Algoritmo Inmune de Selección Clonal para el problema de Job Shop Scheduling

Diaz Diaz Nelson Eduardo
Escuela de Ingeniería de Sistemas e Informática
Universidad Industrial de Santander
Bucaramanga, Colombia
nelson.diaz@correo.uis.edu.co

Luna Martínez Leydy Johana
Escuela de Ingeniería de Sistemas e Informática
Universidad Industrial de Santander
Bucaramanga, Colombia
leydy.luna@correo.uis.edu.co

Gomez Bueno Wilfredo Ariel
Escuela de Ingeniería de Sistemas e Informática
Universidad Industrial de Santander
Bucaramanga, Colombia
wilfredo.gomez@correo.uis.edu.co

Bautista Rozo Lola Xiomara
Escuela de Ingeniería de Sistemas e Informática
Universidad Industrial de Santander
Bucaramanga, Colombia
lxbautis@uis.edu.co

***Abstract*— En el presente artículo se establece una implementación de un algoritmo inmune artificial conocido como CLONALG para dar solución al problema de Job shop Scheduling (JSP), este es un problema de gran interés para la industria. Una de sus principales características es su naturaleza combinatoria y su complejidad, NP-Hard, la cual implica altos costos computacionales. Por tanto, se hace necesario abordar este tipo de problemáticas por medio de métodos que a pesar de que no encuentren soluciones óptimas en la mayoría de los casos, produzcan aproximaciones competitivas, a este tipo de métodos se les denomina metaheurísticas. Éstas garantizan soluciones aproximadas, y reducen el tiempo de cómputo, una de ellas los sistemas inmunes artificiales (SIA) que buscan, a partir de abstracciones, aprovechar las virtudes del sistema inmune biológico para brindar soluciones algorítmicas a problemas complejos en nuestro caso de optimización. El interés creciente en esta técnica de computación blanda radica en que es una técnica con amplio espectro de aplicación, ágil y poco demandante en términos complejidad computacional y uso de memoria.**

***Keywords*— CLONALG, Job Shop Scheduling, Optimización Combinatoria, Planificación de Trabajos, Sistemas Inmunes Artificiales**

I. INTRODUCCIÓN

Los problemas de Scheduling aparecen constantemente en la vida real en numerosos ambientes productivos y de servicios. Son problemas en los cuales se requiere organizar la ejecución de trabajos que compiten entre sí por el uso de un conjunto finito de recursos, y que a su vez están sujetos a un conjunto de restricciones.

Estos problemas son de naturaleza combinatoria, es decir, hay que elegir una entre un conjunto exponencialmente grande de combinaciones posibles. Este tipo de complejidad pertenece a la clase NP-Hard [1], en otras palabras, problemas en los que el tiempo de cómputo necesario para resolverlos crece desproporcionadamente conforme aumenta el tamaño del problema. Por esta razón los investigadores han desarrollado distintos métodos o algoritmos que procuren soluciones efectivas, ya sea de forma determinística o no [2].

Lo anterior plantea el reto de optimizar uno o varios criterios que se representan mediante funciones objetivo y que están relacionados con el costo, el beneficio o el tiempo de ejecución [3]. El objetivo de la optimización es maximizar o minimizar los criterios sujetos a las restricciones. Por ejemplo, maximizar la eficiencia de la utilización de las máquinas, minimizar el tiempo de ejecución, minimizar el tiempo de retardo entre trabajos, etc.

Debido a su complejidad, los problemas de scheduling precisan de algoritmos de búsqueda eficientes para encontrar soluciones aceptables en tiempos razonables [4]. Por lo que en la literatura se pueden encontrar aproximaciones a los problemas de scheduling basadas en los algoritmos de búsqueda heurística propios de áreas como la Investigación Operativa y la Inteligencia Artificial [5], [6], entre estos en la literatura se relacionan nuevas implementaciones de algoritmos inspirados en la naturaleza como lo son los sistemas inmunes artificiales, que son el objeto de estudio del presente artículo.

Este artículo se encuentra organizado de la siguiente forma: en la sección II se presenta un estado del arte de trabajos relacionados con Job Shop Scheduling en la sección III se presenta en detalle una descripción formal de Job Shop Scheduling. En la sección IV se hace una descripción de los datos empleados para el presente estudio. En la sección V se describe la metodología propuesta. En la sección VI se

presentan y discuten los resultados. Este trabajo finaliza en la sección VII donde se muestran las conclusiones.

