de estos algoritmos.

IV.1. El algoritmo genético basado en la evolución de bibliotecas generadoras de anticuerpos (EBGA).

Este algoritmo se inspira en la forma en la que se ensambla el **g**en de los **r**eceptores superficiales de un linfocito (GR). Cada vez que se forma un nuevo linfocito en la médula ósea se ensambla un GR que formará parte de su genoma (ver sección III.2). Aquí se está denominando *biblioteca* a aquella parte del genoma del organismo que contiene las partes necesarias para formar los GRs. De hecho sólo la parte del GR que expresa la parte variable de los receptores es la que en este trabajo se utilizará, por tanto, de aquí en adelante cada vez que se mencione el GR en realidad se estará haciendo mención sólo a esa parte a menos que se indique otra cosa.

IV.1.1 Las bibliotecas y los anticuerpos.

Las bibliotecas generan GRs, y los GRs expresan receptores o anticuerpos. El modelo no pierde nada conceptualmente y si gana en simplificación si se postula que las bibliotecas generan directamente los anticuerpos[2]. Por esta razón de aquí en adelante se tratará a las bibliotecas como generadoras de anticuerpos. En este trabajo se considera que una biblioteca se divide en *partes*, el número de partes de una biblioteca es igual al número de partes en que se divide (arbitrariamente) el anticuerpo. Cada parte contiene una o más *variaciones*. Para ensamblar un anticuerpo se debe tomar al azar una de las variaciones de la parte1, otra de la parte 2 y así sucesivamente hasta que se ha tomado una variación de cada una de las partes. En

[1] Está basado en un algoritmo similar propuesto por Hightower, Forrest y Perelson [HFP95]. Ellos lo usan para estudiar la evolución de la organización que surge en las bibliotecas de genes del sistema inmune.

[2] Por supuesto, desde el punto de vista bioquímico sí hay deferencia, los genes están hechos de DNA y los receptores de proteínas. Para estos modelos lo que importa es sólo la información, por eso es que el mismo gen equivale a aquello que expresa (el receptor).

el EBGA todas las partes tienen igual número de variaciones. Entonces una biblioteca se puede visualizar como una matriz de dimensiones $n_p \times n_v$, siendo n_p el número de partes y n_v el número de variaciones por parte. Un anticuerpo cualquiera se forma tomando un elemento al azar de cada uno de los renglones de dicha matriz. Hay que recalcar que no se puede intercambiar las posiciones de cada parte. Esto es, una variación de la primera parte de una biblioteca formará la primera parte del anticuerpo, otra de la segunda formará la segunda parte del anticuerpo, y así sucesivamente.

IV.1.2 Qué representan las bibliotecas y los anticuerpos.
Un anticuerpo es una representación de un punto en el espacio de búsqueda *S* (ver capítulo I). Tal representación es una cadena binaria. Entonces una biblioteca puede visualizarse como un productor de puntos en *S* representados en forma de cadena binaria. Es importante aclarar que el número de partes de una biblioteca se fija arbitrariamente (al menos así se hizo en este trabajo), de manera independiente de la dimensionalidad de *S*. Tanto el número de partes como el de variaciones por parte deben ser fijadas por el usuario.

IV.1.3 Cómo lleva a cabo el EBGA la búsqueda del óptimo.
El mecanismo por medio del cual se lleva a cabo la búsqueda del óptimo es hacer evolucionar las bibliotecas con un algoritmo genético que invoca a veces un escalador por mutación aleatoria ([Kur99]pp. 215-219). Esto equivale a llevar acabo la evolución de una población de organismos de manera darwiniana y lamarckiana.

IV.1.4 Analogía entre la región factible y los patógenos.
En el capítulo II se ha visto que el SIM tiene por misión defender al organismo de los patógenos (exógenos y endógenos), para combatirlos debe tener la capacidad de distinguirlos de las partes del organismo. Esto lo logra procurando eliminar a aquellos linfocitos cuyos receptores se enlazan con partes del propio organismo (selección negativa de los linfocitos B en la médula ósea y selección negativa de los linfocitos T en el timo) o eliminando a aquellos linfocitos cuyos receptores no se unen a un catálogo de epitopes que han sido clasificados como pertenecientes a patógenos (selección negativa de los linfocitos B en los centros germinales –ver sección III.2) o bien permitiendo que se "editen" (modifiquen) los receptores de los linfocitos que se han enlazado con algún epitope propio para comprobar si esta modificación impide que se enlacen otra vez con algún epitope propio (edición de receptores de los linfocitos B en la médula ósea).

Nótese que el SIM busca dejar trabajar a 1) los linfocitos que ha identificado que no se unen con un epitope propio y 2) los linfocitos que se unen con algunos epitopes de un catálogo de epitopes pertenecientes a patogenos[3].

En el contexto de la optimización de funciones restringidas (OFR) lo que se busca es que el método de búsqueda encuentre el óptimo en la región factible. Una forma en la que por lo general se puede lograr esto es buscar puntos en dicha región.

Por lo anterior, se ve que lo natural es hacer una analogía entre los epitopes de los patógenos[4] y la región factible.

[3] Cuando el SIM comete errores epitopes propios pueden quedar en ese catálogo y es posible que se desencadene alguna enfermedad autoinmune.

IV.1.5 Analogía entre la región no factible y los epitopes propios.
Si la región factible *F* (ver 1.5 en el capítulo I) es análoga a los epitopes de los patógenos, por consecuencia la región no factible $S-(F\cap S)$ será análoga a los epitopes propios.

IV.1.6 Como busca el SIN los anticuerpos con mayor afinidad a los epitopes de los patógenos.
Del bosquejo que se ha visto del SIN en el capítulo III se puede observar que la búsqueda de anticuerpos con mayor afinidad a los epitopes de los patógenos (búsqueda del óptimo en la región factible en la OFR) se logra eliminado a los linfocitos que 1) son *suficientemente* afines a epitopes propios o 2) no son *suficientemente* afines a epitopes de patógenos. Se ha resaltado la palabra *suficientemente* porque se requiere pasar de un umbral de afinidad (ver capítulo III) para que se dispare la respuesta inmune. En algunos casos se aplica la alternativa de mandar modificar al linfocito para ver si sus nuevos receptores no son afines a la región factible. Por último se ha mencionado que existen algunos linfocitos afines a epitopes propios que se escapan a la selección y quedan libres para circular por el organismo, y (no se sabe a ciencia cierta por qué) no provocan enfermedades autoinmunes (es decir no desencadenan la respuesta inmune contra las partes del organismo que contienen los epitopes a los que los receptores de los esos linfocitos son afines).

Resumiendo:
1) El sistema inmune clasifica los linfocitos en base a la afinidad de sus receptores con los epitopes.
2) Los linfocitos suficientemente afines a epitopes propios son eliminados.
3) Los linfocitos que no son suficientemente afines a epitopes de patógenos también son eliminados.
4) Hay fugas en (3), algunos linfocitos afines a epitopes propios se escapan de la eliminación.

El manejo de restricciones que hace el EBGA se basa en los puntos (1), (3) y (4).

IV.1.7 La afinidad de un anticuerpo por los epitopes propios. La transgresión.
Para describir cómo hace el EBGA la búsqueda del óptimo en región factible (el problema equivalente para el SIM es la búsqueda de anticuerpos con mayor afinidad a los epitopes de los patógenos), tarea a la que se le llama también *manejo de restricciones*, es necesario describir la analogía entre la afinidad de un anticuerpo por los epitopes propios y la transgresión de un punto en el espacio de búsqueda *S*.

Ya se ha dicho arriba que un anticuerpo representa un punto en *S*. Si dicho punto se encuentra en la región no factible $S-(F\cap S)$, significa que es afín a un epitope propio. A este tipo de anticuerpos se les llama *autoanticuerpos*. Ya se ha mencionado líneas arriba (ver IV.1.6) que se requiere que la afinidad por un epitope (de patógeno o propio) sea *suficiente* (más adelante se tratará de aclarar qué tanto) para que se dispare la respuesta inmune. Se puede establecer la

[4] A veces se usa el término *antígen*o como sinónimo de *patógeno* o *epitope de patógeno*, sin embargo hay que tener cuidado al usarlo ya que el término antígeno se define como "Una molécula que se enlaza a un anticuerpo o receptor de célula T (TCR)."... [AL01] p. 262, y , como se ha visto a lo largo de este trabajo, también moléculas del propio organismo (epitopes propios) pueden enlazarse a los receptores de los linfocitos. Por eso el término *antígeno* debe usarse como sinónimo de *patógeno* cuando no haya duda de que los receptores de los linfocitos y los anticuerpos sólo se unirán a moléculas de patógenos.

analogía entre la fuerza de la afinidad de un autoanticuepo por un epitope y la distancia de un punto no factible (un punto en $S-(F\cap S)$) a F. La transgresión es una medida de tal distancia.

Una forma de definir la transgresión se puede obtener expresando las restricciones de igualdad (I.4) de la siguiente manera:

$$|h_i(\boldsymbol{x})|-\delta_i=0, \text{ para } i=p+1,\ldots,m \quad \text{(IV.1)}$$

donde $\delta_i\in R^+$ y $\delta_i<<|h_i|$ es una tolerancia, o sea que (IV.1) son aproximaciones de las restricciones de igualdad. Definiendo

$$g_i(\boldsymbol{x})=|h_i(\boldsymbol{x})|-\delta_i \text{ para } i=p+1,\ldots,m \qquad \text{(IV.2)}$$

todas las restricciones son entonces restricciones de desigualdad, tomando en cuenta (I.3) y (IV.2) la transgresión, *transg*, se define aquí como:

$$transg=\sum_{i=1}^{m}\tau_i \quad \text{donde} \begin{cases}\tau_i=0 & \text{si } g_i(\boldsymbol{x})\le 0\\ \tau_i=g_i(\boldsymbol{x}) & \text{de otra manera}\end{cases} \qquad \text{(IV.3).}$$

IV.1.8 La aptitud de las bibliotecas y de los anticuerpos.

El EBGA no asigna un valor de aptitud a los anticuerpos[5]. Para identificar si un anticuerpo es más apto que otro hace una comparación entre los dos En el contexto del SIN esta comparación se haría en base a dos medidas: 1) afinidad del anticuerpo por los epitopes de los patógenos y 2) afinidad del anticuerpo por los epitopes propios. En el contexto de la OFR tales medidas corresponden a 1) el valor de la función objetivo (ver ec. I.1 en el capítulo I) evaluada en el punto que representa el anticuerpo, y 2) el valor de la *transgresión* de ese punto, respectivamente. A continuación se explica cómo se hace dicha comparación:

Dados dos anticuerpos a_1 y a_2 que representan dos puntos $\boldsymbol{x}_1$ y $\boldsymbol{x}_2\in S$ respectivamente, a_1 tiene mayor aptitud que a_2 si:

1) $transg(\boldsymbol{x}_1)=0$ y $transg(\boldsymbol{x}_2)>0$

y tiene mayor o igual aptitud si:

2) $(transg(\boldsymbol{x}_1)>0$ y $transg(\boldsymbol{x}_2)>0)$ y $transg(\boldsymbol{x}_1)\le transg(\boldsymbol{x}_2)$

3) $(transg(\boldsymbol{x}_1)=0$ y $transg(\boldsymbol{x}_2)=0)$ y $f(\boldsymbol{x}_1)\succeq f(\boldsymbol{x}_2)$ (ver ec. I.7 en el capítulo I)

El EBGA asigna a cada biblioteca los valores de *transg* y de *f* del anticuerpo más apto de los de una muestra de los anticuerpos que la biblioteca produce (recuérdese que todos los anticuerpos producidos por una biblioteca son obtenidos escogiendo variaciones de partes al azar (ver IV.1.1), así que la muestra es forzosamente aleatoria). Para saber cuál de dos bibliotecas es más apta se les aplica la prueba de las mismas tres condiciones mostradas arriba para dos anticuerpos. El tamaño de la muestra de anticuerpos de la biblioteca lo fija el usuario.

[5] Esto no significa que dados dos anticuerpos no pueda distinguir cuál es el más apto de los dos (o si los dos son igualmente aptos). Lo que significa es que no da un valor *absoluto* de aptitud a un anticuerpo dado.

Obsérvese que los valores de *transg* y f que se le asignan a una biblioteca pueden variar mucho de muestra a muestra de anticuerpos, ya que no se asigna a la biblioteca un valor promedio de estas cantidades, sino que se toman los valores del mejor de los anticuerpos de la muestra para asignárselos. Es por eso que debe existir un registro del mejor anticuerpo obtenido en todo momento, ya que es probable que sea muy difícil que se vuelva a obtener, sobre todo si el número de anticuerpos que puede producir una biblioteca es muy grande.

IV.1.9 El algoritmo genético.

El algoritmo genético usado es una variación del EGA (de sus siglas en inglés *eclectic genetic algorithm*). El EGA es un híbrido compuesto por un GA y un RMHC (acrónimo de *Random Mutation Hill Climber*, escalador por mutación aleatoria) A continuación se describen las características de este GA aplicado a la evolución de bibliotecas:

1) El genoma de cada individuo de la población que hace evolucionar está constituido por dos cadenas binarias, una de estas cadenas contiene codificados los valores de: a) una bandera que indica si el individuo fue obtenido por el algoritmo genético (GA) o por la escalada por mutación aleatoria (RMHC), b) el número de copias de sí mismo que pondrá el individuo en la población intermedia, c) la probabilidad de cruzamiento, d) la probabilidad de mutación, e) el número de veces que será ejecutado el RMHC cuando sea invocado, expresado como una fracción del tamaño de la población de padres.

El objetivo de incluir esta cadena en el genoma es el someterla a los operadores genéticos lo cual provoca que los parámetros que contiene también se vayan optimizando. Hay que recalcar que es necesario el suministrar los intervalos dentro de los cuales podrán tomar valores los parámetros. Para fijar los límites de estos intervalos es recomendable hacer unas cuantas corridas previas, modificando los límites de estos intervalos, para elegir los valores que mejores resultados den. Esta es una afinación tosca. La afinación más detallada se lleva a cabo automáticamente durante la corrida (por medio de la evolución de la cadena binaria que contiene codificados a estos parámetros, como ya se ha mencionado).

La otra cadena contiene codificado al individuo que representa una posible solución del problema, en este caso es una biblioteca, que como ya se ha indicado líneas arriba (ver IV.1.2), es un productor de anticuerpos que a su vez son puntos en *S*.

2) El número de individuos en la población intermedia es por lo general distinto al de la población de padres.
Cuando una generación la obtiene el GA el tamaño de esta población es igual o mayor al número de individuos de la población de padres, esto es el resultado de usar el parámetro (b) descrito en el punto (1), ya que el intervalo en el que puede tomar valores ese parámetro tiene como límite inferior 1. La copias del mejor individuo van primero, las del segundo mejor después y así hasta colocar las copias del peor.
Cuando una generación la obtiene el RMHC, el tamaño de la población intermedia es menor o igual al tamaño de la población de padres, ya que el número de individuos que produce el RMHC es cuando mucho igual al de la población de padres.

3) La formación de parejas en la población intermedia se lleva a cabo deterministamente de la siguiente manera: el primer individuo se cruza con el último de esa población, el segundo con el penúltimo y así sucesivamente. Todos los cruces se llevan a cabo con una probabilidad p_c. A esta manera de formar las parejas se le llama de *Vasconcelos* ([Kur99]pp. 215-219). El tipo de cruzamiento es el anular.
La probabilidad p_c se calcula sacando el promedio aritmético de los valores de probabilidad codificados en el genoma de cada uno de los individuos de la población de padres (es el parámetro mencionado en el inciso (c) del punto 1).

4) La mutación es uniforme, es decir, cada uno de los componentes de la cadena (bits) tiene la misma probabilidad p_m de ser cambiado por su complemento.
De igual manera que la probabilidad p_c, la probabilidad p_m se calcula sacando el promedio aritmético de los valores de probabilidad codificados en el genoma de cada uno de los individuos de la población de padres (es el parámetro mencionado en el inciso (d) del punto 1).

5) La selección de las bibliotecas que formarán la siguiente generación se lleva a cabo tomando las mejores bibliotecas de entre la población de padres y la población de hijos. Esto se logra ordenando la población de la mejor a la peor biblioteca y tomando de ese arreglo ordenado a las primeras mejores en una cantidad igual al tamaño de la población de padres.

Tal ordenamiento se hace comparando de dos en dos bibliotecas y aplicando las condiciones mostradas en IV.1.8.

Obsérvese que de esta forma de proceder es similar a la del SIM: aplica pena de muerte a los linfocitos con receptores que presentan gran afinidad a los propios epitopes y favorece a la existencia de aquellos linfocitos cuyos epitopes presentan gran afinidad por los epitopes de los patógenos.

Si la forma de obtener los parámetros que sirven para ordenar a los individuos de una población es determinista, esta forma de selección constituye el llamado *elitismo total*. En el caso de las bibliotecas no lo es, ya que la forma de evaluarlas no es determinista (ver último párrafo de IV.1.8). Por tanto no es seguro que todas las mejores bibliotecas obtenidas hasta el momento estén presentes en la última población.

6) Invocación automática de un RMHC. Ya se ha visto que el EGA consiste de un GA y un RMHC. El GA invoca al RMHC con una probabilidad dentro del intervalo $[\eta_\lambda, \eta_\mu]$, los límites de este intervalo son fijados por el usuario. Una vez que es invocado el RMHC, el algoritmo hace una búsqueda local (líneas abajo se muestra el RMHC), durante esta búsqueda va creando individuos (bibliotecas) en la población intermedia. El número de individuos creados es igual al número de iteraciones que se ejecuta el RMHC. Este número de iteraciones es una fracción del número de individuos en la población de padres. El valor de esta fracción es el promedio aritmético de los valores de las fracciones que contiene codificado en su genoma cada uno de los individuos de la población de padres (es la fracción mencionada en el inciso (e) del punto 1 de esta subsección). Estos individuos tienen encendida la bandera mencionada en el inciso (a) del punto 1 de esta subsección. La siguiente vez que se calcula la probabilidad de invocar al RMHC, se toma en cuenta el número de individuos creados por el RMHC y que subsistieron a la selección, a mayor número de éstos, mayor probabilidad de invocar nuevamente al RMHC, pero siempre dentro del intervalo $[\eta_\lambda, \eta_\mu]$.

IV.1.10 El escalador por mutación aleatoria (RMHC).
De la población de bibliotecas padre se escoge al azar alguna de las mejores cinco. El GA manda esa biblioteca al RMHC. El RMHC funciona de la siguiente manera:

1 Almacena la biblioteca inicial en *MejorBiblioteca*;
2 Repetir *N* veces
3 Copiar la cadena de *MejorBiblioteca* a *NuevaBiblioteca*;
4 Mutar un bit al azar de *NuevaBiblioteca* y almacenar *NuevaBiblioteca* en la población intermedia del GA;
5 Si *f* de *NuevaBiblioteca* es mejor que *f* de *MejorBiblioteca* copiar la cadena de *NuevaBiblioteca* a *MejorBiblioteca* ;

Tabla IV.1 Analogías entre el SIM y el EBGA.

	Contexto del SIN	**Contexto del EBGA**
1	Epitopes de patógenos	Región factible en *S*: $F \cap S$
2	Epitopes propios	Región no factible en *S*: $S-(F \cap S)$
3	Los genes creadores de receptores de linfocitos y anticuerpos son ensamblados a partir de bibliotecas con diversas partes para esos genes contenidas en el genoma del organismo.	Los anticuerpos que representan puntos en *S* son obtenidos de bibliotecas que contienen diferentes variaciones de sus partes.
4	Las bibliotecas evolucionan con la población de organismos; los anticuerpos evolucionan dentro del cuerpo de un organismo (maduración de la afinidad)	Las bibliotecas evolucionan por poblaciones; los anticuerpos no evolucionan (a diferencia de lo que ocurre en el SIN).
5	Grado de afinidad entre anticuerpos y epitopes de patógenos.	Valores de la función en regiones factibles.
6	Grado de afinidad entre anticuerpos y epitopes propios	Valores de la transgresion, *transg* (ver ec.III. 3)
7	Entre más afines son los receptores de un linfocito al epitope de un patógeno más probabilidades tiene de subsitir. Entre más afines son los receptores de un linfocito a epitopes propios más probabilidades tiene de ser eliminado.	Entre mejor sea el valor de la función objetivo evaluada en un punto representado por un anticuerpo, más probabilidades tiene la biblioteca que creó al anticuerpo de formar parte de la siguiente generación de padres. Entre mayor sea el valor de la transgresión de un autoanticuerpo mayor probabilidad tiene la biblioteca que lo creó de no aparecer en la próxima generación de padres.
8	Algunos linfocitos con receptores afines a un epitope propio logran escaparse de la selección negativa.	El RMHC además de ser un buscador local, a veces se convierte en una fuente generadora de autoanticuerpos.

Se ha mencionado en el punto 4 de IV.1.6 que el SIN presenta normalmente pequeñas “fugas” en la eliminación de linfocitos con receptores afines a epitopes propios, lo que provoca la existencia de autoanticuerpos circulando por organismos sanos. De manera similar, en la línea

4 de este algoritmo se va escogiendo la mejor biblioteca sólo en base a su valor de *f* (ver párrafo antepenúltimo de IV.1.8) sin tomar en cuenta su transgresión, *transg*. Esto puede llevar la búsqueda fuera de regiones factibles. Las razones para hacer esto son 1) muchas veces hay que pasar por la región no factible para encontrar partes de la región factible donde hay mejores puntos y 2) se agrega más diversidad a la población, si un individuo fuera de región factible logra ser seleccionado y puede intercambiar su información con otro por medio del cruzamiento.

IV.1.11 Resumen de las Analogías entre el SIN y el EBGA. En la tabla IV.1 se muestran las analogías entre el SIN y el EBGA.

IV.1.12 El Algoritmo
A continuación se describe los pasos a seguir para aplicar el EBGA:

inicio
Obtener una población aleatoria de bibliotecas B^0 ;
Calcular *f* y *transg* de cada biblioteca en B^0 (ver párrafo antepenúltimo de IV.1.8) ;
Ordenar la población de bibliotecas B^0 ;
$i = 0$;
mientras(i < número máximo de generaciones)
{
Escoger aleatoriamente el método de optimización en esta generación
si(método de optimización = RMHC)
{
Escoger al azar una de las cinco mejores bibliotecas;
Invocar al RMHC mandándole como semilla la biblioteca escogida;
}
si-no // el método escogido fue el GA
{
Calcular $p_c = \frac{1}{\mu}\sum_{i=1}^{\mu}(p_c)_i$ y $p_m = \frac{1}{\mu}\sum_{i=1}^{\mu}(p_m)_i$ // μ es el tamaño de B^i
// $(p_c)_i$ y (p_m) son los valores de las probabilidades de
// cruzamiento y mutación codificados en los genomas de cada una
//de las bibliotecas
Obtener la población intermedia $B^{1/2}$ colocando en ella el número de copias de cada biblioteca codificado en el genoma de cada una de ellas;
Cruzar las parejas de $B^{1/2}$ usando la formación de parejas tipo Vasconcelos y el cruzamiento anular con una probabilidad p_c (ver punto 3 de IV.1.9);
Mutar cada uno de los bits de las cadenas de la población $B^{1/2}$ con una probabilidad p_m ;
Calcular *transg* y *f* de cada biblioteca de $B^{1/2}$;
}
Unir en una sola las poblaciones B^i y $B^{1/2}$ y ordenar dicha población (con base en los criterios mostrados en IV.1.8);
Formar la población B^{i+1} con los primeros μ elementos de la población que se acaba

de ordenar;
$i = i + 1$;
}
fin.

IV.1.13 EGA-DR, un caso especial del EBGA.

Un caso especial del EBGA se presenta cuando el número máximo de anticuerpos que puede producir una biblioteca es uno. Esto ocurre cuando el número de variaciones para cada una de las partes es uno, sin importar en cuantas partes estén divididas las bibliotecas.

En la analogía número 4 de la tabla IV.1 se dice que las bibliotecas evolucionan por poblaciones, con esto se está diciendo que es el intercambio de material genético por cruzamiento entre los miembros de una población, aunado a las esporádicas mutaciones que sufre ese material, el causante de la evolución de las bibliotecas. También en la analogía número 4 de la tabla IV.1, se menciona que en el EBGA sólo evolucionan las bibliotecas, no los anticuerpos. Por supuesto que al evolucionar las bibliotecas, los anticuerpos por ellas *producidos* van siendo también distintos (mejores en general). Lo que se quiere decir en esa analogía es que son las bibliotecas los operandos *directos* de lo operadores genéticos (cruzamiento, mutación y selección) no los anticuerpos.

Cuando el número máximo de anticuerpos producidos por una biblioteca es uno, ocurre que la biblioteca es ese mismo anticuerpo. Cuando una población está constituida por este tipo de bibliotecas, lo que ocurre es que los operadores genéticos actúan *directamente* sobre los anticuerpos. El EBGA deja de representar la evolución de bibliotecas y pasa a representar la evolución de anticuerpos, que hacen las veces de bibliotecas "implícitas" de las que se pide información cada vez que se hace un cruzamiento para formar nuevos anticuerpos. A continuación se ilustra este punto.

Para fines ilustrativos, supóngase que se tiene una biblioteca de cuatro partes y tres variaciones por parte (ver IV.1.1), miembro de una población de bibliotecas:

$$b = v_{1,1}v_{1,2}v_{1,3} \quad v_{2,1}v_{2,2}v_{2,3} \quad v_{3,1}v_{3,2}v_{3,3} \quad v_{4,1}v_{4,2}v_{4,3}$$

tómense dos de los 3^4 anticuerpos que la biblioteca puede producir:

$$a_1 = v_{1,2}v_{2,3}v_{3,1}v_{4,2}$$

$$a_2 = v_{1,3}v_{2,1}v_{3,3}v_{4,1}$$

Ahora supóngase que se tienen dos miembros de una población de anticuerpos. Sus genomas tienen la misma longitud que los de los anticuerpos mostrados sobre estas líneas y, para fines de comparación, han sido divididos en cuatro partes. Han sido etiquetadas con los mismos nombres aquellas partes que contienen exactamente la misma información que las partes de los anticuerpos de arriba:

$$a_{\mathrm{I}} = v_{1,2}v_{2,1}v_{3,3}v_{4,2}$$

$$a_{\mathrm{II}} = v_{1,3}v_{2,3}v_{3,1}v_{4,1}$$

En la figura IV.1 se observa que por cruzamiento anular, por ejemplo, se podría obtener los mismos anticuerpos que se obtuvieron a partir de la biblioteca.

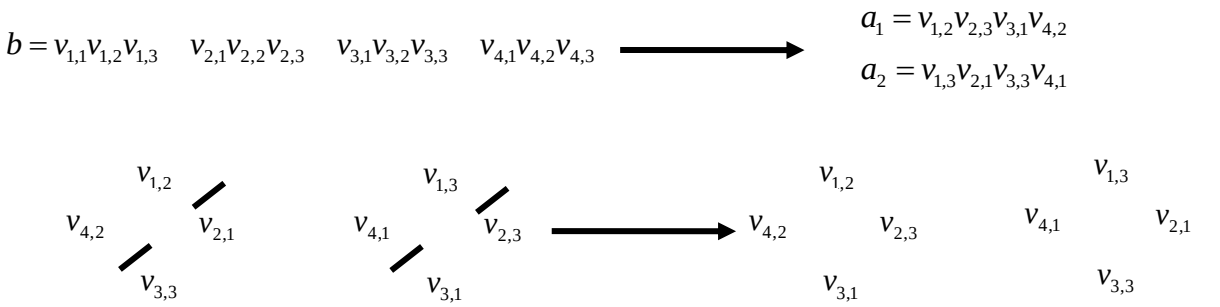


Figura IV.1. Los anticuerpos resultantes del cruzamiento anular de los dos anticuerpos "padres" (parte inferior) podrían producir los mismos anticuerpos producidos por una biblioteca (parte superior). Esto sugiere que en una población de anticuerpos la misma población podría ser interpretada como una biblioteca cuya información es usada a la hora del cruzamiento.

Por supuesto, la evolución de bibliotecas propiamente dichas (bibliotecas con más de una variación por parte) ofrece distintas posibilidades a la evolución de anticuerpos. Por ejemplo, podría darse la posibilidad de que parte del material genético que en un momento dado pareciera no ser tan bueno, subsistiera dentro de la biblioteca y se manifestara en un momento propicio. Cuando se usa la evolución de anticuerpos es más probable que dicho material se pierda y obligue al proceso evolutivo a crearlo otra vez cuando se necesite posteriormente.

Por presentar características distintas a los otros casos del EBGA, al caso en el que una biblioteca corresponde a un anticuerpo ha sido llamado de manera especial: EGA-DR por las siglas en inglés de algoritmo genético ecléctico con ordenamiento determinista (*eclectic genetic algorithm with deterministic ranking*). El término algoritmo genético ecléctico (EGA) es porque, como ya se mencionó (ver IV.1.9), el algoritmo genético usado es una variación del EGA y los criterios para ordenar los anticuerpos (ver IV.1.8) constituyen un ordenamiento determinista. Según esos criterios, primero van los puntos en la región factible y después los que no están en región factible. De los puntos fuera de región factible primero van los que están más cerca de ella; de los puntos en región factible primero va el menor de todos si se está minimizando, o el mayor de todos si se está maximizando.

IV.1.13.1. El papel del RMHC en el EGA-DR.

Ya se ha visto en el capítulo III que la maduración de la afinidad de los receptores de los linfocitos B es uno de los mecanismos que ayudan a la creación de receptores cada vez más afines a los epitopes de los patógenos. La maduración de la afinidad se lleva a cabo mutando aproximadamente sólo la mínima parte mutable de la parte del genoma que expresa la región variable de los receptores cada vez que se multiplica un linfocito B en los centros germinales. Eliminando posteriormente aquellas mutaciones que no se enlazan con la fuerza suficiente a un epitope de patógenos, y alentando a subsistir a aquellas mutaciones que sí se enlazan con la suficiente fuerza a un epitope de patógenos. Puede observarse que la similitud de la maduración de la afinidad con el RMHC es muy grande: prácticamente es el mismo algoritmo. Se puede decir entonces que cuando la biblioteca representa a un anticuerpo, el RMHC hace las veces de la maduración de la afinidad.

IV.1.13.2 Una modificación al RMHC usado en el EGA-DR.

Aquí se propone una modificación al RMHC usado con el EGA-DR. En lugar de usar siempre la mutación puntual (mutación de un solo bit) se puede dejar al azar el número de bits que puedan ser mutados a la vez. Aquí se propone que el número máximo de bits que se puedan mutar a la vez sea una fracción de la longitud del genoma del anticuerpo y que su valor mínimo sea 1. La ventaja que puede traer el mutar más de un bit a la vez es que si se da el caso

de que el RMHC quede atrapado en un óptimo local, se incremente la probabilidad de que salga de él. En la ecuación IV.4 se define el número máximo de bits, $n_{b,max}$, que pueden ser mutados a la vez en RMHC usando en el EGA-DR.

$$n_{b,max} = \lfloor 1 + f_{nb} \times l \rfloor \qquad \text{(IV.4)}$$

donde

l es la longitud en bits de la longitud de la cadena binaria que representa a tanto a los parámetros automodificables como de la subcadena que representa un punto en la región factible.

f_{nb} es una fracción, $f_{nb} \in [0, 1]$, es un parámetro que debe ser fijado por el usuario.

De la población de anticuerpos padre se escoge al azar alguno de los mejores cinco. El GA manda esa biblioteca al RMHC. El RMHC funciona de la siguiente manera:

1 $n_{b,max} = \lfloor 1 + f_{nb} \times l \rfloor$;
2 Almacenar *AnticuerpoInicia*l en *MejorAnticuerpo*;
3 Repetir *N* veces
4 Copiar la cadena de *MejorAnticuerpo* a *NuevoAnticuerpo*;
5 Obtener un número entero aleatorio *r* con distribución uniforme en $\{1, \ldots, n_{b,max}\}$;
6 Mutar *r* bits al azar de *NuevoAnticuerpo* y almacenar *NuevoAnticuerpo* en la población intermedia del GA. El bit que indica la procedencia de la cadena {RMHC, GA} no debe ser mutado;
7 Si *f* de *NuevoAnticuerpo* es mejor que *f* de *MejorAnticuerpo* copiar la cadena de *NuevoAnticuerpo* a *MejorAnticuerpo*;

IV.1.13.3. Resumen de las analogías entre el SIN y el EGA-DR.

En la tabla IV.2 se muestran las analogías entre el SIN y el EGA-DR.

Tabla IV.2 Analogías entre el SIM y el EGA-DR.

	Contexto del SIN	**Contexto del EGA-DR**
1	Epitopes de patógenos	Región factible en *S*: $F \cap S$
2	Epitopes propios	Región no factible en *S*: $S - (F \cap S)$
3	Los anticuerpos evolucionan dentro del cuerpo de un organismo por maduración de la afinidad.	Los anticuerpos evolucionan cruzándose entre sí y por maduración de la afinidad.
4	Grado de afinidad entre anticuerpos y epitopes de patógenos.	Valores de la función en regiones factibles.
5	Grado de afinidad entre anticuerpos y epitopes propios	Valores de la transgresion, *transg* (ver ec.III. 3)
6	Entre más afines son los receptores de un linfocito al epitope de un patógeno más probabilidades tiene de subsitir. Entre más afines son los receptores de un linfocito a epitopes propios más probabilidades tiene de ser eliminado.	Entre mejor sea el valor de la función objetivo evaluada en un punto representado por un anticuerpo, más probabilidades tiene éste de formar parte de la siguiente generación de padres. Entre mayor sea el valor de la transgresión de un autoanticuerpo mayor probabilidad tiene de no aparecer en la próxima generación de padres.
7	Algunos linfocitos con receptores afines a un epitope propio logran escaparse de la selección negativa.	El RMHC además de ser un buscador local, a veces se convierte en una fuente generadora de autoanticuerpos.

IV.2. El algoritmo evolutivo basado en la evolución de anticuerpos y autoanticuerpos (EAA).

El algoritmo está inspirado en la maduración de la afinidad (MA) con edición de receptores (ER) que sufren algunos linfocitos en los centros germinales en el baso y en los nodos linfáticos después de que han sido creados en la médula ósea. Como ya se ha visto (ver capítulo III), la MA consiste en la *mutación puntual* (ver siguiente punto) de la parte del genoma del linfocito B que expresa la región variable de los receptores de los linfocitos y de los anticuerpos, aproximadamente cada vez que se divide (i.e., que se reproduce asexualmente) un linfocito B. La afinidad de los receptores se ve incrementada de generación en generación de linfocitos B. Éstos se reproducen asexualmente. El responsable de los pequeños cambios que incrementan la afinidad es el mecanismo llamado *hipermutación somática* (ver capítulo III). George y Gray [GG99] sugieren que también existe un mecanismo que causa que los receptores de los linfocitos B sufran grandes cambios, este mecanismo es la *edición de receptores*. En la figura IV.2 se esquematizan ambos mecanismos en una gráfica de afinidad contra los epitopes de patógenos. La figura IV.2 es sólo un bosquejo, en realidad son estructuras tridimensionales de las moléculas las que deben embonar unas con otras, si además se toman en cuenta otras propiedades tales como la electronegatividad de las regiones de diferentes partes de las moléculas, la dimensionalidad requerida para representar la afinidad del enlace receptor-epitope es de entre cinco y ocho ([GG99], [SFP97]).

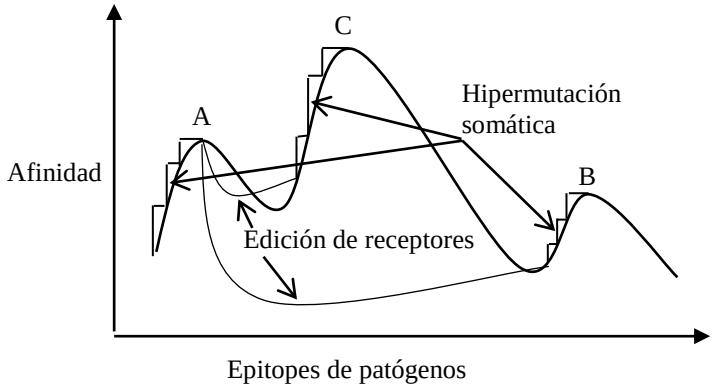


Figura IV.2 Durante la maduración de la afinidad la afinidad entre los receptores de los linfocitos B y las regiones determinantes antigénicas de los patógenos se incrementa por medio de dos mecanismos, la hipermutación somática (HS) y la edición de receptores (ER). La HS sube a lo largo de los montes mientras que la ER da grandes saltos en la gráfica de la afinidad. Los saltos de la ER pueden causar que la búsqueda termine en la punta de un monte de baja altura (*e.g.*, un salto desde A provoca que se alcance B), pero también pueden hacer que la búsqueda termine en una cima más alta (un salto desde A también podría provoca que se alcance C). (adaptada de la figura 1 in George, A.J.T., Gray, D.: Receptor editing during affinity maturation. Immunology Today. (1999) Vol. 20, No. 4, 196)

IV.2.1 La mutación puntual.

Al mutar una sola de las mínimas unidades de almacenamiento de la información se le llama *mutación puntual* (PM, por sus siglas en inglés *point mutation*). En el caso de la representación con cadenas binarias se refiere a la mutación de un solo bit.

IV.2.2 Poblaciones de anticuerpos.

El EAA se basa en la evolución de poblaciones de linfocitos B *artificiales*. Son artificiales porque no sólo se multiplican por división, asexualmente, sino que también por cruzamiento entre sí, sexualmente. Esto último no ocurre en los sistemas inmunes naturales.

Estos linfocitos evolucionan independientemente uno de otro por medio de maduración de la afinidad, hasta que se determina que deben intercambiar información entre ellos por medio del cruzamiento.

En la figura IV.3 se esquematiza el paso de la información para identificar antígenos a través de diversas estructuras del SIM. La información para identificar los epitopes de un antígeno, que es la que se encuentra en la región variable de los receptores de los linfocitos y en la región variable de los anticuerpos, es la información de interés para este trabajo. Es por eso que, para simplificar, a los miembros de las poblaciones de los linfocitos artificiales que usa el EAA, los llamaremos de aquí en adelante *anticuerpos*, como se ha hecho en el caso el EGA-DR (IV.1.13).

Los anticuerpos son cadenas binarias y representan un punto dentro del espacio de búsqueda *S*, como en el EBGA. Su aptitud se mide como se describe en IV.1.8.

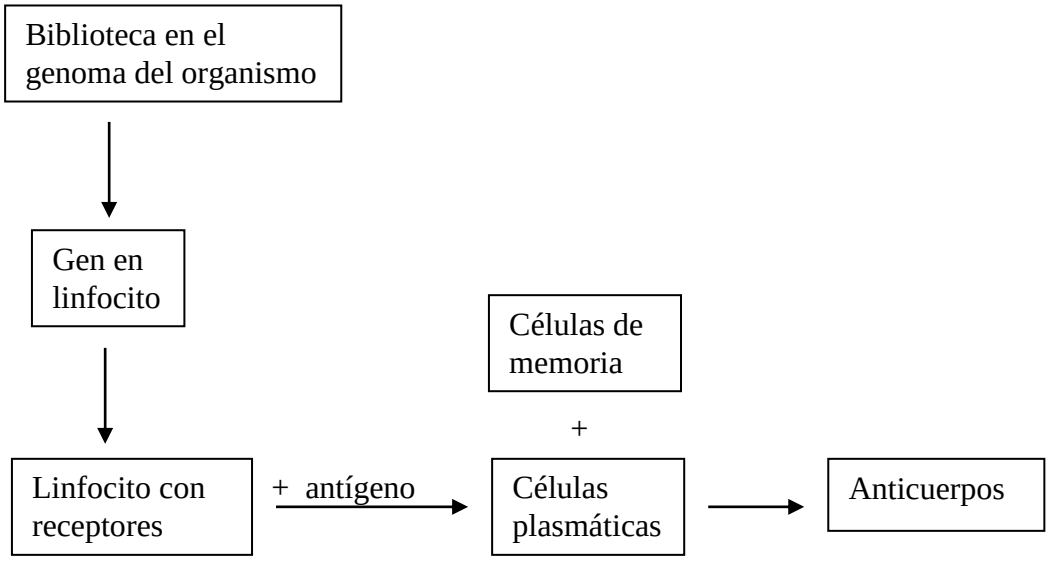


Figura IV.3 En esta figura se esquematiza cómo pasa la información entre diferentes estructuras del SIN.

IV.2.3 La maduración de la afinidad con edición de receptores en el EAA.

El EAA usa una población de cadenas binarias que representan receptores de los linfocitos B o, como ya se ha comentado arriba, anticuerpos. Para representar la maduración de la afinidad se puede definir un parámetro que describa la proclividad de cada receptor a sufrir edición de receptores (ER). En esta versión del algoritmo, uno de los receptores de la población tiene una proclividad nula y otro tiene la proclividad más grande a sufrir ER. Llámese T_i a la proclividad del *i-ésimo* individuo a sufrir ER. Supóngase que la población está ordenada de alguna manera y que es de tamaño ω, entonces $T_1 = 0$ y $T_\omega = T_{max}$. Para el i-ésimo receptor:

$$T_i = \frac{T_{max}}{(\omega - 1)}(i-1) \quad \text{para } i = 1,\ldots,\omega$$

T_{max} puede estar en el intervalo [0, 1], pero parece ser razonable el darle valores en [0, 0.5] porque si se fijara un valor cercano a 1 resultaría que el último individuo llevaría a cabo una búsqueda sin dirección.

El algoritmo se basa en un procedimiento que se inspira en la maduración de la afinidad con edición de receptores, este procedimiento es llamado aquí MAER.

El que el descendiente no cumpla con los criterios mostrados en IV.1.8 al compararlo con su progenitor puede significar tres cosas: 1) que el descendiente no esté en región factible y el progenitor sí, 2) que ambos no estén en región factible y que el descendiente tenga una transgresión mayor que el progenitor, o 3) que ambos estén en región factible y que el descendiente tenga un valor de *f* peor que el de su progenitor.

Entonces el que el número aleatorio r sea menor o igual que el umbral T_i, puede significar que un miembro de la población se salga de la región factible, se aleje de ella o siga en ella pero pase a un punto peor al que estaba el padre. Todo esto lleva inmediatamente a lugares peores del espacio de búsqueda al anticuerpo. Pero también le permite alejarse de lo que pueden ser mínimos locales y le dan la posibilidad de iniciar la escalada de mayores "colinas". En el contexto del SIM significa que en algunos casos se permite la formación de autoanticuerpos. De ahí el nombre de *evolución de anticuerpos y autoanticuerpos*.

IV.2.4 Mutación puntual y recombinación ¿son necesaria ambas?

Tanto [GG00] como [Coo00] coinciden en que es necesario usar la mutación puntual y la recombinación para desarrollar linfocitos cuyos receptores tengan gran afinidad por los epitopes de los antígenos (ver sección III.2).

Para obtener una idea sobre si son necesarios los dos operadores analícese un caso sencillo en el contexto de optimización de funciones donde la representación de los puntos en *S* son cadenas binarias de longitud *l*. Utilícese mutación puntual y el cruzamiento anular para tratar de encontrar el óptimo. Supóngase que se conoce la cadena que representa el punto donde se encuentra el óptimo global, cadena óptima. Supóngase que se tiene dos cadenas cuya distancia de Hamming al óptimo es de $l-1$, esto significa que cada una de ellas tiene únicamente un bit en común con la cadena óptima. Si se aplica una mutación puntual a cualquiera de las cadenas, la probabilidad de obtener una cadena mejor, esto es, la probabilidad de obtener una cadena con dos bits en común con la cadena óptima es:

$$p_{\text{mej. cad. por m.p.}} = \frac{l-1}{l} \qquad \text{(IV.5)}$$

Para grandes valores de *l* se tiene que :

$$p_{\text{mej. cad. por m. p.}} \approx 1 \qquad \text{(IV.6)}$$

Si se aplica el cruzamiento anular, se requiere que los arcos que se deben intercambiar entre las dos cadenas sean tales que el arco de una de las cadenas contenga al único bit bueno de esa cadena, y el arco de la otra cadena no contenga al único bit bueno de esa cadena. El resultado será que se obtendrá una cadena que no tendrá ni un solo bit en común con la cadena óptima, pero también se tendrá otra cadena que tendrá dos bits en común con ella. Para calcular la probabilidad promedio de que esto pase plantéese el siguiente razonamiento: el cruzamiento anular consiste en doblar las dos cadenas de manera tal que formen un anillo, quedando su primer y ultimo bits uno junto al otro. Para recombinar a ambas cadenas deben obtenerse dos números aleatorios enteros $r_1 \in \{1, 2, \ldots, l\}$ y $r_2 \in \{1, 2, \ldots, l-1\}$ ambos de distribución uniforme. Los arcos de ambas cadenas iniciando en el bit número r_1 y con una longitud de r_2 bits, deben ser intercambiados entre las dos cadenas. Para calcular el número de veces que un

arco con un bit bueno será intercambiado por un arco que no contiene ni un bit bueno, se puede visulizar las posiciones de los dos bits buenos en la misma cadena. Asígnesele el número 1 a la posición de uno de los bits buenos, el número 2 al siguiente bit en sentido de las manecillas del reloj y así sucesivamente hasta llegar a la posición número *l*. Llámese $i \in \{2, 3, \ldots, l\}$ a la posición del otro bit bueno. El número de arcos de todas las posibles longitudes, n_{arcos}, que contienen al primer bit bueno pero no al segundo cuando éste último está en la posición i son:

$$n_{arcos} = (i-1)(l-i+1) \qquad \text{(IV.7)}$$

así que la probabilidad de intercambiar los arcos deseados cuando el segundo bit bueno está en la posición deseada es:

$$p_{\text{cruz. anul.}} = \frac{(i-1)(l-i+1)}{l(l-1)} \qquad \text{(IV.8)}$$

El promedio de las probabilidades de obtener los arcos deseados cuando del segundo bit está en cualquiera de las posibles posiciones es:

$$\overline{p}_{\text{cruz. anul.}} = \frac{\sum_{i=2}^{l}(i-1)(l-i+1)}{l(l-1)} = \frac{l+1}{6l} \qquad \text{(IV.9)}$$

para magnitudes grandes de *l*:

$$\overline{p}_{\text{cruz. anul.}} \approx \frac{1}{6} \qquad \text{(IV.10)}$$

Compárese los valores mostrados en (IV.6) y (IV.10). Evidentemente en este ejemplo es mucho más probable el obtener una mejor cadena por mutación puntual que por cruzamiento anular. O sea que es mejor usar la mutación puntual que el cruzamiento anular.

