Resolviendo Problema de Calendarización Utilizando Optimización por Enjambre de Partículas y Algoritmos Genéticos

Adolfo Esteban Fragoso Magallanes

Facultad de Ciencias

Universidad Autónoma de Baja California
fragosoa@uabc.edu.mx

Resumen. En este trabajo se estudia una aproximación a la solución de un problema de calendarización basada en Algoritmos Genéticos (GA) y Optimización por Enjambre de Partículas Binario (BPSO). El principal objetivo de este trabajo fue probar si la solución a través de técnicas metaheurísticas representan una mejora a la solución determinística del problema.

Palabras clave: Algoritmo Genético, BPSO, Problema de decisión.

1 Introducción

Se tiene un conjunto de personas miembros de una familia, cada una se encuentra en una ciudad distinta a lo largo de los Estados Unidos. Todos desean viajar a la ciudad de Nueva York (NYC) desde su ciudad de orígen, se planea que todos los miembros de la familia lleguen el mismo día al aeropuerto y que todos se regresen el mismo día a sus respectivas ciudades, de ser necesario, rentar un hotel para hospedarse en caso de que sus vuelos salgan hasta el dia siguiente. Se tiene un conjunto de datos donde se tienen registrados todos los vuelos desde las distintas ciudades de orígen hacia Nueva York en diferentes horarios, así como los vuelos de regreso desde Nueva York hacia todas las ciudades. Los vuelos varían además del horario también en costo y duración del viaje. Se desea obtener una configuración apropiada de vuelos, de forma que el costo total de los vuelos de ida y retorno, así como el tiempo de espera en el aeropuerto entre todos los miembros de la familia sea el óptimo. Una solución que garantiza encontrar la combinación ideal de vuelos para este problema sería hacerlo utilizando fuerza bruta, donde se pruebe con cada combinación posible de vuelos para cada uno de los miembros de la familia, y en cada configuración calcular y guardar el valor de una función objetivo de forma que al finalizar la búsqueda se garantice encontrar la mejor solución. Si se considera m como el número de vuelos desde una ciudad cualquiera hasta NYC y de igual forma desde NYC hacia la ciudad de orígen, y n la cantidad de personas a viajar, se puede acotar el número de operaciones necesarias para resolver este problema como $\Theta(m^{2n})$ asumiendo que el número de operaciones para calcular el valor de la función objetivo está acotado como $\Theta(1)$. A pesar de que es fácil encontrar el valor óptimo, no se conoce mejor solución determinística más que explorar todas las posibilidades, lo cual trae consigo un alto costo computacional en términos de tiempo de ejecución.

Esto motiva a explorar técnicas alternativas que requieran invertir menor tiempo de ejecución para aproximar bien al valor óptimo. En este trabajo se presentan dos esquemas: *Algoritmos Genéticos* y *Optimización por enjambre de partículas binario* los cuales se comparan entre sí tanto en las soluciones obtenidas como en el tiempo de ejecución.

1.1 Algoritmos Genéticos

Los Algoritmos Genéticos, Genetic Algorithms (GA) son métodos computacionales de optimización y búsqueda, inspirados en el proceso genético de los organismos vivos. A lo largo de la las generaciones, las

poblaciones evolucionan en la naturaleza de acorde con los principios de la selección natural y la supervivencia de los más fuertes, postulados en la teoría de la evolución Darwiniana[1]. Los principios básicos de los Algoritmos Genéticos fueron establecidos por Holland[2], y se encuentran descritos en varios textos [3,4]. Los Algoritmos Genéticos usan una analogía directa con el comportamiento natural. Trabajan con una población de individuos, cada uno de los cuales representa una solución factible a un problema dado. A cada individuo se le asigna un valor o puntuación, relacionado con la bondad de dicha solución. En la naturaleza esto equivaldría al grado de efectividad de un organismo para competir por un recurso o por reproducirse. Cuanto mayor sea la adaptación de un individuo al problema, mayor será la probabilidad de que el mismo sea seleccionado para reproducirse, cruzando su material genético con otro individuo seleccionado. Este cruce permitirá nuevos individuos, los cuales comparten características de sus padres. A través del tiempo, es posible realizar un reemplazo generacional, sustituyendo todos los padres por los hijos o bien, aumentando el tamaño de la población conservando tanto padres como hijos y después seleccionar los mejores para la siguiente generación, a esto se le conoce como elitismo. Así, a lo largo de las generaciones las buenas características se propagan a través de la población y como consecuencia se encuentra una solución óptima al problema.

