UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA:
INGENIERÍA DE SISTEMAS

Trabajo de titulación previo a la obtención del título de:
Ingeniero de Sistemas

TEMA:

DESARROLLO DE UN ALGORITMO HEURÍSTICO, CON PROGRAMACIÓN MULTIAGENTES, PARA LA GENERACIÓN DE LA MATRIZ DE RUTA DEL SISTEMA LINEAL Y=AX UTILIZADO EN LA ESTIMACIÓN DE LA MATRIZ ORIGEN-DESTINO

AUTOR:
LUIS ALBERTO FLORES FLORES

TUTORA:
LINA PATRICIA ZAPATA MOLINA

Quito, agosto del 2020

CESIÓN DE DERECHOS DE AUTOR

Yo, Luis Alberto Flores Flores, con documento de identificación Nº 1717792319, manifiesto mi voluntad y cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del trabajo de titulación intitulado: "**DESARROLLO DE UN ALGORITMO HEURÍSTICO, CON PROGRAMACIÓN MULTIAGENTES, PARA LA GENERACIÓN DE LA MATRIZ DE RUTA DEL SISTEMA LINEAL Y=AX UTILIZADO EN LA ESTIMACIÓN DE LA MATRIZ ORIGEN-DESTINO"**, mismo que ha sido desarrollado para optar por el título de: INGENIERO DE SISTEMAS, en la Universidad Politécnica Salesiana, que dando la Universidad faculta da para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en mi condición de autor me reservo los derechos morales de la obra antes citada. En concordancia, suscribo este documento en el momento que hago entrega del trabajo final en digital a la Biblioteca de la Universidad Politécnica Salesiana.

……..

Luis Alberto Flores Flores

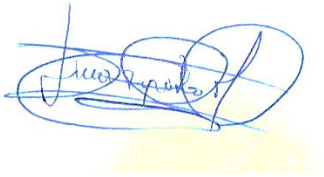
Cédula: 1717792319

Fecha: Quito, agosto 2020

DECLARATORIA DE COAUTORÍA DEL DOCENTE TUTOR

Yo, declaro que bajo mi dirección y asesoría fue desarrollado el Artículo académico, con el tema: DESARROLLO DE UN ALGORITMO HEURÍSTICO, CON PROGRAMACIÓN MULTIAGENTES, PARA LA GENERACIÓN DE LA MATRIZ DE RUTA DEL SISTEMA LINEAL Y=AX UTILIZADO EN LA ESTIMACIÓN DE LA MATRIZ ORIGEN-DESTINO, realizado por Luis Alberto Flores Flores, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana, para ser considerados como trabajo final de titulación.

Quito, agosto 2020



……..

Lina Patricia Zapata Molina

C.I: 0501877278

DEDICATORIA

En primer lugar, a Dios por darme las fuerzas para seguir adelante y además de ser la guía en cada uno de mis pasos, A mis padres que cada día me alentaban y no me dejaban solo y gracias a su gran apoyo me han convertido en una gran persona con grandes valores y principios los amo mucho. A mis dos angelitos mis padres que desde el cielo sé que siempre me ven y me guían cada día les digo lo logre y es para ustedes mi carrera. A mi familia por el apoyo recibido durante toda mi carrera universitaria y a lo largo de mi vida. A la Ing. Lina Zapata que gracias a sus consejos y correcciones hoy puedo culminar este trabajo, además me enseño que no hay limitantes para cumplir las metas y sueños que se proponen en la vida. A mi prima Heydi Salazar que con su apoyo incondicional siempre me supo alentar y dar el empujón diario para culminar esta meta y ha sido como una hermana te quiero mucho.

.

Luis Alberto Flores Flores

AGRADECIMIENTO

A la Universidad Politécnica Salesiana por haberme inculcado buenos valores y sobre todo una buena ética profesional, a los Ingenieros que cada de mi carrera compartieron sus conocimientos y gracias a ellos he aprendido mucho. A mi madre por ser siempre mi mano derecha mi soporte en las buenas y malas que siempre con su sonrisa su amor fueron mi gran ayuda a nunca decaer, a mi padre por su apoyo, cariño y sobre todo sus consejos que son, para mí de gran sabiduría, finalmente a mis amigos que durante toda la carrera siempre fueron de gran apoyo y de grandes consejos.