Analícese ahora un segundo ejemplo. Supóngase que se tienen dos cadenas binarias distintas entre sí con una distancia de Hamming a la cadena óptima de 1, esto significa que cada una tiene sólo un bit malo, es decir, un bit cuyo valor es distinto al correspondiente de la cadena óptima. Si se aplica la mutación puntual a cualquiera de ellas, la probabilidad de cambiar al bit malo por el bueno es de:

$$p_{\text{mej. cad. m. p.}} = \frac{1}{l} \qquad \text{(IV.11)}$$

que para valores grandes de l es una cantidad muy pequeña. Si se usa el cruzamiento anular aplicando el razonamiento de los bits buenos ahora a los bits malos obtendremos que la probabilidad promedio de obtener una mejor cadena es:

$$\overline{p}_{\text{cruz. anul.}} \approx \frac{1}{6} \quad \text{(IV.12)}$$

Compárese (IV.11) con (IV.12), es evidente que para longitudes grandes (donde *longitudes grandes* significa longitudes mayores de 6 bits), es mejor el cruzamiento anular.
Se puede concluir que tanto la mutación puntual como el cruzamiento, son necesarios para encontrar el óptimo global. Si la función no tiene óptimos locales y las cadenas son largas, el

cruzamiento ayuda cuando las cadenas están cerca del óptimo global. Cuando la función tiene óptimos locales, el cruzamiento puede ayudar a liberar el proceso cada vez que las cadenas queden atrapadas en óptimos locales (se entiende que queden atrapadas en óptimos locales distintos entre sí).

IV.2.5 Intercalando las operaciones de cruzamiento y mutación puntual.

Al principio, cuando es probable que las cadenas estén lejos de la cadena óptima, la mutación puntual debe usarse hasta que se encuentre dicha cadena o hasta que las cadenas se queden atoradas en óptimos locales. Si éste último es el caso, se debe aplicar una operación de cruzamiento. Así que un ciclo de varias operaciones de mutación puntual y cruzamiento, parecen ser una buena estrategia para la optimización de funciones.

IV.2.6 Analogías entre el SIN y la EAA.

En la tabla IV.3 se muestran las analogías entre el SIN y el EAA.

Tabla IV.3. Analogías entre el SIM y el EAA.

	Contexto del SIM	**Contexto del EAA**
1	Epitopes de patógenos	Región factible en *S*: $F \cap S$
2	Epitopes propios	Región no factible en *S*: $S-(F \cap S)$
3	Los anticuerpos evolucionan dentro del cuerpo de un organismo por maduración de la afinidad.	Los anticuerpos evolucionan por casi escalamiento por mutación aleatoria y por cruzamiento entre sí, por esta última causa se requiere usar poblaciones de anticuerpos.
4	Grado de afinidad entre anticuerpos y epitopes de patógenos.	Valores de la función en regiones factibles.
5	Grado de afinidad entre anticuerpos y epitopes propios	Valores de la transgresion, *transg* (ver ec.IV. 3)
6	Entre más afines son los receptores de un linfocito al epitope de un patógeno más probabilidades tiene de subsitir. Entre más afines son los receptores de un linfocito a epitopes propios más probabilidades tiene de ser eliminado.	Entre mejor sea el valor de la función objetivo evaluada en un punto representado por un anticuerpo, más probabilidades tiene de formar parte de la siguiente generación de padres. Entre mayor sea el valor de la transgresión de un autoanticuerpo mayor probabilidad tiene de no aparecer en la próxima generación de padres.
7	Algunos linfocitos con receptores afines a un epitope propio logran escaparse de la selección negativa.	Los anticuerpos con umbrales mayores que cero en el MAER tienen la probabilidad de evolucionar en forma de autoanticuerpos. En el GA, si en la población constituida por la población de padres y la población intermedia hay por lo menos un anticuerpo que represente un punto en región factible, esta población se ordena tomando únicamente en cuenta el valor de la función objetivo evaluada en el punto representado por cada anticuerpo. La siguiente población se obtiene tomando los primeros individuos de esta población ordenada. Esto puede provocar también la creación de autoanticuerpos.

IV.2.7 El algoritmo.

La rutina principal del EAA es un algoritmo genético (GA) . Este algoritmo tiene las siguientes características:

- No manda evaluar directamente ni la función objetivo ni las restricciones, en lugar de esto invoca otro algoritmo que determina el número de iteraciones que correrá el MAER y de él recibe los valores de la función objetivo y de las restricciones.
- Usa cadenas binarias para representar los puntos en *S* y un conjunto de tres parámetros que son autoadaptables. Estos parámetros son: 1) el número de de descendientes por cada anticuerpo, n_o, 2) la probabilidad de mutación p_m, y 3) la probabilidad de cruzamiento, p_c. La probabilidad de mutación se usa para aplicar el operador de mutación únicamente a la parte de las cadenas que representan los tres parámetros autoadaptables que se acaban de mencionar.
- El tamaño de la población intermedia es variable y depende del valor del parámetro n_o de cada anticuerpo.
- Ordena la población constituida por la población de padres y la población intermedia, $A^i + A^{1/2}$, en una de las dos siguientes maneras 1) cuando no hay ni un punto factible en esta población, el punto con una menor transgresión va primero, y el que tiene la mayor transgresión va al último; 2) cuando hay por lo menos un anticuerpo factible en esta población, el anticuerpo que tiene mejor valor de la función objetivo (el que tiene el mayor valor si se maximiza y menor si se minimiza) va primero.
- Como en el caso del EBGA, forma las parejas, que se cruzan con una probabilidad p_c, de manera determinista con un pareo llamado *Vasconcelos*. El tipo de cruzamiento es anular, también como en el EBGA.

El EAA debe recibir los valores de

- El número de evaluaciones de la función objetivo, *NMaxEv*, fijado por el usuario.
- El tamaño de la población de padres, μ, fijado por el usuario.
- El máximo valor del umbral para no ascender en un momento dado (ver IV.2.3), T_{max}, fijado por el usuario.
- Los números mínimo y máximo de descendientes de cada anticuerpo en la población de padres, $n_o \in \{n_{o,min}, n_{o,max}\}$, fijados por el usuario. El valor de $n_{o,min}$ debe ser al menos de 1.
- Los valores mínimos y máximos de la probabilidad de mutación para aplicar al operador de mutación uniforme a los bits que codifican los parámetros autoadaptables, $p_m \in [p_{m,min}, p_{m,max}]$, fijados por el usuario.
- Los valores mínimos y máximos de la probabilidad de cruzamiento, $p_c \in [p_{c,min}, p_{c,max}]$, fijados por el usuario.

El EAA regresa

- El mejor anticuerpo encontrado durante la búsqueda.

El algoritmo es:

inicio

1 Generar una población inicial aleatoria de anticuerpos, A^0, de tamaño μ. Son cadenas binarias de longitud $l = l_{pa} + l_v$; l_{pa} es la longitud de la subcadena que representa los valores de los tres parámetros autoadaptables: n_o, p_m y p_c ; l_v es la longitud de la subcadena que representa un punto en el espacio de búsqueda *S*.

2 Calcular el valor de la función objetivo y de la transgresión del primer anticuerpo y almacenar este anticuerpo junto con los mencionados valores en el registro *MejorAnticuerpo*. Este registro será modificado únicamente por la rutina del MAER. La función objetivo ha sido evaluada una vez, por lo que el contador del número de evaluaciones debe ser actualizado: $NEv = NEv + 1$.

3 Se invoca la rutina:
Administrador-de-MAER(NEv, A^0)
Cuando se regresa de esta rutina la variable *NEv* y la población en el argumento (en este caso A^0) están actualizadas. La rutina invocada, *Administrador-del-MAER*, monitorea el desempeño de la MAER y decide si la MAER continúa la búsqueda del óptimo o si regresa el control a esta rutina principal,

4 $i = 0$;

5 **mientras**(*NEv* < *NMaxEv*)

6 Calcular $p_m = \sum_{i=1}^{\mu} p_{m,i} \Big/ \mu$; $p_c = \sum_{i=1}^{\mu} p_{c,i} \Big/ \mu$; $p_{m,i}$ y $p_{c,i}$ son los valores de las probabilidades de mutación y cruzamiento que están codificadas en la cadena binaria de cada individuo, en la subcadena de parámetros autoadaptrables.

7 Hacer la población intermedia, $A^{1/2}$, de tamaño λ copiando los individuos de la población de padres. El número de copias que cada anticuerpo pone en la población intermedia está dado por el valor del parámetro n_o representado en la subcadena de parámetros autoadaptables de cada anticuerpo. Cada individuo pone por lo menos una copia en $A^{1/2}$, por lo que $\lambda \geq \mu$. Obsér-vese que n_o puede cambiar durante la corrida para cualquier individuo, así que λ también puede cambiar de generación en generación.

8 Aplicar el pareo *Vasconcelos* y el cruzamiento anular con probabilidad p_c, calculada en el paso 6, a los anticuerpos en $A^{1/2}$. El pareo *Vasconcelos* ([Kur99] p. 216), es una manera determinista de formar las parejas de anticuerpos que se cruzarán: la primera cadena se cruza con la última, la segunda con la penúltima, y así consecutivamente. Todos los cruces se hacen con una probabilidad p_c. La razón de hacerlo de esta manera es el tratar de preservar la variedad genética. En el cruzamiento anular, el genoma no se ve como una estructura lineal de bits sino como un anillo. Una cadena se transforma en un anillo conectando su extremo izquierdo con su extremo derecho. Cuando se aplica el cruzamiento anular hay dos parámetros que se deben tomar en cuenta:1) el punto de inicio del cruce, que puede ser cualquiera de los *l* bits y 2) la longitud del arco que puede tomar los valores de 1 a *l*-1. El cruzamiento en un punto destruye los esquemas de mayor longitud definitoria y tiende a preservar los esquemas de menor longitud definitoria, sin destruir muchos esquemas. El cruzamiento uniforme no favorece ni a los esquemas de

longitud definitoria grande ni a los de longitud definitoria pequeña, pero tiende a destruir muchos esquemas. El cruzamiento anular trata de no favorecer ni a los esquemas de longitud definitoria grande ni a los de longitud definitoria pequeña sin destruir muchos esquemas, por lo que adopta un compromiso entre el cruzamiento en un punto y el cruzamiento uniforme [DS91].

9 Aplicar al operador de mutación con la probabilidad p_m calculada en el paso 6, a los l_{pa} bits de los parámetros autoadaptables de cada cadena de la población $A^{1/2}$. Las otras partes de las cadenas, los bits que representan los puntos en *S*, son mutados con la MAER.

10 Se invoca laa rutina:
Administrador-de-MAER(NEv, $A^{1/2}$)

11 Ordenar los individuos de la población $A^i + A^{1/2}$ en una de las dos siguientes maneras: 1) si hay por lo menos un anticuerpo que representa un punto factible en $A^i + A^{1/2}$ los anticuerpos son ordenados de acuerdo a sus valores de la función objetivo, sin importar si son factibles o no; si se minimiza, el anticuerpo con el menor valor de la función objetivo va primero; si se maximiza, el anticuerpo con el mayor valor de la función objetivo va primero; 2) si no hay ni un solo anticuerpo que represente un punto factible en $A^i + A^{1/2}$, entonces el individuo con el menor valor de transgresión es el que va primero.
Ordenar de la manera (1) fomenta la exploración de lo que parecen ser mejores regiones sin importar si son factibles o no. Se espera que esta forma de proceder no acarree muchos problemas, ya que el algoritmo ya ha encontrado por lo menos un punto en región factible: *si lo ha hecho una vez, lo hará otra vez* (eso se espera). Ordenar de la forma (2) es más convencional, no hay ni un punto factible en la población de padres ni en la población intermedia; esto significa dos cosas a) la búsqueda se ha desplazado de la región factible hacia la región no factible, o peor, b) el algoritmo no ha encontrado ni un punto factible en toda la historia de la corrida. Así que el GA concentra todos sus esfuerzos en encontrar al menos un punto factible.

12 La población A^{i+1} se hace con los primeros μ anticuerpos de la población ordenada $A^i + A^{1/2}$.

13 $i = 1+1$;

14 **fin de mientras**

15 Regresa al mejor anticuerpo encontrado durante toda la corrida y almacenado en el registro *MejorAnticuerpo.*

fin.

La rutina *Administrador-de-MAER* es:

Administrador-de-MAER(número *NEv*, población de anticuerpos *A*)
inicio

1 Calcula el número de evaluaciones de la función objetivo asignado a cada anticuer-po en *A* (que es igual al número de mutaciones puntuales que se aplicará a cada anticuerpo en *A*), N_{ea}, como una función de la longitud de la subcadena del anticuerpo que

representa al punto en *S*: $N_{ea} = f_{nea} \cdot l_v$. El parámetro $f_{nea} \in [0,1]$ es dado por el usuario, en este trabajo se le dio un valor de 0.1.

2 Calcular un umbral para cada uno de los miembros de la población, representa la facilidad de saltar de una cadena padre a una peor cadena hija, entre más grande el umbral más grande la probabilidad de hacer este tipo de salto:

$$T_i = \frac{T_{max}}{N\text{-}1}(i-1) \text{ para } i = 1,\ldots,N$$

donde

$0 \le T_{max} \le 1$ es el umbral máximo, es un dato dado por el usuario.

N es el tamaño de la población *A*, si *A* es la población de padres su valor es μ, si es una población intermedia su valor es λ.

Nótese que el primer individuo tiene un umbral de 0 y el último individuo tiene el máximo umbral.

3 Calcular el número de evaluaciones que quedan por realizarse:
$NEv_{restantes} = NMaxEv - NEv$;

4 **repetir**

5 Revisar si el número de evaluaciones restantes es suficiente para evaluar N_{ea} veces a cada miembro de la población, si no es suficiente reduce N_{ea} :

si($\frac{NEv_{restantes}}{N_{ea}} < N$) **entonces** $N_{ea} = \left\lceil \frac{NEv_{restantes}}{N} \right\rceil$

Con esto se ajusta (se incrementa un poco) el número máximo de evaluaciones fijado por el usuario si es que el número de evaluaciones restantes no alcanza para aplicar N_{ea} veces la mutación puntual a cada anticuerpo.

6 *NSaltosBuenos* = 0; inicia el contador de buenos saltos, este contador es actualizado por la rutina del MAER

7 $MAER(N_{ea}, N)$;

8 $NEv = NEv + N \cdot N_{ea}$;

9 **mientras**($NEv < NMaxEv$ y $NSaltosBuenos/(N \cdot N_{ea}) \ge T_{\text{saltos buenos}}$); $T_{\text{saltos buenos}} \in [0,1]$ es un umbral dado por el usuario.

fin.

Observaciones. Los parámetros f_{nea} y $T_{\text{saltos buenos}}$ son muy importantes para la evaluación del desempeño la rutina MAER. La idea es tratar de determinar si una fracción importante de los puntos está atorada en óptimos locales, si este es el caso, la búsqueda por mutación puntual debe detenerse y se debe regresar el control al GA. El parámetro f_{nea} está relacionado con el número de mutaciones puntuales dadas a la rutina MAER, y $T_{\text{saltos buenos}}$ con el número de saltos a puntos iguales o mejores, para decidir si la búsqueda por mutación puntual es lo suficientemente buena.

Nótese que si las condiciones en (9) siempre se cumplen durante la corrida, toda la optimización se lleva a cabo por medio de la MAER.

Características del MAER.
El MAER debe tener acceso a :

- Una población de anticuerpos especificada por la función invocadora: $A = (a_1, \ldots, a_N)$;
- Un registro donde se almacena al mejor punto encontrado por el algoritmo hasta el momento: *MejorAnticuerpo*;
- Un registro de un contador de "buenos saltos" (esto es cuando el descendiente es igual o mejor que el padre), *NSaltosBuenos*;

Y debe recibir los valores de:

- El número de mutaciones puntuales, N_{ea}, que se aplicará a cada miembro de A;
- El tamaño de la población N, que es μ si A es una población de padres, y λ si es una población intermedia.
- Los valores de los umbrales de todos los individuos.

El MAER regresa:

- El registro *MejorAnticuerpo* actualizado;
- La población actualizada *A*, con los mejores descendientes encontrados para cada cadena binaria.

El algoritmo es:

MAER(número N_{ea}, número N)
inicio
1 **para** *i* = 1 **hasta** *N*
2 **para** j = 1 **hasta** N_{ea}
3 Hacer una copia de la cadena, a_i, y escoger al azar uno de los bits de la parte de la cadena que representa un punto en *S* y substituir el valor del bit por su complemento (p.ej. si el valor del bit es 1 se cambia por 0). Llamar a la copia con la mutación puntual *b*;
4 Evaluar la función objetivo y las restricciones en *b*;
5 **si**(*MejorQueOIgualA*(*b*, a_i))
6 *NSaltosBuenos* = *NSaltosBuenos* + 1; aquí se actualiza el contador de buenos saltos
7 Substituir a_i con *b*, almacenar *b* en el registro temporal *Mejor-AnticuerpoTemp* y si(*MejorQueOIgualA*(*b*, *MejorAnticuerpo*)) actualizar el registro *MejorAnticuerpo* almacenando en él el punto *b*, *f*(*b*) y *transg*(*b*)
8 **si no** obtener un número aleatorio con distribución uniforme $r \in [0,1]$, si ($r \leq T_i$) substituir a_i con *b*;
9 **fin de si**
10 **fin de para**
11 Almacenar al mejor descendiente de la *i-ésima* cadena: a_i = *MejorAnticuepoTemp*;
12 **fin de para**
fin

donde la función *MejorQueOIgualA* es:

MejorQueOIgualA(anticuerpo *b*, anticuerpo *a*)
inicio
1 **si**(el anticuerpo *b* está en la región factible y *a* no
2 o
3 tanto el anticuerpo *a* como el *b* no están en región factible y $transg(b) \leq transg(a)$
4 o
5 ambos anticuerpos *a* y *b* están en región factible y $f(b) \succeq f(a)$)
6 **regresa** verdadero;
7 **si no regresa** falso;
fin.

IV.3 Comparación entre los algoritmos aquí propuestos y el de Hajela y Yoo.
En la tabla IV.4 se muestran diferentes aspectos de los algoritmos aquí propuestos y el de Hajela y Yoo, denominado aquí HY (ver [HY96]).

Los cuatro algoritmos usan cadenas binarias para representar los puntos en el espacio de búsqueda.

Según Hajela y Yoo [HY96], su algoritmo utiliza una analogía con el ensamblado de los genes que expresan a los receptores de los linfocitos y a los anticuerpos. El ensamblado de dichos genes se hace por medio de la evolución de una población de anticuerpos con un algoritmo genético. Los algoritmos aquí propuestos utilizan esa misma analogía del ensamblado de esos genes y además utilizan el concepto de maduración de la afinidad y la autoinmunidad. El EBGA hace una analogía más directa con el ensamblado de genes de los receptores y de los anticuerpos, ya que utiliza bibliotecas, a partir de las cuales se puede expresar una gran variedad de ellos (aquí llamados anticuerpos, ver segundo y tercer párrafos de IV.2.2). El uso que de este concepto hacen el EGA-DR y el EAA es muy similar al algoritmo de Hajela y Yoo. El uso de la maduración de la afinidad no lo usa Hajela y Yoo. De los tres algoritmos propuestos es el EBGA el que en menor grado la utiliza, en un mayor grado la usa el EGA-DR y en mayor medida el EAA. Los tres algoritmos permiten un poco de autoinmunidad, es decir permiten y a veces hata favorecen que la búsqueda se desplace hacia regiones no facibles. Esto con la esperanza de que al atravesar una parte de la región no factible se encuentre otra parte mejor de la factible.

Hajela y Yoo [HY96] mencionan que “Una población típica de evolución genética contiene una mezcla de diseños factibles y no factibles” (ellos utilizan su sistema inmune para optimizar el diseño de una armadura, es por eso que a cada solución la llaman “diseño”), y en su algoritmo proponen usar las cadenas que representan a los puntos factibles para que sirvan como muestra para que por medio de un proceso evolutivo se haga que las cadenas que representan a los puntos no factibles evolucionen para representar puntos en y/o más cercanos a dicha región.

Tabla IV.4 Comparación de los métodos aquí propuestos y el de Hajela y Yoo [HY96].

	Descripción general	Separa la búsqueda de la región factible de la búsqueda del óptimo.	Forma de buscar la región factible.	Forma de buscar el óptimo.
HY	Un GA está anidado dentro de otro. El GA interno acerca a una población de anticuerpos a la región factible; el externo busca el óptimo.	Sí.	El GA interno hace que una población de anticuerpos (cadenas fuera de región factible) tiendan a parecerse a una muestra de antígenos (cadenas en región factible)	El GA externo busca el óptimo usando el valor de la función objetivo como valor de aptitud.
EBGA	Cada miembro de la población es una *biblioteca*, que es una cadena binaria de la que se obtienen *anticuerpos*, que son representaciones de las soluciones. Un GA y un RMHC invocado aleatoriamente hacen evolucionar las bibliotecas.	No.	El GA ordena las bibliotecas en base al mejor anticuerpo conocido que producen. Siempre van primero aquéllas cuyo mejor anticuerpo está en región factible.	El GA hace la búsqueda de la biblioteca cuyo mejor anticuerpo conocido tiene afinidad por el óptimo restringido. El RMHC hacen la búsqueda de la biblioteca cuyo mejor anticuerpo tiene afinidad por el óptimo no necesariamente restringido
EGA-DR	Cada miembro de la población es un anticuerpo, que es una representación de una solución. Un GA y un RMHC invocado aleatoriamente hacen evolucionar los anticuerpos	No.	El GA ordena los anticuerpos , primero van todos los que están en región factible.	El GA mencionado en la anterior columna y el RMHC hacen la búsqueda del óptimo. El RMHC busca el óptimo sin tomar en cuenta las restricciones (puede favorecer *autoinmunidad*).
EAA	Cada miembro de la población es un anticuerpo, que es una representación de una solución. Un procedimiento basado en la mutación puntual (denominado aquí MAER) es usado para *madurar la afinidad* (buscar la región factible y el óptimo en ella) de cada anticuerpo. Si hay visos de que el MAER se estanca, se cruzan los anticuerpos usando el pareo *Vasconcelos*.	No.	El MAER hace la búsqueda de la región factible favoreciendo las más de las veces los saltos hacia los puntos en esa región. Cuando ni en la población de padres ni en la intermedia hay una sola cadena factible, el GA hace un ordenamiento poniendo siempre primero a las cadena más cercanas a la región factible	El MAER busca *preferentemente* el óptimo restringido. Si en la población de padres o en la intermedia hay por lo menos un punto en región factible, el GA ordena de manera tal que se favorece la búsqueda del óptimo sin tomar en cuenta las restricciones (puede favorecer *autoinmunidad*)

Es posible hacer esto para problemas en los que el tamaño de la región factible es tal que permite que desde una población inicial obtenida aleatoriamente se obtengan puntos factibles. Por ejemplo, si el número de puntos factibles en el espacio de búsqueda equivale al cinco porciento de todos los puntos en dicho espacio, para las poblaciones iniciales de cien individuos el promedio esperado de puntos factibles será de cinco. Pero si la región factible es demasiado pequeña, por ejemplo, un punto factible por cada diez mil o más puntos del espacio de búsqueda, entonces ya será un problema el encontrar un punto factible, cuanto más encontrar el óptimo restringido.

Los algoritmos aquí propuestos no requieren que al principio se conozcan puntos factibles, pueden encontrarlos y buscar los óptimo restringidos aun para regiones factibles muy pequeñas.

IV.4 Una posible clasificación de los algoritmos propuestos.

Los tres algoritmos se pueden clasificar como algoritmos genéticos porque comparten las siguientes características: usan cadenas de caracteres para representar los puntos en el espacio de búsqueda, cada uno de los componentes del vector de posicionamiento en el espacio de búsqueda toma valores en un intervalo acotado por límites inferior y superior cerrados, usan el operador de cruzamiento. Todas estas características pertenecen a los algoritmos genéticos que podrían ser llamados *tradicionales*. En cuanto a su clasificación en relación con los sistemas inmunes dos de ellos, el EBGA y el EAA están inspirados débilmente en la recombinación somática y la maduración de la afinidad con edición de receptores, el EGA-DR es un caso extremo del EBGA, no está inspirado directamente en los sistemas inmunes. Tanto el EBGA como el EAA siguen de lejos el modelo de la selección clonal.

Capítulo V
Experimentación

V.1 Programas desarrollados y equipo de cómputo usado.

Los tres algoritmos propuestos en el capítulo IV:

1) Algoritmo genético basado en la Evolución de Bibliotecas Generadoras de Anticuerpos (EBGA).
2) El caso especial del EBGA: Algoritmo Genético Ecléctico con Ordenamiento Determinista (EGA-DR por sus siglas en ingles *eclectic genetic algorithm with deterministic ranking*) .
3) El algoritmo evolutivo basado en la evolución de anticuerpos y autoanticuerpos maduración de la afinidad con edición de receptores (EAA).

fueron programados en lenguaje C++. Se usó el entorno para desarrollo rápido de aplicaciones: *Borland C++ Builder 5*. Los programas fueron desarrollados y ejecutados en computadoras personales con el sistema operativo *Windows 2000*. Se usaron dos computadoras personales: una de ellas con el procesador *pentium IV* a 1.9 GHz, 256 MB de memoria, con un disco duro de 60 GB; la otra con el procesador *pentium IV* a 2.0 GHz, 512 MB de memoria, con un disco duro de 100 GB.

V.2 La codificación.

Se usaron cadenas binarias para representar los puntos en los espacios de búsqueda. La forma de codificar los puntos que se usó es la sugerida por Michalewicz en ([Mic96] p. 33). La ventaja de usar esta codificación es que todas las cadenas corresponden a un punto dentro del espacio de búsqueda, *S*, determinado por los límites de cada una de las variables del problema (ver ec. I.2 en el capítulo I): la cadena 00 . . . 0 corresponde a los límites inferiores de las variables, y la cadena 11 . . . 1 corresponde a sus límites superiores. De esta manera, sólo hay que trabajar para que se cumplan las restricciones de desigualdad e igualdad mostradas en I.3 y I.4 (ver capítulo I). Las relativas a los límites de cada una de las variables se cumplen automáticamente. A continuación se explica a detalle esa forma de codificación.

Sea *real_a_cbg* (número real a cadena binaria en código *gray*) una función para transformar un número real $x_i \in [li_i,\ ls_i]$ a una cadena binaria de longitud $l < \infty$ en código *gray*. Esta es una forma de discretizar al número real, ya que la cadena binaria es finita. Los valores discretos que puede tomar el número real x_i están distribuidos homogéneamente a lo largo del intervalo $[li_i,\ ls_i]$. Se debe *proponer* un valor de la precisión, Δ_p, que es la diferencia entre dos de los números discretos consecutivos en el intervalo. Los argumentos de *real_a_cbg* son el número real, los límites del intervalo y la precisión. Un ejemplo de invocación es: $v_g = real\_a\_cbg(x_i,\ li_i,\ ls_i,\ \Delta_p)$; donde $v_g \in \{0,\ 1\}_g^l$, es un entero binario (cadena binaria) escrito en código *gray*. La función es:

cadena binaria en gray **función** $real\_a\_cbg(x_i,\ li_i,\ ls_i,\ \Delta_p)$

1 **inicio**

2 $l = \left\lceil \log_2 \frac{ls_i - li_i}{\Delta_P} \right\rceil$;

3 $\Delta_r = \frac{ls_i - li_i}{2^l - 1}$;

4 $z = \frac{x_i - lb_i}{\Delta_r}$;

5 $v_w = bin(z,\ l)$;

6 $v_g = gray(v_w)$;

7 **regresar** v_g;

8 **fin**.

donde

Δ_r es la precisión real;

z es un entero positivo escrito en sistema decimal;

v_w es un entero binario (cadena binaria) de longitud *l* escrito en código binario pesado, $v_w \in \{0,\ 1\}_w^l$. Si es necesario, se escriben 0s a la izquierda del bit más significativo hasta que el número quede de longitud *l*;

bin es una función que transforma un entero en sistema decimal a un entero en código binario pesado;

gray es una función que transforma una cadena en código binario pesado a una cadena en código *gray*.

Ejemplo. Para $x_i \in [0, 10]$, $x_i = 2.14$**,** y $\Delta_P = 1\times10^{-4}$, se tiene $l = 17$**,** $\Delta_r = 7.62945\times10^{-5}$, z = 28,049, v_w = 0 0110 1101 1001 0001, v_g = 0 0101 1011 0101 1001.

La función inversa de *real_a_cbg* es *cbg_a_real* (cadena binaria en código gray a un número real). Los argumentos de la unción son la cadena binaria en código *gray*, el límite inferior del intervalo del número real y la precisión real. La función se invoca así $x_i = cbg\_a\_real(v_g,\ li_i,\ \Delta_r)$. La función es:

real función *cbg_a_real*(v_g, li_i, Δ_r)

1 **inicio**

2 $v_w = wb(v_g)$;

3 $z = itg(v_w)$;

4 $x_i = z * \Delta_r + lb_i$;

5 regresar x_i

6 **fin**.

donde

wb es una función para transformar una cadena en código *gray* a una cadena en código binario pesado.

itg es una funión para transformar una cadena en código binario pesado a un entero no negativo en sistema decimal.

Ejemplo**.** Para v_g = 0 1011 0010 1100 1111, li_i=0, y $\Delta_r = 7.62945 \times 10^{-5}$, se tiene v_w = 0 1101 1100 1000 1010, z = 56,458, x_i = 4.3074.

Para representar un punto en el dominio con una cadena binaria, las cadenas binarias que representan a cada una de las coordenadas del punto deben concatenarse una con otra (ver figura V.1).

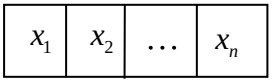


Figura V.1. Las cadenas binarias que representan las coordenadas de un punto en S se concatenan unas con otras para formar una sola cadena binaria.

V.2.1 Bibliotecas y anticuerpos del EBGA.

Las cadenas binarias que representan las bibliotecas del EBGA tienen la forma que se representa en la figura V.2.

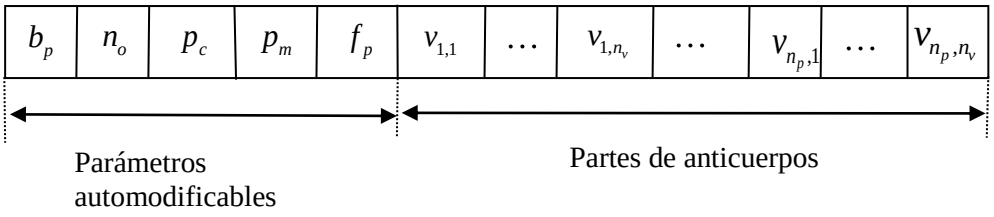


Figura V.2. Cadena binaria que representa una biblioteca del EBGA con los parámetros automodificables.

Los parámetros automodificables son: $b_p = \{\text{"GA", "RMHC"}\}$ es una bandera que indica si la biblioteca fue obtenida por el GA o el RMHC; n_o es el número de copias de la biblioteca que serán puestos en la población intermedia; p_c es un valor de la probabilidad de cruzamiento; p_m es un valor de la probabilidad de mutación; $f_p \in (0,\ 1]$, es una fracción de la población de padres, e indica el número de iteraciones que se aplicará el RMHC cuando sea invocado. Este es igual al número de individuos de la población intermedia que generará esa función.

De la subcadena de partes de anticuerpos se extraen los segmentos para formar los anticuerpos (en realidad para formar los genes de los que se expresan los receptores y los anticuerpos, ver figura IV.2). La biblioteca contiene n_p partes y cada parte tiene n_v variaciones. El número de partes es fijado por el usuario independientemente del número de dimensiones del espacio de búsqueda. El número de variaciones por parte es también fijado por el usuario. La longitudes de las variaciones pertenecientes a cada parte son obtenidas con el siguiente algoritmo:

1 **inicio**
2 **para** i = 1 **hasta** n_p
3 **inicio**

4 $l_i = \left\lfloor \frac{l_v}{n_p} \right\rfloor;$

5 **fin de para;**

6 **si**($residuo\left(\frac{l_v}{n_p}\right) > 0$)

7 **inicio**

8 $i = residuo\left(\frac{l_v}{n_p}\right);$

9 **mientras**($i > 0$)

10 **inicio**

11 $l_i = l_i + 1;$

12 $i = i - 1;$

13 **fin de mientras**

14 **fin de si**

15 **fin.**

Donde

l_i es la longitud de las variaciones de la *i-ésima* parte de la biblioteca,

l_v es la longitud de una cadena como la que se muestra en la figura V.1, que representa un punto en *S*.

Un anticuerpo se obtiene tomando al azar una variación de cada una de las partes de la biblioteca, la cadena resultante (el anticuerpo) es la representación de un punto en *S*. En la figura V.3 se muestra el ejemplo de un anticuerpo obtenido de una cadena de cuatro partes con tres variaciones por parte para un espacio de búsqueda de dos dimensiones ($n_p = 4, \quad n_v = 3, \quad n = 2$).

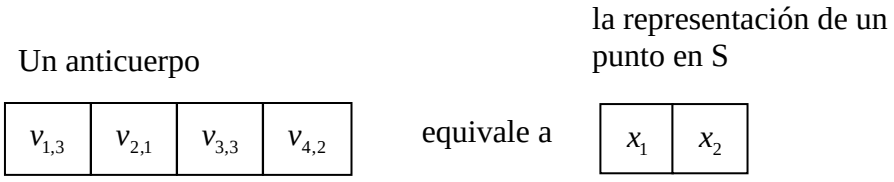


Figura V.3. En este ejemplo se muestra un anticuerpo proveniente de una biblioteca de cuatro partes, con tres variaciones por parte, para producir anticuerpos para un espacio de búsqueda de dos dimensiones. Se puede intentar el resolver el mismo problema con bibliotecas con distintos números de partes y distintos números de variaciones por parte.

V.2.2 Los anticuerpos del EGA-DR.

Se ha dicho en el capítulo IV que el EGA-DR es un caso especial del EBGA en el que el número de variaciones por parte es uno. En este caso, sin importar cuantas partes tenga la biblioteca, se puede producir un solo anticuerpo a partir de ella. Esto significa que la

biblioteca es ese mismo anticuerpo. Por tanto la cadena de un anticuerpo es como se muestra en la figura V.4. Los parámetros automodificables son los mismos que en el EBGA.

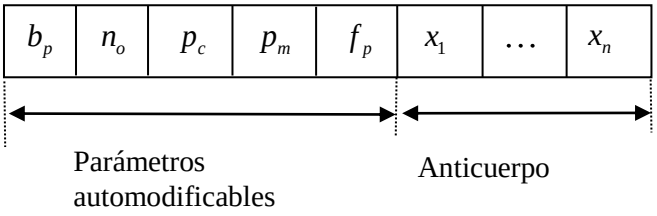


Figura V.4. Cadena binaria que representa el genoma usado por el EGA-DR.

V.2.3 Los anticuerpos del EAA.

El genoma del EAA está compuesto por una subcadena de parámetros automodificables y por una subcadena que representa un punto en el espacio de búsqueda *S*, esta última subcadena es el anticuerpo propiamente dicho. Son tres los parámetros automodificables: n_o, p_c y p_m, que tienen el mismo significado descrito en V.2.1. Hay que recalcar que los valores de la probabilidad de mutación que son codificados en los individuos de la población son promediados aritméticamente, y la probabilidad promedio así obtenida se aplica únicamente a los bits de la subcadena de los parámetros automodificables. Al contrario de lo que se hace el EBGA y en su caso particular el EGA-DR, en los que la probabilidad promedio se obtiene de la misma manera y se aplica a todos los bits de la cadena, con la única excepción del primer bit, el que sirve de bandera para indicar de dónde proviene la biblioteca, en el caso de EBGA, o el anticuerpo, en el caso del EGA-DR. En la figura V.5 se representa el genoma del EAA.

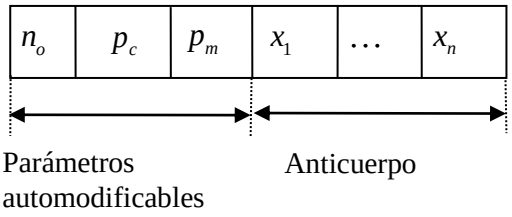


Figura V.5. Genoma del EAA, tiene tres parámetros automodificables y una subcadena que representa un punto en el espacio de búsqueda.

V.3 Prueba de algoritmos.

Una vez que se diseña un algoritmo de optimización global es necesario probar que tan bien funciona. Una forma de hacerlo es usar funciones de prueba para buscar sus óptimos con el algoritmo. Se puede escoger funciones diseñadas específicamente para que sirvan de prueba o tomar funciones que surgen de aplicaciones reales. Otra opción es diseñar algoritmos que generen funciones de prueba. A continuación se comenta sobre las colecciones de funciones y algoritmos generadores de problemas de prueba.

V.3.1 Uso de colecciones de funciones.

Jones comenta en [Jon00] que desde los inicios del campo de la optimización global se ha reconocido la necesidad de tener problemas estándares de prueba para facilitar las

comparaciones entre algoritmos. Comenta que en 1975 Dixon y Szegö desarrollaron una pequeña colección de funciones compuesta por problemas de baja dimensionalidad, sin restricciones y muy fáciles de resolver, propusieron a los participantes en un congreso llamado *Towards Global Optimization* que hicieran uso de ella. Por los defectos descritos, tal colección no constituía un medio ideal para probar algoritmos. Continúa Jones [Jon00] comentando que en 1981 Hock y Schittkowski publicaron un colección incluyente de problemas, pero éstos eran más bien de optimización local y no de optimización global.

Es pertinente el hacer el siguiente comentario a lo dicho por Jones hasta aquí: Dixon y Szegö publicaron dos trabajos tres años después (1978) [DS78a][DS78b] en los que muestran más funciones (también sin restricciones). Las funciones de Dixon y Szegö han sido utilizadas por varios autores hasta la fecha para probar sus algoritmos de optimización global no restringida (por ejemplo: [AS97], [HA97], [HN99], [OGJ01], [TK02]).

Sigue Jones [Jon00] comentando que en 1990 Floudas y Pardalos publicaron el libro *A Collection of Test Problems for Constrained Global Optimization* [FP90], cubriendo con él parte del vacío que en esta materia había. Yo he tratado de usar algunos de los problemas que trae ese libro. Hay algunos muy interesantes que corresponden a aplicaciones reales, los hay con regiones factibles convexas y no convexas, uni y multimodales, con baja y alta dimensionalidad. Sin embargo, he encontrado que algunos de los problemas sobre aplicaciones reales no muestran algunas de las restricciones necesarias para resolverlos, lo que obliga a replantearlos para descubrir esas restricciones faltantes (por ejemplo, problemas 4.8 y 4.9 en [FP90] pp. 32-35). También me he encontrado con otros que tienen errores, presentan como solución un punto que no cumple con algunas de las restricciones del problema planteado (ver problema 4.4.3 en [FP90] p. 29).

El libro *Handbook of Test Problems in Local and Global Optimization* [FP99] incluye muchos de los problemas del libro *A Collection of Test…*[FP90] revisados.

V.3.2 Uso de programas generadores de funciones de prueba.

Otra aproximación al problema de probar algoritmos de optimización global es el uso de programas que generan funciones de prueba. Las ventajas que presenta el uso de estos programas es que, por lo general, se puede fijar de antemano la posición del óptimo global, el número de mínimos o máximos locales, en algunos casos el tamaño de las cuencas de atracción hacia el óptimo [GL98], el tamaño de las regiones factibles, y otras características del problema. Algunas de las desventajas son que por lo general no presentan suficiente variedad, a veces las restricciones no se pueden expresar de manera explícita.

En [GL98] se presenta un algoritmo para producir problemas de prueba para la optimización no lineal sin restricciones; en [LZ00] se presenta una forma de producir problemas de optimización no lineal restringida usando redes neuronales de perceptrones; en [MDSS00] se presenta un método para obtener funciones para optimización no lineal restringida fijando seis parámetros que determinan la complejidad de la función generada.

V.3.3 Las pruebas de los algoritmos propuestos.

Para probar los algoritmos propuestos en este trabajo se usó la colección de funciones numéricas restringidas usadas por Runarsson y Yao [RY00], quienes a su vez las tomaron de [KM99] y [Mic95]. Las trece funciones de esta colección se presentan en el apéndice A.

En la tabla V.1 se muestran algunas características de estas trece funciones.

Tabla V.1 Algunas características de los problemas de prueba para los algoritmos propuestos.

Problema	Tipo de función objetivo	No. de variables	No. total de restricciones	No. de restricciones lineales	No. de restricciones de igualdad	$\frac{F \cap S}{S}$
g01	no lineal	13	9	9	0	0.000002
g02	no lineal	20	2	1	0	0.999973
g03	no lineal	10	1	0	1	0.000002
g04	no lineal	5	6	0	0	0.269744
g05	no lineal	4	5	2	3	$< 10^{-6}$
g06	no lineal	2	2	0	0	0.000063
g07	no lineal	10	8	3	0	0.000002
g08	no lineal	2	2	0	0	0.008677
g09	no lineal	7	4	0	0	0.005261
g10	lineal	8	6	4	0	0.000003
g11	no lineal	2	1	0	1	0.000102
g12	no lineal	3	1	0	1	0.047771
g13	no lineal	5	3	0	3	$< 10^{-6}$

También se usó el tipo de prueba que se propone en [Mic95], [KM99] y [RY00], consistente en asignar un valor a cada uno de los parámetros que deben fijarse del algoritmo, a continuación son realizadas varias corridas independientes del método con esos valores fijados. Este procedimiento debe repetirse para cada función. Una vez hechas las corridas se calcula la media, la mediana, la desviación estándar, se aparta el mejor y el peor valores obtenidos.

Las ecuaciones V.1 sirven para medir de manera cuantitativa el desempeño de los métodos.

$$\xi_{mejor} = \begin{cases} \dfrac{min(|f(\boldsymbol{x}^*)|, |f(\boldsymbol{x}_{mejor})|)}{max(|f(\boldsymbol{x}^*)|, |f(\boldsymbol{x}_{mejor})|)} & \text{si } f(\boldsymbol{x}^*) \succeq f(\boldsymbol{x}_{mejor}) \text{ y } max(|f(x^*)|, |f(x_{mejor})|) \neq 0 \\ 1 & \text{de otra forma} \end{cases} \tag{V.1a}$$

$$\xi_{mediana} = \begin{cases} \dfrac{min(|f(\boldsymbol{x}^*)|, |f(\boldsymbol{x}_{mediana})|)}{max(|f(\boldsymbol{x}^*)|, |f(\boldsymbol{x}_{mediana})|)} & \text{si } f(\boldsymbol{x}^*) \succeq f(\boldsymbol{x}_{mediana}) \text{ y } max(|f(x^*)|, |f(x_{mediana})|) \neq 0 \\ 1 & \text{de otra forma} \end{cases} \tag{V.1b}$$

$$\xi_{media} = \begin{cases} \dfrac{min(|f(\boldsymbol{x}^*)|, |f(\boldsymbol{x}_{media})|)}{max(|f(\boldsymbol{x}^*)|, |f(\boldsymbol{x}_{media})|)} & \text{si } f(\boldsymbol{x}^*) \succeq f(\boldsymbol{x}_{media}) \text{ y } \max(|f(x^*)|, |f(x_{media})|) \neq 0 \\ 1 & \text{de otra forma} \end{cases} \tag{V.1c}$$

$$\xi_{peor} = \begin{cases} \dfrac{min(|f(\boldsymbol{x}^*)|, |f(\boldsymbol{x}_{peor})|)}{max(|f(\boldsymbol{x}^*)|, |f(\boldsymbol{x}_{peor})|)} & \text{si } f(\boldsymbol{x}^*) \succeq f(\boldsymbol{x}_{peor}) \text{ y } \max(|f(x^*)|, |f(x_{peor})|) \neq 0 \\ 1 & \text{de otra forma} \end{cases} \quad \text{(V.1d)}$$

donde

$f(\boldsymbol{x}^*)$ es el mejor valor conocido de la función;

$f(\boldsymbol{x}_{mejor})$ es el mejor valor de $f(\boldsymbol{x})$ de las $f(\boldsymbol{x})$ resultantes obtenidas por el método que se está probando de las corridas independientes;

$f(\boldsymbol{x}_{mediana})$ es la mediana de las $f(\boldsymbol{x})$ resultantes obtenidas por el método en las corridas independientes;

$f(\boldsymbol{x}_{media})$ es el promedio aritmético de las $f(\boldsymbol{x})$ resultantes obtenidas por el método en las corridas independientes;

$f(\boldsymbol{x}_{peor})$ es el peor valor de $f(\boldsymbol{x})$ de los resultados obtenidos por el método que se está probando en las corridas independientes;

Obsérvese que $0 \leq \xi_i \leq 1$ para $i \in \{mejor, mediana, media, peor\}$. Entre más similar sea a 1 mejor será el resultado y entre más se acerque a 0 será peor.