1.2 **Optimización por Enjambre de Partículas**

La optimización por enjambre de partículas, *Particle Swarm Optimization* (PSO), propuesto por Eberhart y Kennedy[5] como una técnica de optimización utilizada en espacios de números reales. Se inspira en el comportamiento de los enjambres como algunos insectos y cardúmenes de peces.

Consiste en encontrar soluciones potenciales al problema representado como una partícula con coordenadas x_{id} y una velocidad v_{id} en un espacio D-dimensional. Cada partícula i conformada por d dimensiones, mantiene un récord *personal* de la mejor posición encontrada hasta el momento en un vector p_{id}. Cada iteración comprende una evaluación de cada partícula, realizando un ajuste estocástico de v_{id} tanto en la dirección de la mejor partícula encontrada p_{gd} de toda la población como de p_{id}. De esta forma, el cambio en la velocidad de la i-ésima partícula está dada por la siguiente ecuación:

$$v_{id} = v_{id} + U(0, \phi_1)(p_{id} - x_{id}) + U(0, \phi_2)(p_{gd} - x_{id}) \qquad (1)$$

Donde $U(0, \phi)$ es un número aleatorio entre 0 y ϕ obtenido de una distribución uniforme. De igual manera, el cambio en la posición de la i-ésima partícula está dado por la siguiente ecuación:

$$x_{id} = x_{id} + v_{id} \qquad (2)$$

El algoritmo se ejecuta hasta que se encuentra el óptimo o no hay un mejoramiento significativo en la función objetivo.

Muchos problemas de optimización están dados en un espacio discreto, algunos ejemplos típicos incluyen a problemas los cuales requieren el ordenado de elementos enteros, como en problemas de calendarización o de ruteo. El problema que se aborda en este trabajo no es una excepción, por lo cual se considerará una versión modificada del PSO de forma que facilite trabajar en el espacio discreto.

1.2.1 **Optimización por enjambre de partículas binario (BPSO)**

Como se menciona, el algoritmo original trabaja ajustando las trayectorias de las partículas a través de las coordenadas. La pregunta es, ¿cuál es el significado de trayectoria y velocidad en un espacio discreto?.

En el espacio binario, una partícula puede moverse a las diferentes esquinas de un hipercubo, cambiando algún conjunto de bits. La velocidad puede describirse como la tasa de cambio de la posición con respecto al tiempo, en este caso al número de iteraciones. De esta forma,es posible definir a la velocidad en términos de cambios de probabilidad, es decir, la probabilidad de cambiar un bit. Así, si una partícula se mueve en un espacio restringido a cero y uno, la velocidad v_{id} representa la probabilidad de que x_{id} tome el valor de 1. Por ejemplo, si $v_{id} = 0.2$ entonces hay un 20% de probabilidades que x_{id} sea uno, y un 80% de que sea cero. A partir de esta explicación se tomará como base que la representación de las partículas en BPSO serán vectores binarios, donde x_{id} representa d-ésimo bit de la i-ésima partícula. Además, la velocidad dado que es una probabilidad, debe tomar valores en el intervalo $[0, 1]$. Observando la ecuación 1, se observa que si p_{id} y p_{gd} también están representados como vectores binarios, entonces $(p_{id} - x_{id})$ y $(p_{gd} - x_{id})$ darán como resultado $\{-1, 0, 1\}$ y pueden ser utilizados para ponderar la probabilidad v_{id} en la siguiente iteración. Así mismo, para

asegurarse que v_{id} se mantenga en el intervalo $[0, 1]$ se le aplicará una transformación logística $S(v_{id})$. Finalmente, el cambio de posición de la partícula se modificará mediante la siguiente ecuación:

$$x_{id} = 1 \; si \; U(0,1) < S(v_{id}) \qquad (3)$$
$$x_{id} = 0 \; en \; caso \; contrario$$

Resumiendo: BPSO difiere de PSO en **a)** La representación de las partículas es un vector binario, **b)** Se realiza una transformación logística a la velocidad v_{id} , **c)** la actualización de la velocidad está dada por la ecuación 3, en lugar de la ecuación 2.

