

Algoritmo Genético PARALELO PARA el Problema de la Cadena más Alejada de las Demás

Osvel Chávez Hernández



16 de abril de 2018

CICESE

Índice

[Resumen 2](#_Toc511674124)

[Problema de la cadena más alejada de las demás 3](#_Toc511674125)

[Planteamiento del problema 3](#_Toc511674126)

[Trabajo realizado 3](#_Toc511674127)

[¿Qué se quiere hacer? 4](#_Toc511674128)

[Materiales y métodos 4](#_Toc511674129)

[Algoritmo Genético (AG) 4](#_Toc511674130)

[Optimización por Colonia de Hormigas (ACO por sus siglas en inglés) 5](#_Toc511674131)

[Algoritmo Genético Paralelo (AGP) 6](#_Toc511674132)

[Software 6](#_Toc511674133)

[Clase MainFFMSP 7](#_Toc511674134)

[Clase Iterator 7](#_Toc511674135)

[Clase Ecosystem 7](#_Toc511674136)

[Clase Island 7](#_Toc511674137)

[Clase Build 7](#_Toc511674138)

[Clase Nest 7](#_Toc511674139)

[Clase Individual 7](#_Toc511674140)

[Clase Person 8](#_Toc511674141)

[Clase Ant 8](#_Toc511674142)

[Clase Route 8](#_Toc511674143)

[Clase Solution 8](#_Toc511674144)

[Resultados y discusión 8](#_Toc511674145)

[Propuestas de mejoras 9](#_Toc511674146)

# Resumen

El problema de la cadena más larga consiste en encontrar una cadena que sea lo más diferente posible a un conjunto de cadenas de entrada siguiendo un criterio de evaluación. En el presente trabajo se pretende obtener un algoritmo basado en Algoritmos Genéticos Paralelos que genere cadenas largas diferentes a las demás.

# Problema de la cadena más alejada de las demás

El problema de la cadena más alejada de las demás (FFMSP por sus siglas en inglés), consiste en encontrar una cadena cuya distancia se encuentra por encima de un umbral para tantas cadenas de entrada como sea posible.

# Planteamiento del problema

Una instancia del FFMSP consiste en la tripleta , donde:

* es un conjunto de símbolos finitos llamado alfabeto.
* es un conjunto finito de cadenas que se construyen desde los símbolos del alfabeto , con , donde para y es la longitud para todas las cadenas .
* es el umbral de comprobación para decir que tan alejada es la cadena.

La Distancia de Hamming (HD por sus siglas en inglés), plantea que: sean y dos cadenas finitas construidas a partir del mismo alfabeto de símbolos; la distancia entre ellas, es el conteo de los símbolos que son diferentes en la misma posición:

(1)

La solución candidata es una cadena de símbolos construida desde los símbolos del alfabeto , . Se dice que la cadena es alejada de si . El objetivo es encontrar una cadena que se encuentre alejada de todas las , por lo que es necesario maximizar la función (2).

# Trabajo realizado

Como el FFMSP es NP-duro, es atractivo para atacarlo con heurísticas. En el artículo A GRASP-based memetic algorithm with path relinking for the far from most string problema, proponen un algoritmo llamado Memetic Algorithm (MA), el cual usando evolución genética obtienen mejores resultados que los reportados por GRASPMou y GRASPFFR.

Los algoritmos se compararon usando los parámetros de entrada:



Se usaron dos conjuntos de cadenas de entrada diferentes:

* Conjunto aleatorio, seleccionado con una distribución uniforme, según los parámetros .
* Conjunto real, seleccionado de forma aleatoria de un segmento de genoma real llamado Phytophthora Ramorusm’s, disponible en: <http://genome.jgi.doe.gov/Phyra1_1/Phyra1_1.download.htm>.

# ¿Qué se quiere hacer?

Haciendo uso de Evolución Genética (EG), se pretende mejorar los resultados obtenidos por el MA, tanto para cadenas reales como para cadenas obtenidas aleatoriamente.

# Materiales y métodos

## Algoritmo Genético (AG)

Es un algoritmo de permutación, donde ha partir de dos individuos de la población (padres) se obtienen dos nuevos individuos (hijos) permutando el material genético de los padres.

Para la solución propuesta se implementó AG con los siguientes parámetros:

* La longitud de la población es de individuos.
* La longitud de los cromosomas de los individuos es de .
* La selección de los padres se realiza por torneo: al mejor de 5 individuos tomados aleatoriamente.
  + La probabilidad de cruzamiento entre los padres es de 0.9 en una escala de 0 – 1.
  + El cruzamiento se realiza por un corte de cruce, obteniendo el hijo el material genético de uno de sus padres desde el inicio de su genoma hasta el punto de corte, y obteniendo el material genético del otro padre, desde el punto de corte hasta el final del genoma.
    - El corte se realiza mediante una variable aleatoria que tiene valores entre 0.6 – 0.9 \* .
  + La probabilidad de mutación del hijo obtenido es de 0.2.
    - La mutación se realiza por intercambio del material genético del cromosoma de dos elementos tomados al azar.
* La selección de los individuos para la siguiente generación: se mantiene a los mejores 5 individuos obtenidos del cruzamiento por torneo.