V.3.3.1 El problema de asignar valores a los parámetros.

El asignar valores a los parámetros de los que depende un método es en sí mismo un problema de optimización: *cuál es el mejor conjunto de parámetros que se debe fijar para aplicar un método de optimización*. Aquí se ha optado por no buscar tal conjunto. Se han hecho algunas corridas sólo para obtener una idea de cuales podrían ser algunos valores de los parámetros que podrían ser convenientes para llevar a cabo la búsqueda del óptimo. La hipótesis que en esto subyace es que es posible alcanzar el óptimo evaluando la función objetivo y sus restricciones un número *razonable* de veces partiendo de una gran cantidad de distintos conjuntos de valores asignables a los parámetros.

V.3.3.2 Los experimentos realizados.

Para cada uno de los tres métodos se hace una lista de los parámetros que deben fijarse. Para cada método se clasifican los parámetros en dos tipos: unos a los que se les puede llamar *constantes*. A éstos se les asigna un solo valor que no cambiará en ninguna de las corridas. Y otros a los que se les puede llamar *variables*, cuyos valores puede cambiar de un experimento a otro y son tomados de un intervalo designado para tal efecto para cada uno de ellos.

Para cada uno de los métodos se han hecho cuatro experimentos. Para cada uno de ellos parámetros variables han sido escogidos al azar.

Con esta forma de proceder se pretende demostrar que los resultados no dependen muy fuertemente de los valores iniciales de los parámetros variables tomados dentro de los intervalos asignados a cada uno de ellos.

Para poder comparar los tres métodos entre sí, se ha procurado que el número de evaluaciones de la función objetivo y sus restricciones sea el mismo para cada uno de los cuatro experimentos de cada método; el valor fijado es de 500,000 evaluaciones aproximadamente.

V.3.3.2.1 En qué consiste cada experimento.

Una vez que se fijan (que se les asigna un valor a) los parámetros constantes y variables (ver V.3.3.2), se realizan 30 corridas independientes para cada uno de los trece problemas (la función objetivo y sus restricciones) con el método que se desea probar. Para cada una de estas corridas se reporta:

1) el mejor valor de la función objetivo obtenido, $f(\boldsymbol{x})$,
2) el número de evaluaciones de la función objetivo y sus restricciones, necesarias para obtenerlo,
3) indicación de si el punto correspondiente, $\boldsymbol{x}$, está o no en región factible,
4) número de generación en la que se obtuvo el mejor individuo (para el EBGA y el EGA-DR) o número de veces que se invocó el operador de cruzamiento para obtener el mejor individuo (para el EAA).
5) las coordenadas de cada punto, $\boldsymbol{x} = (x_1, x_2, ..., x_n)$.

De las treinta corridas de un problema *se toman solamente en cuenta aquéllas cuyo mejor punto encontrado cayó en región factible*, para obtener la siguiente información:

1) mejor valor de la función objetivo, $f(\boldsymbol{x}_{mejor})$,
2) la mediana de los valores de la función objetivo, $f(\boldsymbol{x}_{mediana})$,
3) la media (promedio aritmético) de los valores de la función objetivo de cada punto, $f(\boldsymbol{x}_{media})$,
4) la desviación estándar de los valores de la función objetivo,
5) el peor valor de la función objetivo, $f(\boldsymbol{x}_{peor})$,
6) la media del número de iteraciones para encontrar al mejor punto de cada corrida
7) la media del número de generaciones para encontrar al mejor punto de una corrida (para el EBGA y el EGA-DR) o el número de veces que se usó el operador de cruzamiento.
8) el porcentaje de puntos factibles.

Con la información obtenida y los mejores valores de la función objetivo mostrados en el apéndice A, se evalúan los parámetros ξ_i, $i \in$ [*mejor*, *mediana*, *media*, *peor*] (ver sección V.3.3) para cada una de las funciones.

V.3.3.2.2 Parámetros del EBGA.

Los parámetros que se deben fijar para usar el EBGA son:

n_p número de partes en las que se divide un anticuerpo (representación en forma de una cadena binaria de un punto en el espacio de búsqueda de la función). Ver IV.1.1;
n_v número de variaciones de cada una de las partes en las que se divide el anticuerpo;
n_m tamaño de la muestra de anticuerpos;
μ tamaño de la población de padres;
$\{n_{o,min}, n_{o,max}\}$ el número de hijos de cada biblioteca tendrá un valor en este intervalo;
$[p_{m,min}, p_{m,max}]$ intervalo donde tomará valores la probabilidad de mutación;
$[p_{c,min}, p_{c,max}]$ intervalo donde tomará valores la probabilidad de cruzamiento;
$[f_{\eta,min}, f_{\eta,max}]$ intervalo donde tomará valores la fracción de población de pádres, f_η. El número de iteraciones que se ejecutará el RMHC cuando sea invocado es $f_\eta \times \mu$.
$[\eta_\lambda, \eta_\mu]$ intervalo donde tomará valores la probabilidad de que sea invocado el RMHC;
n_g número de generaciones que se ejecutará el EBGA.

En la tabla V.2. se muestran los parámetros constantes y en la V.3 los variables (ver V.3.3).

Tabla V.2 Parámetros constantes y los valores que se les han asignado

Variable	Valor asignado
$n_{o,min}$	1
$n_{o,max}$	2
$p_{m,min}$	0.0001
$p_{c,max}$	1
$f_{\eta,max}$	1
n_{λ}	1/8 = 0.125
n_{μ}	1/2 = 0.5

A otras de las variables se les ha asignado un valor dentro de intervalos fijados de manera más o menos arbitraria. Algunas cuantas corridas han sido realizadas para obtener una idea, aunque sea burda, de los valores que podrían tomar estas variables. Los intervalos fijados a cada variable son mostrados en la tabla V.3.

Tabla V.3 Parámetros variables y sus intervalos correspondientes.

Variable	Intervalo en el que toma su valor
n_p	{2, 6}
n_v	{2, 3}
n_m	{1, 5}
μ	{4, 100}
$p_{m,max}$	[0.01, 0.1]
$p_{c,min}$	[0.5, 1]
$f_{\eta,min}$	[0.5, 1]

Suena razonable que el tamaño de la muestra deba ser mayor entre mayor sea el número de anticuerpos que una biblioteca pueda producir (ver IV.1.1). En este caso las bibliotecas que tienen 2 partes y 2 variaciones por parte son las que pueden producir menos anticuerpos distintos: $2^2 = 4$; y las que se dividen en 6 partes y tienen 3 variaciones por parte son las que producen el mayor número de anticuerpos distintos: $3^6 = 729$. De manera arbitraria se ha propuesto la ecuación V.2 para fijar el tamaño de la muestra. Según la ecuación, la biblioteca que puede producir más anticuerpos es evaluada con una muestra de 5 anticuer-pos y la que produce menos con una muestra de 1.

$$n_m = \left\lfloor \frac{5\left(n_v^{n_p} - 1\right)}{724} \right\rfloor + 1 \qquad \text{(V.1)}$$

Recuérdese que el EBGA siempre almacena al mejor anticuerpo (representación del mejor punto) encontrado hasta cualquier momento de la ejecución del algoritmo, es por ello que el hacer que el método use más evaluaciones de la función objetivo y de sus restricciones nunca va en detrimento del resultado final y sí puede ayudar a mejorarlo. Al EBGA no se le alimenta el número de evaluaciones de la función objetivo que se desea que realice, sino un número de

generaciones. Con el objeto de comparar los tres algoritmos se fija un número de evaluaciones (n_{evals}) el número de generaciones del EBGA aproximadamente equivalente se calcula con la ecuación V.3a.

$$n_g = \left\lfloor \frac{n_{evals} - \mu}{n_m \cdot \left(n_{o,max} \cdot \mu \cdot \left(1 - \bar{\eta} \cdot \overline{f_\eta}\right) + \mu \cdot \bar{\eta} \cdot \overline{f_\eta} \right)} + 0.5 \right\rfloor \qquad \text{(V.3a)}$$

donde

$$\overline{f_\eta} = \frac{f_{\eta,min} + f_{\eta,max}}{2} \qquad \text{(V.3b)}$$

$$\bar{\eta} = \frac{\eta_\lambda + \eta_\mu}{2} \qquad \text{(V.3c)}$$

En las primeras seis columnas de la tabla V.4 se muestran los valores obtenidos al azar dentro de los intervalos mostrados en la tabla V.3 para los primeros cuatro experimentos del EBGA. Los valores de la columna 7 han sido obtenidos con la ecuación V.1a, fijando el número de evaluaciones , n_{evals}, en 500,000. En la tabla V.5 se muestran los valores fijados para el quinto experimento del EBGA.

Tabla V.4 Conjuntos de parámetros para los experimentos del EBGA.

Experimento	n_p	n_v	$p_{m,max}$	$p_{c,min}$	μ	$f_{\eta,min}$	n_m	n_g
Experimento 1	4	2	0.0875	0.9342	5	0.7073	1	57,695
Experimento 2	4	2	0.0676	0.8544	63	0.9283	1	4,672
Experimento 3	2	2	0.0283	0.7503	36	0.6775	1	7,991
Experimento 4	5	3	0.0629	0.9838	76	0.7565	5	762

V.3.3.2.3 Parámetros del EGA-DR.

Los parámetros que se deben fijar para usar el EGA-DR son:

μ tamaño de la población de padres;

$\{n_{o,min}, n_{o,max}\}$ el número de hijos de cada individuo tendrá un valor en este intervalo;

$[p_{m,min}, p_{m,max}]$ intervalo donde tomará valores la probabilidad de mutación;

$[p_{c,min}, p_{c,max}]$ intervalo donde tomará valores la probabilidad de cruzamiento;

$[f_{\eta,min}, f_{\eta,max}]$ intervalo donde tomará valores la fracción de población de pádres, f_η. El número de iteraciones que se ejecutará el RMHC cuando sea invocado es $\lceil f_p \times \mu \rceil$.

$[\eta_\lambda, \eta_\mu]$ intervalo donde tomará valores la probabilidad de que sea invocado el RMHC;

f_{nb} es una fracción de la longitud del cromosoma de un anticuerpo. En cada iteración del RMHC el número de bits que puede ser mutado (cambiar el valor que contiene por su complemento) a la vez puede ir de 1 a $\lfloor 1 + f_{nb} \cdot l \rfloor$ según lo determine el azar. Los bits susceptibles de ser mutados son todos los del cromosoma del anticuerpo exceptuando el bit que sirve para indicar la procedencia del anticuerpo (del GA o del RMHC). Aquí se propone un intervalo para obtener algunos valores de este parámetro al azar dentro de este intervalo y hacer pruebas como las que se ha mostrado se hicieron para el EBGA en la sección anterior.

n_g número de generaciones que se ejecutará el EGA-DR.

Los parámetros constantes se muestran en la tabla V.5 y los variables en la V.6(ver sección V.3.3)

Tabla V.5 Parámetros constantes para los experimentos del EGA-DR

Variable	Valor asignado
$n_{o,min}$	1
$n_{o,max}$	2
$p_{m,min}$	0.0001
$p_{c,max}$	1
$f_{\eta,max}$	1
n_{λ}	1/8 = 0.125
n_{μ}	1/2 = 0.5

Tabla V.6 Intervalos de valores fijados para algunas de las variables.

Variable	Intervalo en el que toma su valor
μ	{2, 100}
$p_{m,max}$	[0.01, 0.1]
$p_{c,min}$	[0.5, 1]
$f_{\eta,min}$	[0.5, 1]
f_{nb}	[0, 0.1]

Como en el caso del EBGA, el EGA-DR se ejecuta un número determinado de genera-ciones. Para poder comparar su desempeño con los otros dos algoritmos aquí propuestos, se fija un número de evaluaciones. Para obtener una estimación del número de generaciones a las que equivale un número de evaluaciones del EGA-DR se usa la ecuación V.4.

$$n_g = \left\lfloor \frac{n_{evals}}{n_{o,max} \cdot \mu(1-\bar{\eta}) + \mu \cdot \bar{\eta} \cdot \overline{f_{\eta}}} + 0.5 \right\rfloor \qquad \text{(V.4)}$$

donde $\bar{\eta}$ y $\overline{f_{\eta}}$ se definen en (V.3b) y (V.3c). En las columnas 2 a 6 de la tabla V.7 se muestran valores obtenidos al azar dentro de los intervalos especificados en la tabla V.6. La séptima columna de la tabla, el número de generaciones, fue calculado con la ecuación V.4 fijando un número de evaluaciones, n_{evals}, de 500,000.

Tabla V.7 Conjuntos de parámetros para los experimentos del EGA-DR.

Experimento	$p_{m,max}$	$p_{c,min}$	μ	$f_{\eta,min}$	f_{nb}	n_g
Experimento 1	0.0875	0.9342	5	0.7073	0.9852	57,695
Experimento 2	0.0676	0.8544	63	0.9283	0.9514	4,672
Experimento 3	0.0283	0.7503	36	0.6775	0.6070	7,991
Experimento 4	0.0629	0.9838	76	0.7565	0.6393	762

V.3.3.2.4 Parámetros del EAA.

Los parámetros que se deben fijar para usar el EAA son:

μ tamaño de la población de padres;

$\{n_{o,min}, n_{o,max}\}$ el número de hijos de cada individuo tendrá un valor en este intervalo;
$[p_{m,min}, p_{m,max}]$ intervalo donde tomará valores la probabilidad de mutación;
$[p_{c,min}, p_{c,max}]$ intervalo donde tomará valores la probabilidad de cruzamiento;
T_{max} máximo umbral de probabilidad para saltar a un peor punto, el valor del menor umbral es cero;
f_{nea} es una fracción. El número de mutaciones puntuales que sufrirá cada uno de los anticuerpos antes de probar si debe aplicarse el operador de cruzamiento está dado por la ecuación $\lfloor f_{nea} \times l_v \rfloor$ donde l_v es la longitud de la subcadena del genoma del individuo que representa un punto en *S*. Debe ser fijado por el usuario.
$T_{saltos\ buenos}$ es una fracción. Sirve de umbral, si la fracción de saltos a mejores puntos es menor que él se aplica el operador de cruzamiento, si no, se realizan otras $\lfloor f_{nea} \times l_v \rfloor$ mutraciones puntuales a cada uno de los individuos de la población;
n_{evals} número de evaluaciones que se ejecutará la función.

Los parámetros constantes y los valores que se les asignaron se muestran en la tabla V.8, los variables en la tabla V.9 (ver V.3.3).

Tabla V.8 Variables a los que se les ha asignado un solo valor.

Variable	Valor asignado
$n_{o,min}$	1
$n_{o,max}$	2
$p_{m,min}$	0.0001
$p_{c,max}$	1

Tabla V.9 Variables a las que se les asignan intervalos

Variable	Intervalo asignado
$p_{m,max}$	[0.01, 0.1]
$p_{c,max}$	[0.5, 1]
μ	{2, 10}
T_{max}	[0, 0.5]
f_{nea}	[0.01, 0.5]
$T_{saltos\ buenos}$	[0.01, 0.5]

Los valores obtenidos al azar para los experimentos del EAA se muestran en la tabla V.10. Se ha fijado el número de evaluaciones , n_{evals}, en 500,000.

Tabla V.10 Datos asignados a los parámetros variables de manera aleatoria tomados de los intervalos mostrados en la tabla V.11.

Experimento	$p_{m,max}$	$p_{c,min}$	μ	T_{max}	f_{nea}	$T_{saltos\ buenos}$
Experimento 1	0.0370	0.6742	4	0.0314	0.3261	0.0620
Experimento 2	0.0484	0.7834	3	0.3159	0.1413	0.3360
Experimento 3	0.0834	0.9006	7	0.2880	0.2351	0.2688
Experimento 4	0.0420	0.6236	4	0.4426	0.4995	0.2024

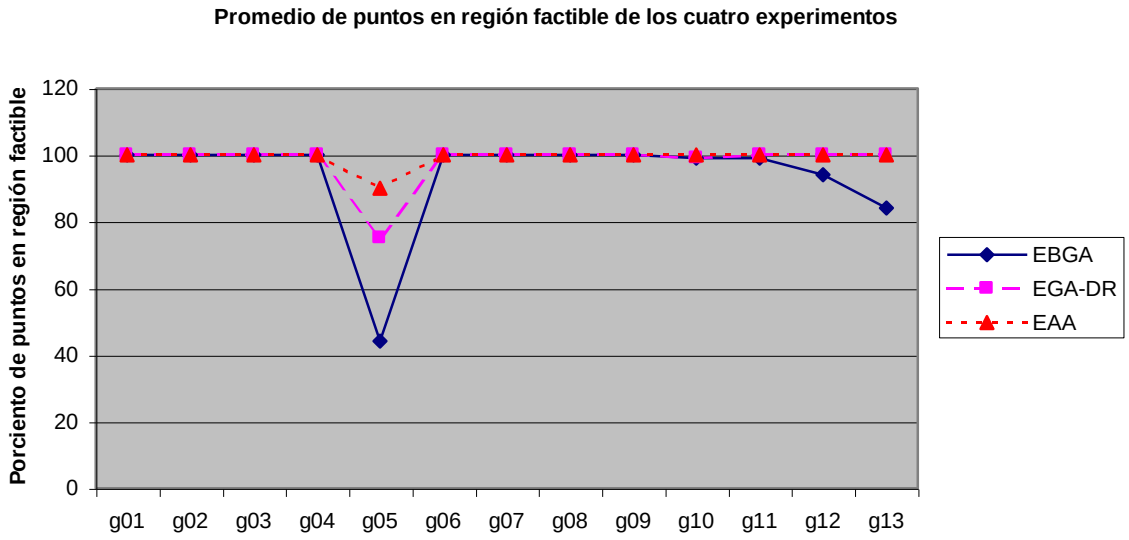
V.3.3.2.5 Los tamaños de los genomas.
Los tamaños de los genomas de las bibliotecas y de los anticuerpos se reportan en el apéndice B en la sección B.I para el EBGA, B.II para el EGA-DR y B.III para el EAA.

V.3.3.3 Los resultados.

Los resultados de los experimentos se muestran condensados en forma de gráficas en la siguiente subsección, las tablas de donde provienen se muestran en la subsección que sigue a ésta.

V.3.3.3.1 Gráficas de los resultados de los primeros cuatro experimentos del EBGA, el EGA-DR y el EAA.

En la gráfica V.1 se muestran los promedios de los porcentajes de los puntos encontrados en la región factible de cada uno de los trece problemas por cada uno de los tres algoritmos propuestos en este trabajo para los primeros cuatro experimentos.

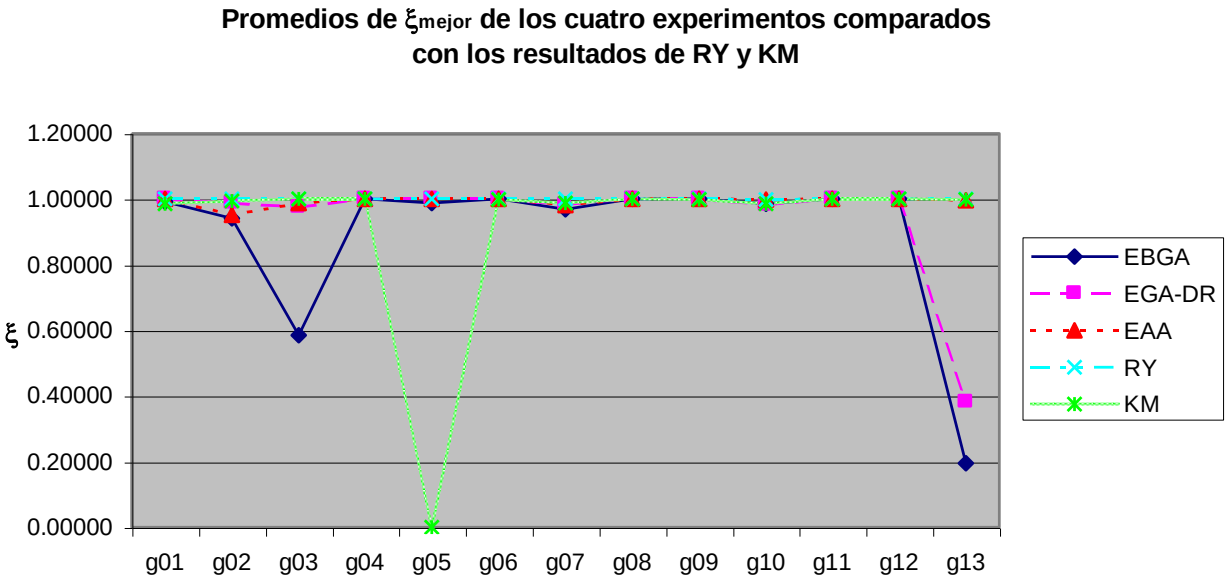


Gráfica V.1 Promedio de los porcentajes de puntos en región factible encontrados en los cuatro experimentos por cada uno de los sistemas inmunes artificiales propuestos.

Se puede observar en esta gráfica que para el EGA-DR y el EAA el mejor punto encontrado siempre cayó en región factible en cada una de las 30 corridas de cada uno de los cuatro experimentos para todos los problemas excepto el g05. Para este problema el EGA-DR encontró un promedio de aproximadamente el 75% de puntos en región factible y el EAA un promedio de aproximadamente 90%. El EBGA encontró por lo menos un punto en región factible para cada una de las corridas de todos los problemas excepto el g05, g10, g11, g12 y g13. Para los problemas g10 y g11 encontró un punto factible en casi el 100% de las corridas, en el problema g12 para aproximadamente un 95%, en el g13 para aproximadamente un 85% y para el g05 aproximadamente un 45%. La gráfica se realizó con los datos de las sextas columnas de las tablas V11 a 13.

Se puede concluir que no obstante el reducido tamaño de las regiones factibles de la mayoría de los problemas el EGA-DR y el EAA no tuvieron muchos problemas, el EBGA tuvo algunos, sobre todo para el problema g05. Pero en general el mejor punto de la gran mayoría de las corridas cayó en región factible.

En la gráfica V.2 se muestra el promedio del parámetro ξ_{mejor} de los cuatro experimentos para cada uno de los trece problemas para cada uno de los tres algoritmos. Se graficó con los datos en las tablas V.14 a 16 para los tres métodos propuestos, V.41 para el RY y V.42 para el .KM



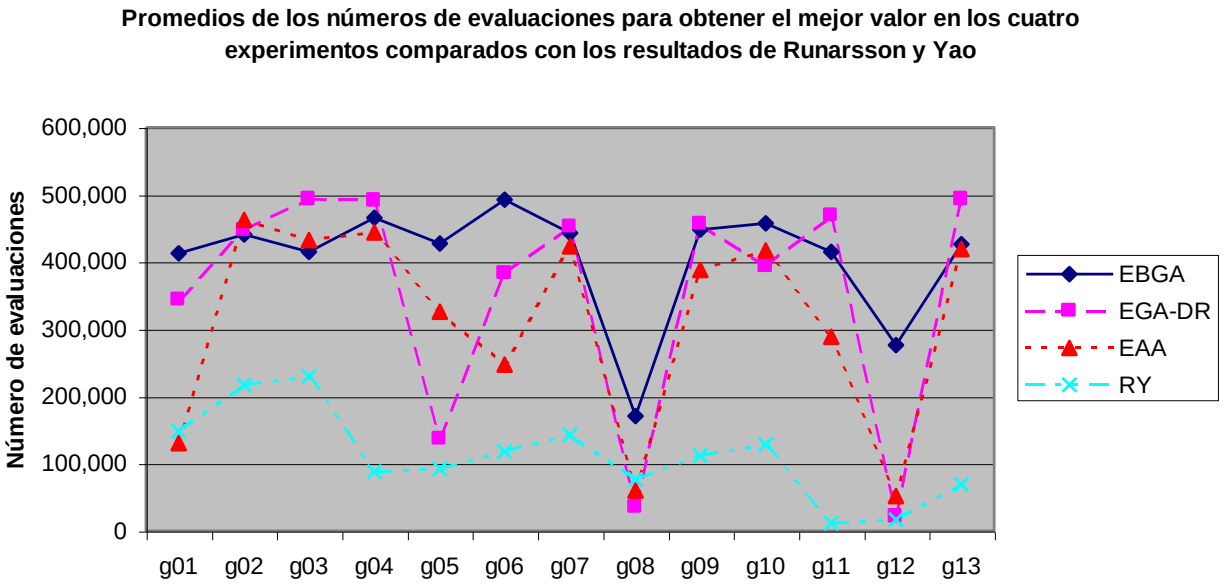
Gráfica V.2 Promedios del parámetro ξ_{mejor} (ver V.3.3) para los tres algoritmos propuestos. También se muestran los mejores valores encontrados para cada uno de los trece problemas por el método de Runarsson y Yao, RY, y por el de Koziel y Michalewicz, KM.

El parámetro ξ toma valores entre 0 y 1 inclusive, su mejor valor es cuando vale 1 y el peor cuando vale 0. En la gráfica se puede observar que para los algoritmos propuestos y la mayoría de las funciones el parámetro ξ_{mejor} tiende a 1. El EBGA tiene dificultades fuertes para las funciones g03 y g13. El EAA tiene dificultades pequeñas con el problema g02. El EGA-DR tiene dificultades fuertes con el problema g13. Obsérvese que los datos reportados en [RY00] muestran un buen comportamiento para RY en todas las funciones y los datos reportados en [RY00] (tomados de [Mic95] y [KM99]) para KM muestran que éste se comporta bien para todos los problemas excepto el g05 en el que presenta fuertes dificultades (en [KM99] se reporta para todos los resultados de este problema "*did not provided quality results*", aquí se le a asignado a cada una de las ξ correspondientes el valor de 0).

En la gráfica V.3 se muestran los promedios de los números de evaluaciones requeridos para encontrar los mejores valores mostrados en la gráfica V.2. De los métodos mostrados, en general el que menos evaluaciones de la función objetivo y sus restricciones requiere para encontrar el mejor valor es el método de RY (los valores fueron obtenidos a partir de los datos reportados en [RY00]). Solamente es superado por el EAA en los problemas g01 y g08, y por el EGA-DR en el problema g08. De los tres algoritmos propuestos el mejor en este rubro es el EAA (supera a los otros dos en los problemas g01, g04, g06, g07, g09, g11 y g13), le sigue el EGA-DR (supera a los otros dos en los problemas g05, g08, g10 y g12) y el último es el EBGA (el único en el que supera a los otros dos es en el problema g03 y es en el que no puede encontrar un buen valor según lo muestra la gráfica V.2).

No se muestra el número de evaluaciones requeridas para encontrar el mejor valor de KM porque los autores no lo reportan, en [KM99] solamente reportan el número total de

iteraciones usado para probar su método, de este dato sólo se puede obtener el número total de evaluaciones usado. Se obtuvo graficando los datos de las sextas columnas de las tablas V.17 a.19, y de la columna n. evals. de la tabla V.41.

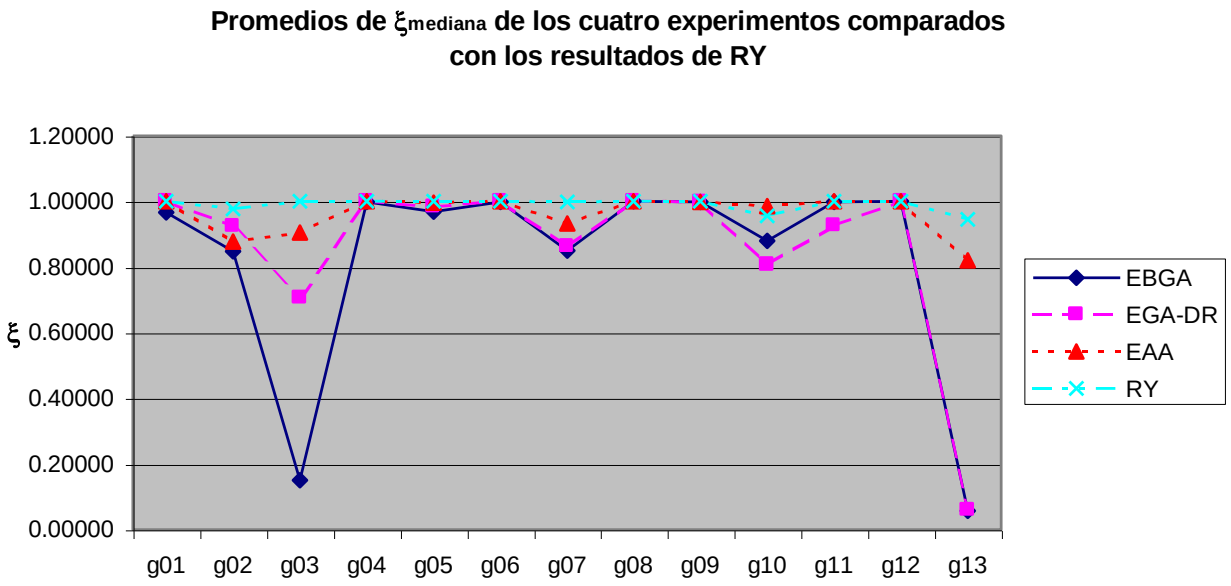


Gráfica V.3 Promedios del número de evaluaciones para obtener los mejores valores de cada uno de los problemas en los cuatro experimentos.

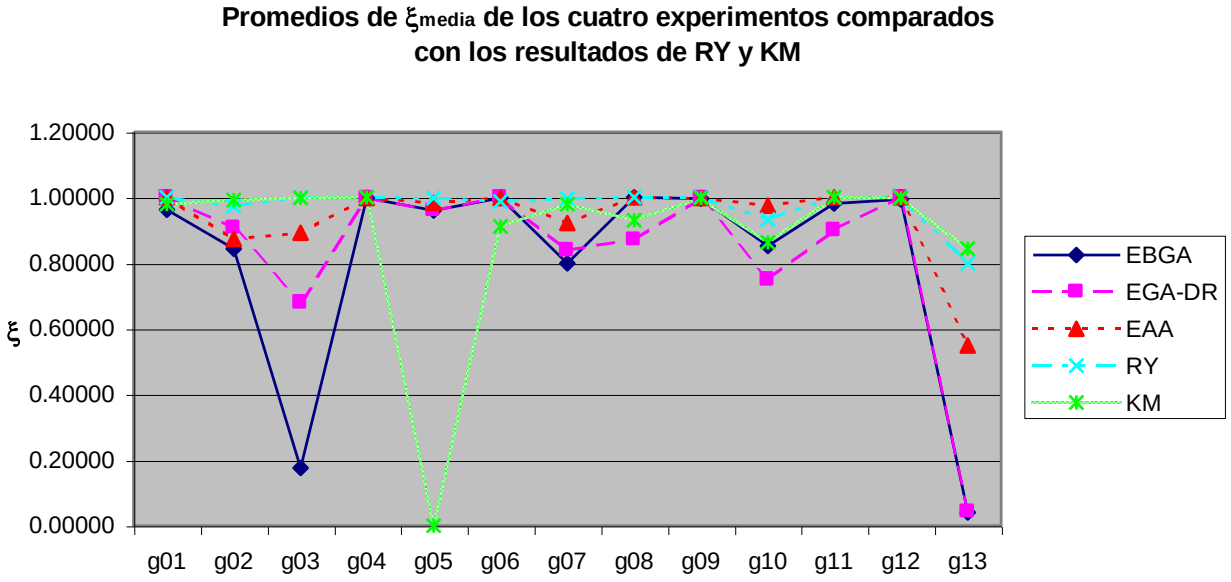
En la gráfica V.4 se muestran los promedios de los parámetros $\xi_{mediana}$ obtenidos por los tres algoritmos propuestos para los trece problemas en los cuatro experimentos, también se muestran los datos de RY obtenidos de [RY00]. Se obtuvo graficando las tablas V.20, V.21 y V.22 y la columna $\xi_{mediana}$ de la tabla V.41.

Recuérdese que la mediana es el dato que queda a la mitad de una lista *ordenada* que tiene un número impar de datos y es el promedio aritmético de los dos datos que quedan a la mitad en una lista *ordenada* que tiene un número par de datos ([WM92]p.207). Es una medida interesante, porque da una idea de que tan bien está la mitad de los datos de una lista, si la mediana de una lista es buena esto significa que la mitad de los datos de la lista son tan buenos o mejores que ella. Esta información no la da la media, por ejemplo, si la mitad de los datos de una lista es buena y la otra mitad es muy mala, la media obtenida será mala y no reflejará que la mitad de los datos obtenidos fue buena.

En este rubro el método que resulta mejor es el RY, solamente es superado por el EAA en el problema g10. De los tres algoritmos propuestos el mejor es el EAA, le sigue el EGA-DR y el último es el EBGA.



Gráfica V.4 Promedio de $\xi_{mediana}$ obtenida por los tres algoritmos propuestos para los trece problemas en los cuatro experimentos y por RY según los datos reportados en [RY00].

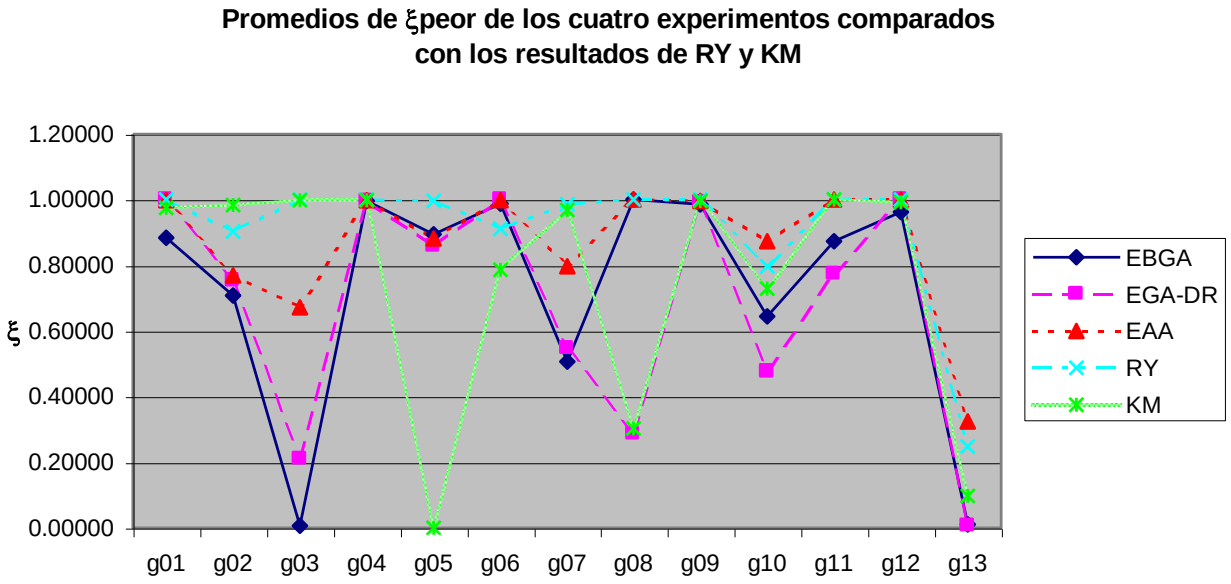


Gráfica V.5 Promedios de ξ_{media} para los tres algoritmos propuestos comparados con RY y KM.

En esta gráfica se observa que el mejor algoritmo es el RY, solamente es superado por el EAA en g10 y por KM en g13 y ligeramente en g02. Le sigue el EAA (supera a KM en g05, g06, Nótese que el EAA en general tiene un buen desempeño, prácticamente "empata" con el RY en los problemas g04, g06, g08, g09, g11, g12, casi empata en el g01 y lo supera en el g10, además no sufre una caída fuerte en ninguno de los problemas como lo hace el EBGA en el g03 y el g13 o el EGA-DR en el g13.

En la gráfica V.5 se muestran los promedios del parámetro ξ_{media} para los tres algoritmos propuestos, RY y KM (obtenidos de [RY00] y [KM99]).

g08 y g10 es superado por KM en g02, g03, g07 y g13; pero un hecho importante hace la diferencia: EAA si pudo encontrar un valor de calidad en g05 mientras que KM no). Después va KM, sigue el EGA-DR y por último el EBGA. Se obtuvo de las tablas V.23 a V.25, la columna ξ_{media} de la tabla V.41, y la columna ξ_{media} de la tabla V.42.
En la gráfica V.6 se muestran los promedios de ξ_{peor} para los primeros cuatro experimentos comparados con RY y KM (datos tomados de [RY00] y [KM99])



Gráfica V.6 Promedio del parámetro ξ_{peor} de los cuatro primeros experimentos comparados con RY y KM.

En esta gráfica se puede observar que el mejor algoritmo es el RY, solamente superado por el EAA en los problemas g10 y g13 y por KM en el g02. Le sigue el EAA, luego el KM, después el EGA-DR y por último el EBGA. Se obtuvo de las tablas V.26 a 28, de la columna 5 de la tabla V.41, y la 3 de la V.42.

V.3.3.3.2 Tablas de los resultados de los cuatro experimentos del EBGA, el EGA-DR y el EAA.

La gráfica V.1, el porcentaje de puntos factibles, muestra los puntos de las sextas columnas de las tablas V.11, V.12 y V.13.

Tabla V.11 Porcentajes de puntos factibles para cada uno de los cuatro experimentos el EBGA.

Problema	Experimento 1	Experimento 2	Experimento 3	Experimento 4	Promedio
g01	100	100	100	100	100
g02	100	100	100	100	100
g03	100	100	100	100	100
g04	100	100	100	100	100
g05	43.33	53.33	76.67	3.33	44.17
g06	100	100	100	100	100
g07	100	100	100	100	100
g08	100	100	100	100	100
g09	100	100	100	100	100
g10	100	100	96.67	100	99.17
g11	100	100	100	96.67	99.17
g12	100	100	100	76.67	94.17
g13	100	100	96.67	40	84.17

Tabla V.12. Porcentajes de puntos factibles para cada uno de los cuatro experimentos del EGA-DR

Problema	Experimento 1	Experimento 2	Experimento 3	Experimento 4	Promedio
g01	100	100	100	100	100
g02	100	100	100	100	100
g03	100	100	100	100	100
g04	100	100	100	100	100
g05	66.67	86.67	53.33	93.33	75
g06	100	100	100	100	100
g07	100	100	100	100	100
g08	100	100	100	100	100
g09	100	100	100	100	100
g10	100	100	96.67	100	99.17
g11	100	100	100	100	100
g12	100	100	100	100	100
g13	100	100	100	100	100

Tabla V.13. Porcentajes de puntos factibles para cada uno de los cuatro experimentos del EAA

Problema	Experimento 1	Experimento 2	Experimento 3	Experimento 4	Promedio
g01	100	100	100	100	100
g02	100	100	100	100	100
g03	100	100	100	100	100
g04	100	100	100	100	100
g05	100	86.67	76.67	96.67	90
g06	100	100	100	100	100
g07	100	100	100	100	100
g08	100	100	100	100	100
g09	100	100	100	100	100
g10	100	100	100	100	100
g11	100	100	100	100	100
g12	100	100	100	100	100
g13	100	100	100	100	100

La gráfica V.2 se obtuvo graficando los puntos en las tablas V.14, V.15 y V.16, que son los promedios del parámetro ξ_{mejor} de los cuatro experimentos del EBGA, el EGA-DR y el EAA. Los puntos de los métodos de RY y de KM se obtuvieron de graficar la columna del parámetro ξ_{mejor} de las tablas V.41 y V.42.

Tabla V.14 Promedios del parámetro ξ_{mejor} para el EBGA. Estos datos son los promedios de las columnas ξ_{mejor} de las tablas V.29 , V.30, V.31 y V.32.

Problema	Promedio de ξ_{mejor} para el EBGA
g01	0.99181
g02	0.94212
g03	0.58555
g04	0.99996
g05	0.98833
g06	0.99990
g07	0.96934
g08	1.00000
g09	0.99982
g10	0.98645
g11	0.99987
g12	1.00000
g13	0.19455

Tabla V.15 Promedios del parámetro ξ_{mejor} para el EGA-DR. Estos datos son los promedios de las columnas ξ_{mejor} de las tablas V.33 , V.34 , V.35 y V.36.

Problema	Promedio de ξ_{mejor} para el EGA-DR
g01	1.00000
g02	0.98806
g03	0.97459
g04	1.00000
g05	0.99972
g06	1.00000
g07	0.98461
g08	1.00000
g09	0.99979
g10	0.98073
g11	0.99977
g12	1.00000
g13	0.38201

Tabla V.16 Promedios del parámetro ξ_{mejor} para el EAA. Estos datos son los promedios de las columnas ξ_{mejor} de las tablas V.37 , V.38, V.39 y V.40.

Problema	Promedio de ξ_{mejor} para el EAA
g01	1.00000
g02	0.95291
g03	0.98735
g04	0.99998
g05	1.00000
g06	0.99998
g07	0.98136
g08	1.00000
g09	0.99978
g10	0.99949
g11	0.99987
g12	1.00000
g13	0.99703

La gráfica V.3 se obtuvo graficando los datos de las sextas columnas de las tablas V.17, V.18 y V.19, que son los datos del número de iteraciones del EBGA, el EGA-DR y el EAA para obtener los mejores puntos. Los datos del método RY se graficaron de la columna n. evals. de la tabla V.41.

V.17 Número promedio[1] de iteraciones usados para obtener los mejores puntos del EBGA

Problema	Experimento 1	Experimento 2	Experimento 3	Experimento 4	Promedio
g01	297646	442085	431827	478376	412483
g02	329770	506086	486113	439973	440485
g03	341493	417037	434560	465020	414527
g04	374934	489860	507221	490572	465647
g05	350292	436301	438201	485502	427574
g06	408526	532207	523487	506883	492776
g07	329974	474944	493869	473966	443188
g08	105412	107436	63324	406118	170572
g09	353460	474688	493940	471064	448288
g10	347024	508873	487424	484332	456913
g11	358969	465096	391475	445600	415285
g12	189282	281526	109707	522558	275768
g13	318963	459555	458823	465551	425723

[1] Cada número mostrado en las columnas 2 a 5 de las tablas V.17, V.18 y V.19 es el promedio del número de iteraciones requerido para encontrar el mejor valor de cada una de las 30 corridas independientes para cada problema.

V.18 Número promedio de iteraciones usados para obtener los mejores puntos del EGA-DR.

Problema	Experimento 1	Experimento 2	Experimento 3	Experimento 4	Promedio
g01	402513.8333	323453	207558	439975	343375
g02	399679.2333	511574	467358	407262	446468
g03	419979.6667	549304	549926	453926	493284
g04	440940.2	516939	525293	481072	491061
g05	129840.4667	126803	130275	157928	136211
g06	270340.9333	491792	332653	438220	383251
g07	401517.8	477235	486581	444029	452341
g08	47989.36667	29398	45022	17909	35080
g09	400520.9333	455399	507776	461489	456296
g10	413361	394765	345771	417669	392891
g11	495986.7667	396710	506809	476624	469033
g12	16051.4	24362	35258	10335	21502
g13	426620.7667	528937	514867	502153	493144

V.19 Número promedio de iteraciones usados para obtener los mejores puntos del EAA.

Problema	Experimento 1	Experimento 2	Experimento 3	Experimento 4	Promedio
g01	92264	222267	146174	61232	130484
g02	461897	458282	445095	483702	462244
g03	478413	317537	459477	473966	432348
g04	491622	400474	404368	477703	443542
g05	338172	261887	376736	326560	325839
g06	277575	307333	192984	209580	246868
g07	395956	435699	426527	435304	423371
g08	8445	156172	65569	10117	60076
g09	415700	287615	394936	451879	387533
g10	390633	375085	452399	449860	416994
g11	334287	195465	234377	389938	288517
g12	74939	19234	77788	34066	51507
g13	474088	353308	402379	444734	418627

Los puntos de la gráfica V.4 se obtuvieron graficando los datos de las tablas V.20, V.21 y V.22 que son los promedios del parámetro $\xi_{mediana}$ para el EBGA, el EGA-DR y el EAA. Los datos del método de RY se graficaron de la columna $\xi_{mediana}$ de la tabla V.41.

Tabla V.20 Promedios del parámetro $\xi_{mediana}$ para el EBGA. Estos datos son los promedios de las columnas $\xi_{mediana}$ de las tablas V.29 , V.30 , V.31 y V.32.

Problema	Promedio de $\xi_{mediana}$ para el EBGA
g01	0.96702
g02	0.84809
g03	0.14983
g04	0.99927
g05	0.96939
g06	0.99948
g07	0.84977
g08	1.00000
g09	0.99777
g10	0.87969
g11	0.99780
g12	0.99996
g13	0.05710

Tabla V.21 Promedios del parámetro $\xi_{mediana}$ para el EGA-DR. Estos datos son los promedios de las columnas $\xi_{mediana}$ de las tablas V.33, V.34 , V.35 y V.36.

Problema	Promedio de $\xi_{mediana}$ para el EGA-DR
g01	1.00000
g02	0.92361
g03	0.70614
g04	0.99982
g05	0.98466
g06	1.00000
g07	0.86335
g08	1.00000
g09	0.99882
g10	0.80696
g11	0.92656
g12	1.00000
g13	0.05931

Tabla V.22 Promedios del parámetro $\xi_{mediana}$ para el EAA. Estos datos son los promedios de las columnas $\xi_{mediana}$ de las tablas V.37 , V.38, V.39 y V.40.