II. REVISION DE LITERATURA

Durante las últimas décadas muchos investigadores han trabajado el problema de Job Shop Scheduling derivando en muchas soluciones. A continuación se presentan diferentes trabajos donde se comparan los Sistemas Inmunes Artificiales ante otras heurísticas:

En el trabajo "*An Artificial Immune Algorithm for the project scheduling problem under resource constraints*" de Mahdi Mobini et al. Aborda el uso de un sistema inmune artificial de Selección Clonal en el cual se tiene en cuenta la minimización de makespan como objetivo para la solución de Job Shop Scheduling. Los resultados obtenidos en el análisis computacional y la comparación con otros: algoritmos genéticos tales como búsqueda de dispersión, algoritmo genético hibrido entre otros, revelan un rendimiento aceptable y competitivo del algoritmo propuesto en dicho estudio [7].

El trabajo de Castro et al. "*An Artificial Immune Network for Multimodal Function Optimization*" presenta la adaptación de un modelo de red inmune diseñado especialmente para resolver problemas de optimización multimodales. Se comparó teóricamente con un algoritmo de selección clonal también aplicado para realizar la optimización multimodal, y estrategias de evolución [8].

En el trabajo de Coello et al "*Job Shop Scheduling using the Clonal Selection Principle"* se propone un algoritmo basado en un sistema inmune artificial (Selección Clonal) para solucionar Job Shop Scheduling. El Algoritmo de Sistema Inmune Artificial (AIS), propuesto es comparado con 3 algoritmos los cuales son: Algoritmo Genético Hibrido (HGA), Algoritmo Genético Paralelo (PGA) y GRASP. Los resultados indican que el método propuesto es muy competitivo con respecto a los otros, ya que presenta una mejora del 0.23%, 0.74% y 0.28% del AIS con respecto a los algoritmos HGA, PGA y GRASP [2].

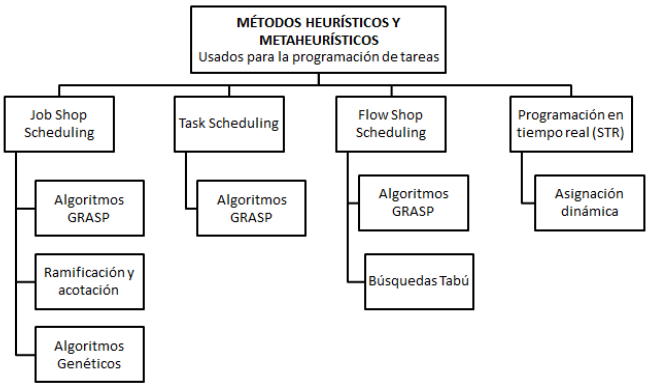
En el desarrollo de Quan Zuo y Shun Fan "*Solving the Job Shop Scheduling Problem by an Immune Algorithm*" se presenta un algoritmo inmune artificial (Selección Clonal) para resolver el Problema de Job Shop Scheduling JSP. El algoritmo inmune utiliza la tecnología de nicho para evitar óptimos locales y cuenta con sistema de caos para mejorar la eficiencia de búsqueda. Los resultados experimentales demuestran la eficacia del método para resolver problemas de programación de tareas puesto a medida que se reduce el retardo también se reducen las iteraciones para encontrar la mejor solución Cuando el retardo es de 200 el algoritmo necesita 39.5 iteraciones para encontrar la mejor solución, sin embargo si el retardo es de 10 el algoritmo se optimiza a 2,2 iteraciones [9].

El trabajo de Chandrasekaran et al. *"Solving job shop scheduling problems using artificial immune system"*, presenta un Algoritmo inmune de Selección Clonal, el cual se ha usado para resolver problemas de programación de tareas JSP con el objetivo de lograr la minimización del makespan. Los resultados son comparados con los algoritmos de Búsqueda Tabú y Procedimiento de Desplazamiento de Cuello de Botella (por sus siglas en inglés SBGLS1). Al compararse dichos algoritmos se obtiene un porcentaje de error relativo, el cual muestra que el algoritmo inmune tiene el menor porcentaje con 1,865 mientras que Búsqueda Tabú tiene 2,56 y Cuello de Botella 3,68 mostrando así que la Selección clonal genera mayor cantidad de valores óptimos [10].

El trabajo de Cortés et al. *"Use of an Artificial Immune System for Job Shop Scheduling"* propone un algoritmo basado en un sistema inmune artificial de Selección Clonal para resolver el problema de Job Shop Scheduling, también dos Procedimientos de Búsqueda Ciega Aleatorizado y Adaptativo GRASP y GRASP+RT (GRASP que utiliza un re-enlace de trayectoria). Los resultados indican que: el AIS mostró tener un porcentaje de desviación menor que GRASP pero mayor que GRASP+RT esto respecto a la mejor solución conocida. Esto se debe que el algoritmo hibrido de GRASP está dotado con memoria el cual lo hace más robusto [3].

