INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN Y DESARROLLO
DE TECNOLOGÍA DIGITAL

MAESTRÍA EN CIENCIAS EN SISTEMAS DIGITALES

"PLANIFICACIÓN ÓPTIMA DISCRETA PARA PROBLEMAS NP-COMPLETOS MEDIANTE ALGORITMOS INMUNOLÓGICOS ARTIFICIALES"

TESIS

QUE PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS EN SISTEMAS DIGITALES

P R E S E N T A:

ING. FRANCISCO JAVIER DÍAZ DELGADILLO

BAJO LA DIRECCIÓN DE:

DR. OSCAR H. MONTIEL ROSS

DR. ROBERTO SEPÚLVEDA CRUZ

JUNIO DE 2011

TIJUANA, B.C., MÉXICO

SIP-14-BIS

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de Tijuana, B.C. siendo las 12:00 horas del día 16 del mes de junio del 2011 se reunieron los miembros de la Comisión Revisora de Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación de CITEDI para examinar la tesis titulada:

PLANIFICACIÓN ÓPTIMA DISCRETA PARA PROBLEMAS NP-COMPLETOS MEDIANTE ALGORITMOS INMUNOLÓGICOS ARTIFICIALES.

Presentada por el alumno:

DÍAZ	DELGADILLO	FRANCISCO JAVIER
Apellido paterno	Apellido materno	Nombre(s)

Con registro:

B	0	9	1	8	1	8

aspirante de:

MAESTRÍA EN CIENCIAS EN SISTEMAS DIGITALES

Después de intercambiar opiniones, los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores de tesis

DR. ROBERTO SEPÚLVEDA CRUZ

DR. OSCAR HUMBERTO MONTIEL ROSS

DR. JUAN JOSÉ TAPIA ARMENTA

DR. KONSTANTIN STARKOV

DR. MOISÉS SÁNCHEZ ADAME

PRESIDENTE DEL COLEGIO DE PROFESORES

DR. LUIS ARTURO GONZÁLEZ HERNÁNDEZ

S. E. P.
INSTITUTO POLITECNICO NACIONAL
DE INVESTIGACION Y DESARROLLO
DE TECNOLOGIA DIGITAL
DIRECCION

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de Tijuana, B.C. el día 20 del mes junio del año 2011, el (la) que suscribe Francisco Javier Díaz Delgadillo alumno (a) del Programa de Maestría en Ciencias en Sistemas Digitales con número de registro B091818, adscrito al Centro de Investigación y Desarrollo de Tecnología Digital, manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección de Dr. Oscar H. Montiel Ross y Dr. Roberto Sepúlveda Cruz y cede los derechos del trabajo intitulado Planificación Óptima Discreta para Problemas NP-completos mediante Algoritmos Inmunológicos Artificiales, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección Av. Del parque No. 1310, mesa de Otay, Tijuana, B.C., c.p. 22510 México. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Francisco Javier Díaz Delgadillo

Nombre y firma

"The whole of science is nothing more than a refinement of everyday thinking."

- Albert Einstein

Agradecimientos

Extiendo mi más sincera gratitud a mis directores de tesis el Dr. Oscar H. Montiel Ross y el Dr. Roberto Sepúlveda Cruz, por el apoyo y orientación dada durante este periodo de mi vida donde he indagado en la curiosidad científica. De la misma manera agradezco al Instituto Politécnico Nacional y al Centro de Investigación y Desarrollo de Tecnología Digital por aceptarme y formarme como estudiante investigador de esta gran institución y al Consejo Nacional de Ciencia y Tecnología por otorgarme el apoyo económico que me ha permitido perseguir este plan de vida. Doy gracias a mis familiares, amigos y colegas con los cuales he compartido las experiencias que han forjado mi carácter y me han permitido seguir adelante. Finalmente agradezco de una manera muy especial a mis padres Javier y Lourdes por su apoyo incondicional, sus consejos y sobre todo su afecto y comprensión del cual he disfrutado toda mi vida y por brindarme de la motivación para mantenerme enfocado en mis metas. También agradezco a Nataly por ser un ejemplo de la dedicación y la responsabilidad hacia el estudio y hacia los demás.

Índice general

Índice de figuras

Índice de cuadros

Índice de algoritmos

Lista de acrónimos

Acrónimo	Significado
ACO	Ant Colony Optimization
AE	Algoritmos Evolutivos
AG	Algoritmo Genético
AIS	Artificial Immune Systems
SA	Selector Aleatorio
SE	Selector Elitista
TSP	Traveling Salesman Problem

PLANIFICACIÓN ÓPTIMA DISCRETA PARA PROBLEMAS NP-COMPLETOS MEDIANTE ALGORITMOS INMUNOLÓGICOS ARTIFICIALES

Resumen

El presente trabajo de tesis tiene como finalidad presentar un nuevo algoritmo que ayuda en la solución de problemas de optimización combinatoria del orden NP-completo basado en el sistema inmunológico humano, siendo su abstracción los Sistemas Inmunológicos Artificiales. En forma relevante se propone una visión distinta a las actuales tendencias de estos algoritmos al introducir el concepto de inmunización por vacunación y aplicar este en la solución.

Se presentan tres algoritmos denominados generación de vacunas por medio de Selector Aleatorio, por medio de Selector Elitista y de Expansión de elementos vacunados. Se realiza un estudio comparativo de la recopilación de los datos de tres experimentos diseñados con la finalidad de evaluar la calidad de las soluciones, los puntos de convergencia y los tiempos de ejecución de los algoritmos. Se demuestra que los algoritmos propuestos influyen favorablemente en la reducción de computacional y en la calidad de la solución.

Palabras clave: *Sistemas Inmunológicos Artificiales, AIS, problema del agente viajero, TSP, vacunación artificial, optimización combinatoria.*

OPTIMAL DISCRETE PLANNING FOR NP-COMPLETE PROBLEMS BY ARTIFICIAL IMMUNE SYSTEMS

Abstract

The purpose of this thesis work is to present a new algorithm that helps in solving combinatorial optimization problems of the NP-complete order which is based on the human immune system, being its abstraction the Artificial Immune Systems. Relevantly this work proposes a different view to the current trends of these algorithms by introducing the concept of vaccine-induced immunity and the application of this to generate solutions.

We present three algorithms called: generation of vaccines by Random Selector, by Elitist Selector and Expansion of vaccinated elements. We performed a comparative study of the collection of data from three experiments designed with the aim of assessing the quality of the solutions, the points of convergence and execution times of the algorithms. We show that the proposed algorithms favorably influence the reduction of computing time and the improvement of the quality of the solution

Key words: *Artificial Immune Systems, AIS, traveling salesman problem, TSP, artificial vaccination, combinatorial optimization.*

1 Introducción

Día a día el ser humano se ve inmerso en el proceso de optimización, a nivel individuo se efectúa constantemente, de manera consciente o inconscientemente, el proceso de optimización: la adquisición de productos básicos para vivir donde se busca maximizar la cantidad de productos obtenidos y minimizar el costo, la planificación de las vacaciones donde se busca maximizar la diversión y minimizar los costos; el lograr más con el menor esfuerzo. A su vez el proceso de optimización forma parte de nuestro legado tecnológico en beneficio de la humanidad y se denota en nuestros procesos industriales, en el diseño ingenieril, en las ciencias computacionales, en la medicina y en muchas otras áreas.

El afán de optimizar aunque es innato en nosotros no lo hace algo sencillo y ha sido un objetivo constante del ser humano desde sus comienzos. El estudio de la misma ha llevado a numerosas personas a lo largo de la historia a dedicar recursos significativos a la solución de problemas de optimización y por ello se ha convertido en un campo de estudio que ha evolucionado en los últimos años, constituyéndose en uno de los más importantes.

Dentro del campo de la ciencias exactas y de la optimización, los problemas de optimización combinatoria forman parte de una rama de la matemática muy particular, debido que hasta hoy en día son considerados problemas abiertos o sin una solución clara. La solución de estos problemas involucra gran cantidad de recursos computacionales y la optimización de los métodos de resolverlos ahorra los mencionados recursos y a su vez permite la resolución de problemas más complejos con los mismos recursos con los que se cuentan en un momento dado.

Es así que se han desarrollado en los últimos años gran cantidad de algoritmos que se centran en la optimización enfocándose a su vez en evitar emplear una búsqueda exhaustiva

de todas las posibilidades. En estos esfuerzos se ha recurrido hacia la naturaleza como la fuente de inspiración para estos nuevos algoritmos, al observar que la naturaleza ha tenido que enfrentarse a lo largo del tiempo a una constante lucha por optimizar y perfeccionar todos sus procesos.

En este trabajo la fuente de inspiración es el sistema inmunológico humano que da como fruto los sistemas inmunológicos artificiales (AIS por sus siglas en inglés Artificial Immune Systems), en forma relevante se propone una visión distinta a las actuales tendencias de los AIS al introducir el concepto de inmunización por vacunación y aplicar este en un nuevo algoritmo.

Se escogió el problema del agente viajero (TSP, por sus siglas en inglés Traveling Salesman Problem) como caso de estudio dentro de los problemas de combinatoria, debido a ser un problema clásico y que ha sido estudiado ampliamente lo cual nos permite tener una gran cantidad de datos y problemas de prueba con sus respectivas soluciones óptimas a nuestro alcance, lo cual permite llegar a conclusiones mucho más certeras al contar con una medidas para comparación exactas.

Los AIS se han utilizado en una gran variedad de áreas, atacando diversas problemáticas que van desde el modelado del mismo sistema inmunológico natural hasta su aplicación en el campo de la computación en reconocimiento de patrones, clasificación de datos, aprendizaje máquina y optimización. Hoy en día se cuenta con una gran cantidad de aportaciones por parte de la comunidad científica en forma de publicaciones, lo cual indica el interés y dedicación mundial hacia los AIS. Sin embargo aún teniendo esto en mente los AIS no han logrado formar parte activa en el campo industrial en contraste con otras técnicas metaheurísticas como los Algoritmos Evolutivos [8]. Este hecho no ha desmotivado a la comunidad científica y el deseo de explorar este planteamiento, lo cual se puede observar con la *International Conference on Artificial Immune Systems* (ICARIS), que está por celebrar su décima edición en este año 2011[26].

En lo que respecta a optimización, los AIS han tenido numerosas historias de éxito en la solución de problemas tradicionales numéricos y de combinatoria [31, 35, 14], aún cuando los algoritmos desarrollados no son del todo fiel a los paradigmas de inmunología, ya que uno de

los argumentos es que el sistema inmune no tiene una labor de optimización donde se tiene un sólo objetivo, sino múltiples y posiblemente contradictorios donde no necesariamente el sistema inmune obtiene el mejor resultado, sino el mayor beneficio para el organismo bajo las condiciones dadas[24].

Con respecto a la matemática combinatoria y la optimización de la misma hoy en día existen dos grandes ramas que tratan de dar solución: las soluciones exactas y las metaheurísticas.

Algunos de los algoritmos exactos son ramificar y acotar (*branch-and-bound*) y la optimización o programación lineal. Por el lado de las soluciones exactas tenemos el caso de *Concorde TSP* desarrollado por David Applegate, Robert E. Bixby, Vašek Chvátal y William J. Cook que hoy en día es considerado uno de los mejores algoritmos de solución para el TSP [12, 22]. Se hace especial énfasis en este algoritmo debido a que ha demostrado ser el mejor conocido hasta el momento, proporcionando resultados sorprendentes en el manejo de un gran número de elementos [22]. Hasta la fecha el *Concorde TSP* ha podido resolver óptimamente un TSP de 15,112 ciudades que formaba parte de la conocida librería de problemas *"Traveling Salesman Problem Library"* (TSPLIB) [10]. Chvátal dijo en una entrevista que generar una solución para un TSP no es necesariamente el problema, este surge cuando uno debe demostrar que su solución es la óptima [13].

Por el lado de las metaheurísticas tenemos numerosos algoritmos desarrollados como son los algoritmos genéticos [25], colonia de hormigas [20], redes neuronales [2]. Cada uno de ellos proporcionan maneras distintas de solucionar problemas de combinatoria, unos son basados en poblaciones y otros en búsqueda basada en agentes. Los AIS son clasificados como metaheurísticos, por lo cual son comparados con estos.

Retomando a los AIS, existen investigaciones que buscan dar solución al TSP. Sin embargo estos trabajos se apegan a los algoritmos clásicos definidos por los AIS como son la *Selección Clonal* [21] y los *mecanismos de cooperación de las células Th* [35] los cuales han demostrado ser una manera efectiva e interesante de dar solución a los problemas de combinatoria.

1.1. Objetivo de la presente tesis de investigación

Desarrollo e implementación de un novedoso algoritmo computacional inspirado en los Algoritmos Inmunológicos Artificiales y en el concepto de inmunización por vacunación. Con este algoritmo se desarrollará una plataforma de software que presentará solución a problemas NP-Completos.

1.1.1. Objetivos específicos

- Analizar y reseñar el campo de los Sistemas Inmunológicos Artificiales.
- Identificar y analizar los problemas pertinentes a la matemática combinatoria, concretamente el problema del agente viajero.
- Con la temática de los Sistemas Inmunológicos Artificiales proponer un algoritmo que mejore la solución del TSP.
- Planificación, desarrollo y creación de una plataforma experimental que servirá como herramienta en la comparación entre algoritmos de prueba y los propuestos.
- Llevar a cabo un estudio comparativo de los resultados.

1.2. Aportaciones del trabajo

Existen múltiples metodológicas y propuestas en la literatura para solucionar los problemas de optimización combinatoria y en concreto el TSP, sin embargo hasta hoy en día sigue siendo un problema abierto ya que todos ellos sufren de alguna desventaja con respecto al resto. Este trabajo tiene como objetivo la implementación de un nuevo concepto para la solución de problemas de combinatoria tomando como fuente de inspiración el concepto de inmunización por vacunación. La importancia de este trabajo radica en los siguientes puntos:

- Se basa en terminología inspirada a partir del funcionamiento del sistema inmunológico humano.

- Se propone un algoritmo con un esquema pre-proceso y pos-proceso el cual tiene como principal ventaja la flexibilidad en su implementación.
- Aporta mejoras significativas en los valores y tiempos para el procesamiento de algoritmos ya propuestos.

Asimismo se desarrolla y presenta un software que permite explorar la versatilidad del algoritmo propuesto y junto con la interfaz gráfica forma parte de una plataforma experimental para el estudio de problemas de optimización combinatoria tanto en esta instancia como para futuras investigaciones. Se ha realizado el registro de autor de la misma. Adicionalmente se ha escrito un artículo de divulgación con los resultados obtenidos el cual fue aceptado para su publicación en Springer Verlag y en revista internacional.

1.3. Organización de la presente tesis de investigación

Este trabajo de tesis se encuentra organizado en cinco capítulos. El segundo capítulo presenta un marco teórico que tiene como finalidad cubrir la temática de optimización, la biología como fuente de inspiración en la solución de problemas de ingeniería, los algoritmos metaheurísticos, la clasificación de la complejidad computacional y los problemas de combinatoria específicamente el TSP. A su vez dada su importancia se plasma una breve reseña del campo de la inmunología y como ésta logra convertirse en una fuente de inspiración para algoritmos computacionales y se mencionan los algoritmos más importantes provenientes de los AIS.

En el tercer capítulo se discute el concepto de inmunización por vacunación y como este concepto es utilizado como fundamento para el algoritmo propuesto. Se habla también de los parámetros definidos para este algoritmo y la metodología de generación de vacunas artificiales, a su vez se describe a fondo la plataforma de software que implementar dicha propuesta. Posteriormente en el capítulo cuatro se hace un planteamiento de los experimentos así como un análisis y presentación de los resultados obtenidos con la plataforma al desarrollar los experimentos diseñados. Finalmente en el capítulo cinco se presentan las conclusiones, recomendaciones y trabajo a futuro.

2 Marco Teórico

2.1. Optimización

La optimización es una disciplina fundamental en los campos de la ciencia tales como ingeniería, informática e inteligencia artificial. Usualmente el proceso de optimizar recibe connotaciones bastante imprecisas [6] ya que se relaciona con la idea de "*hacer mejor*" las cosas. Con la finalidad de conceptualizar de la manera más apropiada, optimización será definida como sigue: **el proceso de buscar y encontrar la mejor solución posible a un problema de optimización, en un tiempo limitado donde se cumple con todas las restricciones que forman parte de él**.

El optimizar puede incluir un rango de problemas muy amplio que tienen como finalidad la búsqueda de algún óptimo. De esta misma manera existen diferentes formas de nombrar y clasificar los problemas de optimización y de la misma forma las técnicas empleadas pueden variar significativamente de un problema a otro. Por este motivo el generar una única respuesta hacia los problemas de optimización no es posible y también debido a que la complejidad de los problemas depende en gran parte de las funciones objetivo y de las limitaciones a las que están sujetas[34].

Con esto en mente es posible definir la mayoría de los problemas de optimización en la siguiente forma matemática:

Minimizar o Maximizar

$$f_i(\boldsymbol{x}), \;\; (i = 1, 2, ..., M) \;\; x \epsilon \mathbb{R}^n$$

Sujeto a

$$\phi_j(\boldsymbol{x}) = 0, \;\; (j = 1, 2, ..., J),$$

$$\psi_k(\boldsymbol{x}) \leq 0, \;\; (k = 1, 2, ..., K)$$

donde $f_i(\boldsymbol{x})$, $\phi_j(\boldsymbol{x})$ *y* $\psi_k(\boldsymbol{x})$ son funciones del vector de diseño

$$\boldsymbol{x} = (x_1, x_2, ..., x_n)^T$$

- *Variables de diseño:* los componentes x_i del vector $\boldsymbol{x}$ reciben el nombre de variables de diseño o decisión y pueden ser reales continuas, discretas o una mezcla de ambas.
- *Función objetivo:* Representadas por $f_i(\boldsymbol{x})$ donde $i = 1, 2, ..., M$ son llamadas funciones objetivo o de costo ya que representa la cantidad a optimizar ya sea por maximizado o minimizado. En el caso que $M = 1$ solo existe un objetivo y en $M > 1$ se dice que es un problema de optimización multiobjetivo. Algunos problemas son $M = 0$ donde no existe objetivo y existen solo restricciones. En este caso se dice que el *problema es posible* ya que cualquier solución que cumpla con las restricciones es óptimo.
- *Conjunto de limitaciones*: Las igualdades representadas por $\phi_j(\boldsymbol{x})$ y las desigualdades por $\psi_k(\boldsymbol{x})$ son llamadas *restricciones.* Su finalidad es restringir o limitar los valores a los que pueden asignarse a las variables. Las restricciones pueden llegar a ser complejas y estas excluir soluciones que serían consideradas óptimas. De esta forma un punto o valor se le denomina *viable* si satisface todas las restricciones.
- *Espacio*: La extensión de las variables de diseño recibe el nombre de e*spacio de diseño o de búsqueda* $\mathbb{R}^n$, y el espacio formado por las funciones objetivo es llamado el *espacio de solución o de respuesta.*

2.1.1. Tipos de problemas de optimización

Los problemas de optimización pueden ser divididos y clasificados de diversas maneras. Algunos autores presentan una propuesta donde se generan dos grandes categorías [3, 4]: aquéllos

en donde la solución esta codificada mediante valores reales y aquéllos cuyas soluciones están codificadas en valores enteros. Dentro de la segunda categoría encontramos los *problemas de combinatoria.*

Sin embargo un desglose mucho más completo se puede apreciar en la Figura 2.1[34]. Todo problema de optimización forzosamente debe de cumplir con todas estas características, donde la clasificación de cada una de ellas es exclusiva.

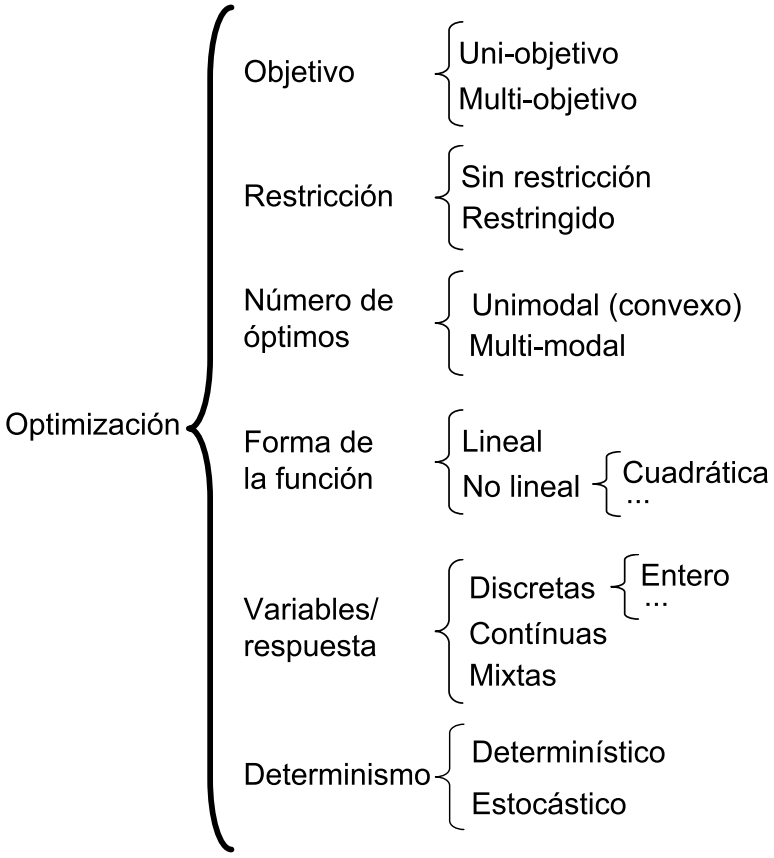


Figura 2.1: Clasificación de los problemas de optimización

- **Objetivo:** Un problema de optimización es clasificado como uni-objetivo si la cantidad a optimizar es expresada utilizando únicamente una sola función objetivo y en el caso contrario se le denomina multi-objetivo.
- **Restricción:** Un problema de optimización se puede clasificar de acuerdo al número de restricciones $J + K$. Si no existe una restricción $J = K = 0$, entonces se dice que es un problema de optimización sin restricciones. Cuando $K = 0$ y $J \geq 1$ se le conoce

como un problema de igualdad con restricciones. Cuando $J = 0$ y $K \geq 1$, se le conoce como un problema de desigualdad con restricciones.

- **Número de óptimos:** Se dice que el problema es *unimodal* si claramente existe una sola solución para éste. Entonces, la optimización podría ser fácilmente realizada mediante el método adecuado para el tipo de problema, y esto se debe a que el mínimo local es también el mínimo global. En contraste si existe más de un solo óptimo el problema es clasificado *multimodal*.
- **Forma de la función:** La función objetivo puede presentar un comportamiento lineal o no lineal; esto ha ocasionado que se presenten dos técnicas de programación: la programación lineal, que consiste en buscar un máximo o un mínimo de una función objetivo lineal sujeta a restricciones lineales; la programación no lineal se presenta en la mayoría de los problemas reales, cuando se presentan problemas en la minimización o maximización de una función objetivo que no se encuentra sujeta a ninguna restricción, o cuando se presentan problemas de una clase importante de minimización, que es la función objetivo cuadrática sujeta a restricciones no lineales. Los problemas de optimización no lineales se clasifican de acuerdo a su complejidad en la definición de las funciones.
- **Variables/Respuesta:** Esta clasificación concierne en la distinción entre un mapa de búsqueda del problema de discreto o continuo. Si las variables de diseño son discretas, entonces la optimización se le denomina como discreta. También se le ha denominado a la optimización discreta como optimización combinatoria, relacionada directamente con teoría de grafos y enrutamiento, como el problema del agente viajero, el árbol de expansión mínima, el problema de las rutas de los vehículos, horarios y el problema de la mochila. Si las variables de diseño son continuas, entonces el problema de optimización es continuo.
- **Determinismo:** Todos los problemas de optimización son deterministas si para un conjunto de variables de diseño, los valores de todas las funciones objetivo y todas las funciones que representan las restricciones se pueden determinar con exactitud. En realidad, solo podemos conocer algunos parámetros con cierta incertidumbre. Cuando

existe esta incertidumbre y ruido (mismo que puede representar la inexactitud del valor) en las variables de diseño, las funciones objetivos y las funciones de restricciones, entonces la optimización se convierte en un problema de optimización estocástico, o en un problema de optimización robusta con ruido.

2.1.2. Algoritmos de optimización

Los algoritmos de optimización puede ser clasificados en dos grandes categorías. Algoritmos determinísticos y algoritmos estocásticos. En la Figura 2.2[34] se puede observar esta clasificación.

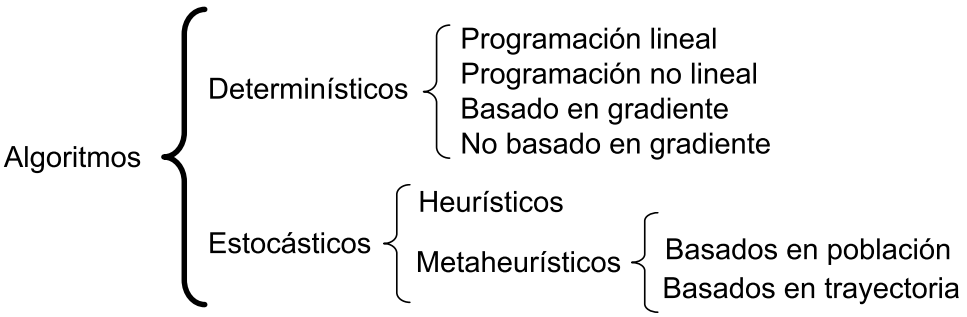


Figura 2.2: Clasificación de los algoritmos de optimización

Los algoritmos determinísticos siguen un riguroso proceso y su trayectoria y valores de tanto las variables y las funciones son repetitivas. Un ejemplo de algoritmo determinístico es un algoritmo de escalada de colina, que dado el mismo punto de comienzo reproducirá los mismos resultados independientemente de que se ejecute el día de hoy o dentro de un año. Y por otro lado tenemos el caso de los algoritmos estocásticos los cuales involucran de alguna manera el concepto de lo aleatorio. Un ejemplo de algoritmos estocásticos son los Algoritmos Genéticos los cuales generan diferentes poblaciones de individuos y de soluciones viables cada vez que este es ejecutado, y aún cuando el resultado puede ser el mismo el camino o trayectoria que este siguió no es el mismo y difícil de repetir. A su vez existen algoritmos mixtos entre lo determinístico y lo estocástico en el sentido de que puede ser

un algoritmo determinístico, pero donde sus parámetros iniciales sean aleatorios, como por ejemplo un algoritmo de escalada de colina con punto inicial aleatorio.