Problema	Promedio de $\xi_{mediana}$ para el EAA
g01	0.99990
g02	0.87770
g03	0.90522
g04	0.99972
g05	0.99682
g06	0.99977
g07	0.93351
g08	1.00000
g09	0.99918
g10	0.98628
g11	0.99987
g12	1.00000
g13	0.82048

Los puntos de la gráfica V.5 se obtuvieron graficando los datos de las tablas V.23, V.24 y V.25 que son los promedios del parámetro ξ_{media} para el EBGA, el EGA-DR y el EAA. Los datos del método de RY se graficaron de la columna ξ_{media} de la tabla V.41, y los del KM de la columna ξ_{media} de la tabla V.42.

Tabla V.23 Promedios del parámetro ξ_{media} para el EBGA. Estos datos son los promedios de las columnas ξ_{media} de las tablas V.29 , V.30 , V.31 y V.32.

Problema	Promedio de ξ_{m} para el EBG
g01	0.96170
g02	0.84367
g03	0.17689
g04	0.99909
g05	0.96045
g06	0.99883
g07	0.80016
g08	1.00000
g09	0.99693
g10	0.85221
g11	0.98068
g12	0.99510
g13	0.03938

Tabla V.24 Promedios del parámetro ξ_{media} para el EGA-DR. Estos datos son los promedios de las columnas ξ_{media} de las tablas V.33, V.34, V.35 y V.36.

Problema	Promedio de ξ_{m} para el EGA-D
g01	1.00000
g02	0.90806
g03	0.67970
g04	0.99953
g05	0.96131
g06	1.00000
g07	0.84062
g08	0.87154
g09	0.99816
g10	0.75134
g11	0.90091
g12	1.00000
g13	0.04272

Tabla V.25 Promedios del parámetro ξ_{media} para el EAA. Estos datos son los promedios de las columnas ξ_{media} de las tablas V.37 , V.38, V.39 y V.40.

Problema	Promedio de ξ_{m} para el EAA
g01	0.99981
g02	0.87371
g03	0.89216
g04	0.99956
g05	0.98347
g06	0.99958
g07	0.92294
g08	1.00000
g09	0.99906
g10	0.97590
g11	0.99987
g12	1.00000
g13	0.54927

Los puntos de la gráfica V.6 se obtuvieron graficando los datos de las tablas V.26, V.27 y V.28 que son los promedios del parámetro ξ_{peor} para el EBGA, el EGA-DR y el EAA. Los datos del método de RY se graficaron de la tabla V.41, y los del KM de la tabla V.42.

Tabla V.26 Promedios del parámetro ξ_{peor} para el EBGA. Estos datos son los promedios de las columnas ξ_{peor} de las tablas V.29 , V.30 , V.31 y V.32.

Problema	Promedio de ξ_{peor} para el EBGA
g01	0.88477
g02	0.70820
g03	0.00684
g04	0.99684
g05	0.89457
g06	0.98826
g07	0.50646
g08	1.00000
g09	0.98677
g10	0.64418
g11	0.87246
g12	0.96223
g13	0.01105

Tabla V.27 Promedios del parámetro ξ_{peor} para el EGA-DR. Estos datos son los promedios de las co-lumnas ξ_{peor} de las tablas V33, V.34, V.35 y V.36.

Problema	Promedio de ξ_{peor} para el EGA-DR
g01	0.99999
g02	0.75532
g03	0.20990
g04	0.99728
g05	0.85772
g06	1.00000
g07	0.54693
g08	0.28646
g09	0.99127
g10	0.47699
g11	0.77321
g12	1.00000
g13	0.00667

Tabla V.28 Promedios del parámetro ξ_{peor} para el EAA. Estos datos son los promedios de las columnas ξ_{peor} de las tablas V.37, V.38, V.39 y V.40.

Problema	Promedio de ξ_{peor} para el EAA
g01	0.99895
g02	0.76933
g03	0.67213
g04	0.99814
g05	0.88095
g06	0.99815
g07	0.79687
g08	1.00000
g09	0.99718
g10	0.87342
g11	0.99988
g12	1.00000
g13	0.32468

En la sección V.3.3.2.1 se explica en que consiste cada experimento. Las tablas V.29, V.30, V.31 y V.32 muestran los parámetros ξ obtenidos de cada uno de los cuatro experimentos del EBGA. Para obtener estas tablas se usaron las tablas V.43 a 46.

Tabla V.29 Tabla obtenida a partir del experimento 1 del EBGA.

Problema	ξ_{mejor}	$\xi_{mediana}$	ξ_{media}	ξ_{peor}
g01	0.99999	0.98443	0.97323	0.89175
g02	0.93547	0.81819	0.81541	0.63929
g03	0.49175	0.07853	0.11610	0.00380
g04	0.99998	0.99954	0.99937	0.99749
g05	0.99851	0.98237	0.96440	0.86660
g06	1.00000	0.99998	0.99997	0.99980
g07	0.96066	0.81041	0.74788	0.44396
g08	1.00000	1.00000	1.00000	1.00000
g09	0.99974	0.99763	0.99708	0.98881
g10	0.98983	0.86848	0.83328	0.64816
g11	0.99987	0.99785	0.96852	0.81092
g12	1.00000	1.00000	1.00000	1.00000
g13	0.39771	0.06023	0.03553	0.00467

Tabla V.30 Tabla obtenida a partir del experimento 2 del EBGA.

Problema	ξ_{mejor}	$\xi_{mediana}$	ξ_{media}	ξ_{peor}
g01	0.99998	0.99333	0.99188	0.97761
g02	0.96234	0.92947	0.91746	0.82708
g03	0.80407	0.20728	0.21666	0.00585
g04	0.99999	0.99968	0.99946	0.99774
g05	0.99978	0.99184	0.97683	0.89286
g06	1.00000	0.99996	0.99995	0.99982
g07	0.99134	0.92113	0.90067	0.71105
g08	1.00000	1.00000	1.00000	1.00000
g09	0.99983	0.99876	0.99845	0.99472
g10	0.99506	0.93316	0.90434	0.71733
g11	0.99988	0.99675	0.97494	0.81024
g12	1.00000	1.00000	1.00000	1.00000
g13	0.06071	0.05400	0.03259	0.00489

Tabla V.31 Tabla obtenida a partir del experimento 3 del EBGA.

Problema	ξ_{mejor}	$\xi_{mediana}$	ξ_{media}	ξ_{peor}
g01	1.00000	0.99986	0.99970	0.99759
g02	0.98117	0.82942	0.83773	0.66435
g03	0.35863	0.12918	0.14887	0.00628
g04	0.99998	0.99837	0.99814	0.99418
g05	0.99879	0.94711	0.94431	0.86258
g06	0.99992	0.99933	0.99846	0.99040
g07	0.98336	0.86116	0.76431	0.29812
g08	1.00000	1.00000	1.00000	1.00000
g09	0.99997	0.99660	0.99467	0.97433
g10	0.99811	0.85061	0.82307	0.52016
g11	0.99987	0.99699	0.98400	0.91540
g12	1.00000	1.00000	1.00000	1.00000
g13	0.25847	0.06014	0.04162	0.00744

Tabla V.32 Tabla obtenida a partir del experimento 4 del EBGA.

Problema	ξ_{mejor}	$\xi_{mediana}$	ξ_{media}	ξ_{peor}
g01	0.96726	0.89044	0.88199	0.67213
g02	0.88950	0.81526	0.80408	0.70207
g03	0.68776	0.18432	0.22595	0.01142
g04	0.99990	0.99948	0.99937	0.99797
g05	0.95625	0.95625	0.95625	0.95625
g06	0.99970	0.99864	0.99696	0.96300
g07	0.94199	0.80638	0.78780	0.57269
g08	1.00000	1.00000	1.00000	1.00000
g09	0.99974	0.99810	0.99751	0.98922
g10	0.96281	0.86650	0.84815	0.69107
g11	0.99987	0.99962	0.99526	0.95327
g12	1.00000	0.99985	0.98039	0.84891
g13	0.06133	0.05403	0.04778	0.02719

Las tablas V.33, V.34, V.35 y V.36 muestran los parámetros ξ obtenidos de cada uno de los cuatro experimentos del EGA-DR. Para obtener estas tablas se usaron las tablas V.47 a 50.

Tabla V.33 Tabla obtenida a partir del experimento 1 del EGA-DR.

Problema	ξ_{mejor}	$\xi_{mediana}$	ξ_{media}	ξ_{peor}
g01	1.00000	1.00000	0.99999	0.99996
g02	0.99871	0.94874	0.93203	0.82767
g03	0.99584	0.93432	0.84321	0.30402
g04	1.00000	0.99999	0.99993	0.99924
g05	0.99969	0.97498	0.95530	0.85083
g06	1.00000	1.00000	1.00000	1.00000
g07	0.97769	0.84065	0.83957	0.51487
g08	1.00000	1.00000	0.90722	0.30414
g09	0.99996	0.99940	0.99937	0.99745
g10	0.97454	0.77890	0.73215	0.44836
g11	0.99987	0.93867	0.89990	0.77733
g12	1.00000	1.00000	1.00000	1.00000
g13	0.58533	0.06261	0.04054	0.00402

Tabla V.34 Tabla obtenida a partir del experimento 2 del EGA-DR.

Problema	ξ_{mejor}	$\xi_{mediana}$	ξ_{media}	ξ_{peor}
g01	1.00000	1.00000	1.00000	1.00000
g02	0.98867	0.91791	0.89622	0.68090
g03	0.99884	0.48418	0.51853	0.07929
g04	1.00000	0.99981	0.99915	0.99406
g05	0.99991	0.98932	0.96457	0.83958
g06	1.00000	1.00000	1.00000	1.00000
g07	0.99384	0.84744	0.79211	0.30323
g08	1.00000	1.00000	0.88283	0.26834
g09	0.99988	0.99817	0.99729	0.98952
g10	0.97935	0.83356	0.74575	0.41747
g11	0.99997	0.96084	0.91015	0.75208
g12	1.00000	1.00000	1.00000	1.00000
g13	0.10098	0.05530	0.04765	0.00854

Tabla V.35 Tabla obtenida a partir del experimento 3 del EGA-DR.

Problema	ξ_{mejor}	$\xi_{mediana}$	ξ_{media}	ξ_{peor}
g01	1.00000	1.00000	1.00000	1.00000
g02	0.97534	0.87138	0.85758	0.66146
g03	0.92003	0.53834	0.52329	0.07271
g04	0.99999	0.99950	0.99902	0.99586
g05	0.99938	0.97910	0.95508	0.83893
g06	1.00000	1.00000	1.00000	1.00000
g07	0.97722	0.84666	0.82182	0.58101
g08	1.00000	1.00000	0.74252	0.26923
g09	0.99935	0.99819	0.99668	0.98071
g10	0.99442	0.71501	0.66431	0.42522
g11	0.99937	0.84981	0.86167	0.75873
g12	1.00000	1.00000	1.00000	1.00000
g13	0.58590	0.05686	0.03931	0.00750

Tabla V.36 Tabla obtenida a partir del experimento 4 del EGA-DR.

Problema	ξ_{mejor}	$\xi_{mediana}$	ξ_{media}	ξ_{peor}
g01	1.00000	1.00000	1.00000	0.99999
g02	0.98951	0.95640	0.94641	0.85123
g03	0.98365	0.86771	0.83377	0.38358
g04	1.00000	1.00000	1.00000	0.99998
g05	0.99989	0.99523	0.97030	0.90153
g06	1.00000	1.00000	1.00000	1.00000
g07	0.98968	0.91867	0.90898	0.78860
g08	1.00000	1.00000	0.95361	0.30414
g09	0.99996	0.99954	0.99931	0.99741
g10	0.97462	0.90035	0.86314	0.61692
g11	0.99987	0.95690	0.93191	0.80468
g12	1.00000	1.00000	1.00000	1.00000
g13	0.25582	0.06246	0.04340	0.00663

Las tablas V.37, V.38, V.39 y V.40 muestran los parámetros ξ obtenidos de cada uno de los cuatro experimentos del EGA-DR. Para obtener estas tablas se usaron las tablas V.51 a 54.

Tabla V.37 Tabla obtenida a partir del experimento 1 del EAA.

Problema	ξ_{mejor}	$\xi_{mediana}$	ξ_{media}	ξ_{peor}
g01	1.00000	1.00000	1.00000	1.00000
g02	0.95238	0.84231	0.84857	0.73183
g03	0.99763	0.89167	0.86695	0.54850
g04	1.00000	0.99945	0.99900	0.99565
g05	1.00000	0.99605	0.98311	0.85773
g06	1.00000	1.00000	1.00000	1.00000
g07	0.97471	0.94031	0.93194	0.83498
g08	1.00000	1.00000	1.00000	1.00000
g09	0.99961	0.99916	0.99911	0.99729
g10	0.99981	0.99012	0.97992	0.89536
g11	0.99987	0.99987	0.99987	0.99987
g12	1.00000	1.00000	1.00000	1.00000
g13	0.99985	0.99497	0.47093	0.11876

Tabla V.38 Tabla obtenida a partir del experimento 2 del EAA.

Problema	ξ_{mejor}	$\xi_{mediana}$	ξ_{media}	ξ_{peor}
g01	1.00000	0.99958	0.99923	0.99579
g02	0.98644	0.93677	0.92131	0.78773
g03	0.99991	0.99565	0.99261	0.96664
g04	0.99999	0.99993	0.99986	0.99920
g05	1.00000	0.99290	0.98072	0.91751
g06	1.00000	1.00000	1.00000	1.00000
g07	0.99626	0.94349	0.93617	0.84651
g08	1.00000	1.00000	1.00000	1.00000
g09	0.99996	0.99911	0.99882	0.99570
g10	1.00000	0.99765	0.99569	0.95449
g11	0.99987	0.99987	0.99987	0.99990
g12	1.00000	1.00000	1.00000	1.00000
g13	0.98851	0.28874	0.19362	0.07760

Tabla V.39 Tabla obtenida a partir del experimento 3 del EAA.

Problema	ξ_{mejor}	$\xi_{mediana}$	ξ_{media}	ξ_{peor}
g01	1.00000	1.00000	1.00000	1.00000
g02	0.91746	0.86124	0.86118	0.80148
g03	0.95955	0.78109	0.77055	0.35752
g04	0.99997	0.99965	0.99956	0.99867
g05	0.99999	0.99842	0.97702	0.85887
g06	0.99991	0.99908	0.99835	0.99279
g07	0.96930	0.91635	0.91424	0.84861
g08	1.00000	1.00000	1.00000	1.00000
g09	0.99980	0.99918	0.99905	0.99752
g10	0.99859	0.98382	0.97622	0.87750
g11	0.99987	0.99987	0.99987	0.99988
g12	1.00000	1.00000	1.00000	1.00000
g13	0.99987	0.99937	0.99754	0.97982

Tabla V.40 Tabla obtenida a partir del experimento 4 del EAA.

Problema	ξ_{mejor}	$\xi_{mediana}$	ξ_{media}	ξ_{peor}
g01	1.00000	1.00000	1.00000	1.00000
g02	0.95537	0.87049	0.86377	0.75627
g03	0.99233	0.95246	0.93853	0.81584
g04	0.99997	0.99985	0.99981	0.99904
g05	1.00000	0.99993	0.99304	0.88971
g06	1.00000	0.99999	0.99997	0.99980
g07	0.98518	0.93389	0.90941	0.65736
g08	1.00000	1.00000	1.00000	1.00000
g09	0.99974	0.99929	0.99925	0.99821
g10	0.99954	0.97354	0.95177	0.76634
g11	0.99987	0.99987	0.99987	0.99987
g12	1.00000	1.00000	1.00000	1.00000
g13	0.99987	0.99883	0.53498	0.12254

La tabla V.41 muestra los parámetros ξ calculados de los resultados obtenidos de un experimento del método de RY mostrados en la tabla II en [RY00], también incluye el número de evaluaciones calculado en base a los datos mostrados en la misma fuente.

Tabla V.41 Datos obtenidos a partir de una corrida del método de RY

Problema	ξ_{mejor}	$\xi_{mediana}$	ξ_{media}	ξ_{peor}	n. evals.
g01	1.00000	1.00000	1.00000	1.00000	148,200
g02	0.99987	0.97783	0.97307	0.90377	217,200
g03	1.00000	1.00000	1.00000	1.00000	229,200
g04	1.00000	1.00000	1.00000	1.00000	88,200
g05	1.00000	0.99983	0.99954	0.99689	91,600
g06	1.00000	1.00000	0.98766	0.91216	118,000
g07	0.99996	0.99791	0.99721	0.98636	143,000
g08	1.00000	1.00000	1.00000	1.00000	76,200
g09	1.00000	0.99998	0.99996	0.99980	111,400
g10	0.99929	0.95615	0.93255	0.79783	128,400
g11	1.00000	1.00000	1.00000	1.00000	11,400
g12	1.00000	1.00000	1.00000	1.00000	16,400
g13	0.99987	0.94639	0.79875	0.24871	69,800

La tabla V.42 muestra los parámetros ξ obtenidos de la tabla III en [RY00].

Tabla V.42 Parámetros ξ calculados a partir de datos de experimentos llevados a cabo con el método de KM mostrados en la tabla III de [RY00].

Problema	ξ_{mejor}	ξ_{media}	ξ_{peor}
g01	0.98576	0.98055	0.97436
g02	0.99491	0.99140	0.98453
g03	0.99970	0.99890	0.99780
g04	0.99997	0.99967	0.99936
g05	0.00000	0.00000	0.00000
g06	0.99860	0.91106	0.78627
g07	0.98725	0.97905	0.96956
g08	1.00000	0.93041	0.30414
g09	0.99959	0.99922	0.99627
g10	0.98621	0.86351	0.72980
g11	1.00000	1.00000	1.00000
g12	1.00000	0.99913	0.99195
g13	0.99907	0.84297	0.09686

Las tablas V.43 a V.46 muestran los datos concentrados de las 30 corridas independientes de cada una de las trece funciones optimizadas con el EBGA.

Tabla V.43 Datos concentrados de las 30 corridas independientes del experimento 1 del EBGA.

Problema	óptimo	mejor	mediana	media	desv. est.	peor	num. evals. promedio	num. gen. promedio	% de factibles
g01(min.)	-15.000000	-14.99978	-14.76648	-14.59839	0.412335	-13.37620	297646	42887.7	100
g02(max.)	0.803619	0.751765	0.657515	0.655277	0.054794	0.513748	329770	45598.3	100
g03(max.)	1	0.491752	0.078533	0.116102	0.11315	0.003798	341493	49194.4	100
g04(min.)	-30665.539	-30664.866	-30651.301	-30646.226	17.450402	-30588.509	374934	54139.1	100
g05(min.)	5126.498	5134.157	5218.477	5315.757	221.35116	5915.620	350292	50597.4	43.33
g06(min.)	-6961.814	-6961.800	-6961.674	-6961.601	0.257934	-6960.393	408526	55473.1	100
g07(min.)	24.306	25.301	29.992	32.500	7.015623	54.748	329974	46693.8	100
g08(max.)	0.095825	0.095825	0.095825	0.095825	0	0.095825	105412	14399.0	100
g09(min.)	680.63	680.806	682.248	682.620	1.681256	688.333	353460	51434.6	100
g10(min.)	7049.331	7121.728	8116.839	8459.725	1058.314	10875.972	347024	49050.3	100
g11(min.)	0.750	0.750	0.752	0.774	0.04926	0.925	358969	50820.8	100
g12(max.)	1	1.000	1.000	1.000	0	1.000	189282	26657.9	100
g13(min.)	0.05395	0.13565	0.89577	1.51826	2.305042	11.56321	318963	47823.4	100

Tabla V.44 Datos concentrados de las 30 corridas independientes del experimento 2 del EBGA.

Problema	óptimo	mejor	mediana	media	desv. est.	peor	num. evals. promedio	num. gen. promedio	% de factibles
g01(min.)	-15.000000	-14.99969	-14.89997	-14.87816	0.105001	-14.66420	442085	4315.4	100
g02(max.)	0.803619	0.773356	0.746938	0.737288	0.02823	0.664655	506086	4538.0	100
g03(max.)	1	0.804065	0.207284	0.216659	0.174963	0.005846	417037	4233.8	100
g04(min.)	-30665.539	-30665.248	-30655.644	-30648.867	17.098552	-30596.273	489860	4637.3	100
g05(min.)	5126.498	5127.605	5168.654	5248.118	168.61215	5741.668	436301	4519.6	53.33
g06(min.)	-6961.814	-6961.798	-6961.542	-6961.485	0.297117	-6960.564	532207	4714.9	100
g07(min.)	24.306	24.518	26.387	26.987	2.249294	34.183	474944	4352.9	100
g08(max.)	0.095825	0.095825	0.095825	0.095825	0	0.095825	107436	1001.9	100
g09(min.)	680.63	680.748	681.477	681.687	0.756462	684.242	474688	4485.8	100
g10(min.)	7049.331	7084.316	7554.262	7795.011	691.61915	9827.232	508873	4524.6	100
g11(min.)	0.750	0.750	0.752	0.769	0.042908	0.926	465096	4257.8	100
g12(max.)	1	1.000	1.000	1.000	0	1.000	281526	2500.2	100
g13(min.)	0.05395	0.88869	0.99898	1.65538	1.985258	11.04178	459555	4556.6	100

Tabla V.45 Datos concentrados de las 30 corridas independientes del experimento 3 del EBGA.

Problema	óptimo	mejor	mediana	media	desv. est.	peor	num. evals. promedio	num. gen. promedio	% de factibles
g01(min.)	-15.000000	-15.00000	-14.99793	-14.99552	0.007762	-14.96382	431827	7492.7	100.00
g02(max.)	0.803619	0.788487	0.666540	0.673217	0.06622	0.533883	486113	7848.6	100.00
g03(max.)	1	0.358627	0.129184	0.148870	0.097468	0.006279	434560	7704.1	100.00
g04(min.)	-30665.539	-30664.934	-30615.571	-30608.652	46.856677	-30486.962	507221	7915.4	100.00
g05(min.)	5126.498	5132.724	5412.780	5428.838	257.55714	5943.196	438201	7233.7	76.67
g06(min.)	-6961.814	-6961.258	-6957.138	-6951.066	13.643815	-6895.013	523487	7978.2	100.00
g07(min.)	24.306	24.717	28.225	31.801	10.684849	81.531	493869	7705.3	100.00
g08(max.)	0.095825	0.095825	0.095825	0.095825	0	0.095825	63324	972.0	100.00
g09(min.)	680.63	680.647	682.949	684.275	4.151806	698.563	493940	7727.3	100.00
g10(min.)	7049.331	7062.676	8287.380	8564.715	1489.6006	13552.302	487424	7891.7	96.67
g11(min.)	0.750	0.750	0.752	0.762	0.019483	0.819	391475	6039.9	100.00
g12(max.)	1	1.000	1.000	1.000	0	1.000	109707	1669.3	100.00
g13(min.)	0.05395	0.20873	0.89700	1.29630	1.393542	7.25197	458823	7758.4	96.67

Tabla V.46 Datos concentrados de las 30 corridas independientes del experimento 4 del EBGA.

Problema	óptimo	mejor	mediana	media	desv. est.	peor	num. evals. promedio	num. gen. promedio	% de factibles
g01(min.)	-15.000000	-14.50893	-13.35655	-13.22980	0.780934	-10.08196	478376	729.6	100.00
g02(max.)	0.803619	0.714817	0.655157	0.646170	0.035132	0.564195	439973	646.1	100.00
g03(max.)	1	0.687762	0.184320	0.225948	0.141711	0.011421	465020	697.5	100.00
g04(min.)	-30665.539	-30662.347	-30649.494	-30646.285	13.558543	-30603.354	490572	723.7	100.00
g05(min.)	5126.498	5361.040	5361.040	5361.040	0	5361.040	485502	698.0	3.33
g06(min.)	-6961.814	-6959.751	-6952.311	-6940.632	45.303936	-6704.253	506883	747.7	100.00
g07(min.)	24.306	25.803	30.142	30.853	3.349074	42.442	473966	698.5	100.00
g08(max.)	0.095825	0.095825	0.095825	0.095825	0	0.095825	406118	604.0	100.00
g09(min.)	680.63	680.805	681.926	682.328	1.58968	688.050	471064	693.0	100.00
g10(min.)	7049.331	7321.629	8135.426	8311.428	584.42713	10200.560	484332	721.0	100.00
g11(min.)	0.750	0.750	0.750	0.754	0.007391	0.787	445600	689.4	96.67
g12(max.)	1	1.000	1.000	0.980	0.037066	0.849	522558	762.0	76.67
g13(min.)	0.05395	0.87974	0.99858	1.12923	0.341116	1.98382	465551	722.5	40.00

Las tablas V.47 a V.50 muestran los datos concentrados de las 30 corridas independientes de cada una de las trece funciones optimizadas con el EGA-DR.

Tabla V.47 Datos concentrados de las 30 corridas independientes del experimento 1 del EGA-DR.

Problema	óptimo	mejor	mediana	media	desv. est.	peor	num. evals. promedio	num. gen. promedio	% de factibles
g01(min.)	-15.000000	-15.00000	-14.99996	-14.99991	0.000132	-14.99944	402514	32278.2	100
g02(max.)	0.803619	0.802580	0.762424	0.748999	0.038233	0.665132	399679	32630.9	100
g03(max.)	1	0.995839	0.934319	0.843210	0.198621	0.304024	419980	32714.0	100
g04(min.)	-30665.539	-30665.507	-30665.274	-30663.540	4.859902	-30642.156	440940	31259.9	100
g05(min.)	5126.498	5128.089	5258.042	5366.383	264.96253	6025.261	129840	10108.1	66.67
g06(min.)	-6961.814	-6961.814	-6961.814	-6961.814	0	-6961.813	270341	17679.4	100
g07(min.)	24.306	24.861	28.913	28.950	4.253583	47.208	401518	31833.8	100
g08(max.)	0.095825	0.095825	0.095825	0.086934	0.022667	0.029144	47989	3029.0	100
g09(min.)	680.63	680.660	681.042	681.058	0.328769	682.371	400521	30074.8	100
g10(min.)	7049.331	7233.482	9050.334	9628.277	2310.4241	15722.544	413361	32713.0	100
g11(min.)	0.750	0.750	0.799	0.833	0.070735	0.965	495987	31255.4	100
g12(max.)	1	1.000	1.000	1.000	0	1.000	16051	1088.2	100
g13(min.)	0.05395	0.09217	0.86164	1.33065	2.394856	13.43209	426621	31102.2	100

Tabla V.48 Datos concentrados de las 30 corridas independientes del experimento 2 del EGA-DR.

Problema	óptimo	mejor	mediana	media	desv. est.	peor	num. evals. promedio	num. gen. promedio	% de factibles
g01(min.)	-15.000000	-15.00000	-15.00000	-15.00000	0	-15.00000	323453	2376.3	100
g02(max.)	0.803619	0.794514	0.737651	0.720217	0.059998	0.547186	511574	3901.6	100
g03(max.)	1	0.998835	0.484177	0.518530	0.252048	0.079290	549304	3901.8	100
g04(min.)	-30665.539	-30665.504	-30659.624	-30639.554	44.107886	-30483.250	516939	3718.5	100
g05(min.)	5126.498	5126.939	5181.822	5314.828	266.75209	6106.046	126803	1028.7	86.67
g06(min.)	-6961.814	-6961.814	-6961.814	-6961.813	0	-6961.813	491792	3504.5	100
g07(min.)	24.306	24.457	28.682	30.685	9.651939	80.158	477235	3394.6	100
g08(max.)	0.095825	0.095825	0.095825	0.084597	0.025112	0.025714	29398	233.0	100
g09(min.)	680.63	680.709	681.877	682.481	1.829443	687.836	455399	3240.2	100
g10(min.)	7049.331	7197.963	8456.910	9452.715	2517.6271	16885.714	394765	2826.7	100
g11(min.)	0.750	0.750	0.781	0.824	0.080877	0.997	396710	2818.7	100
g12(max.)	1	1.000	1.000	1.000	0	1.000	24362	189.1	100
g13(min.)	0.05395	0.53428	0.97554	1.13229	1.039842	6.31901	528937	3749.6	100

Tabla V.49 Datos concentrados de las 30 corridas independientes del experimento 3 del EGA-DR.

Problema	óptimo	mejor	mediana	media	desv. est.	peor	num. evals. promedio	num. gen. promedio	% de factibles
g01(min.)	-15.000000	-15.00000	-15.00000	-15.00000	0	-15.00000	207558	10302.4	100
g02(max.)	0.803619	0.783800	0.700259	0.689167	0.066984	0.531565	467358	25455.9	100
g03(max.)	1	0.920025	0.538344	0.523285	0.213456	0.072713	549926	25365.6	100
g04(min.)	-30665.539	-30665.229	-30650.311	-30635.440	35.231354	-30538.437	525293	24586.3	100
g05(min.)	5126.498	5129.658	5235.942	5367.586	304.53747	6110.771	130275	9416.4	53.33
g06(min.)	-6961.814	-6961.814	-6961.814	-6961.814	0.000092	-6961.813	332653	15821.2	100
g07(min.)	24.306	24.873	28.708	29.576	3.813312	41.834	486581	22527.5	100
g08(max.)	0.095825	0.095825	0.095825	0.071152	0.032435	0.025799	45022	2065.2	100
g09(min.)	680.63	681.072	681.861	682.898	2.987889	694.020	507776	23222.0	100
g10(min.)	7049.331	7088.868	9859.003	10611.510	2553.3234	16578.012	345771	16665.9	96.66667
g11(min.)	0.750	0.750	0.883	0.870	0.079414	0.988	506809	23445.7	100
g12(max.)	1	1.000	1.000	1.000	0	1.000	35258	1638.7	100
g13(min.)	0.05395	0.09208	0.94887	1.37260	1.635055	7.18982	514867	23574.0	100

Tabla V.50 Datos concentrados de las 30 corridas independientes del experimento 4 del EGA-DR.

Problema	óptimo	mejor	mediana	media	desv. est.	peor	num. evals. promedio	num. gen. promedio	% de factibles
g01(min.)	-15.000000	-15.00000	-14.99999	-14.99998	0.000038	-14.99979	439975	4624.2	100
g02(max.)	0.803619	0.795192	0.768581	0.760553	0.028757	0.684068	407262	4619.4	100
g03(max.)	1	0.983649	0.867706	0.833769	0.152387	0.383581	453926	4628.4	100
g04(min.)	-30665.539	-30665.536	-30665.477	-30665.419	0.167262	-30664.826	481072	4607.8	100
g05(min.)	5126.498	5127.071	5151.054	5283.392	190.67358	5686.455	157928	1545.0	93.33
g06(min.)	-6961.814	-6961.814	-6961.814	-6961.814	0	-6961.814	438220	3786.2	100
g07(min.)	24.306	24.559	26.458	26.740	1.425806	30.822	444029	4603.0	100
g08(max.)	0.095825	0.095825	0.095825	0.091380	0.016633	0.029144	17909	167.7	100
g09(min.)	680.63	680.659	680.944	681.097	0.411654	682.394	461489	4559.8	100
g10(min.)	7049.331	7232.937	7829.571	8167.090	1015.4926	11426.581	417669	4483.7	100
g11(min.)	0.750	0.750	0.784	0.805	0.057839	0.932	476624	4060.1	100
g12(max.)	1	1.000	1.000	1.000	0	1.000	10335	107.5	100
g13(min.)	0.05395	0.21089	0.86376	1.24310	1.605233	8.13697	502153	4618.1	100

Las tablas V.51 a V.54 muestran los datos concentrados de las 30 corridas independientes de cada una de las trece funciones optimizadas con el EAA.

Tabla V.51 Datos concentrados de las 30 corridas independientes del experimento 1 del EAA.

Problema	óptimo	mejor	mediana	media	desv. est.	peor	num. evals. promedio	num. gen. promedio	% de factibles
g01(min.)	-15.000000	-15.00000	-15.00000	-15.00000	0	-15.00000	92264	0.1	100
g02(max.)	0.803619	0.765352	0.676895	0.681926	0.052542	0.588115	461897	0	100
g03(max.)	1	0.997626	0.891670	0.866954	0.109395	0.548497	478413	8.4	100
g04(min.)	-30665.539	-30665.528	-30648.761	-30634.787	36.100718	-30532.113	491622	0	100
g05(min.)	5126.498	5126.484	5146.832	5214.577	185.75843	5976.824	338172	1.2	100
g06(min.)	-6961.814	-6961.814	-6961.813	-6961.813	0.001009	-6961.808	277575	3.3	100
g07(min.)	24.306	24.937	25.849	26.081	1.089143	29.110	395956	3.4	100
g08(max.)	0.095825	0.095825	0.095825	0.095825	0	0.095825	8445	1.0	100
g09(min.)	680.63	680.895	681.203	681.237	0.315588	682.482	415700	4.0	100
g10(min.)	7049.331	7050.647	7119.661	7193.793	208.04903	7873.170	390633	0.6	100
g11(min.)	0.750	0.750	0.750	0.750	0	0.750	334287	177.0	100
g12(max.)	1	1.000	1.000	1.000	0	1.000	74939	2.3	100
g13(min.)	0.05395	0.05394	0.05422	0.11456	0.119805	0.45427	474088	38.4	100

Tabla V.52 Datos concentrados de las 30 corridas independientes del experimento 2 del EAA.

Problema	óptimo	mejor	mediana	media	desv. est.	peor	num. evals. promedio	num. gen. promedio	% de factibles
g01(min.)	-15.000000	-14.99999	-14.99376	-14.98848	0.013832	-14.93683	222267	21.0	100
g02(max.)	0.803619	0.792724	0.752807	0.740380	0.040242	0.633033	458282	337.3	100
g03(max.)	1	0.999910	0.995646	0.992606	0.007328	0.966641	317537	415.6	100
g04(min.)	-30665.539	-30665.210	-30663.341	-30661.314	4.917058	-30641.030	400474	127.7	100
g05(min.)	5126.498	5126.492	5163.165	5227.256	131.75484	5587.401	261887	1464.9	86.67
g06(min.)	-6961.814	-6961.814	-6961.814	-6961.813	0.002112	-6961.803	307333	5606.9	100
g07(min.)	24.306	24.397	25.762	25.963	1.250864	28.713	435699	159.4	100
g08(max.)	0.095825	0.095825	0.095825	0.095825	0	0.095825	156172	980.5	100
g09(min.)	680.63	680.656	681.238	681.433	0.639486	683.572	287615	265.4	100
g10(min.)	7049.331	7049.317	7065.938	7079.860	58.99495	7385.453	375085	7.8	100
g11(min.)	0.750	0.750	0.750	0.750	0.000005	0.750	195465	2308.1	100
g12(max.)	1	1.000	1.000	1.000	0	1.000	19234	181.5	100
g13(min.)	0.05395	0.05458	0.18685	0.27864	0.218521	0.69521	353308	1709.9	100

Tabla V.53 Datos concentrados de las 30 corridas independientes del experimento 3 del EAA.

Problema	óptimo	mejor	mediana	media	desv. est.	peor	num. evals. promedio	num. gen. promedio	% de factibles
g01(min.)	-15.000000	-15.00000	-15.00000	-15.00000	0	-15.00000	146174	0	100
g02(max.)	0.803619	0.737288	0.692105	0.692063	0.022599	0.644083	445095	2.9	100
g03(max.)	1	0.959549	0.781088	0.770545	0.121728	0.357523	459477	11.7	100
g04(min.)	-30665.539	-30664.696	-30654.889	-30652.143	9.295055	-30624.708	404368	0.1	100
g05(min.)	5126.498	5126.528	5134.627	5247.070	236.43463	5968.906	376736	59.6	76.67
g06(min.)	-6961.814	-6961.206	-6955.420	-6950.347	11.64036	-6911.611	192984	11.4	100
g07(min.)	24.306	25.076	26.525	26.586	1.047646	28.642	426527	0.7	100
g08(max.)	0.095825	0.095825	0.095825	0.095825	0	0.095825	65569	27.7	100
g09(min.)	680.63	680.765	681.188	681.280	0.377975	682.319	394936	7.5	100
g10(min.)	7049.331	7059.269	7165.275	7221.024	186.81013	8033.412	452399	0	100
g11(min.)	0.750	0.750	0.750	0.750	0.000002	0.750	234377	86.0	100
g12(max.)	1	1.000	1.000	1.000	0	1.000	77788	23.7	100
g13(min.)	0.05395	0.05394	0.05398	0.05408	0.000264	0.05506	402379	48.4	100

Tabla V.54 Datos concentrados de las 30 corridas independientes del experimento 4 del EAA.

Problema	óptimo	mejor	mediana	media	desv. est.	peor	num. evals. promedio	num. gen. promedio	% de factibles
g01(min.)	-15.000000	-15.00000	-15.00000	-15.00000	0	-15.00000	61232	0	100
g02(max.)	0.803619	0.767753	0.699541	0.694139	0.035041	0.607750	483702	0.1	100
g03(max.)	1	0.992328	0.952458	0.938526	0.046097	0.815843	473966	3.0	100
g04(min.)	-30665.539	-30664.571	-30660.991	-30659.671	5.841573	-30636.136	477703	0	100
g05(min.)	5126.498	5126.485	5126.850	5162.439	129.81626	5761.968	326560	9.4	96.67
g06(min.)	-6961.814	-6961.812	-6961.748	-6961.584	0.36562	-6960.393	209580	8.4	100
g07(min.)	24.306	24.672	26.026	26.727	2.346079	36.975	435304	0.2	100
g08(max.)	0.095825	0.095825	0.095825	0.095825	0	0.095825	10117	4.3	100
g09(min.)	680.63	680.806	681.115	681.143	0.21232	681.849	451879	1.5	100
g10(min.)	7049.331	7052.574	7240.935	7406.532	455.41634	9198.724	449860	0	100
g11(min.)	0.750	0.750	0.750	0.750	0	0.750	389938	92.2	100
g12(max.)	1	1.000	1.000	1.000	0	1.000	34066	1.8	100
g13(min.)	0.05395	0.05394	0.05401	0.10084	0.105907	0.44026	444734	29.1	100

Capítulo VI
Análisis de Resultados y Conclusiones

VI.1 Una forma de calificar los resultados de los experimentos.

Para tener medidas cuantitativas de los resultados de los experimentos se pueden usar las tablas donde se reportan los valores del parámetro ξ para cada uno de ellos. Defínase una calificación, *c*, para cada experimento como lo muestra la ecuación VI.1.

$$c = \sum_i \xi_{mejor,i} + \sum_i \xi_{media,i} + \sum_i \xi_{peor,i} \qquad \text{(VI.1)}$$
$$i \in \{\text{g01},\ldots,\text{g13}\}$$

Aplicando esta fórmula a cada uno de los experimentos (ver tablas V.29, 30, 31, 32 para el EBGA; V.33, 34, 35, 36, para el EGA-DR; y V.37, 38, 39, 40 para el EAA) se obtienen las tablas VI.1, VI.2 y VI.3. En estas tablas se puede observar que tanto aporta cada uno de los tipos de parámetro ξ a la calificación de cada experimento. La máxima calificación posible es 39.

Tabla VI.1 Calificaciones de los experimentos del EBGA.

	ξ_{mejor}	ξ_{media}	ξ_{peor}	Calificación
Experimento 1	11.77	10.45	9.30	31.52
Experimento 2	11.81	10.91	9.94	32.67
Experimento 3	11.58	10.53	9.23	31.34
Experimento 4	11.47	10.52	9.39	31.37
Promedio	11.66	10.61	9.46	31.73

Tabla VI.2 Calificaciones de los experimentos del EGA-DR.

	ξ_{mejor}	ξ_{media}	ξ_{peor}	Calificación
Experimento 1	12.53	11.15	9.03	32.71
Experimento 2	12.06	10.75	8.33	31.15
Experimento 3	12.45	10.46	8.59	31.50
Experimento 4	12.19	11.45	9.65	33.30
Promedio	12.31	10.95	8.90	32.16

Tabla VI.3 Calificaciones de los experimentos del EAA.

	ξ_{mejor}	ξ_{media}	ξ_{peor}	Calificación
Experimento 1	12.92	12.08	10.98	35.98
Experimento 2	12.97	12.02	11.54	36.53
Experimento 3	12.84	12.49	11.71	37.05
Experimento 4	12.93	12.19	11.00	36.13
Promedio	12.92	12.20	11.31	36.42

Aplicando la fórmula VI.1 a los datos que se tienen (ver tablas V.41 y 42) de los métodos RY y KM se obtiene la tabla VI.4.

Tabla VI.4 Calificaciones de los algoritmos de RY y de KM.

	ξ_{mejor}	ξ_{media}	ξ_{peor}	Calificación
RY	13.00	12.69	11.85	37.53
KM	11.95	11.50	9.83	33.28

VI.2 Variaciones entre las calificaciones obtenidas por los experimentos para probar un algoritmo.

Obsérvense las quintas columnas de las tablas VI.1, 2 y 3. Nótese que no hay una gran diferencia entre las calificaciones obtenidas por cada experimento de un mismo algoritmo. Defínase una medida adimensional de la variación en las calificaciones obtenidas por *experimentos de un mismo algoritmo*, *v*, con la formúla VI.2.

$$v = \frac{\begin{pmatrix}\text{mayor calificación}\\ \text{del algoritmo}\end{pmatrix} - \begin{pmatrix}\text{menor calificación}\\ \text{del algoritmo}\end{pmatrix}}{\begin{pmatrix}\text{promedio de todas las calificaciones}\\ \text{del algoritmo}\end{pmatrix}} \qquad \text{(VI.2)}$$

Aplicando la ecuación VI.2 a las quintas columnas de las tablas VI.1,2 y 3 se obtienen los resultados mostrados en la tabla V.5.

Tabla VI.5 Variaciones de las calificaciones obtenidas por los experimentos de los algoritmos propuestos.

Algoritmo	Variación
EBGA	0.042
EGA-DR	0.067
EAA	0.029

Las variaciones mostradas en esta tabla van de un 2.9% a un 6.7%. La suposición de que los resultados obtenidos en diversos experimentos en los que se cambian los parámetros variables dentro de intervalos dados (ver secciones V.3.3.2, V.3.3.2.2, V.3.3.2.3 y V.3.3.2.4) no provocarían grandes diferencias en los resultados queda confirmada (por lo menos para los conjuntos de valores asignados a los parámetros constantes y variables que produjeron estos resultados).

VI.3 Cual fue el mejor de los algoritmos propuestos.

De los promedios de calificaciones reportados en las tablas VI.1 a VI.3 se puede observar que el EAA fue el mejor de los tres propuestos con una calificación de 36.42/39, le sigue el EGA-DR con una calificación de 32.16/39 y cercano a él quedó el EBGA con 31.73/39.

Analizando los componentes de estas calificaciones se puede observar lo siguiente:
El EAA superó a los otros dos algoritmos en las tres medidas que constituyen la calificación: obtuvo el mejor promedio de los mejores puntos, el mejor promedio de las medias y el mejor promedio de los peores puntos (compárense las últimas filas de las tablas VI.1 a VI.3).

El EGA-DR obtuvo una calificación ligeramente mejor que el EBGA, lo supera en la medida de los mejores puntos y en la medida de la media pero es superado por éste en la medida de los peores puntos encontrados.

VI.4 Cómo queda el desempeño de los algoritmos propuestos con respecto a los algoritmos de runarsson y Yao [RY00] y de Koziel y Michalewicz [KM99].

Según las tablas VI.1 a VI.4 el orden de los algoritmos poniendo primero al que mejores puntos encontró es: 1) el de RY, 2) el EAA, 3) el de KM, 4) el EGA-DR y 5) el EBGA.

Hay que hacer las siguientes observaciones:
La calificación obtenida por el algoritmo de KM puede mejorar un poco, ya que en [KM99] dicen los autores para el problema g05 que "no encontró valores de calidad" ("*did not provide quality results*") sin indicar si el algoritmo encontró por lo menos un punto en región factible. Si éste es el caso, el problema g05 podría contribuir con algunas décimas a la calificación del KM si los resultados no están demasiado lejanos al óptimo. De esta manera su calificación no quedaría tan cercana a la del EGA-DR (de hecho, aunque la calificación promedio del EGA-DR es ligeramente menor a la del KM, el experimento 4 del EGA-DR supera a éste último muy ligeramente). No obstante, no podría alcanzar al EAA, porque éste lo supera por más de tres puntos.

El número de evaluaciones que invierte el RY para encontrar los mejores puntos en general es menor que los números de evaluaciones que invierten los algoritmos propuestos (ver gráfica V.3 y tablas V.17 a V.19 y V.41). En la tabla VI.6 se muestran las relaciones entre los promedios de las evaluaciones para encontrar los mejores puntos usados por los algoritmos propuestos y el número de evaluaciones usado por el RY.

Tabla VI.6 Relación del número promedio de evaluaciones usadas por los algoritmos propuestos y el numero de evaluaciones usadas por el RY para encontrar los mejores puntos.

	EBGA/RY	EGA-DR/RY	EAA/RY
g01	2.78	2.32	0.88
g02	2.03	2.06	2.13
g03	1.81	2.15	1.89
g04	5.28	5.57	5.03
g05	4.67	1.49	3.56
g06	4.18	3.25	2.09
g07	3.10	3.16	2.96
g08	2.24	0.46	0.79
g09	4.02	4.10	3.48
g10	3.56	3.06	3.25
g11	36.43	41.14	25.31
g12	16.82	1.31	3.14
g13	6.10	7.07	6.00
Suma	93.01	77.13	60.50

En la tabla VI.6 se puede observar que son muy pocos los casos en los que los algoritmos propuestos usan menos evaluaciones que RY para encontrar los mejores valores. De los algoritmos aquí propuestos es el EAA el que menos evaluaciones requiere, le sigue el EGA y el que más evaluaciones necesita es el EBGA, como se puede observar en la última fila de la tabla.