2 Metodología

Se habló sobre el problema que se trata en este trabajo: se tiene un conjunto de $n = 6$ personas que parten cada una, de alguna de las ciudades siguientes $\{Boston, Dallas, Akron, Miami, Chicago, Omaha\}$ hacia un destino común, el aeropuerto de La Guardia (LGA) en la ciudad de Nueva York. Se tiene un conjunto de datos el cual tiene almacenado $m = 10$ vuelos desde cada una de las ciudades de orígen hacia LGA y de regreso. La tabla 1 muestra a las personas y sus ciudades de origen, mientras que la tabla 2 muestra un ejemplo de cinco entradas del conjunto de datos.

Tabla 1: Datos de orígen de los miembros de la familia

Nombre	Ciudad de orígen
Adolfo Fragoso	Boston
Valeria Pringle	Dallas
Erasmo Sauceda	Akron
José Ayala	Miami
Alan Alvarado	Chicago
Everardo Gutierrez	Omaha

Tabla 2: Ejemplo de entradas del conjunto de datos. El conjunto de datos completo contiene 10 vuelos desde Boston (BOS) hacia LGA y otros 10 desde LGA a Boston, de igual forma hay 10 vuelos desde Miami (MIA) hacia LGA y 10 de regreso, así para una de las 6 ciudades.

Ciudad de orígen	Ciudad destino	Hora de salida	Hora de llegada	Costo ($)
LGA	MIA	20:27	23:42	169
MIA	LGA	19:53	22:21	173
LGA	BOS	6:39	8:09	86
BOS	LGA	6:17	8:26	89
LGA	BOS	8:23	10:28	149

Se desea encontrar la configuración adecuada de vuelos tanto de ida como de regreso que minimicen el costo total tanto del precio de los vuelos, como el tiempo de espera de las personas en el aeropuerto.

2.1 Representación de las soluciones

Para una solución dada independientemente del esquema desde el cual se ataque el problema, es necesario considerar solamente los vuelos que tomará cada persona. En el caso de GA, es posible representar un cromosoma como una lista de enteros positivos de tamaño $2n$. Donde el i-ésimo elemento (gen) corresponde al número de vuelo que toma la i-ésima persona desde su ciudad de orígen hasta LGA, y el (i+1)-ésimo elemento será el número de vuelo que toma la i-ésima persona desde LGA de regreso a su ciudad de orígen, donde $i = \{1, 3, 5...11\}$. Por ejemplo, considerando lo siguiente:

$$\{0, 3, 2, 3, 6, 3, 5, 2, 1, 2, 4, 2\}$$

Representa una solución en la cual Adolfo toma el primer vuelo (para propósitos de implementación, consideramos el conteo de vuelos a partir de 0) desde Boston hacia LGA, y el cuarto desde LGA a Boston. Similarmente Valeria toma el tercer vuelo desde Dallas hacia LGA y el segundo vuelo desde LGA a Dallas.

Análogamente para el caso de BPSO, la representación de las partículas es muy similar a como se hizo con los cromosomas en GA, únicamente difiere en que la representación de los elementos de la lista están en su forma binaria. Tomando en cuenta el ejemplo anterior, se muestra en la figura 1 la representación de una solución aplicable para el BPSO.

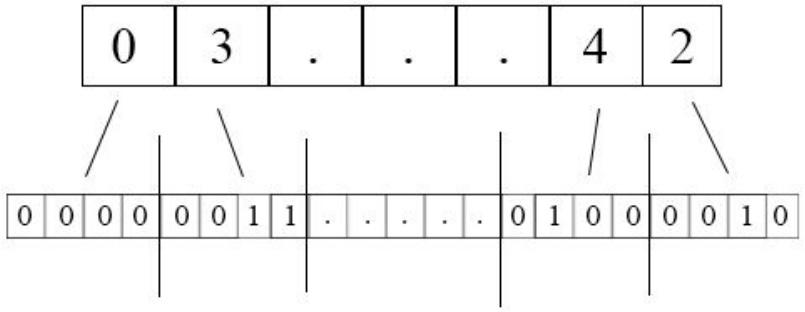

Figura 1: Representación de una solución para BPSO. Considerando el número de vuelos por ciudad que es constante con $m = 10$, los valores posibles que puede tener cada elemento de una solución cualquiera están en el intervalo $[0, 9]$ por tanto, una representación a 4 bits de los valores de la solución es apropiado.