Mientras la tecnología computacional avanza a pasos agigantados es importante en estos días optimizar la programación de una mejor manera con el fin de obtener mejores resultados, es por esto que se ha dado el uso de algoritmos inspirados en sistemas biológicos, dado que presentan una mayor optimización que el uso de heurísticas convencionales. En la figura 4 muestra las diferentes técnicas usadas para la solución a problemas de calendarización de tareas.

Figura 4: Métodos Heurísticos y Meta Heurísticos para la solución de problemas de programación de tareas.



Fuente :Tupia, M y Sánchez, M. D [11].

III. JOB SHOP SCHEDULING

En Job Shop Scheduling problem (JSP), un conjunto finito de trabajos es procesado sobre un conjunto finito de máquinas. Cada trabajo se caracteriza por un orden fijo de las operaciones, cada una de las cuales será procesada en una maquina específica por una duración especificada. Cada máquina puede procesar a lo más un trabajo en un tiempo y una vez que un trabajo ha iniciado sobre una maquina se debe completar su procesamiento sobre esa máquina por un tiempo ininterrumpido. Un Calendario es una asignación de operaciones en intervalos de tiempo sobre las máquinas. El makespan es el máximo tiempo en completar todos los trabajos. El objetivo de JSP es encontrar un calendario que minimice el makespan.

Formalmente, el JSP puede ser definido como se muestra en el trabajo de Witkowski et al [12]. Dado un conjunto M de máquinas ($|M|$ denota el tamaño de M) y un conjunto J de trabajos ($|J|$ denota el tamaño de J), sean $\sigma_1^j \prec \sigma_2^j \prec \cdots \prec \sigma_{|M|}^j$ sea el orden de un conjunto $|M|$operaciones del trabajo j, donde $\sigma_k^j \prec \sigma_{k+1}^j$ indica que la operación σ_{k+1}^j solo puede empezar el procesamiento después de completar la operación σ_k^j. Sea O el conjunto de operaciones. Cada operación es definida por dos parámetros: M_k^j es la maquina sobre la cual σ_k^j es procesada y $p_k^j = p(\sigma_k^j)$ es el tiempo de procesamiento de la operación σ_k^j. Definiendo $t(\sigma_k^j)$ como el tiempo de inicio de la $k-esima$ operación $\sigma_k^j \in O$, una formulación de programación disyuntiva para el JSP se muestra a continuación:

$$\min C_{max}$$

$sujeto\ a$:

$C_{max} \geq t(\sigma_k^j) + p(\sigma_k^j),\ para\ toda\ \sigma_k^j \in O,$

$(1a).\ t(\sigma_k^j) \geq t(\sigma_l^j) + p(\sigma_l^j),\quad para\ toda\ \sigma_l^j \prec \sigma_k^j,$

$(1b).\ t(\sigma_k^j) \geq t(\sigma_l^i) + p(\sigma_l^i) \vee$

$t(\sigma_l^i) \geq t(\sigma_k^j) + p(\sigma_k^j)\ para\ todo\ i,j \in J \ni M_{\sigma_l^i} = M_{\sigma_k^j},$

$t(\sigma_k^j) \geq 0, para\ toda\ \sigma_k^j \in O$

C_{max} Es el makespan a ser minimizado.

Una solución factible puede ser construida de una permutación de J sobre cada una de las máquinas M, observando las restricciones de precedencia, la restricción de que cada maquina puede procesar solo una operación a la vez y la restricción que garantiza el procesamiento de una operación de manera ininterrumpida en una maquina hasta ser completada. Cada conjunto de permutaciones tiene un correspondiente Calendario. Por tanto, el objetivo del JSP es encontrar un conjunto de permutaciones con el makespan más pequeño.

IV. DATOS

Para la implementación del algoritmo se usó la familia de instancias del problema de Job shop Scheduling conocidas como LA. Las instancias de Lawrence (LA) tienen 40 problemas de 8 diferentes tamaños propuestos [13]: 10 x 5, 15 x 5, 20 x 5, 10 x 10, 15 x 10, 20 x 10, 30 x 10 y 15 x 15. Lawrence llamó Fl-5, Gl-5, Hl-5, Al-5, Bl-5, Cl-5, Dl-5, y Il-5 a las instancias respectivamente. Sin embargo el nombre LA fue dado por Applegate y Cook [14] y es uno de los más comúnmente utilizados. Cada ejemplo se compone de una línea de descripción; cada fila contiene el número de puestos de trabajo y el número de máquinas, y luego una línea para cada puesto de trabajo, indicando el número de la máquina y el tiempo de procesamiento de cada paso del trabajo. Las máquinas se numeran empezando por 0, a continuación se presenta un ejemplo de una instancia de Lawrence (La01).