El número de iteraciones que invierte el KM para encontrar los mejores puntos no está reportado en [KM99], de los datos que allí se suministran se puede calcular el número total de evaluaciones usado, que es de 1,400,000; el número total de evaluaciones usado por RY calculado a partir de los datos dados en [RY00] es de 350,000 y el usado aquí es de aproximadamente 500,000.

VI.5 Grado de dificultad que presentaron los problemas para su optimización por medio de los tres algoritmos propuestos.

Defínase el grado de dificultad, $d \in [0,1]$, en la solución del problema $l \in \{g01, ..., g13\}$ con la ecuación VI.3.

$$d_l = 1 - f_{f,l} \cdot \frac{\sum_i \sum_j \sum_k \xi_{i,j,k,l}}{4 \times 4 \times 3} \qquad \text{(VI.3)}$$

donde

$f_{f,l}$ es la fracción promedio total de puntos factibles del problema l, se define en la ecuación VI.4.

$\xi_{i,j,k,l}$ es el parámetro ξ del problema $l \in \{g01, \ldots, g13\}$ del experimento $k = 1, \ldots, 4$ del tipo $j \in \{\text{mejor, mediana, media, peor}\}$ resuelto con el algoritmo $i \in \{\text{EBGA, EGA-DR, EAA}\}$

$$f_{f,l} = \frac{\sum_i \sum_j \sum_k \varphi_{i,j,k,l}}{3 \times 4 \times 30} \qquad \text{(VI.4)}$$

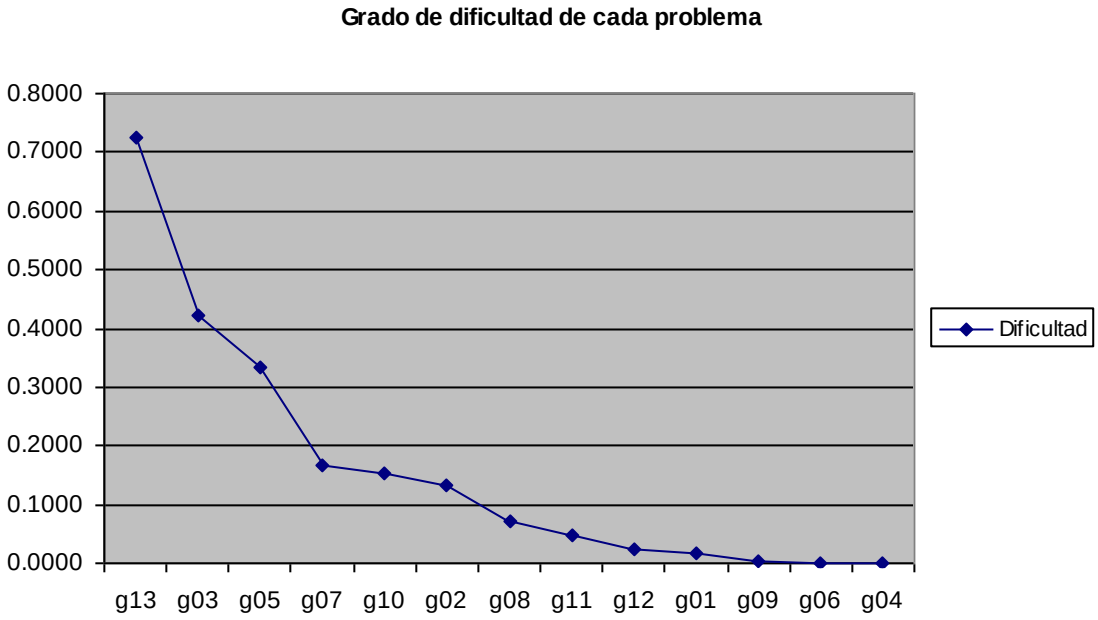
donde $\varphi_{i,j,k,l}$ vale 1 si la corrida $k = 1, \ldots, 30$ del experimento $j = 1, \ldots, 4$ del algoritmo $i \in \{\text{EBGA, EGA-DR, EAA}\}$ para optimizar el problema l da por resultado un punto en región factible, de lo contrario vale 0.

En la tabla VI.7 se muestran los resultados de aplicar la ecuación VI.3 a los datos de los trece problemas.

Tabla VI.7 Grado de dificultad experimentado por los tres algoritmos propuestos para resolver cada uno de los tres problemas. El parámetro *d* definido en la ecuación VI.3 toma valores entre 0 y 1; 0 representa la mínima dificultad y 1 la máxima. Los datos están ordenados de la mayor a la menor dificultad.

Problema	*d*
g13	0.7250
g03	0.4211
g05	0.3331
g07	0.1670
g10	0.1536
g02	0.1341
g08	0.0702
g11	0.0484
g12	0.0229
g01	0.0163
g09	0.0030
g06	0.0013
g04	0.0009

En la gráfica VI.1 se representan los puntos de la tabla VI.7.



Gráfica VI.1 Grado de dificultad en la solución con los tres algoritmos de los problemas de prueba.

Es interesante comparar el grado de dificultad con los tamaños relativos de la región factible reportados en la séptima columna de la tabla V.1. La tabla VI.8 reporta esa información ordenada de menor a mayor tamaño relativo de la región factible.

Tabla VI.8 Problemas de prueba ordenados según el tamaño relativo de su región factible.

Problema	Tamaño relativo de la región factible $((F \cap S)/S)$
g05	0
g13	0
g01	0.000002
g03	0.000002
g07	0.000002
g10	0.000003
g06	0.000063
g11	0.000102
g09	0.005261
g08	0.008677
g12	0.047771
g04	0.269744
g02	0.999973

Obsérvese que cinco de los seis primeros problemas de estas últimas dos tablas son los mismos. Estos son: g13, g03, g05, g07 y g10, listados según su grado de dificultad. Esto indica una fuerte relación entre el tamaño de la región factible y el grado de dificultad en la optimización del problema por parte de los tres algoritmos propuestos. Sin embargo éste no es el único factor que determina el grado de dificultad en la optimización. Esto se demuestra con

el dramático caso del problema g02, que siendo el que tiene la región factible con el mayor tamaño relativo (casi todo el dominio de su función objetivo pertenece a la región factible) quedó dentro de los seis problemas más difíciles de optimizar, esto es debido a que su función objetivo presenta un alto grado de multimodalidad y de dimensionalidad (20 variables, la mayor de los problemas de prueba aquí usados). Otro caso que demuestra que el tamaño relativo de la región factible no es el único factor que determina el grado de dificultad de la optimización de una función es el del problema g01, que teniendo una región factible cuyo tamaño relativo es el segundo más pequeño de los problemas de prueba usados, presenta el cuarto menor grado de dificultad, esto es debido a que la función es unimodal (pero a pesar de tener una dimensionalidad alta dentro de esta colección de problemas: 13 variables, la segunda de la colección). Esto coincide con [MDSS00] donde se indica que el grado de dificultad en la optimización de una función restringida depende de: el número de variables del problema, número de óptimos en el espacio de búsqueda (modalidad), número restricciones, conectividad de la región factible, tamaño relativo de la región factible y rugosidad de la función.

VI.6 Conclusiones.

En este trabajo se ha interpretado, en base a los estudios presentados por los investigadores de los sistemas inmunes, qué hace este sistema para identificar a los patógenos para su posterior destrucción. Con esto, aunado a la teoría y la práctica sobre los algoritmos genéticos, se obtuvieron algoritmos para resolver algunas instancias del problema de programación no lineal. Estos algoritmos se han utilizado con éxito para la optimización numérica de funciones restringidas.

El principal aporte científico de este trabajo es el aplicar las ideas sobre la evolución genética de las bibliotecas productoras de receptores (anticuerpos) y la evolución somática, en la solución de algunas instancias del problema de programación no lineal. La evolución genética de bibliotecas junto con la evolución somática de receptores ya han sido usadas en sistemas inmunes artificiales en el área de reconocimiento de patrones y estudios de la memoria asociativa del sistema inmune (Perelson, Hightower & Forrest, 1996 [PHF96]; Hightower, Forrest & Perelson 1996 [HFP96]; Oprea & Forrest, 1998 [OF98]; Oprea & Forrest, 1999 [OF99]) pero no en la de manejo de restricciones en la optimización numérica. Se ha tomado directamente del área de los sistemas inmunes naturales el uso de la *edición de receptores* (George & Gray, 1999, [GG99]) durante la evolución somática de los receptores (hipermutación somática) para obtener mejores resultados.

Otro aporte científico de este trabajo es la métrica empleada para evaluar el desempeño de los algoritmos evolutivos propuestos (ver secciones V.3.3, VI.1 y VI.2).

Los resultados obtenidos han sido alentadores. Los algoritmos aquí propuestos tienen una gran proclividad hacia la región factible: en un alto porcentaje de las corridas encuentran puntos en ella aunque su tamaño relativo sea pequeño (igual o menor a 10^{-5}). Además, en la mayoría de los casos analizados, los algoritmos encuentran el mejor punto reportado o un punto con un valor de aptitud muy cercano a él.

Hay que recalcar que los algoritmos aquí mostrados toman ideas de los sistemas inmunes naturales y artificiales, pero al diseñarlos no se pretendió ni simular o imitar al sistema inmune natural o el apegarse estrictamente a las puntos generales característicos de alguno de sus mecanismos, así en el EAAMRE, se propone después de la evolución somática (hipermutación

somática) de los linfocitos B el cruzarlos entre sí, esto no ocurre en los sistemas inmunes. Simplemente se adoptó esta estrategia porque el algoritmo así funcionaba mejor.

Cumplimiento de los objetivos. Los objetivos de este trabajo de investigación han sido cumplidos, se tomaron abstracciones de ideas de los sistemas inmunes naturales y basados en algunos trabajos de los investigadores del área de los sistemas inmunes artificiales, se propusieron algoritmos genéticos que, haciendo uso de estas ideas junto con las ya tradicionales en el área de algoritmos genéticos y de manejo de restricciones en optimización numérica con algoritmos evolutivos, dieron por resultado algoritmos capaces de optimizar funciones numéricas, la mayoría de ellas restringidas, encontrando puntos en sus regiones factibles un alto porcentaje de veces y encontrando buenas soluciones en la región factible.

Trabajo futuro. Los puntos en el espacio de búsqueda se representaron en esta investigación como cadenas binarias. Es conveniente el usar codificación con números reales para probar estos algoritmos. Por supuesto, para esta nueva codificación deberán modificarse los operadores genéticos.

Sería interesante probar un enfoque con sistemas distribuidos. En la presente investigación se han usado los algoritmos genéticos que con su *paralelismo implícito* ([Hol75], [Hol92]) hacen la búsqueda del óptimo. Se puede probar con sistemas basados en agentes el realiza esa búsqueda. Estos agentes representarían los efectores del sistema inmune (linfocitos B y T, anticuerpos, etc). Esta sería una búsqueda con un *paralelismo explícito*. Se podría usar una teoría como la propuesta por Segel y colaboradores (ver en el punto A.1 en el capítulo III en la parte *El Modelo de la Retroalimentación Difusa y de la Red de Información Difusa*) para hacer funcionar este sistema distribuido.

Apéndice A.

Colección de funciones utilizadas para probar los algoritmos propuestos, tomadas de la presentadas por Runarsson y Yao en [RY00] quienes a su vez las tomaron de [KM99] y [Mic95].

Problema g01

Minimizar $f(\boldsymbol{x}) = 5x_1 + 5x_2 + 5x_3 + 5x_4 - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=1}^{13} x_i$

sujeta a :

1) $2x_1 + 2x_2 + x_{10} + x_{11} \le 10$,
2) $-8x_1 + x_{10} \le 0$,
3) $-2x_4 - x_5 + x_{10'} \le 0$,
4) $2x_1 + 2x_3 + x_{10} + x_{12} \le 10$,
5) $-8x_2 + x_{11} \le 0$,
6) $-2x_6 - x_7 + x_{11} \le 0$,
7) $2x_2 + 2x_3 + x_{11} + x_{12} \le 10$,
8) $-8x_3 + x_{12} \le 0$,
9) $-2x_8 - x_9 + x_{12} \le 10$

los límites de las variables son:
$0 \le x_i \le 1$, $i = 1, \ldots, 9$,
$0 \le x_i \le 100$, $i = 10, 11, 12$,
$0 \le x_{13} \le 1$
óptimo: $f(\boldsymbol{x}) = -15$, $\boldsymbol{x} = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$

Problema g02

Maximizar $f(\boldsymbol{x}) = \left| \dfrac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i\, x_i^2}} \right|$

sujeta a:

$\prod_{i=1}^{n} x_i \ge 0.75$,

$\sum_{i=1}^{n} x_i \le 7.5\, n$

Los límites de las variables son:
$0 \le x_i \le 10$, i=1, . . . , n.

donde $n = 20$.
óptimo: $f(\boldsymbol{x}) = 0.803619$

Problema g03

Maximizar $f(\boldsymbol{x}) = (\sqrt{n})^n \cdot \prod_{i=1}^{n} x_i$

sujeta a :

$\sum_{i=1}^{n} x_i^{\,2} = 1$
Las variables son:
$0 \le x_i \le 1$ for $i = 1, \ldots, n$
donde $n = 10$
óptimo: $f(\boldsymbol{x}) = 1$

$$(x_1, \ldots, x_n) = \left(\frac{1}{\sqrt{n}}, \ldots, \frac{1}{\sqrt{n}} \right)$$

Problema g04
Minimizar $f(\boldsymbol{x}) = 5.3578547\, x_3^2 + 0.8356891\, x_1 x_5 + 37.293239\, x_1 - 40792.141$
subject to
1) $0 \le 85.334407 + 0.0056858\, x_2\, x_5 + 0.0006262 \quad x_1\, x_4 - 0.0022053\, x_3\, x_5$
2) $85.334407 + 0.0056858\, x_2\, x_5 + 0.0006262\, x_1\, x_4 - 0.0022053\, x_3\, x_5 \le 92$
3) $90 \le 80.51249 + 0.0071317\, x_2\, x_5 + 0.0029955\, x_1\, x_2 + 0.0021813\, x_3^2$
4) $80.51249 + 0.0071317\, x_2\, x_5 + 0.0029955\, x_1\, x_2 + 0.0021813\, x_3^2 \le 110$
5) $20 \le 9.300961 + 0.0047026\, x_3\, x_5 + 0.0012547\, x_1\, x_3 + 0.0019085\, x_3\, x_4$
6) $9.300961 + 0.0047026\, x_3\, x_5 + 0.0012547\, x_1\, x_3 + 0.0019085\, x_3\, x_4 \le 25$

los límites de las variables son:
$78 \le x_1 \le 102$,
$33 \le x_2 \le 45$,
$27 \le x_i \le 45$, $i = 3, 4, 5$

óptimo: $f(\boldsymbol{x}) = -30665.539$, $\boldsymbol{x} = (78,\ 33,\ 29.995256025682,\ 45,\ 36.775812905788)$

Problema g05
Minimizar $f(\boldsymbol{x}) = 3\, x_1 + 0.000001\, x_1^3 + 2\, x_2 + 0.000002/3\, x_2^3$
sujeto a
1) $x_4 - x_3 + 0.55 \ge 0$
2) $x_3 - x_4 + 0.55 \ge 0$
3) $1000\ \mathrm{sen}(-x_3 - 0.25) + 1000\ \mathrm{sen}(-x_4 - 0.25) + 894.8 - x_1 = 0$,
4) $1000\ \mathrm{sen}(x_3 - 0.25) + 1000\ \mathrm{sen}(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$,
5) $1000\ \mathrm{sen}(x_4 - 0.25) + 1000\ \mathrm{sen}(x_4 - x_3 - 0.25) + 1294.8 = 0$,

los límites de las variables son:
$0 \le x_i \le 1200$, $i = 1,2$
$-0.55 \le x_i \le 0.55$, $i = 3, 4$
óptimo $f(\boldsymbol{x}) = 5126.4981$, $\boldsymbol{x} = (679.9453, 1026.067, 0.1188764, -0.3962336)$

Problema g06
Minimizar $f(\boldsymbol{x}) = (x_1 - 10)^3 + (x_2 - 20)^3$
sujeta a:
1) $(x_1 - 5)^2 + (x_2 - 5)^2 - 100 \ge 0$,

2) $-(x_1 - 6)^2 - (x_2 - 5)^2 + 82.81 \geq 0$
los límites de las variables son:
$13 \leq x_1 \leq 100$
$0 \leq x_2 \leq 100$
óptimo: $f(\boldsymbol{x})$ = -6961.81388, $\boldsymbol{x}$ = (14.095, 0.84296)

Problema g07
Minimizar
$$f(\boldsymbol{x}) = x_1^2 + x_2^2 + x_1 x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 +$$
$$7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$$

sujeta a:
1) $105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 \geq 0$,
2) $-10x_1 + 8x_2 + 17x_7 - 2x_8 \geq 0$,
3) $8x_1 - 2x_2 - 5x_9 + 2x_{10} + 12 \geq 0$,
4) $3x_1 - 6x_2 - 12(x_9 - 8)^2 + 7x_{10} \geq 10$
5) $-3(x_1 - 2)^2 - 4(x_2 - 3)^2 - 2x_3^2 + 7x_4 + 120 \geq 0$,
6) $-x_1^2 - 2(x_2 - 2)^2 + 2x_1 x_2 - 14x_5 + 6x_6 \geq 0$
7) $-5x_1^2 - 8x_2 - (x_3 - 6)^2 + 2x_4 + 40 \geq 0$,
8) $-0.5(x_1 - 8)^2 - 2(x_2 - 4)^2 - 3x_5^2 + x_6 + 30 \geq 0$

los límites de las variables son:
$-10.0 \leq x_i \leq 10.0$, $i = 1, \ldots, 10$.
óptimo: $f(\boldsymbol{x})$ = 24.3062091, $\boldsymbol{x}$ = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927)

Problema g08
Maximizar $f(\boldsymbol{x}) = \dfrac{\text{sen}^3(2\pi x_1)\,\text{sen}(2\pi x_2)}{x_1^3(x_1 + x_2)}$,
sujeta a :
1) $x_1^2 - x_2 + 1 \leq 0$,
2) $1 - x_1 + (x_2 - 4)^2 \leq 0$
los límites de las variables son:
$0 \leq x_1 \leq 10$
$0 \leq x_2 \leq 10$
óptimo: $f(\boldsymbol{x})$ = 0.0958215, $\boldsymbol{x}$ = (1.2279713, 4.2453733)

Problema g09
Minimizar $f(\boldsymbol{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6 x_7 - 10x_6 - 8x_7$
sujeta a:
1) $127 - 2x_1^2 - 3x_2^4 - x_3 - 4x_4^2 - 5x_5 \geq 0$,
2) $282 - 7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 \geq 0$,
3) $196 - 23x_1 - x_2^2 - 6x_6^2 + 8x_7 \geq 0$,
4) $-4x_1^2 - x_2^2 + 3x_1 x_2 - 2x_3^2 - 5x_6 + 11x_7 \geq 0$

los límites de las variables son:

$-10 \le x_i \le 10.0, \text{ i} = 1,\ldots,7$

óptimo: $f(\boldsymbol{x})$ = 680.6300573, $\boldsymbol{x}$ = (2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227)

Problema g10

Minimizar $f(\boldsymbol{x}) = x_1 + x_2 + x_3$

sujeta a:

1) $1 - 0.0025(x_4 + x_6) \ge 0$,
2) $1 - 0.0025(x_5 + x_7 - x_4) \ge 0$,
3) $1 - 0.01(x_8 - x_5) \ge 0$,
4) $x_1x_6 - 833.33252\, x_4 - 100x_1 + 83333.333 \ge 0$,
5) $x_2x_7 - 1250x_5 - x_2x_4 + 1250x_4 \ge 0$,
6) $x_3x_8 - 1250000 - x_3x_5 + 2500x_5 \ge 0$.

los límites de las variables son:

$100 \le x_1 \le 10000$,

$1000 \le x_i \le 10000$, i = 2, 3,

$10 \le x_i \le 1000$, i = 4,. . .,8

óptimizar: $f(\boldsymbol{x})$ = 7049.3307, $\boldsymbol{x}$ = (579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.9799, 286.4162, 395.5979)

Problema g11

Minimizar $f(\boldsymbol{x}) = x_1^2 + (x_2 - 1)^2$,

sujeta a :

$x_2 - x_1^2 = 0$

los límites de las variables son:

$-1 \le x_i \le 1$, i = 1,2.

óptimos: $f(\boldsymbol{x})$ = 0.75000455, $\boldsymbol{x}$ = (0.70711, 0.5) y $f(\boldsymbol{x})$ = 0.75000455, $\boldsymbol{x}$ = (-0.70711, 0.5)

Problema g12

Maximizar $f(\boldsymbol{x}) = (100 - (x_1 - 5)^2 + (x_2 - 5)^2 - (x_3 - 5)^2)/100$

sujeta a:

$(x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \le 0$

donde

p, q, r = 1, . . . , 9.

Los límites de las variables son:

$0 \le x_i \le 10$, i = 1, 2, 3.

óptimo: $f(\boldsymbol{x})$ = 1, $\boldsymbol{x}$ = (5, 5, 5)

Problema g13

Minimizar $f(\boldsymbol{x}) = e^{x_1x_2x_3x_4x_5}$

sujeto a:

$$x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$$

$$x_2 x_3 - 5 x_4 x_5 = 0$$

$$x_1^3 + x_2^3 + 1 = 0$$

los límites de las variables son:

$-2.3 \leq x_i \leq 2.3$, $i = 1, 2$

$-3.2 \leq x_i \leq 3.2$, $i = 3, 4, 5$.

óptimo : $f(\boldsymbol{x}) = 0.0539498$, $\boldsymbol{x}$ = (-1.717143, 1.595709, 1.827247, -0.7636413, -0.763645)

Apéndice B.

En este apéndice se muestra corridas de prueba para cada uno de los tres algoritmos propuestos en este trabajo (ver capítulo IV).

B.I Corridas del EBGA.

B.I.1 Datos de los genomas de los anticuerpos:

Tabla B.I.1 Número de bits de la cadena de parámetros automodificables.

Subcadena de parámetros automodificables	
Descripción del campo	Longitud en bits
Bandera de procedencia	1
n_o, número de hijos por individuo	1
p_m, probabilidad de mutación,	10
p_c, probabilidad de cruzamiento,	10
f_η (Número de evaluaciones cuando el RMHC es invocado es igual a: f_η × tamaño de la población de padres)	8

Tabla B.I.2 Número de bits de las subcadenas que representan los puntos.

Subcadena para representar un punto en el espacio de búsqueda	
Descripción del campo	Longitud en bits
Cada uno de los componentes $x_i \in \boldsymbol{x} \in R^n$	32

Tabla B.I.3 Tamaños de los genomas de los anticuerpos en cada uno de los problemas.

Problema	Número de variables	Tamaño del genoma en bits
g01	13	416
g02	20	640
g03	10	320
g04	5	160
g05	4	128
g06	2	64
g07	10	320
g08	2	64
g09	7	224
g10	8	256
g11	2	64
g12	3	96
g13	5	160

B.I.2 Datos del experimento 2 del EBGA, con él se obtuvieron los mejores puntos.

Para el experimento 2 del EBGA, como para los otros tres experimentos de este algoritmo, se usaron los datos de la subsección B.I.1. Los demás datos para este experimento se muestran en las tablas B.I.4 y B.I.6. La tabla B.I.5 simplemente muestra cómo quedan los genomas de las bibliotecas con los datos alimentados.

Tabla B.I.4 Particionamiento de las bibliotecas

Datos de segmentación de las bibliotecas	
n_p, número de partes por biblioteca	4
n_v, número de variaciones por parte	2

Tabla B.I.5 Tamaños de los genomas de las bibliotecas del EBGA para cada uno de los problemas.

Problema	Número de variables	Tamaño de subcadena de partes de anticuerpo	Tamaño de subcadena de parámetros automodificables	Tamaño total del genoma en bits
g01	13	832	30	862
g02	20	1280	30	1310
g03	10	640	30	670
g04	5	320	30	350
g05	4	256	30	286
g06	2	128	30	158
g07	10	640	30	670
g08	2	128	30	158
g09	7	448	30	478
g10	8	512	30	542
g11	2	128	30	158
g12	3	192	30	222
g13	5	320	30	350

Tabla B.I.6 Valores de parámetros.

Parámetros del GA	
μ , tamaño de la población de padres	63 individuos
p_{m} , probabilidad de mutación	[0.0001, 0.0676]
p_c , probabilidad de cruzamiento	[0.8544, 1]
n_o, número de hijos por individuo	{1, 2}
ParáMeros RMHC	
n_λ , mínima probabilidad de que sea invocado el RMHC	1/8
n_μ, máxima probabilidad de que sea invocado el RMHC	½
f_η (Número de evaluaciones cuando el RMHC es invocado es igual a: f_η × tamaño de la población de padres)	[0.9283, 1]

Número de generaciones = 4672

B.I.3 Resultados de las corridas del experimento 2 del EBGA

Tabla B.I.7(a) Los mejores puntos encontrados para cada uno de los problemas.

	Problema g01	Problema g02	Problema g03	Problema g04	Problema g05
$f(\boldsymbol{x})$	-14.99969413257883	0.7733556302302179	0.8040649172340772	-30665.24760648832	5127.604664562975
x_1	0.9999996172264217	3.10720223074481	0.2845294248043861	78.00000015646219	665.3869932203989
x_2	0.999999989522621	3.134347736633929	0.2468094006755411	33.00054383743567	1041.665843185425
x_3	0.9999980900902297	3.072358796622688	0.329281411443204	29.9971143936266	0.1292778588550347
x_4	0.9999920772388559	3.060350749422878	0.3944300192860956	44.99999923724681	-0.3913075172438537
x_5	0.9999378786887829	3.125821743888273	0.3532451804152795	36.77111416863536	
x_6	0.9999935445375725	3.00605772365957	0.2763095487086823		
x_7	0.9999956963583817	3.042748845890805	0.288470471810659		
x_8	0.9999919368419777	2.941422970719036	0.366566121896395		
x_9	0.9999944099690752	2.826951426180767	0.2600520565780001		
x_{10}	2.999909641919637	0.4351442913606633	0.3284573113379202		
x_{11}	2.999951760983083	0.2989842510546987			
x_{12}	2.999972902005532	0.3569194069963226			
x_{13}	0.9999974905513221	1.271756051870006			
x_{14}		0.2498584357672041			
x_{15}		0.4636051669864927			
x_{16}		0.3600379709061324			
x_{17}		0.2819603472673242			
x_{18}		0.3801609157538416			
x_{19}		0.4112396390203479			
x_{20}		0.3186700610254589			

Tabla B.I.7(b)

	Problema g06	Problema g07	Problema g08	Problema g09	Problema g10
$f(\boldsymbol{x})$	-6961.797912882338	24.51820737123418	0.09582504141806104	680.7481459950776	7084.316381529047
x_1	14.09500658351346	2.214276621168078	1.227971351901489	2.318352566174779	424.0218930700845
x_2	0.8429749870772881	2.287079853072548	4.245373367854714	1.941337686018398	1622.55379036594
x_3		8.712789774572661		-0.6106421143306983	5037.740698093023
x_4		5.091682489749902		4.393681994265337	166.963975002748
x_5		0.9119961901828638		-0.5936060172025128	298.5063368148418
x_6		1.284909651448231		1.095985255924981	232.1601391309314
x_7		1.391741026051282		1.635648326863453	268.3067273437759
x_8		9.875739263341702			398.4989696597911
x_9		8.308788393230362			
x_{10}		8.203283426864838			

Tabla B.I.7(c)

	Problema g11	Problema g12	Problema g13
$f(\boldsymbol{x})$	0.7499099294961556	1	0.8886871821001843
x_1	0.709260935827452	5.000000033760443	-0.9088667986935161
x_2	0.5031510748674979	5.000000010477379	-0.6301619081595359
x_3		5.000000008149073	1.277833346574994
x_4			0.06033951269004967
x_5			-2.672331558045077

B.II Corridas del EGA-DR.

B.II.1 Datos de los genomas de los anticuerpos del EGA-DR.

Tabla B.II.1 Número de bits de la cadena de parámetros automodificables.

Subcadena de parámetros automodificables	
Descripción del campo	Longitud en bits
Bandera de procedencia	1
n_o, número de hijos por individuo	1
p_m, probabilidad de mutación,	10
p_c, probabilidad de cruzamiento,	10
f_η (Número de evaluaciones cuando el RMHC es invocado es igual a: f_η × tamaño de la población de padres)	8

Tabla B.II.2 Número de bits de las subcadenas que representan los puntos.

Subcadena para representar un punto en el espacio de búsqueda	
Cada uno de los componentes $x_i \in \boldsymbol{x} \in R^n$	32

Tabla B.II.3 Tamaños de los genomas de los anticuerpos en cada uno de los problemas.

Problema	Número de variables	Tamaño en bits de la subcadena que representa un punto en S (l_v)	Tamaño en bits de la subcadena de parámetros automodificables (l_{pa})	Tamaño total del genoma en bits
g01	13	416	30	446
g02	20	640	30	670
g03	10	320	30	350
g04	5	160	30	190
g05	4	128	30	158
g06	2	64	30	94
g07	10	320	30	350
g08	2	64	30	94
g09	7	224	30	254
g10	8	256	30	286
g11	2	64	30	94
g12	3	96	30	126
g13	5	160	30	190

B.II.2 Datos del experimento 1 del EGA-DR, con él se obtuvieron los mejores resultados.

Los datos sobre el genoma de los anticuerpos para el experimento 4 del EGA-DR, y para los otros tres experimentos, se muestran en la sección B.II.1. En la tabla B.2.3 se muestran los datos para los otros parámetros del experimento 4.

Tabla B.II.4 Valores de parámetros.

Parámetros del GA	
μ , tamaño de la población de padres	66
p_{m} , probabilidad de mutación	[0.0001, 0.0634]
p_c , probabilidad de cruzamiento	[0.9078, 1]
n_o, número de hijos por individuo	{1, 2}
Parámeros RMHC	
n_λ , mínima probabilidad de que sea invocado el RMHC	1/8
n_μ, máxima probabilidad de que sea invocado el RMHC	1/2
n_b, número de bits máximo que se cambian de una sola vez	$\lfloor 0.0166 \times l \rfloor + 1$ l es la longitud del genoma del anticuerpo
f_η (Número de evaluaciones cuando el RMHC es invocado es igual a: f_η × tamaño de la población de padres)	[0.6393, 1]
Número de generaciones	
4,644	

B.II.3 Resultados de las corridas del experimento 1 del EGA-DR

Tablas B.II.5(a) Los mejores puntos encontrados para cada uno de los problemas por el EGA-DR.

	Problema g01	Problema g02	Problema g03	Problema g04	Problema g05
$f(\boldsymbol{x})$	-14.99999994062819	0.8025795970528886	0.9958393483983592	-30665.50690267668	5128.088773284258
x_1	1	3.150768783677083	0.3147534956957105	78.000000346452	697.4325865268318
x_2	1	3.13025019670144	0.3075058644887772	33.0002350537107	1007.443384129424
x_3	1	3.106698103506746	0.3121249466929876	29.99545826786092	0.1064344843282445
x	1	3.046609967259367	0.3161778113143933	44.99999744351953	-0.4021550479000796
x_5	1	3.032562807908413	0.3287468860691289	36.77530266432448	
x_6	1	2.995193931971489	0.3133439028899521		
x_7	1	2.9579471431109	0.321905718958449		
x_8	1	2.916495775085989	0.320343121960839		
x_9	1	0.4775987380364907	0.3207162477357118		
x_{10}	2.999999980209395	0.5083418661980754	0.3060254131690658		
x_{11}	2.999999980209395	0.4782789224941001			
x_{12}	2.999999980209395	0.4980507843424684			
x_{13}	1	0.4542447511233028			
x_{14}		0.4934197758541954			
x_{15}		0.4455633136549879			
x_{16}		0.4882848869283415			
x_{17}		0.3900908563262995			
x_{18}		0.4574909644335254			
x_{19}		0.4738164135426787			
x_{20}		0.4299736000667265			

Tablas B.II.5(b)

	Problema g06	Problema g07	Problema g08	Problema g09	Problema g10
$f(\boldsymbol{x})$	-6961.8136450595	24.86052226438305	0.09582504141806106	680.6603552419795	7233.481978469874
x_1	14.09500010150834	2.122253322070989	1.227971354229793	2.303614605288863	297.4713984405323
x_2	0.8429609939556012	2.471853977179121	4.245373367854714	1.952883622598109	1000.000067055225
x_3		8.770663165201121		-0.4805210210570416	5936.010512974116
x_4		5.001687261974832		4.365557933777002	152.6090088353048
x_5		1.007443259239067		-0.5998625630978175	262.5596955494396
x_6		1.42694755723396		1.041413576118977	247.3803566739383
x_7		1.290297091773315		1.564828327755637	290.0483571202607
x_8		9.780289749563739			362.5596546806767
x_9		8.189902496102709			
x_{10}		8.457597018791734			

Tablas B.II.5(c)

	Problema g11	Problema g12	Problema g13
$f(\boldsymbol{x})$	0.7499000466266511	1	0.09217054075501517
x_1	-0.7071885754603866	5.000000052386895	-1.485154360948399
x_2	0.5002156811533998	5.000000043073669	1.315375658733624
x_3		5.000000001164153	-2.153799707943993
x_4			0.5295573720078819
x_5			-1.070008974771483

B.III Corridas del EAA.

B.III.1 Datos de los genomas de los anticuerpos.

Tabla B.III.1 Número de bits de la cadena de parámetros automodificables.

Subcadena de parámetros automodificables	
Descripción del campo	Longitud en bits
n_o, número de hijos por individuo	1
p_m, probabilidad de mutación,	10
p_c, probabilidad de cruzamiento,	10

Tabla B.III.2 Número de bits de las subcadenas que representan los puntos.

Subcadena para representar un punto en el espacio de búsqueda	
Descripción del campo	Longitud en bits
Cada uno de los componentes $x_i \in \boldsymbol{x} \in R^n$	32

Tabla B.III.3 Tamaños de los genomas de los anticuerpos en cada uno de los problemas.

Problema	Número de variables	Tamaño en bits de la subcadena que representa un punto en S (l_v)	Tamaño en bits de la subcadena de párametros automo-dificables (l_{pa})	Tamaño del genoma en bits
g01	13	416	21	437
g02	20	640	21	661
g03	10	320	21	341
g04	5	160	21	181
g05	4	128	21	149
g06	2	64	21	85
g07	10	320	21	341
g08	2	64	21	85
g09	7	224	21	245
g10	8	256	21	277
g11	2	64	21	85
g12	3	96	21	117
g13	5	160	21	181

B.III.2 Datos del experimento 2 del EAA, con él se obtuvieron los mejores puntos.
Los datos sobre el genoma del experimento 2, así como de los otros tres experimentos, se muestran en la sección B.III.1.

Tabla B.III.4 Valores de parámetros.

Parámetros del GA	
μ , tamaño de la población de padres	3 individuos
p_{m} , probabilidad de mutación	[0.0001, 0.0484]
p_c , probabilidad de cruzamiento	[0.7834, 1]
n_o, número de hijos por individuo	{1, 2}
Parámeros RMQHC	
T_{max}, máxima probabilidad de saltar a un punto peor de aquél donde se está	0.3159
Parámetros para intercalar una operación de cruzamiento entre operaciones de mutación puntual	
f_{nea} (ver IV.2.7)	0.1413
$T_{saltos\ buenos}$ (ver IV.2.7)	0.3360

B.III.3 Resultados de las corridas del experimento 2 del EAA

Tablas B.III.5(a) Los mejores puntos encontrados para cada uno de los problemas por el EAA.

	Problema g01	Problema g02	Problema g03	Problema g04	Problema g05
$f(\boldsymbol{x})$	-14.99999390402843	0.7927240726744539	0.999909657117535	-30665.20965788688	5126.491688855381
x_1	1	6.214870853865256	0.3159244946008372	78.00000587292016	680.6419695449625
x_2	0.9999999378342181	3.093726344661258	0.315569828105059	33.00156410504169	1025.320659537176
x_3	0.9999997459817677	3.090356002349955	0.3133784426174542	29.99734284658854	0.1183788521607358
x_4	0.9999999988358468	3.032484204283097	0.3159730353662681	44.99994164519011	-0.396469601976328
x_5	0.9999982882291074	2.993158251744033	0.3213201140336041	36.77056380868204	
x_6	0.9999999995343387	2.999268957180732	0.3173012443159943		
x_7	0.9999999983701855	2.910087847362758	0.3160004469836132		
x_8	0.9999999683350325	2.915220915552978	0.3177458458388564		
x_9	0.999999996740371	0.4852055107441743	0.3161274120947642		
x_{10}	2.999997884733602	0.4525881447951747	0.3129849306570797		
x_{11}	2.999999887077138	0.4052744550642731			
x_{12}	2.999999561114237	0.5282579386905436			
x_{13}	0.9999999066349119	0.4768481129959338			
x_{14}		0.4230958666706215			
x_{15}		0.4826507648645553			
x_{16}		0.3978401446709969			
x_{17}		0.3992572939952969			
x_{18}		0.4138472211579436			
x_{19}		0.4016614799391621			
x_{20}		0.4521475197868765			

Tablas B.III.5(b)

	Problema g06	Problema g07	Problema g08	Problema g09	Problema g10
$f(\boldsymbol{x})$	-6961.813828574372	24.39728427771874	0.09582504141806104	680.6563887584439	7049.317420238935
x_1	14.09500002048328	2.171848395879345	1.227971351901489	2.35104729709007	583.6858164015426
x_2	0.8429608309741506	2.363266309807837	4.245373365526407	1.949361868190896	1353.798670310946
x_3		8.753205290705246		-0.3814981995107369	5111.832933526447
x_4		5.035451356096996		4.360739321997562	182.3809007048562
x_5		0.9257982417302668		-0.6116692932815452	295.5266863469795
x_6		1.279485205020636		1.030864671578366	217.6157180214337
x_7		1.324170145980122		1.600663478393262	286.8526667232748
x_8		9.829821069685234			395.5266848941163
x_9		8.283668700206016			
x_{10}		8.385208341848386			

Tablas B.III.5(c)

	Problema g11	Problema g12	Problema g13
$f(\boldsymbol{x})$	0.7499000154247282	1	0.05457668204502379
x_1	0.7070855586107554	5.000000054715201	-1.719263583752155
x_2	0.5000699766678899	5.000000001164153	1.598160650743674
x_3		5.000000010477379	-1.819701839662088
x_4			0.8274029472906618
x_5			-0.7029693538097127

Apéndice C
Otros Resultados Obtenidos

Solución de un sistema de ecuaciones algebraicas no lineales

Cervantes utilizó parte del software desarrollado en el presente trabajo, para resolver un problema que se le presentó durante su trabajo de investigación en la maestría del Posgrado en Ciencias e Ingeniería de la Computación de la UNAM [Cer04]. Dicho problema involucra el encontrar algunas de las soluciones de un sistema de 64 ecuaciones algebraicas no lineales (polinomios cúbicos) con 16 incógnitas. El software que utilizó le fue proporcionado por el autor de este trabajo de investigación doctoral. Dicho software corre el algoritmo EGA-DR (descrito en este documento y en [HK03]).

El problema de la solución de un sistema de ecuaciones algebraicas no lineales se puede plantear de la siguiente manera:

Encontrar $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$,
tal que
$h_1(\boldsymbol{x}) = 0$
$h_2(\boldsymbol{x}) = 0$
…
$h_m(\boldsymbol{x}) = 0$
donde $m \geq n$

Este problema se puede plantear como un problema de minimización, de la siguiente manera:
Minimizar *tolerancia*
Sujeta a :
$|h_1(\boldsymbol{x})| - tolerancia \leq 0$
$|h_2(\boldsymbol{x})| - tolerancia \leq 0$
…
$|h_n(\boldsymbol{x})| - tolerancia \leq 0$

Entre menor valor de *tolerancia* se encuentre, más precisa será la solución del sistema de ecuaciones. Esta fue la estrategia utilizada para resolver ese sistema de ecuaciones utilizando el EGA-DR. Cervantes [Cer04] obtuvo varias soluciones para ese sistema con precisión satisfactoria. Sus resultados se encuentran reportados en su tesis de maestría ya disponible en el sistemas de bibliotecas de la UNAM.

Diseño de un algoritmo evolutivo inspirado en la maduración de la afinidad

Villalobos [Vill04] ha diseñado un algoritmo evolutivo para la optimización numérica de funciones restringidas durante su trabajo de investigación de Maestría en Ingeniería con Especialización en Informática, dirigido por el autor de este trabajo de investigación doctoral. Dicho algoritmo partió del EAA con la rutina MAER propuesto en este trabajo de investigación doctoral. El algoritmo propuesto por Villalobos [Vill04], utiliza números reales para la codificación y ha sido probado con la misma colección de funciones con la que han sido probados los algoritmos propuestos en este trabajo de investigación doctoral.. Ha usado 350,000 evaluaciones para cada función y ha obtenido resultados comparables a los de

Runarsson y Yao [RY00]. Ya se encuentra concluyendo su trabajo de investigación de maestría.

Uso del EAA-MAER en la solución de instancias del problema de programación no lineal mezclada con enteros (MINLP, mixed integer non-linear programming)

El autor de esta investigación doctoral propone usar el EAA-MAER también para resolver algunas instancias del MINLP, en [Her04], usa este algoritmo para resolver una colección de problemas tomados de [FPA99] entre los que se encuentran algunas instancias de ese problema. Los resultados obtenidos han sido satisfactorios.

Probando el EAA-MAER con un problema MINIMAX

La forma de plantear un problema de optimización puede cambiar drásticamente la dificultad para resolverlo. En el capítulo II se mencionó que una de entre varias formas de manejar las restricciones es usar una representación tal que todas las soluciones propuestas por el método, o una proporción muy grande de ellas, estén en región factible. Otra estrategia es "arreglar" toda solución no factible propuesta por el método de manera tal que se convierta en una solución factible. En este anexo se ejemplifica la aplicación de esta segunda estrategia con el problema que a continuación se enuncia:

"Encontrar un polinomio de grado n, $P^n(x)$, con todas sus raíces reales y dentro del intervalo [-1, 1], y cuyo coeficiente para x^n sea 1, tal que $\max \left|P^n(x)\right|$ sea mínimo."

La Representación

Cada una de las posibles soluciones se representa como una cadena binaria resultado de concatenar n subcadenas binarias en código Gray cada una de 32 bits y cada una representando un entero binario que a su vez representa una de las raíces del polinomio. Cada entero binario codifica un número real en el intervalo [-1, 1], el entero binario 00 . . .0 representa el valor -1 y el entero binario 11 . . . 1 representa al valor 1. En la ecuación C1.0 se muestra la función de correspondencia.

$$f = \frac{1-(-1)}{2^l - 1} b + (-1) \qquad \text{(C1.0)}$$

donde

b es la representación en base decimal del entero binario,
l es la longitud de las cadenas usadas para representar los enteros binarios
f es el número real correspondiente al entero b.

Las restricciones

Sean $r_1, r_2, \ldots, r_n$ las n raíces reales que representa una solución, represéntese esa solución de la siguiente manera: $r_1 r_2 \ldots r_n$. Para que la solución sea factible debe cumplir las siguientes restricciones:

1) $r_1 \leq r_2$
2) $r_2 \leq r_3$
   . . .
n) $r_{n-1} \leq r_n$

Arreglando Soluciones No Factibles

Sea $r_{s1} r_{s2} \ldots r_{sn}$ una solución donde $s1 = 1, s2 = 2, s3 = 3, \ldots, sn = n$, supóngase que no se cumple la restricción (2) y que las demás restricciones sí se cumplen. La solución puede

hacerse factible simplemente intercambiando los valores de *s*2 y *s*3: *s*1 = 1, *s*2 =3, *s*3 = 2, . . . , *sn* = *n*. De manera general cualquier solución no factible puede convertirse en una factible ordenando las raíces que representa en orden creciente. Si se ordenan de esta manera todas las soluciones propuestas por al algoritmo evolutivo, entonces todas ellas serán factibles.
Obsérvese que este procedimiento no equivale a disminuir el espacio de búsqueda (desde el punto de vista del algoritmo de optimización los puntos que *podría* visitar son los mismos cuando se ordenan que cuando no se ordenan las soluciones), más bien sirve para convertir los puntos no factibles en puntos factibles. Equivale a introducir en el espacio de búsqueda *n*!-1 copias de cada punto factible y retirar los puntos no factibles.

Solución Analítica
Ya existe una solución analítica para este problema. Los polinomios que cumplen con el enunciado de arriba son los llamados polinomios de Chebyshev. La ecuación que define un polinomio de Chebyshev de grado *n* es la C1.1, y la que da sus raíces es la C1.2 ([DH03] pp. 195).

$$T_n(x) = \frac{1}{2}((x+\sqrt{x^2-1})^k + (x-\sqrt{x^2-1})^k) \text{ para } x \in R \qquad \text{(C1.1)}$$

$$x_k = \cos\left(\frac{2k-1}{2n}\pi\right) \quad \text{para } k = 1,\ldots,n \qquad \text{(C1.2)}$$

Se puede usar estas ecuaciones para comparar el desempeño del algoritmo evolutivo usado

Solución con el EAA-MAER
Una vez que la rutina encargada de descodificar la cadena binaria que representa las *n* raíces termina su trabajo, invoca una rutina que ordena de manera ascendente los *n* valores. Obsérvese que de esta manera las *n*! cadenas binarias que representan todas las permutaciones de esos valores son equivalentes entre sí.