Luis Alberto Flores Flores

DESARROLLO DE UN ALGORITMO HEURÍSTICO, CON PROGRAMACIÓN MULTIAGENTES, PARA LA GENERACIÓN DE LA MATRIZ DE RUTA DEL SISTEMA LINEAL Y=AX UTILIZADO EN LA ESTIMACIÓN DE LA MATRIZ ORIGEN-DESTINO

Luis Alberto Flores Flores [1], Lina Patricia Zapata Molina[2]

Resumen

El actual documento ilustra el diseño y la elaboración de un algoritmo heurístico implementado en dos diferentes escenarios: el primero orientado a objetos y el segundo con programación multiagente. El algoritmo realiza el cálculo y generación de la matriz de ruta A considerada en el modelo, $Y = AX$ donde A representa la matriz de ruta. Y el número de pasajeros que ingresan-salen de la parada o el número de pasajeros que desciende del bus entre los pares origen-destino y, X es el número de pasajeros estimado que viajan desde una estación origen hacia una estación destino. Una vez elaborado los dos algoritmos se realizan pruebas de rendimiento para saber cuál es más eficiente en cuanto a tiempo de ejecución.

Palabras Clave: Programación orientada a objetos, Sistemas multiagentes, Jade, FIPA - ACL

Abstract

The current document illustrates the design and elaboration of a heuristic algorithm implemented in two different scenarios: the first object-oriented and the second with multi-agent programming. The algorithm performs the calculation and generation of the route matrix A considered in the model $Y = AX$, where A represents the route matrix, Y the number of passengers entering-leaving the stop or the number of passengers getting off the bus between origin-destination pairs, and X is the estimated number of passengers traveling from an origin station to a destination station. Once the two algorithms have been developed, performance tests are carried out to find out which is more efficient in terms of execution time.

Keywords: Object Oriented Programming, Multi-Agent Systems, Jade, FIPA - ACL

[1] Estudiante de Ingeniería de Sistemas, Universidad Politécnica Salesiana, sede Quito, Ecuador. Autor para correspondencia: lfloresf@est.ups.edu.ec

[2] Docente de la carrera de Ingeniería de Sistemas, Universidad Politécnica Salesiana, sede Quito, Ecuador. Autor para correspondencia: lzapata@ups.edu.ec

1. Introducción.

En áreas como la red de transporte público, existe un gran interés en la estimación de la matriz origen-destino (MOD), la misma que suministra información sobre el actual (vigente) flujo de pasajeros entre todos los posibles pares de nodos origen y destino [1]. El uso de esta matriz de tráfico, demanda de mecanismos de observación o medición directa del tráfico origen-destino (OD) y, que generalmente no son realizables por los costos en que incurren estos mecanismos. Es habitual superar este tipo de inconvenientes recurriendo a las estimaciones indirectas de los elementos de la matriz de tráfico a través de mediciones del tráfico sobre enlaces.

Según Rahman [2] las matrices de tráfico son inferidas desde conexiones medidas a través de la estimación utilizando, diferentes técnicas y en su trabajo llegó a determinar, que el método basado en inferencia estadística (conocido como técnicas de tomografía de red) que utiliza modelos probabilísticos para tráfico OD apuntan a estimar parámetros adecuados mediante, el método de máxima verosimilitud de Vardi [3] o por métodos Bayesianos [1] [4] [5].

Vardi [3] plantea el algoritmo para la estimación de MOD partiendo del modelo en base al conocimiento previo del flujo de enlaces directos en la red representado, por $Y = AX$ siendo Y el flujo de los enlaces directos, X la matriz OD y A la matriz de rutas. Para resolver el sistema diseña el criterio de máxima verosimilitud mediante, el Método de los Momentos. Por su parte, Tebaldi y West [1] resuelven el problema de la estimación de matrices OD a partir, del modelo propuesto por Vardi mediante la inferencia Bayesiana con el método de Monte Carlo en cadenas de Markov.

La restrictiva que se exhiben en estos algoritmos al momento de construir el sistema de ecuación matricial $Y = \mathrm{AX}$ es la representación de los datos de redes de transportes grandes donde la matriz de ruta A llega a ser de gran dimensión pretender, llenarla en forma manual solicita de un arduo trabajo, tiempo y esfuerzo. En la actualidad no existe un algoritmo que facilite la generación automática de la matriz A en función del tamaño de la red de transporte, lo que dificulta realizar trabajos de estimación de flujos de pasajeros o vehículos con redes grandes.