2.2 Selección de parámetros

Se abordará el problema utilizando dos meta-heurísticas distintas: Algoritmos Genéticos y Optimización por Enjambre de Partículas Binario. En en caso de GA se dió una población inicial de 300 individuos, y el número máximo de iteraciones se estableció en 200. Análogamente, en el caso de PSO el número de partículas se estableció en 300, y el número máximo de iteraciones se dejó como 200. Se sabe que las funciones objetivo de problemas combinatorios pueden caer en óptimos locales, por lo que una estrategia de *diversidad* en las soluciones es apropiada para atacar este problema. Una forma de hacerlo, es establecer un tamaño de población grande, por tanto en ambos algoritmos el tamaño de población/número de partículas se decidió por 300, de igual forma es necesario que ambos algoritmos tengan los mismos parámetros para que estuvieran bajo iguales condiciones.

Los parámetros elegidos para correr GA fueron: probabilidad de mutación : $mutprob = 0.2$, porcentaje de individuos elegidos para la siguiente generación: $elite = 0.2$ empleando un esquema $(\mu + \lambda)$. Además se incluye un parámetro de probabilidad de cruzamiento para individuos débiles $weakcrossover = 0.05$ que elige individuos con fitness bajos para el cruzamiento. La mutación es de tipo 1-punto, y la mutación de los cromosomas consta de alterar el valor de un gen en ± 1. Cabe mencionar que no se siguió ningún método de optimización de selección de parámetros para el GA, los valores establecidos fueron elegidos de manera intuitiva.

En el caso de BPSO se realizó una búsqueda exhaustiva para encontrar la combinación de parámetros ϕ_1 y ϕ_2 que mejores resultados diera en la función objetivo. Se probaron los valores $\phi = \{0, 0.2, 1\}$ ya que recordando la definición de velocidad en la ecuación 1, dependiendo de los valores que se asignen a ϕ_1 y ϕ_2 se dará más importancia a la parte de $(p_{id} - x_{id})$ o a la parte de $(p_{gd} - x_{id})$ respectivamente. Considerando las reglas de selección de parámetros de PSO, se necesitaban considerar todos los casos:

- $\phi_1 > 0$, $\phi_2 = 0$
- $\phi_1 = 0$, $\phi_2 > 0$
- $\phi_1 > \phi_2$
- $\phi_1 < \phi_2$
- $\phi_1 = \phi_2 > 0$
- valores grandes de ϕ_1 y ϕ_2
- valores pequeños de ϕ_1 y ϕ_2

Para este problema en específico se considera grande un valor de $\phi = 1$ ya que recordando la ecuación 3, se realiza una transformación logística a v_{id} como $S(v_{id}) = \frac{1}{1+e^{-v_{id}}}$ por tanto, al tener un valor de ϕ más grande que 1 provocaría que $S(v_{id})$ converja a 1, lo cual es innecesario puesto que se está considerando como un valor de probabilidad. Para tener un contexto sobre los parámetros elegidos, primero se definirá la función objetivo antes de mostrar los resultados de este experimento.

2.3 **Función objetivo**

La función objetivo es el punto clave para resolver cualquier problema de optimización, usualmente es complicado determinar una función objetivo apropiada. Para este problema podríamos considerar muchas variables relevantes, por ejemplo, el precio total, tiempo de viaje, tiempo de espera. Hay muchas posibilidades para definir la función objetivo, la función propuesta en este trabajo toma en cuenta los siguientes aspectos:

- El costo total del viaje: representado como el precio del vuelo de la ciudad de orígen a LCA y de regreso para cada uno de los miembros de la familia.
- El tiempo total de espera en los aeropuertos: Cada miembro llega a LGA desde su ciudad de orígen y tienen que esperar en el aeropuerto hasta que el último miembro llegue. El costo por tiempo de espera está dado por $1 x minuto.
- Cuando la familia tiene que regresar a sus ciudades, deben llegar juntos al aeropuerto y esperar sus vuelos. De igual forma, el costo por tiempo de espera está dado por $1 x minuto.
- Finalmente, si la última llegada a LCA es más tarde que la primera salida desde LCA esto quiere decir que la familia tendrá que pagar un costo extra de $50 para hospedarse en un hotel y pasar la noche, para al siguiente día tomar el vuelo que sea más temprano.