Una vez que se tienen las *n* raíces $r_1, r_2, \ldots, r_n$ se invoca una rutina que se encarga de encontrar el valor máximo de $\left|P^{(n)}(x)\right|$ (el absoluto del polinomio). Esta rutina trabaja de la siguiente manera: busca todos los puntos en el conjunto *A* definido en C1.3:

$$A = \{x \mid x \text{ satisface la ecuación} \left|\frac{d}{dx}P^n(x)\right| = 0\} \qquad \text{(C1.3),}$$

y busca el máximo del conjunto $\{\left|P^{(n)}(x)\right|$ para $x \in A$, $\left|P^{(n)}(-1)\right|$, $\left|P^{(n)}(1)\right|\}$, ese valor corresponde a $\max\left|P^{(n)}(x)\right|$ para $x \in [-1, 1]$. Este es el valor que se asigna como aptitud a la cadena.

El EAA-MAER se aplica a poblaciones de estas cadenas.

Experimentos y Resultados
Se hicieron diferentes experimentos para algunos valores de *n*. En la tabla C1.1 se muestran los valores de los parámetros para un experimento en el que se fijó un valor de *n* = 25. Se llevaron a cabo 30 corridas independientes entre sí. En la tabla C1.2 se muestran los mínimos

de los valores máximos de los polinomios obtenidos para cada una de estas corridas. En la tabla C1.3 se muestran las raíces del mejor polinomio obtenido (aquél cuyo valor máximo es el mínimo de todos los valores máximos de todos los polinomios probados).

Tabla C1.1 Valores de los parámetros del EAA-MAER para minimizar el máximo valor de un polinomio de grado 25 (n = 25)

Parámetro	Valor fijado
Tamaño de la población	2
Número de hijos por individuo para la creación de la población intermedia	{1, 2}
Proclividad máxima a la edición de receptores	0.1
Número de evaluaciones por individuo	0.01 × longitud del cromosoma
Fracción de buenos saltos	0.5
Probabilidad de cruzamiento	1
Número de evaluaciones	1,000,000

Tabla C1.2 Mínimos valores de los máximos de los mejores polinomios obtenidos para cada una de las 30 corridas independientes (n = 25).

Número de corrida	Mínimo obtenido del máximo valor del polinomio con el EAAMRE		Número de corrida	Mínimo obtenido del máximo valor del polinomio con el EAAMRE
1	1.41398E-07		16	1.49136E-07
2	1.10707E-07		17	1.02579E-07
3	1.67761E-07		18	1.25867E-07
4	8.93058E-08		19	1.19103E-07
5	1.01814E-07		20	8.78159E-08
6	9.1879E-08		21	1.69329E-07
7	**7.92485E-08**		22	9.08084E-08
8	1.28125E-07		23	1.14711E-07
9	1.00148E-07		24	1.4211E-07
10	1.26015E-07		25	1.41655E-07
11	1.76697E-07		26	1.46958E-07
12	1.61325E-07		27	1.53674E-07
13	1.59678E-07		28	1.40834E-07
14	9.40767E-08		29	9.51879E-08
15	1.73834E-07		30	1.12576E-07

En la tabla C1.3 se muestran los valores de las raíces obtenidos con el EAA-MAER y las exactas, obtenidas con la ecuación C1.2. El error absoluto se define en la ecuación C1.4.

$$|error\ absoluto| = |r_{EAAMRE} - r_{exacta}| \qquad \text{(C1.4)}$$

donde

r_{EAAMRE} es el valor de la raíz obtenida con el EAA-MAER,

r_{exacta} es el valor de la raíz obtenida con la ecuación C1.2.

El error relativo se define en la ecuación C1.5.

$$|error\ relativo| = \frac{|r_{EAAMRTE} - r_{exacta}|}{(|r_{EAAMRE}| + |r_{exacta}|)/2} \quad \text{para } |r_{EAAMRE}| \neq 0 \text{ y } |r_{exacta}| \neq 0 \qquad \text{(C.5)}$$

Para la raíz 13, en la columna del error relativo se ha escrito |*error absoluto*|.

Tabla C1.3 Raíces del polinomio con menor valor máximo encontrado, es el correspondiente a la corrida 7 de la tabla C1.2.

Número de raíz	Raíz obtenida por el EAAMRE	Raíz	|error absoluto|	error relativo
1	0.9979	0.9980	6.0E-05	6.0E-05
2	0.9816	0.9823	7.1E-04	7.2E-04
3	0.9490	0.9511	2.1E-03	2.2E-03
4	0.8993	0.9048	5.5E-03	6.1E-03
5	0.8201	0.8443	2.4E-02	2.9E-02
6	0.7935	0.7705	2.3E-02	2.9E-02
7	0.6844	0.6845	6.9E-05	1.0E-04
8	0.5879	0.5879	1.2E-05	2.0E-05
9	0.4748	0.4818	7.0E-03	1.5E-02
10	0.3554	0.3681	1.3E-02	3.5E-02
11	0.2269	0.2487	2.2E-02	9.2E-02
12	0.1420	0.1253	1.7E-02	1.2E-01
13	-0.0044	0.0000	4.4E-03	4.4E-03
14	-0.1394	-0.1253	1.4E-02	1.1E-01
15	-0.3234	-0.2487	7.5E-02	2.6E-01
16	-0.3795	-0.3681	1.1E-02	3.0E-02
17	-0.3991	-0.4818	8.3E-02	1.9E-01
18	-0.5784	-0.5879	9.5E-03	1.6E-02
19	-0.6828	-0.6845	1.7E-03	2.4E-03
20	-0.7655	-0.7705	5.0E-03	6.5E-03
21	-0.8444	-0.8443	5.4E-05	6.4E-05
22	-0.9073	-0.9048	2.5E-03	2.7E-03
23	-0.9562	-0.9511	5.1E-03	5.4E-03
24	-0.9908	-0.9823	8.5E-03	8.6E-03
25	-0.9944	-0.9980	3.6E-03	3.6E-03
Promedio			1.3E-02	3.9E-02

En las tablas C1.4, C1.5 y C1.6, se muestran tablas similares para un polinomio de grado 100 (n = 100).

Tabla C1.4 Valores de los parámetros del EAA-MAER para minimizar el máximo valor de un polinomio de grado 100 (n = 100)

Parámetro	Valor fijado
Tamaño de la población	2
Número de hijos por individuo para la creación de la población intermedia	{1, 2}
Proclividad máxima a la edición de receptores	0.1
Número de evaluaciones por individuo	0.01 × longitud del cromosoma
Fracción de buenos saltos	0.05
Probabilidad de cruzamiento	1
Número de evaluaciones	2,000,000

Tabla C1.5 Mínimos valores de los máximos de los mejores polinomios obtenidos para cada una de las 10 corridas independientes (n = 100).

Número de corrida	Mínimo obtenido del máximo valor del polinomio con el EAA-MAER
1	1.06869E-29
2	1.44782E-29
3	1.98895E-29
4	1.25361E-29
5	**7.74955E-30**
6	1.10957E-29
7	1.55852E-29
8	1.1134E-29
9	1.6287E-29
10	1.12278E-29

Tabla C1.6 Raíces del polinomio con menor valor máximo encontrado, es el correspondiente a la corrida 5 de la tabla C1.5

No. de la raíz	Raíz obtenida por el EAA	Raíz	|error absoluto|	|error relativo|		No. de la raíz	Raíz obtenida por el EAA	Raíz	|error absoluto|	|error relativo|
1	0.99983	0.9999	7.3E-05	7.3E-05		26	0.68640	0.6959	9.5E-03	1.4E-02
2	0.99766	0.9989	1.2E-03	1.2E-03		27	0.67795	0.6730	4.9E-03	7.3E-03
3	0.99253	0.9969	4.4E-03	4.4E-03		28	0.63756	0.6494	1.2E-02	1.8E-02
4	0.99152	0.9940	2.5E-03	2.5E-03		29	0.62746	0.6252	2.3E-03	3.6E-03
5	0.99128	0.9900	1.3E-03	1.3E-03		30	0.61340	0.6004	1.3E-02	2.1E-02
6	0.99066	0.9851	5.6E-03	5.6E-03		31	0.57559	0.5750	5.9E-04	1.0E-03
7	0.97300	0.9792	6.2E-03	6.4E-03		32	0.52963	0.5490	1.9E-02	3.6E-02
8	0.96948	0.9724	2.9E-03	3.0E-03		33	0.49080	0.5225	3.2E-02	6.3E-02
9	0.96762	0.9646	3.0E-03	3.1E-03		34	0.46770	0.4955	2.8E-02	5.8E-02
10	0.94470	0.9558	1.1E-02	1.2E-02		35	0.44067	0.4679	2.7E-02	6.0E-02
11	0.92868	0.9461	1.7E-02	1.9E-02		36	0.44063	0.4399	7.3E-04	1.7E-03
12	0.92738	0.9354	8.0E-03	8.6E-03		37	0.42624	0.4115	1.5E-02	3.5E-02
13	0.92194	0.9239	2.0E-03	2.1E-03		38	0.38908	0.3827	6.4E-03	1.7E-02
14	0.91200	0.9114	6.0E-04	6.6E-04		39	0.36786	0.3535	1.4E-02	4.0E-02
15	0.90289	0.8980	4.9E-03	5.4E-03		40	0.36251	0.3239	3.9E-02	1.1E-01
16	0.88592	0.8838	2.1E-03	2.4E-03		41	0.27472	0.2940	1.9E-02	6.8E-02
17	0.86964	0.8636	6.0E-03	7.0E-03		42	0.27108	0.2639	7.2E-03	2.7E-02
18	0.86804	0.8526	1.5E-02	1.8E-02		43	0.25264	0.2334	1.9E-02	7.9E-02
19	0.84693	0.8358	1.1E-02	1.3E-02		44	0.16923	0.2028	3.4E-02	1.8E-01
20	0.82972	0.8181	1.2E-02	1.4E-02		45	0.15561	0.1719	1.6E-02	9.9E-02
21	0.82829	0.7997	2.9E-02	3.5E-02		46	0.12943	0.1409	1.1E-02	8.5E-02
22	0.78513	0.7804	4.7E-03	6.0E-03		47	0.12448	0.1097	1.5E-02	1.3E-01
23	0.75655	0.7604	3.8E-03	5.1E-03		48	0.05009	0.0785	2.8E-02	4.4E-01
24	0.74438	0.7396	4.8E-03	6.4E-03		49	0.02680	0.0471	2.0E-02	5.5E-01
25	0.70644	0.7181	1.2E-02	1.6E-02		50	0.02302	0.0157	7.3E-03	3.8E-01

No. de la raíz	Raíz obtenida por el EAA	Raíz	|error absoluto|	|error relativo|	No. de la raíz	Raíz obtenida por el EAA	Raíz	|error absoluto|	|error relativo|
51	-0.01143	-0.0157	4.3E-03	3.1E-01	76	-0.71367	-0.7181	4.4E-03	6.2E-03
52	-0.03150	-0.0471	1.6E-02	4.0E-01	77	-0.72599	-0.7396	1.4E-02	1.9E-02
53	-0.03825	-0.0785	4.0E-02	6.9E-01	78	-0.77294	-0.7604	1.3E-02	1.6E-02
54	-0.11405	-0.1097	4.3E-03	3.9E-02	79	-0.78917	-0.7804	8.8E-03	1.1E-02
55	-0.12497	-0.1409	1.6E-02	1.2E-01	80	-0.79184	-0.7997	7.9E-03	9.9E-03
56	-0.13510	-0.1719	3.7E-02	2.4E-01	81	-0.80565	-0.8181	1.2E-02	1.5E-02
57	-0.20138	-0.2028	1.4E-03	7.0E-03	82	-0.83894	-0.8358	3.1E-03	3.8E-03
58	-0.26377	-0.2334	3.0E-02	1.2E-01	83	-0.85975	-0.8526	7.2E-03	8.4E-03
59	-0.26406	-0.2639	1.6E-04	6.0E-04	84	-0.87502	-0.8636	1.1E-02	1.3E-02
60	-0.27866	-0.2940	1.5E-02	5.4E-02	85	-0.91407	-0.8838	3.0E-02	3.4E-02
61	-0.32543	-0.3239	1.5E-03	4.7E-03	86	-0.91734	-0.8980	1.9E-02	2.1E-02
62	-0.33867	-0.3535	1.5E-02	4.3E-02	87	-0.92910	-0.9114	1.8E-02	1.9E-02
63	-0.38578	-0.3827	3.1E-03	8.0E-03	88	-0.93206	-0.9239	8.2E-03	8.8E-03
64	-0.44958	-0.4115	3.8E-02	8.8E-02	89	-0.93285	-0.9354	2.5E-03	2.7E-03
65	-0.47472	-0.4399	3.5E-02	7.6E-02	90	-0.93406	-0.9461	1.2E-02	1.3E-02
66	-0.50078	-0.4679	3.3E-02	6.8E-02	91	-0.93769	-0.9558	1.8E-02	1.9E-02
67	-0.50130	-0.4955	5.8E-03	1.2E-02	92	-0.94206	-0.9646	2.3E-02	2.4E-02
68	-0.52807	-0.5225	5.6E-03	1.1E-02	93	-0.96469	-0.9724	7.7E-03	8.0E-03
69	-0.52892	-0.5490	2.0E-02	3.7E-02	94	-0.97825	-0.9792	9.5E-04	9.7E-04
70	-0.53353	-0.5750	4.1E-02	7.5E-02	95	-0.98577	-0.9851	6.7E-04	6.8E-04
71	-0.59107	-0.6004	9.3E-03	1.6E-02	96	-0.98678	-0.9900	3.2E-03	3.3E-03
72	-0.59224	-0.6252	3.3E-02	5.4E-02	97	-0.99283	-0.9940	1.2E-03	1.2E-03
73	-0.64772	-0.6494	1.7E-03	2.6E-03	98	-0.99518	-0.9969	1.7E-03	1.7E-03
74	-0.66883	-0.6730	4.2E-03	6.2E-03	99	-0.99934	-0.9989	4.4E-04	4.4E-04
75	-0.69873	-0.6959	2.8E-03	4.1E-03	100	-0.99950	-0.9999	4.0E-04	4.0E-04
					Promedio			1.2E-02	5.5E-02

Como se puede observar en las tablas C1.3 y C1.6, el EAA-MAER se acercó bastante a la solución exacta de cada uno de los problemas. El promedio de los errores relativos para cada experimento fue de 3.9% para el primer experimento y de 5.5% para el segundo experimento (ver última fila de ambas tablas).

Probando el EAA-MAER con un problema de maximización de la concentración del producto deseado en un sistema de reacciones químicas complejas en una red de cinco reactores de mezcla completa

Introducción

Cuando en un sistema se llevan a cabo dos o más reacciones que comparten especies químicas entre sí, se dice que se tiene un *sistema complejo de reacciones* [FB90]. Las reacciones pueden ser en paralelo, en serie o en serie y paralelo. A continuación se muestran ejemplos de cada una de estas:

Reacciones en paralelo: $A \xrightarrow{k_1} B$ y $A \xrightarrow{k_2} C$ (C2.1)

Reacciones en serie: $A \xrightarrow{k_1} B \xrightarrow{k_2} C$ (C2.2)

Reacciones en serie y paralelo: $A \xrightarrow{k_1} B \xrightarrow{k_2} C$ y $2A \xrightarrow{k_3} D$ (C2.3)

Por lo general únicamente uno de los productos es el que se desea obtener, ése es el que se denomina *producto deseado*. Cuando se llevan a cabo reacciones complejas en un sistema, se procura que el sistema esté en condiciones tales que se favorezca la formación del producto deseado y que su concentración en la mezcla resultante sea la mayor posible. En esta sección se denomina *B* al producto deseado.
Los reactores ideales son recipientes donde suceden las reacciones. Se llaman ideales porque se plantean algunas suposiciones sobre su funcionamiento que hacen que se simplifiquen fuertemente su modelación matemática. En el caso de los reactores de mezcla completa, se supone que el mezclado es perfecto y que, por tanto, no cambian ni las concentraciones ni la temperatura de un punto a otro de la mezcla reaccionante dentro del reactor.

El problema
Floudas *et al.*, presentan un problema ([FPA99]pp. 128-138) en donde hay que maximizar la concentración del producto B en una reacción del tipo van de Vusse (reacción (C2.3) arriba) que se lleva a cabo en un sistema de cinco reactores de mezcla completa (CSTR, *constant stirred tank reactor*). En la figura C2.1 se esquematiza el sistema de reactores.

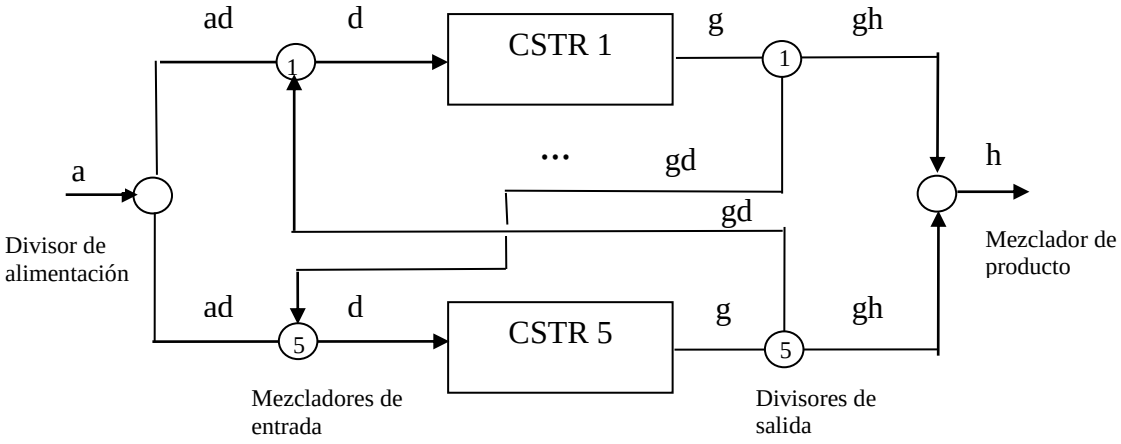


Figura C2.1 Sistema de reactores de mezcla completa para llevar a cabo una reacción del tipo van de Vusse

Sean:

$c_{l,i}^{m}$ concentración de la substancia *i* en la corriente *m* asociada al reactor *l*. Unidades: mol/L,

c_{i}^{m} concentración de la substancia *i* en la corriente *m*, donde m es la corriente de entrada global *a*, o la de salida global *h*. Unidades mol/L,

F_{l}^{m} flujo volumétrico de la corriente *m* asociada al reactor *l*. Unidades: L/hr,

$F_{l,l'}^{m}$ flujo volumétrico de la corriente *m* que va del reactor *l* al *l*'. Unidades: L/hr,

F^{a} flujo de entrada global. Unidades: L/hr,

F^{h} flujo de salida global. Unidades: L/hr

V_l volumen de la mezcla reaccionante en el reactor *l*. Unidades: L,

T_l temperatura de la mezcla reaccionante en el reactor *l*. Unidades: ^{o}K,

$\nu_{i,j}$ coeficiente estequiométrico de la substancia *i* en la reacción *j*. Sin unidades, es negativo para los reactivos y positivo para los productos,

$r_{l,j}$ velocidad de reacción. Unidades mol/(h L)

k_j factor preexponencial de la ecuación de Arrhenius. Unidades: para las reacciones 1 y 2: h^{-1}; para la reacción 3: L/(mol h),

E_j energía de activación de la reacción *j*. Unidades kcal/mol,

R constante de la ecuación del gas ideal = 1.987 cal/(ºK mol).

El problema está planteado para casos en los que la densidad es la misma para todas las corrientes del sistema. Las ecuaciones que deben cumplirse (restricciones de igualdad) son:

1) Balance en el Divisor de alimentación:

$$F^a = \sum_{l=1}^{5} F_l^{ad}$$ 1 ecuación

2) Balances en los mezcladores de entrada:

$$F_l^d = F_l^{ad} + \sum_{l'=1}^{5} F_{l',l}^{gd} \quad \text{para } l = 1, ..., 5$$ 5 ecuaciones

$$c_{i,l}^d F_l^d = c_i^a F^a + \sum_{l'=1}^{5} c_{i,l'}^g F_{l',l}^{gd} \quad \text{para } i \in \{A, B, C, D\},\ l = 1, ..., 5$$ 20 ecuaciones

3) Balances en los reactores:

$$F_l^d = F_l^g \quad \text{para } l = 1, ..., 5$$ 5 ecuaciones

$$c_{i,l}^g F_l^g = c_{i,l}^d F_l^d + V_l \sum_{j=1}^{3} \nu_{i,j} r_{l,j} \quad \text{para } i \in \{A, B, C, D\},\ l = 1, ..., 5$$ 20 ecuaciones

4) Velocidades de reacción en los reactores:

$$r_{l,1} = k_1 \exp\left(-\frac{E_1}{R \cdot T_l}\right) \cdot c_{A,l}^g \quad \text{para } l = 1, ..., 5$$ 5 ecuaciones

$$r_{l,2} = k_2 \exp\left(-\frac{E_2}{R \cdot T_l}\right) \cdot c_{B,l}^g \quad \text{para } l = 1, ..., 5$$ 5 ecuaciones

$$r_{l,3} = k_3 \exp\left(-\frac{E_3}{R \cdot T_l}\right) \cdot \left(c_{A,l}^g\right)^2 \quad \text{para } l = 1, ..., 5$$ 5 ecuaciones

5) Balances en los divisores de salida:

$$F_l^g = \sum_{l'=1}^{5} F_{l,l'}^{gd} \quad \text{para } l = 1, ..., 5$$ 5 ecuaciones

6) Balances en el mezclador de productos:

$$F^h = \sum_{l=1}^{5} F_l^{gh}$$ 1 ecuación

$$c_i^h F_l^h = \sum_{l=1}^{5} c_{i,l}^g F_l^{gh} \quad \text{para } l = 1, ..., 5$$

para $i \in \{A, B, C, D\}$, $l = 1, ..., 5$ 5 ecuaciones

Total : 76 ecuaciones

El problema de optimización puede plantearse entonces así:
Maximizar c_B^h, sujeta a las 76 restricciones de igualdad de arriba.

Los datos
Se dan como datos:
1) El flujo de entrada global y las concentraciones de entrada globales:
$F^a = 100, c_A^a = 1, c_B^a = 0, c_C^a = 0, c_D^a = 0$.
2) Los coeficientes estequiométricos de las tres reacciones químicas:

$$\begin{matrix} & \text{reac1} & \text{reac2} & \text{reac3} \end{matrix}$$
$$v_{i,j} = \begin{pmatrix} -1 & 0 & -2 \\ 1 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

3) Los factores preexponenciales de la ecuación de Arrhenius (las ecuaciones (4) arriba) para cada una de las tres reacciones:
$k_{1,0} = 5.4 \times 10^9, k_{2,0} = 1.6 \times 10^{12}, k_{3,0} = 3.6 \times 10^5$.

4) Las energías de activación de las tres reacciones: E_1, E_2, E_3. Ya divididas entre la constante del gas ideal son: $E_1/R = 7971.82, E_2/R = 11957.7, E_3/R = 3985.91$.

El número de variables cuyos valores debe explorar el algoritmo evolutivo se muestra en la tabla 8.1. Para un reactor de mezcla completa el recircular parte de su corriente de salida hacia su entrada no modifica el desempeño del reactor por lo que el flujo de las cinco recirculaciones de este tipo se fija en 0 ($F_{1,1}^{gd} = F_{2,2}^{gd} = F_{3,3}^{gd} = F_{4,4}^{gd} = F_{5,5}^{gd} = 0$). Entonces el número total de variables a explorar es 110.

La codificación
Cada coordenada de un punto en el espacio de búsqueda se codificó como un número entero binario, asignando a la cadena constituida por ceros (000 . . .0) el límite inferior de la coordenada y a la cadena constituida por unos (111 . . . 1) el limite superior. La cadena binaria resultante de concatenar cada una de estas subcadenas que representan a cada coordenada es la representación de un punto en el espacio de búsqueda. Cada subcadena binaria está codificada en código Gray. Se usaron 32 bits para cada subcadena.

Normalización de las 76 Restricciones de Igualdad
Cada una de las restricciones de igualdad puede representarse como se muestra en (4).

$$e.i. = e.d. \qquad \text{(C2.4)}$$

donde
e.i. significa expresión del lado izquierdo,
e.d. significa expresión del lado derecho.

Tabla C2.1 Variables del proceso.

Descripción de la parte del proceso	Descripción de las variables	Número total de variables
Entradas a los reactores	Para un reactor: Flujo volumétrico 1 Concentraciones 4 Total: 5	$5 \times 5 = 25$
Salidas de los reactores	Para un reactor: Flujo volumétrico 1 Concentraciones 4 Total: 5	$5 \times 5 = 25$
Variables del reactor	Para un reactor: Velocidades de reacción 3 Temperatura 1 Volumen de la mezcla reaccionante 1 Total 5	$5 \times 5 = 25$
Flujos de un reactor a otro reactor	Para un reactor: 5	$5 \times 5 = 25$
Entradas frescas a los reactores	Entradas a los reactores (F^{ad}) 5	5
Salidas del proceso	Flujo global (F^{h}) 1 Salidas de los reactores (F^{gh}) 5 Concentraciones 4 Total 10	10
	Número total de variables =	115

Los valores que puede tomar *e.i.* y *e.d.* dependen de las magnitudes que se manejen en cada restricción (flujos volumétricos, concentraciones, etc.) y por tanto el error en la igualdad más que medirse como un error absoluto, conviene que se mida como un error relativo, es decir, como una relación. Se puede usar una expresión como la propuesta en (C2.5a) para estimar el error en el cumplimiento de la restricción de igualdad.

$$\frac{|e.i. - e.d.|}{(|e.i| + |e.d.|)/2} \quad \text{para } |e.i.| + |e.d.| \neq 0 \qquad \text{(C2.5a)}$$

Nótese que cuando *e.i.* = *e.d.*, entonces la relación (C2.5) vale 0, y cuando $|e.i.| - |e.d.|$ tiende a ser muy grande, entonces la relación (C2.5) tiende a 2 por abajo. Tenemos entonces que

$$0 \leq \frac{|e.i. - e.d.|}{(|e.i| + |e.d.|)/2} < 2 \text{ para } |e.i.| + |e.d.| \neq 0 \qquad \text{(C2.5b)}$$

La *i-ésima* restricción de igualdad normalizada queda entonces como se muestra en (C2.6).

$$h_i^n = \frac{|e.i._i - e.d._i|}{(|e.i._i| + |e.d._i|)/2} = 0 \text{ para } |e.i.| + |e.d.| \neq 0 \qquad \text{(C2.6)}$$

donde

h_i^n es la *i-ésima* restricción de igualdad normalizada.

La ventaja de normalizar cada una de las restricciones de igualdad es que pueden ser convertidas en restricciones de desigualdad usando una variable de tolerancia que será la misma para todas las restricciones normalizadas como se muestra en (C2.7).

$$g_i^n = \frac{|e.i._i - e.d._i|}{(|e.i._i| + |e.d._i|)/2} - tolerancia \leq 0 \text{ para } |e.i.| + |e.d.| \neq 0 \qquad \text{(C2.7)}$$

donde

g_i^n es la *i-ésima* restricción de igualdad normalizada y convertida en restricción de desigualdad.

Obsérvese que cuando $|e.i._i - e.d._i|$ tiende a cero entonces la relación (5a) tiende a ser el error relativo en el cumplimiento de la *i-ésima* restricción. Por ejemplo, si la desigualdad (C2.7) se cumple con un valor de *tolerancia* igual a 0.001, esto significa que la *i-ésima* restricción se ha satisfecho con un error relativo máximo de 0.001 ó 0.1%.

El problema puede entonces replantearse así:

Maximizar c_B^h (C2.8a)

Sujeta a:

$$g_i^n = \frac{|e.i._i - e.d._i|}{(|e.i._i| + |e.d._i|)/2} - tolerancia \leq 0 \text{ para } |e.i._i| + |e.d._i| \neq 0; \text{ para } i = 1, \ldots, 76 \qquad \text{(C2.8b)}$$

Resultados

Se intentó resolver el problema planteado en (C2.8) fijando *tolerancia* en 0.0001 para algunas corridas y 0.001 para otras. Se usó el EAA-MAER con 500, 000 evaluaciones. No se obtuvo ni un solo punto en región factible. Se aumentó el número de evaluaciones hasta 10 millones y tampoco se obtuvo ni un punto en la región factible.

Se cambió el problema para tratar de ver cómo trabaja el EAA-MAER. Se planteó el problema:

Minimizar *tolerancia* (C2.9a)

Sujeta a:

$$g_i^n = \frac{|e.i._i - e.d._i|}{(|e.i._i| + |e.d._i|)/2} - tolerancia \leq 0 \text{ para } |e.i._i| + |e.d._i| \neq 0; \text{ para } i = 1, \ldots, 76 \qquad \text{(C2.9b)}$$

De esta manera se puede estudiar que tan bueno es el EAA-MAER para acercarse a la región factible. La variable *tolerancia* se codificó en el genoma y sus límites se fijaron en el intervalo [0,1]. Se usó el EAA-MAER con los valores de los parámetros mostrados en la tabla C2.2:

Tabla C2.2 Valores de los parámetros para la corrida de prueba de acercamiento a la región factible

Parámetro	Valor fijado
Tamaño de la población	4
Número de hijos por individuo para la creación de la población intermedia	2
Número de evaluaciones por individuo	0.01 × longitud del cromosoma
Fracción de buenos saltos	0.5
Probabilidad de cruzamiento	1
Número de evaluaciones	20,000,000

En la tabla C2.3 se muestran algunos de los valores de la *tolerancia* a lo largo de la corrida.

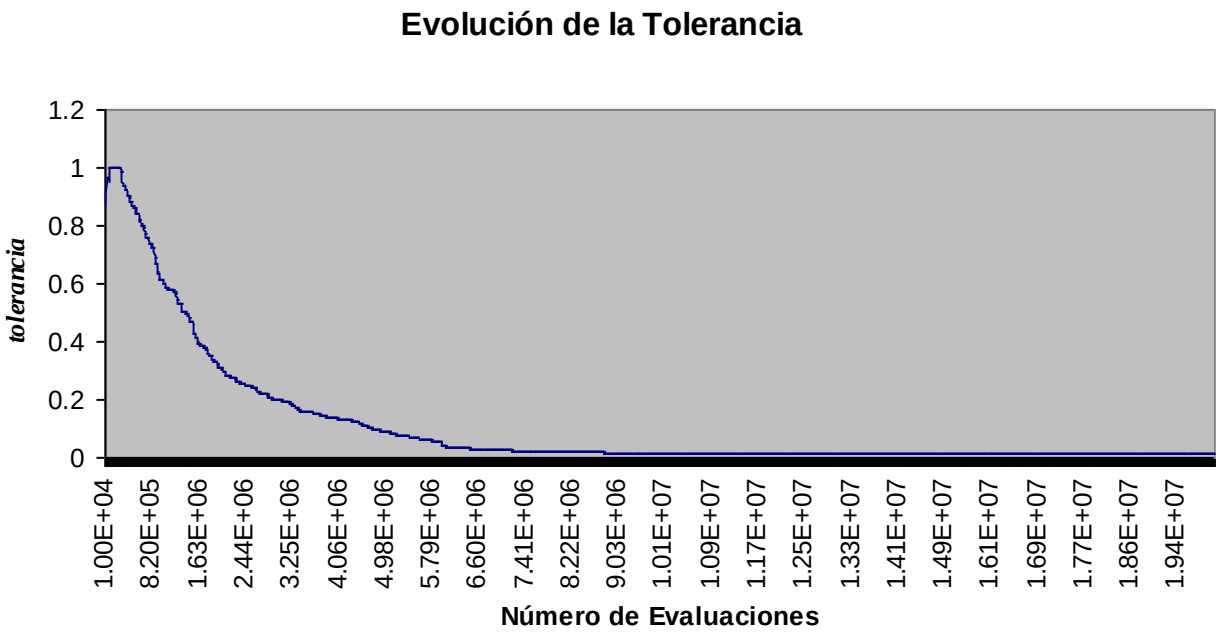
Tabla C2.3 Valores de la tolerancia a lo largo de una corrida

Número de Evaluaciones	*tolerancia*		Número de Evaluaciones	*tolerancia*
10,000	0.87797939		11,000,000	0.012996714
1,000,000	0.61527272		12,000,000	0.012513446
2,000,000	0.31201697		13,000,000	0.011719821
3,000,000	0.200234479		14,000,000	0.011436699
4,000,000	0.135778841		15,000,000	0.011417385
5,000,000	0.087938123		16,000,000	0.011417385
6,000,000	0.043684501		17,000,000	0.011417385
7,000,000	0.026653328		18,000,000	0.011417385
8,000,000	0.019894577		19,000,000	0.011417385
9,000,000	0.016729505		20,000,000	0.011253356
10,000,000	0.014473720			

La gráfica C2.1 muestra la evolución de los valores de *tolerancia* con el número de evaluaciones.

Obsérvese el comportamiento de la *tolerancia* al inicio de la evolución. A las diez mil evaluaciones, que es donde empieza la gráfica, la tolerancia vale menos de uno, aproximadamente hasta la mitad del intervalo [10,000, 810,000] sube el valor de la tolerancia hasta el valor máximo que puede tomar:1, este es el límite superior del intervalo que se le asignó (de 0 a 1). Esto indica que en esta parte de la corrida el algoritmo ha optado por aumentar la tolerancia para hacer que se cumplan las restricciones, en lugar de hacer que la expresión del lado izquierdo. *e.i.*, se parezca a la del lado derecho, *e.d.*, que es lo deseable. Sin embargo, a partir de la mitad del intervalo [10,000, 810,000] comienza a descender de manera consistente. Esto indica que comienza ha hacer que *e.i.* se parezca a *e.d.*, que es el comportamiento que se desea que tenga. Pero Aproximadamente a partir de los 14 millones de evaluaciones la disminución de la tolerancia se hace muy lenta. Al final de esta corrida se ha obtenido una tolerancia de 0.01125, esto es un error del 1.125% como máximo para cada una de las 76 restricciones. Este desempeño es esperanzador, pero no es lo suficientemente

bueno, se requiere por lo menos el disminuir en un orden de magnitud la *tolerancia*, esto es que sea de 0.001 o menos.



Gráfica C2.1 Evolución de los valores de la *tolerancia* con respecto al número de evaluaciones usados por el EAA-MAER.

Se ha probado con otros valores de los parámetros, con muchas más evaluaciones (cien millones), pero el comportamiento ha sido el mismo, no disminuye con la velocidad requerida el valor de la tolerancia.

El problema aquí presentado es mucho más complejo que los otros que se han abordado en este trabajo de investigación.

El autor de este trabajo de investigación doctoral ha expuesto estos avances en la 25ª Reunión de la AMIDIQ [Her04b]. Un problema de optimización de la concentración del producto deseado en un sistema mucho más sencillo de un reactor discontinuo usando el EAA-MAER fue presentado en al XXXVII Congreso de Química de la Sociedad Química de México.

Otras Estrategias para Resolver el Problema
Como en el caso anterior del problema MINIMAX, este problema se puede plantear de otra manera para facilitárselo al algoritmo evolutivo. Por ejemplo, se puede plantear como un problema de optimización sin restricciones. Así lo ha planteado el autor de este trabajo y ha podido obtener un buen resultado.

Expresando el problema sin restricciones.
La cuestión es expresar el problema sin las 76 restricciones de igualdad y respetando la no negatividad de todas sus variables, ya que ninguna de ellas debe ser negativa. Una forma de lograrlo es escoger un conjunto de variables independientes, proponer un valor para cada una de estas variables y resolver el resto de las variables, las variables dependientes, usando las 76

restricciones de igualdad. El evitar que las variables independientes sean negativas se puede lograr simplemente asignándoles valores no negativos. El problema de evitar que las variables dependientes tomen valores negativos se puede resolver introduciendo algunas variables y ecuaciones más y proponiendo una secuencia de cálculo en la que no aparezca el operador de resta cuando no se pueda garantizar que el substraendo sea menor o igual que el minuendo. Para los divisores se puede definir una fracción $R_{i,j}$, que es una fracción que representa la relación entre el *j*-ésimo flujo de salida del divisor y la corriente de entrada al divisor *i* (ver figura C2.2)

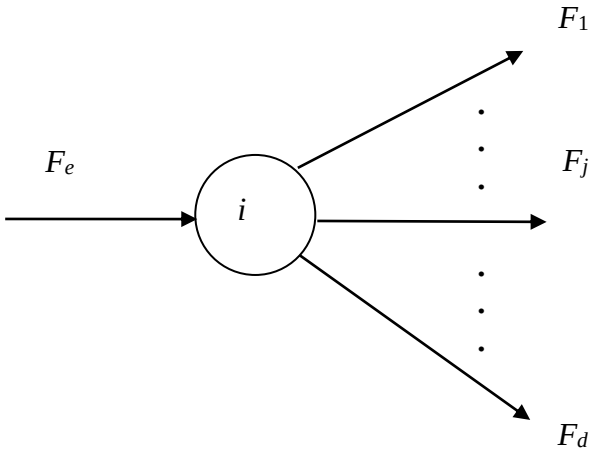


Figura C2.2 Flujos de entrada y salida de un divisor *i*.

La fracción $R_{i,j}$ se calcula entonces: $R_{i,j} = \frac{F_j}{F_e}$

Obsérvese que
$R_{i,j} \in [0, 1]$
y
$\sum_{j=1}^{d} R_{i,j} = 1$

El fijar *d*-1 fracciones $R_{i,j}$ a la vez, para después calcular la última fracción con la ecuación
$R_{i,d} = 1 - \sum_{j=1}^{d-1} R_{i,j}$
no siempre dará por resultado un valor no negativo para $R_{i,d}$. Para evitar que surja alguna fracción negativa se debe evitar realiza la resta en la expresión a la izquierda del signo igual. Una forma de evitar esto es asignar valores no negativos a cada uno de los *d* flujos a la salida de un divisor; después se calculan las relaciones $R_{i,j}$. Cuando se deba calcular los flujos a la salida del divisor se multiplicará la corriente de entrada por la fracción correspondiente a cada una de las corrientes de salida.
Las variables independientes
Se escogieron como variables independientes todas las fracciones $R_{i,j}$ de los divisores y se calcularon como se indica en la sección anterior, las temperaturas y los volúmenes de todos los reactores. El número total de variables se muestran en la tabla C2.4.

Tabla C2.4 Variables independientes en la solución del problema sin restricciones

Descripción de la parte del proceso	Descripción de las variables a las que se les fijaron valores.	Número de variables
Divisor de entrada global	Los cinco flujos de salida, uno para cada reactor.	5
Divisores a la salida de cada uno de los cinco reactores	Seis flujos de salida: cuatro hacia los otros reactores, uno de recirculación al mismo reactor y otro hacia el mezclador de la salida global. Los valores que el AG asigna a estas variables sólo sirven para calcular los valores de las fracciones $R_{i,j}$.	$5 \times (4 + 1 + 1) = 30$
Reactor	Temperatura y volumen	$5 + 5 = 10$
		Total= 45

En realidad son 40 variables, porque para un reactor de mezcla completa, como los que se usan en esta red de reactores, no se modifica la eficiencia del reactor por el hecho de poner una corriente de recirculación de su salida hacia su alimentación, por lo que esta corriente de recirculación se fijó en 0 para cada uno de los reactores.

La secuencia de cálculo

1) Se calculan las constantes cinéticas para cada una de las tres reacciones en cada uno de los reactores. También se calculan las fracciones $R_{i,j}$ de los divisores.
2) Se hace converger los valores de los flujos volumétricos que pasan por: los mezcladores a la entrada de los reactores, los reactores, los divisores a la salida de los rectores. Como valores iniciales de los flujos de salida de los divisores se usan los valores de esos flujos usados para calular las fracciones $R_{i,j}$.
3) Se hace converger los valores de las concentraciones.

Optimización con el EAA-MAER

El EAA-MAER "propone" valores para las 40 variables independientes, invoca a la rutina de la función objetivo que se encarga de aplicar la secuencia de cálculo arriba descrita. Esta rutina regresa como valor de aptitud el valor de la concentración de *B* en la corriente de salida del sistema (c_B^h).

Experimentación

Se llevaron a cabo 30 corridas independientes. Los datos de los parámetros de las corridas se muestran en la tabla C2.5.

Tabla C2.5 Parámetros del EAA-MAER

Parámetro	Valores
Individuos en la población de padres	4
Número de hijos por individuo	Aleatorio en {1, 2}
T_{Max}	0.1
f_{nea}	0.1
$T_{saltos\ buenos}$	0.3

Resultados

Los resultados de la mejor corrida se muestran en esta parte. En la tabla C2.6 se muestran los flujos del proceso. En las filas de la parte superior de la tabla se muestran los valores de las corrientes que salen de los divisores hacia los distintos mezcladores y hacia el mezclador de la salida global; por ejemplo, del divisor 1 se recircula un flujo de 0 hacia el mezclador del reactor 1, se manda un flujo de 0.00013 hacia el mezclador del reactor 2, se manda un flujo de 3.0476E-5 al mezclador del reactor 3, se manda u flujo de 100.0036 hacia el mezclador del reactor 4, se manda un flujo de 5.1036E-7 hacia el mezclador del reactor 5 y un flujo de 5.821E-7 hacia el mezclador de la salida global.

Tabla C2.6 Flujos volumétricos en la red de rectores. Resultados sin truncar.

		. . . al mezclador de entrada al reactor (F^{gd})					. . . al mezclador de salida global (F^{gh})
		1	2	3	4	5	
Del divisor	1	0	0.00013065	3.0476E-05	100.003602	5.1036E-07	5.821E-07
	2	0.00152643	0	100.015834	9.109E-05	6.0998E-05	0.0610613
	3	9.783E-07	0.0603698	0	0.01430029	0.00229827	99.9389658
	4	0.00229222	4.6574E-08	6.9861E-08	0	100.016313	9.315E-08
	5	1.1153E-05	100.01801	5.43E-05	0.00061231	0	3.3534E-06
						Salida global (F^{h}):	100
		1	2	3	4	5	
Alimentación de los divisores al mezclador de ent. al reac. (F^{gd}):		0.00383078	100.07851	100.015919	100.018606	100.018673	
Alimentación fresca al mezclador de entrada al react. (F^{ad}):		99.9999334	6.5264E-05	5.588E-07	4.6567E-08	7.4507E-07	Alimentación fresca global (F^{a}): 100
Alimentación total al reactor (F^{d}):		100.003764	100.078576	100.015919	100.018606	100.018674	
Salida del reactor (F^{g}):		100.003764	100.078576	100.015919	100.018606	100.018674	

En las columnas centrales de la tabla C2.6 se muestran los flujos de entrada a los mezcladores de los reactores; por ejemplo, al mezclador 2 entran los flujos de 0.00013065, 0, 0.0603698, 4.6574E-8 y 100.01801 provenientes de los divisores a la salida de los reactores, la suma de estos flujos es de 0.00383078; también entra un flujo de 99.9999334 proveniente del divisor de la entrada global, por eso se dice que es una alimentación fresca. La suma de estas dos últimas cifras da la alimentación total a ese mezclador, que es también la entrada total al reactor 2: 100.003764. En la última fila de esa columna se puede ver que el flujo volumétrico de salida del reactor es el mismo que entró a él.

Para apreciar con mayor claridad los resultados de esta tabla, en la tabla C2.7 se muestran los mismos valores de la tabla C2.6 truncados a 2 números después del punto decimal.

Tabla C2.7 Flujos volumétricos en la red de rectores. Se ha redondeado a dos cifras decimales.

		. . . al mezclador de entrada al reactor (F^{gd})					. . . al mezclador de salida global (F^{gh})
		1	2	3	4	5	
Del divisor	1	0.00	0.00	0.00	100.00	0.00	0.00
	2	0.00	0.00	100.02	0.00	0.00	0.06
	3	0.00	0.06	0.00	0.01	0.00	99.94
	4	0.00	0.00	0.00	0.00	100.02	0.00
	5	0.00	100.02	0.00	0.00	0.00	0.00
						Salida global (F^{h}):	100
		1	2	3	4	5	
Alimentación de los divisores al mezclador de ent. al reac. (F^{gd}):		0.00	100.08	100.02	100.02	100.02	
Alimentación fresca al mezclador de entrada al react. (F^{ad}):		100.00	0.00	0.00	0.00	0.00	Alimentación fresca global (F^{a}): 100
Alimentación total al reactor (F^{d}):		100.00	100.08	100.02	100.02	100.02	
Salida del reactor (F^{g}):		100.00	100.08	100.02	100.02	100.02	

Analizando los resultados de la tabla C2.7 se puede ver que prácticamente los reactores están colocados en serie: la alimentación fresca va toda hacia el reactor 1, de ahí al reactor 4, de éste al reactor 5, de éste al reactor 2, de ahí al reactor 3 y de él casi todo a la salida global: 99.94, 0.06 lo recircula al reactor 2 y 0.01 lo recircula al reactor 4. En la figura C2.3 se muestra esto.

En las tablas C2.8, C2.9 y C2.10 se muestran las concentraciones de las diferentes substancias en la entrada, salida de cada reactor y en la salida global de la red de reactores. La concentración en la corriente de salida global de la red se puede observar en la tabla C2.10, es de 0.8172. El mejor valor reportado en [FPA99] p. 138 es de 0.818992, también es el arreglo en serie el que da ese resultado.

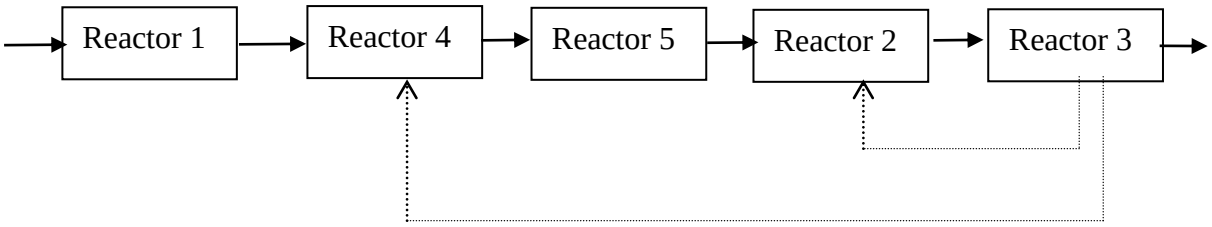


Figura C2.3. En la mejor solución encontrada por el EAA-MAER prácticamente todos los reactores están conectados en serie. Las líneas punteadas significan flujos volumétricos despreciables.