V. METODOLOGÍA PROPUESTA.

El diseño de la solución propuesta emplea permutación con repetición [15]. Esta representación permite codificar las diferentes soluciones del problema. La permutación con repetición asigna un orden de procesamiento a las operaciones que componen cada uno de los trabajos.

La figura 1 muestra una instancia 3x4, con tres trabajos y 4 máquinas. En la figura 2 se muestra la manera en que se asignan las operaciones que componen cada trabajo. Esta asignación se visualiza utilizando un diagrama de Gantt figura 3.

Figura 1 Instancia 3x4

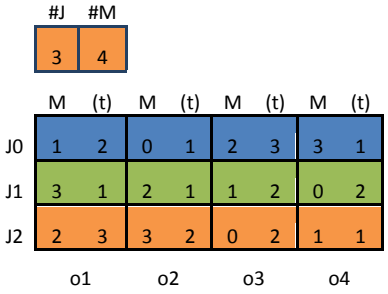
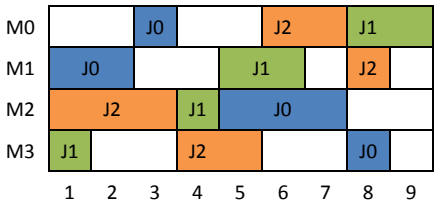


Figura 2 Permutación con repetición instancia 3x4

Para facilitar la visualización de la programación de los trabajos, se hace uso de planes de trabajos donde se acomodan los trabajos en las máquinas de tal forma que cumplan con las restricciones del problema, dicha visualización se conoce como diagrama de Gantt, como se observa en la figura 3. El makespan es de 9 unidades de tiempo, dado que la última operación termina en ese tiempo.

Figura 3: Diagrama de Gantt



Los diagramas de Gantt representan cada máquina en un renglón diferente y cada cuadro representa una operación. Los cuadros están marcados con el número de trabajo al que corresponde, es decir se identifica por el color y el número especificado, por último en el eje x se encuentran las unidades de tiempo que los trabajos gastan en completar cada una de las operaciones.

Algoritmo inspirado en el sistema inmune

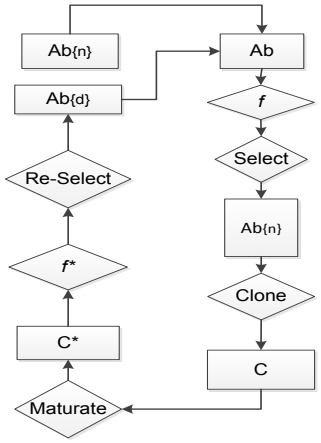
A. Selección Clonal

El algoritmo de selección clonal (CLONALG) [16] se inspira en la teoría de la selección clonal de la inmunidad adquirida, la cual implica la selección de anticuerpos (soluciones candidatas) basados en la afinidad, estas se puede dar por medio de la comparación con un patrón de antígeno o mediante la evaluación de la función de costo. Los anticuerpos seleccionados se someten a la clonación la cual es proporcional al valor de la afinidad y la hipermutación de clones que es inversamente proporcional al valor de la afinidad.

El conjunto resultante de clones compite con la población existente de anticuerpos para ser miembro de la siguiente generación. Los miembros de la población que presentan baja afinidad son remplazados por anticuerpos generados aleatoriamente. La variación del patrón de reconocimiento del algoritmo incluye la permanencia de la memoria del conjunto de soluciones las cuales en conjunto representan la solución al problema [17]

El algoritmo CLONALG presentado en este artículo está basado en la versión propuesta por Castro [16], [18] el cual se muestra en el diagrama de flujo de la Figura 5.

Figura 5: Diagrama de flujo de CLONALG



Fuente: Castro y Zuben [18]

El algoritmo empieza con la generación de la población. Esta población de anticuerpos tiene tamaño N. Cada individuo de la población corresponde a una solución candidata, que lleva acabo cada uno de los siguientes pasos:

1. El antígeno Ag corresponde a una función de costo llamada función de afinidad la cual depende del makespan, un anticuerpo $ab_{m,j}$ tiene un tamaño $(m\ x\ j)$ donde m es la cantidad de máquinas y j corresponde a la cantidad de trabajos para cada maquina. El arreglo del anticuerpo se presenta con una estructura como la mostrada en la figura 2.