El presente trabajo tiene como objetivo el diseñar, y desarrollar, una solución algorítmica que genere la matriz de ruta A para redes de transportes o de comunicación de n tamaño. Inicialmente el algoritmo se implementó con programación concurrente y secuencial orientada a objetos, pero los tiempos de respuesta fueron poco eficientes. Para una mejor solución se elaboró el algoritmo con sistemas multiagentes sobre la plataforma JADE (Java Agent Development Framework) [6]. En consideración de que un agente, es un programa autónomo con una identidad y un canal de comunicación, para que los agentes puedan intercambiar mensajes entre sí. Esto facilitó la creación del algoritmo multitarea con tres agentes cada uno con un comportamiento diferente, pero con un mismo fin contribuir a la generación de la matriz de ruta A.

El primer agente, se encarga de asignar los valores a la matriz A tomando en cuenta información referente a la cantidad de pasajeros existentes en la estación origen y la estación destino (enlaces) y los pares OD, el segundo agente genera y asigna valores a la matriz

A, a partir del número de pasajeros que descienden en las paradas o estaciones y los pares OD, el tercer agente es el supervisor el cuál coordina la integración de los valores emitidos por los otros agentes en una sola matriz A resultante.

El algoritmo con sistemas multiagentes reduce el tiempo de ejecución en un 17% aproximadamente. agentes en una sola matriz A resultante.

2. Materiales y Métodos.

2.1 Matriz Origen-Destino

Existen varios mecanismos de la estimación de MOD estática. En el presente trabajo empleamos el modelo propuesto por Tebaldi y West [1] para la estimación de la matriz Origen-Destino.

Hazelton [5] menciona en su trabajo que el número de pares Origen-Destino al ser mayor que el número de enlaces directos (nodo origen–nodo destino sin estaciones o nodos intermedios) se obtendría un sistema lineal indeterminado (múltiples soluciones) y que esto mejoraría si se añade más información medida. Para nuestro caso puntual se consideró la incorporación de datos referente a la cantidad de pasajeros que descienden de los buses en las distintas paradas o estaciones, en periodo de tiempo determinado [7].

En la Figura 1 se muestra el esquema de una pequeña red de transporte hipotética. Donde, las flechas representan el flujo de pasajeros entre nodos, y los círculos o nodos son las estaciones o paradas de pasajeros. Las variables consideradas en el esquema son:

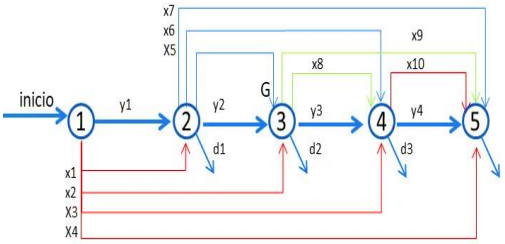


Figura 1. Esquema de una Red de transporte público hipotética.

$x_j,\ j = 1,..,10$ cantidad de pasajeros que viajen desde el nodo origen hacia el nodo destino (pares OD) [7].

$y_i,\ i = 1,\dots,4$ flujos de los pasajeros que viajen desde el nodo origen hacia el nodo destino sin nodos o paradas intermedias (enlaces directos).

d_1, d_2, d_3 estimaciones de pasajeros que bajan o descienden del bus en las paradas o estaciones 1, 2, 3 ... [7].

La relación entre los pares origen-destino y los enlaces directos se representa como un sistema lineal (matricial) de la forma:

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} 1111000000 \\ 0111111000 \\ 0011011110 \\ 0001001011 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \end{pmatrix} \quad (1)$$

La relación entre el número de pasajeros que descienden a las paradas (destinos) y pares origen-destino es:

$$\begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{pmatrix} = \begin{pmatrix} 1000000000 \\ 0100100000 \\ 0010010100 \\ 0001001011 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \end{pmatrix} \quad (2)$$

Al observar (1) y (2) tenemos que las terceras filas son iguales, en otras palabras los dos sistemas son linealmente dependientes. Generalizando tenemos una red $G\ (N, M)$ donde N representa el sistema línea (1) y M el sistema lineal (2) [7]. En forma de ecuaciones los dos sistemas pueden escribirse como:

$$\mathrm{Y_{mx1}} = \mathrm{A_{mxn}X_{nx1}} \quad (3)$$

$$D_{k\mathbf{x1}} = B_{kxn}X_{nx1} \quad (4)$$

Dónde $X = (x_1, x_2, \ldots, x_n)^T$ representa el vector cuyos elementos es la n cantidad de pasajeros que viajen desde el nodo origen hacia el nodo destino (n pares OD). $Y = (y_1, \ldots, y_m)^T$ es el vector cuyos elementos son los m enlaces directos y finalmente la matriz de ruta $A = (a_{ij})$ $para$ i $= 1, \ldots, m$ y j $= 1, \ldots, n$ cuyos elementos son valores entre 0 y 1, el 1 representa que existe un enlace entre un nodo origen y nodo destino y 0 [7].

