Estudio Comparativo de Algoritmos Evolutivos Bioinspirados y Trayectoriales para la solución de los problemas agente viajero y programación de horarios

Marlene Lopez[1] and Gustavo López[1]

[1]Instituto Tecnológico de León, Maestría en Ciencias de la Computación

May 31, 2018

Abstract

En el presente trabajo se muestran los resultados obtenidos tras la comparación de los Algoritmos Metahurísticos Bioinspirados; Algoritmo Genético (GA) , Memético (MA) y Sistema Inmune (ISA). La instancia tomada como prueba fue el Problema OneMax o Conteo de Bits, para poblaciones de 500, 1,000, 2,000 y 5,000. El objetivo establecido fue observar el comportamiento de los mismos; los parámetros que influyen en sus resultados, así como determinar el de mejor desempeño. Bajo pruebas estadísticas empleadas, el algoritmo MA obtuvo mejores resultados déspues el algoritmo ISA y GA respectivamente, siendo evaluados con distintas configuraciones cada uno.

1 Introducción

La optimiz ación es un problema matemático comunmente encontrado en todas las disciplinas de ingeniería. Esta, literalmente significa encontrar la mejor solución posible / deseable. Los algoritmos de optimización pueden ser de naturaleza determinista o estocástica. Los métodos anteriores para resolver problemas de optimización requieren enormes esfuerzos computacionales, que tienden a fallar a medida que aumenta el tamaño del problema. Esta es la motivación para emplear algoritmos de optimización estocástica bio-inspirados como alternativas computacionalmente eficientes al enfoque determinista. Las metaheurísticas se basan en la mejora de una población de soluciones o una única solución y emplean mayormente aleatorización (Algoritmo Genético,GA) y búsqueda local (Algoritmo Memético, MA) para resolver el problema dado. Los algoritmos metahurísticos gozan de una buena reputación para resolver problemas de optimización como Traveling Salesman Problem y Timetabling. Algunos de estos algoritmos se consideran como trayectoriales debido a la forma en que buscan dirigir un solo individuo hacia el óptimo global.

En el presente trabajo se muestra la implementación de algoritmos metaheurísticos poblacionales: Algoritmo Genético, Memético y Sistema Inmune así como, algoritmos trayectoriales: Recocido Simulado y Búsqueda Local Iterada, para la resolución de los problemas OneMax, Traveling Salesman Problem y Timetabling.

2 Antecedentes

Los Algoritmos Evolutivos son sistemas computacionales diseñados para solucionar problemas altamente no-lineales, tomando ideas del proceso avolutivo y de adaptación de la naturaleza, las cuales se grupan bajo la denominada teoría Neo-Darwiniana de la evolución [1, 2].

Gran parte de la terminolóigia utilizada en los algoritmos evolutivos se ha tomado prestada de los conceptos biológicos en los que se inspira (por ejemplo, cromosoma, alelos, etc.).

Una característica notable de los algoritmos evolutivos es que son poblacionales, lo que significa que manipulan simultáneamente un conjunto de soluciones potenciales al problema y no una solución única, como suelen hacer las técnicas clásicas de optimización. El uso de una población de soluciones potenciales a un problema hace a los algoritmos evolutivos menos suceptibles de quedar atrapados en óptimos locales [3].

Los operadores genéticos básicos son los siguientes:

- **Selección:** Consiste de un mecanismo (probabilístico o determinista) que permite elegir a los individuos que fungirán como padres de la siguiente generación.
- **Cruza o Recombinación:** Se refiere al intercambio de información (material genético) entre dos padres que han sido seleccionados con base en su aptitud.
- **Mutación:** Consiste en hacer pequeñas perturbaciones a los individuos recién creados para la nueva población con la finalidad de explorar zonas del espacio de búsqueda que la cruza no pueda alcanzar.

Sin embargo, también existen algoritmos evolutivos que son considerados como trayectoriales dado que para encontrar las soluciones óptimas parten de un punto específico y dirigen la exploración a través de la inspección de soluciones vecinas siguiendo aquellas que resultan ser más aptas. La necesidad de proveer a estos algoritmos de los aspectos tanto exploratorio como explotatorio, i.e. la capacidad de realizar una búsqueda suficientemente amplia y específica del espacio de búsqueda, implica dotarlos de algún mecanismo que controle la probabilidad de aceptar soluciones peores conforme el espacio de solución es explorado.

3 Métodos

3.1 Algoritmos Poblacionales

3.1.1 Algoritmo Genético (GA)

El algoritmo genético es un algoritmo inspirado en la evolución biológica y su base genético-molecular, que simula la progresión temporal de una población de individuos sometiéndola a una serie de transformaciones aleatorias y selecciones semejantes a las que ocurren en la naturaleza evolutiva biológica[4]. Los GAs buscan dentro de un conjunto los individuos más aptos, en cuyos cromosomas codifican la información de una posible solución al problema usualmente representada como una cadena de números; en cada paso del ciclo iterativo se evolucionan los individuos y se evalúan con alguna medida de aptitud hasta que alguna condición de paro se cumple. La evolución se logra aplicando repetidamente sobre los individuos los operadores généticos de selección, cruza, mutación y reemplazo.

La selección consiste en determinar los individuos de la población que serán propicios para cruzar, a través de algún criterio específico. Entre estos figuran la selección por k-torneo[5] y la selección Vasconcelos[6].