La gran mayoría de los algoritmos clásicos son determinísticos. Por ejemplo aquellos que son basados en una función o modelo matemático se denominan programación lineal (por ejemplo el algoritmo Simplex de Dantzig) o son basados en cálculo de gradientes (por ejemplo Newton-Raphson o método de la gradiente descendiente) e inclusive existen aquellos que trabajan primordialmente con los valores de las funciones (por ejemplo búsqueda de patrones Hooke-Jeeves)[34].

Con respecto a los algoritmos estocásticos se dividen en dos tipos aunque la diferencia entre ellos es pequeña: heurísticos y metaheurísticos. Se define como *heurístico* como "encontrar" o "descubrir por prueba y error". Su principal ventaja es su capacidad de encontrar soluciones aceptables a problemas de optimización difíciles en un tiempo razonable. Sin embargo su principal desventaja es que no garantizan que la solución proporcionada sea la óptima, de hecho la gran mayoría del tiempo lo que realmente entregan es un subóptimo. Se dice que estos algoritmos funcionan la mayoría del tiempo, pero no para todos los casos[34].

2.1.3. Metaheurísticas

Desarrollos posteriores sobre los algoritmos heurísticos son denominados *metaheurísticos*. La palabra *meta* recibe la connotación de "ir más allá" o "nivel superior" y es atribuido así debido a que estos algoritmos generalmente tienen un mejor desempeño que los heurísticos. Adicionalmente una de las características principales de las metaheurísticas es la utilización de lo aleatorio y la búsqueda local. Recientemente existe una tendencia a denominar como metaheurísticos a todos los algoritmos estocásticos que incluyan algún tipo de aleatoriedad y búsqueda local.

La principal ventaja de esta estrategia es que al proveer un algoritmo con algún elemento aleatorio este ahora tiene la capacidad de moverse del plano de la búsqueda local hacia una escala global en los problemas. Por este motivo casi todos los algoritmos metaheurísticos tienen como finalidad la optimización global.

Aún siendo una tema de interés en la comunidad científica a nivel mundial, las metaheurísticas sufren de no estar concretadas del todo en su definición. Sin embargo numerosos autores se han dado a la tarea de intentar proporcionar una definición que capture la esencia no sólo de lo que se ha logrado hacer con las metaheurísticas sino de sus posibilidades futuras. A continuación se presentan dos definiciones formales por dos distintos autores las cuales han destacado entre muchas otras:

- *"Las metaheurísticas son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son efectivos. Las metaheurísticas proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los procedimientos estadísticos"[15].*
- *"Una metaheurística es un proceso iterativo maestro que guía y modifica las operaciones de una heurística subordinada para producir eficientemente soluciones de alta calidad. Las metaheurísticas pueden manipular una única solución completa (o incompleta) o una colección de soluciones en cada iteración. La heurística subordinada puede ser un procedimiento de alto (o bajo) nivel, una búsqueda local, o un método constructivo"[27].*

2.1.4. Biología como fuente de inspiración de las metaheurísticas

Desde su comienzo la humanidad ha observado la naturaleza con gran asombro, este asombro no sólo proviene del hecho que de ella obtiene todo lo necesario para sobrevivir sino que a su vez al observar al mundo natural el ser humano también ha ideado teorías de cómo es que la misma funciona.

En la actualidad el estudio de la naturaleza ha tomado nuevas connotaciones, de ella surge la inspiración para resolver problemas presentes en el ámbito de la ciencia, la tecnología y de la ingeniería. En el área de optimización podemos hablar de la genética, el comportamiento de los insectos, de fenómenos físicos e inclusive de la misma estructura cerebral humana.

Los motivos por los cuales estos temas en particular son importantes se debe a numerosos factores, pero su principal característica es que la naturaleza siempre tiene una manera

de solucionar hasta las problemáticas más adversas en las que se encuentre, y esta es una característica que los sistemas artificiales quisiéramos que tuvieran.

Como ejemplo de sistemas artificiales tenemos precisamente a los algoritmos computacionales empleados en la solución de problemas, y en concreto en los algoritmos metaheurísticos, los cuales han mostrado que una de las temáticas más efectivas cuando se busca solución a las problemáticas es la naturaleza. No es ninguna coincidencia que gran parte de los algoritmos metaheurísticos desarrollados tienen como fundamento una inspiración biológica o natural.

2.1.5. Ejemplos de algoritmos metaheurísticos

En esta sección enlistaremos brevemente algunos de los algoritmos metaheurísticos más importantes en la actualidad.

Algoritmos Evolutivos: Los Algoritmos Evolutivos (AE) son algoritmos metaheurísticos basados en población que utilizan mecanismos inspirados en la biología como lo son la mutación, el cruzamiento, la selección natural y la sobrevivencia del más apto con la finalidad de refinar un conjunto de posibles soluciones de una manera iterativa.

Su ventaja principal comparado con otros métodos de optimización es su esquema en forma de "caja negra" donde solamente se hacen pocas suposiciones acerca de las funciones objetivo subyacentes al problema.

En el Algoritmo 2.1 se presenta el algoritmo clásico de los AE[32]. Como es de esperarse en los algoritmos metaheurísticos este consiste en un proceso iterativo, el cual toma como entrada una función de comparación de la población y una tamaño de población a ser generada, después se procede a crear la población y comenzar en un proceso de selección, cruzamiento y reproducción en base a su aptitud hasta que un criterio de paro sea alcanzado.

Algoritmo 2.1 Algoritmo clásico de un Algoritmo Evolutivo.

Entrada: cmp_F: la función comparadora la cual es usada para comparar la utilidad de dos posibles candidatos a solución.
ps: el tamaño de población

Dato: t: el contador de las generaciones
Pop: la población
Cruce: el repertorio de cruzamiento
v: la función de aptitud resultante del proceso de asignación de aptitud

Salida: X^*: El conjunto de elementos encontrados

1. **comenzar**
   a) $t \leftarrow 0$
   b) **while** !criterioDeParo() **do**:
      1) $v \leftarrow$assignarAptitud(*Pop*,cmp_F)
      2) *Cruce* $\leftarrow$seleccionar(*Pop*,v,ps)
      3) $t \leftarrow t + 1$
      4) *Pop* $\leftarrow$ reproducorPop(*Cruce*)
   c) **regresar** extraerFenotipos(extraerConjuntoOptimo(*Pop*))
2. **terminar**

Algoritmos Genéticos: Los Algoritmos Genéticos (AG) son una subclase de los algoritmos evolutivos donde los elementos del espacio de búsqueda $\mathbb{G}$ son cadenas de caracteres binarias ($\mathbb{G} = \mathbb{B}^*$) o arreglos de otro tipo de elementos. Su inspiración biológica a su vez es similar ya que se inspira también en la mutación, el cruzamiento, la selección natural y la sobrevivencia del más apto y es un proceso iterativo. El algoritmo correspondiente es similar al Algoritmo 2.1 con las consideraciones previas.

Optimización por Colonia de Hormigas: El algoritmo de Optimización por Colonia de Hormigas (Ant Colony Optmization - ACO) se encuentra basado en la metáfora de como las hormigas buscan la comida.

Algoritmo 2.2 Algoritmo clásico de Optimización por medio de Colonia de Hormigas (ACO).

Entrada: h: número de hormigas
n: número de elementos de combinatoria
α: parámetro correspondiente al rastro
β: parámetro correspondiente a la heurística

Dato: i: el contador de las iteraciones
P_{ij}: Matriz de probabilidades
τ_{ij}: Rastro de feromona
η_{ij}: Matriz del inverso de la distancia

Salida: S: Mejor solución

1. **comenzar**
2. inicializar() //coeficientes de evaporación, matrices de distancias
3. **for** $c1 = 1$ **to** i
   a) **for** $c2 = 1$ **to** h
      1) **for** $c3 = 1$ **to** n
         a' $s \leftarrow$construirSolucion($c3$)
         b' $P_{ij}(c3) = \begin{cases} \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum \tau_{ih}^{\alpha}\eta_{ih}^{\beta}} \\ 0 \end{cases}$
      2) **end for**
   b) **end for**
4. Guardar mejor solución s en S
5. Actualizar rastro τ_{ij}
6. **end for**
7. **regresar** S
8. **terminar**

Para lograr esto, la hormiga sale de su hormiguero y comienza a desplazarse de forma aleatoria en diferentes direcciones. Mientras la hormiga se desplaza esta comienza a dejar un rastro químico llamado feromona. Entonces, cuando las hormigas han encontrado comida, esta es capaz de regresar al hormiguero al seguir el rastro de feromona. Al hacer esto, una nueva capa de feromona es agregada y de esta forma otras hormigas son capaces de seguir este rastro dada cierta probabilidad de elegirla, hasta que eventualmente el camino llega a tener

una densidad o concentración de feromona muy alta. Con el tiempo la feromona se evapora al no tener hormigas adicionales cruzando por el camino (por ejemplo, cuando la comida se acaba). A grandes rasgos este comportamiento es el que se describe el algoritmo ACO, y se encuentra caracterizado por el rastro de feromona y la probabilidad que la misma genera.

En la Algoritmo 2.2 se presenta el algoritmo clásico de ACO[32].

Optimización por Temple simulado: El algoritmo de Temple Simulado (Simulated Annealing - SA) es un algoritmo de optimización global originalmente desarrollado para ser aplicado a problemas de optimización combinatoria y numérica, es clasificado como un método de optimización que puede ser aplicado a búsquedas arbitrarias y espacios de problemas. Sólo requiere de un único individuo inicial como punto de partida y una operación de búsqueda. Se basa en el principio del templado que consiste en un tratamiento de material con la finalidad de alterar sus propiedades tal como su dureza y usualmente requiere un proceso en el cual se eleva el material a altas temperaturas, garantizando que la energía almacenada en las partículas del material, por ejemplo los cristales de metal, sea utilizada y estas sufran un cambio en su estructura. Después al dejar que el material se enfríe de manera lenta la estructura es transformada al alcanzar un punto de equilibrio.

En este proceso se basa el algoritmo. En la Algoritmo 2.3 se presenta el algoritmo en forma de pseudo-código para el Temple Simulado[32].

Algoritmo 2.3 Algoritmo clásico de Temple Simulado (SA).

Entrada: f: la función objetivo a ser minimizada

Dato: p_{nueva}: el individuo recién generado
p_{act}: el individuo que está siendo explorado
p^*: el individuo mejor individuo encontrado hasta el momento
T: la temperatura del sistema que ha decrecido con el tiempo
t: índice que representa al tiempo (iteraciones)
$\triangle E$: la diferencia de energía entre x_{nueva} y x_{act}

Salida: X^*: El conjunto de elementos encontrados

1. **comenzar**
   - *a)* $p_{nueva}.g \leftarrow$crear() //implícitamente: $p_{nueva}.x \leftarrow$gpm($p_{nueva}.g$)
   - *b)* $p_{act} \leftarrow p_{nueva}$
   - *c)* $p^* \leftarrow p_{nueva}$
   - *d)* $t \leftarrow 0$
   - *e)* **while** !criterioDeParo() **do**:
     1) $\triangle E \leftarrow f(p_{nueva}.x) - f(p_{act}.x)$
     2) **if** $\triangle E \leq 0$ **then**
        - *a′* $p_{act} \leftarrow p_{nueva}$
        - *b′* **if** $f(p_{act}.x) < f(p^*.x)$ **then** $p^* \leftarrow p_{act}$
     3) **else**
        - *a′* $T \leftarrow$obtenerTemperatura(t)
        - *b′* **if** $random_u() < e^{-\frac{\triangle E}{k_B T}}$ **then** $p_{act} \leftarrow p_{new}$
     4) **end if**
     5) $p_{new.g\leftarrow}$ mutar($p_{act}.g$) //implícitamente : $p_{nueva}.x \leftarrow$gpm($p_{nueva}.g$)
     6) $t \leftarrow t + 1$
   - *f)* **regresar** $p^*.x$
2. **terminar**

Sistemas Inmunológicos Artificiales: Los Sistemas Inmunológicos Artificiales (Artificial Immune Systems - AIS) es un conjunto de metodologías y algoritmos de optimización que han sido desarrollados con la inspiración biológica del sistema inmune humano. Siendo esta la temática fundamental de este trabajo de tesis, en la sección 2.6 titulada “Sistemas Inmunológicos Artificiales” se detalla este paradigma.

2.2. Matemática Discreta

Las matemáticas discretas son una rama de las ciencias exactas en la cual el punto focal de estudio son las estructuras matemáticas que son fundamentalmente representada por enteros, grafos y estados lógicos. Sus elementos no varían suavemente ya que son valores distintivos y separados. Una de sus principales características es que pueden ser enumerados por números enteros.

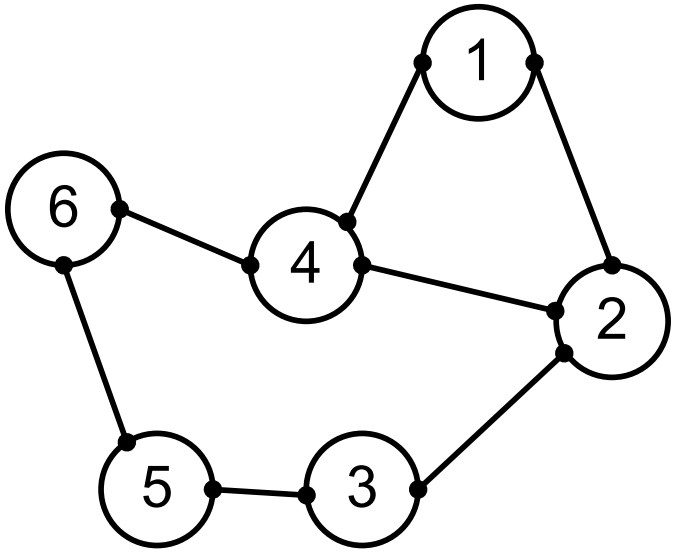


Figura 2.3: Ejemplo de un grafo.

Grafos como el presentado en la Figura 2.3 son algunos de los objetos de estudio de las matemáticas discretas por tener propiedades muy interesantes, por su gran cantidad de aplicaciones en el mundo real y por su utilidad e importancia en la implementación y desarrollo de algoritmos computacionales.

2.2.1. Grafos

Definimos a un grafo G como un conjunto E, denotado $E(G)$, de pares no ordenados de elementos distintos y otro conjunto de elementos V, denotado $V(G)$. V es el conjunto de vértices del grafo y el conjunto E es el de aristas del grafo[30].

$$G = (V, E)$$

$$V = v_1, v_2, ..., v_n$$

$$E = v_i v_j, v_n v_m, ...$$

Dos vértices v_i, v_j son adyacentes si son los extremos de una arista, es decir, si el par de vértices V es un elemento de E. A continuación se enlistan algunos tipos de grafos.

- *Multigrafo:* es un grafo con varias aristas entre dos vértices.
- *Pseudografo:* tiene aristas cuyos extremos coinciden (origen y fin en el mismo vértice), tales aristas se denominan lazos.
- *Digrafo (grafo dirigido):* A cada arista se le asigna un orden en sus extremos, al dibujarse se indica con una flecha. Los pares que forman los elementos de E están ordenados.

A continuación se presentan algunos conceptos relacionados con los grafos:

- *Camino o ruta:* En un grafo G es una sucesión finita de vértices y aristas alternos, donde cada arista tiene por extremos los vértices adyacentes.
  - $(v_0, v_0v_1, v_1, v_1v_2, ..., v_{n-1}, v_{n-1}v_n, v_n)$
  - A v_0 y v_n se les denomina extremos del camino.
- *Longitud del camino:* Es el número de aristas que contiene.
- *Camino cerrado:* Los extremos coinciden, $v_0 = v_n$. En un grafo (no un multigrafo), un camino puede expresarse por la sucesión de vértices:
  - $(v_0, v_1, ..., v_{n-1}, v_n)$
- *Camino simple:* En la sucesión de vértices no hay ninguno repetido.
- *Ciclo:* Es un camino cerrado donde el primero y último vértice son el mismo (camino simple cerrado). En un multigrafo se considera ciclo a aquellos caminos cerrados que no repiten aristas.
- *Circuito:* Es un camino cerrado que no repite aristas.
- *Camino Hamiltoniano:* Es un camino simple que contiene todos los vértices del grafo sin repetir ninguno.

- *Ciclo Hamiltoniano:* Es un camino Hamiltoniano cerrado.
- *Grafo Hamiltoniano:* Es aquel que contiene un ciclo Hamiltoniano.
  - Todo grafo completo contiene un ciclo Hamiltoniano.
  - Sea $G = (V, E)$ un grafo tal que el número de vértices sea mayor o igual a tres ($V \geq 3$), si G es hamiltoniano, para cada subconjunto $\geq U$ de V el subgrafo de G cuyos vértices son $V - U$ y sus aristas son todas las de G que tienen extremos en $V - U$, tienen a lo más U componentes.

2.2.2. Matemática combinatoria

La matemática combinatoria, que constituye el estudio de las permutaciones, combinaciones y particiones, se encuentra enfocado con determinar el número de posibilidades lógicas de algún evento sin la necesidad de identificar cada uno de los posibles casos.

2.2.2.1. Factorial

El factorial $n!$ de un número $n\epsilon\mathbb{N}_0$ es el producto de n y todos los números naturales menores a él. Es una especialización de la función Gamma para números enteros positivos y se describe matemáticamente de la siguiente manera:

$$n! = \prod_{i=1}^{n} i \text{ donde } 0! = 1$$

En matemática combinatoria, usualmente se busca saber en cuantas maneras se puede ordenar $n\epsilon\mathbb{N}$ elementos de un conjunto Ω con $M = |\Omega| \geq n$ elementos. Para esto se pueden hacer dos distinciones en los resultados que se pueden obtener:

- *Combinaciones*: el orden de los elementos en el arreglo no juega ningún papel. El número de posibles combinaciones $C(M, n)$ de $n\epsilon\mathbb{N}$ elementos de un conjunto Ω con $M = |\Omega| \geq n$ es $\begin{pmatrix} M \\ n \end{pmatrix} + \begin{pmatrix} M \\ n-1 \end{pmatrix}$. Si existe repetición entonces tenemos $C(M + n - 1, n - 1)$.

- *Permutaciones*: el orden de los elementos en el arreglo es importante. El número de posibles permutaciones $Perm(M, n)$ de $n \epsilon \mathbb{N}$ elementos de un conjunto Ω con $M = |\Omega| \geq n$ es $\left(\frac{M!}{(M-N)!}\right)$. Si existe repetición entonces tenemos M^n.

De esta manera (a, b, c) y (c, b, a) por ejemplo, denotan la misma combinación pero distintas permutaciones de elementos $\{a, b, c\}$. A si mismo se puede hace una distinción entre los arreglos donde cada elemento de Ω sólo puede ocurrir cuando mucho una vez (sin repetición) y arreglos donde el mismo elemento puede ocurrir múltiples veces (con repetición)[32].

2.3. Complejidad algorítmica

Del punto de vista matemático los problemas puede ser clasificados según la dificultad en encontrar una solución a los mismos por medio de una computadora. De esta manera se han definido diversas clases de problemas de los cuales destacan las clases *P*, *NP*, *NP-completo* y *NP-duro*. A continuación se procede a definir cada una de ellas.

Si el tiempo de ejecución de un algoritmo necesario para resolver un problema puede relacionar el tamaño de entrada con una fórmula polinómica entonces se dice que el problema es solucionable en un tiempo polinómico. Los problemas para los que existe un algoritmo polinómico se le denominan *P*. A si mismo se considera que los problemas tipo *P* pueden ser resueltos en un tiempo de ejecución razonable con respecto a la tecnología de hardware y de software disponible.

En el ámbito de la optimización combinatorio gran parte de los problemas son difíciles de resolver y esto es debido a que no se ha encontrado ningún algoritmo capaz de obtener la solución óptima en un tiempo polinómico. Por consiguiente no son posibles de resolver en un tiempo de ejecución razonable. A si mismo este tipo de problemas pueden ser categorizados dependiendo de la dificultad de su solución.

Se presenta el caso de problemas donde aún cuando no se ha encontrado un algoritmo polinómico que los resuelva, si es posible saber en tiempo polinómico si un valor corresponde a la solución del problema. Por ejemplo el cálculo de la raíz cuadrada de un número puede ser un problema complicado, mientras que el saber si un determinado valor es la raíz cuadrada

de otro es bastante sencillo, ya que es solo cuestión de elevar ese número al cuadrado. Este tipo de problemas son llamados *NP*. Sin realizar un análisis adecuado puede uno inferir que la gran mayoría de los problemas son *NP*, ya que uno puede argumentar que comprobar una solución es más sencillo que calcularla. Sin embargo en los problemas de optimización se presenta precisamente este caso, y comprobar que los valores obtenidos corresponden a la solución óptima no es tarea fácil.

Otro dato importante es que los problemas *P* también son problemas *NP*, ya que siempre es posible comprobar que un valor es solución al problema en tiempo polinómico. A reserva de encontrar una manera más sencilla de hacerlo una siempre tiene la opción de volver a ejecutar el algoritmo polinómico que lo resuelve y comprar el resultado con el valor obtenido.

De lo anterior se deduce entonces que el conjunto de problemas *P* es en realidad un subconjunto de los problemas *NP*. Esta relación y otros conceptos que se presentarán a continuación se presentan en la Figura 2.4[23].

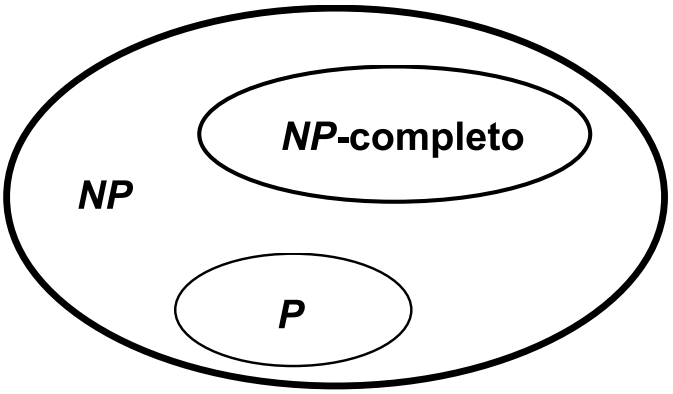


Figura 2.4: Relación entre problemas P, NP y NP-completo.

En la Figura 2.4 también se observa que existen otro tipo de problemas que son denominados *NP-completos* que se caracterizan por no tener un algoritmo que sea capaz de resolverlos en un tiempo polinómico. En la Figura 2.4 podemos apreciar que los problemas *NP-completos* son un subconjunto de problemas *NP* y en otras palabras esto quiere decir que existe un algoritmo polinómico que puede determinar si el valor obtenido es la solución al problema.

Para saber si un problema es *NP-completo*, al menos un problema *NP-completo* tiene que ser reducible a ese problema. Un problema A, que es *NP-completo*, es reducible a otro problema

B, cuando se puede crear un algoritmo que resuelva el problema A utilizando como una caja negra un algoritmo para resolver el problema B. Es decir, existe una algoritmo para resolver A de la siguiente forma[23]:

- Toma los datos de entrada del problema A, los transforma de manera que puedan utilizarse como entrada de una caja negra que resuelve el problema B, y la solución a B se pueda transformar a su vez en una solución para el problema A.
- La caja negra que resuelve el problema B puede utilizarse una única vez o un número polinómico de veces dentro del algoritmo que resuelve A.

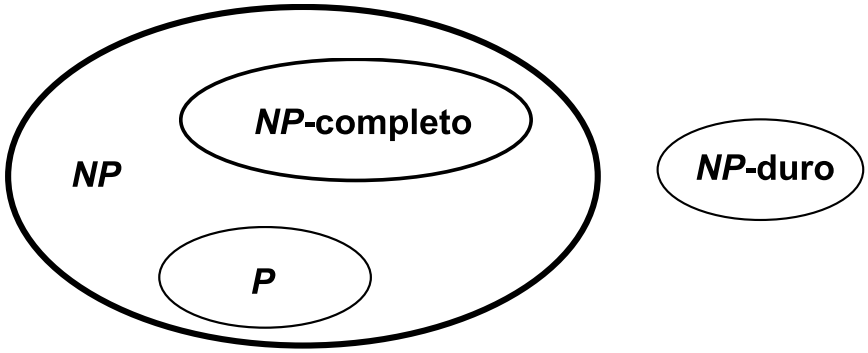


Figura 2.5: Relación entre problemas P, NP, NP-completo y NP-duro.

Por último existe otro tipo de problemas denominados *NP-duros* los cuales se definen como aquellos problemas que son al menos tan difíciles como los problemas *NP-completos* sin embargo estos no cuentan de un algoritmo polinómico que nos permita verificar una solución dada. En la Figura 2.5[23] podemos observar que estos problemas no pertenecen al subconjunto de los problemas *NP* y que realmente forman un conjunto propio.

2.4. Problema del Agente Viajero

El Problema del Agente Viajero (Traveling Salesman Problem, TSP por sus siglas en inglés) es posiblemente uno de los problemas de optimización más conocidos y estudiados[9]. El problema consiste en encontrar la ruta más corta de un agente viajero que comienza desde una ciudad de origen, visitando una serie de ciudades predeterminadas y regresar de nuevo

a su ciudad de partida. La distancia recorrida depende obviamente del orden en que visite dichas ciudades, es aquí donde el problema surge al tratar de encontrar la ruta "óptima" en el cual el orden de las ciudades visitadas produzca la distancia menor posible. Su restricción consiste en no poder regresar a ninguna ciudad que ya haya sido visitada.

Como se puede inferir de esta definición, no se necesario contar con herramientas matemáticas demasiado sofisticadas para formular matemáticamente el TSP. Sin embargo, el TSP es un problema NP-Completo [16], es muy difícil e incluso en algunas ocasiones imposible de tratar [9] y resolver instancias de gran tamaño de forma óptima. Para un grafo no dirigido completo de n vértices el tamaño del espacio de búsqueda es de $(n-1)!/2$ [11, 36].

El TSP tiene una gran cantidad de aplicaciones que van más allá de lo obvio correspondiente a planeación de rutas de un viajero de comercio y se extrapola a diferentes áreas del conocimiento, incluyendo las Matemáticas, Ciencias de la Computación, Investigación Operativa, Genética, Ingeniería y Electrónica[11].