$D = (d_1, \ldots, d_k)^T$ representa el vector que contiene $k = m - 1$ elementos cuyo valor es el número de pasajeros de descienden a cada parada o estación destinos menos el último nodo de la red [7]. La matriz $B = (b_{ij})$, donde i $= 1, \ldots,$ k y j $= 1, \ldots,$ n, contiene elementos cuyo valor esta entre 0 y 1, cuando existe relación entre el número de pasajeros que descienden a las paradas y el par origen-destino se asigna el valor de 1 en B, caso contrario se asigna el valor de 0 [7].

Al unir las ecuaciones (3) y (4) se forma un solo sistema línea que se representa de la siguiente manera.

$$\binom{Y}{D}_{(2m-1)\mathbf{x1}} = \binom{A}{B}_{(2m-1)xn} X_{nx1} \quad (5)$$

En forma resumida tenemos:

$$Y_D = A_B\mathrm{X} \quad (6)$$

Donde la ecuación (6) representa el modelo para la estimación de la MOD.

Sea:

n: $número\ de\ nodos\ o\ estaciones$

m: $número\ de\ pares\ OD, m = n(n-1)$

Para el ejemplo de la Figura 1, $n = 5, d = 4$

$m = 20$ enlaces o pares origen destino (OD).

Por su parte, la matriz de rutas A, queda definida por 7 filas (enlaces + destinos) y por 10 columnas (se considera el recorrido en un solo sentido nodo 1 hacia nodo 5 y no en sentido contrario) según (6).

Expresando matricialmente, tenemos.

$Y = (y1, y2, \ldots., y7)$

$X = (x1, x2, \ldots., x10)$

$A = (a11, a12, \ldots\ldots, a10$
$a71, a72, \ldots, a710$
$\ldots).$

Siendo así $Y = AX$ un sistema de ecuaciones lineales con 7 ecuaciones y 10 incógnitas.

En la Tabla 1 se detalle cada par, OD y, en la Figura 2 se observa el sistema lineal $Y = AX$ con valores asignados.

Tabla 1. Pares origen-destino, ***X***.

Origen	Destino	2	3	4	5
1		x_1	x_2	x_3	x_4
2			x_5	x_6	x_7
3				x_8	x_9
4					x_{10}

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{pmatrix} 1111000000 \\ 0111111000 \\ 0011011110 \\ 0001001011 \\ 1000000000 \\ 0100100000 \\ 0010010010 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \end{pmatrix}$$

Figura 2. Valores asignados al sistema matricial $\boldsymbol{Y = AX}$.

2.2. Agentes y la programación orientada a agentes

2.2.1. Agente

Un agente personifica una abstracción de software un pensamiento o noción pareja a los métodos de la programación orientada a objetos. La noción de un agente, representa una compleja entidad de software que es apto de ejercer, con un cierto grado de autonomía con el fin de efectuar, trabajos en representación de individuos. Los objetos se detallan mediante métodos y atributos en cambio un agente de software es explicado por sus comportamientos [8].

2.2.2. Sistemas Multiagente

Se denomina un sistema multiagente (MAS), al sistema que consta de un grupo de agentes, que se relacionan entre sí, y al sub-campo de los principios y diseños se denomina AI distribuido [9].

2.2.3. Plataforma de desarrollo de multiagentes Jade

JAVA consta con el middleware JADE y es una herramienta, en el cual se desarrollan los sistemas multiagentes y, para la comunicación usa estándares FIPA-ACL, que son para el envío de mensajes entre agentes y está a su vez es asíncrona [10].

En la Figura 3 se ilustra la plataforma de JADE.

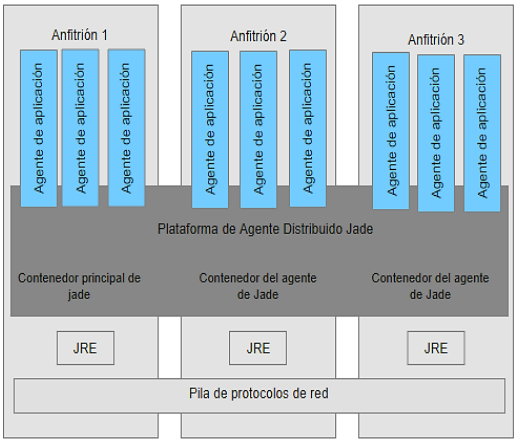


Figura 3. Plataforma de Jade [6].