Table 1: Configuraciones para el Algoritmo Genético

Algoritmo	Máximo Generaciones	Máximo de llamadas a función	Probabilidad de muta	Porcentaje de elitismo	Selección	Cruza
GA1	3,000	1,000,000	0.05	0.3	k-tournament	Single point

La cruza es la acción de combinar los individuos seleccionados para obtener nuevos individuos; existen criterios de cruza como el *single-point crossover* que divide los padres en un cromosoma específico y los recombina a partir de ello. La mutación consiste en realizar sutiles modificaciones aleatorias sobre el código genético de un individuo, con baja probabilidad de ocurrencia, para introducir variaciones estocásticas sobre la población. El reemplazo ocurre al obtener una nueva generación de individuos generados a partir de la generación anterior. En la figura 1 se denota el algoritmo genético como un diagrama de flujo.

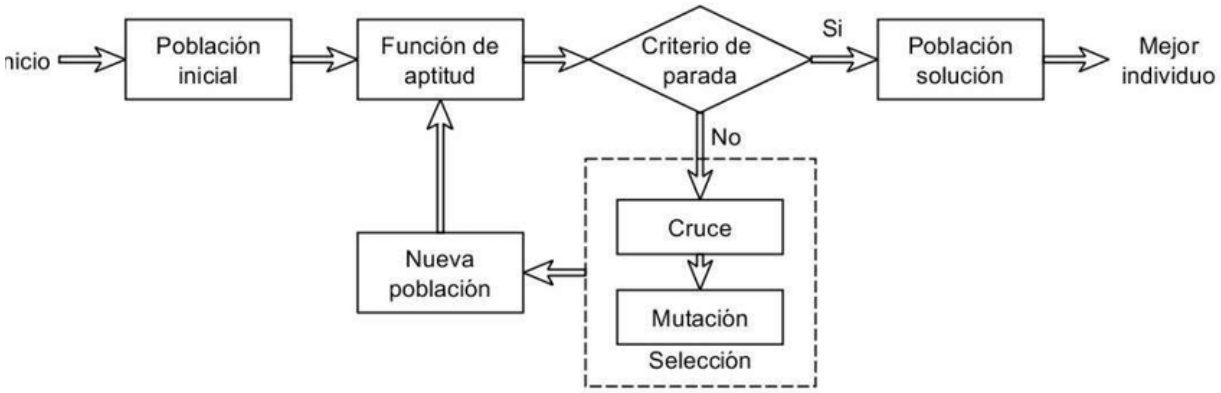

Figure 1: Diagrama de flujo del Algoritmo Genético

Con estos mecanismos se busca que la población de individuos presente iterativamente una mejora en la evaluación de sus aptitudes, para determinar eventualmente aquellos que representen soluciones lo más cercano posible a la solución óptima.

En este trabajo se determinaron dos variaciones del algoritmo genético para dar solución al problema de Onemax. Dichas configuraciones incluyen la selección de heurísticas como se indica en la tabla 1.

3.1.2 Algoritmo Memético (MA)

El Algoritmo Memético representa una sinergización con el Algoritmo Genético o con cualquier enfoque de algoritmos poblacionales en el que se utilizan aprendizaje individual o procedimientos para mejora local para el problema de búsqueda. Está inspirado tanto en los principios de la evolución natural de Charles Darwin, como en la noción de Clinton Richard Dawkins del *meme* como una idea, comportamiento o estilo que se esparce de persona a persona en una cultura. El término fue introducido por Moscato [7].

Figure 2: Seudocódigo del Algoritmo Memético

Table 2: Configuraciones para el Algoritmo Memético

Algoritmo	No. Max.Gen	Calls Functi	Prob. muta	% Elitismo	Selección	Búsquedas locales	Perturbación
MA1	3,000	1,000,000	0.05	0.3	k-tournament	7	kflip 5

Table 3: Configuraciones para el algoritmo Sistema Inmune.

Algoritmo	Máximo de generaciones	Maximo de llamadas a función	Elitismo	No. de Clones	Perturbación
ISA1	3,000	1,000,000	0.3	3	K-flip 8

3.1.3 Algoritmo Sistema Inmune (ISA)

El AIS se basa en el funcionamiento del sistema inmune humano, que es capaz de reconocer en una forma muy eficiente cualquier agente patógeno. Esta es una teoría inmune clásica para entender el sistema inmunológico como un sistema que identifica a self (agente propio) o a non self(agente patógeno) [8].

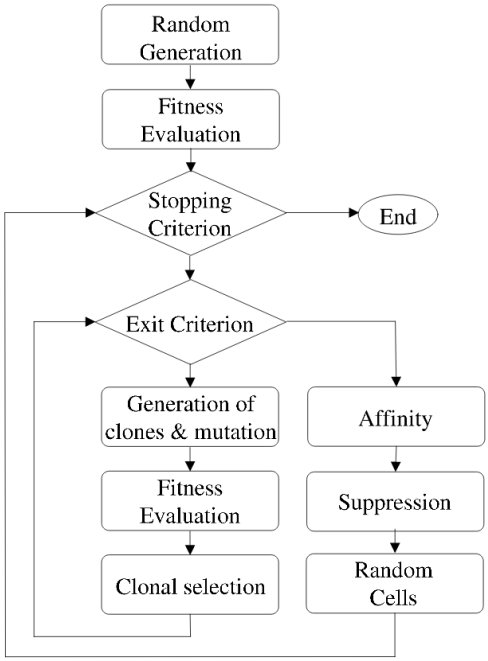

Figure 3: [8]Diagrama de flujo del algoritmo de optimización ISA

Table 4: Configuraciones para el algoritmo Búsqueda Local Iterada (ILS).