A su vez dado a que el TSP es un problema muy estudiado numerosas variaciones del mismo han surgido a lo largo de los años, algunas de ellas son simplemente restricciones al concepto original, mientras otras son problemas de aplicación del mismo concepto. Algunas de estas variaciones son las siguientes[11]:

- TSP Simétrico (Symmetric TSP)
- TSP Máximo (MAX TSP)
- TSP con Cuello de Botella (Bottleneck TSP)
- TSP con Múltiples Visitas (TSP with Multiple Visits)
- Problema del Mensajero (Messenger Problem)
- TSP Agrupado (Clustered TSP)
- TSP Generalizado (Generalized TSP)
- TSP Dinámico (Dynamic TSP)

Matemáticamente el TSP puede ser descrito de numerosas maneras, una de ellas es referirse al TSP como un problema de grafos el cual estipula que dado $G = (V, E)$ siendo un grafo

(dirigido o no dirigido) y $\mathbb{F}$ la familia de ciclos Hamiltoneanos (rutas) en G por cada arista o conexión $e \epsilon E$ un costo (peso) c_e es designado. De esta manera el problema del agente viajero es encontrar una ruta (ciclo Hamiltoneano) en G en la cual la suma de los cotos de los aristas o conexiones de la ruta sea lo más pequeño posible. Se asume que G es un grafo completo.

Otra definición es considerar al TSP como una matriz de pesos representativos de los costos o de distancia entre ciudades, Tal matriz se define como $C = (c_{ij})_{n \times n}$ donde el valor (i, j) correspondiente a c_{ij} es el costo o distancia de unir el nodo i con el nodo j en G.

A su vez el TSP puede definirse como un problema de permutación. Se define como $\mathbb{P}_n$ la colección de todas las permutaciones del conjunto $\{1, 2, ..., n\}$. Entonces el TSP es encontrar un $\pi = (\pi(1), \pi(2), ..., \pi(n))$ de tal forma que $C_{\pi(n)\pi(1)} + \sum_{i=1}^{n-1} C_{\pi(i)\pi(i+1)}$ sea mínimo. Sin embargo esta definición es utilizada primordialmente en contexto de problemas de secuenciamiento, y recibe el nombre de *permutación cuadrática representativa[11].*

2.5. Sistemas Inmunológicos Artificiales

Los sistemas inmunológicos artificiales son sistemas adaptativos inspirados por la inmunología teórica que son aplicados para solución de problemas matemáticos, de modelado o de la ingeniería. Los algoritmos y metodologías generadas buscan imitar total o parcialmente funciones del sistema inmunológico humano, tomando en cuenta sus principales ventajas para dar las mejores soluciones a las problemáticas presentadas.

Un ejemplo de la abstracción viene de la capacidad del sistema inmune de detectar antígenos y de la búsqueda de patrones en ellos, con la finalidad de identificarlo como un patógeno. Este concepto se aplicará en la elaboración de un antivirus computacional.

La capacidad de reconocimiento de patrones en el sistema inmunológico viene dado por los linfocitos (células blancas). Teniendo esta información como base, uno procede a identificar de que manera los linfocitos hacen su labor. Esto se logra a través de una serie de células que se encuentran en la superficie de los linfocitos, llamados anticuerpos.

Los anticuerpos tienen la labor de adherirse a los antígenos en búsqueda de patrones. El grado de afinidad hacia un patógeno en particular esta dado por la cantidad de anticuerpos adherido a él. A su vez, dependiendo del grado de afinidad es la respuesta del cuerpo hacia este antígeno. A grandes rasgos, este procedimiento describe una de las maneras en las cuáles el sistema inmune del cuerpo humano responde a elementos externos.

Esta misma metodología es extrapolada a un algoritmo computacional. Se debe de identificar los elementos principales de la abstracción pasada:

- Capacidad de reconocer patrones.
- Tenemos unas células que circulan por todo el cuerpo en búsqueda de agentes externos.
- Una vez identificado un posible patógeno, comienza un proceso de reconocimiento extensivo.
- La respuesta proporcionada por el sistema es directamente proporcional al grado de afinidad obtenido.

Dado las características descritas sobre el proceso inmune podemos hacer una relación con la labor de un antivirus computacional, que de igual manera el sistema inmunológico, busca que la computadora se encuentre en condiciones sanas (libre de virus informáticos). Por este motivo, el comportamiento del sistema inmune es un buen modelo a seguir para un antivirus. A continuación se presenta un contraste entre el sistema inmune y lo que sería el antivirus:

- La capacidad de reconocer patrones puede estar dada ya sea reconociendo cadenas de bits en los programas informáticos, o por comportamientos de ciertos software al momento de ejecutarse (por ejemplo, modificación de archivos del sistema operativo que normalmente no deben de ser accesados o modificados).
- Tener un proceso en el sistema operativo que está constantemente revisando todos los demás procesos ejecutándose en la computadora.
- Una vez detectado una cadena de bits o cierto comportamiento, comenzar a monitorear y analizar de manera extensiva ese software.
- Dependiendo del resultado, se decide permitir o denegar la ejecución de ese software.

Debido a la gran cantidad de diversas funciones que lleva acabo del sistema inmune, es difícil definir de manera definitiva las áreas y maneras de aplicación de los AIS. A continuación describiremos características de interés sobre los sistemas inmunes, las cuales pueden ser extrapoladas a resolver problemáticas de la misma manera que el ejemplo anterior.

2.5.1. El sistema inmunológico: Características de interés

El sistema inmunológico tiene diversas características que son de especial interés para su analogía con el modelado matemático y los algoritmos. La gran mayoría de ellos tienen una relación directa con diversos paradigmas desarrollados en las ciencias computacionales, como lo es la computación distribuida y reconocimiento de patrones. A continuación se enumeran algunas de estas características:

- **Reconocimiento de patrones:** Las células y moléculas del sistema inmunológico tienen diversas maneras de reconocer patrones, siendo esto una de sus principales métodos de detección de patógenos. Hay moléculas superficiales que se pueden adherir a los antígenos o reconocer señales moleculares, también hay moléculas inter-celulares que se adhieren a proteínas y otras células.
- **Singularidad:** Cada individuo tiene su propio sistema inmunológico, con sus propias capacidades y vulnerabilidades, que son adquiridas a partir de la experiencia al paso de los años y al estar en contacto con diversos patógenos.
- **Identificación propia:** La singularidad del sistema inmunológico nos lleva a que cualquier célula, molécula y tejido que no es nativo al cuerpo puede ser reconocido y eliminado por el sistema inmunológico, esto es, tiene la capacidad de distinguir entre sí mismo y lo que no es.
- **Diversidad:** Existen diversos elementos (entre ellos, las células, moléculas, proteínas, etc.) que juntos, llevan a cabo la labor de identificar al cuerpo y de protegerlo de los invasores malignos y las células que no funcionan adecuadamente. Adicionalmente, existen diversas líneas de defensa, como lo es el sistema inmune innato y el inmune adaptativo.

- **Dispensabilidad (*Disposability*):** No existe una sola célula o molécula que por sí sola sea esencial para el funcionamiento del sistema inmune. Estas células y moléculas están siendo reemplazadas constantemente, aunque existen algunas excepciones de células que tienen largos plazos de vida, donde cumplen la función de una memoria para el sistema inmune.
- **Autonomía:** No existe un elemento central que controle al sistema inmune; no requiere de intervención del exterior o de mantenimiento. Actúa de manera autónoma al clasificar y eliminar los patógenos y es capaz de reparase a sí mismo al reemplazar sus propias células dañadas.
- **Multicapa:** Múltiples capas de diferentes mecanismos que actúan cooperativamente y competitivamente son combinados para obtener un grado de seguridad mayor. Esto quiere decir que al tener diversos mecanismos que buscan el mismo objetivo, como lo es el eliminar agentes malignos del sistema, tenemos un mayor grado de fidelidad en las respuestas del mismo y nos provee la seguridad que difícilmente pase desapercibido algún problema.
- **No existe una capa segura:** Cualquier célula del organismo puede ser atacada por el sistema inmunológico, inclusive el mismo sistema inmunológico.
- **Detección de anomalías:** El sistema inmune tiene la capacidad de reconocer y reaccionar a los patógenos que el cuerpo nunca antes ha experimentado.
- **Sistema Distribuido:** Las células, moléculas, y órganos que conforman al sistema inmune están distribuidos por todo el cuerpo, y más importante aún, no son sujetas a un control centralizado.
- **Tolerancia al ruido:** Un reconocimiento total y absoluto de los patógenos no es requerido, el sistema toma medidas al respecto si un reconocimiento parcial es detectado. El sistema inmunológico es tolerante a ruido molecular.
- **Capacidad para adaptarse:** Aún cuando las perturbaciones pueden reducir el funcionamiento del sistema inmunológico este es capaz de persistir considerando estas perturbaciones. Cuando el organismo está agotado o desnutrido, el sistema inmunológico

es menos efectivo al requerir mayor energía para recuperar y mantener a organismo, pero aún es capaz de efectuar su labor.

- **Tolerancia a los errores:** Si una respuesta inmune a un patógeno ha sido creada, y por algún motivo la célula encargada del mismo es removida, esta acción hará que otro tipo de células respondan al patógeno. De la misma manera, muchas otras asignaciones por el sistema inmunológico permiten la reasignación de tareas a otros elementos en caso de que alguno de ellos falle.
- **Robustez:** La gran diversidad y número de células y moléculas, junto con su distribución son en gran parte responsables de la robustez del sistema inmune.
- **Aprendizaje inmune y memoria:** Las moléculas del sistema inmune pueden adaptarse a sí mismas, estructuralmente y en cantidades, a los retos antigénicos. Estos mecanismos de adaptación son seguidos por una rigurosa precisión selectiva, que permite a los mejores individuos adaptados permanecer en el sistema inmune por largos periodos de tiempo. Estos individuos con periodos largos de vida tienen el nombre de células de memoria y promueven respuestas más rápida y efectivas al mismo o simular antígeno. A si mismo las células y moléculas puede distinguirse a sí mismas, proporcionando al sistema inmune un comportamiento automatizado propio.
- **Respuestas cazador-presa:** El sistema inmune replica células para lidiar con patógenos que también se están replicando, ya que de no hacerlo el cuerpo se vería invadido por el patógeno de una manera rápida.
- **Organización propia:** Cuando un patrón antigénico interactúa con el sistema inmune, no existe información de cómo las células y las moléculas deberían de adaptarse para lidiar con el antígeno. La selección clonal y la afinidad de maduración son responsables de seleccionar y expandir las células mejor adaptadas, para que tengan mayores periodos de vida.

2.5.2. Breve Historia de Inmunología

En la medicina el término *inmunidad* se refiere a una condición del cuerpo humano en la cual el organismo puede resistir enfermedades contagiosas. Sin embargo dando una definición más amplia podemos definir a la inmunidad como la reacción del cuerpo hacia agentes externos (patógenos) con la finalidad de protegerse. El sistema inmunológico consiste de una serie de órganos, células y moléculas, pero con un fuerte énfasis en su habilidad para trabajar en forma armónica y coordinada para combatir los agentes externos que buscan dañar a nuestro cuerpo.

La Inmunología es una ciencia relativamente nueva. Su origen es atribuido a Edward Jenner quien en 1796 descubrió que al introducir cantidades pequeñas de *vaccinia* en un animal provocaría que ese animal generara defensas contra esa enfermedad que usualmente era mortal. Este es el origen de la palabra *vacunación* y hoy en día podemos definirla como la inoculación de individuos sanos con muestras debilitadas de agentes causantes de enfermedades, de tal forma que se fomente una protección hacia futuras infecciones de esa enfermedad[18].

En la Tabla 2.1 se tiene un resumen de la evolución del campo de la inmunología desde el ámbito biológico hasta el año de 1990[18].

Gracias a estos avances en la inmunología los AIS lograron concebirse pero no fue hasta mediados de 1980 con los trabajos sobre Redes Inmunológicas de Farmer, Packard y la Perelson (1986) y Bersini y Varela (1990). Sin embargo, no fue sino hasta mediados de 1990 que los AIS se convirtieron en un tema de gran interés en la comunidad científica. Autores como Forrest et al., Kephart et al. [17] publicaron sus primeros artículos sobre AIS (en los conceptos de selección negativa) en 1994, mientras que por su cuenta Dasgupta realizó varios estudios sobre los mismos algoritmos. Hunt y Cooke comenzó las obras en los modelos de Redes Inmunológicas en 1995; Timmis y Neal continuó esta labor e hizo algunas mejoras. De Castro y Von Zuben y el trabajo de Nicosia & Cutello (en la Selección Clonal) se hizo notable en 2002. El primer libro sobre los Sistemas Inmunológicos Artificiales se editó por Dasgupta en 1999[18].

Hoy en día nuevas ideas, como la teoría de peligro (*Danger Theory*) y los algoritmos inspirados en el sistema inmune innato, también están siendo exploradas. Otros acontecimientos

Tabla 2.1: Periodos en la historia de la inmunología

Tendencias	**Periodo**	**Pioneros**	**Conceptos**
Aplicación	1796-1870	E. Jenner	Inmunización
		R. Koch	Patología
	1870-1890	L. Pasteur	Inmunización
		E. Metchnikoff	Fagocitosis
Descripción	1890-1910	E. von Begring y S. Kitasato	Anticuerpos
		P. Ehrlich	Receptores celulares
	1910-1930	J. Bordet	Especificidad/ Complemento
		K. Landsteiner	Haptenos/Tipos de Sangre
Mecanismos	1930-1950	A. Breinl y F. Haurowitz	Sintetización de Anticuerpos
		L. Pauling	Instruccionismo
	1950-1980	M. Burnet y Talmage	Selección Clonal
		N. Jerne	Cooperación de la red y células
Molecular	1980-1990	S. Tonegawa	Estructura y diversidad de los receptores

recientes incluyen la exploración de la degeneración en modelos de AIS[1, 19] ya que está investigación se ve motivada por la hipótesis de que juega un gran papel en las cuestiones de aprendizaje y evolución del sistema inmune y otros sistemas biológicos[7, 33].

Aunque originalmente los AIS se propusieron con la finalidad de crear abstracciones del sistema inmune, recientemente se ha generado gran interés en el modelado de los procesos biológicos por medio de estos estudios y su aplicación en los campos de la matemática, la computación y la ingeniería.

En 2008, Dasgupta y Nino publicaron un libro de texto sobre computación inmunológica que presenta un compendium del estado del arte en las técnicas basadas en inmunidad y describe una amplia variedad de aplicaciones[5].

2.5.3. Áreas de aplicación de los AIS

Como ya fue mencionado, uno de los principales campos de aplicación de los Sistemas Inmunológicos Artificiales es en el reconocimiento de patrones. Sin embargo, sus campos de aplicación son mucho más amplios y no debemos limitarnos a solamente a la idea de reconocimiento de patrones con la finalidad de eliminarlos (como en caso del software antivirus para las computadoras), como sería una abstracción directa del funcionamiento del sistema inmune. A continuación se presentan algunas de las áreas de aplicación de los AIS [18], sin embargo no debemos pensar que se encuentran limitados o que sean enumerados todos ellos, ya que día tras día nuevas formas de aplicar los paradigmas del sistema inmune son investigados y llevados a la práctica:

- Reconocimiento de patrones
- Detección de fallas y anomalías
- Búsqueda y clasificación de datos
- Sistemas basados en agentes
- Creación de horarios
- Aprendizaje máquina
- Control y navegación autónoma
- Optimización
- Vida artificial

2.5.4. Selección Negativa

La inspiración biológica de este algoritmo proviene de el proceso de eliminación de los linfocitos auto-reactivos denominada supresión clonal y se lleva a cabo a través de un mecanismo llamado selección negativa que opera en los linfocitos durante su proceso de maduración. Para las células T esto ocurre principalmente en el timo, que ofrece un entorno rico en células presentadoras de antígeno que presentan antígenos propios (*self-antigens*). Células

T inmaduras que unen firmemente a estos antígenos experimentar una muerte controlada (apoptosis). Por lo tanto, las células T que sobreviven a este proceso debe ser no reactivas a los antígenos propios (*self-antigens*). La propiedad de los linfocitos de no reaccionar a sí mismos se llama *tolerancia inmunológica*.

Algoritmo 2.4 Algoritmo de Selección Negativa.

Entrada: $S_{conocidos}$: conjunto de elementos propios conocidos

Salida: D: El conjunto de los detectores generados

1. **comenzar**
   - *a*) **while** !criterioDeParo() **do**:
     1) $P \leftarrow$generarDetectorAleatorio()
     2) $A \leftarrow$DeterminarAfinidad(P,$S_{conocidos}$)
     3) **if** $A > LimiteDeDetección$ **then**
        a' Rechazar
     4) **else**
        a' $D = D + P$
   - *b*) **end while**
   - *c*) **regresar** D
2. **terminar**

Algoritmos de selección negativa se inspiran en el mecanismo principal del timo, que produce un conjunto de células T maduras capaces de unirse sólo a antígenos propios (*self-antigens*). El primer algoritmo de selección negativa fue propuesto por Forrest et al en 1994 para detectar la manipulación de datos causada por un virus en un sistema informático[28]. El punto de partida de este algoritmo es para producir un conjunto de cadenas propias (*self-strings*), S, que definen el estado normal del sistema. La tarea, entonces, es generar un conjunto de detectores, D, que sólo se unen al reconocer el complemento de S. Estos detectores se pueden aplicar a los nuevos datos con el fin de clasificar como si mismos (*self*) o como no mismo (*non-self*), destacando el hecho de que los datos han sido manipulados. El algoritmo produce el conjunto de detectores a través del proceso descrito en el Algoritmo 2.4[28, 18, 5].

2.5.5. Selección Clonal

De acuerdo con la teoría de Burnet presentada en 1959 sobre la Selección Clonal, el repertorio del sistema inmunológico se somete a un mecanismo de selección durante la vida del individuo[5]. La teoría establece que en la unión con un antígeno adecuado, se produce la activación de los linfocitos. Una vez activado, los clones de los linfocitos expresan receptores idéntico al linfocito original que se encontró con el antígeno. Así, una expansión clonal del linfocito original se produce. Esto asegura que sólo los linfocitos específicos para la activación de un antígeno se producen en grandes cantidades. La teoría de la selección clonal también manifestó que cualquier linfocito que tenga receptores de antígenos específicos para las moléculas del propio cuerpo del organismo debe ser eliminado durante el desarrollo de los linfocitos, de tal forma que únicamente el antígeno de un patógeno puede causar un linfocito a expandirse clonalmente y así obtener una respuesta destructiva inmune adaptativa. En este sentido, el sistema inmunológico puede ser visto como un clasificador de antígenos en dos categorías: en antígenos propios (self-antigens) y antígenos no propios (non-self antigens), asumiendo que el antígeno no propio proviene de un patógeno y por consiguiente es eliminado del cuerpo.

Durante la expansión clonal de las células B (pero no las células T), la afinidad de los anticuerpos promedio se incrementa para el antígeno que provocó la expansión clonal. Este fenómeno es la maduración llamada afinidad, y es responsable por el hecho de que en una posterior exposición al antígeno la respuesta inmune es más eficaz debido a los anticuerpos que tienen una mayor afinidad para el antígeno.

La maduración de la afinidad es causada por una hipermutación somática y el mecanismo de selección que se produce durante la expansión clonal de las células B. Hipermutación somática altera la especificidad de los anticuerpos mediante la introducción de cambios aleatorios en los genes que codifican para ellos.

La teoría de la selección clonal se ha utilizado inspiración para el desarrollo de los AIS que realizan la optimización computacional y las tareas de reconocimiento de patrones. En particular, la inspiración se ha tomado de un proceso impulsado por el antígeno de maduración de la afinidad de las células B, con su mecanismos de hipermutación asociados. Estos AIS

también a menudo utilizan la idea de las células de memoria para mantener una buena solución al problema a resolver. En el libro de Castro y Timmis [18], exponen dos importantes características de maduración de la afinidad en las células B que pueden ser explotadas desde el punto de vista computacional. El primero de ellos es que la proliferación de células B es proporcional a la afinidad del antígeno que se une, por lo tanto cuanto mayor sea la afinidad, más clones son producidos. En segundo lugar, las mutaciones sufridas por los anticuerpos de las células B son inversamente proporcionales a la afinidad del antígeno que se une.

De Castro y Zuben Von desarrollaron uno de los AIS de Selección Clonal más ampliamente utilizado llamado *CLONALG*, el cual se ha empelado en tareas de búsqueda de patrones y optimización de funciones multi-modales[18, 5], aunque muchos otros existen[29].

Algoritmo 2.5 Algoritmo de Selección Clonal.

Entrada: S: conjunto de patrones por reconocer

n: número de los peores elementos para ser removidos

Salida: M: conjunto de detectores de memoria capaces de clasificar patrones no conocidos

1. **comenzar**
   - *a*) $A \leftarrow$crearAnticuerpos()
   - *b*) **foreach** elemento (patrón) en S **do**:
     1) $Aff \leftarrow$determinarAfinidad(A)
     2) $C \leftarrow$generarClones(Aff) //Conjunto de los anticuerpos con mayor afinidad; número de clones directamente proporcional a la afinidad correspondiente
     3) Mutar los atributos de A con C, copiar los anticuerpos de más alta afinidad a M
     4) Reemplazar(A,n) //en base a los anticuerpos de menor afinidad, y generar nuevos anticuerpos de manera aleatoria.
   - *c*) **end while**
   - *d*) **regresar** M
2. **terminar**

Desde el punto de vista de reconocimiento de patrones el algoritmo de Selección Clonal *CLONALG* consiste en relacionar un conjunto de elementos S (antígenos) y producir un conjunto

de memoria inmune M (anticuerpos) correspondientes. En el Algoritmo 2.5 se denota este procedimiento a manera de pseudo-código[18, 5].

2.5.6. Redes Inmunológicas

En 1974, Jerne propuso una teoría de red inmune para ayudar a explicar algunas de las propiedades observadas emergentes del sistema inmunológico, tales como el aprendizaje y la memoria. La premisa de la teoría de red inmune es que cualquier receptor de linfocitos dentro de un organismo puede ser reconocido por un subconjunto del repertorio de receptores total. Los receptores de reconocimiento de este conjunto tienen su propio reconocimiento y así sucesivamente una red de interacciones inmunológicas se forman. Redes inmunes a menudo se conoce como redes idiotipo.

En ausencia de antígeno foráneo, Jerne concluyó que el sistema inmune debe mostrar un comportamiento o actividad que resulta de las interacciones con sí mismo, y surgen de estas interacciones comportamiento inmunológico tales como la tolerancia y la memoria[18, 5]. En el Algoritmo 2.6 se presenta el pseudo-código correspondiente a una Red Inmune genérica[18, 5].

Algoritmo 2.6 Algoritmo de Red Inmune.

Entrada: S: conjunto de patrones por reconocer

nt: el umbral de afinidad de la red

ct: el umbral de número de elementos clonados

h: número de clones de mayor afinidad

a: número de nuevos anticuerpos para introducir

Salida: N: conjunto de detectores de memoria capaz de clasificar los patrones invisibles

1. **comenzar**
   - *a*) $N \leftarrow$CrearAnticuerposRed()
   - *b*) **while** !criterioDeParo() **do**
     1) **foreach** elemento (patrón) en S **do**:
        - a' $Aff \leftarrow$determinarAfinidad(N)
        - b' $C \leftarrow$generarClones(Aff) //Conjunto de los anticuerpos con mayor afinidad; número de clones directamente proporcional a la afinidad correspondiente
        - c' Mutar los atributos de estos clones con el conjunto A. Tomar a y reemplazar h número de clones con la más alta afinidad en un conjunto de memoria C
        - d' Eliminar todos los elementos de C que estén por debajo de ct
        - e' Determinar la afinidad de todos anticuerpos dentro de C consigo mismos. Eliminar todos aquellos que estén por debajo de ct.
        - f' $N \leftarrow C$ //Incorporar elementos restantes de C a N.

     2) **end foreach**

     3) Determinar la afinidad de todos anticuerpos dentro de N consigo mismos. Eliminar todos aquellos que estén por debajo de ct.

     4) Introducir un número aleatorio de anticuerpos generados aleatoriamente en N.
   - *c*) **end while**
   - *d*) **regresar** M
2. **terminar**

2.5.7. Tendencias futuras de los AIS

Los Sistemas Inmunológicos Artificiales constituyen un paradigma de computación inteligente emergente. Por este motivo no es difícil pensar que tengan un gran rango de posibles vías de investigación. Estas posibilidad pueden ser englobadas en cuatro grandes campos, que se relacionan directamente con el trabajo que ya existe sobre los AIS y a su vez de sus tendencias futuras[18]. Estos campos son:

1. **La mejora y el aumento de los AIS actuales:** Mejorar el desempeño de los AIS actuales, particularmente hablando sobre optimizar su funcionamiento en sus aplicaciones.
2. **Desarrollo de nuevos AIS:** *En contraste con el primer punto, este esta relacionado con la extracción de nuevas metáforas y el desarrollo de algoritmos novedosos al explorar y explotar aún más las ideas, conocimientos y conceptos que se tienen sobre el sistema inmune.*
3. **Aplicaciones novedosas para los AIS:** Hoy en día se ha dedicado una gran cantidad de esfuerzo por encontrar áreas de aplicación donde los AIS logren encontrar un nicho. Se espera que esta tendencia siga y refuerce la importancia que han tenido en diversos campos de aplicación.
4. **Extensión del marco de los AIS:** Este punto hace referencia a desarrollar el marco teórico y conceptual sobre el cual los AIS son diseñados.

Aún cuando todas estas tendencias son de vital importancia para que los AIS sigan evolucionando, en particular el punto número 2 es de interés. Esto se debe a que este trabajo de tesis busca precisamente indagar en la terminología inmunológica en búsqueda de una metáfora que completamente una problemática muy importante en el mundo de la matemática combinatoria. Esta metáfora consistió en basarse en el concepto de inmunización por vacunación, un concepto que no había sido explorado del todo en el ámbito de los AIS.

Con este concepto se ha desarrollado toda una metodología que busca ser fiel al proceso de vacunación llevado a cabo en el sistema inmune.