Las clases en agentes se extienden de agent, y tienen el método denominado setup, para iniciar el agente. Para inicializar los comportamientos que tienen los agentes se usa action, que determinan las tareas a realizar por el agente [10].

2.3 Desarrollo de Algoritmos

2.3.1. Algoritmo Orientado a Objetos

El algoritmo secuencial orientado a objetos posee de dos clases: MetodosMatriz y Principal; la primera clase tiene dos métodos: la Figura 4 ilustra el primer método, para generar la parte de la matriz *A* asignando el valor de 1 (en forma de triángulos) si existe enlaces entre los pares OD de lo contrario se asigna 0 (por defecto).

Método Matriz triangular que genera la matriz *A* con el valor de 1 cuando existe enlaces entre el nodo origen y nodo destino.

```
1.  método MATRIZTRIANGULAR (argumentos valor, nodos) hacer
2.  variables: valor, acumulador, r
3.   para recorrer filas hasta nodos menos uno
4.    para recorrer las columnas desde cero hasta el valor
5.      matriz [ fila] [ columna] = 1
6.      si sentido es igual a dos
7.        matriz [ fila] [ columna] = 1
8.      fin si
9.    fin para
10.   valor = nodos menos uno
11.   incrementar r en uno
12.  fin para
13.  valor = filas en uno
14.  acumulador = parte desde 0 hasta el valor de nodos-1
```

Figura 4. Pseudocódigo método matriztriangular.

El segundo método, ver Figura 5 representa el método Matriz Diagonal cuya función es completar la generación de la matriz *A* asignando, el valor de 1 si existen destinos (Pasajeros que descienden a la parada o estación) en relación a los pares OD.

Método Matriz Diagonal que genera la matriz *A* con el valor de 1 al valor de los destinos en relación a los pares OD.

```
1.  Método MATRIZDIAGONAL (argumentos valorn, nodosd, nodos) hacer
2.   variables: valor: almacena los nodos
3.   af = nodos menos uno
4.  para recorrer la matriz hasta los nodos menos uno
5.   i = el valor de af
6.   para recorrer la matriz hasta los nodos menos uno
7.     matriz [ fila] [ columna] = 1
8.     si sentido es igual a dos
9.       matriz [ fila] [ columna] = 1
10.    fin si
11.      incrementa en uno los nodos
12.    fin para
13.      incrementa af en uno hasta nodos-1
14.      incrementa el ac hasta nodos-1
15.      disminuye el valorn hasta nodos-1
16.  fin para
```

Figura 5. Pseudocódigo método matrizdiagonal.

La clase Principal se muestra en la Figura 6 y, se encarga de la creación del objeto instanciando en la clase, y a través de este objeto realizar, la llamada a los dos métodos secundarios enviando como argumentos: el número de nodos, filas y columnas para generar la matriz de Ruta *A*.

Clase principal es la ejecutora de los métodos para la obtención de la Matriz.

```
1.  variables: nodo, sentido, matriz
2.   ingreso de nodos
3.   ingreso de sentido
4.    si sentido es uno
5.      cálculo de número de filas y columnas
6.    // fin si
7.      si sentido es dos
8.      cálculo de número de filas y columnas
9.     // fin si
10.   Matriz inicial [ filas] [ columnas]
11.  Instanciar la clase de MetodosMatriz
12.  Llamada al metodo MatrizTriangular
13.  Llamada al metodo MatrizDiagonal
14.  Llamada al metodo Impresión
```

Figura 6. Pseudocódigo clase principal.

2.3.2. Algoritmo multiagentes

2.3.2.1. Construcción de Agentes

El número de agentes requeridos para la generación de la matriz de ruta son tres. Los agentes se muestran en la Figura 7, pero cada uno, posee un comportamiento, o tareas diferentes. Los multiagentes generan multitareas lo que permitió que cada agente trabaje en la generación de una parte de la matriz *A* de forma simultánea. Para tener una idea, se muestra el proceso para realizar un agente.

- Crear una clase que representar al agente, y heredar de la clase jade.core.Agent, con el metodo setup() el cual inicializa al agente.
- Añadir un comportamiento que especifica las tareas que realiza un agente, para cumplir con sus

objetivos, y con el método action () que ejecuta las tareas.