Algoritmo	Máximo de generaciones	Maximo de llamadas a función	No. LS	Perturbación
ILS1	3,000	1,000,000	5	K-flip 1
ILS2	3,000	1,000,000	9	Inversing

3.2 Algoritmos Trayectoriales

3.2.1 Búsqueda Local Iterada

La idea principal de búsqueda local iterativa es tratar de generar vecinos que nos generen pozos de atracción vecinos (o al menos diferentes)[9]. El paso clave es determinar el primer vecino que pertenece a otro pozo de atracción. Con esto, podríamos ir generando pozos de atracción vecinos.

Sin embargo, es un proceso computacionalmente muy costoso, por la cantidad de llamadas a búsqueda local que se requieren [9].

En la imagen 4 se puede observar la serie de pasos para desarrollar el algoritmo de Búsqueda Local Iterada.

En la tabla 4 se pueden observar los parámetros establecidos para dos distintas configuraciones del algoritmo.

procedimiento *Búsqueda Local Iterativa*
s_0 = genera solución inicial
$s*$ = búsqueda local (s_0)
repeat
  s' = perturbación(s^*,historia)
  $s*'$ = búsqueda local (s')
  $s*$ = criterio de aceptación($s*, s*'$,historia)
until criterio de terminación

Figure 4: Algoritmo Busqueda Local Iterada

3.2.2 Recocido Simulado

El recocido simulado está inspirado en el proceso de recocido en la metalurgia, el cual implica una técnica que incluye el calentado y enfríado controlado de un material para aumentar el tamaño de sus cristales (arquitectura molecular interna) y así reducir sus defectos.

Esta noción del cambio controlado de temperatura es interpretada como un descenso pequeño en la probabilidad de aceptar soluciones menos aptas a medida que el espacio de búsqueda es explorado, lo que implica una búsqueda extensiva.

A cada paso, el algoritmo seleccion aleatoriamente una solución cercana a la actual, mide su calidad, y después decide si moverse hacia ella o quedarse en la actual basado en alguna de las dos probabilidades

Table 5: Configuraciones para el algoritmo Recocido Simulado (SA).

Algoritmo	Generaciones Max.	Máximo Iter Temp	Maximo Fun-Calls	Prob. Acept	Esquema Temp	Perturbación
SA1	3,000	100	1,000,000	1	2	K-flip 2
SA2	3,000	100	1,000,000	1	2	K-flip 1

entre las que escoje basado en el hecho de si la nueva solución es mejor o peor que la actual. Durante la búsqueda, el descenso en la temperatura se refleja en una mayor probabilidad de dirigirse siempre a la mejor solución, y en una menor probabilidad de dirijirse a soluciones menos aptas [10].

En la tabla 5 se pueden observar los parámetros establecidos para dos distintas configuraciones del algoritmo.

Entrada: Solución inicial (x_0).
Salida: Solución final (x^*).

$x = x^* = x_0$ $T = T_0$ Iter = 0
Mientras iter < MaxIter
    Generar solución x' en el vecindario $N(x)$ de forma aleatoria
    Si $f(x') < f(x)$ $x = x'$ Sino
        r = Aleatorio (0,1)
        Si $r < e^{-\frac{f(x')-f(x)}{T}}$
            $x = x'$ Si $f(x') < f(x^*)$
        $x^* = x'$
    $T = \propto * T$
    iter = iter+1
Fin

Figure 5: Algoritmo Recocido Simulado

4 Operadores Genéticos Básicos

4.1 Métodos de selección

4.1.1 Selección mediante Ruleta

Es una técnica proporcional, lo que significa que la probabilidad de un individuo de ser seleccionado está en proporción a su aptitud. En esta selección se simula que a cada individuo se le asigna una porción de una ruleta, de forma que al girarla, los individuos con una porción mayor tienen mayor probabilidad de ser seleccionados. Sin embargo, debido a su naturaleza probabilística, este esquema permite que aún los individuos menos aptos tengan alguna probabilidad de ser seleccionados para reproducirse[3]. En la Fig. 6 se puede observar el comportamiento de está selección.

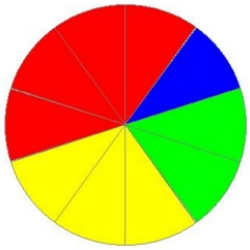

Figure 6: La selección por el método de ruleta es proporcional. En este caso, los individuos que pertenecen a la zona roja y amarilla tienen mayor probabilidad de ser elegidos respectivamente. Sin embargo el individuo de la zona azul tambien puede ser seleccionado, aunque con una probabilidad muy baja.

4.2 Cruza

4.2.1 Cruza a un punto

En la cruza a un punto, dos padres se cruzan para generar dos hijos, y cada uno de ellos aporta un segmento de su cromosoma. El punto a partir del cual se realiza la cruza se define aleatoriamente. Cada hijo recibe un segmento de cada uno de los padres. La figura 7 muestra este mecanismo.