2. Se calcula la afinidad de los anticuerpos $ab_{m,j}$ en relación al antígeno Ag. La afinidad f de los anticuerpos está dada por la ecuación (1):

$$f = 1 - \frac{makespan}{rango} \quad (1)$$

Donde el rango está dado por:

$$rango = \max(makespan) - \min(makespan) \quad (2)$$

El mayor y menor *makespan* corresponden al valor más alto y al más bajo en la población actual.

3. Los anticuerpos con afinidad más alta son seleccionados para ser clonados, generando el subconjunto de anticuerpos.

4. Se producirá una población de clones C. La cantidad N_c de clones para cada anticuerpo corresponde a un parámetro del algoritmo, el cual está dado por la ecuación:

$$N_c = (\beta * N) \quad (3)$$

Donde β es un factor de clonación y N es la cantidad de anticuerpos.

5. La población de clones se somete a un proceso de maduración de la afinidad (entre más alta la afinidad, menor la tasa de mutación), produciendo entonces una nueva población de clones.

Dicha tasa mutación esta dada por:

$$p = e^{(-\rho * f)} \quad (4)$$

Donde, ρ es el factor de mutación y f la afinidad.

6. La población de clones es evaluada, y su afinidad es calculada en relación al antígeno o función de costo.

7. Luego se seleccionan los anticuerpos maduros que tengan mayor afinidad. Estos serán la siguiente población, siempre y cuando la afinidad sea mayor que la de los anticuerpos originales.

Los peores anticuerpos son removidos y remplazados por una nueva población generada aleatoriamente.

VI. RESULTADOS Y DISCUSIÓN

Despues de un análisis de sensibilidad y bajo el apoyo de la literatura se determinaron los siguientes parametros para la configuracion del algortimo:

Tabla 1: Configuración Algoritmo CLONALG

Parámetro CLONALG	Valor
Población	100 anticuerpos
Numero de generaciones	300 generaciones
Factor de mutación	0,1
Factor de clonación	0,1
Factor de generación	10%

Para medir el desempeño de los algoritmos se realizaron comparaciones del número de evaluaciones de la función objetivo, que en el caso del problema corresponde al cálculo del makespan. Para medir la calidad de las soluciones, se realizó una comparación directa del makespan obtenido por el algoritmo propuesto. En cada una de las instancias se ejecutaron 500 iteraciones del algoritmo se seleccionó el menor makespan y se comparó con el mejor conocido. Las medidas estadísticas que se reportan son el promedio $\bar{x}$, la varianza S^2 y la desviación estándar; ver tabla 3.

Las técnicas fueron programadas en lenguaje Java y las pruebas se desarrollaron en un equipo portátil con sistema operativo Windows 7, procesador Intel Core I5 primera generación a 2.53 Ghz y 4Gb de memoria RAM. Estas prestaciones fueron suficientes y no se requirió de apoyo adicional en procesamiento ni memoria.

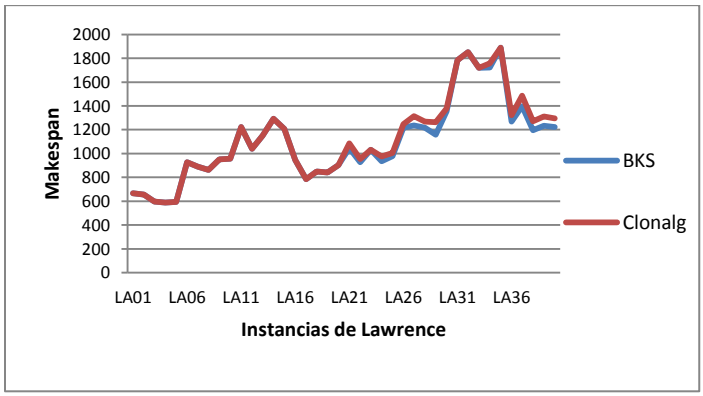
En la Tabla 2 y en la Figura 6, se observa el compendio de los resultados en cada instancia de Lawrence, se listan el mejor makespan C_{max} obtenido por el algoritmo CLONALG, el error relativo frente al makespan de la mejor solución conocida, además se incluye la mejor solución conocida, el tamaño de cada instancia y el número de evaluaciones para CLONALG.

Tabla 2: Comportamiento de los Makespan de los algoritmos para el JSP (Best Know Solution [BKS], Algoritmo de Seleccion Clonal [CLONALG].