3 Planificación Óptima Discreta para Problemas NP−Completos mediante Algoritmo de Vacunación

3.1. Planteamiento de la problemática

El desempeño de la gran mayoría de algoritmos para la solución de problemas de optimización combinatoria decrece al aumentar el número de elementos con los que se trabajan. Usualmente esta caida en el desempeño se presenta ya sea en la calidad de la soluciones que los algoritmos son capaces de encontrar o en un incremento substancial en el tiempo necesario para encontrar una solución viable. Esta situación presenta una gran problemática para los algoritmos desarrollados y obtener alguna metodología para ayudar aliviar este problema es el enfoque de este trabajo.

De esta manera se realizará investigación y se desarrollará una metodología que permita mejorar el desempeño de algoritmos para la solución del TSP basados en población, pero que estos no sean modificados ni conceptual (retienen sus terminologías originales) ni estructuralmente (no es necesario modificar el algoritmo).

3.2. Propuesta de solución

El planteamiento propuesto consiste en lograr construir una infraestructura alrededor del algoritmo principal solucionador del TSP la cual trabaja modificando el listado original de ciudades, siendo estas los elementos básicos en el problema del TSP y de todos los algoritmos que los solucionan. De esta manera se desarrolla un algoritmo que agrupa conjuntos de ciudades, efectivamente reduciendo el número de elementos con los cuales trabaja el algoritmo de solución del TSP.

El agrupamiento y reducción de ciudades se hace tratando de imitar la manera en que un ser humano observaría y segmentaría el problema, que es uniendo aquellas rutas que parecen obvias ya sea por su cercanía a otras o por la falta de ciudades próximas. Esto quiere decir que el algoritmo intenta plasmar el concepto de tener un "Experto[1]" artificial que decide en que partes del repertorio de ciudades se pueden hacer reducciones.

La solución propuesta es utilizar el concepto de vacunación, el cual se define como conjunto de subrutas que agrupa un número de ciudades con algún criterio, y una vez creado el repertorio de vacunas las ciudades englobadas son removidas del repertorio original y posteriormente se agrega una nueva ciudad que surge como una representación de las ciudades removidas.

3.2.1. Vacuna

La vacuna es el elemento básico con el cual se construye el algoritmo. Podemos definirla como conjunto de elementos que representa una subruta utilizada para reducir el repertorio de ciudades a tratarse por algún otro algoritmo de solución al TSP. En la Figura 3.1 se presenta un esquema de cómo debe ser organizados lógicamente los datos que la conforman y a continuación se presenta una explicación de cada uno de ellos:

- *Listado de ciudades*: Las vacunas deben de tener ya sea los datos que representan a cada ciudad como sus coordenadas, índice en el arreglo original, un identificador único o en caso de un lenguaje orientado a objetos el objeto que representa a la ciudad. A su

[1]Definiremos a un experto como una persona que tiene los datos, información, conocimiento y experiencia respecto a la solución de un problema.

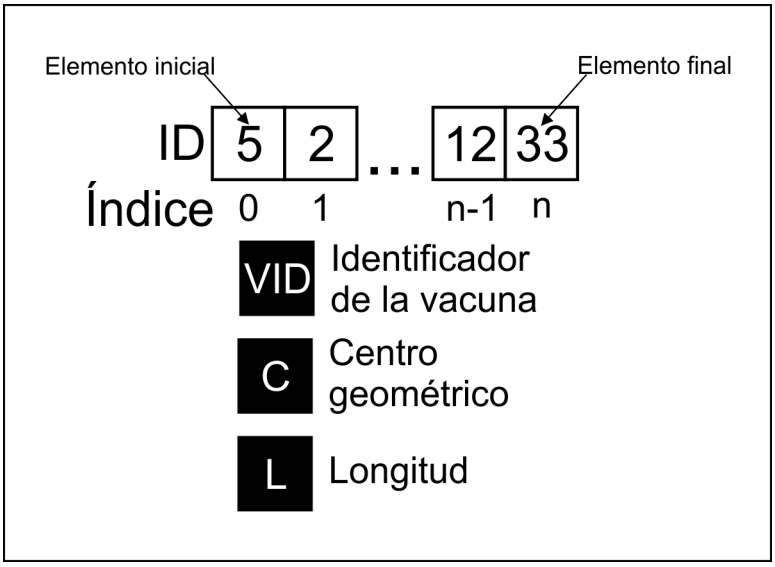


Figura 3.1: Esquema de la representación lógica de una vacuna.

vez, el orden en el cual están almacenados en este listado es el orden que representa la ruta.

- *Número de ciudades que contiene*: Adicional al listado de ciudades, es necesario tener el conteo de las mismas.
- *Ciudad inicial*: El identificador a la ciudad inicial debe estar claramente definido.
- *Ciudad final*: El identificador a la ciudad final debe estar claramente definido.
- *Vacuna ID*: El identificador de la vacuna. Se define por un número consecutivo, pero negativo.
- *Centro geométrico*: La localización de la vacuna en el plano. Al ser parte de la generación de vacunas el remover elementos surge la necesidad de crear una posición equivalente. Este parámetro es de suma importancia ya que provee las coordenadas del nuevo elemento que será agregado al listado de ciudades a tratar por el algoritmo principal.
  - Centro en x: $x_C = \frac{\sum_{i=0}^{n}(x_i)}{n}$
  - Centro en y: $y_C = \frac{\sum_{i=0}^{n}(y_i)}{n}$

- *Longitud*: La distancia correspondiente a las ciudades englobadas por la vacuna. Calculada por distancia Euclidiana y redondeado al entero más próximo.

$$L = \sum_{i=0}^{n-1} \left(\sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} \right)$$

Se definen dos maneras de generar las vacunas. La diferencia entre ambas es el factor de decisión con respecto a que ciudades serán elegidas para crear las vacunas. El primer método consiste en utilizar un Selector Aleatorio (SA) el cual elige aleatoriamente las ciudades requeridas del repertorio de ciudades y posteriormente con éstas se genera su vacuna correspondiente. El siguiente método se caracteriza por tener un Selector Elitista (SE) que consiste en generar la matriz de distancias de cada una de las ciudades y en base a esta elegir las vacunas que su ruta sean las más cortas. La generación de vacunas será explicada a detalle en la sección 3.2.5 y 3.2.6.

3.2.2. Proceso de vacunación

Adicional al concepto de la Vacuna en la Figura 3.2 se presenta el diagrama de flujo del proceso de vacunación. Como podemos observar el algoritmo requiere como entrada el repertorio de elementos a trabajar por el algoritmo, que en este caso son las ciudades para el algoritmo que soluciona el TSP. De ahí pasa a la generación de vacunas que es el cual como se puede observar consiste en un pre-proceso de agrupamiento y reducción de elementos, lo cual tiene como resultado un nuevo repertorio de ciudades reducidas. Una vez generado el nuevo repertorio de ciudades este pasa a ser solucionado por el algoritmo solucionador del TSP, y una vez que este proporcione su resultado en forma de la ruta óptima generada junto con los elementos originales y los elementos reducidos se procede a un pos-proceso que consiste en la expansión de la ruta obtenida a los términos originales. Finalmente se obtiene la longitud real que de la ruta generada para el TSP.

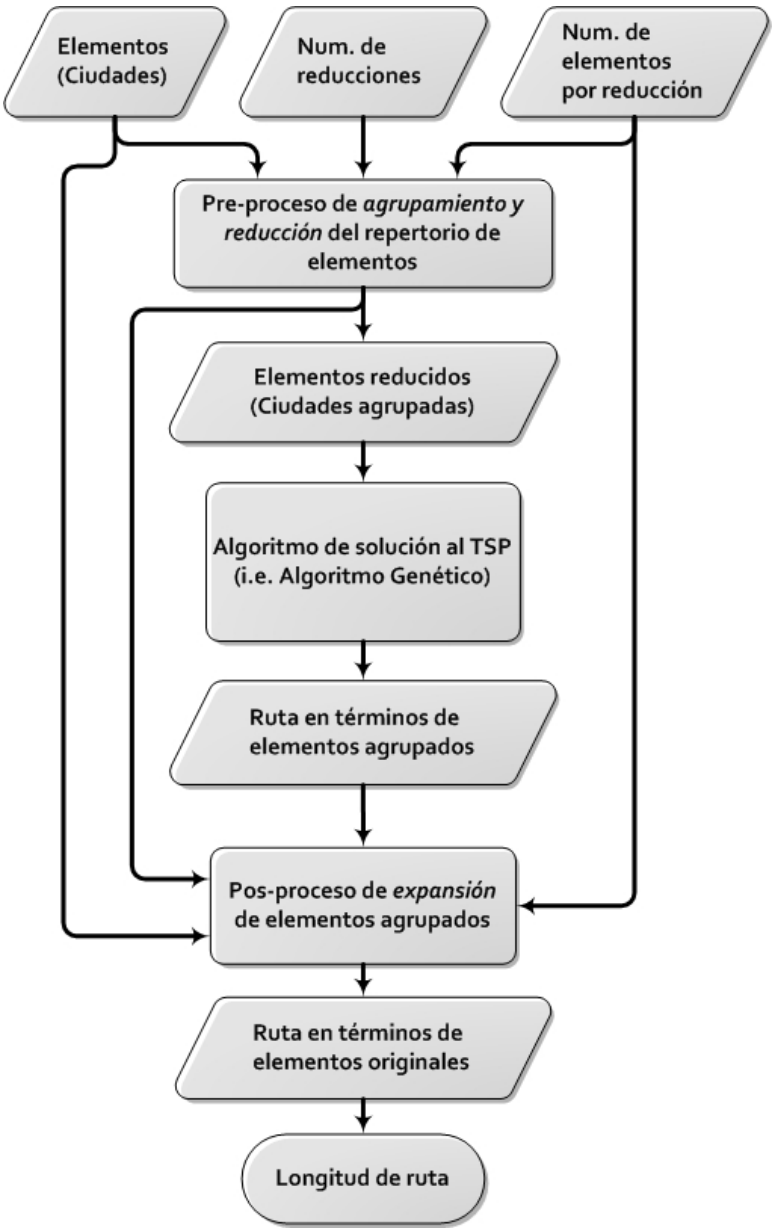


Figura 3.2: Diagrama de flujo del proceso de Vacunación.

3.2.3. Conceptos, variables y parámetros del algoritmo

En la Tabla 3.1 se presentan y describen los conceptos, variables y parámetros que forman parte del algoritmo de vacunación.

Tabla 3.1: Parámetros del algoritmo de vacunación.

Parámetro	Abreviación	Descripción
Número de elementos	*NE*	Define el número de elementos total del listado original.
Número de vacunas	*NV*	Parámetro que define cuantas reducciones (vacunas) tendrá el listado original.
Número de elementos por vacuna	*EV*	Parámetro que define la longitud en elementos de cada vacuna.
Listado original	*LO*	El listado original de elementos.
Listado reducido	*LR*	El listado reducido (vacunado) de elementos.
Listado expandido	*LE*	El listado que se obtiene después de expandir la ruta obtenida por el algoritmo de solución al TSP.

3.2.4. Restricciones

Las vacunas a su vez se encuentras restringidas por algunas limitaciones matemáticas. A continuación se enlistan las características principales que deben cumplir las vacunas generadas:

- Toda vacuna tiene como longitud mínima 2 y máxima igual a NC:

$$2 < NV < NC$$

- Número de vacunas (NV) y elementos por vacuna (EV) están sujetos a que su producto sea menor al número de elementos (NE):

$$NV \cdot EV \leq NE$$

La primera restricción nos indica que no puede existir una vacuna consistente en una ciudad, debido a que para lograr un agrupamiento se requieren tener dos ciudades como mínimo, y una vacuna puede como máximo igual al número total de ciudades, ya que si una vacuna es igual en longitud a NC entonces tendríamos una vacuna que representa una solución para

el problema y el excederse no es posible. La segunda restricción existe para garantizar que uno no esté agrupando más allá del número total de ciudades.

3.2.5. Generación de Vacunas: Selector Aleatorio

El objetivo de este selector consiste en proporcionar una manera de generar las vacunas que sea rápida y eficiente. Dado que la desventaja principal de un algoritmo al tratar con un conjunto de elementos variables es la falta de conocimiento acerca de los mismos, el Selector Aleatorio *(SA)* es el primer planteamiento presentado que busca dar solución a este problema.

La selección aleatoria se utiliza para elegir aquellos elementos que servirán como el elemento inicial de la secuencia que formará la vacuna. Una vez obtenido el elemento inicial se procede a calcular los elementos que se encuentran más próximos a él utilizando la fórmula del cálculo de la distancia euclidiana.

$$distancia = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

De esta manera, la vacuna se construye calculando la distancia hacia el resto de los elementos a partir de la ciudad elegida aleatoriamente. Se mantiene calculando el siguiente elemento hasta cumplir la condición *EV.*

En el Algoritmo 3.1 se presenta un pseudocódigo de la generación de vacunas con Selector Aleatorio:

Algoritmo 3.1 Generación de vacunas por Selector Aleatorio.

1. Definir: *Num. de vacunas*, *Num. Ciudades por vacuna*
2. Mientras no se cumpla con el *Num. de vacunas*
   - *a*) Elegir aleatoriamente una ciudad
   - *b*) Calcular ruta con *Num. Ciudades* por vacuna
   - *c*) Generar vacuna con la ruta calculada; calcular centro geométrico para la vacuna.
   - *d*) Remover ciudades implicadas del listado original de ciudades
   - *e*) Agregar vacuna al listado
   - *f*) Remover del repertorio de ciudades la ciudad elegida aleatoriamente
3. Se regresa el listado de vacunas y ciudades modificado

3.2.6. Generación de Vacunas: Selector Elitista

El siguiente método propuesto es el de generar un repertorio de vacunas que tengan como factor de decisión lo que se ha definido como un Selector Elitista (*SE*). El Selector Elitista tiene como finalidad proporcionar una medida de decisión para qué elementos deben de formar las vacunas. Debido a que no podemos determinar cuáles son las mejores vacunas se decidió ordenar las ciudades por la ruta más corta a su ciudad inmediata y en base a esto hacer la selección del número de vacunas requeridas considerando el número de ciudades a agrupar. De esta manera en el Algoritmo 3.2 se presenta el pseudocódigo de generación de vacunas con Selector Elitista.

Algoritmo 3.2 Generación de vacunas por Selector Elitista.

1. Definir: *Num. de vacunas*, *Num. Ciudades por vacuna*
2. Generar matriz de distancias
3. Crear listado de *Num. Ciudades* por vacuna más cercana por cada ciudad
4. Ordenar por distancia
5. Por cada ciudad
   - *a*) Si no se ha cumplido con el *Num. de vacunas*
     - 1) Tomar la ruta de las ciudades más próximas
     - 2) Generar las dos posibles rutas (der. - izq., izq. - der.)
     - 3) Si las rutas no están en la lista de vacunas
       - a' Se genera la vacuna con las ciudades implicadas; calcular centro geométrico para la vacuna.
       - b' Remover ciudades implicadas del listado original de ciudades
       - c' Agregar vacuna al listado
     - 4) De lo contrario
       - a' No se agrega
   - *b*) De lo contrario
     - 1) Detener
6. Se regresa el listado de vacunas y ciudades modificado

Podemos observar que a diferencia del selector aleatorio este tiene que calcular la matriz de distancias de todas las ciudades y realizar una operación de ordenamiento. Esto sin duda es la desventaja más grande de utilizar este método, ya que el cálculo de la matriz de distancias es de complejidad $O(n^2)$ al ser necesario ciclar dos veces por cada una de las ciudades y también dependiendo del algoritmo de ordenamiento puede agregar a la complejidad en la generación de vacunas.

3.2.7. Expansión de elementos vacunados

Algoritmo 3.3 Algoritmo de expansión de elementos vacunados.

1. Entra: Listado original *LO,* listado reducido *RV,* ruta óptima, Listado de Vacunas
2. Se elije la primera liga de la ruta optima
3. Se obtiene la primera ciudad de la conexión de la liga
4. Por cada conexión en la ruta óptima
   - *a*) Si no es una vacuna
     1) Agregar al listado de ciudades expandida
   - *b*) Si es vacuna
     1) Recuperar la vacuna
     2) Obtener la distancia *d1* y *d2* que representan la distancia de la ciudad actual a la ciudad inicial y la ciudad final de la vacuna respectivamente
     3) Si $d1 \leq d2$
        - a' Agregar al repertorio de ciudades expandidas la ciudad inicial
        - b' Agregar el resto de la ciudades
     4) De lo contrario
        - a' Agregar al repertorio de ciudades expandidas la ciudad final
        - b' Agregar el resto de la ciudades
   - *c*) Definir la siguiente ciudad basándose a la ciudad actual
5. Regresar el listado de ciudades expandidas

La expansión de elementos vacunados tiene la finalidad de reconstruir la ruta final encontrada por el algoritmo de solución al TSP en términos del listado original de ciudades, ya que recordemos que ésta se encuentra en términos de las ciudades agrupadas y no representa la verdadera longitud de ruta.

Independientemente de que método de generación de vacunas empleado, la expansión de los elementos funciona de la misma manera. Esto se debe a que aún cuando ambos algoritmos de generación de vacunas producen distintos repertorios de ciudades reducidas, ambos producen las mismas variables requeridas por la expansión. En el Algoritmo 3.3 se presenta el pseudo-código correspondiente al proceso de expansión. Una vez que la ruta óptima es expandida obtenemos un nuevo arreglo de ciudades en el cual el orden indica la ruta óptima.

3.3. Desarrollo de la plataforma experimental de software

Con la finalidad de poder hacer pruebas y obtener datos para evaluar el algoritmo propuesto, se ha desarrollado una plataforma experimental en software. Esta aplicación tiene las siguientes características:

1. Implementación de *lectura y escritura de archivos .tsp*
2. *Cinco (5) TSP predefinidos* que facilitan la experimentación y recreación de los datos presentados en este trabajo de tesis
3. Solución al TSP por medio de *Algoritmo Genético*, con facilidad para modificar los parámetros de ejecución
4. Implementación del algoritmo de vacunación con *Selector Aleatorio* y *Selector Elitista*, con facilidad para modificar los parámetros de generación
5. Capacidad de *salvado manual y automático en base de datos* que facilita el análisis estadístico

Con respecto al punto #1 la implementación del formato de archivo .tsp descrito por la librería "TSPLIB" que tiene como finalidad establecer un formato estándar para la creación, utilización y distribución de listado de ciudades, soluciones óptimas y resultados de obtenidos

por otros miembros. TSPLIB proporciona ejemplos de TSP de diversos números de ciudades y con sus respectivas soluciones óptimas, lo cual proporciona las herramientas necesarias para comparar los resultados obtenidos por nuevos algoritmos y metodologías.

Este formato estándar consiste en uno de los elementos más importantes en la implementación de la plataforma de software ya que gracias a ella se han obtenido problemas de prueba con sus soluciones óptimas.

3.3.1. Descripción de los paneles

En la Figura 3.3 se muestra las partes que conforman la plataforma experimental de software. En la parte superior se encuentra la barra con las opciones principales que ofrece esta aplicación. A continuación se describe cada una de ellas:

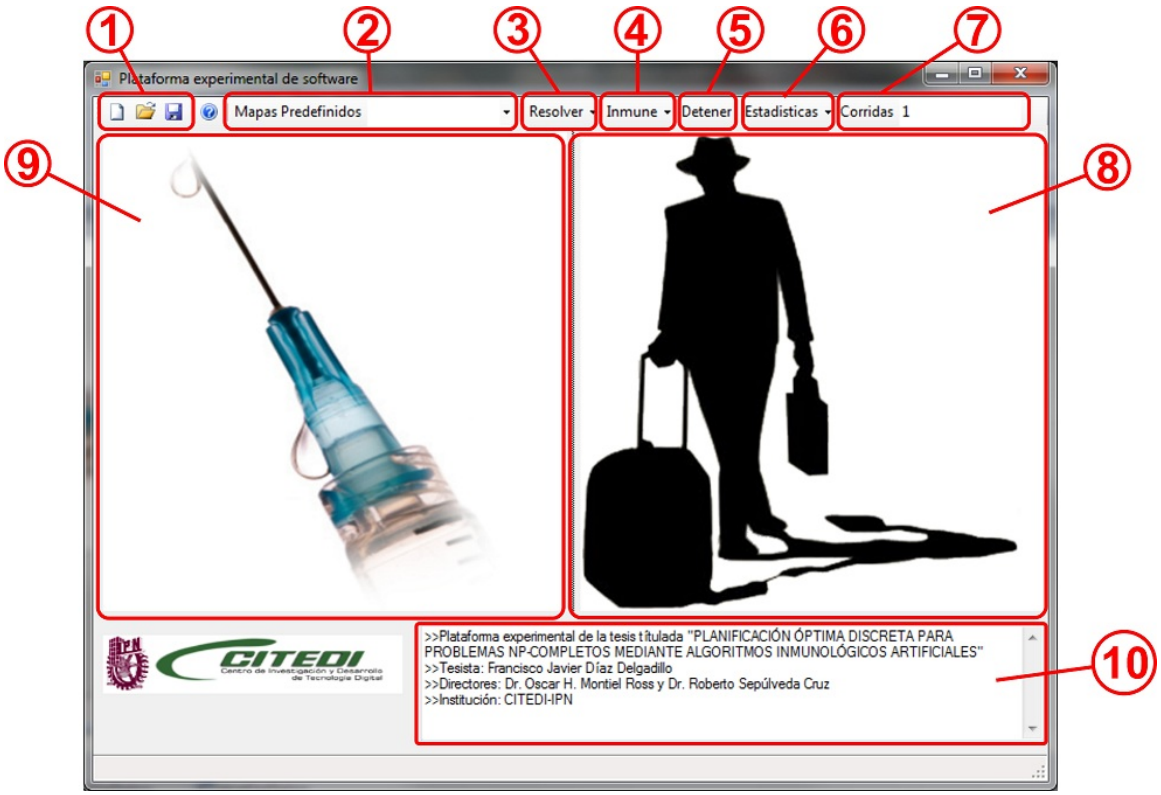


Figura 3.3: Captura de pantalla de la plataforma experimental.

1. **Menú principal:** contiene los botones para crear nuevo documento, abrir documentos o salvar el documento actual.

2. **Sección de mapas predefinidos:** Se eligieron 5 diferentes mapas de cantidad de ciudades distintas, las cuales están predeterminados para su acceso rápido.
3. **Menú de Resolver:** Aquí se encuentra el acceso a la subforma de solución a través de Algoritmo Genético
4. **Menú de Inmune:** En este menú se encuentran los dos procesos del algoritmo, la generación de vacunas y la expansión de la solución a el número de ciudades originales.
5. **Detener:** Detiene la ejecución del algoritmo.
6. **Estadísticas:** En este menú se encuentra la opción de grabar los datos actuales de la ejecución del algoritmo en la base de datos.
7. **Corridas:** Define cuantas veces ciclará el algoritmo completo, útil para obtener múltiples resultados con los parámetros especificados.
8. **Panel de Ciudades:** Este panel muestra las ciudades originales, y una vez vacunados, el repertorio de ciudades con las que trabaja el Algoritmo Genético.
9. **Panel de Vacunas:** En este panel se muestran las vacunas generadas a partir de las ciudades cargadas.
10. **Consola:** Despliega información útil como el archivo abierto, el número de iteraciones y valor de solución óptima en ese momento.

En la Figura 3.4 se presenta la captura de pantalla de la Interfaz para la ejecución del Algoritmo Genético. Como se puede observar los parámetros que pueden ser modificados son el tamaño de población, las generaciones máximas, el porcentaje de mutación y la semilla para ser utilizada por el generador de números aleatorios.

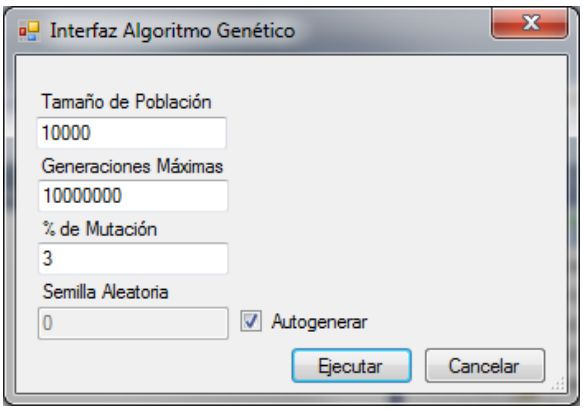


Figura 3.4: Captura de pantalla de la interfaz del Algoritmo Genético.

En la Figura 3.5 se presenta la Interfaz Generadora de Vacunas. Los parámetros que pueden ser modificados son el porcentaje de Número de Vacunas por generar (NV), los Elementos por Vacuna (EV) y entre la generación con Selector Aleatorio o Selector Elitista. Es importante mencionar que se optó por utilizar porcentajes para determinar el número de vacunas generar ya que resulta ser una medida más objetiva.

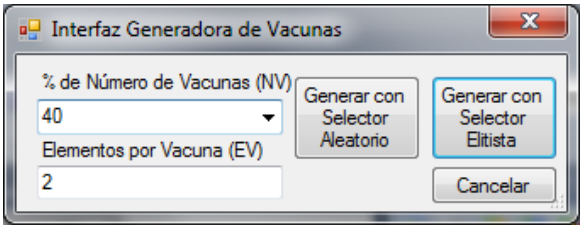


Figura 3.5: Captura de pantalla de la interfaz generadora de Vacunas.

4 Resultados Experimentales

En este capítulo se explica la metodología seguida en la experimentación. En términos generales se presentan tres experimentos que tiene como finalidad recaudar los datos necesarios para desarrollar el estudio comparativo. A su vez, cada experimento trabaja con cinco problemas que varían en el número de ciudades del TSP y que por consiguiente aumenta la dificultad en su solución. Posteriormente se presentan los resultados de estos experimentos y un estudio comparativo de los mismos es realizado donde se analizan los datos por cada caso de TSP.

4.1. Metodología

Debido a la implementación de la lectura de los archivos .tsp es posible utilizar para las pruebas cualquiera de los problemas que se encuentran en el repertorio del TSPLIB. Debido a que hay una gran cantidad de problemas, se decidió limitar los experimentos a cinco problemas diferentes con cantidad de ciudades que van desde 131 hasta 711. En la Tabla 4.1 se presentan los problemas seleccionados, el número de ciudades, su óptimo global y el nombre dado por el TSP, así como el nombre del archivo con su ruta óptima. Estos problemas representan un nivel de complejidad significativo en la solución del TSP.

De la Figura 4.1 a la Figura 4.5 se muestran las rutas óptimas para cada uno de los problemas elegidos.

Tabla 4.1: Desglose de los problemas de prueba.