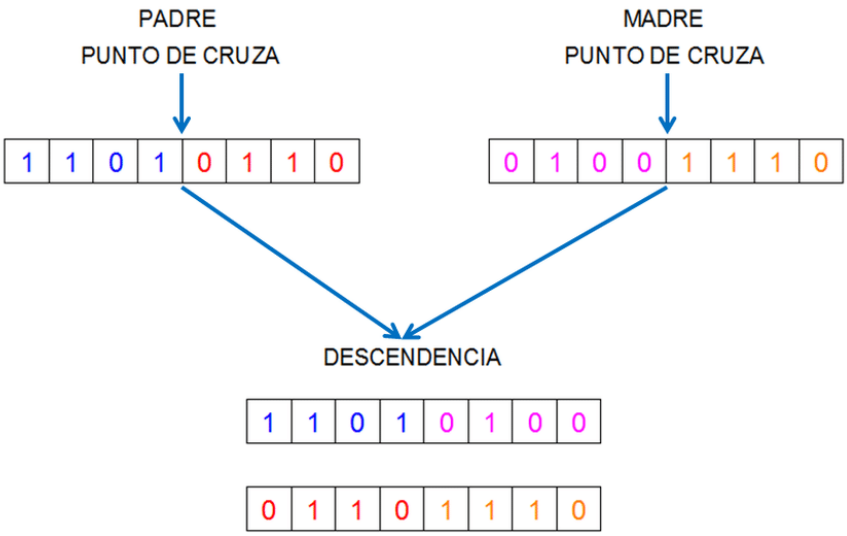

Figure 7: Representación de la cruza en un punto. Dos padres generan dos hijos.

4.3 Muta

4.3.1 Muta Uniforme

iforme consiste en cambiar el valor de cada bit de la cadena cromosómica con una probabilidad determinada, (se recomienda que sea pequeña, entre 0.01 y 0.001). La finalidad de la mutación es explorar regiones del espacio de búsqueda que el operador de cruza no pudiera alcanzar[3].

4.3.2 Mutación Flip

Basado en una mutación generada cromosoma, voltear un poco implica cambiar 0 a 1 y 1 a 0. Se considera un padre y una mutación el cromosoma se genera al azar. Por un 1 en cromosoma de mutación, el bit correspondiente en la matriz el cromosoma se voltea (0 a 1 y 1 a 0) y el hijo del cromosoma es producido. Se usa comúnmente en codificación binaria [11].

4.3.3 Mutación Swap

La mutación de intercambio funciona seleccionando al azar dos genes en el genotipo y cambiándolos entre si[12]. Un ejemplo de mutación de intercambio se da en la Figura. 8.

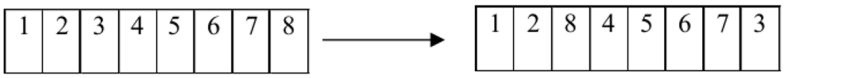

Figure 8: Mutación Swap, los genes 3 y 8 fueron intercambiados.

4.3.4 Mutación Inversa

La mutación inversa trabaja seleccionando un subconjunto de los genotipos e invirtiendo el orden de los genes. En la Figura puede verse un ejemplo de está muta[12].

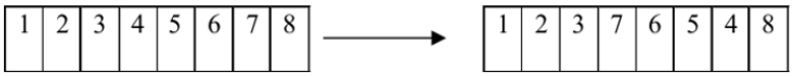

Figure 9: Mutación Inversa, el subconjunto que va de 4 a 7 fueron seleccionados y su orden invertido.

5 Resultados

5.1 El problema TimeTabling

- un conjunjo de eventos $E = \{e_1, e_2, ..., e_n\}$
- un conjunto de intervalos de tiempo $T = \{t_1, t_2, ..., t_n\}$
- un grafo $G(E, L)$ con nodos E y bordes $L \subseteq \{\{u, v\}|u, v \in E\}$.

El grafo G se conoce como grafo de conflicto. Dos eventos se llaman conflictivos si son adyacentes en G. Una tarea es definida por un par ordenado (e,t) tal que $e \in E$ y $t \in T$ con la interpretación “. Dos eventos e_1, e_2 se traslapan si $\{e_1, e_2\} \in h_1$. Un horario es llamado “libre de traslape”

Las instancias de prueba utilizadas para el desarrollo del experimento se observan en el siguiente listado. Estas instancias relacionan solo horarios con eventos, siendo estos los conjuntos que conforman el grafo de conflictos.

- 1

- 2
- 16
- 17
- 24
- 25
- 34
- 35

5.2 El problema TSP

Las instancias de prueba utilizadas para el desarrollo del experimento se encuentran en [13] y se observan en el siguiente listado.

- KroA100
- KroA150
- KroA200
- KroB100
- KroB150
- KroB200
- KroC100
- KroD100

5.3 Comparativa de los algoritmos y su desempeño

Las configuraciones establecidas para los algoritmos estan definidas en las tablas 1 para el algoritmo Genético con las etiquetas GA1 y GA2. Para el algoritmo Memético 2. Finalmente para el algoritmo Sistema Inmune en la Tabla. 3 con las etiquetas ISA1 e ISA2 para cada configuración.

5.4 Solución al problema TSP

En la fig 10 se observan los resultados obtenidos en 33 distintos experimentos realizados de cada algoritmo para la solución al óptimo encontrado en la instancia "KroA100". Las distintas instancias tuvieron un comportamiento muy parecido, las líneas de color rojo corresponden a los minimos encontrados dentro de cada generación mientras que la línea azul representa el promedio de la población, la esperanza dentro de ella. Podemos observar en la fig 10 a) al algoritmo genético en el cual el promedio y el minimo encontrado se encuentran bastante cercanos, debido a que la población se ve afectada en un porcentaje muy bajo, que a su vez puede o no presentarse entre cada generación, los cambios en el mínimo encontrado son pequeños y se mantienen por valgunas generaciones. En la fig 10 b) encontramos al algorimo memético, en este caso los saltos hacia un mínimo cada vez más cerca al óptimo son más grandes y se mantienen por más generaciones que el genético, en este caso el promedio de la población se ve directamente afectado ya que, los individuos

mutan en cada generación. En el caso del sistema inmune el promedio y el mínimo encontrado tienen una separación menos pronunciada que en el genético, el sistema inmune tiene una velocidad de convergencia mayor debido a que la exploración del mejor uindividuo entre cada generación recurre a una explotación determinada por la clonación del minimo entontrado.