Se define la función objetivo como $cost(s)$. Donde s es una solución representada como se hizo con GA. La conversión de la representación binaria de BPSO a GA y viceversa es despreciable. El pseudocódigo de la función objetivo se muestra a continuación:

```
1 //Pseudocódigo 1: Función objetivo
2 //Parámetros: s,una solución representada como una lista de enteros
3 //Funciones auxiliares:
```

```
4     // obtenerUltimaLlegada: regresa la hora del último vuelo a LGA
5     // obtenerPrimeraSalida: regresa la hora del primer vuelo
6     //                              desde LGA
7  función cost(s):
8     costo_total = 0
9     para cada persona p:
10          costo_total += costo_de_vuelo_de_ida_de_persona p
11          costo_total += costo_de_vuelo_de_regreso_de_persona p
12    ultima_llegada = obtenerUltimaLlegada
13    primera_salida = obtenerPrimeraSalida
14    tiempo_de_espera = 0
15    para cada persona p:
16          tiempo_de_espera += ultima_llegada - hora_de_llegada_de_p
17          tiempo_de_espera += hora_de_salida_de_p - primera_salida
18    si ultima_llegada>primera_salida:
19          costo_total += 50
20    regresa costo_total+tiempo_de_espera
```

2.4 **Atacando el problema**

Utilizando la función objetivo descrita anteriormente, así como los parámetros explicados en la sección anterior, se realizaron pruebas para atacar el problema utilizando los algoritmos GA y BPSO. Se presentan los pseudocódigos de ambos esquemas:

Algoritmo Genético

```
1 //Pseudocódigo 2: GA para problema de decisión
2 //Parámetros:
3 // mutprob = 0.2: Probabilidad de mutación
4 // popsize = 300: tamaño de población
5 // elite = 0.2  : porcentaje de individuos seleccionados
6 // maxiter = 200: máximo número de iteraciones
7 // Funciones Auxiliares:
8     //selecciona_mejores_individuos(elite): Selecciona
9     //  el  elite%  de  los  mejores  individuos  de  la  población
utilizando cost(s), la función objetivo
10    //selecciona_individuos_crossover:
11    //    elige dos individuos de los seleccionados para
12    //    hacer crossover, en el 5% de los casos
13    //    elige a cualquiera de toda la población
14 // Regresa: una lista de números enteros, representando al mejor
15    // individuo encontrado
16 geneticOptimize(mutprob,popsize,elite,maxiter):
17    pop = genera_poblacion_inicial(popsize)
18  repetir no máximo número de iteraciones
19    nueva_poblacion = selecciona_mejores_individuos(elite)
20    mientras nueva_poblacion.tamaño < popsize:
21          i1,i2 = selecciona_individuos_crossover
22          nuevo_individuo = crossover(i1,i2)
23          offspring = mutate(nuevo_individuo)
24          nueva_poblacion.insertar(offspring)
25    poblacion = nueva_poblacion
26  regresar mejor_individuo
```

BPSO

```
1 //Pseudocódigo 3: BPSO para problema de decisión
2 // Parámetros:
3 //  popsize = 300: tamaño de la población
4 //  maxiter = 200: máximo número de iteraciones
5 //  phi1 = 1     : parámetro para ajuste de velocidad
6 //  phi2 = 0.2   : parámetro para ajuste de velocidad
7 //BPSO(popsize,maxiter,phi1,phi2):
8     particulas = genera_conjunto_de_particulas(popsize)
9     repetir mientras no número máximo de iteraciones
10          para cada partícula x:
11               calcula p_id para cada x_id
12          calcula p_gd de todas las x
13          para cada particula x:
14               para cada componente d de x:
15                    c1 = U(0,phi1)
16                    c2 = U(0,phi2)
17                    actualiza v_id con ecuación 1
18                    si random < sigmoid(v_id)
19                         x_id = 1
20                    si no
21                         x_id = 0
22    regresa p_gd
```