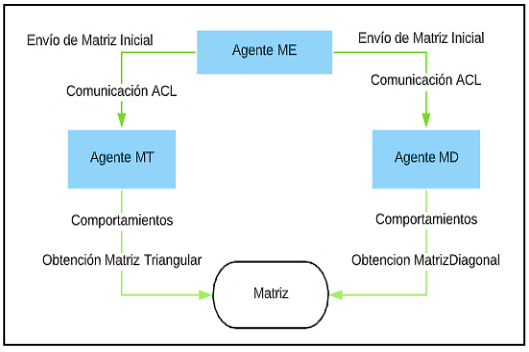


Figura 7. Estructura del algoritmo multiagente.

2.3.2.2. Comunicación de los agentes

Los agentes para comunicarse utilizan el mecanismo de comunicación ACL de FIPA. La Figura 8 muestra el diagrama de secuencia del programa en donde se detallan los tres agentes que tiene el algoritmo, además de la forma de comunicación entre agentes que es ACL, y los comportamientos que van a tener los agentes para la generación de la matriz *A*.

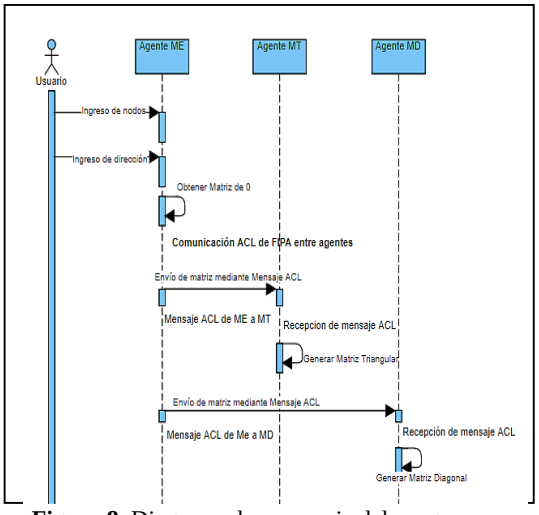


Figura 8. Diagrama de secuencia del programa.

En base al Diagrama de secuencia se procede a describir, el proceso de comunicación que existe entre los agentes, donde el agente Matriz Emisor, que se indica en la Figura 9 es el agente considerado Master, por ser, responsable de la gestión del trabajo realizado por los otros agentes y es el encargado de iniciar y terminar la comunicación entre los agentes.

Agente Matriz Emisor. - tiene la función de gestionar la creación y envío de la matriz inicial de 0 hacia los agentes MT y MD.

```
1.  clase Agente Emisor Agent
2.  iniciar agente
3.  añadir un comportamiento
4.  variables nodos, nodD, sentido, matriz [] []
5.  comienzo de la tarea a realizar
6.  crear ID`s para identificar los mensajes a MT, MD
7.    ingreso de nodos
8.    ingreso de dirección
9.     si sentido es uno
10.       calculo de numero de filas y columnas
11.    fin si
12.    si sentido es dos
13.       calculo de numero de filas y columnas
14.    fin si
15.    Matriz inicial [filas][columnas]
16.   instanciar la clase MetodosMatriz
17.   ---envío de nodos a matriz triangular---
18.    estándar fipa de comunicación
19.    id para enviar un mensaje
20.    Id de quien recibe el mensaje
21.    contenido del mensaje nodos
22.    envío del mensaje-> nodos
23.   --envío de objeto matriz a matriz triangular---
24.    estándar fipa de comunicación
25.    id de la clase que envía el mensaje
26.    Id del agente que recibe
27.    contenido del objeto matriz
28.    envío del mensaje-> matriz objeto
29.   --envío de nodos y nodD a matriz diagonal---
30.    estándar fipa de comunicación
31.    id para enviar un mensaje
32.    Id de quien recibe el mensaje
33.    contenido del mensaje nodos, nodD
34.    envío de mensaje-> nodos, nodD
35.   --envío de objeto matiz a matriz triangular---
36.    estándar fipa de comunicación
37.    id de la clase que envía el mensaje
38.    Id del agente que recibe
39.    contenido del objeto matriz
40.    envío de mensaje-> matriz objeto
41.   //fin de la tarea
42.   //fin del comportamiento
43.  //fin del agente
44.  //fin clase
```

Figura 9. Pseudocódigo agente matriz emisor.

3. Resultados y Discusión.

Las pruebas de rendimiento son elaboradas para poner a prueba el rendimiento de un software en tiempo de ejecución, dentro del entorno de un sistema integrado [11].