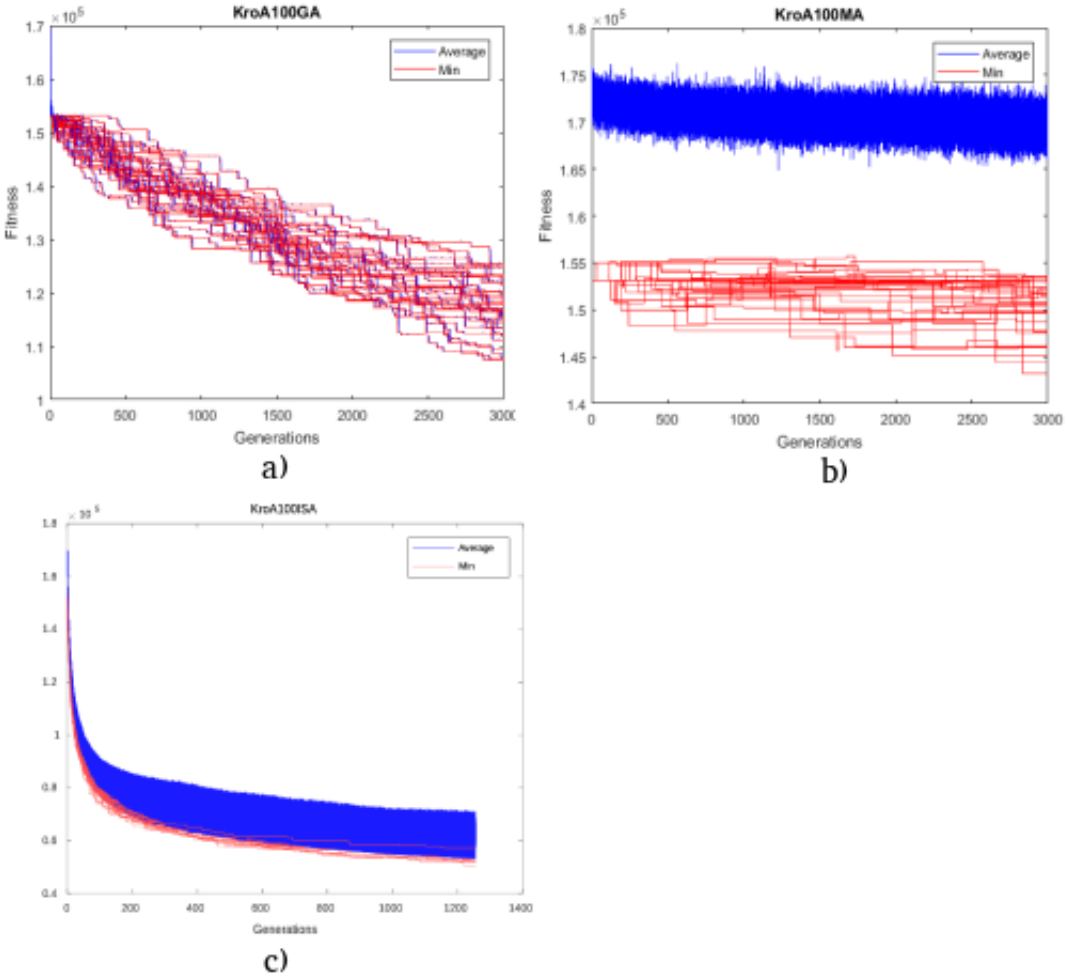

Figure 10: Gráficas del comportamiento de los algoritmos poblacionales en la solución a una instancia del problema TSP.

En la fig 11 se observa la tendencia de los algoritmos en cada una de las soluciones experimentales obtenidas, puediendose observar que, el ISA llego a encontrar minimos más pequeños que GA y MA respectivamente. En este caso se presentan 3 de las 10 instancias probpuestas de prueba, dichas graficas mantienen su comportamiento para cada una de las instancias.

Trás el desempeño de los algoritmos pobblacionales se puede observar el ISA como el algoritmo más fuerte de este experimento.