Tabla C2.8. Concentraciones en las entradas de los reactores

	d1	d2	d3	d4	d5
CA	1.0000	0.1021	0.0090	0.7531	0.4465
CB	0.0000	0.7401	0.8124	0.2166	0.4696
CC	0.0000	0.0697	0.0793	0.0139	0.0380
CD	0.0000	0.0440	0.0496	0.0082	0.0229

Tabla C2.9. Concentraciones en la salida del reactor

	g1	g2	g3	g4	g5
CA	0.7532	0.0090	0.0024	0.4465	0.1022
CB	0.2165	0.8124	0.8172	0.4696	0.7401
CC	0.0139	0.0793	0.0806	0.0380	0.0697
CD	0.0082	0.0496	0.0499	0.0229	0.0440

Tabla C2.10. Concentraciones en la salida global de la red de reactores

	h
CA	0.0024
CB	0.8172
CC	0.0806
CD	0.0499

En la tabla C2.11 se muestra el volumen de la mezcla reaccionante en cada reactor, la temperatura de operación, las constantes cinéticas de reacción y las velocidades de las tres reacciones químicas en cada reactor obtenidas en la mejor corrida del EAA-AMRE.

Tabla C2.11 Volumen de la mezcla reaccionante y temperatura de operación de cada reactor

	Reactor 1	Reactor 2	Reactor 3	Reactor 4	Reactor 5
Volumen, V	0.0113	8692.4280	9999.2865	0.1475	22.9406
Temperatura, T	549.6864	323.1295	305.6896	487.0392	401.5424
Cte. cinética 1, k1	2716.7126	0.1042	0.0255	420.6245	12.8928
Cte. Cinética 2, k2	570.9770	1.3572E-04	1.6435E-05	34.7855	0.1867
Cte. Cinética 3, k3	255.3451	1.5817	0.7825	100.4738	17.5905
Vel. de reac. 1, r1	2046.2696	9.4251E-04	6.1682E-05	187.8237	1.3175
Vel. de reac. 2, r2	123.6405	1.1026E-04	1.3431E-05	16.3339	0.1382
Vel. de reac. 3, r3	72.4329	6.4648E-05	2.2867E-06	10.0169	0.0919

Apéndice D

Métodos Tradicionales para Resolver el Problema de Programación No Lineal.

El problema de programación no lineal puede expresarse de la siguiente manera:

Problema P:
Minimizar $f(\boldsymbol{x})$ (D.1)
Sujeta a
$g_i(\boldsymbol{x}) \leq 0$ para $i = 1, \ldots, m$ (D.2)
$h_i(\boldsymbol{x}) = 0$ para $i = 1, \ldots, l$ (D.3)
$\boldsymbol{x} \in X \subset R$ (D.4)
$f, g_i, h_i : R^n \rightarrow R$ (D.5)

donde por lo menos alguna de las funciones mostradas en D.5 es no lineal. D.1 es la función objetivo, D.2 son la restricciones de desigualdad, D.3 son las restricciones de igualdad. La definición del subconjunto X puede involucrar otras restricciones fáciles de cumplir tales como los límites inferior y superior de cada una de las variables x_i.

Se llaman aquí *métodos tradicionales* de solución del problema de programación no lineal a aquellos métodos que llevan a cabo la búsqueda del óptimo pasando de un punto a otro (y no de una colección de puntos a otra colección de puntos, como es el caso de los algoritmos genéticos). Algunos de estos métodos no usan las primeras derivadas, ni otras derivadas de orden superior, de la función objetivo o de las restricciones; otros requieren las primeras derivadas de la función objetivo; otros requieren las primeras y segundas derivadas de la función objetivo; otros más requieren las primeras derivadas de la función objetivo y de las restricciones. Para garantizar la convergencia al óptimo (y no a un óptimo local), todos piden que la función objetivo sea unimodal, condición que se garantiza cumplirá la función pidiendo que sea convexa o similar a una función convexa.
Cuando se cree haber llegado al óptimo hay que probar si el punto al que se llegó cumple con las condiciones de optimalidad. Estas condiciones son necesarias pero no suficientes si se está optimizando funciones multimodales, pero si se está optimizando funciones unimodales, son también suficientes.
Es necesario entonces, antes de hablar de los métodos de optimización, el definir las funciones convexas y otras funciones con algunas características similares a ellas, y hablar sobre las condiciones de optimalidad. En las secciones inmediatas siguientes se hablará sobre estos temas.

1 Funciones convexas y generalizaciones de funciones convexas.

Funciones convexas.

Sea una función f: $S \rightarrow R$ donde S es un conjunto no vacío en R^n. Se dice que la función f es *convexa* en S si

$$f(\lambda \boldsymbol{x}_1 + (1 - \lambda)\, \boldsymbol{x}_2) \leq \lambda\, f(\boldsymbol{x}_1) + (1 - \lambda)\, f(\boldsymbol{x}_2)$$

para cada $\boldsymbol{x}_1, \boldsymbol{x}_2 \in S$ y para cada $\lambda \in (0, 1)$. Si la función $-f$ cumple con estas condiciones se dice que la función f es *cóncava* en S.

Si el término a la izquierda de la desigualdad es estrictamente igual al término a la derecha para cada pareja de puntos distintos $\boldsymbol{x}_1$ y $\boldsymbol{x}_2$ en S y para toda $\lambda \in (0, 1)$, se dice que la función f es estrictamente convexa. Si es una función $-f$ la que cumple con las condiciones que se acaban de mencionar se dice que la función f es *estrictamente cóncava* en S. En las figuras D.1 se muestran ejemplos de estas funciones.

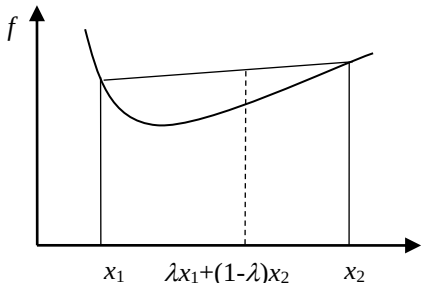


Figura D.1a. Una función estrictamente convexa.

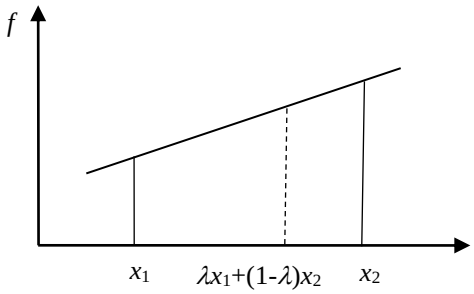


Figura D1.b. Una función convexa

Funciones casiconvexas.

Sea f: $S \rightarrow R$, donde S es conjunto no vacío convexo en R^n. Se dice que la función f es casiconvexa en S si

$$f(\lambda \boldsymbol{x}_1 + (1 - \lambda)\, \boldsymbol{x}_2) \leq \max\{f(\boldsymbol{x}_1), f(\boldsymbol{x}_2)\}$$

para cada $\boldsymbol{x}_1, \boldsymbol{x}_2 \in R^n$ y para cada $\lambda \in (0, 1)$. Si la función $-f$ es casiconvexa entonces la función f es *casicóncava*. En la figura D.2 se muestra un ejemplo de función casi casiconvexa.

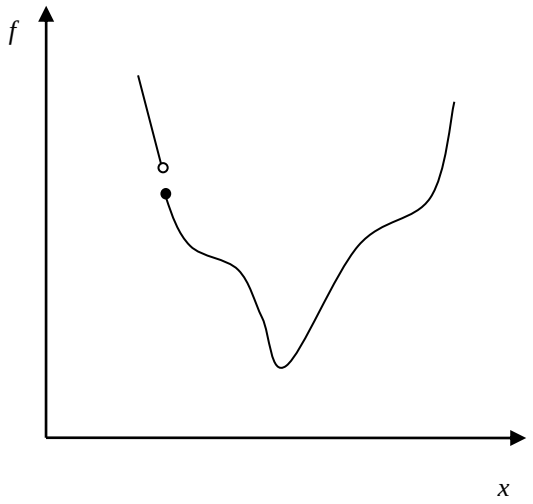


Figura D2. Una función casiconvexa

Funciones pseudoconvexas.

Sea S un conjunto no vacío abierto en R^n, sea f: $S \rightarrow R$ diferenciable en S. Se dice que la función f es pseudoconvexa si para cada $\boldsymbol{x}_1, \boldsymbol{x}_2 \in S$ con $\nabla f(\boldsymbol{x}_1)^t(\boldsymbol{x}_2 - \boldsymbol{x}_1) \geq 0$ se tiene que $f(\boldsymbol{x}_2)$

$\geq f(\boldsymbol{x}_1)$; o, de manera equivalente, si $f(\boldsymbol{x}_2) < f(\boldsymbol{x}_1)$, entonces $\nabla f(\boldsymbol{x}_1)^t(\boldsymbol{x}_2 - \boldsymbol{x}_1) < 0$ (obsérvese que éstas no son implicaciones *fuertes* sino *débiles*; por ejemplo, el que se cumpla que $\nabla f(\boldsymbol{x}_1)^t(\boldsymbol{x}_2 - \boldsymbol{x}_1) < 0$ *no implica* que se deba cumplir que $f(\boldsymbol{x}_2) < f(\boldsymbol{x}_1)$). En la figura D.3 se muestra una función pseudoconvexa.
Si la función $-f$ es pseudoconvexa, entonces la función f es casicóncava.

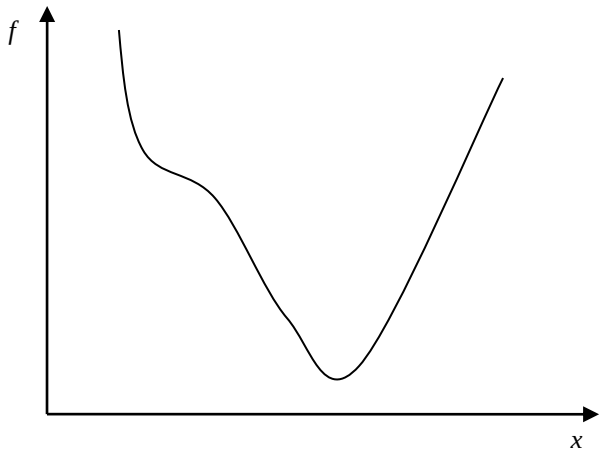


Figura D.3. Una función pseudoconvexa.

2 Condiciones de optimalidad para el problema de programación no lineal.
Las condiciones de optimalidad necesarias de Karush-Kuhn-Tucker[1] (KKT) se pueden expresar así:
Si $\boldsymbol{x}^*$ es un *óptimo local* al problema P, existe un vector $(\boldsymbol{u}, \boldsymbol{v})$ tal que

$$\nabla f(\boldsymbol{x}^*) + \sum_{i \in I} u_i \nabla g_i(\boldsymbol{x}^*) + \sum_{i=1}^{l} v_i \nabla h_i(\boldsymbol{x}^*) = \mathbf{0}$$

$$u_i \geq 0 \quad \text{para } i \in I$$

donde
u_i y v_i son los multiplicadores de Lagrange asociados con las restricciones $g_i(\boldsymbol{x}) \leq 0$ y $h_i(\boldsymbol{x}) = 0$, respectivamente,
$I = \{i: g_i(\boldsymbol{x}) = 0\}$.

Si además f es pseudoconvexa, g_i es casiconvexa para $i \in I$ y h_i es casiconvexa si $v_i > 0$ y casicóncava si $v_i < 0$, entonces $\boldsymbol{x}^*$ además de ser un *óptimo local* es el *óptimo* del problema P.
Algunos métodos para resolver el problema P consisten en resolver sucesivamente el problema sin restricciones. Es por eso que en la sección siguiente se mostrarán algunos

[1] Así las nombran, p.ej., Bazaara *et al.* [BSS93], Rardin [Rar98], en otras fuentes (p. ej. Edgar [EH89], Miller [Mil00])simplemente las llaman condiciones de Kuhn-Tucker (KT). Bazaraa *et al.* [BSS93] mencionan que Karush reporta parte de estas condiciones varios años antes que Kuhn y Tucker (quienes las descubrieron independientemente el uno del otro), es por eso que le dan crédito también a él.

métodos de optimización para problemas no restringidos, en la sección que le sigue se mostrarán algunos métodos de optimización restringida.

3 Métodos de solución del problema sin restricciones.
El problema sin restricciones es:

Problema P':
Minimizar $f(\boldsymbol{x})$
$f : R^n \rightarrow R$

Algunos métodos para resolverlo se describen a continuación.

3.1 Búsqueda sobre líneas (búsqueda unidimensional).
Estos métodos consisten en que dado un punto $\boldsymbol{x}_i$ se obtiene una dirección $\boldsymbol{d}_i$ y luego se busca un mínimo sobre la línea definida por la expresión $\boldsymbol{x}_i + \lambda \boldsymbol{d}_i$. El problema de encontrar un mínimo sobre esa línea, es un problema de optimización de una variable, λ. Llámesele λ_i al valor de λ donde se encuentra el mínimo, y $\boldsymbol{x}_{i+1}$ al punto donde se encuentra el mínimo: $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \lambda_i \boldsymbol{d}_i$. Se obtiene otra dirección $\boldsymbol{d}_{i+1}$ con el punto $\boldsymbol{x}_{i+1}$ y se repite el proceso.

3.1.1 Búsqueda unidimensional sin usar derivadas
Para encontrar el mínimo sin derivar con respecto a λ a la función $\phi(\lambda) = f(\boldsymbol{x}_i + \lambda \boldsymbol{d}_i))$, se pueden usar métodos tales como 1) búsqueda uniforme, 2) búsqueda dicótoma, 3) método de la sección dorada y 4) búsqueda de Fibonacci. Para aplicar estos métodos es necesario identificar un intervalo $[a_1, b_1]$ para λ dentro del cual se encuentre el mínimo. La función $\phi(\lambda)$ debe ser estrictamente casiconvexa para que se pueda garantizar que el mínimo obtenido no será un mínimo local.
3.1.1.1 Búsqueda uniforme. Consiste en dividir el intervalo $[a_1, b_1]$ en subintervalos menores de tamaño δ_t, se evalúa la función en los puntos $a, a+\delta, a+2\delta, \ldots, b$. Se identifica el subintervalo $[a_2, b_2] \in [a_1, b_1]$, de tamaño $b_2 - a_2 = 2\delta$, dentro del cual se encuentra el óptimo y se divide en subintervalos menores de tamaño δ_2. Se repite este procedimiento hasta que el tamaño del intervalo sea igual o menor a una mínima distancia $l > 0$ fijada de antemano.
3.1.1.2 Búsqueda dicótoma. Se colocan en el intervalo $[a_1, b_1]$ dos puntos λ_1 y μ_1 que se encuentran separados entre sí una distancia 2ε fijada de antemano y a una distancia ε del centro del intervalo. Si $\phi(\lambda_1) > \phi(\mu_1)$ entonces el mínimo se encuentra entre λ_1 y b_1, ese será el nuevo intervalo de búsqueda; por el contrario, si $\phi(\mu_1)$ es mayor que $\phi(\lambda_1)$ entonces el mínimo se encuentra entre μ_1 y a_1, ese será el nuevo intervalo de búsqueda. Este proceso se repite hasta que el tamaño del intervalo sea menor o igual a una mínima distancia $l > 0$ fijada de antemano.
3.1.1.3 Método de la sección dorada. Se obtienen los puntos λ_1 y μ_1, que caen dentro del intervalo $[a_1, b_1]$, usando las ecuaciones: $\lambda_1 = a_1 + (1 - \alpha)(b_1 - a_1)$ y $\mu_1 = a_1 + \alpha(b_1 - a_1)$ donde $\alpha = 0.618$. Si $\phi(\lambda_1) > \phi(\mu_1)$ el nuevo intervalo de búsqueda es $[\lambda_1, b_1]$; si por el contrario, $\phi(\mu_1) > \phi(\lambda_1)$ el nuevo intervalo de búsqueda es $[a_1, \mu_1]$. Se calculan los nuevos puntos λ_2, μ_2 usando las ecuaciones de arriba con los nuevos límites del intervalo. Se repite

este procedimiento hasta que el tamaño del intervalo sea menor o igual a una mínima distancia $l > 0$ fijada de antemano.

3.1.1.4 Búsqueda de Fibonacci. Se fija el tamaño del intervalo final que contendrá al mínimo, que es la mínima distancia final permitida $l < (b_1 - a_1)$. Se debe escoger el número de evaluaciones que se harán de la función tales que $F_n > [b_1 - a_1]/l$, donde n es el n-ésimo número de la serie de números de Fibonacci: 1, 1, 2, 3, 5, 8, 13, 21, 34, ... Se calculan dos puntos λ_1 y μ_1 dentro del intervalo $[a_1, b_1]$ usando las ecuaciones: $\lambda_1 = a_1 + (F_{n-2}/F_n)(b_1 - a_1)$ y $\mu_1 = a_1 + (F_{n-1}/F_n)(b_1 - a_1)/l$. Si $\phi(\lambda_1) > \phi(\mu_1)$ entonces el mínimo se encuentra en el intervalo $[\lambda_1, b_1]$; si $\phi(\mu_1) > \phi(\lambda_1)$ entonces el mínimo se encuentra en el intervalo $[a_1, \mu_1]$. Se vuelven a calcular otros puntos λ_2 y μ_2 dentro del intervalo donde se encuentre el mínimo usando las ecuaciones con las que se calcularon los puntos λ_1 y μ_1, pero substituyendo los valores de los límites del nuevo intervalo y los números F_{n-3} y F_{n-4} de la serie de Fibonacci. Se continúa de esta manera hasta que se usen las n evaluaciones de la función.

3.1.2 Búsqueda unidimensional usando derivadas.

Otra forma de encontrar el mínimo de la función $\phi(\lambda)$ es resolver la ecuación $\phi'(\lambda) = 0$ (ϕ' es la derivada de ϕ). Si la función ϕ' es casiconvexa sólo un punto satisfará esta ecuación ese será el mínimo. Se pueden aplicar los métodos de búsqueda de una raíz para la función ϕ'. Algunos de esos métodos son: método de la bisección, método de la regla falsa, método de la secante, método de Newton-Raphson.

3.2 Búsqueda multidimensional.

En estos métodos se minimiza en cada paso con respecto a más de una dirección de manera secuencial, *i.e.* , primero con respecto a una de las direcciones, después con respecto a otra de ellas y así sucesivamente hasta que se minimiza con respecto a todas las direcciones propuestas, se comprueba si el punto al final del paso, $\boldsymbol{x}_{i+1}$, está lo suficientemente cerca del punto al inicio del paso, $\boldsymbol{x}_i$, *i.e.* si $||\boldsymbol{x}_{i+1} - \boldsymbol{x}_i|| < \varepsilon$ (ε se fija de antemano), si se cumple esta condición se para, si no se cumple, se continúa con la búsqueda. Si la función es estrictamente casiconvexa se llegará al mínimo. Cada vez que se minimiza con respecto a una dirección se debe usar uno de los métodos de búsqueda unidimensional descritos en la sección 3.1 y sus subsecciones.

3.2.1 Búsqueda multidimensional sin usar derivadas.

Estos métodos no requieren que se calcule ninguna derivada de la función para ser aplicados.

3.2.1.1 El método de coordenadas cíclicas.

0) $i = 0$. 1) Dado el punto $\boldsymbol{x}_i \in R^n$ se definen n vectores: $\boldsymbol{d}_1, \ldots, \boldsymbol{d}_n$ cada uno de ellos paralelo a cada uno de los ejes cartesianos, por ejemplo: $\boldsymbol{d}_1 = (1, 0, \ldots, 0)$, $\boldsymbol{d}_2 = (0, 1, 0, \ldots, 0)$, . . . , $\boldsymbol{d}_n = (0, ,0, \ldots, 1)$. Estos son los vectores de dirección.

2) Se copia el vector $\boldsymbol{x}_i$ en el vector $\boldsymbol{y}$. Luego se minimiza la función $\phi_1(\lambda) = f(\boldsymbol{y} + \lambda \boldsymbol{d}_1)$, usando alguno de los métodos de búsqueda unidimensional vistos en 3.1. El punto donde se encuentra el óptimo se almacena en $\boldsymbol{y}$, actualizando de esta manera el valor de este vector, luego se minimiza la función $\phi_2(\lambda) = f(\boldsymbol{y} + \lambda \boldsymbol{d}_2)$, una vez más el punto donde se encuentra al mínimo se almacena en el vector $\boldsymbol{y}$, actualizando de nuevo a este vector. Se repite el

proceso hasta que se minimiza la función $\phi_n(\lambda) = f(\boldsymbol{y} + \lambda \boldsymbol{d}_n)$, el valor de este último mínimo se almacena en el vector $\boldsymbol{x}_{i+1}$.
3) Si $||\boldsymbol{x}_{i+1}-\boldsymbol{x}_i|| < \varepsilon$ (ε se fija de antemano), se detiene la búsqueda, el mejor punto encontrado es $\boldsymbol{x}_{i+1}$; de lo contrario, $i = i + 1$ y se repite el proceso desde el paso (2).

3.2.1.2 El método de Hooke y Jeeves.

Este método alterna un paso del método de las coordenadas cíclicas con una búsqueda sobre la dirección definida por la resta de los dos últimos puntos: $\boldsymbol{x}_i - \boldsymbol{x}_{i-1}$.
0) $i = 0$.
1) Dado el punto $\boldsymbol{x}_i \in R^n$ se definen n vectores: $\boldsymbol{d}_1, \ldots, \boldsymbol{d}_n$ cada uno de ellos paralelo a cada uno de los ejes cartesianos, por ejemplo: $\boldsymbol{d}_1 = (1, 0, \ldots, 0)$, $\boldsymbol{d}_2 = (0, 1, 0, \ldots, 0), \ldots, \boldsymbol{d}_n = (0, ,0, \ldots, 1)$. Estos son los vectores de dirección.
2) Se copia el vector $\boldsymbol{x}_i$ en el vector $\boldsymbol{y}$. Luego se minimiza la función $\phi_1(\lambda) = f(\boldsymbol{y} + \lambda \boldsymbol{d}_1)$, usando alguno de los métodos de búsqueda unidimensional vistos en 3.1. El punto donde se encuentra el óptimo se almacena en $\boldsymbol{y}$, actualizando de esta manera el valor de este vector, luego se minimiza la función $\phi_2(\lambda) = f(\boldsymbol{y} + \lambda \boldsymbol{d}_2)$, una vez más el punto donde se encuentra al mínimo se almacena en el vector $\boldsymbol{y}$, actualizando de nuevo a este vector. Se repite el proceso hasta que se minimiza la función $\phi_n(\lambda) = f(\boldsymbol{y} + \lambda \boldsymbol{d}_n)$, el valor de este último mínimo se almacena en el vector $\boldsymbol{x}_{i+1}$.
3) Si $||\boldsymbol{x}_{i+1}-\boldsymbol{x}_i|| < \varepsilon$ (ε se fija de antemano), se detiene la búsqueda, el mejor punto encontrado es $\boldsymbol{x}_{i+1}$; de lo contrario ir al paso (4).
4) Sea $\boldsymbol{d} = \boldsymbol{x}_{i+1} - \boldsymbol{x}_i$, se hace una búsqueda unidireccional para encontrar el punto donde la función $\phi(\lambda) = f(\boldsymbol{x}_{i+1}, \lambda \boldsymbol{d})$ sea mínima, ese punto se almacena en el vector $\boldsymbol{x}_{i+2}$; $i = i + 2$; ir al paso (2).

3.2.1.3 Método de Rosenbrock.

0) Se fija un intervalo de tolerancia ε. Se escogen n vectores ortogonales entre sí: $\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_n$, por ejemplo: $\boldsymbol{d}_1 = (1, 0, \ldots, 0)$, $\boldsymbol{d}_2 = (0, 1, 0, \ldots, 0), \ldots, \boldsymbol{d}_n = (0, \ldots, 0, 1)$; $i = 0$.
1) Hacer $\boldsymbol{y} = \boldsymbol{x}_i$.
2) Sea λ_1 el escalar que resuelve el problema de minimización de $f(\boldsymbol{y} + \lambda \boldsymbol{d}_1)$, almacénese el punto donde se encuentra ese mínimo en $\boldsymbol{y}$. A continuación se busca el óptimo de la función $f(\boldsymbol{y}+\lambda \boldsymbol{d}_2)$, sea λ_2 el escalar que resuelve ese problema de minimización, almacénese el punto donde se encuentra ese mínimo en $\boldsymbol{y}$, actualizando así ese vector. Repítase este procedimiento hasta que se optimice la función $f(\boldsymbol{y} + \lambda \boldsymbol{d}_n)$, sea λ_n el escalar que resuelve ese problema, el punto donde se encuentra ese mínimo se almacena en $\boldsymbol{x}_{i+1}$.
3) Si $||\boldsymbol{x}_{i+1} - \boldsymbol{x}_i|| < \varepsilon$ parar, si no ir al paso (4). 4) Obtener n nuevas direcciones ortogonales entre sí usando las ecuaciones:

$$\boldsymbol{a}_j = \begin{cases} \boldsymbol{d}_j & \text{si } \lambda_j = 0 \\ \sum_{i=j}^{n} \lambda_j \boldsymbol{d}_j & \text{si } \lambda_j \neq 0 \end{cases}$$

$$\boldsymbol{b}_j = \begin{cases} \boldsymbol{a}_j & j = 1 \\ \boldsymbol{a}_j - \sum_{i=1}^{j-1} (\boldsymbol{a}_j^t \bar{\boldsymbol{d}}_i) \bar{\boldsymbol{d}}_i & j \geq 2 \end{cases}$$

$$\overline{\boldsymbol{d}_j} = \frac{\boldsymbol{b}_j}{\|\boldsymbol{b}_j\|}$$

hacer $\boldsymbol{d}_i = \overline{\boldsymbol{d}_i}$ para $i = 1, \ldots, n$. Ir al paso (2).

3.2.2 Búsqueda multidimensional usando derivadas.

Estos métodos utilizan las primeras y en algunos casos también las segundas derivadas en su búsqueda del óptimo.

3.2.2.1 Método del descenso más pronunciado o método del gradiente.

0) Fijar la máxima tolerancia permitida para el tamaño del gradiente al final de la búsqueda, ε, escoger un punto inicial $\boldsymbol{x}_0$. Hacer $i = 1$.

1) Hacer $\boldsymbol{d} = -\nabla f(\boldsymbol{x}_i)$, si $||\boldsymbol{d}|| < \varepsilon$ parar, si no continuar.

2) Encontrar $\lambda > 0$ tal que minimice $f(\boldsymbol{x}_i + \lambda \boldsymbol{d})$ usando alguno de los métodos de búsqueda unidimensional.

3) Hacer $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \lambda \boldsymbol{d}$.

4) Hacer $i = i + 1$, ir al paso (1).

3.2.2.2 Método de Newton.

Considérese la siguiente aproximación cuadrática de la función f:

$$q(\boldsymbol{x}) = f(\boldsymbol{x}_i) + \nabla f(\boldsymbol{x})^t (\boldsymbol{x} - \boldsymbol{x}_i) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_i)^t \boldsymbol{H}(\boldsymbol{x}_i)(\boldsymbol{x} - \boldsymbol{x}_i) \ .$$

Una condición necesaria para la existencia de un mínimo en la aproximación es que $\nabla q(\boldsymbol{x}) = \boldsymbol{0}$ (y que la matriz hessiana sea definida positiva). Se tiene entonces que en el mínimo: $\nabla f(\boldsymbol{x}_i) + H(\boldsymbol{x}_i)(\boldsymbol{x} - \boldsymbol{x}_i) = \boldsymbol{0}$. Esta última ecuación se puede usar para calcular un punto $\boldsymbol{x}_{i+1}$ que esté más cerca del mínimo de la función f, suponiendo que exista la inversa de la matriz hessiana: $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \boldsymbol{H}(\boldsymbol{x}_i)^{-1} \nabla f(\boldsymbol{x}_i)$. Se puede aplicar reiterativamente esta ecuación haciendo $i = i + 1$ al final de cada iteración, hasta que $||\boldsymbol{x}_{i+1} - \boldsymbol{x}_i|| < \varepsilon$, donde ε es un valor para la condición de terminación prefijado de antemano. Este es el método de Newton

3.2.2.3 Método de Levenberg –Marquardt.

El método de Newton presenta varios problemas para encontrar el óptimo, entre ellos está el que la matriz $\boldsymbol{H}(\boldsymbol{x}_i)$ frecuentemente tiende a ser singular conforme este método se acerca al óptimo y, por tanto, no se puede continuar con su búsqueda al no poderse calcular la inversa de esa matriz. Los métodos del tipo Levenberg-Marquardt resuelven esto haciendo una búsqueda que se encuentra entre el método de Newton y el método del gradiente. Estos métodos consisten en obtener un nuevo punto $\boldsymbol{x}_{i+1}$ resolviendo el sistema

$$[\varepsilon_i \boldsymbol{I} + \boldsymbol{H}(\boldsymbol{x}_i)](\boldsymbol{x}_{i+1} - \boldsymbol{x}_i) = -\nabla f(\boldsymbol{x}_i)$$

Dado un punto $\boldsymbol{x}_i$ y un parámetro $\varepsilon_i > 0$, comprobar que la matriz $\varepsilon_i \boldsymbol{I} + \boldsymbol{H}(\boldsymbol{x}_i)$ es positiva definida tratando de aplicarle la factoración de Cholesky, obteniendo así la matriz $\boldsymbol{L}\boldsymbol{L}^t$ (ver más abajo). Si no es posible, hacer $\varepsilon_i = 4\ \varepsilon_i$ hasta que sí lo sea. Después se resuelve el sistema $\boldsymbol{L}\boldsymbol{L}^t(\boldsymbol{x}_{i+1} - \boldsymbol{x}_i) = -\nabla f(\boldsymbol{x}_i)$ para obtener $\boldsymbol{x}_{i+1}$. Se calcula $f(\boldsymbol{x}_{i+1})$ y la relación

$$R_i = \frac{f(x_i) - f(x_{i+1})}{q(x_i) - q(x_{i+1})}$$

donde q es la aproximación cuadrática mostrada en el punto 3.2.2.2. Entre más se acerque esta relación a uno, menor será el valor de ε_i que se pueda usar. Tomando esto en cuenta, si $R_i < 0.25$, hacer $\varepsilon_{i+1} = 4\varepsilon_i$; si $R_i > 0.75$, hacer $\varepsilon_{i+1} = \varepsilon_i/2$; de otra manera $\varepsilon_{i+1} = \varepsilon_i$. Si $R_i \leq 0$, regresar al punto anterior: $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i$. Luego hacer $i = i+1$ y volver a iterar hasta encontrar un punto cuyo gradiente sea 0.

Factoración de Cholesky. Para factorar una matriz definida positiva y simétrica $\boldsymbol{B}$ en una matriz de la forma $\boldsymbol{LL}^t$, donde $\boldsymbol{L}$ es una matriz triangular inferior:

$$\begin{bmatrix} l_{1,1} & 0 & 0 & \dots & 0 \\ l_{2,1} & l_{2,2} & 0 & \dots & 0 \\ l_{3,1} & l_{3,2} & l_{3,3} & \dots & 0 \\ \dots & & & & \\ l_{n,1} & l_{n,2} & l_{n,3} & \dots & l_{n,n} \end{bmatrix}$$

La matriz $\boldsymbol{LL}^t$ es entonces:

$$\begin{bmatrix} l_{1,1}^2 & & & & \\ l_{2,1}l_{1,1} & (l_{2,1}^2 + l_{2,2}^2) & & \text{SIMETRICA} & \\ l_{3,1}l_{1,1} & (l_{2,1}l_{3,1} + l_{2,2}l_{3,2}) & (l_{3,1}^2 + l_{3,2}^2 + l_{3,3}^2) & & \\ \vdots & \vdots & \vdots & & \\ l_{n,1}l_{1,1} & (l_{2,1}l_{n,1} + l_{2,2}l_{n,2}) & (l_{3,1}l_{n,1} + l_{3,2}l_{n,2} + l_{3,3}l_{n,3}) & \dots & (l_{n,1}^2 + \dots + l_{n,n}^2) \end{bmatrix}$$

Se pueden plantear las ecuaciones:

$$\begin{array}{llll} l_{1,1}^2 = b_{1,1}, & l_{2,1}l_{1,1} = b_{2,1}, & l_{3,1}l_{1,1} = b_{3,1}, & \dots, \; l_{n,1}l_{1,1} = b_{n,1} \\ & l_{2,1}^2 + l_{2,2}^2 = b_{2,2}, & l_{2,1}l_{3,1} + l_{2,2}l_{3,2} = b_{3,2}, & \dots, \; l_{2,1}l_{n,1} + l_{2,2}l_{n,2} = b_{n,2} \\ & & l_{3,1}^2 + l_{3,2}^2 + l_{3,3}^2 = b_{3,3}, & \dots, \; l_{3,1}l_{n,1} + l_{3,2}l_{n,2} + l_{3,3}l_{n,3} = b_{n,3} \\ & & & \vdots \\ & & & l_{n,1}^2 + \dots + l_{n,n}^2 = b_{n,n} \end{array}$$

Las incógnitas $l_{i,j}$ pueden calcularse resolviendo una tras otra las ecuaciones que arriba se muestran. Las ecuaciones son bien definidas para una matriz simétrica y definida positiva $\boldsymbol{B}$. La matriz $\boldsymbol{LL}^t$ será definida positiva si y sólo si $l_{i,i} > 0$ para toda $i = 1, 2, \dots, n$.

3.2.2.4 Un método casinewtoniano: el método de Davidon-Fletcher-Powell.

0) Fijar el escalar $\varepsilon > 0$ para determinar la tolerancia para terminar el método. Escoger un punto inicial $\boldsymbol{x}_1 \in R^n$ y una matriz $\boldsymbol{D}_1$ definida positiva simétrica. Sea $\boldsymbol{y}_1 = \boldsymbol{x}_1$, i= 1;

1)

para j = 1 hasta n – 1

{

si $||\nabla f(\boldsymbol{y}_j)|| < \varepsilon$ **parar**;

si no

{

$\boldsymbol{d}_j = -\boldsymbol{D}_j \nabla f(\boldsymbol{y}_j)$;

encontrar λ_j tal que minimice $f(\boldsymbol{y}_j+\lambda_j d_j)$ para $\lambda_j \geq 0$;

$\boldsymbol{y}_{j+1} = \boldsymbol{y}_j + \lambda_j \boldsymbol{d}_j$;

Actualizar $\boldsymbol{D}$ así:

$$\boldsymbol{D}_{j+1} = \boldsymbol{D}_j + \frac{\boldsymbol{p}_j \boldsymbol{p}_j^t}{\boldsymbol{p}_j^t \boldsymbol{q}_j} - \frac{\boldsymbol{D}_j \boldsymbol{q}_j \boldsymbol{q}_j^t \boldsymbol{D}_j}{\boldsymbol{q}_j^t \boldsymbol{D}_j \boldsymbol{q}_j}$$

donde

$\boldsymbol{p}_j = \lambda_j \boldsymbol{d}_j = \boldsymbol{y}_{j+1} - \boldsymbol{y}_j$; $\boldsymbol{q}_j = \nabla f(\boldsymbol{y}_{j+1}) - \nabla f(\boldsymbol{y}_j)$

}

}

encontrar λ_j tal que minimice $f(\boldsymbol{y}_j+\lambda_j d_j)$ para $\lambda_j \geq 0$;

$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n+\lambda_n \boldsymbol{d}_n$;

2) $i = i$ +1; $\boldsymbol{x}_i = \boldsymbol{y}_{n+1}$; $\boldsymbol{y}_1 = \boldsymbol{y}_{n+1}$; ir a (1).

3.2.2.5 El método de gradiente conjugado de Fletcher y Reeves.

0) Fijar un escalar $\varepsilon > 0$ para determinar cuando parar y escoger un punto inicial. Hacer $\boldsymbol{y}_1 = \boldsymbol{x}_1$, $\boldsymbol{d}_1 = -\nabla f(\boldsymbol{y}_1)$, i = 1.

1)

para j = 1 hasta n-1

{

si $||\nabla f(\boldsymbol{y}_j)|| < \varepsilon$ parar

si no

{

encontrar λ_j tal que minimice $f(\boldsymbol{y}_j+\lambda_j \boldsymbol{d}_j)$ para $\lambda_j \geq 0$;

$\boldsymbol{y}_{j+1} = \boldsymbol{y}_j + \lambda_j \boldsymbol{d}_j$;

Actualizar la dirección: $\boldsymbol{d}_{j+1} = -\nabla f(\boldsymbol{y}_{j+1}+\alpha_j \boldsymbol{d}_j)$, donde

$$\alpha_j = \frac{\left\|\nabla f(\boldsymbol{y}_{j+1})\right\|^2}{\left\|\nabla f(\boldsymbol{y})\right\|^2}$$

}

encontrar λ_j tal que minimice $f(\boldsymbol{y}_j+\lambda_j \boldsymbol{d}_j)$ para $\lambda_j \geq 0$;

$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n+\lambda_n \boldsymbol{d}_n$;

2) $\boldsymbol{x}_{i+1} = \boldsymbol{y}_{n+1}$; $\boldsymbol{y}_1 = \boldsymbol{y}_{n+1}$; $\boldsymbol{d}_1 = -\nabla f(\boldsymbol{y}_1)$; $i = i$ +1; ir a (1).

4 Métodos para resolver problemas no lineales restringidos (problemas de programación no lineal).

4.1 Uso de funciones de penalización.

Estos métodos también son llamados de *función de penalización del exterior*, ya que se caracterizan por iniciar en puntos no factibles e ir acercándose y entrar a la región factible conforme se va aplicando el método. Consisten en usar una función que adiciona un término positivo a la función objetivo cuando es evaluada en un punto que *no se encuentre en la región factible*. Este último término constituye una penalización. A este tipo de función se le llama *función auxiliar*. Se aplica un método de optimización no restringida a la función auxiliar para buscar así el óptimo restringido. La forma típica de la función auxiliar, f_a, es:

$$f_a(\boldsymbol{x}) = f(\boldsymbol{x}) + \mu\, \alpha(\boldsymbol{x}) \qquad \text{(D6)}$$

donde
$f(\boldsymbol{x})$ es la función objetivo,
$\mu \in R^+$ es el parámetro de penalización,

$$\alpha(\boldsymbol{x}) = \sum_{i=1}^{m} \phi[g(\boldsymbol{x})] + \sum_{i=1}^{l} \psi[h(\boldsymbol{x})]$$

donde ϕ y ψ son funciones continuas tales que:
$\phi(y) = 0$ si $y \le 0$ y $f(y) > 0$ si $y > 0$
$\psi(y) = 0$ si $y = 0$ y $\psi(y) > 0$ si $y \ne 0$

El escalar μ debe tener un valor grande para evitar que la búsqueda termine en un punto no factible. Sin embargo, cuando desde el principio de la búsqueda se usan valores grandes para μ, el problema por lo general se convierte en un problema mal condicionado y es difícil encontrar el óptimo restringido. Es por eso que es conveniente resolver el problema con un valor pequeño de μ y luego volver a resolverlo con un valor de μ mayor, tomando como punto inicial el punto obtenido en el anterior proceso de optimización. Este proceso se repite hasta encontrar el óptimo restringido. El siguiente algoritmo muestra este proceso:

0) Fíjese un valor $\varepsilon > 0$ que servirá de tolerancia en la obtención del óptimo. Propóngase un punto $\boldsymbol{x}_1$ de inicio de la búsqueda, un valor para el parámetro de penalización μ_1 y un escalar $\beta > 1$; hacer $i = 1$.
1) Resolver el problema: minimizar $f(\boldsymbol{x}_i) + \mu_i\, a(\boldsymbol{x}_i)$, sujeto a $\boldsymbol{x} \in X$
Sea $\boldsymbol{x}_{i+1}$ una solución óptima.
2) Si $\mu_i\, \alpha(\boldsymbol{x}_{i+1}) < \varepsilon$, parar; si no hacer $\mu_{i+1} = \beta\, \mu_i$, $i = i + 1$ e ir al paso (1).

4.1.1 Un algoritmo que usa una función de Lagrange aumentada: el método de los multiplicadores de Lagrange.

La función es:

$$F_{ALAG}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}) = f(\boldsymbol{x}) + \sum_{i=l+1}^{m+l} \mu_i \max^2\left\{ g_i(\boldsymbol{x}) + \frac{u_i}{2\mu_i}, 0 \right\} - \sum_{i=l+1}^{m+l} \frac{u_i^2}{4\mu_i} + \sum_{i=1}^{l} v_i h_i(\boldsymbol{x}) + \sum_{i=1}^{l} \mu_i h_i^2(\boldsymbol{x})$$

donde por comodidad al aplicar el algoritmo se ha llamado g_{l+1} a la restricción g_1, g_{l+m} a la restricción g_m.
El algoritmo:
0)Seleccionar algunos multiplicadores de Lagrange $\boldsymbol{v}$, $\boldsymbol{u}$ iniciales, así como valores positivos para los parámetros de penalización: $\mu_1, \ldots, \mu_l, \mu_{l+1}, \ldots, \mu_{l+m}$. Defínase la función VIOL($\boldsymbol{x}$) para cualquier $\boldsymbol{x} \in R^n$, VIOL($\boldsymbol{x}$) = max{$|h_i(\boldsymbol{x})|$: 1, . . . , l; max { $g_i(\boldsymbol{x})$, 0: i = l, . . . , m}}. Hacer k = 1 e ir al paso (1)
1) Resolver el problema no restringido para minimizar $F_{ALAG}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v})$ sujeto a $\boldsymbol{x} \in R^n$, y llámese $\boldsymbol{x}_k$ a la solución óptima que se obtenga. Si $|\text{VIOL}(\boldsymbol{x}_k)| < \varepsilon$ (siendo $\varepsilon > 0$ una tolerancia fijada desde el inicio) , entonces parar, $\boldsymbol{x}_k$ es un punto KKT . Si no, si VIOL($\boldsymbol{x}_k$) ≤ ¼ VIOL($\boldsymbol{x}_{k-1}$) ir al paso (2). Si no, si VIOL($\boldsymbol{x}_k$) > ¼ VIOL($\boldsymbol{x}_{k-1}$), entonces para cada restricción i = 1, . . . , l para la cual $|h_i(\boldsymbol{x}_k)| >$ ¼ VIOL($\boldsymbol{x}_{k-1}$) y para cada restricción $i = l$+1, . . . , m+l para la que max{$g_i(\boldsymbol{x})$, 0} > ¼ VIOL($\boldsymbol{x}_{k-1}$) remplazar el correspondiente parámetro de penalización μ_i por $10\mu_i$ y repetir este paso (1).
2) Actualización de los multiplicadores de Lagrange.
Remplazar $\boldsymbol{v}$ por $\boldsymbol{v}_{nuevo}$, donde
$\boldsymbol{v}_{nuevo} = \boldsymbol{v}_i + 2\mu_i h_i(\boldsymbol{x}_k)$ para i = 1, . . . , l
Remplazar $\boldsymbol{u}$ por $\boldsymbol{u}_{nuevo}$ donde
$\boldsymbol{u}_{nuevo} = \boldsymbol{u}_i - \max\{2\mu_i g_i(\boldsymbol{x}_k), -\boldsymbol{u}_i\}$ para $i = l$+1, . . . , l+m .
Hacer $k = k$ +1 e ir al paso (1).

4.2 Métodos de funciones de barrera.
En estos métodos se resuelve el Problema de Barrera, este problema se puede enunciar de la siguiente manera:

Minimizar $\theta(\mu)$
Sujeta a:
$\mu \geq 0$
donde $\theta(\mu) = \inf\{f(\boldsymbol{x}) + \mu B(\boldsymbol{x}): \boldsymbol{g}(\boldsymbol{x}) < \boldsymbol{0}, \boldsymbol{x} \in X\}$

B es una función de barrera no negativa y continua en la región $\{\boldsymbol{x}: \boldsymbol{g}(\boldsymbol{x}) < \boldsymbol{0}\}$ y se aproxima a ∞ cuando se avanza hacia la frontera de la región desde su interior

$$B(\boldsymbol{x}) = \sum_{i=1}^{m} \phi[g_i(\boldsymbol{x})]$$

donde ϕ es una función de una variable que es continua en $\{y: y < 0\}$ y satisface $\phi(y) \geq 0$ si $y < 0$ y $\lim_{y \to 0^-} \phi(y) = \infty$. Una función de barrera típica es de la forma:

$$B(\boldsymbol{x}) = \sum_{i=1}^{m} -\frac{1}{g_i(\boldsymbol{x})}$$

Un algoritmo de aplicación de una función de barrera:
0) Fijar $\varepsilon > 0$, un escalar para probar condición de terminación. Escoger un punto $\boldsymbol{x} \in X$ con $\boldsymbol{g}(\boldsymbol{x}) < 0$. Hacer $\mu > 0$, escoger β en (0, 1), hacer k = 1.