Se observa que en el algoritmo finalmente se utilizan los valores de $\phi_1 = 1$ y $\phi_2 = 0.2$. Estos valores fueron elegidos después de realizar una búsqueda exhaustiva de los parámetros propuestos en la sección anterior. Se realizó haciendo 30 ejecuciones del algoritmo de BPSO contemplando las nueve combinaciones posibles de parámetros, para cada una se obtuvo una muestra de tamaño 30 donde se calculó el promedio y desviación estándar muestrales. A continuación se muestra la figura 2 acompañada de la tabla 3 con los datos generados por la búsqueda. Puede observarse que la combinación C8, corresponde al menor costo, con un costo promedio de 2463.83 y una desviación estándar de 117.19. De igual forma, C9 tiene un promedio de 2524.03 que es también considerablemente bajo, en comparación con el resto, pero tiene una desviación de 187.93.

A partir de los datos muestrales, se realiza la prueba de los rangos con signo de Wilcoxon al 5% para determinar si hay una diferencia significativa entre las medias de BPSO (C8) y BPSO (C9), obteniendo para la prueba de una cola, un valor p de 0.123, como no es menor a 0.05, no se puede rechazar la hipótesis nula, y se concluye que no hay diferencia significativa con un nivel de confianza del 95%. A pesar de eso, consideramos C8 como el mejor conjunto de parámetros ya que tienen menor costo promedio y menor desviación estándar.

Tabla 3: Registro de costo promedio realizando 30 ejecuciones para cada combinación de parámetros. Nota que C1 corresponde a la combinación $\phi_1 = 0, \phi_2 = 0$ de la figura 2, C2 a $\phi_1 = 0, \phi_2 = 0.2$ y así sucesivamente.

	C1	C2	C3	C4	C5	C6	C7	C8	C9
Costo Promedio($)	3429.76	2612.90	2542.09	2931.53	2596.16	2568.96	2629.80	2463.83	2524.03
Desviación	134.96	159.55	201.37	96.29	158.40	178.29	83.76	117.19	187.93

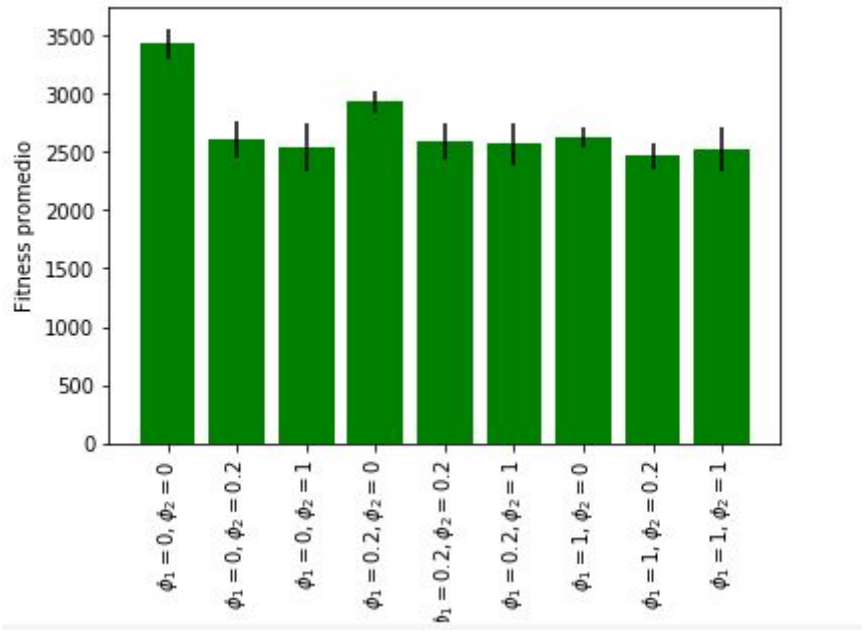

Figura 2: Costo promedio y desviación estándar de 30 ejecuciones por cada combinación de parámetros

3 Resultados

Tanto para el Algoritmo Genético como para el BPSO se hicieron pruebas realizando para cada algoritmo 30 ejecuciones y se consideró tanto el costo promedio como el tiempo de ejecución. La tabla 4 muestra los datos del costo promedio y desviación estándar de ambos algoritmos. Mientras que la tabla 5 muestra los datos del tiempo de ejecución promedio y desviación estándar.

Tabla 4: Comparación del costo promedio y desviación del GA contra BPSO configurado con los parámetros de C8.