Instancia	Tamaño	BKS	CLONALG		Evaluaciones CLONALG
			C_{max}	E.r %	
LA01	10 x 5	666	666	0.00	6902
LA02	10 x 5	655	655	0.00	18751
LA03	10 x 5	597	597	0.00	18541
LA04	10 x 5	590	590	0.00	13383
LA05	10 x 5	593	593	0.00	4
LA06	15 x 5	926	926	0.00	3133
LA07	15 x 5	890	890	0.00	26441
LA08	15 x 5	863	863	0.00	8182
LA09	15 x 5	951	951	0.00	3131
LA10	15 x 5	958	958	0.00	1121
LA11	20 x 5	1222	1222	0.00	6357
LA12	20 x 5	1039	1039	0.00	3313
LA13	20 x 5	1150	1150	0.00	5257
LA14	20 x 5	1292	1292	0.00	1153
LA15	20 x 5	1207	1207	0.00	27391
LA16	10 x 10	945	945	0.00	14252
LA17	10 x 10	784	784	0.00	27851
LA18	10 x 10	848	848	0.00	16662
LA19	10 x 10	842	842	0.00	22382
LA20	10 x 10	902	902	0.00	19941
LA21	15 x 10	1046	1085	3.73	100341
LA22	15 x 10	927	953	2.80	38011
LA23	15 x 10	1032	1032	0.00	61711
LA24	15 x 10	935	976	4.39	46591
LA25	15 x 10	977	1005	2.87	42701
LA26	20 x 10	1218	1248	2.46	109331
LA27	20 x 10	1235	1313	6.32	121141
LA28	20 x 10	1216	1269	4.36	57081
LA29	20 x 10	1157	1264	9.25	92211
LA30	20 x 10	1355	1379	1.77	49741
LA31	30 x 10	1784	1784	0.00	147701
LA32	30 x 10	1850	1850	0.00	228481
LA33	30 x 10	1719	1719	0.00	118301
LA34	30 x 10	1721	1756	2.03	100141
LA35	30 x 10	1888	1888	0.00	69801
LA36	15 x 15	1268	1322	4.26	65091
LA37	15 x 15	1397	1486	6.37	51611
LA38	15 x 15	1196	1272	6.35	96195
LA39	15 x 15	1233	1311	6.33	82691
LA40	15 x 15	1222	1294	5.89	68821

En la tabla 2 se observa la clasificación de los promedios de los errores relativos de cada algoritmo para cada uno de los diferentes tamaños de las instancias de Lawrence, 5 instancias corresponden a cada tamaño y en total son 8 tamaños diferentes por las 40 instancias. Se observa que los primeros cuatro tamaños presentan los promedios de los errores relativos más bajos, y los promedios de los errores relativos más altos se encuentran en las instancias de 20x10.

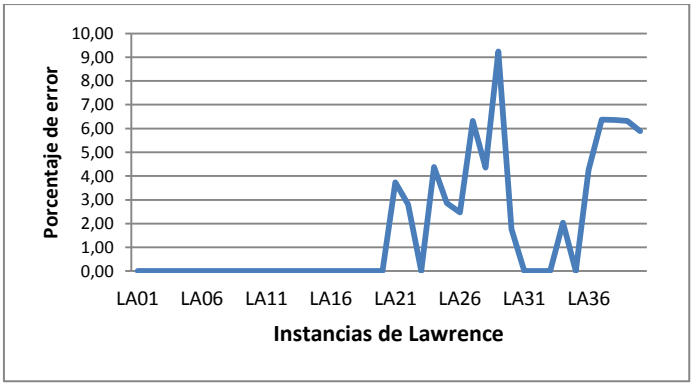
Figura 6: Comportamiento de los makespan para el problema de JSP (Best Know Solution [BKS], Algoritmo de Selección Clonal [CLONALG],



En la figura 7 se observa el comportamiento del error a lo largo de las 40 instancias del algoritmo implementado, se observa que el algoritmo CLONALG alcanza el BKS en las primeras 20 instancias. Se observa que conforme las instancias son más grandes y de mayor complejidad los errores crecen en la mayoría de los casos haciendo una excepción en la

instancia 23 y el rango de la 30 a la 35, este comportamiento es similar en los 2 algoritmos.

Figura 7 Comportamiento del error relativo para las instancias de Lawrence (algoritmo de selección clonal [CLONALG].



En la Tabla 3 se listan los errores relativos promedios, el mayor error la desviación estándar, y la varianza de cada uno de los algoritmos para el total de las 40 instancias.