ID	Nombre	# ciudades	Óptimo	N. Archivo	N. Archivo Ruta Opt.
1	xqf131	131	564	xqf131.tsp	xqf131.tour
2	xqg237	237	1019	xqg237.tsp	xqg237.tour
3	pbl395	395	1281	pbl395.tsp	pbl395.tour
4	pbm436	436	1443	pbm436.tsp	pbm436.tour
5	rbx711	711	3115	rbx711.tsp	rbx711.tour

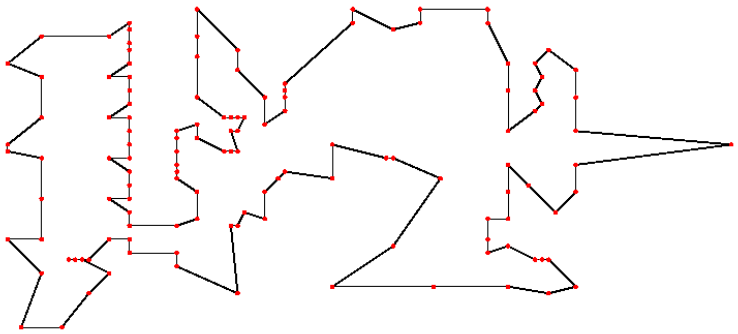


Figura 4.1: Ruta óptima para el TSP de 131 ciudades.

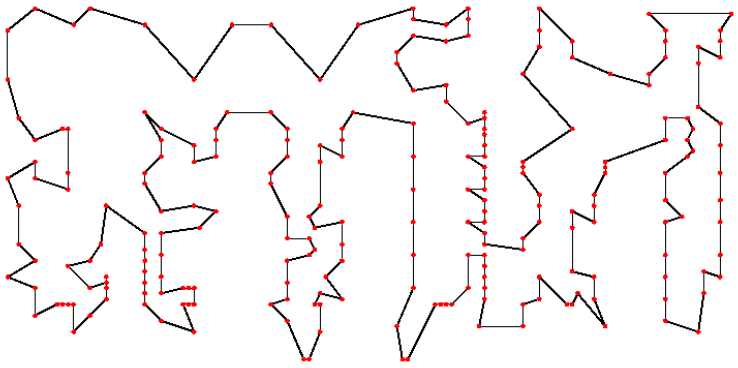


Figura 4.2: Ruta óptima para el TSP de 237 ciudades.

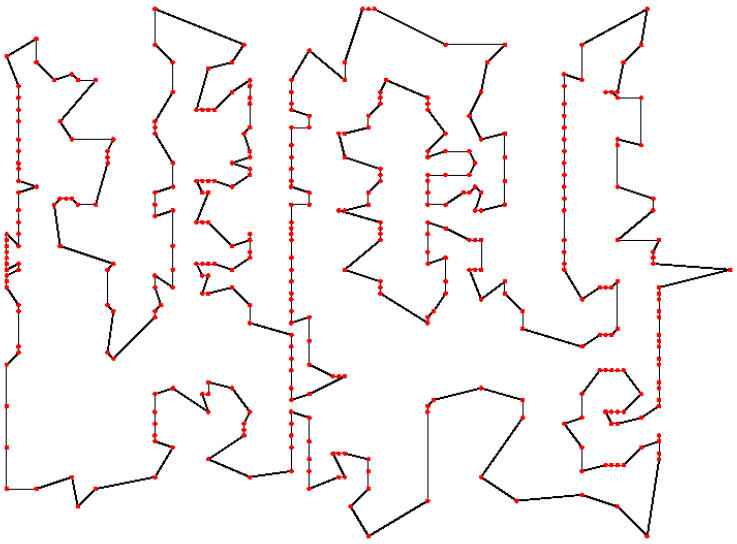


Figura 4.3: Ruta óptima para el TSP de 395 ciudades.

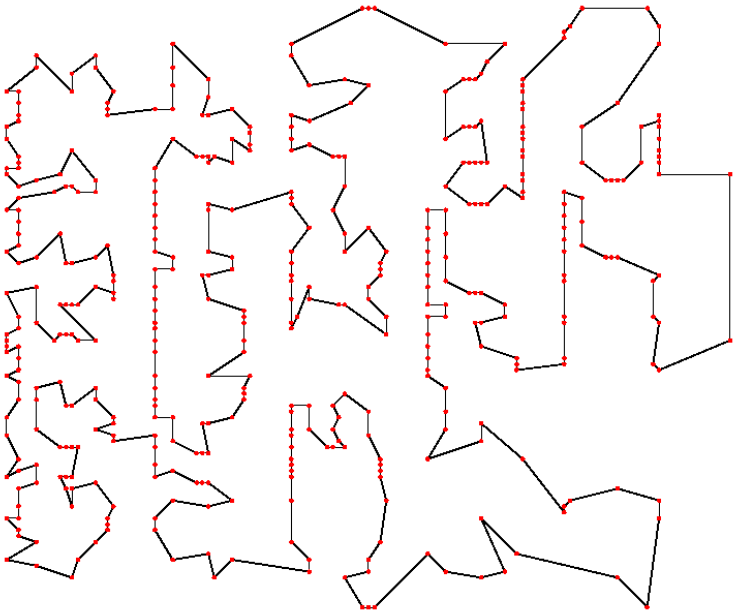


Figura 4.4: Ruta óptima para el TSP de 436 ciudades.

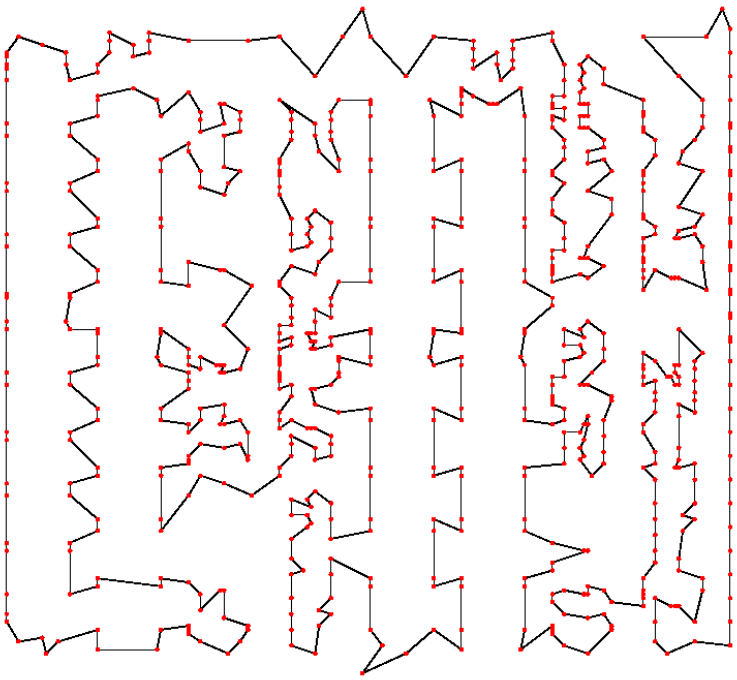


Figura 4.5: Ruta óptima para el TSP de 711 ciudades.

4.1.1. Algoritmo Genético

Una vez definido los problemas de prueba se procede a definir que algoritmo será utilizado para aplicar el proceso de vacunación como fue definido en la Figura 3.2. Se eligió utilizar un Algoritmo Genético clásico debido a las siguientes características:

- Algoritmo basado en población
- El concepto es fácil de comprender
- Parámetros ajustables
- Siempre nos proporciona un resultado, el cual mejora al seguir iterando
- Gran espacio de búsqueda

Se decidió por generar los propios datos estadísticos, de tal forma de que se tienen 20 soluciones generadas por el algoritmo genético para cada una de los mapas de ciudades. A su vez para probar el algoritmo tenemos 20 datos con 40 % de vacunación.

Los parámetros para el algoritmo genético se encuentran descritos en la Tabla 4.2.

Tabla 4.2: Parámetros de ejecución del Algoritmo Genético.

Parámetro	Valor
Tamaño de población	1,000
Generaciones	10,000,000
Porcentaje de Mutación	3 %
Semilla generadora de números aleatorios	Distinta cada ejecución

Es importante notar que estos parámetros se mantendrán consistentes a lo largo de todos los experimentos. Es necesario definirlo así para que los resultados de comparación sean válidos entre las distintas implementaciones del algoritmo de vacunación.

De los experimentos nos interesan varios factores que serán utilizados como punto de comparación. A continuación se enlistan:

1. Que el algoritmo efectivamente proporcione mejores soluciones al seguir iterando.
2. El valor de la ruta obtenida en distintas fases de ejecución y el valor final de la ruta obtenida por el algoritmo.
3. Punto de convergencia de los datos, i.e. el valor donde el algoritmo tiende y no donde ya no se avanza significativamente (menor al 1 % de avance).
4. Tiempos de ejecución de los algoritmos.

4.2. Pruebas realizadas con la plataforma experimental

Los experimentos están diseñados con la finalidad de obtener los datos necesarios para generar un estudio comparativo que lleve a conclusiones claras y sobre todo reproducibles. Debido a que el algoritmo utilizado es una metaheurística y los resultados entre una ejecución del algoritmo y la siguiente varían dado su misma naturaleza, todos los experimentos son repetidos 20 veces y se obtiene un promedio de los datos, los cuales se presentan tabular y gráficamente. Se espera que aún dado los elementos de aleatoriedad la respuesta del algoritmo sea similar. A continuación se presenta una breve explicación de los experimentos

- Experimento #1 - “Datos de Control”: Este experimento tiene como finalidad obtener los datos de control por medio de la ejecución del Algoritmo Genético sin ninguna

modificación.

- Experimento #2 - "Prueba de vacunación con Selector Aleatorio": En este experimento se explora el uso del algoritmo de vacunación con Selector Aleatorio como ayuda al Algoritmo Genético.
- Experimento #3 "Prueba de vacunación con Selector Elitista": En este experimento se explora el uso del algoritmo de vacunación con Selector Elitista como ayuda al Algoritmo Genético.

Finalmente, después de presentar los experimentos y sus resultados correspondientes, se procede con la comparación de los mismos desglosado por problema.

4.2.1. Experimento #1: Datos de control por medio del Algoritmo Genético sin Vacunación

El primer conjunto de experimentos tiene como finalidad obtener los datos de control que servirán como punto de partida para el análisis comparativo, lo cual en este caso consiste en ejecutar el Algoritmo Genético con los parámetros definidos anteriormente y observar los resultados. En la Tabla 4.3 se muestra el promedio de las rutas obtenidas desglosadas por porcentaje total de avance de iteraciones en incrementos de 5 % a partir del 10 %.

Tabla 4.3: Promedios de las diferentes generaciones con Algoritmo Genético.

	Ciudades				
Iteraciones	131	237	395	436	711
1,000,000	825.15	2491.95	3951.50	4902.53	15109.80
1,500,000	686.95	1855.40	3159.75	4138.16	13427.20
2,000,000	640.80	1626.95	2816.80	3650.47	12039.27
2,500,000	621.90	1492.95	2576.00	3351.74	10986.80
3,000,000	616.15	1404.25	2407.15	3119.16	10265.80
3,500,000	613.90	1341.50	2279.35	2924.21	9635.80
4,000,000	611.75	1294.00	2175.35	2786.11	9128.73
4,500,000	610.25	1256.35	2087.85	2675.32	8725.33
5,000,000	609.35	1235.40	2015.75	2567.37	8331.27
5,500,000	608.60	1214.25	1958.10	2474.58	7996.00
6,000,000	608.45	1202.15	1905.35	2398.11	7737.67
6,500,000	608.00	1190.95	1859.55	2337.16	7460.60
7,000,000	607.30	1181.98	1814.54	2276.43	7264.45
7,500,000	606.90	1178.50	1783.60	2228.21	7009.40
8,000,000	606.10	1173.95	1752.20	2183.39	6841.00
8,500,000	606.00	1169.20	1727.80	2146.00	6653.47
9,000,000	605.65	1166.40	1707.80	2114.63	6484.13
9,500,000	604.80	1163.55	1687.45	2079.89	6364.29
10,000,000	604.50	1161.40	1668.35	2053.47	6232.00

A su vez en la Tabla 4.4 se presenta que tan cercanas están las soluciones obtenidas por el Algoritmo Genético con respecto óptimo global de cada uno de los problemas. Se puede observar que entre más crece el problema en complejidad el Algoritmo Genético presenta soluciones significativamente más alejadas del óptimo global.

Tabla 4.4: Cuadro comparativo de la solución encontrada por el Algoritmo Genético contra el óptimo global.

Problema	131	237	395	436	711
Algoritmo Genético	604.50	1161.40	1668.35	2053.47	6232.00
Óptimo	564.00	1019.00	1281.00	1443.00	3115.00
Porcentaje para alcanzar el óptimo	6.70 %	12.26 %	23.22 %	29.73 %	50.02 %

De la Figura 4.6 a la Figura 4.10 se presentan las gráficas de la mejor solución obtenida por avance en las iteraciones por problema.

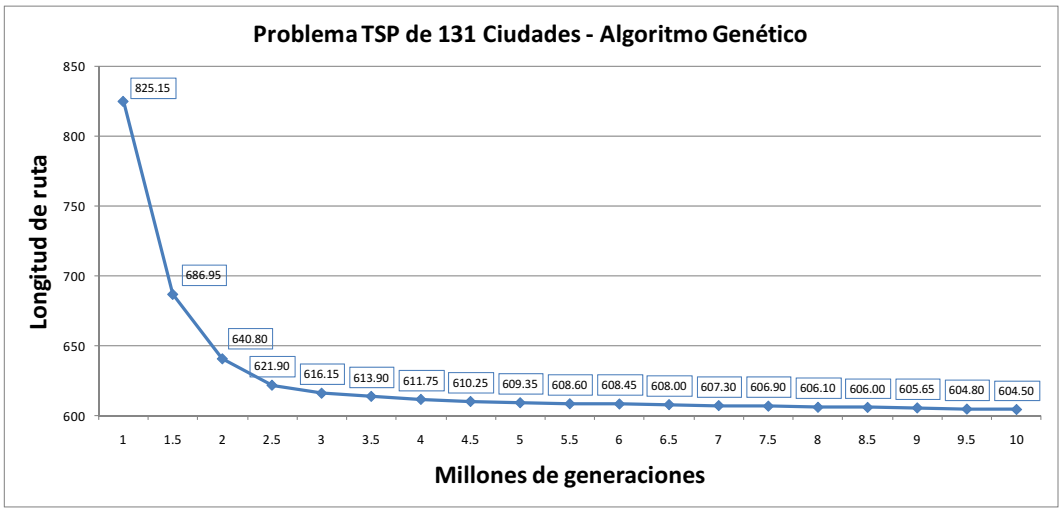


Figura 4.6: 19 series de promedios de longitudes de ruta contra incremento de generaciones, para el TSP de 131 ciudades, resuelto mediante Algoritmo Genético.

En la Figura 4.6 correspondiente al TSP de 131 ciudades podemos observar los resultados de promedio de las mejores rutas obtenidas, y particularmente nos interesa ver que a partir de la iteración 3,000,000 no existe una mejora significativa con las generaciones siguientes.

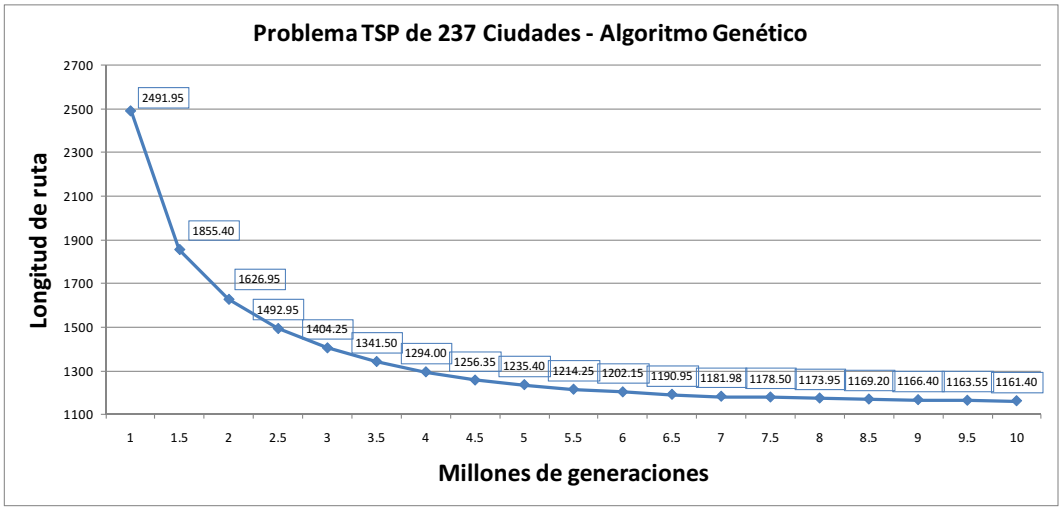


Figura 4.7: 19 series de promedios de longitudes de ruta contra incremento de generaciones, para el TSP de 237 ciudades, resuelto mediante Algoritmo Genético.

En la Figura 4.7 tenemos un comportamiento similar con el TSP de 237 ciudades, pero en este caso el comportamiento es observado en la generación 6,500,000.

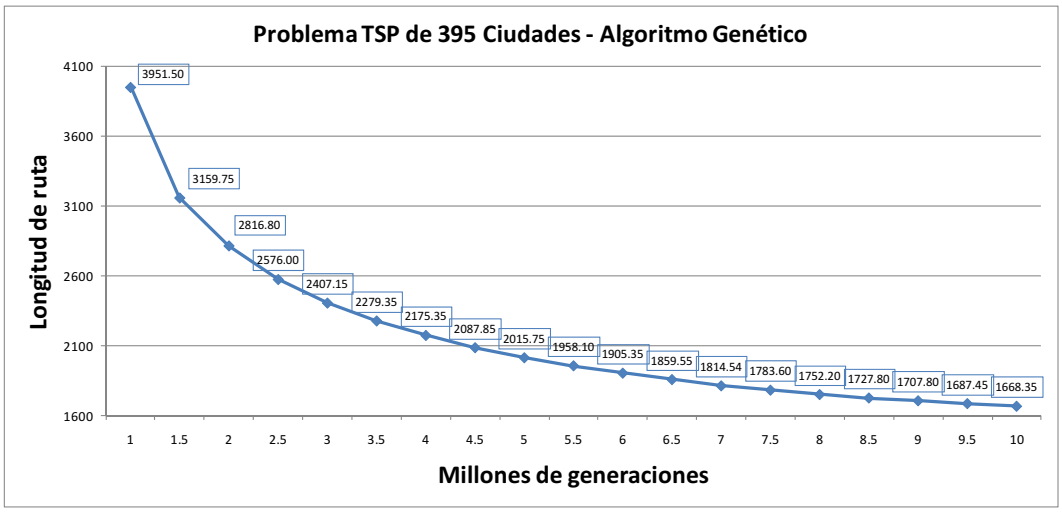


Figura 4.8: 19 series de promedios de longitudes de ruta contra incremento de generaciones, para el TSP de 395 ciudades, resuelto mediante Algoritmo Genético.

En la Figura 4.8 que corresponde al TSP de 395 no se presenta un comportamiento similar a los dos casos anteriores, ya que al llegar al límite de 10,000,000 de generaciones aún presenta mejoras significativas.

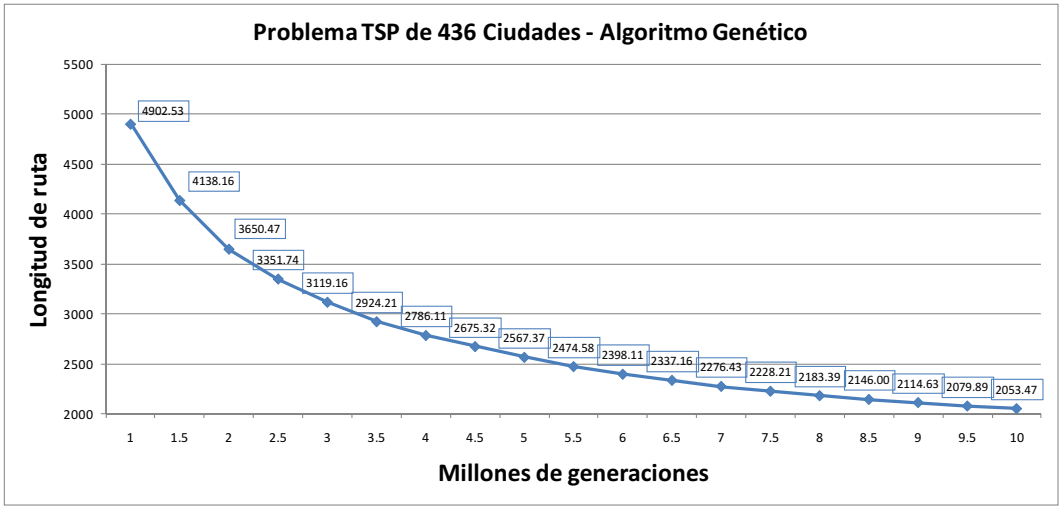


Figura 4.9: 19 series de promedios de longitudes de ruta contra incremento de generaciones, para el TSP de 436 ciudades, resuelto mediante Algoritmo Genético.

En la Figura 4.9 correspondiente al TSP de 436 se puede observar el mismo comportamiento que el caso anterior con el TSP de 395, ya que no se posible determinar un punto donde no existe avance significativo.

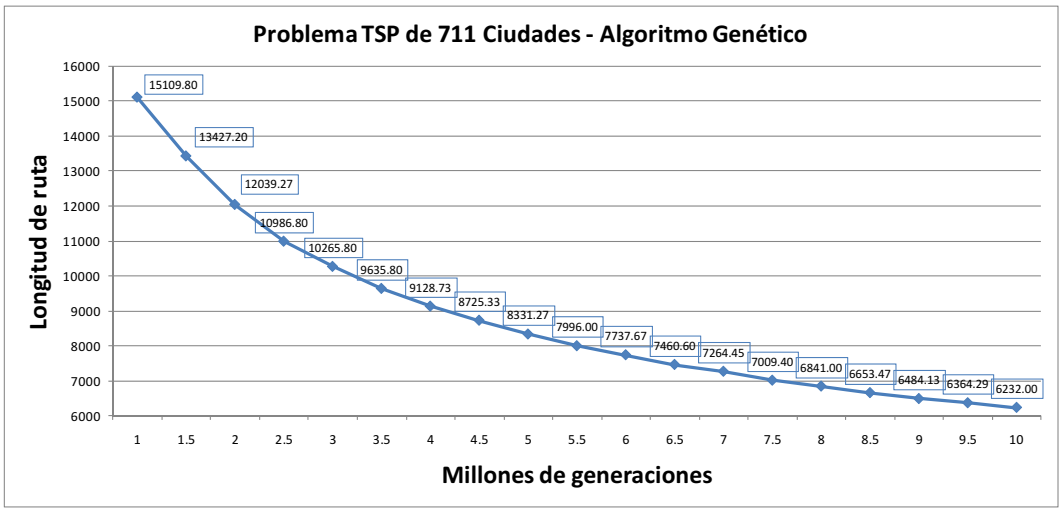


Figura 4.10: 19 series de promedios de longitudes de ruta contra incremento de generaciones, para el TSP de 711 ciudades, resuelto mediante Algoritmo Genético.

En la Figura 4.10 la cual pertenece al TSP de 711 es aún más notorio esta tendencia, y desde este momento podemos observar que el Algoritmo Genético requiere de significativamente más generaciones para proporcionar una respuesta adecuada. Esto se refuerza con el dato presentado en la Tabla 4.4 que nos dice que la mejor solución se encuentra a 50.02 % del óptimo.

Adicionalmente a las distancias de cada una de las rutas, la Figura 4.11 nos indica el siguiente parámetro a registrar el cual es el tiempo de ejecución del algoritmo.

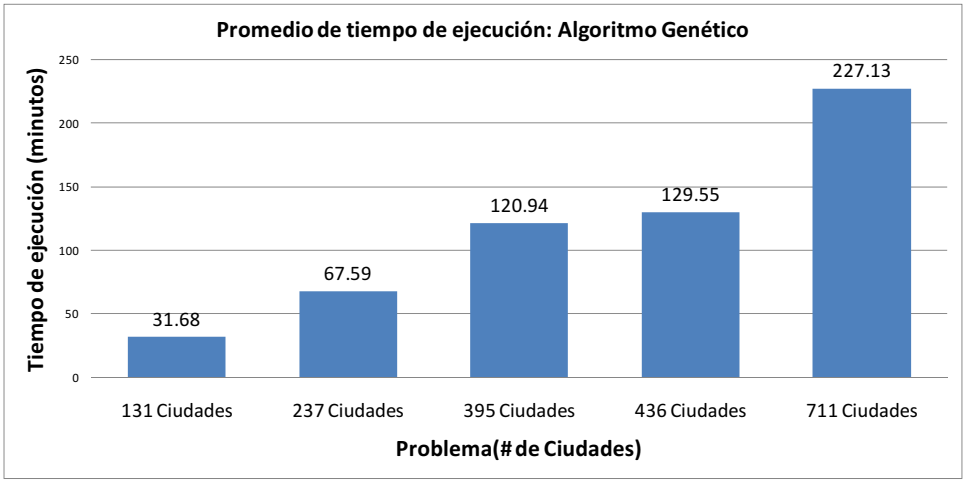


Figura 4.11: Gráfica de tiempo de ejecución del Algoritmo Genético.

Como es de esperarse, entre mayor sea el número de ciudades definidas para cada uno de los problemas, el Algoritmo Genético presenta un mayor tiempo de ejecución, independientemente de que el número de generaciones este fijo. Esto se debe principalmente a que el tamaño de cada uno de los elementos de la población es mayor, lo que produce matrices de distancias más grandes y operadores de cruzamiento más complejos.

4.2.2. Experimento #2: Prueba del Algoritmo de Vacunación con Selector Aleatorio

El segundo experimento consiste en utilizar el algoritmo generador de vacunas con Selector Aleatorio y aplicarlo al repertorio de ciudades con las cuales trabajará el algoritmo genético. Los parámetros de la vacunación son $NV = NC * 0.4$ y $EV = 2$.

En la Tabla 4.5 se muestra el promedio de las rutas obtenidas desglosadas por porcentaje total de avance de iteraciones en incrementos de 5 % a partir del 10 %.