	Algoritmo Genético	**BPSO (C8)**
Costo promedio ($)	2544.46	2463.83
Desviación estándar	187.60	117.19

Tabla 5: Comparación del tiempo promedio y desviación del GA contra BPSO configurado con los parámetros C8

	Algoritmo Genético	**BPSO (C8)**
Tiempo promedio (s)	43.55	104.37
Desviación estándar	1.52	0.27

Se observa que los datos de costo promedio y desviación estándar del algoritmo genético son muy similares a los datos generados de BPSO con C9. Se realiza la prueba de hipótesis, para probar si hay una diferencia significativa entre el costo promedio de GA y el costo promedio de BPSO (C8) realizando la prueba de los rangos con signo de Wilcoxon al **5%**. Obteniendo un **valor p de 0.03**, de esta forma se rechaza la hipótesis nula y se concluye que hay diferencia significativa.

De igual forma, observando la tabla 5, notamos que a pesar de que BPSO se desempeña mejor en la optimización de la función objetivo, también tiene un tiempo mayor de ejecución. Lo cual es razonable ya que la representación de las partículas en BPSO contiene más datos y por tanto requiere un mayor tiempo de cómputo para procesarlos.
Las figuras 3 y 4 muestran el desempeño del fitness a lo largo de las 30 ejecuciones, al igual que la tabla, notamos que para el caso de GA hay mayor variabilidad en las soluciones, mientras que en BPSO hay menos.

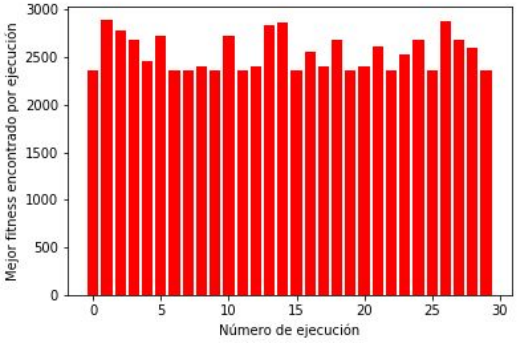

Figura 3: Fitness obtenido con GA por ejecución

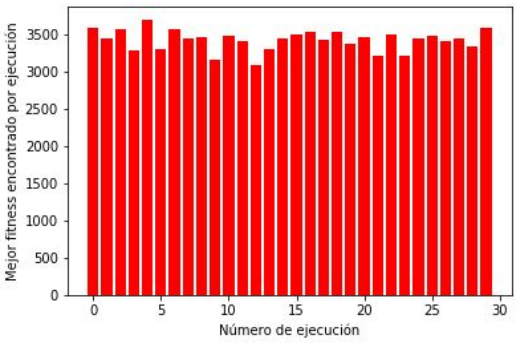

Figura 4: Fitness obtenido con BPSO (C8) por ejecución

4 Conclusiones

Observando los resultados obtenidos podemos notar que BPSO tiene un mayor rendimiento en cuanto a costo en comparación con GA para este problema en particular, aunque los parámetros del algoritmo genético no fueron optimizados como el caso de BPSO. Resultaría de interés cerrar el análisis del trabajo realizando una optimización de los parámetros para GA, y volver a comparar resultados, para propósitos de este trabajo se buscaba explorar las posibles aplicaciones de BPSO. De igual forma, analizando los resultados en cuanto a tiempo de ejecución, GA es notablemente más rápido que BPSO, resulta importante destacar que dependiendo del problema que se quiera atacar, valdrá la pena invertir más tiempo para obtener mejores resultados o no. Finalmente se concluye que independientemente del método utilizado, se tiene una solución buena del problema en términos de tiempo de ejecución, mucho mejor a lo que pudo haber obtenido la técnica por fuerza bruta.

Referencias

[1] C.Darwin (1859). *On the Origin of Species Means of Natural Selection*,Murray,London.
[2] J. Holland(1975). *Adaptation in Natural and Artificial Systems*, University of Michigan Press,Ann Arbor.
[3] D.E. Goldberg (1989), *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA.
[4] L. Davis, J.C. Principe (1991). *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York.
[5] Eberhart, R. C., and Kennedy, J. (1995). *A new optimizer using particle swarm theory*. Proc. Sixth Intl. Symposium on Micro Machine and Human Science(Nagoya, Japan), IEEE Service Center,Piscataway, NJ, 39-43.