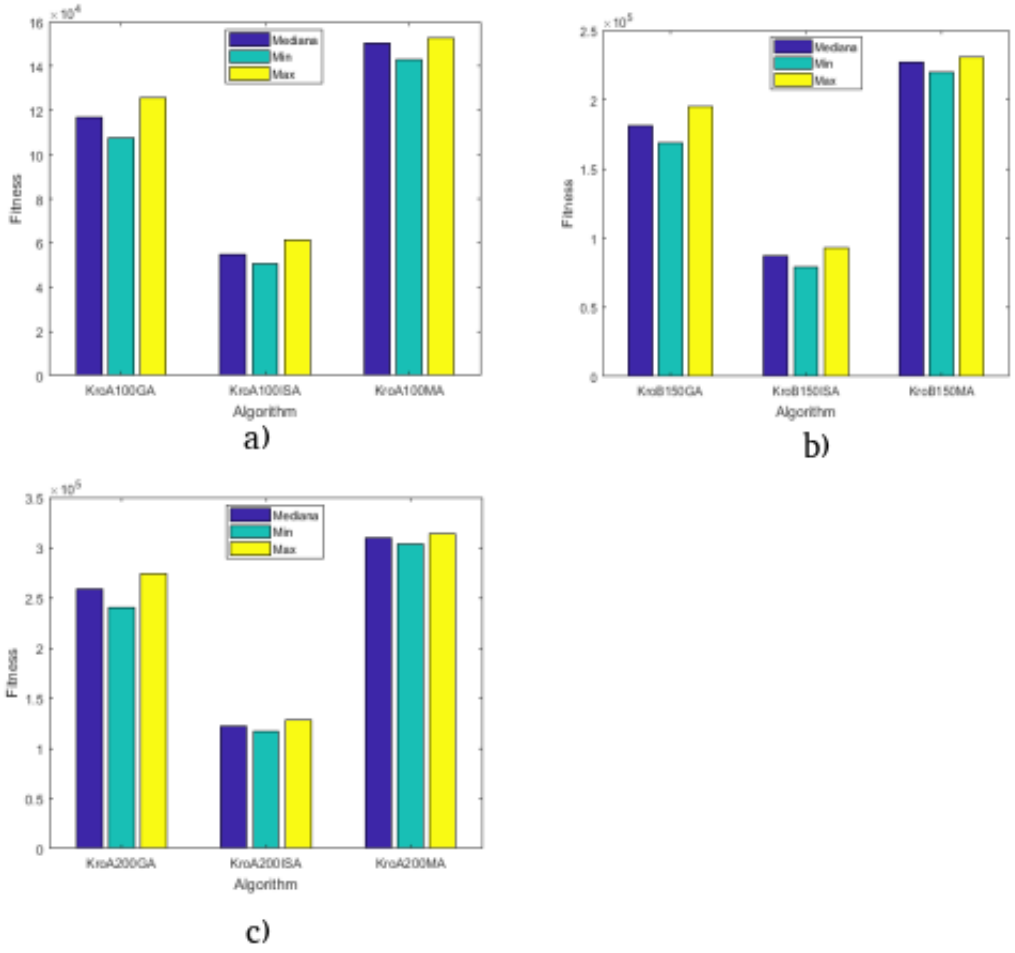

Figure 11: Medidas de tendencia de los experimentos realizados en las instancias a) KroA100,b) KroA150 y c) KRoA200.

5.5 Solución al problema TimeTabling

En la fig 12 se observan los resultados obtenidos en 33 distintos experimentos realizados de cada algoritmo para la solución al óptimo encontrado en la instancia 1. Las distintas instancias tuvieron un comportamiento muy parecido, las líneas de color rojo corresponden a los minimos encontrados dentro de cada generación mientras que la línea azul representa el promedio de la población, la esperanza dentro de ella. Podemos observar en la fig 12 a) al algoritmo genético en el cual el promedio y el minimo encontrado se encuentran bastante cercanos, debido a que la población se ve afectada en un porcentaje muy bajo, que a su vez puede o no presentarse entre cada generación, los cambios en el mínimo encontrado son pequeños y se mantienen por valgunas generaciones. En la fig 12 b) encontramos al algorimo memético, en este caso los saltos hacia un mínimo cada vez más cerca al óptimo son más grandes y se mantienen por más generaciones que el genético, en este caso el promedio de la población se ve directamente afectado ya que, los individuos mutan en cada generación. En el caso del sistema inmune el promedio y el mínimo encontrado tienen una separación menos pronunciada que en el genético, el sistema inmune tiene una velocidad de convergencia mayor debido a que

la exploración del mejor uindividuo entre cada generación recurre a una explotación determinada por la clonación del minimo entontrad, en este caso el criterio de paro para encontrar el mínimo fue el número de iteraciones o en su defecto, el fitness = 0. En la gráfica de Sistema inmune la diferencia en el número de iteraciones para encontrar el mínimo es bastante visible a comparacion del MA y GA respectivamente. En la fig 12 d) esta diferencia en la batalla por encontrar el óptimo es significativamente visible.