1) Dando como punto inicial $\boldsymbol{x}_k$, resolver el siguiente problema

Minimizar $f(\boldsymbol{x}) + \mu_k B(\boldsymbol{x})$

Sujeto a

$\boldsymbol{g}(\boldsymbol{x}) < \boldsymbol{0}$

$\boldsymbol{x} \in X$

sea $\boldsymbol{x}_{k+1}$ una solución óptima. Ir a (2)

2) Si $\mu_k B(\boldsymbol{x}_{k+1}) < \varepsilon$, parar. De otra forma hacer $\mu_{k+1} = \beta \mu_k$, hacer $k = k + 1$. Ir a (1).

Nótese que en el paso (1) se hace un recordatorio de que se debe verificar en cada paso que el punto se encuentre en región factible. Esto es porque al aplicar métodos que dan saltos discretos en su búsqueda del mínimo se puede salir de la región factible. Si este último fuera el caso hay que retroceder al punto anterior (que está en región factible) y probar con un paso más corto. Nótese también que este método trabaja sólo con restricciones de desigualdad.

4.3 Un método de direcciones factibles: el método de Zoutendijk.

Este método consiste en generar una dirección factible y luego optimizar a lo largo de esa dirección. Considérese el problema de minimizar $f(\boldsymbol{x})$ sujeto a $\boldsymbol{x} \in S$, donde f: $R^n \rightarrow R$ y S es un conjunto no vacío en R^n. Un vector $\boldsymbol{d}$ distinto de cero es llamado una *dirección factible* en $\boldsymbol{x} \in S$ si existe una $\boldsymbol{d} > 0$ tal que $\boldsymbol{x} + \lambda \boldsymbol{d} \in S$ para todo $\lambda \in (0, \delta)$. Es más, $\boldsymbol{d}$ es llamada una *dirección factible mejoradora* en $\boldsymbol{x} \in S$, si existe una $\boldsymbol{d} > 0$ tal que $f(\boldsymbol{x} + \lambda \boldsymbol{d}) < f(\boldsymbol{x})$ y $\boldsymbol{x} + \lambda \boldsymbol{d} \in S$ para todo $\lambda \in (0, \delta)$.

4.3.1 Aplicación del método de Zoutendijk para restricciones lineales.

Aquí las restricciones lineales de desigualdad e igualdad se expresan así: $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$ y $\boldsymbol{E}\boldsymbol{x} = \boldsymbol{e}$, donde $\boldsymbol{A}$ es una matriz de tamaño $m \times n$, $\boldsymbol{E}$ es una matriz de tamaño $l \times n$, $\boldsymbol{b}$ es un vector de dimensión m y $\boldsymbol{e}$ es un vector de dimensión l.

0) Encontrar un punto factible $\boldsymbol{x}_1$ con $\boldsymbol{A}\boldsymbol{x}_1 \leq \boldsymbol{b}$ y $\boldsymbol{E}\boldsymbol{x}_1 = \boldsymbol{e}$. Hacer $k = 1$ e ir al paso (1)

1) Dado $\boldsymbol{x}_k$, suponer que $\boldsymbol{A}^t$ y $\boldsymbol{b}^t$ se descomponen en $(\boldsymbol{A}_1^t, \boldsymbol{A}_2^t)$ y $(\boldsymbol{b}_1^t, \boldsymbol{b}_2^t)$, de manera tal que $\boldsymbol{A}_1\boldsymbol{x}_k = \boldsymbol{b}_1$ y $\boldsymbol{A}_2\boldsymbol{x}_k < \boldsymbol{b}_2$. Sea $\boldsymbol{d}_k$ una solución óptima al problema:

Minimizar $\nabla f(\boldsymbol{x}_k)^t \boldsymbol{d}$

Sujeta a

$\boldsymbol{A}_1\boldsymbol{d} \leq \boldsymbol{0}$

$\boldsymbol{E}\boldsymbol{d} = \boldsymbol{0}$

$-1 \leq d_j \leq 1$ para j = 1, . . . , n

Si $\nabla f(\boldsymbol{x}_k)^t \boldsymbol{d}_k = 0$, parar. $\boldsymbol{x}_k$ es un punto KKT. De otro modo ir al paso (2)

2) Sea λ_k la solución óptima al siguiente problema de búsqueda sobre una línea:

Minimizar $f(\boldsymbol{x}_k + \lambda \boldsymbol{d}_k)$

Sujeta a $0 \leq \lambda \leq \lambda_{\max}$

donde

$$\lambda_{\max} = \begin{cases} \min\{\hat{b}_i / \hat{d}_i : \hat{d}_i > 0\} & \text{si } \hat{\boldsymbol{d}} > \mathbf{0} \\ \infty & \text{si } \hat{\boldsymbol{d}} \leq \mathbf{0} \end{cases}$$

$$\hat{\boldsymbol{b}} = \boldsymbol{b}_2 - \boldsymbol{A}_2 \boldsymbol{x}_k$$

$$\hat{\boldsymbol{d}} = \boldsymbol{A}_2 \boldsymbol{d}_k$$

Hacer $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \lambda_k \boldsymbol{d}_k$. Identificar el Nuevo conjunto de *restricciones activas* (estas son las restricciones de desigualdad para las que se cumple estrictamente la igualdad) en $\boldsymbol{x}_{k+1}$ y actualizar con base en esto $\boldsymbol{A}_1$ y $\boldsymbol{A}_2$. Hacer $k = k + 1$. Ir al paso 1.

4.3.2 Aplicación del método de Zoutendijk para restricciones de desigualdad no lineales.

0) Escoger un punto inicial $\boldsymbol{x}_1$ tal que $g_i(\boldsymbol{x}_1) \leq 0$ para $i = 1, \ldots, m$. Hacer $k = 1$.

1) Sea $I = \{i: g(\boldsymbol{x}_k) = 0\}$ y resolver el siguiente problema:

Minimizar z

Sujeta a $\nabla f(\boldsymbol{x}_k)^t \boldsymbol{d} - z \leq 0$

$\nabla g_i(\boldsymbol{x}_k)^t \boldsymbol{d} - z \leq 0$ para $i \in I$

$-1 \leq d_j \leq 1$ para $j = 1, \ldots, n$

Sea $(z_k, \boldsymbol{d}_k)$ una solución óptima. Si $z_k = 0$, parar. Si $z_k < 0$, ir al paso (2)

2) Sea λ_k una solución óptima al siguiente problema de búsqueda sobre una línea

Minimizar $f(\boldsymbol{x}_k + \lambda \boldsymbol{d}_k)$

Sujeta a $0 \leq \lambda \leq \lambda_{max}$

Donde $\lambda_{max} = \sup\{\lambda: g(\boldsymbol{x}_k + \lambda \boldsymbol{d}_k) \leq 0$ para $i = 1, \ldots, m\}$. Hacer $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \lambda_k \boldsymbol{d}_k$. Hacer $k = k + 1$. Ir al paso (1).

4.3.3 Aplicación del método de Zoutendijk incluyendo restricciones de igualdad no lineales.

La idea es moverse por una dirección tangencial a las restricciones de igualdad y después hacer un movimiento de regreso hacia ella. Esto es moverse por una dirección $\boldsymbol{d}_k$ tal que $\nabla \boldsymbol{h}(\boldsymbol{x}_k)^t \boldsymbol{d}_k = \mathbf{0}$, y después regresar hacia la región factible.

Considérese el problema:

Minimizar f(x)

Sujeta a

$g_i(\boldsymbol{x}) \leq 0$ para $i = 1, \ldots, m$

$h_i(\boldsymbol{x}) = 0$ para $i = 1, \ldots, l$.

Sea $\boldsymbol{x}_k$ un punto factible e $I = \{i: g_i(\boldsymbol{x}_k) = 0\}$. Hay que resolver el siguiente problema lineal:

Minimizar $\nabla f(\boldsymbol{x}_k)\boldsymbol{d}$

Sujeta a:

$\nabla g_i(\boldsymbol{x}_k)^t \boldsymbol{d} \leq 0$ para $i \in I$

$\nabla h_i(\boldsymbol{x}_k)^t \boldsymbol{d} = 0$ para $i = 1, \ldots, l$

La dirección resultante $\boldsymbol{d}_k$ es tangencial a las restricciones de igualdad y algunas restricciones activas de desigualdad. Se hace una búsqueda a lo largo de d_k y luego se hace

un movimiento de regreso hacia la región factible que lleva hacia $\boldsymbol{x}_{k+1}$. Todo el proceso se vuelve a repetir hasta llegar cerca del óptimo.

Bibliografía.

[AC94] H. Adeli, N.T. Cheng. *Augmented lagrangian genetic algorithm for structural optimization*. Journal of Aerospace Engineering, Vol 7, No. 1, pp. 104-118, 1994.

[AC02] U. Aickelin and S. Cayzer. *The Danger Theory and Its Application to Artificial Immune Systems*. In J. Timmis and P.J. Bentley (eds.) Proceedings of the First International Conference on Artificial Immune Systems (ICARIS '02), pp. 141-148, 2002.

[AL01] A.K. Abbas and A.H. Lichtman. *Basic Immunology*. Saunders Company, Philadelphia, London, New York, 2001.

[AS97] M.M. Ali and C. Storey. *Aspiration Based Simulated Annealing Algorithm*. Journal of Global Optimization, 11, 181-191, 1997.

[ATLDD02] M. Ayara, J.Timmis, R. de Lemos, L.N. de Castro, R. Duncan. *Negative Selection: How to Generate Detectors*. In J. Timmis and P.J. Bentley (eds.) Proceedings of the First International Conference on Artificial Immune Systems (ICARIS '02), pp. 89-98, 2002.

[AZGL02] K.P. Anchor, J.B. Zydallis, G.H. Gunsh, and G.B. Lamont. *Extending the Computer Defense Immune System: Network Intrussion Detection with a Multiobjective Evolutionary Programming Approach*. In J. Timmis and P.J. Bentley (eds.) Proceedings of the First International Conference on Artificial Immune Systems (ICARIS '02), pp. 12-21, 2002.

[Bäc96] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, Oxford, 1996.

[Bal1896] J.Mark Baldwin. *A Nnew Factor in Evolution*. The American Naturalist, 30, pp. 441-451, 536-553, June 1896. This paper is reprinted in *Adaptive Individuals in Evolving Populations: Models and Algorithms*, edited by R. K. Belew and M. Mitchell (SFI Studies in the Sciences of Complexity, Proc. Vol. XXVI, Addison-Wesley, Reading, MA, 1996).

[Ber99] H. Bersini. *The Endogenous Double Plasticity of the Immune Network and the Inspiration to be Drawn for the Engineering Artifacts*. In D. Dasgupta (ed.) Artificial Immune System and Their Applications, Springer, Berlin Heidelberg New York, pp. 22-41, 1999.

[Ber02] H. Bersini. *Self-Assertion versus Self-Recognition: A Tribute to Francisco Varela*. In J. Timmis and P.J. Bentley (eds.) Proceedings of the First International Conference on Artificial Immune Systems (ICARIS '02), pp. 154-160, 2002.

[BHS00] C. Bergmann, J. Leo van Hemmen, and L.A. Segel. *Th1 or Th2: How an appropriate T helper response can be made*. Bull. Math. Biol. 63, p.p. 405-430, 2000.

[BMWLES03] P.D. Bardwell, A. Martin, E. Wong, Z. Li, W. Edelmann, and M.D. Scharff. *Cutting Edge: The G-U Mismatch Glycosylase Methyl-CpG Binding Domain 4 is Dispensable for Somatic Hypermutation and Class Switch Recombination*. Journal of Immunology, 170, pp. 1620-1624, 2003.

[BOT00] D. Bradley, C. Ortega-Sanchez and A. Tyrrell. *Embrionics+Immunotronics: A Bio-Inspired Approach to Fault Tolerance*. In proceedings of 2nd NASA/DoD Workshop on Evolvable Hardware. Silicon Valley,USA. July 2000.

[BSS93] M.S. Bazaraa, H.D. Sherali, and C.M. Shetty. *Nonlinear Programing*. John Wiley & Sons, 2nd edition, New York, Chichester, Brisbane, Toronto, Singapore, 1993.

[Bro99] T.A. Brown. *Genomes*. John Wiley & Sons, New York, 1999.

[BT00] D.W. Bradley, A.M. Tyrrell. *Hardware Fault Tolerance: An Immunological Solution*. In Proceedings of IEEE Conference on Systems, Man and Cybernetics. Nashville, USA. Volume 1, pp. 107-112, Octiber 2000

[BT00b] D.W. Bradley and A.M. Tyrrell. *Immunotronics: Hardware Fault Tolerance Inspired by the Immune System*. In proceedings of the 3rd INternational Conference on Evolvable Systems. Lecture Notes in Computer Science, Springer-Verlag. Volume 1801, pp. 11-20, 2000

[BT01] D.W.Bradley, A.M. Tyrrell. *The Architecture for a Hardware Immune System*. In proceedings of 3rd NASA/Dod Workshop on Evolvable Hardware. Long Beach, California, USA, pp. 193-200, july 2001.

[BT02] D.W. Bradley, A.M. Tyrrell. *A Hardware Immune System for Benchmark State Machine Error Detection*. In proceedings of the Congress on Evolutionary Computation 2002 (CEC2002). Honolulu, Hawaii, May 2002.

[CA02] S. Cayzer and U. Aickelin. *On the Effects of Idiotipyc Interactions for Recommendation Communities in Artificial Immune Systems*. In J. Timmis and P.J. Bentley (eds.) Proceedings of the First International Conference on Artificial Immune Systems (ICARIS '02), pp. 154-160, 2002.

[CC02] C.A. Coello and N. Cruz. *An Approach to Solve Multiobjective Optimization Problems Based on an Artificial Immune System*. In J. Timmis and P.J. Bentley (eds.) Proceedings of the First International Conference on Artificial Immune Systems (ICARIS '02), pp. 212-221, 2002.

[Cer04] J. Cervantes Ojeda. *Análisis Computacional de las Ecuaciones de Yang-Baxter (YBE*) y Temas Relacionados*. Tesis de invesigación de maestría del Posgrado en Ciencias e Ingeniería de la Computación. Univesidad Nacional Autónoma de México, 2004.

[CF02] D.L. Chao and S. Forrest. *Information Immune Systems*. In J. Timmis and P.J. Bentley (eds.) Proceedings of the First International Conference on Artificial Immune Systems (ICARIS '02), pp. 132-140, 2002.

[Coe99] Carlos A. Coello Coello. *A Survey of Constraint Handling Techniques used with Evolutionary Algorithms*, Lania-IR-99-05, Laboratorio Nacional de Informática Avan-zada, 1999.

[Coo98] E.L. Cooper, *Innate Immunity*, Vol. 3, pp. 1387-1389, en [DR98].

[Coo00] M. Cook. *Receptor editing (and the evolution of sex).* Immunology Today, Vol. 21, No. 1, pp. 55,56.

[Cru00] N. Cruz. *Uso de Emulaciones del Sistema Inmune para Manejo de Restricciones en Algoritmos Evolutivos*. Tesis de Maestría en Inteligencia Artificial, Universidad Veracruzana, Xalapa, Veracruz, México, 2000.

[CSSW02] M.K. Cabatingan, M.R. Schmidt, R. Sen, and R.T. Woodland. *Naïve B Lymphocytes Undergo Homeostatic Proliferation in Response to B Cell Deficit*. Journal of Immunology, 169. pp. 6795-6805, 2002.

[CSBA] Susan Carlson Skalak, Ron Shonkwiler, Sani Babar, M. Aral, *Annealing a Genetic Algorithm over Constraints*, en
http ://vlead.mech.virginia.edu/publications/shenkpaper/shenkpaper.html.

[CT02] R.O. Canham and A.M. Tyrrell. *A Multilayered Immune System for Hardware Fault Tolerance within an Embryonic Array*. In J. Timmis and P.J. Bentley (eds.) Proceedings of the First International Conference on Artificial Immune Systems (ICARIS '02), pp. 3-11, 2002.

[Dar60] C. Darwin. *The Origin of Species by Means of Natural Selection, or the Preservation of Favored Races in the Struggle for Life*. A.L. Burt, New York, 1860.

[Das99] D.Dasgupta, *Artificial Immune System and Their Applications*, Springer, Berlin, Heidelberg, New York, 1999.

[Das99] D. Dasgupta. *An Overview of Artificial Immune Systems and Their Applications*. In D. Dasgupta (ed.) Artificial Immune System and Their Applications, Springer, Berlin Heidelberg New York, pp. 3-21, 1999.

[DG98] L.Djavadi-Ohanianu and M.E. GoldBerg. *Affinity*. Vol. 1, pp. 43-47, en [DR98].

[DH03] P. Deuflhard and A. Hohmann. *Numerical Analysis in Modern Scientific Computing. An Introduction*. Springer Verlag, 2nd edition, New York Berlin Heidelberg, 2003.

[DJG03] D. Dasgupta, Z. Ji, F. González. *Artificial Immune System (AIS) Research in the Last Five Years*. To appear in the proceedings of the international conference on Evolutionary Computation Conference (CEC), Canberra, Australia, December 8-12, 2003.

[DM02] Dipankar Dasgupta and Nivedita Sumi Majumdar. *Anomaly Detection in Multidimensional Data Using Negative Selection Algorithm*. In the proceedings of the Congress on Evolutionary Computation at WCCI, pp 1039-1044, Hawaii, May 14, 2002.

[DR98] P.J. Delves and I.M. Roitt (editors), *Enciclopedia of Immunology*, 2nd edition, Academic Press, San Diego, London, Boston, New York, Sydney Tokyo, Toronto, 1998.

[DR99] G. Dighiero and N.R. Rose. *Critical self-epitopes are key to the understanding of self-tolerance and autoimmunity*. Immunology Today, Vol. 20, No. 9, pp. 423-427, 1999.

[DS91] K. De Jong and W. Spears. *An Analysis of the Interacting Roles of Population Size and Crossover in Genetic Algorithms*. Computer Science Dept. George Mason University, Faurfax, VA, 1991.

[DT02] L. N. de Castro and J.Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer, London, Berlin, Heildelberg, New York, 2002.

[DT02b] L.N. de Castro, and J. Timmis. *Hierarchy and Convergente of Immune Networks: Basic Ideas and Preliminary Results*. In J. Timmis and P.J. Bentley (eds.) Proceedings of the First International Conference on Artificial Immune Systems (ICARIS '02), pp. 231-240, 2002.

[DT03] L. N. de Castro, & , J. I. Timmis. *Artificial Immune Systems as a Novel Soft Computing Paradigm*. Soft Computing journal, 7(7), July. 2003.

[DV01] L. N. de Castro, & F. J. Von Zuben. *Immune and Neural Network Models: Theoretical and Empirical Comparisons*. International Journal of Computational Intelligence and Applications (IJCIA), 1(3), pp. 239-257. 2001

[DV02] L. N. de Castro, & F. J. Von Zuben. *Learning and Optimization Using the Clonal Selection Principle*. IEEE Transactions on Evolutionary Computation, Special Issue on Artificial Immune Systems, 6(3), pp. 239-251. 2002.

[DV02b] L. N. de Castro, & F. J. Von Zuben. *Automatic Determination of Radial Basis Function: An Immunity-Based Approach*. International Journal of Neural Systems (IJNS), Special Issue on Non-Gradient Learning Techniques, 11(6), pp. 523-535. 2002.

[DV03] L. N. de Castro, & F. J. Von Zuben, (2003). *The Construction of a Boolean Competitive Neural Network Using Ideas From Immunology*. Neurocomputing, 50C, pp. 51-85.

[DYM03] D. Dasgupta, S. Yu and N. S. Majumdar. *MILA - Multilevel Immune Learning Algorithm*. In the proceedings of the Genetic and Evolutionary Computation Conference (GECCO), Chicago, July 12-16, 2003.

[EH89] T.F. Edgar & D.M. Himmelblau. *Opimization of Chemical Processes*. McGraw-Hill, New York St. Louis San Francisco, 1989.

[FB90] G.F. Froment y K.B. Bischoff. *Chemical Reactor Analysis and Design*. John Wiley & Sons, 2ª edición, New York, 1990 pp. 14-20.

[FH00] S. Forrest and S.A. Hofmeyr. *Immunology as information processing*. In Design Principles for the Immune System and Other Distributed Autonomous Systems, edited by L.A. Segel and I. Cohen. Santa Fe Institute Studies in the Sciences of

Complexity. New York: Oxford University Press, 2000.

[FHS97] S. Forrest, S.A.Hofmeyr, A. Somayaji. *Computer immunology*. S. Forrest, S. Hofmeyr, and A. Somayaji. Communications of the ACM , Vol. 40, No. 10, pp. 88-96, 1997.

[FH01] S. Forrest and S. Hofmeyr. *Engineering an immune system*. Graft, 4(5):5-9, 2001.

[FM94] Robert M. French and Adam Messinger. *Genes, Phenes and the Baldwin Effect: Learning and Evolution in a Simulated Population*. In R. Brooks and P. Maes (eds), Artificial Life IV, Cambridge, MA, The MIT Press, 1994, pp. 277-282.

[FMT99] T. Fukuda, K. Mori, and M. Tsukiyama. *Parallel Search for Multi-Modal Function* Optimization with Diversity and Learning of Immune Algorithm. Immune System for Real World Applications. In D. Dasgupta (ed.) Artificial Immune System and Their Applications, Springer, Berlin Heidelberg New York, pp. 210-219, 1999.

[FP90] Floudas, Christodoulos A. and Pardalos, Panos M. *A Collection of Test Problems for Constrained Global Optimization Algorithms*. Springer, Berlin, Heidelberg, New York 1990.

[FPA99] C.A. Floudas, P.M. Pardalos, C.S. Adjiman, W.R. Esposito, Z.H. Gümüs, S.T. Harding, J.L.Kleips, C.A. Meyer, and C.A. Schweiger. *Handbook of Test Problems in Local and Global Optimization*. Kluwer Academic Publishers, Dordrecht, Boston, London, 1999.

[FPP86] Farmer, J.D., Pakard, N.H. & Perelson, A.S. *The Immune System, Adaptation, and Machine Learning*. Physica, 22D, 187-204, 1986.

[FR00] A. Freitas and B. Rocha. *Population biology of Lymphocytes: the Flight for Survival*. Annual Review of Immunology, 18, 83-111, 2000.

[FSJP93] S. Forrest, R.E. Smith, B. Javornik, and A.S. Perelson. *Using genetic algorithms to explore pattern recognition in the immune system*. Evolutionary Computation, 1(3):191-211 (1993).

[GD02] F. González and D. Dasgupta. *Neuro-Immune and Self-Organizing Map Approaches to Anomaly Detection: A Comparison*. In J. Timmis and P.J. Bentley (eds.) Proceedings of the First International Conference on Artificial Immune Systems (ICARIS ‘02), pp. 203-211, 2002.

[GDG03] F. Gonzalez, D. Dasgupta, and J. Gomez. *The effect of binary matching rules in negative selection*. In the proceedings of the Genetic and Evolutionary Computation Conference (GECCO), Chicago, USA, July 12-16, 2003.

[GDK02] Fabio Gonzalez, Dipankar Dasgupta and Robert Kozma. *Combining Negative Selection and Classification Techniques for Anomaly Detection*. In the proceedings of the Congress on Evolutionary Computation, pp 705-710, Hawaii, May 12-17, 2002.

[GG99] A.J.T. George and D.Gray. *Receptor editing during affinity maturation*. Immunology Today, Vol. 20, No. 4, p. 196, 1999.

[GG00] A.J.T. George and D. Gray. *Jumping or walking which is better?*. Immunology Today, Vol. 21, No. 1, p. 56, 2000.

[GGD03] J. Gómez, F. González and D. Dasgupta. *An immuno-fuzzy approach to anomaly detection*. In Proceedings of The IEEE International Conference on Fuzzy Systems, St. Louis, MO, May 2003.

[GH02] A. Gaspar and B. Hirsbrunner. *From Optimization to Learning in Changing Environements: The Pittsburgh Immune Classifier System*. In J. Timmis and P.J. Bentley (eds.) Proceedings of the First International Conference on Artificial Immune Systems (ICARIS '02), pp. 190-199, 2002.

[GKO00] R.A. Goldsby, T.J. Kindt, and B.A. Osborne. *Kuby Immunology*. Freeman and Company, New York, 2000.

[GL98] M. Gaviano and D. Lera. *Test Functions with Variable Attraction Regions for Global Optimization Problems*. Journal of Global Optimization, 13, 207-223, 1998.

[Gol83] D. E. Goldberg, *Computer-Aided Gas Pipeline Operation Using Genetic Algorithms and Rule Learning*, Ph.D. dissertation (Civil Engineering), University of Michigan, Ann Arbor, 1983.

[Gol89] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[HD98] P.S. Hienstra and M.R. Daha, *Opsonization*, Vol. 4, pp. 1885-1888, en [DR98].

[HA97] M.F. Hussain and K.S. Al-Sultan. *A Hybrid Genetic Algorithm for Nonconvex Function Minimization*. Journal of Global Optimization, 11, 313-324, 1997.

[Her04] R.P. Hernández García. *An Evolutionary Algorithm Partially Inspired in the Vertebrates´ Immune System for Performing Numerical Optimization*. In M.H. Hamza (ed.) Proceedings of the IASTED International Conference on Artificial Intelligence and Applications, Innsbruck, Austria, vol. 1, pp. 121-128, February 2004.

[Her04b] R.P. Hernández García. *Avances en el Desarrollo de un Algoritmo para Maximizar la Concentración del Producto Deseado en una Red de CSTRs*. En las memorias del 25° Encuentro Nacional de la Academia Mexicana de Investigación y Docencia en Ingeniería Química. Trabajo SIM 27, Puerto Vallarta, Jalisco, mayo de 2004.

[HF00] S. A. Hofmeyr & S. Forrest. *Architecture for an Artificial Immune System*. S. A Hofmeyr and S. Forrest. Evolutionary Computation 7(1):45-68. 2000.

[HFP95] R.R. Hightower, S. Forrest, A.S. Perelson. *The Evolution of Emergent Organization in Immune System Gene Libraries*. R. Hightower, S. Forrest, and A.S. Perelson. In L.J.

Eshelman (Ed.) Proc. of the Sixth Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Francisco, CA. 1995.

[HK03] R.P. Hernández García y A.F. Kuri Morales. *Using the EGA, a Non-Traditional GA, and Deterministic Ranking for Optimization of Constrained Functions*. En S. Botello, A. Hernández y C. Coello (eds.) , Memorias del Primer Congreso Mexicano de Computación Evolutiva (COMCEV'03), Guanajuato, Guanajuato, pp. 13-25, mayo de 2003.

[HFP96] R. Hightower, S. Forrest & A.S. Perelson. *The Baldwin effect in the immune system: learning by somatic hypermutation*. R. Hightower, S. Forrest, and A.S.
Perelson. In R.K. Belew and M. Mitchell, (eds.), Adaptive Individuals in Evolving Populations, Addison-Wesley, Reading, MA, pp. 159-167, 1996.

[HN99] W. Huyer and A. Neumaier, *Global Optimization by Multilevel CoordinateSearch*, Journal of Global Optimization, 14, 331-355, 1999.

[Hol75] John H. Holland. *Adaptation in Natural and Artificial Systems*. Ann Arbor, University of Michigan Press, Michigan, USA, 1975.

[Hol92] John H. Holland. *Adaptation in Natural and Artificial Systems*. The MIT Press, Cambridge, Massachusetts, 1992.

[HQL94] A.Homaifar, C. X. Qi, S. H. Lai. *Constrained Optimization Via Genetic Algorithms*. vol. 62, no. 4, pp. 242-254, 1994.

[HR02] E. Hart and P. Ross. *Exploiting the analogy between immunology and sparse distributed memories: a system for clustering non-stationary data*. In J. Timmis and P.J. Bentley (eds.) Proceedings of the First International Conference on Artificial Immune Systems (ICARIS '02), pp. 49-58, 2002.

[HTCNK99] J. Hunt, J. Timmis,D. Cooke, M. Neal, and C. King. *The Development of an Artificial Immune System for Real World Applications*. In D. Dasgupta (ed.) Artificial Immune System and Their Applications, Springer, Berlin Heidelberg New York, pp. 157-164, 1999.

[HY96] P. Hajela, J. Yoo. *Constrained genetic search via schema adaptation an immune network solution.* Structural Optimization, 12, pp. 11-15, 1996

[Inm78] Inman, J. *The Antibody Combining Region: Speculations on the Hypothesis of General Multispecificity*. In G. Bell, A. Perelson, G. Pimbley (Eds.), Theoretical Immunology, pp. 243-278, New York, Marcel Dekker, 1978.

[Jer74] N.K. Jerne. *Towards a Network Theory of the Immune System*. Ann. Immunol. (Inst. Pasteur), 125C, pp. 435-441.

[JH94] J. Joines and C. Houck. *On the use of Non-Stationary Penalty Functions to Solve Nonlinear Constrained Optimization Problems with Gas*. en David Fogel, ed., Proceedings of the first IEEE Conference on Evolutionary Computation, pp. 579-584, IEEE Press, Orlando, Florida, 1994.

[Jon00] D. Jones. Book Review of *Handbook of Test Problems in Local and Global Optimization*, by Floudas, Pardalos et al. in Nonconvex Optimization and Its Applications, volume 33, Kluwer Academic Publishers, 1999, Kluwer Academic Publishers, 16: 299-300, 2000.

[Jon01] D. Jones, *A Taxonomy of Global Optimization Methods Based on Response Surfaces*, Journal of Global Optimization, 21, 345-383, 2001.

[KB02] J. Kim and P.J. Bentley. *A Model of Gene Library Evolution in the Dynamic Clonal Selection*. In J. Timmis and P.J. Bentley (eds.) Proceedings of the First International Conference on Artificial Immune Systems (ICARIS '02), pp. 182-189, 2002

[KB02b] J. Kim and P.J. Bentley. *Immune Memory in the Dynamic Clonal Selection Algorithm*. In J. Timmis and P.J. Bentley (eds.) Proceedings of the First International Conference on Artificial Immune Systems (ICARIS '02), pp. 59-67, 2002.

[KG01] A. F. Kuri-Morales, J. Gutiérrez-García. *Penalti Function Methods for Constrained Optimization with Genetic Algorithms: a Statistical Analysis*. en http://cursos.itam.mx/akuri/CONACYT.

[KM99] S. Koziel, Z. Michalewicz. *Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization*. MIT, Evolutionary Computation, vol. 7, no. 1, pp. 19-44, 1999.

[Kur99] A. F. Kuri-Morales. *A Comprehensive Approach to Genetic Algorithms in Optimization and Learning*. Colección de Ciencia de la Computación, IPN, México 1999.

[KV98] A. F. Kuri-Morales, C. Villegas. *A Universal Eclectic Genetic Algorithm for Constrained Optimization*. en Proceedings of Sixth European Congress on Intelligent Techniques and Soft Computing, EUFIT'98, pp. 518-522, Aachen, Germany, Verlag Mainz, September 1998.

[KWV02] J. Kaers, R. Wheeler, and H. Verrelst. *Building a Robust Distributed Artificial Immune System*. In J. Timmis and P.J. Bentley (eds.) Proceedings of the First International Conference on Artificial Immune Systems (ICARIS '02), pp. 124-131, 2002.

[KZT02] R.A. Krohling, Y. Zhou, and A.M. Tyrrell. *Evolving FPGA-based robot controllers using an evolutionary algorithm*. In J. Timmis and P.J. Bentley (eds.) Proceedings of the First International Conference on Artificial Immune Systems (ICARIS '02), pp. 41-46, 2002.

[Lam1809] J.B.P.A.D.M. Lamarck, *Philosophie zoologique, ou exposition des considerations relatives a l'histoire naturelle des animaux*, Paris, Dentu, 1809.

[Lew00] B. Lewin. *Genes*. Oxford University Press, Oxford, 2000

[LAR99] G. W. Litman, M.K. Anderson and J.P. Rast. *Evolution of Antigen Binding Receptors*. Annu. Rev. Immunology, Vol. 17, pp. 109-147.

[LBZMBD00] H. Lodish, A. Berk, S.L. Zipursky, P.Matsudaira, D. Baltimore, and J.E. Darnell. *Molecular Cell Biology*. 4th edition, Freeman an Company, 2000.

[LY00] M.H. Lim and S. Omatu. *Exchange Local Search Policy for the Quadratic Assignment Problem*. Computational Optimization and Applications, Kluwer Academic Press, 15, 249-268, 2000.

[LZ00] D. Liu and X.S. Zhang. *Test Problem Generator by Neural Network for Algorithms that Try Solving Nonlinear Programming Problems Globally*. Journal of Global Optimization, 16, 229-243, 2000.

[MA02] T. Morrison and U. Aickelin. *An Artificial Immune System as a Recommender for Web Sites*. In J. Timmis and P.J. Bentley (eds.) Proceedings of the First International Conference on Artificial Immune Systems (ICARIS '02), pp. 161-169, 2002.

[Mar98] S.J. Martin, *Apoptosis*, Vol 1, pp. 220-227, en [DR98].

[MB02] G. Marwah and L. Bogges. *Artificial Immune Systems for Classifications: Some Issues*. In J. Timmis and P.J. Bentley (eds.) Proceedings of the First International Conference on Artificial Immune Systems (ICARIS '02), pp. 149-153, 2002.

[MDSS00] Z. Michalewicz, K. Deb, M. Schmidt and T. Stidsen, *Test-Case Generator for Nonlinear Parameter Optimization Techniques*, IEEE Transactions on Evolutionary Computation, 4, 3, 197-215.

[Mic95] Z. Michalewicz, *Genetic Algorithms, Numerical Optimization, and Constraints*, en Larry J. Eshelman, ed., Proceedings of the Sixth International Conference on Genetic Algorithms, University of Pittsburgh, pp. 344-350, Morgan Kaufmann Publishers, San Francisco, 1995.

[Mic96] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed., Springer, Berlin, Heidelberg, New York, 1996.

[Mit96] Melanie Mitchell, *An Introduction to Genetic Algorithms*, The MIT Press, Cambridge, MA, 1996.

[Mil00] R.E. Miller. *Optimization*. John Wiley & Sons. New York Chichester Weinheim. 2000

[Nas77] L. Nashelsky. *Introduction to Digital Computer Technology*. 2nd edition, John Wiley & Sons, New York, Santa Barbara, London, 1977.

[NC00] D.L. Nelson, and M.M. Cox. *Lehninger Principles of Biochemistry*. 3rd edition, Worth Publishers, New York, 2000.

[NGCD03] O. Nasaroui, F. González, C. Cardona, and D. Dasgupta. *A Scalable Artificial Immune System Model for Dynamic Unsupervised Learning*. In the proceedings of the Genetic and Evolutionary Computation Conference (GECCO), Chicago, USA, July 12-16, 2003.

[Nea02] M. Neal. *An artificial immune system for continuous analysis of time-varying data*. In J. Timmis and P.J. Bentley (eds.) Proceedings of the First International Conference on Artificial Immune Systems (ICARIS '02), pp. 76-85, 2002.

[Neu01] C. Neuhauser, *Mathematical Models in Population Geneticas*, pp. 153-177, en D.J. Bolding, M.Bishop and C. Cannings (editors), *Handbook of Statistical Genetics*, John Wiley & Sons, Baffins Lane, Chichester, England, 2001.

[OF98] M. Oprea & S. Forrest. *Simulated evolution of antibody gene libraries under pathogen selection*. M. Oprea and S. Forrest. IEEE International Conference on Systems, Man and Cybernetics. 1998.

[OF99] M. Oprea, S. Forrest. *How the immune system generates diversity: Pathogen space coverage with random and evolved antibody libraries*. Genetic and Evolutionary Computation Conference (GECCO) July 1999.

[OGJ01] P.M. Ortigosa, I. García and M. Jelasity. *Reliability and Performance of UEGO, a Clustering-based Global Optimizer*. Journal of Global Optimization, 19, 265-289, 2001.

[Par94] J. Paredis. *Co-evolutionary Constraint Satisfaction*. en Proceedings of the Third Conference on Parallel Problem Solving from Nature, pp. 46-55, Springer Verlag, 1994.

[Per02] A.S. Perelson. *Modeling Viral and Immune System Dynamcs*. Nature Reviews, Immunology, Vol. 2, pp. 28-36, January 2002

[PGL02] M. Pelikan, D.E. Goldberg, and F.G. Lobo. *A Survey of Optimization by Building and Using Probabilistic Models*. Computational Optimization and Applications, 21, 5-20, 2002.

[PHF96] A.S. Perelson, R. Hightower, S. Forrest. *Evolution (and learning) of v-region genes*. A. Perelson, R. Hightower, and S. Forrest. Research in Immunology
Vol. 147, pp. 202-208, 1996.

[PKD02] Shahidul Pramanik, Robert Kozma and Dipankar Dasgupta. *Dynamical Neuro-Representation of an Immune Model and its Application for Data Classification*. In the proceedings of IJCNN at WCCI, pp 130-135, May 12-17, 2002.

[PS93] D. Powell and M. M. Skolnick. *Using Genetic Algorithms in Engineering Design Optimization with Nonlinear Constraints*. en Stephanie Forrest, ed., Proceedeings of the Fifth International Conference on Genetics Algorithms, pp. 424-431, Morgan Kaufmann Publishers, San Mateo, California, 1993.

[PW97] A.S. Perelson and G. Weisbuch. *Immunology for physicists*. Rev. Modern Phys., Oct. 1997

[Rar98] R. L. Rardin. *Optimization in Operations Research*. Prentice-Hall. Upper Saddle River, New Jersey. 1998.

[Raj98] K. Rajewsky. *Affinity Maturation*. Vol. 1, 52-54, en [DR98].

[RPLM89] J.T. Richardson, M.R. Palmer, G. Liepins, and M. Hilliard. *Some guidelines for genetic algorithms with penalty functions*. En J. David Schaffer (edt.) Proceedings of the Third International Conference on Genetic Algorithms, pp. 191-197, Morgan Kaufmann Publishers, 1989.

[RY00] T.P. Runarsson, Xin Yao. *Stochastic Ramking for Constrained Evolutionary Optimization*. IEEE Transactions on Evolutionary Computation, Vol. 4, no. 3, pp.284-294, 2000.

[SB98] A. Sperting and J.A. Bluestone. *Acquired Immune Response*. Vol. 1, pp. 13-15, en [DR98].

[SB99a] L.A. Segel & R.L. Bar-Or. *Immunology Viewed as the Study of an Autonomus Decentralized System*. In D. Dasgupta (ed.) Artificial Immune System and Their Applications, Springer, Berlin Heidelberg New York, pp. 65-88, 1999.

[SB99b] L.A. Segel & R.L. Bar-Or. *On the Role of Feedback in Promoting Conflicting Goals of the Adaptive Immune System*. The Journal of Immunology, 163, p.p. 1342-1349, 1999.

[SC00] M.Schwartz and I.R. Cohen, *Autoimmunity can benefit self-manteinance*, Immunology Today, Vol. 21, No. 6, pp. 265-268.

[SC99] J. Stewart & J. Carneiro. *The Central and the Peripheral Immune Systems: What is the Relationship?* In D. Dasgupta (ed.) Artificial Immune System and Their Applications, Springer, Berlin Heidelberg New York, pp. 48-64, 1999.

[Sch98] R.H. Schwartz. *Anerg., T Cell*, Vol. 1, pp. 109-111, en [DR98].

[Seg01] L.A. Segel. *Diffuse Feedback from a Diffuse Informal Network*. In L.A. Segel and L. Cohen (eds.) Design Principles for the Immune System and other Distributed Autonomous Systems. Oxford University Press, pp. 203-226, 2001.

[Seg 01b] L.A. Segel. *How does the immune system see to it that it is doing a good job*. Graft, 4, 6, p.p. 15-18, 2001.

[Seg01c] L.A. Segel. *Controlling the Immune System: Diffuse Feedback Via a Diffuse Informational Network*. In G. Bock & J. Goode (eds.), Complexity in Biological Information Processing Symposium 239, John Wiley & Sons, Sussex, U.K., p.p. 31-39, 2001.

[Seg01d] L.A. Segel. *How Can Perception of Context Improve the Immune Response?*. In L. Steinman (ed.), Autoimmunity and Emerging Diseases, Center for the Study of Emerging Disease, Jerusalem, 2001.

[Seg02] L.A. Segel. *Some spatio-temporal models in immunology*. Bifurcation and Chaos 12 (2002), 2343-2347

[SFHP97] D. J. Smith, S. Forrest, R. R. Hightower, and A. S. Perelson. *Deriving Shape Space Parameters from Immunological Data*, Journal of Theoretical Biology, Vol. 189, No. 2, 141-150, Nov. 1997.

[SFP98] D. J. Smith, S. Forrest, and A. S. Perelson. *Using lazy evaluation to simulate realistic-size repertoires in models of the immune system*. Bulletin of Mathematical Biology, Vol 60, 647-658, 1998.

[SFP99] D.J. Smith, S. Forrest and A.S. Perelson. *Immunological Memory is Associative*. In D. Dasgupta (ed.) Artificial Immune System and Their Applications, Springer, Berlin Heidelberg New York, pp. 105-112, 1999.

[SFAP99] D.J. Smith, S. Forrest, D.H. Ackley, and A.S. Perelson. *Modeling the Effects of Prior Infection on Vaccine Efficacy*. In D. Dasgupta (ed.) Artificial Immune System and Their Applications, Springer, Berlin Heidelberg New York, pp. 144-152, 1999.

[SHF98] A. Somayaji, S. Hofmeyr, S. Forrest. *Principles of a Computer Immune System*. New Security Paradigms Workshop, pp75-82, ACM, 1997.

[Sin02] S. Singh. *Anomaly detection using negative selection based on the r-contiguous matching rule*. In J. Timmis and P.J. Bentley (eds.) Proceedings of the First International Conference on Artificial Immune Systems (ICARIS '02), pp. 99-106, 2002.

[SS02] S. Sathyanath and F. Sahin. *AISIMAM – An Artificial Immune System Based Intelligent Multi Agent Model and its Application to a Mine Detection Problem*. In J. Timmis and P.J. Bentley (eds.) Proceedings of the First International Conference on Artificial Immune Systems (ICARIS '02), pp. 22-31, 2002.

[SSo02] S.P. Sokolova, L.A. Sokolova. *Immunocomputing for Complex Interval Objects*. In J. Timmis and P.J. Bentley (eds.) Proceedings of the First International Conference on Artificial Immune Systems (ICARIS '02), pp. 222-230, 2002.

[SX93] M. Schoenauer, S. Xanthakis. *Constrained GA Optimization*. en Stephanie Forrest, ed., Proceedeings of the Fifth International Conference on Genetics Algorithms, pp. 573-580, Morgan Kaufmann Publishers, San Mateo, California, 1993.

[TAH03] S.G. Tangye, D.T. Avery and P.D. Hodgkin. *A Division-Linked Mechanism for the Rapid Generation of Ig-Secreting Cells from Human Memory B Cells*. Journal of Immunology, No. 170, pp. 261-269, 2003

[Tar98] D.M. Tarlinton, *Anergy, B Cell*, Vol. 1, pp. 105-108, en [DR98].

[TGGKS02] A.O. Tarakanov, L.B. Goncharova, T.V. Gupalova, S.V. Kvachev, and A.V. Sukhorukov. *Immunocomputing for Bioarrays*. In J. Timmis and P.J. Bentley (eds.) Proceedings of the First International Conference on Artificial Immune Systems (ICARIS '02), pp. 32-40, 2002.

[TS00] D.M. Tarlinton, K.G.C. Smith. *Dissecting affinity maturation: a model explaining selection of antibody-forming cells and memory B cells in the germinal center*. Immunology Today, Vol. 21, No. 9, pp. 436-441, 2000.

[TK02] T.B. Trafalis and S. Kasap. *A novel metaheuristic approach for continuous global optimization*. Journal of Global Optimization, 23, 171-190, 2002.

[Tur96] Peter D. Turney. *Myths and Legends of the Baldwin Effect*. 13th International Conference on Machine Learning, Workshop on Evolutionary Computation and Machine Learning, Bari, Italy, (1996), 135-142.

[VCV02] P.A.Vargas, L.N. de Castro, and F.J. Von Zuben. Artificial Immune Systems as Complex Adaptive Systems. In J. Timmis and P.J. Bentley (eds.) Proceedings of the First International Conference on Artificial Immune Systems (ICARIS '02), pp. 115-123, 2002.

[Vill04] E. Villalobos. *Diseño de un Algoritmo Evolutivo para Optimización Numérica Restringida Inspirado en la Maduración de la Afinidad*. Proyecto de tesis de la Maestría en Ingeniería Eléctrica, Facultad de Ingeniería de la Universidad Nacional Autónoma de México, se proyecta su terminación en 2004.

[Whi94] D. Whitley. *A Genetic Algorithm Tutorial*. Statistics and Computing, no. 4, pp. 65-85, 1994.

[Wil90] B. Wilson. *Systems: Concepts, Methodologies and Applications*. John Wiley & Sons, Chichester, New York, Brisbane, Toronto, Singapore, 1990 (reprinted with corrections March 1992).

[WIU99] Y. Watanabe, A. Ishiguro and Y. Uchikawa. *Decentralized Behavior Arbitration Mechanism for Autonomous Mobile Robot Using Immune Network*. In D. Dasgupta (ed.) Artificial Immune System and Their Applications, Springer, Berlin Heidelberg New York, pp. 187-209, 1999.

[WK02] S.T. Wierzchon and U. Kuzelewska. *Stable Cluster Formation in an Artificial Immune System*. In J. Timmis and P.J. Bentley (eds.) Proceedings of the First International Conference on Artificial Immune Systems (ICARIS '02), pp. 68-75, 2002.

[WT02] A. Watkins and J. Timmis. *Artificial Immune Recognition System (AIRS): Revisions and Refinements*. In J. Timmis and P.J. Bentley (eds.) Proceedings of the First International Conference on Artificial Immune Systems (ICARIS '02), pp. 173-181, 2002.