Los dos algoritmos se pusieron a prueba para saber, cual tiene mejor, rendimiento en tiempo de ejecución para lo cual se realizó un promedio de todos los tiempos, por cada algoritmo, y para ello se utilizó un timer al inicio de cada método para obtener el tiempo de total de ejecución del programa medido en segundos.

Se realizaron 10 pruebas con igual número de nodos (60, 90, 120, 150, 180, 210) a los dos algoritmos, y sus resultados se indican en las siguientes tablas: la Tabla 2 muestra en resumen los datos obtenidos del algoritmo con objetos. De forma continua en la Tabla 3 se representan los valores de las pruebas realizadas al algoritmo con el sistema multiagentes. En la Tabla 4 se muestran los promedios obtenidos, por cada prueba, y se observa que el algoritmo con multiagentes es el más eficiente en tiempo de ejecución.

En la Figura 10 se observa la comparación entre los dos algoritmos el algoritmo con multiagentes es mucho más rápido en tiempo de ejecución para llegar a la obtención de la matriz A.

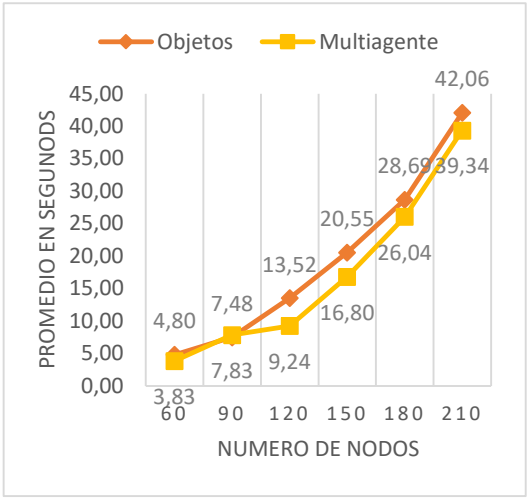


Figura 10. Promedios entre tiempos de algoritmos.

En la Figura 11 se muestra el porcentaje de cada algoritmo y se evidencia que el promedio en tiempo de ejecución del algoritmo con multiagentes es menor para llevar a cabo la tarea.

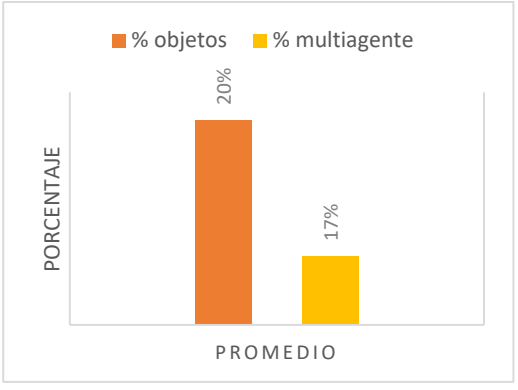


Figura 11. Porcentajes de tiempo entre algoritmos.

Tabla 2. Tiempos de ejecución de cada prueba

Algoritmo con objetos						
Pruebas/tiempo(seg)	60 nodos	90 nodos	120 nodos	150 nodos	180 nodos	210 nodos
Prueba 1	5,38	7,16	13,22	17,26	29,39	42,95
Prueba 2	4,53	7,07	12,32	24,50	29,30	41,61
Prueba 3	4,73	7,20	16,85	21,80	29,13	42,49
Prueba 4	4,97	6,25	13,36	19,59	29,13	43,12
Prueba 5	4,85	6,65	13,59	20,48	28,17	42,75
Prueba 6	5,16	7,66	13,41	20,87	28,25	42,75
Prueba 7	4,49	7,74	13,41	20,29	28,30	42,13
Prueba 8	4,99	8,13	12,56	19,59	28,89	41,64
Prueba 9	4,71	8,25	13,93	20,98	27,46	40,10
Prueba 10	4,17	8,66	12,57	20,12	28,90	41,06
Total	47,97	74,76	135,22	205,47	286,91	420,60
Promedio	4,80	7,48	13,52	20,55	28,69	42,06

Tabla 3. Tiempos de ejecución de cada prueba

Algoritmo con sistema multiagentes						
Pruebas/tiempo(seg)	60 nodos	90 nodos	120 nodos	150 nodos	180 nodos	210 nodos
Prueba 1	4,12	7,94	10,85	17,63	26,98	39,02
Prueba 2	4,02	7,05	8,99	19,12	25,76	37,83
Prueba 3	3,52	8,20	8,93	16,28	24,70	38,17
Prueba 4	3,87	7,11	9,58	15,12	24,27	39,15
Prueba 5	3,65	8,18	9,46	15,17	31,22	38,12
Prueba 6	3,73	7,97	9,49	16,31	24,98	40,22
Prueba 7	3,87	8,26	9,01	15,85	24,39	40,77
Prueba 8	3,78	8,18	8,77	16,79	26,83	39,58
Prueba 9	3,92	7,56	8,55	17,73	25,88	40,46
Prueba 10	3,84	7,82	8,78	18,01	25,42	40,08
Total	38,32	78,28	92,41	168,01	260,41	393,37
Promedio	3,83	7,83	9,24	16,80	26,04	39,34