## Optimización por Colonia de Hormigas (ACO por sus siglas en inglés)

Es un algoritmo donde ha partir de un grafo de posibles caminos, un actuador, en este caso hormigas, se mueven por cada nodo del grafo, hasta generar una solución al final de su recorrido. Las conexiones entre los nodos del grafo contienen feromonas, una sustancia biológica volátil, la cual es segregada por las hormigas y seguidas por el resto de la colonia de hormigas.

Para la solución propuesta se implementó ACO con los siguientes parámetros:

* La longitud de la colonia es de hormigas.
* La longitud de todos los posibles caminos es de .
  + El grafo de caminos tiene niveles.
  + En cada nivel encontramos un nodo por cada símbolo del alfabeto .
  + Cada nodo se encuentra enlazado con todos los nodos del siguiente nivel.
  + El enlace entre cada par de nodos, contiene feromonas.
    - El valor de las feromonas se encuentra entre 0.25 – 0.75 \* .
    - Al inicio cada camino tiene un valor de feromona de 0.5 \* .
  + El grafo contiene dos nodos especiales, inicio y fin de camino, los cuales no aportan información a la cadena generada por la hormiga, solamente son el origen y el destino de la hormiga.
* Cada hormiga realiza un recorrido partiendo desde el nodo de inicio y llegando al nodo de fin.
  + La hormiga que se encuentra en un nodo de un nivel para moverse a su correspondiente nodo de un nivel , chequea su deseo de no tomar el camino con cierta cantidad de feromonas, si su deseo de no tomar el camino es menor que las feromonas del mismo, toma dicho camino; si es mayor, genera un nuevo deseo y chequea con el siguiente camino, hasta encontrar un camino que su deseo de no tomarlo sea menor que la cantidad de feromonas en el mismo.
  + En cada nodo, la hormiga adiciona a la cadena que va construyendo, la letra correspondiente a dicho nodo.
* Al finalizar las hormigas de realizar sus recorridos, son adicionadas al camino una cantidad de feromona igual a la evaluación de la cadena construida por la hormiga .
* Luego de adicionar todas las feromonas a los caminos, se realiza la evaporación.
  + Las feromonas se evaporan siguiendo una normalización de las feromonas, donde los valores de feromonas regresaran al rango 0.25 – 0.75 \* .

## Algoritmo Genético Paralelo (AGP)

Para atacar el problema del FFMSP se implementó el AGP. AGP plantea que se deben mantener islas separadas donde diferentes poblaciones atacan al mismo problema. Luego de un tiempo llamado temporada, los individuos de las islas emigran hacia las islas vecinas refrescando de esta manera el material genético en las mismas.

Teniendo en cuenta el modelo a dos islas: una de las islas fue implementada usando AG y la otra isla fue implementada usando ACO.

Las características de la implementación se describen a continuación:

* Las temporadas en cada isla tiene una duración de 1 minuto.
* La migración se realiza primeramente filtrando todas las soluciones de ambas islas de forma tal que las soluciones repetidas no se tengan en consideración para la migración.
* La isla AG recibe el material genético desde ACO y genera nuevos individuos adicionándolos a la población
* La isla ACO recibe el material genético desde AG y planta las feromonas en el grafo de posibles caminos.
* El algoritmo termina a los 10 minutos de vida.

# Software

El AGP fue implementado en un software usando el lenguaje Java con la máquina virtual en su versión 1.8.0\_161. Se usó como IDE de desarrollo Eclipse en su versión Oxigen.2. Para realizar las iteraciones se usó la biblioteca concurrent que permite ejecutar tareas de forma multicore; cada iteración del problema se ejecuta de forma concurrente.

Para los experimentos se usó una PC (Laptop) DELL Inspiron 15, microprocesador Intel Core i3 a 1.70GHz y 4GB de RAM, y Windows 10 como sistema operativo.

Se realizó una interfaz de consola. Para ejecutar el programa se debe abrir una consola, ir hasta la carpeta donde se encuentra el ejecutable “FFMSP.jar” y ejecutar el siguiente comando: “java -jar FFMSP.jar”. En consola aparecerán otras indicaciones para la ejecución. Una vez terminada cada iteración, se mostrará en consola el tiempo total de la ejecución del programa en milisegundos. El fichero de salida se generará en el mismo lugar donde se encuentra el fichero de entrada con el prefijo “solution\_” + nombre del fichero de entrada.