A su vez en la Tabla 4.6 se presenta que tan cercanas están las soluciones obtenidas por la vacunación con Selector Aleatorio con respecto óptimo global de cada uno de los problemas. Se puede observar que entre más crece el problema en número de ciudades se presenta soluciones más alejadas del óptimo global con respecto al problema anterior, sin embargo de la misma manera entre más elementos tiene el TSP el algoritmo de vacunación con Selector Aleatorio nos proporciona mejores resultados que el Experimento #1.

Tabla 4.5: Promedios de las diferentes generaciones con vacunación mediante Selector Aleatorio.

	Ciudades				
Iteraciones	131	237	395	436	711
1,000,000	660.58	1537.05	2971.00	3602.65	10872.75
1,500,000	650.65	1451.60	2323.95	3136.15	9469.20
2,000,000	650.28	1356.45	2226.10	2830.80	8532.70
2,500,000	650.20	1303.60	2170.45	2632.05	7820.30
3,000,000	649.05	1276.15	2036.30	2463.05	7251.05
3,500,000	648.33	1261.55	1940.55	2333.85	6810.40
4,000,000	647.63	1252.85	1859.85	2231.50	6437.10
4,500,000	647.45	1248.30	1805.40	2141.55	6122.70
5,000,000	647.13	1242.30	1761.85	2074.70	5862.60
5,500,000	646.80	1241.60	1727.10	2030.95	5638.35
6,000,000	646.68	1239.40	1704.20	1990.20	5441.85
6,500,000	646.38	1238.80	1681.05	1958.40	5267.80
7,000,000	646.36	1237.15	1663.70	1934.40	5118.60
7,500,000	646.35	1235.30	1653.00	1910.95	4979.10
8,000,000	646.33	1234.10	1639.15	1893.90	4860.30
8,500,000	645.95	1233.00	1632.35	1884.05	4761.30
9,000,000	645.85	1233.25	1625.65	1876.30	4667.15
9,500,000	645.80	1232.45	1619.90	1868.40	4597.80
10,000,000	645.63	1231.50	1615.60	1863.80	4527.55

Tabla 4.6: Cuadro comparativo de la solución encontrada por la vacunación con Selector Aleatorio contra el óptimo global.

Problema	131	237	395	436	711
Vacunación con Selector Aleatorio	645.63	1231.50	1615.60	1863.80	4527.55
Óptimo	564.00	1019.00	1281.00	1443.00	3115.00
Porcentaje para alcanzar el óptimo	12.64 %	17.26 %	20.71 %	22.58 %	31.20 %

De la Figura 4.12 a la Figura 4.16 se presentan las gráficas de la mejor solución obtenida por avance en las iteraciones por problema con el algoritmo de vacunación con SA.

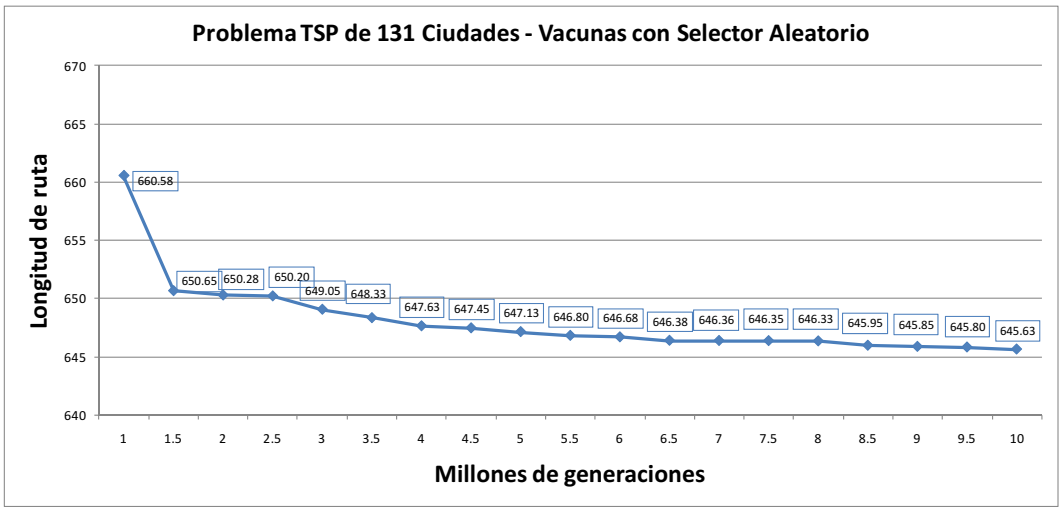


Figura 4.12: 19 series de promedios de longitudes de ruta contra incremento de generaciones, para el TSP de 131 ciudades, resuelto mediante Selector Aleatorio.

En la Figura 4.12 observamos los datos referentes al TSP de 131 ciudades. Podemos observar que la solución obtenida en la iteración 2,000,000 se encuentra muy cercano a la mejor solución obtenida en la iteración 10,000,000, lo cual indica que no hubo avances significativos al iterar más allá de 2 millones. La solución proporcionada se encuentra a 12.64 % del óptimo global.

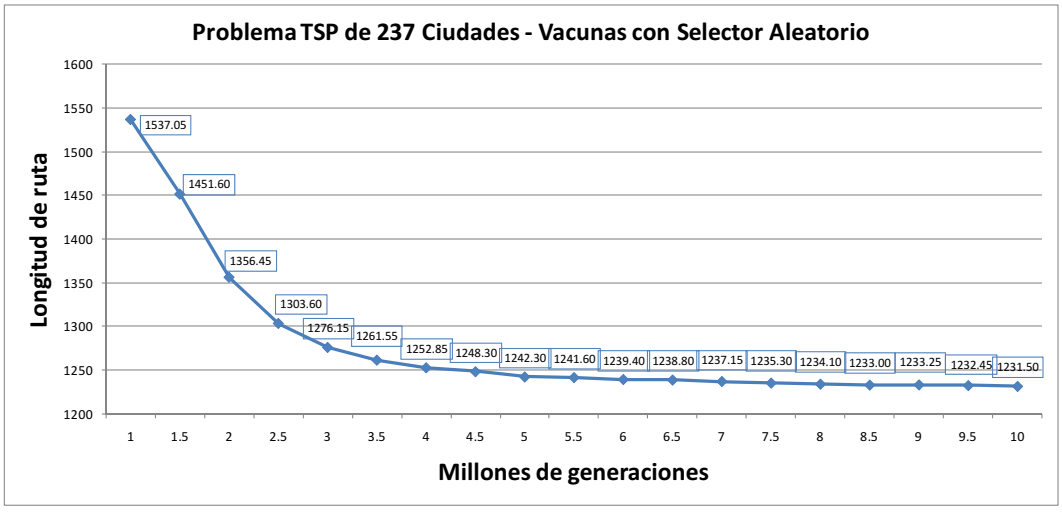


Figura 4.13: 19 series de promedios de longitudes de ruta contra incremento de generaciones, para el TSP de 237 ciudades, resuelto mediante Selector Aleatorio.

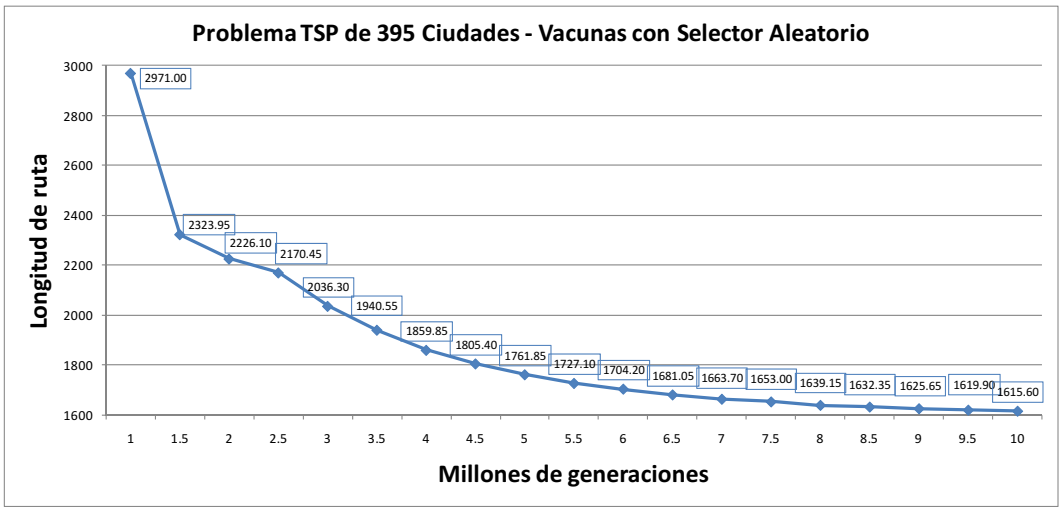


Figura 4.14: 19 series de promedios de longitudes de ruta contra incremento de generaciones, para el TSP de 395 ciudades, resuelto mediante Selector Aleatorio.

La Figura 4.13 proporciona la gráfica de los datos correspondientes al TSP de 237 ciudades y en esta podemos observar que el algoritmo deja de avanzar significativamente en la iteración 4,000,000. La solución proporcionada se encuentra a 17.26 % del óptimo global.

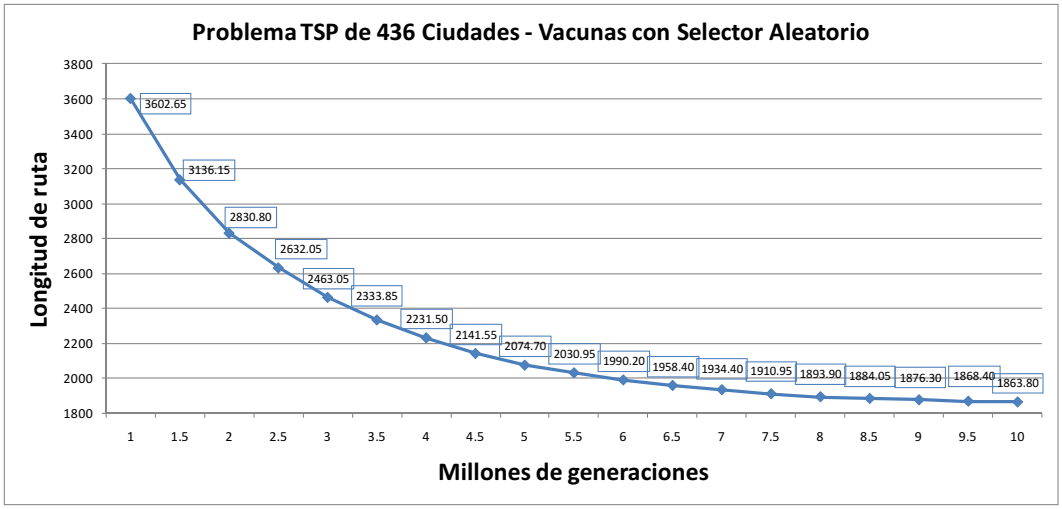


Figura 4.15: 19 series de promedios de longitudes de ruta contra incremento de generaciones, para el TSP de 436 ciudades, resuelto mediante Selector Aleatorio.

En la Figura 4.14 correspondiente al TSP de 395 ciudades sí se observa el punto de convergencia y este se encuentra localizado en la iteración 7,500,000. La solución proporcionada se encuentra a 20.71 % del óptimo global.

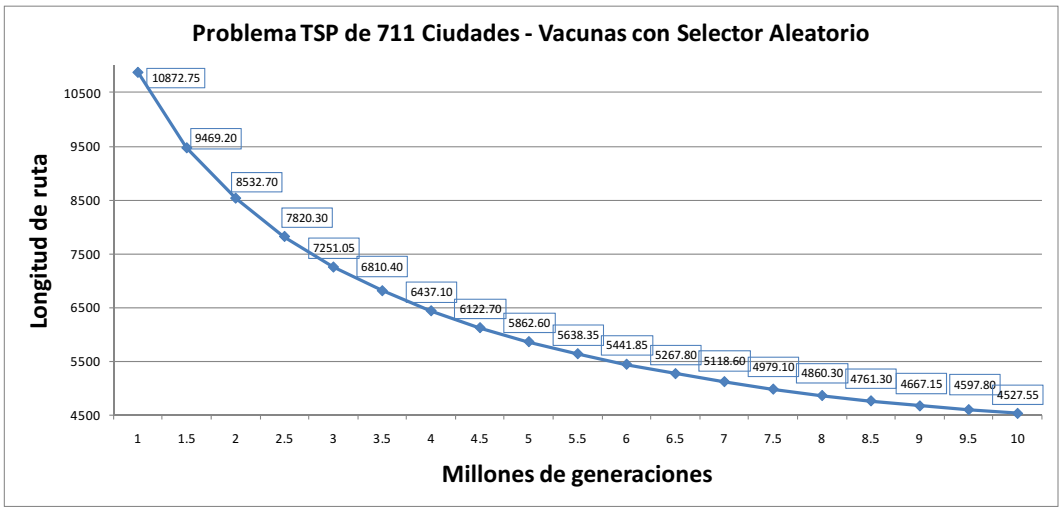


Figura 4.16: 19 series de promedios de longitudes de ruta contra incremento de generaciones, para el TSP de 711 ciudades, resuelto mediante Selector Aleatorio.

En la Figura 4.15 correspondiente al TSP de 436 ciudades se sigue observando el punto de convergencia y este se encuentra en la iteración 8,000,000. La solución proporcionada se encuentra a 22.58 % del óptimo global.

En la Figura 4.16 correspondiente al TSP de 711 ciudades no es posible localizar un punto de convergencia ya que de seguir iterando seguiría dando resultados por encima del 1 % de avance. La solución proporcionada se encuentra a 31.20 % del óptimo global.

En la Figura 4.17 una vez más se presentan los tiempos de ejecución pero ahora del algoritmo de vacunación con SA.

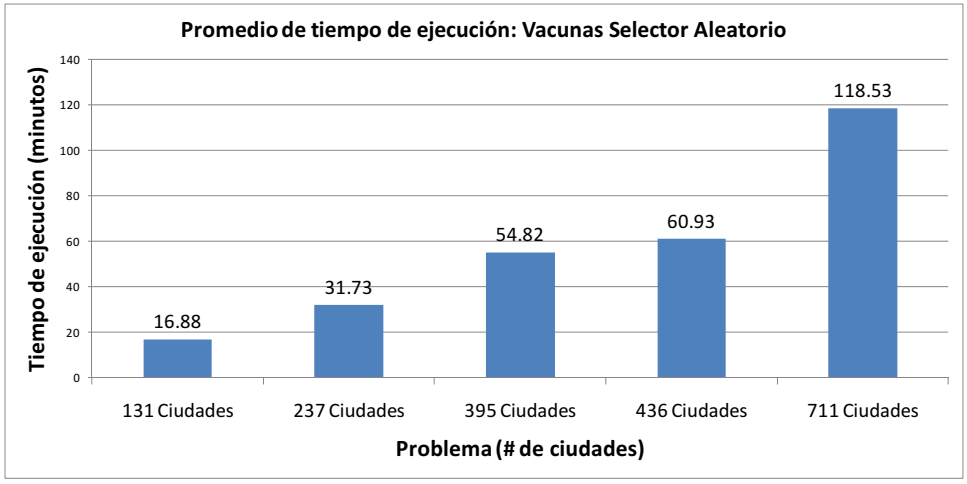


Figura 4.17: Gráfica de tiempo de ejecución con vacunas mediante Selector Aleatorio.

Tenemos la misma tendencia en este experimento que en el experimento #1 ya que al aumentar el número de ciudades los tiempos de ejecución aumentan para el algoritmo de vacunación con Selector Aleatorio.

4.2.3. Experimento #3: Prueba del Algoritmo de Vacunación con Selector Elitista

El tercer experimento consiste en emplear el algoritmo generador de vacunas con Selector Elitista y aplicarlo al repertorio de ciudades con las cuales trabajará el algoritmo genético. Los parámetros de la vacunación son $NV = NC * 0.4$ y $EV = 2$.

En la Tabla 4.7 se muestra el promedio de las rutas obtenidas desglosadas por porcentaje total de avance de iteraciones en incrementos de 5 % a partir del 10 %.

Tabla 4.7: Promedios de las diferentes generaciones con vacunación mediante Selector Elitista.

	Ciudades				
Iteraciones	131	237	395	436	711
1,000,000	629.75	1488.67	2660.35	3158.55	9688.90
1,500,000	622.00	1324.14	2230.00	2771.35	8532.40
2,000,000	619.95	1247.38	2125.55	2516.65	7740.30
2,500,000	619.10	1206.10	1973.15	2338.95	7151.95
3,000,000	618.40	1184.95	1862.55	2203.40	6672.70
3,500,000	616.30	1172.81	1783.90	2099.55	6288.85
4,000,000	615.40	1169.05	1725.70	2022.35	5965.15
4,500,000	615.40	1165.95	1680.30	1956.50	5696.90
5,000,000	615.40	1164.48	1645.25	1908.80	5466.05
5,500,000	614.95	1162.67	1615.80	1872.45	5273.70
6,000,000	614.60	1161.76	1597.70	1842.05	5099.20
6,500,000	614.10	1160.48	1584.00	1824.85	4938.15
7,000,000	613.35	1159.30	1571.40	1794.70	4818.20
7,500,000	613.05	1157.81	1563.00	1794.50	4705.75
8,000,000	612.85	1156.86	1556.35	1781.75	4619.65
8,500,000	612.80	1156.00	1547.00	1770.80	4527.50
9,000,000	612.65	1153.76	1541.50	1763.15	4451.75
9,500,000	612.35	1151.57	1537.35	1756.80	4377.35
10,000,000	612.25	1151.19	1533.75	1750.15	4306.65

A su vez en la Tabla 4.8 se presenta que tan cercanas están las soluciones obtenidas por la vacunación con Selector Elitista con respecto óptimo global de cada uno de los problemas. Se puede observar que entre más crece el problema en número de ciudades se presenta soluciones más alejadas del óptimo global con respecto al problema anterior, sin embargo de la misma manera entre más elementos tiene el TSP el algoritmo de vacunación con Selector Elitista nos proporciona mejores resultados que el Experimento #1.

Tabla 4.8: Cuadro comparativo de la solución encontrada por la vacunación con Selector Elitista contra el óptimo global.

Problema	131	237	395	436	711
Vacunación con Selector Elitista	612.25	1151.19	1533.75	1750.15	4306.65
Óptimo	564.00	1019.00	1281.00	1443.00	3115.00
Porcentaje para alcanzar el óptimo	7.88 %	11.48 %	16.48 %	17.55 %	27.67 %

De la Figura 4.18 a la Figura 4.22 se presentan las gráficas de la mejor solución obtenida por avance en las iteraciones por problema con el algoritmo de vacunación con SE.

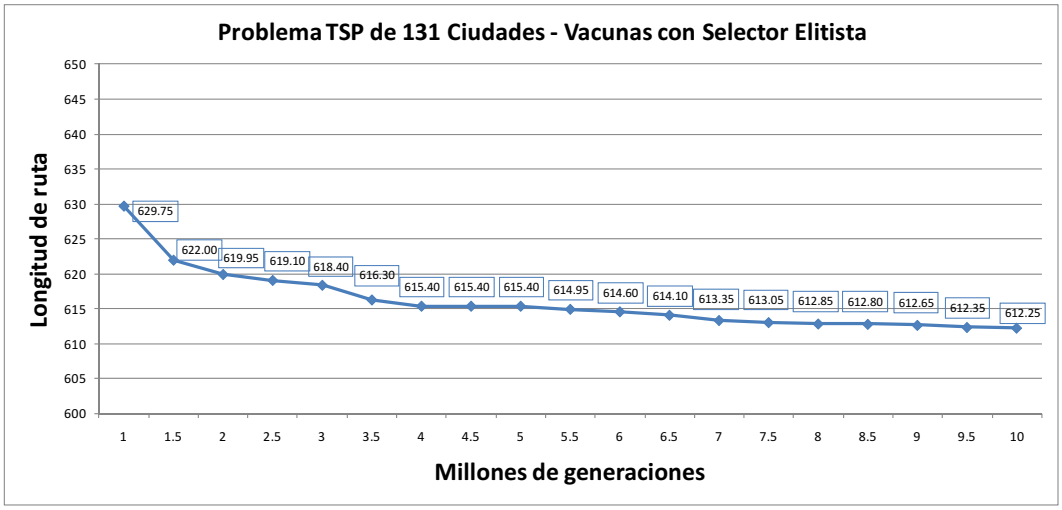


Figura 4.18: 19 series de promedios de longitudes de ruta contra incremento de generaciones, para el TSP de 131 ciudades, resuelto mediante Selector Elitista.

En la Figura 4.18 observamos los datos referentes al TSP de 131 ciudades. Podemos observar que la solución obtenida en la iteración 2,000,000 se encuentra muy cercano a la mejor solución obtenida en la iteración 10,000,000, lo cual indica que no hubo avances significativos

al iterar más allá de 2 millones. La solución proporcionada se encuentra a 7.88 % del óptimo global.

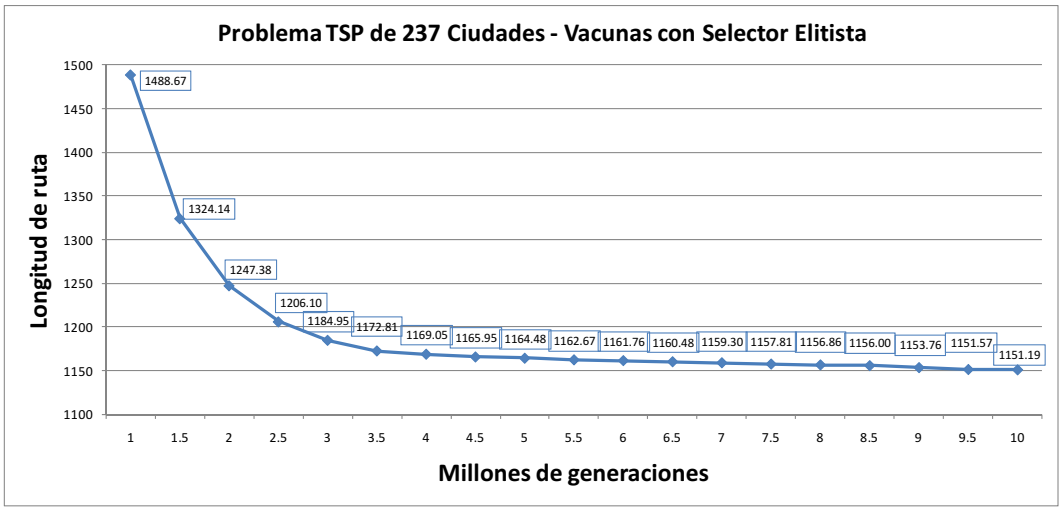


Figura 4.19: 19 series de promedios de longitudes de ruta contra incremento de generaciones, para el TSP de 237 ciudades, resuelto mediante Selector Elitista.

La Figura 4.19 proporciona la gráfica de los datos correspondientes al TSP de 237 ciudades y en esta podemos observar que el algoritmo deja de avanzar significativamente en la iteración 4,000,000. La solución proporcionada se encuentra a 11.48 % del óptimo global.

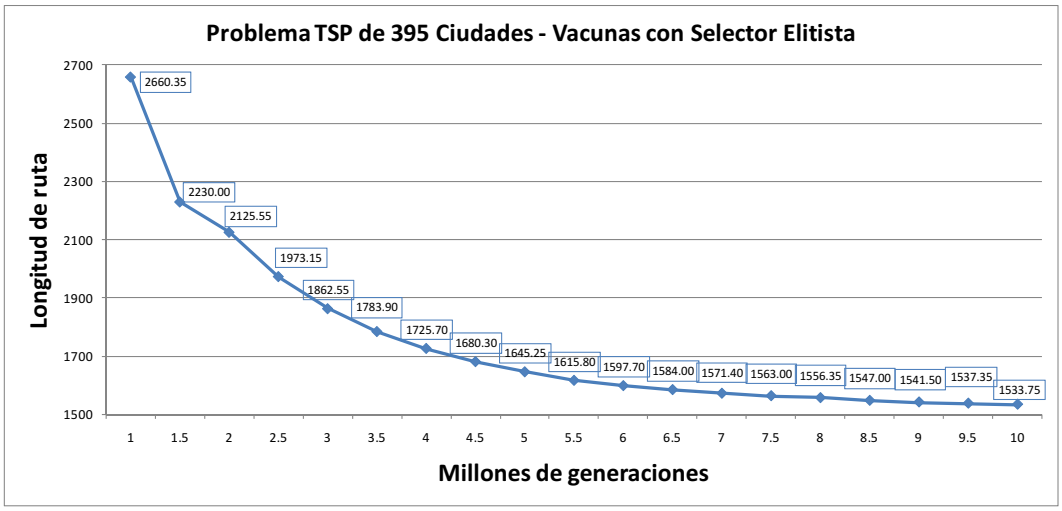


Figura 4.20: 19 series de promedios de longitudes de ruta contra incremento de generaciones, para el TSP de 395 ciudades, resuelto mediante Selector Elitista.

En la Figura 4.20 correspondiente al TSP de 395 ciudades sí se observa el punto de conver-

gencia y este se encuentra localizado en la iteración 6,500,000. La solución proporcionada se encuentra a 16.48 % del óptimo global.

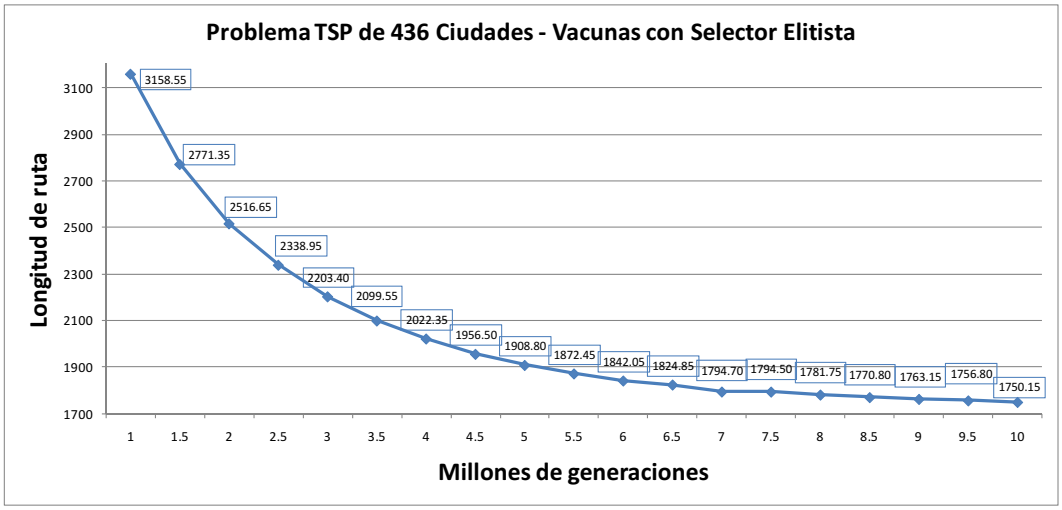


Figura 4.21: 19 series de promedios de longitudes de ruta contra incremento de generaciones, para el TSP de 436 ciudades, resuelto mediante Selector Elitista.