Tabla 4. Promedios del tiempo de ejecución de cada algoritmo

Promedios y porcentajes de los algoritmos				
Nodos/Promedio(seg)	Objetos	% Objetos	Multiagentes	% Multiagentes
60	4,80	4%	3,83	4%
90	7,48	6%	7,83	8%
120	13,52	12%	9,24	9%
150	20,55	18%	16,80	16%
180	28,69	25%	26,04	25%
210	42,06	36%	39,34	38%
Total	117,10	100%	103,08	100%
Promedio	19,52	20%	17,18	17%

4. Conclusiones.

En base a las pruebas realizadas con cada algoritmo, y con los resultados obtenidos, se puede evidenciar que la programación con multiagentes es más eficiente en tiempo de ejecución con un 17% ya que trabajan varios agentes al mismo tiempo y esto marca una gran diferencia en el tiempo de respuesta para obtener una solución.

En base a los tiempos de ejecución mostrados anteriormente, se evidencia el gran potencial que se tiene al usar la programación multiagente, para realizar una tarea y obtener una solución, tal es que en nuestro caso el buen diseño del algoritmo heurístico más la implementación de multiagentes se logra obtener la matriz de ruta *A*.

Referencias

[1] C. Tebaldi and M. West, "Bayesian Inference on Network Traffic Using Link Count Data," *J. Am. Stat. Assoc.*, vol. 93, no. 442, p. 557, Jun. 1998, doi: 10.2307/2670105.

[2] M. M. Rahman, S. Saha, U. Chengan, and A. S. Alfa, "IP traffic matrix estimation methods: Comparisons and improvements," in *IEEE International Conference on Communications*, 2006, vol. 1, pp. 90–96, doi: 10.1109/ICC.2006.254710.

[3] Y. Vardi, "Network tomography: Estimating source-destination traffic intensities from link data," *J. Am. Stat. Assoc.*, vol. 91, no. 433, pp. 365–377, Mar. 1996, doi: 10.1080/01621459.1996.10476697.

[4] E. Castillo, J. M. Menéndez, and S. Sánchez-Cambronero, "Traffic Estimation and Optimal Counting Location Without Path Enumeration Using Bayesian Networks," *Comput. Civ. Infrastruct. Eng.*, vol. 23, no. 3, pp. 189–207, Apr. 2008, doi: 10.1111/j.1467-8667.2008.00526.x.

[5] M. L. Hazelton, "Statistical Inference for Transit System Origin-Destination Matrices," *Technometrics*, vol. 52, no. 2, pp. 221–230, May 2010, doi: 10.1198/TECH.2010.09021.

[6] G. (University of P. Bellifemine, Fabio; Caire, Giovanni; Trucco, Tiziana (TILAB, formerly CSELT); Rimassa, "JADE PROGRAMMER'S GUIDE," 2000. https://jade.tilab.com/doc/programmersguide.pdf (accessed Jun. 17, 2020).

[7] L. P. Zapata, M. Flores, V. Larios, R. Maciel, and E. A. Antunez, "Estimation of people flow in public transportation network through the origin-destination problem for the South-Eastern corridor of Quito city in the smart cities context," in *5th IEEE International Smart Cities Conference, ISC2 2019*, Oct. 2019, pp. 181–186, doi: 10.1109/ISC246665.2019.9071778.

[8] H. S. Nwana, "Software agents: an overview," *Knowl. Eng. Rev.*, vol. 11, no. 3, pp. 205–244, Sep. 1996, doi: 10.1017/s026988890000789x.

[9] N. Vlassis, "A Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence," *Synth. Lect. Artif. Intell. Mach. Learn.*, vol. 1, no. 1, pp. 1–71, Jan. 2007, doi: 10.2200/s00091ed1v01y200705aim002.

[10] F. Bellifemine, F. Bergenti, G. Caire, and A. Poggi, "Jade — A Java Agent Development Framework," 2005, pp. 125–147.

[11] R. S. Pressman, *Ingeniería del software : un enfoque práctico*. 2010.