Tabla 1 Medidas de calidad respecto al error relativo de los algoritmos (Algoritmo de Selección Clonal [CLONALG].

Algoritmo	CLONALG
Promedio Error Relativo %	1.73
Mayor Error Relativo %	9.25
Desviación	2.55
Varianza	6.529

Para verificar los resultados se consultaron algunos desarrollos reportados en la literatura de técnicas bioinspiradas similares y otras no bioinspiradas; como indicadores de calidad se contrastaron en función del error promedio, adicional se validan la cantidad y porcentaje de las mejores soluciones alcanzadas. Los algoritmos que se listan en la Tabla 4 son los siguientes con sus respectivas referencias tomadas de los trabajos dirigidos por el profesor Coello [2], [19]:

- (**AS**) Este algoritmo es una variante del algoritmo Ant System basado en el comportamiento de forrajeo de las hormigas. [20]
- (**AIS**) Algoritmo Inmune artificial que se basa en el principio de selección clonal y utiliza expansión clonal, así como hipermutación somática [19]
- (**CULT**) Algoritmo cultural basado en teorías sociales y arqueológicas las cuales tratan de modelar la evolución cultural. [19]
- (**INSA**) Heurística utilizada por Tabú Search para la construcción de la solución inicial. [21]
- (**TS**)Búsqueda Tabú es una metaheurística designada para encontrar una solución vecina óptima en problemas de optimización combinatoria. [21]
- (**GRASPL**) Algoritmo GRASP procedimiento de Búsqueda Miope Aleatoria y Adaptativa Utiliza una representación de grafos. [22]
- (**GA**) Es un algoritmo genético simple que utiliza representación de llaves aleatorias [23].

Tabla 4 Comparación resultados frente a algoritmos de la literatura para el JSP

Algoritmo	AS	AIS	CULT	INSA	TS	GRASPL	GA	CLONALG
Error Relativo Promedio %	0,34	0,16	0,97	9,26	0,05	1,87	0,92	1.73
Cantidad de BKS alcanzados	22	31	25	4	34	22	21	25
Porcentaje BKS alcanzado %	55	77,5	62,5	10	85	55	52,5	62.5

En la tabla 4 se muestra una comparación de la implementación realizada con otros algoritmos encontrados en la literatura. Se puede apreciar que CLONALG supera a GA, GRASPL, INSA, AS respecto al porcentaje de mejores soluciones encontradas y empata con CULT.

Tabla 5 Comparación evaluaciones frente a algoritmos de carácter evolutivo de la literatura para el JSP

Algoritmo	Evaluaciones promedio	Desviación	Max	Min
SIA	175058	219287,11	600328	27
CULT	454525	656763,39	1800000	2000
CLONALG	49796,05	49699,31	228481	4

En la tabla 5 se comparan algoritmos de naturaleza evolutiva en función de la cantidad promedio de evaluaciones requeridas para alcanzar las soluciones a las 40 instancias de Lawrence, se observa que el algoritmo implementado tiene un desempeño superior frente al de los algoritmos consultados en la literatura, en referencia al otro algoritmo inmune se requieren menos de la tercera parte de las evaluaciones promedio y frente al algoritmo cultural, la cantidad de evaluaciones promedio es aproximadamente 9 veces inferior.

VII. CONCLUSIONES

Los soluciones generadas por el algoritmo inmune implementado son competitivas frente a resultados presentados en la literatura especialmente para el caso de porcentaje de mejores soluciones conocidas alcanzadas, en la comparación se muestra que la implementación superó a cuatro de ellas, se igualo con una y fue inferior a dos de ellas.

Se observa que frente al porcentaje de error relativo la solución supera a tres técnicas de las consultadas en la literatura.

Respecto a los resultados de desempeño alcanzados por el presente trabajo, medidos respecto a la cantidad promedio de evaluaciones requeridas en las 40 instancias de Lawrence, se observa que se superó a las 2 técnicas de naturaleza evolutiva que se listaron de la literatura, y además se observa una disminución muy significativa en dicho parámetro.

Se concluye que las metaheuristicas bioinspiradas como la implementada en el presente trabajo presentan buenos rendimientos frente a las soluciones de otras técnicas de mayor madurez encontradas en la literatura.

VIII. BIBLIOGRAFÍA

[1] M. R. Johnson y G. a. D. S., «Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences),» *W. H. Freeman,* p. 340, 1979.

[2] C. A. Coello C, D. C. Rivera y N. C. and Cortés, «Job Shop Scheduling using the Clonal».