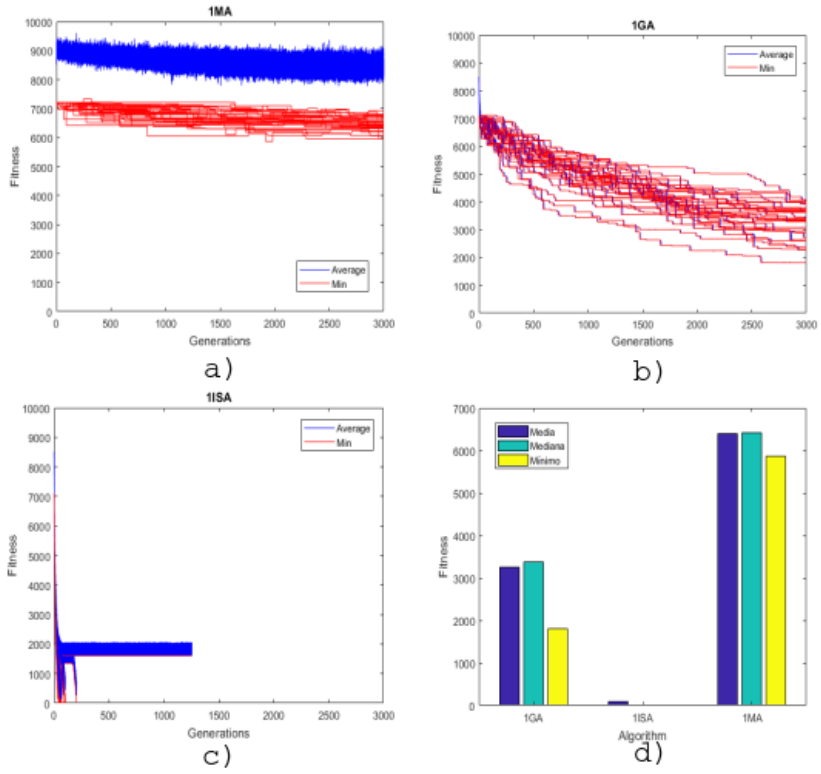

Figure 12: Solución al problema TT, a)MA, b)GA,c)ISA, d) Tendencias.

En la fig 13 se observa el comportamiento de los algoritmos trayectoriales y sus respectivas configuraciones a) ILS1, b) ILS2, c)SA1, d) SA2, de manera visual el algoritmo Recodido Simulado tiene un factor de convergencia al mínimo más pronunciado que el algoritmo ILS en ambas configuraciones.

Las tendencias de estos algorimos puede verse en la fig 14. En la cual se observa que el algoritmo SA en ambas configuraciones es mejor que la búsqueda local iterada.

5.6 Pruebas no paramétricas

Los algoritmos fueron sometidos a pruebas no paramétricas para determinar la existencia de diferencia estadísticamente significativa en el comportamiento entre cada uno. Las pruebas fueron realizadas mediante el software KEEL [14]. El nivel de significancia establecido fue del 0.05.Donde, el resultado obtenido fue la existencia de diferencia en el comportamiento de los algoritmos propuestos, con p-valores de 0.00335, 0.067, 0.000034731 para Friedman, Friedman Alineado y Quadetest respectivamente en el problema TSP,

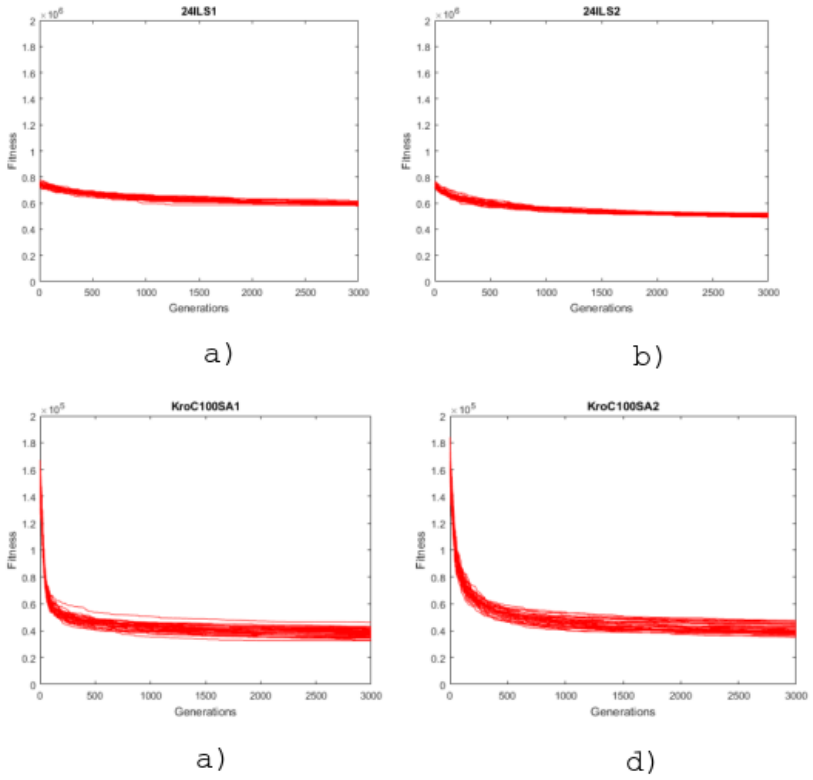

Figure 13: Algoritmos trayectoriales y sus respectivas configuraciones a) ILS1, b) ILS2, c) SA1, d) SA2.

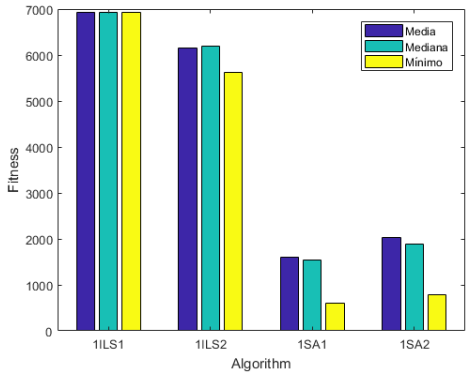

Figure 14: This is a caption

en el cual la conslusión es que existe diferencia significativa entre el comportamiento de los algoritmos.

En el caso de la prueba para los algoritmos poblacionales y el problema timetabling los p valores obtenidos fueron 0.000335, 0.06937, 0.00001567 respectivamente, obteniendo diferencia estadísticamente significativa de nuevo.

En la tabla 6 se pueden observar los lugares que ocupa cada algoritmo en el desempeño general. Siendo que,

Table 6: Rankeo Obtenido a partir de la aplicación de las pruebas No Paramétricas para los Algoritmos Poblacionales.