El código fuente se encuentra en la siguiente dirección <https://github.com/ochavezhdez/QAP>.

## Clase MainFFMSP

Construye la interfaz de consola del software.

## Clase Iterator

Es la clase encargada de realizar las iteraciones del problema. Esta clase implementa una unidad de trabajo concurrente. En el método compute realiza todas las iteraciones de los objetos de tipo ecosystem. Realiza los cálculos del promedio del tiempo de cálculo, el promedio de las evaluaciones, la desviación estándar del tiempo y de las evaluaciones.

## Clase Ecosystem

Es la clase encargada de ejecutar todas las islas de nuestra ejecución. Controla las migraciones entre las mismas. Brinda información sobre el tiempo que demoró en dar respuesta.

## Clase Island

Es una clase abstracta, de la cual se crean las islas que contendrá la implementación del AG y de ACO. Controla los individuos que actuaran en la solución.

## Clase Build

Es una clase de tipo isla que controla los individuos que actuaran en el AG. Construye la población inicial, adiciona los individuos que emigran a esta isla, selecciona los padres candidatos a cruzar y selecciona los individuos a la siguiente generación.

## Clase Nest

Es una clase de tipo isla que controla los individuos que actuaran en el ACO. Construye y controla el grafo de posibles caminos, construye la población inicial, los individuos que emigran a esta isla, orienta a cada hormiga a realizar su camino, actualiza las soluciones encontradas por las hormigas, actualiza las feromonas en los caminos del grafo y evapora las feromonas en los caminos del grafo.

## Clase Individual

Es una clase abstracta de la cual se crean los individuos que actuaran en las islas. Contiene una solución o cromosoma, la evaluación de la solución y las secuencias base o de comprobación. La evaluación la calcula utilizando la HD.

## Clase Person

Es una clase de tipo individual, es la encargada de realizar las actividades controladas por la clase build en la ejecución de una iteración AG. Construye las propuestas de soluciones, realiza el cruzamiento de genético, así como la mutación de los nuevos individuos.

## Clase Ant

Es una clase de tipo individual, es la encargada de realizar las actividades controladas por la clase nest en la ejecución de una iteración ACO. Construye las propuestas de soluciones ha partir de los posibles caminos, adiciona las nuevas feromonas importadas desde la isla AG.

## Clase Route

Es la encargada de gestionar las posibles rutas que tomarán cada una de las ant en su búsqueda de las mejores soluciones por los diferentes caminos. Evapora las feromonas ha partir de la normalización de la cantidad de feromonas.

## Clase Solution

Es la encargada de almacenar las cadenas mejores punteadas en cada iteración, así como su evaluación.

# Resultados y discusión

Se ejecutaron 30 iteraciones del AGP propuesto, sobre las bases de datos propuestas, dando los resultados que se muestran a continuación.

Para la base de datos de cadenas basadas en genoma:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n | m | Promedio | | | Desviación Estándar de la evaluación |
| Evaluación | % | Cantidad |
| 100 | 300 | 226.6 | 76 | 2.1 | 1.87 |
| 600 | 450.53 | 75 | 1.9 | 1.93 |
| 800 | 600.17 | 75 | 1.9 | 2.35 |
| 200 | 300 | 220.23 | 73 | 2.5 | 1.17 |
| 600 | 443.63 | 74 | 2.8 | 1.99 |
| 800 | 591.63 | 74 | 3.1 | 1.67 |

Para la base de datos de cadenas aleatorias:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n | m | Promedio | | | Desviación Estándar de la evaluación |
| Evaluación | % | Cantidad |
| 100 | 300 | 223.90 | 75 | 1.3 | 0.99 |
| 600 | 447.83 | 75 | 1.6 | 1.31 |
| 800 | 596.47 | 75 | 1.5 | 1.98 |
| 200 | 300 | 218.33 | 73 | 2.4 | 0.76 |
| 600 | 440.4 | 73 | 3.1 | 1.04 |
| 800 | 589.1 | 74 | 2.8 | 1.58 |
| 300 | 300 | 216.4 | 72 | 3.7 | 0.77 |
| 600 | 437.4 | 73 | 3.7 | 1.07 |

Donde n es el numero de secuencias, m la longitud de las secuencias, la evaluación promedio es el promedio de las 30 iteraciones; el % es el % promedio de las evaluaciones; la cantidad promedio denota el promedio de cadenas diferentes generadas por cada iteración; y la desviación estándar de la evaluación refiere a la desviación estándar de las evaluaciones de cada iteración.

Como se observa a una evaluación más discreta, corresponden mayor cantidad de cadenas diferentes generadas.

## Propuestas de mejoras

En experimentos a priori, se comprobó que una mutación de más de un lugar consecutivo en los genes de los individuos reportaba soluciones más discretas, por lo que ACO no aporta buenos genes a la población general. Recomiendo realizar los experimentos con islas AG.