[3] C. A. Coello, D. C. Rivera y N. C. C, «Use of an Artificial Immune System for Job Shop Scheduling,» vol. 2787, 2003.

[4] M. R. Garey, D. S. Johnson and R. and Sethi, "The Complexity of Flowshop and Jobshop Scheduling.," *Mathematics of Operations Research.,* vol. 1, no. 2, pp. 117-129, 1976.

[5] F. Ghedjati, «Heuristics and a hybrid meta-heuristic for a generalized job-shop scheduling problem,» *IEEE Congress on Evolutionary Computation,* 8 July 2010..

[6] C. Blum y A. Roli, «Metaheuristics in Combinatorial Optimization : Overview and Conceptual Comparison,",» vol. 35, nº 3, pp. 268-308, 2003.

[7] M. Mobini, Z. Mobini y M. Rabbani, «An Artificial Immune Algorithm for the project scheduling problem under resource constraints.,» *Applied Soft Computing,* vol. 11, nº 2, pp. 1975-1982, 2011.

[8] L. N. de Castro y J. Timmis, «An artificial immune network for multimodal function optimization,» *Proceedings of the 2002 Congress on Evolutionary Computation.,* vol. 1, pp. 699-704, 2002.

[9] X. Zuo y Y. Fan, «Solving the job shop scheduling problem by an immune algorithm,» *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on,* vol. 6, pp. 3282-3287, 18-21 Aug. 2005.

[10] M. Chandrasekaran, P. Asokan, S. Kumanan, T. Balamurugan y S. Nickolas, «Solving job shop scheduling problems using artificial immune system,» *International Journal of Advanced Manufacturing Technology.,* vol. 31, nº 5-6, pp. 580-593, Jan 2006.

[11] M. Tupia y D. M. Sánchez, «Un algoritmo Grasp para resolver el problema de programación de tareas dependiente en maquinas diferentes,» pp. 129-139, Sep. 2004.

[12] T. Witkowski, P. Antczak y A. Antczak, «Solving the Flexible Open-Job Shop Scheduling Problem with GRASP and Simulated Annealing,» *Artificial Intelligence and Computational Intelligence (AICI),* vol. 2, pp. 437,442, 23-24 Oct. 2010.

[13] J. E. Beasley, «OR-Library: Distributing Test Problems by Electronic Mail.,» *Journal of the Operational Research Society,* vol. 41, nº 11, pp. 1069-1072, 1990.

[14] D. Applegate y W. Cook, «A study of the Job shop scheduling problem.,» *ORSA Journal on Computing,* vol. 3, pp. 149-156, 1991.

[15] T. Yamada y N. R, «Job Shop scheduling,» *Genetic Algorithms in Engineering Systems,* pp. 134-160, 1997.

[16] d. C. Leandro N. y Z. Fernando J. Von, «Learning and Optimization Using the Clonal Selection Principle,» vol. 6, nº 3, pp. 239-251, Jun 2002.

[17] J. Brownlee, «Clever Algorithms: Nature-Inspired Programming Recipes,» 2011.

[18] L. N. De Castro and F. J. Von Zuben, « "The Clonal Selection Algorithm with Engineering Applications,,» *in Proceedings of GECCO,* vol. 3637, pp. 36-37, July 2000.

[19] C. A. Coello y R. Landa, «A cultural algorithm for solving the job shop scheduling problem,» *Knowledge Incorporation in Evolutionary Computation Computation,* p. 37–55, 2005.

[20] E. T. Enriquez, «Uso de una Colonia de Hormigas on de Horarios Resumen," Laboratorio Nacional de Informática avanzada,» 2007.

[21] E. Ñowicki y C. Smutnicki, «A fast taboo search algorithm for the job shop problem,» *Management Science,* vol. 42, nº 6, pp. 797-813, 1996.

[22] S. Binato, W. J. Hery y D. M. Loewenstern, «A grasp for job shop scheduling,» pp. 1-17, March 2000.

[23] J. F. G. a. N. C. Beirao, «Um algoritmo gennetico baseado em chaves aleatorias para sequenciamento de operacoes,» *Revista Associacao Portuguesa de Desenvolvimento e Investigacao Operacional.,* vol. 19:123 137, 1999.

[24] L. N. D. C. a. F. J. V. Zuben, «The Clonal Selection Algorithm with Engineering Applications,» *," in Proceedings of GECCO, 2000, vol. 3637, no. July, pp. 36–37.*.

[25] D. Cortés Rivera, «Un Sistema Inmune Artificial para resolver el problema del Job Shop Scheduling,» *CINVESTAV,* 2004.