En la Figura 4.21 correspondiente al TSP de 436 ciudades se sigue observando el punto de convergencia y este se encuentra en la iteración 6,500,000. La solución proporcionada se encuentra a 17.55 % del óptimo global.

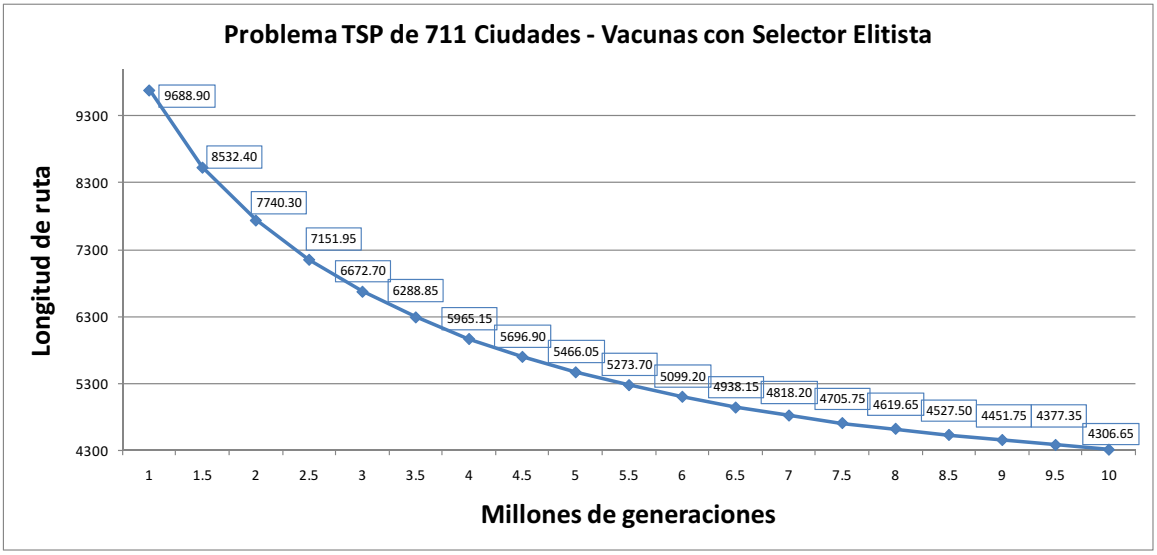


Figura 4.22: 19 series de promedios de longitudes de ruta contra incremento de generaciones, para el TSP de 711 ciudades, resuelto mediante Selector Elitista.

En la Figura 4.22 correspondiente al TSP de 711 ciudades no es posible localizar un punto de convergencia ya que de seguir iterando seguiría dando resultados por encima del 1 % de avance. La solución proporcionada se encuentra a 27.67 % del óptimo global.

En la Figura 4.23 una vez más se presentan los tiempos de ejecución pero ahora del algoritmo de vacunación con SE.

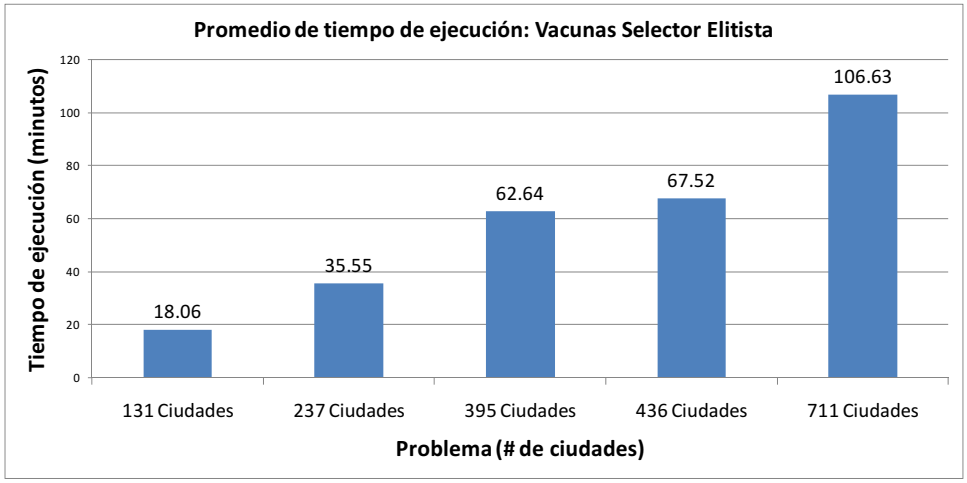


Figura 4.23: Gráfica de tiempo de ejecución con vacunas mediante Selector Elitista.

De manera similar al experimento #2 tenemos la misma tendencia en este experimento ya que al aumentar el número de ciudades los tiempos de ejecución aumentan para el algoritmo de vacunación con Selector Elitista.

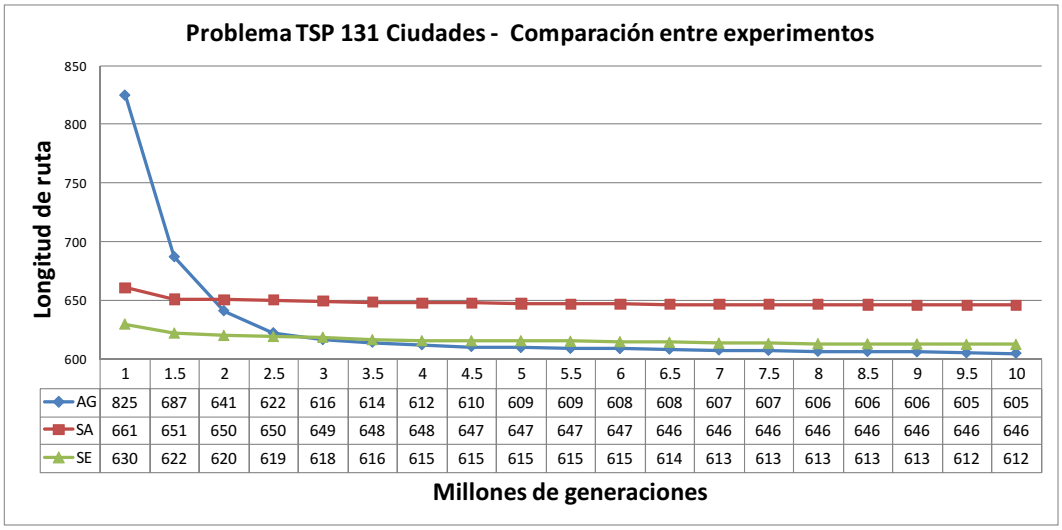
4.3. Estudio comparativo

A continuación se procede a realizar un estudio comparativo de los experimentos realizados. Como se mencionó en la sección de metodología se busca estudiar los siguientes puntos:

1. Que el algoritmo efectivamente proporcione mejores soluciones al seguir iterando
2. El valor de la ruta obtenido en distintas fases de ejecución y el valor final de la ruta obtenida por el algoritmo.
3. Punto de convergencia de los datos, i.e. el valor donde el algoritmo tiende y no donde ya no se avanza significativamente (menor al 1 % de avance).

4. Tiempos de ejecución de los algoritmos

4.3.1. Caso #1: TSP 131 ciudades



	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
AG	825	687	641	622	616	614	612	610	609	609	608	608	607	607	606	606	606	605	605
SA	661	651	650	650	649	648	648	647	647	647	647	646	646	646	646	646	646	646	646
SE	630	622	620	619	618	616	615	615	615	615	615	614	613	613	613	613	613	612	612

Figura 4.24: Caso TSP de 131 ciudades.- Comparación de promedios de longitudes de ruta contra incremento de generaciones entre AG, SA y SE.

En la Figura 4.24 podemos observar una gráfica donde se comparan los tres experimentos correspondientes al Algoritmo Genético, a la vacunación con Selector Aleatorio y a la vacunación con Selector Elitista. De la Tabla 4.9 podemos observar que el punto de convergencia del Algoritmo Genético es obtenido en la iteración 3,000,000 y en los algoritmos de vacunación es alcanzado de manera más rápida en la iteración 2,000,000, este valor nos indica que a partir de estas iteraciones el Algoritmo no será capaz de proporcionar soluciones significativamente mejores a las anteriores lo que puede utilizarse como un criterio de paro para el algoritmo. Sin embargo en el caso de los algoritmos de vacunación tanto con SA como con SE los valores finales resultan ser peores que el Algoritmo Genético con una diferencia de *6.37%* y *1.27%* respectivamente.

Tabla 4.9: Cuadro comparativo de los porcentajes de mejoramiento en la solución con respecto a la generación pasada para el TSP de 131 ciudades. Se indica el porcentaje del valor de convergencia.

	Algoritmo					
Iteraciones	AG		SA		SE	
1,000,000	825.15	0 %	660.58	0 %	629.75	0 %
1,500,000	686.95	16.75 %	650.65	1.50 %	622.00	1.23 %
2,000,000	640.80	6.72 %	650.28	**0.06 %**	619.95	**0.33 %**
2,500,000	621.90	2.95 %	650.20	0.01 %	619.10	0.14 %
3,000,000	616.15	**0.92 %**	649.05	0.18 %	618.40	0.11 %
3,500,000	613.90	0.37 %	648.33	0.11 %	616.30	0.34 %
4,000,000	611.75	0.35 %	647.63	0.11 %	615.40	0.15 %
4,500,000	610.25	0.25 %	647.45	0.03 %	615.40	0.00 %
5,000,000	609.35	0.15 %	647.13	0.05 %	615.40	0.00 %
5,500,000	608.60	0.12 %	646.80	0.05 %	614.95	0.07 %
6,000,000	608.45	0.02 %	646.68	0.02 %	614.60	0.06 %
6,500,000	608.00	0.07 %	646.38	0.05 %	614.10	0.08 %
7,000,000	607.30	0.12 %	646.36	0.00 %	613.35	0.12 %
7,500,000	606.90	0.07 %	646.35	0.00 %	613.05	0.05 %
8,000,000	606.10	0.13 %	646.33	0.00 %	612.85	0.03 %
8,500,000	606.00	0.02 %	645.95	0.06 %	612.80	0.01 %
9,000,000	605.65	0.06 %	645.85	0.02 %	612.65	0.02 %
9,500,000	604.80	0.14 %	645.80	0.01 %	612.35	0.05 %
10,000,000	604.50	0.05 %	645.63	0.03 %	612.25	0.02 %

En la Figura 4.25 se presentan los resultados del promedio del tiempo de ejecución de los tres algoritmos y en ella podemos observar que existe una mejora significativa con la vacunación. Si comparamos el tiempo de ejecución requerido por los algoritmos de vacunación tenemos una disminución del *46.7 %* con el Selector Aleatorio y un *42.9 %* con el Selector Elitista. Esto se debe a que el número efectivo de términos con el que trabaja el algoritmo genético es de:

$$Términos = NC - (NV * EV) + NV$$

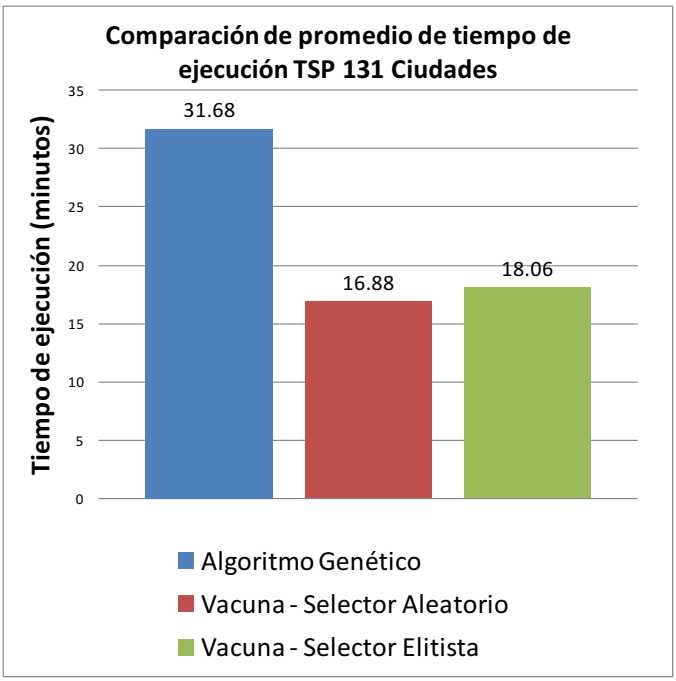


Figura 4.25: Comparación del tiempo de ejecución del TSP 131 ciudades entre AG, SA y SE.

Esta fórmula nos dice que, por cada vacuna se deben de contar los elementos que engloba, y estos se restan al número de ciudades, además después se agrega a este listado el número de vacunas ya que recordemos que se deben representar las ciudades agrupadas por un nuevo centro definido por la vacuna. En la Tabla 4.10 se muestra los valores finales después de aplicar vacunación del 40 % a los problemas. Con los parámetros definidos, se obtiene una reducción efectiva del 40 % del total de ciudades. En el TSP de 131 ciudades los algoritmos de vacunación al 40 % del número de ciudades nos deja con 79 elementos.

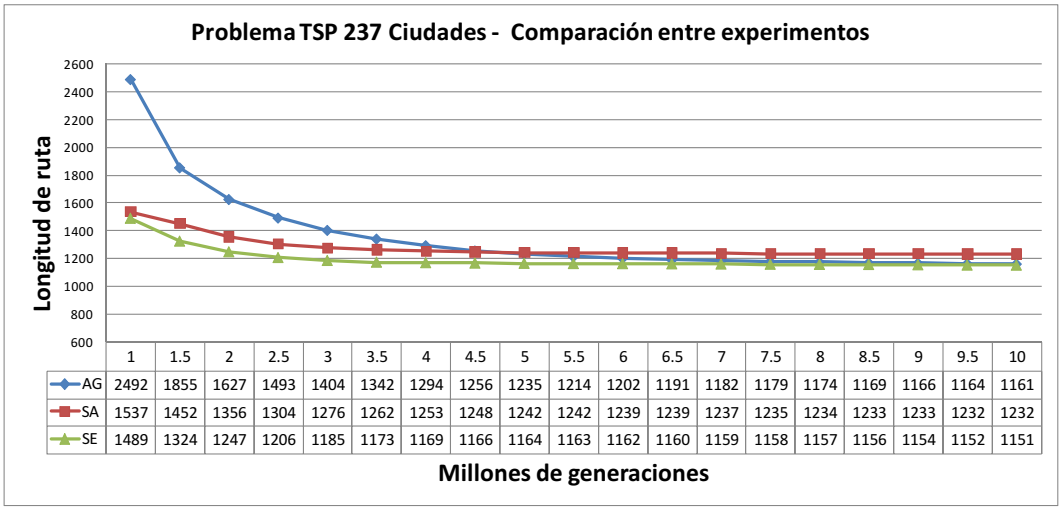
Tabla 4.10: Número de elementos después de aplicar la vacunación, $NV = NC * 0.4$ y $CV = 2$

Problema	Fórmula	Elementos finales
TSP 131	131-((131*.4)*2)+(131*.4)	79
TSP 237	237-((237*.4)*2)+(237*.4)	142
TSP 395	131-((395*.4)*2)+(395*.4)	237
TSP 436	131-((436*.4)*2)+(436*.4)	256
TSP 711	131-((711*.4)*2)+(711*.4)	427

Esta reducción afecta directamente los tiempos de ejecución al proporcionar al algoritmo

con listados significativamente más cortos de ciudades, lo cual afecta operaciones tales como cálculo de matrices de distancias, operaciones de ordenamiento y cálculos en general sobre el listado de elementos.

4.3.2. Caso #2: TSP 237 ciudades



	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
AG	2492	1855	1627	1493	1404	1342	1294	1256	1235	1214	1202	1191	1182	1179	1174	1169	1166	1164	1161
SA	1537	1452	1356	1304	1276	1262	1253	1248	1242	1242	1239	1239	1237	1235	1234	1233	1233	1232	1232
SE	1489	1324	1247	1206	1185	1173	1169	1166	1164	1163	1162	1160	1159	1158	1157	1156	1154	1152	1151

Figura 4.26: Caso TSP de 237 ciudades.- Comparación de promedios de longitudes de ruta contra incremento de generaciones entre AG, SA y SE.

Similar al caso anterior con el TSP de 131 ciudades en la Figura 4.26 se presenta la gráfica comparativa del TSP de 237 ciudades para el AG, SA, y el SE. En la Tabla 4.11 se presenta la tabla de porcentaje de mejora entre soluciones correspondiente a este problema del cual podemos observar que el punto de convergencia del AG es en la iteraciones 6,500,000 mientras que para el SA y SE es en la iteraciones 4,000,000, lo cual nos indica que se sigue presentando la tendencia de aproximarse al mejor valor más rápido que con el AG solo. Ahora bien con respecto a los valores finales que proporciona cada algoritmo tenemos que a diferencia del caso anterior de 131 ciudades el SE presenta una mejor solución que el AG en un 0.89 %, mientras que el SA sigue proporcionando una solución más larga en un 5.69 %.

Tabla 4.11: Cuadro comparativo de los porcentajes de mejoramiento en la solución con respecto a la generación pasada para el TSP de 237 ciudades. Se indica el porcentaje del valor de convergencia.

	Algoritmo					
Iteraciones	AG		SA		SE	
1,000,000	2491.95	0 %	1537.05	0 %	1488.67	0 %
1,500,000	1855.40	25.54 %	1451.60	5.56 %	1324.14	11.05 %
2,000,000	1626.95	12.31 %	1356.45	6.55 %	1247.38	5.80 %
2,500,000	1492.95	8.24 %	1303.60	3.90 %	1206.10	3.31 %
3,000,000	1404.25	5.94 %	1276.15	2.11 %	1184.95	1.75 %
3,500,000	1341.50	4.47 %	1261.55	1.14 %	1172.81	1.02 %
4,000,000	1294.00	3.54 %	1252.85	**0.69 %**	1169.05	**0.32 %**
4,500,000	1256.35	2.91 %	1248.30	0.36 %	1165.95	0.26 %
5,000,000	1235.40	1.67 %	1242.30	0.48 %	1164.48	0.13 %
5,500,000	1214.25	1.71 %	1241.60	0.06 %	1162.67	0.16 %
6,000,000	1202.15	1.00 %	1239.40	0.18 %	1161.76	0.08 %
6,500,000	1190.95	**0.93 %**	1238.80	0.05 %	1160.48	0.11 %
7,000,000	1181.98	0.75 %	1237.15	0.13 %	1159.30	0.10 %
7,500,000	1178.50	0.29 %	1235.30	0.15 %	1157.81	0.13 %
8,000,000	1173.95	0.39 %	1234.10	0.10 %	1156.86	0.08 %
8,500,000	1169.20	0.40 %	1233.25	0.07 %	1156.00	0.07 %
9,000,000	1166.40	0.24 %	1233.00	0.02 %	1153.76	0.19 %
9,500,000	1163.55	0.24 %	1232.45	0.04 %	1151.57	0.19 %
10,000,000	1161.40	0.18 %	1231.50	0.08 %	1151.19	0.03 %

En la Figura 4.27 se presentan los resultados del promedio del tiempo de ejecución de los 3 algoritmos y en ella podemos observar una vez más que existe una mejora significativa con la vacunación. Si comparamos el tiempo de ejecución requerido por los algoritmos de vacunación tenemos una disminución del *46.94* % con el Selector Aleatorio y de *52.59 %* con el Selector Elitista.

Retomando la Tabla 4.10 observamos que el algoritmo de vacunación reduce los términos de 237 ciudades a 142 elementos, lo cual produce una reducción en los tiempos de ejecución.

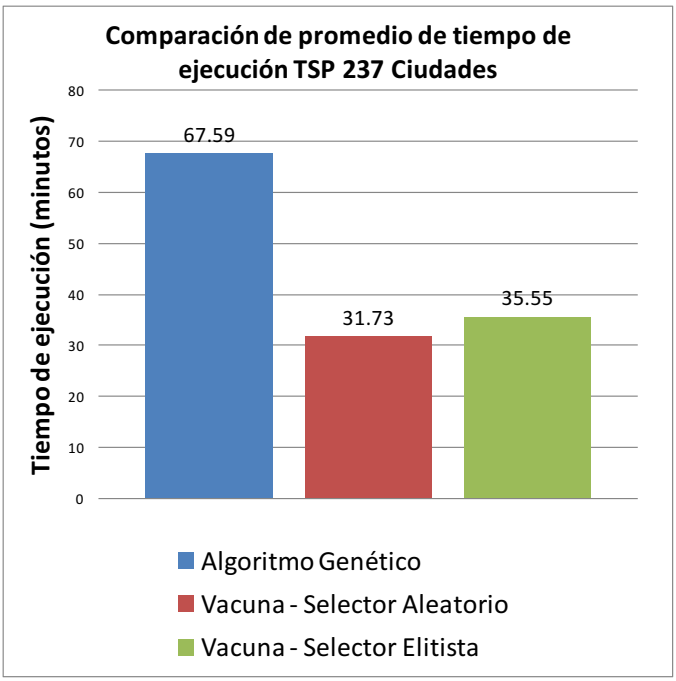
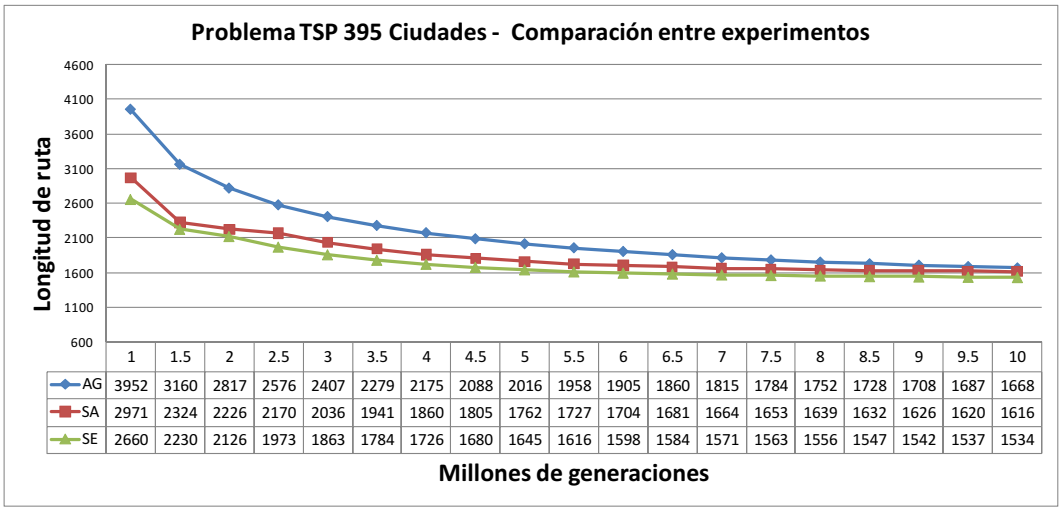


Figura 4.27: Comparación del tiempo de ejecución del TSP 237 ciudades entre AG, SA y SE.

4.3.3. Caso #3: TSP 395 ciudades



	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
AG	3952	3160	2817	2576	2407	2279	2175	2088	2016	1958	1905	1860	1815	1784	1752	1728	1708	1687	1668
SA	2971	2324	2226	2170	2036	1941	1860	1805	1762	1727	1704	1681	1664	1653	1639	1632	1626	1620	1616
SE	2660	2230	2126	1973	1863	1784	1726	1680	1645	1616	1598	1584	1571	1563	1556	1547	1542	1537	1534

Figura 4.28: Caso TSP de 395 ciudades.- Comparación de promedios de longitudes de ruta contra incremento de generaciones entre AG, SA y SE.

En la Figura 4.28 se presenta la gráfica de comparación de los resultados sobre el TSP 395 ciudades entre el AG, SA y SE. En esta gráfica podemos observar de una manera más clara a diferencia a los otros casos la tendencia en la mejora de los datos entre las generaciones de los diferentes algoritmos. Debido a que el no cambiar la manera en que funciona el algoritmo base (el Algoritmo Genético en este caso) es uno de los puntos principales del algoritmo de vacunación, esto es muy importante de mostrar.

Tabla 4.12: Cuadro comparativo de los porcentajes de mejoramiento en la solución con respecto a la generación pasada para el TSP de 395 ciudades. Se indica el porcentaje del valor de convergencia.

	Algoritmo					
Iteraciones	AG		SA		SE	
1,000,000	3951.50	0 %	2971.00	0 %	2660.35	0 %
1,500,000	3159.75	20.04 %	2323.95	21.78 %	2230.00	16.18 %
2,000,000	2816.80	10.85 %	2226.10	4.21 %	2125.55	4.68 %
2,500,000	2576.00	8.55 %	2170.45	2.50 %	1973.15	7.17 %
3,000,000	2407.15	6.55 %	2036.30	6.18 %	1862.55	5.61 %
3,500,000	2279.35	5.31 %	1940.55	4.70 %	1783.90	4.22 %
4,000,000	2175.35	4.56 %	1859.85	4.16 %	1725.70	3.26 %
4,500,000	2087.85	4.02 %	1805.40	2.93 %	1680.30	2.63 %
5,000,000	2015.75	3.45 %	1761.85	2.41 %	1645.25	2.09 %
5,500,000	1958.10	2.86 %	1727.10	1.97 %	1615.80	1.79 %
6,000,000	1905.35	2.69 %	1704.20	1.33 %	1597.70	1.12 %
6,500,000	1859.55	2.40 %	1681.05	1.36 %	1584.00	**0.86 %**
7,000,000	1814.54	2.42 %	1663.70	1.03 %	1571.40	0.80 %
7,500,000	1783.60	1.71 %	1653.00	**0.64 %**	1563.00	0.53 %
8,000,000	1752.20	1.76 %	1639.15	0.84 %	1556.35	0.43 %
8,500,000	1727.80	1.39 %	1632.35	0.41 %	1547.00	0.60 %
9,000,000	1707.80	1.16 %	1625.65	0.41 %	1541.50	0.36 %
9,500,000	1687.45	1.19 %	1619.90	0.35 %	1537.35	0.27 %
10,000,000	1668.35	1.13 %	1615.60	0.27 %	1533.75	0.23 %

En la Tabla 4.12 se muestra una vez más el porcentaje de mejora en las iteraciones con respecto a la anterior de este problema del cual podemos hacer la observación que el AG no llega al criterio de tener menos del 1 % entre sus iteraciones, por lo tanto dentro de las 10,000,000 de iteraciones no logramos identificar un punto de convergencia, mientras que en el caso del SA y SE si lo encontramos en la iteración 7,500,000 y 6,500,000 respectivamente. En cuestión de valor final ambos algoritmos de vacunación presentaron una mejor solución

que el AG, con una mejora del 3.27 % para el SA y de 8.78 % para el SE.