Algoritmo	Friedman TSP	QuadeTest TSP	Friedman TT	QuadeTest TT
GA1	2	2	2	9.5
MA1	3	3	7.5	7.5
ISA1	1	1	1	20.5

Table 7: Rankeo Obtenido a partir de la aplicación de las pruebas No Paramétricas para los algoritmos trayectoriales.

Algoritmo	Friedman TSP	QuadeTest TSP	Friedman TT	QuadeTest TT
ILS1	1.9375	1.9861	1	1
ILS2	1.0625	1.0139	2	2
SA1	4	4	4	4
SA2	3	3	3	3

no se muestra el ranking obtenido por Friedman Alineado ya que la prueba no pudo determinar diferencia significativa al obtener un p-valor mayor al nivel de significancia. Se muestra que el Algoritmo Sistema Inmune obtuvó mejores resultados que el Algoritmo Memético y el Algoritmo Genético respectivamente.

En la tabla 7 se pueden observar los lugares que ocupa cada algoritmo en el desempeño general. Siendo que, no se muestra el ranking obtenido por Friedman Alineado ya que la prueba no pudo determinar diferencia significativa al obtener un p-valor mayor al nivel de significancia. Se muestra que el Algoritmo Recocido Simulado obtuvo mejores resultados que el algoritmo de búsqueda local iterada.

Finalmete las pruebas no paramétricas realizadas para ambas familias de algoritmos, bajo un nivel de significancia del 0.05, los p valores fueron 0.00025, 0.103826, 0. Siendo dos de tres pruebas de nuevo las que aceptaron la diferencia en el comportamiento de los algoritmos.

Table 8: Rankeo Obtenido a partir de la aplicación de las pruebas No Paramétricas (Algoritmos Poblacionales vs Algoritmos Trayectoriales).

Algoritmo	Friedman TSP	QuadeTest TSP	Friedman TT	QuadeTest TT
GA	3.5	3.3889	20.125	4
ISA	5	5	46.5	7
MA	1.625	1.6667	15.5	2
ILS1	3.1875	3.2917	12.875	1
ILS2	1.6875	1.6528	17.5	3
SA1	7	7	44.25	6
SA2	6	6	42.75	5

6 Conclusiones

El algoritmo “Sistema Inmune” (ISA) tiene un mejor resultado que el algoritmo “Memético”(MA) y ‘Con el desarrollo de pruebas estadísticas con un nivel de significancia del 0.05, el Algoritmo Recocido Simulado SA1 y SA2 (con las configuraciones establecidas) obtuvo un mejor rendimiento que el algoritmo Búsqueda Local Iterativa ILS1 e ISL2 para los problemas TSP y TT, respectivamente. Los algortimos trayectoriales son más veloces en tiempo de ejecución no obstante el algoritmo ISA demostró ser mas robusto aunque la eficiencia del mismo afecta directamente los recursos de la computadora.

Como trabajo futuro queda pendiente la realización de las pruebas estadísticas pos’hoc para determinar el mejor algoritmo mediante pruebas estadísticas especializadas y no solo al rankeo de las pruebas ya realizadas.

References

[1] David B. Fogel. *Evolutionary Computation*. John Wiley & Sons Inc., oct 2005.

[2] David B Fogel. Introduction to evolutionary computation. In *Evolutionary Computation 1*. IOP Publishing Ltd.

[3] N Cruz Cortes and Carlos A Coello Coello. *Sistema inmune artificial para solucionar problemas de optimización*. PhD thesis, PhD thesis, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, 2004.

[4] J H Holland. Adaptation in Natural and Artificial Systems. *Ann Arbor MI University of Michigan Press*, Ann Arbor:183, 1975.

[5] Brad L. Miller and David E. Goldberg. Genetic Algorithms, Tournament Selection, and the Effects of Noise. 9, 11 1995.

[6] Angel Kuri. The Application of Genetic Algorithms to the Evaluation of Software Reliability. pages 29–49, 01 2010.

[7] Pablo Moscato. On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts - Towards Memetic Algorithms. 10 1989.

[8] Dong Li, Shulin Liu, and Hongli Zhang. Negative selection algorithm with constant detectors for anomaly detection. *Applied Soft Computing*, 36:618–632, 2015.

[9] Eduardo Morales Manzanares. Búsqueda Local Iterativa (Iterated Local Search), 2004.

[10] S. V. Semenovskaya, K. A. Khachaturyan, and A. G. Khachaturyan. Statistical mechanics approach to the structure determination of a crystal. *Acta Crystallographica Section A*, 41(3):268–273, May 1985.

[11] Nitasha Soni and Tapas Kumar. Study of various mutation operators in genetic algorithms. *International Journal of Computer Science and Information Technologies*, 5(3):4519–4521, 2014.

[12] Eugene Ruben Ramirez. *Using genetic algorithms to solve high school course timetabling problems*. PhD thesis, San Diego State University, 2010.

[13] Gerhard Reinelt. TSPLIB—A traveling salesman problem library. *ORSA journal on computing*, 3(4):376–384, 1991.

[14] Jesus Alcala-Fdez, Salvador Garcia, Francisco Jose Berlanga, Alberto Fernandez, Luciano Sanchez, M.J. del Jesus, and Francisco Herrera. KEEL: A data mining software tool integrating genetic fuzzy systems. In *2008 3rd International Workshop on Genetic and Evolving Systems*. IEEE, mar 2008.