En la Figura 4.29 se presentan los resultados del promedio del tiempo de ejecución de los 3 algoritmos y en ella podemos observar que la tendencia a tener una mejora significativa con la vacunación sigue presente. Si comparamos el tiempo de ejecución requerido por los algoritmos de vacunación tenemos una disminución del *45.32 %* con el Selector Aleatorio y de *51.79 %* con el Selector Elitista.

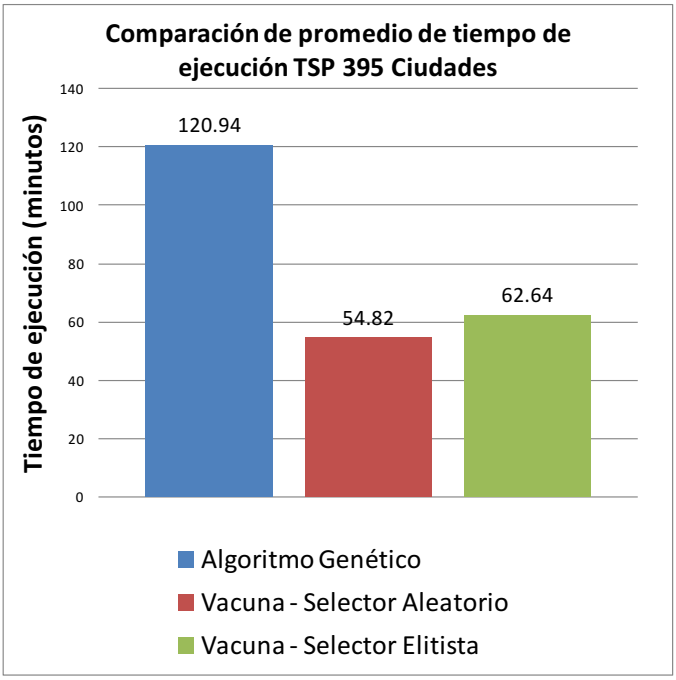
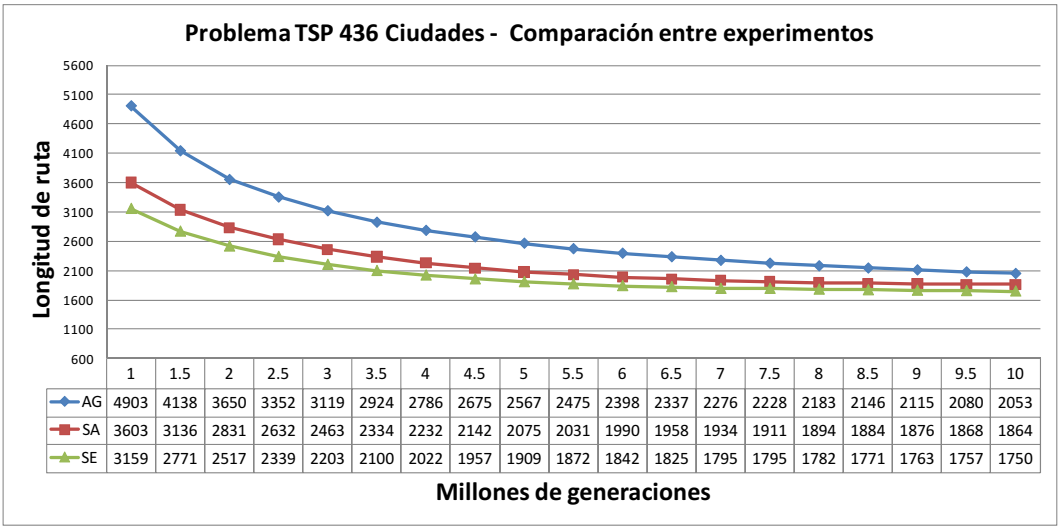


Figura 4.29: Comparación del tiempo de ejecución del TSP 395 ciudades entre AG, SA y SE.

Una vez más hacemos referencia a la Tabla 4.10 donde observamos que el algoritmo de vacunación reduce los términos de 395 ciudades a 237 elementos explicando la reducción de tiempo de ejecución significativamente.

4.3.4. Caso #4: TSP 436 ciudades



	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
AG	4903	4138	3650	3352	3119	2924	2786	2675	2567	2475	2398	2337	2276	2228	2183	2146	2115	2080	2053
SA	3603	3136	2831	2632	2463	2334	2232	2142	2075	2031	1990	1958	1934	1911	1894	1884	1876	1868	1864
SE	3159	2771	2517	2339	2203	2100	2022	1957	1909	1872	1842	1825	1795	1795	1782	1771	1763	1757	1750

Figura 4.30: Caso TSP de 436 ciudades.- Comparación de promedios de longitudes de ruta contra incremento de generaciones entre AG, SA y SE.

En este caso se presentan el TSP de 436. En la Figura 4.30 se presenta la gráfica de comparación de resultados entre el AG, el SA y el SE. En esta gráfica podemos observar la tendencia de los algoritmos de vacunación de proporcionar mejores resultados que el AG entre más grande los TSP son. En la Tabla 4.13 se muestra una vez más el porcentaje de mejora en las iteraciones con respecto a la anterior de este problema del cual podemos hacer la observación que el AG no llega al criterio de tener un avance del 1 % entre sus iteraciones, y una vez más los algoritmos de vacunación presentan esta característica pero ahora en la iteración 8,000,000 para el SA y en 6,500,000 para el SE. Con lo que respecta al valor final ambos algoritmos de vacunación presentaron una mejor solución que el AG, con una mejora del 10.18 % para el SA y de 17.33 % para el SE, ambos valores son significativamente mejores que el AG.

Tabla 4.13: Cuadro comparativo de los porcentajes de mejoramiento en la solución con respecto a la generación pasada para el TSP de 436 ciudades. Se indica el porcentaje del valor de convergencia.

	AG		SA		SE	
1	4902.53	0 %	3602.65	0 %	3158.55	0 %
1.5	4138.16	15.59 %	3136.15	12.95 %	2771.35	12.26 %
2	3650.47	11.79 %	2830.80	9.74 %	2516.65	9.19 %
2.5	3351.74	8.18 %	2632.05	7.02 %	2338.95	7.06 %
3	3119.16	6.94 %	2463.05	6.42 %	2203.40	5.80 %
3.5	2924.21	6.25 %	2333.85	5.25 %	2099.55	4.71 %
4	2786.11	4.72 %	2231.50	4.39 %	2022.35	3.68 %
4.5	2675.32	3.98 %	2141.55	4.03 %	1956.50	3.26 %
5	2567.37	4.03 %	2074.70	3.12 %	1908.80	2.44 %
5.5	2474.58	3.61 %	2030.95	2.11 %	1872.45	1.90 %
6	2398.11	3.09 %	1990.20	2.01 %	1842.05	1.62 %
6.5	2337.16	2.54 %	1958.40	1.60 %	1824.85	**0.93 %**
7	2276.43	2.60 %	1934.40	1.23 %	1794.70	1.65 %
7.5	2228.21	2.12 %	1910.95	1.21 %	1794.50	0.01 %
8	2183.39	2.01 %	1893.90	**0.89 %**	1781.75	0.71 %
8.5	2146.00	1.71 %	1884.05	0.52 %	1770.80	0.61 %
9	2114.63	1.46 %	1876.30	0.41 %	1763.15	0.43 %
9.5	2079.89	1.64 %	1868.40	0.42 %	1756.80	0.36 %
10	2053.47	1.27 %	1863.80	0.25 %	1750.15	0.38 %

En la Figura 4.31 se presentan los resultados del promedio del tiempo de ejecución de los tres algoritmos y en ella podemos observar que la tendencia a proporcionar tiempos de ejecución mucho más cortos con los algoritmos de vacunación sigue presente. Si comparamos el tiempo de ejecución de los algoritmos de vacunación tenemos una reducción del *47.03 %* con el Selector Aleatorio y de *52.11 %* con el Selector Elitista.

Retomando la Tabla 4.10 una vez más observamos que el algoritmo de vacunación reduce los términos de 436 ciudades a 256 elementos, explicando la reducción de tiempo de ejecución significativamente.

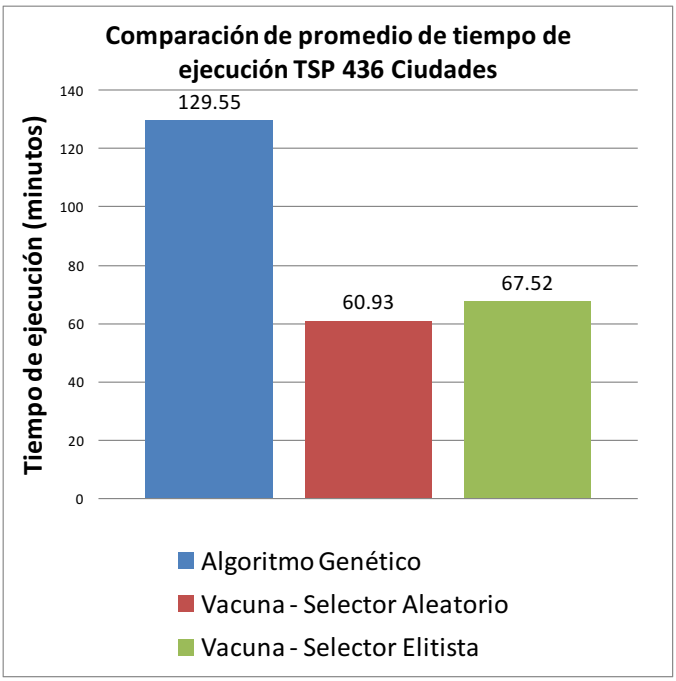
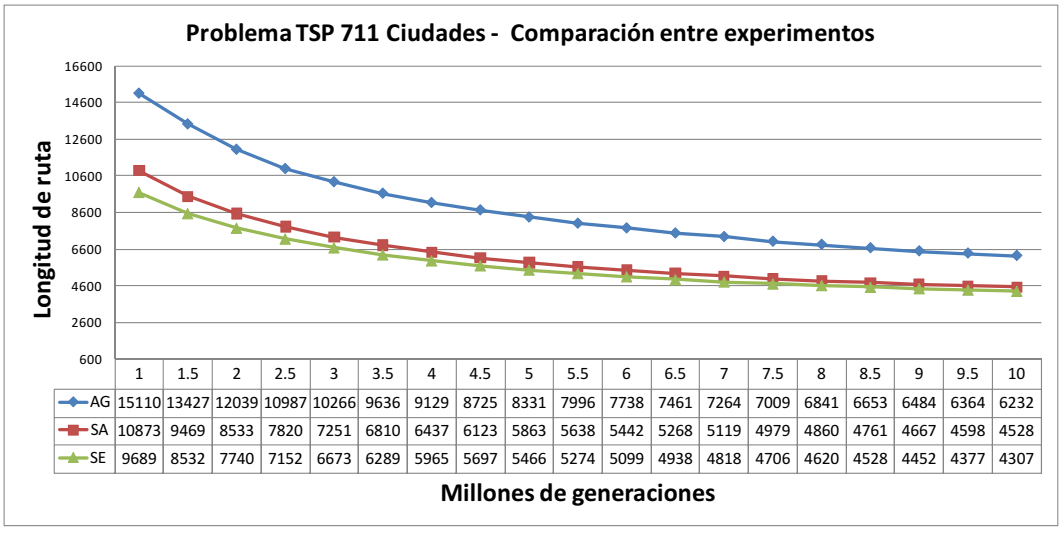


Figura 4.31: Comparación del tiempo de ejecución del TSP 436 ciudades entre AG, SA y SE.

4.3.5. Caso #5: TSP 711 ciudades



	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
AG	15110	13427	12039	10987	10266	9636	9129	8725	8331	7996	7738	7461	7264	7009	6841	6653	6484	6364	6232
SA	10873	9469	8533	7820	7251	6810	6437	6123	5863	5638	5442	5268	5119	4979	4860	4761	4667	4598	4528
SE	9689	8532	7740	7152	6673	6289	5965	5697	5466	5274	5099	4938	4818	4706	4620	4528	4452	4377	4307

Figura 4.32: Caso TSP de 711 ciudades.- Comparación de promedios de longitudes de ruta contra incremento de generaciones entre AG, SA y SE.

Por último tenemos el caso del TSP con 711. En la Figura 4.32 se presenta la gráfica comparativa entre los tres algoritmos AG, SA y SE. En esta gráfica resulta evidente que los valores proporcionados por los algoritmos de vacunación son mucho mejores a lo largo de la ejecución del AG, y a su vez observamos que los tres algoritmos se comportan de una manera similar.

Tabla 4.14: Cuadro comparativo de los porcentajes de mejoramiento en la solución con respecto a la generación pasada para el TSP de 711 ciudades. Se indica el porcentaje del valor de convergencia.

	AG		SA		SE	
1	15109.80	0 %	10872.75	0 %	9688.90	0 %
1.5	13427.20	11.14 %	9469.20	12.91 %	8532.40	11.94 %
2	12039.27	10.34 %	8532.70	9.89 %	7740.30	9.28 %
2.5	10986.80	8.74 %	7820.30	8.35 %	7151.95	7.60 %
3	10265.80	6.56 %	7251.05	7.28 %	6672.70	6.70 %
3.5	9635.80	6.14 %	6810.40	6.08 %	6288.85	5.75 %
4	9128.73	5.26 %	6437.10	5.48 %	5965.15	5.15 %
4.5	8725.33	4.42 %	6122.70	4.88 %	5696.90	4.50 %
5	8331.27	4.52 %	5862.60	4.25 %	5466.05	4.05 %
5.5	7996.00	4.02 %	5638.35	3.83 %	5273.70	3.52 %
6	7737.67	3.23 %	5441.85	3.49 %	5099.20	3.31 %
6.5	7460.60	3.58 %	5267.80	3.20 %	4938.15	3.16 %
7	7264.45	2.63 %	5118.60	2.83 %	4818.20	2.43 %
7.5	7009.40	3.51 %	4979.10	2.73 %	4705.75	2.33 %
8	6841.00	2.40 %	4860.30	2.39 %	4619.65	1.83 %
8.5	6653.47	2.74 %	4761.30	2.04 %	4527.50	1.99 %
9	6484.13	2.55 %	4667.15	1.98 %	4451.75	1.67 %
9.5	6364.29	1.85 %	4597.80	1.49 %	4377.35	1.67 %
10	6232.00	2.08 %	4527.55	1.53 %	4306.65	1.62 %

En la Tabla 4.14 se muestra una vez más el porcentaje de mejora en las iteraciones con respecto a la anterior de la cual podemos hacer la observación que ninguno de los tres algoritmos logró cumplir con el parámetro menor al 1 % de avance entre las iteraciones, lo cual significa que mejores soluciones eran posibles con mayores iteraciones. Sin embargo los algoritmos de vacunación muestran una tendencia a llegar a este punto antes que el AG, como puede ser observado en los demás problemas. Con lo que respecta al valor final ambos algoritmos de vacunación presentaron una mejor solución que el AG, con una mejora del

37.65 % para el SA y de 44.71 % para el SE, ambos valores son significativamente mejores que el AG. Esto sin duda es una de las mayores aportaciones del trabajo, ya que nos indica que para repertorio de ciudades muy grandes el algoritmo de vacunación proporciona mejoras significativas en los resultados finales.

En la Figura 4.33 se presentan los resultados del promedio del tiempo de ejecución de los tres algoritmos y en ella podemos observar que la tendencia observado a lo largo de todos los problemas. Si comparamos el tiempo de ejecución de los algoritmos de vacunación tenemos una reducción del *52.18* % con el Selector Aleatorio y de *46.94* % con el Selector Elitista.

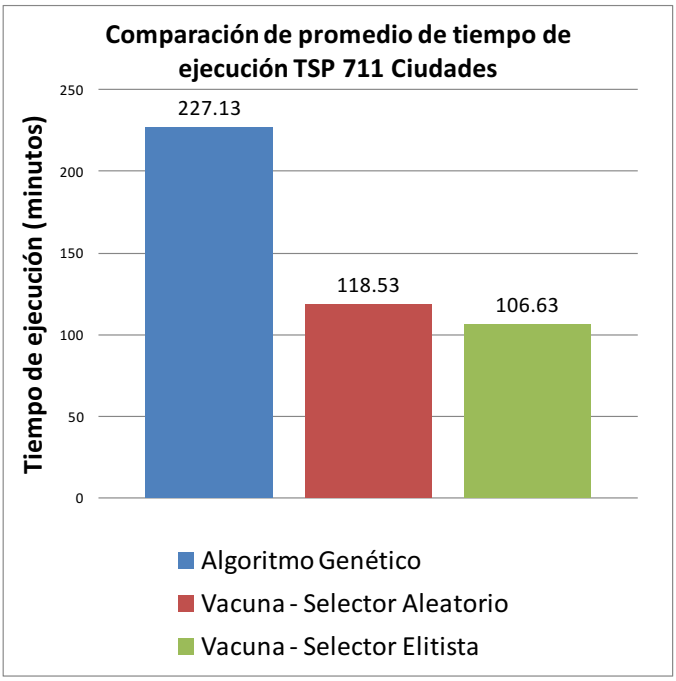


Figura 4.33: Comparación del tiempo de ejecución del TSP 711 ciudades entre AG, SA y SE.

Por último retomando la Tabla 4.10 se observa una vez más que el algoritmo de vacunación reduce los términos de 711 ciudades a 427 elementos, lo cual explica la reducción de tiempo de ejecución.

5 Conclusiones

En este trabajo se ha presentado un nuevo algoritmo en el área de los Algoritmos Inmunológicos Artificiales inspirado en el concepto de inmunización por vacunación. El algoritmo propuesto permite mejorar el desempeño de prácticamente cualquier Algoritmo Evolutivo para la solución de problemas de matemática combinatoria (caso de estudio el problema del agente viajero o TSP) basados en población y adicionalmente no los modifica ni conceptual (retienen sus terminologías originales) ni estructuralmente (no es necesario modificar el algoritmo).

Las metodologías propuestas: la vacunación con Selector Aleatorio y Selector Elitista, han demostrado mejoras significativas con respecto al Algoritmo Genético estudiado. En el caso con el mayor número de ciudades se demostró una mejora de la solución en un *37.65 %* con el Selector Aleatorio y de *44.71 %* con el Selector Elitista. Adicionalmente los tiempos de ejecución de todos los casos estudiados recibieron una disminución en su tiempo de ejecución del 45 % al 52 %.

Dado los resultados se puede concluir que los algoritmos de vacunación son mucho más efectivos cuando se trabajan con grandes cantidades de ciudades (150 o más), sin embargo aún cuando esto fue lo observado con los problemas de prueba no se debe descartar otras configuraciones como efectivas (por ejemplo, el reducir el número de vacunas o incrementar el número de ciudades por vacuna). A su vez, el concepto de vacunación puede ser expandido aún más al idear nuevas maneras de crear las vacunas adicionales a las propuestas en este trabajo de tesis.

A su vez, en este trabajo de tesis se desarrollo una aplicación que permite hacer pruebas de los algoritmos desarrollados y de explorarlos con un sinnúmero de diferentes mapas de TSP

y de configuraciones de los mismos algoritmos.

Otro punto importante es que debido a que el Algoritmo Genético requiere del uso de números aleatorios, cada ejecución se proporciona una semilla distinta para ser utilizada por el generador de números pseudoaleatorio.

En aquellos casos en los cuales se utilizaba la misma semilla el Algoritmo Genético presentaba comportamiento muy similar entre sí, lo cual daba como resultado el mismo comportamiento por parte del algoritmo referente a la calidad de las soluciones que podía proporcionar.

5.1. Discusión y trabajo a futuro

Dado a que este presenta una nueva metodología existen numerosas puntos de interés de discusión y de posible trabajo a futuro. A continuación se enlistan algunas de estas ideas.

- La generación de vacunas demostró ser un punto de gran interés, ya que existen diversos parámetros y metodologías que no han sido exploradas en este trabajo de tesis, como lo puede ser el emplear funciones de selección basadas en geometría más complejas, siendo un ejemplo de esta idea el utilizar los ángulos generados entre los elementos.
- Sería conveniente explorar la modificación del parámetro de número de ciudades por vacuna y analizar estos resultados.
- La implementación del algoritmo de vacunación con otros métodos ya estudiados para solucionar el TSP, como puede ser de búsqueda exhaustiva clásica u otros métodos matemáticos como el de ramificar y acotar (*Branch-and-Bound*).
- Existe la posibilidad de utilizar las vacunas de otra manera, que es afectando directamente las rutas iniciales que forman parte de la población a ser tratada por los algoritmos. De esta manera es posible implementar las vacunas de restricción, las cuales evitan que los elementos de la población inicial presenten ciertas subrutas que demuestran ser malas soluciones.

Bibliografía

[1] T. Andrews. A computational model of degeneracy in a lymph node. *Lecture Notes in Computer Science 4163*, 2006.

[2] M. Budinich. A self-organising neural network for the travelling salesman problem that is competitive with simulated annealing. *Neural Computation Volume 8*, 1996.

[3] A. R. C. Blum. *Metaheuristics in combinatorial optimization: Overview and conceptual comparison*. ACM Computing Surveys, 2003.

[4] K. S. C.H. Papadimitriou. *Combinatorial Optimization*. Dover Publications, INC, 1998.

[5] L. F. N. Dipankar Dasgupta. *Immunological Computation Theory and Applications*. CRC Press, 2009.

[6] F. G. Dorigo M. Corne, D. *New Ideas in Optimization*. McGraw Hill, 1999.

[7] G. Edelman. Degeneracy and complexity in biological systems. *Proceedings of the National Academy of Sciences*, pages 13763–13768, 2001.

[8] J. T. Emma Hart. Application areas of ais: The past, the present and the future. *Applied Soft Computing 8*, pages 191–201, 2008.

[9] A. P. G. Gutin. *The traveling salesman problem and its variations*. Kluwer Academic Publishers, 2004.

[10] T. Gatech. Solution of a 15,112-city traveling salesman problem, Mayo 2005 disponible: http://www.tsp.gatech.edu/d15sol/index.html.

[11] A. P. P. Gregory Gutin. *Travelling Salesman Problem and Its Variations*. Kluwer Academic Publishers, 2002.

[12] S. R. A. Z. Gregory Gutin, Helmut Jakubowicz. Seismic vessel problem. In *Communic. in DQM 8*, pages 13–20, 2005.

[13] K. Herland. Have you heard the one about the travelling salesman? March 2007.

[14] M. X. Jingui Lu. *Immune-Genetic Algorithm for Traveling Salesman Problem.*

[15] I. O. J.P. Kelly. *Meta-Heuristic: Theory and Applications.* Kluwer Academic Publisher, 1996.

[16] R. Karp. Reducibility among combinatorial problems. *R. Miller and J. Thatcher (eds.): Complexity of Computer Computations*, pages 85–103, 1972.

[17] J. O. Kephart. A biologically inspired immune system for computers. *Proceedings of Artificial Life IV: The Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 130–139, 1994.

[18] J. T. Leonardo N. de Castro. *Artificial Immune Systems: A New Computational Intelligence Approach.* Springer, 2002.

[19] e. a. M. Mendao, J. Timmis. The immune system in pieces: Computational lessons from degeneracy in the immune system. *Foundations of Computational Intelligence*, 2007.

[20] L. M. G. Marco Dorigo. Ant colonies for the traveling salesman problem. *IRIDIA*, 1997.

[21] e. a. Ming-Yuan Zhao, Ke Tang. A novel clonal selection algorithm and its application. *Apperceiving Computing and Intelligence Analysis*, pages 385–388, 2008.

[22] W. D. C. Mulder, Samuel A. Million city traveling salesman problem solution by divide and conquer clustering with adaptive resonance neural networks. *Neural Networks 16*, pages 827–832, 2003.

[23] A. D. Munoz. *Metaheuristicas.* Dykinson, 2008.

[24] G. Orosz. An introduction to immuno-ecology and immuno-informatics. *Design Principles for the Immune Systems and Other Distributed Systems*, pages 125–150, 2001.

[25] R. M. I. I. S. D. P. Larranaga, C. Kuijpers. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, 13:129–170, 1999.

[26] R. Raven. 10th international conference on artificial immune systems, 2011 disponible: http://www.researchraven.com/conference/2011/7/18/10th-international-conference-on-artificial-immune-systems.aspx.

[27] I. O. C. R. S. Voss, S. Martello. *Metaheuristic: Advances and Trends in Local Search Paradigms for Optimization*. Kluwer Academic Publishers, 1999.

[28] L. A. R. C. Stephanie Forrest, Alan S. Perelson. *Self-Nonself Discrimination in a Computer*. IEEE Computer Society Press, 1994.

[29] G. N. M. P. V. Cutello, G. Narzisi. Clonal selection algorithms: A comparative case study using effective mutation potentials, optia versus clonalg. *4th Int. Conference on Artificial Immune Systems, ICARIS 2005*, pages 13–28, 2005.

[30] R. J. Vera. Matematica discreta: Teoria de grafos. 2002.

[31] C. A. C. C. Victoria S. Aragon, Susana C. Esquivel. Artificial immune system for solving constrained optimization problems, 2007.

[32] T. Weise. *Global Optimization Algorithms Theory and Application*. 2009.

[33] J. M. Whitacre. Degeneracy: a link between evolvability, robustness and complexity in biological systems. *Theoretical Biology and Medical Modelling 7*, 2010.

[34] Xin-SheYang. *Engineering Optimization An Introduction with Metaheuristic Applications*. WILEY, 2010.

[35] H. D. S. G. Yunyi Zhu, Zheng Tang. Cooperation artificial immune system with application to traveling salesman problem. *ICIC Express Letters*, pages 143–148, 2008.

[36] D. F. Z. Michalewicz. *How to Solve it: Modern Heuristic*. Springer, 2000.