Universidad Autónoma Metropolitana Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Clasificación de servicios web mediante el algoritmo Colonia Artificial de Abejas

Proyecto Tecnológico
Trimestre 2016 Invierno

Flores Sánchez Iván
Matrícula: 206241486
al206241486@azc.uam.mx

Asesores:
Dra. Maricela Claudia Bravo Contreras
Asociado D
Departamento de Sistemas
mcbc@azc.uam.mx

Dr. Román Anselmo Mora Gutiérrez
Asociado D
Departamento de Sistemas
mgra@azc.uam.mx

Abril de 2016

Yo, Claudia Maricela Contreras Bravo, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Yo, Román Mora Gutiérrez, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Yo, Iván Flores Sánchez, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Índice General

Índice de Figuras

Índice de Tablas

1 Resumen

El Proyecto de Integración consistió en la creación de un programa que clasifica las descripciones de servicios web, como también el de analizar y medir el desempeño del mismo, las descripciones de los servicios web están contenidos en una ontología, la búsqueda se realizo de manera simple y más específica con un enfoque semántico.

La clasificación se realiza mediante la utilización de medidas de similitud semántica, es decir se comparan semánticamente servicios web, obteniendo de cada métrica un valor, que es el valor de similitud de cada servicio entre todos los demás servicios, estos datos son almacenados en diferentes matrices, para su posterior calculo mediante el programa creado y obtener una mejor clasificación de los servicios web.

En el Proyecto de Integración se emplearon métodos de clasificación eficientes como también un método innovador bio-inspirado sin dejar de lado los métodos heurísticos que se emplearon para encontrar la mejor solución en un menor tiempo de ejecución para la clasificación de los servicios web.

2 Introducción

"Hoy en día, el principal uso de la WWW es para el acceso a documentos y aplicaciones. En la mayoría de los casos, tal uso es por parte de los usuarios, que típicamente trabajan a través de buscadores, audio, video, u otros sistemas interactivos. La web puede crecer significativamente en poder y espacio si está sustentado en el soporte de la comunicación y las aplicaciones de un programador a otro."

Traducido de W3C XML Procol Working Group Chate.[1]

Un servicio web es un componente de software reutilizable que está disponible a través de Internet, utiliza un sistema de mensajería basada en XML[2] estandarizada, y no está ligada a ningún sistema operativo o un lenguaje de programación [3]. Un servicio es una interface de programación al que se puede acceder a través de Internet (ver Figura 1). Igual que una página web está definida por un URL[3], un servicio web que está definido por un URI[4] y por su interface, a través del cual se puede acceder a él.

La arquitectura de servicios web (ver Figura 2) está basada en una serie de protocolos, lenguajes y estándares interoperables que son ampliamente utilizados en redes públicas como Internet. Para la descripción de las interfaces de programación de los servicios web se emplea el lenguaje WSDL[5], el cual es un estándar de la W3C[6] que le permite a los proveedores de servicios publicar sus servicios y a los clientes solicitantes de servicios realizar búsquedas y utilizar la descripción del servicio seleccionado para crear un objeto de llamada remota (ver Figura 3).

Figura 1. Estructura de un servicio web básico

[1] http://www.w3.org/
[2] eXtensible Markup Language.
[3] Uniform Resource Locator.
[4] Uniform Resource Identification.
[5] Web Service Description Language.
[6] World Wide Web Consortium.

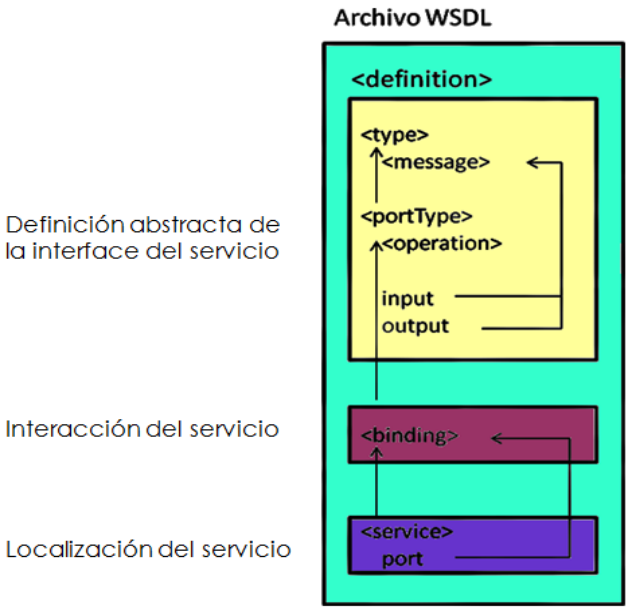


Figura 2. Estructura de un archivo WSDL

En la actualidad los servicios web se han incrementando conforme a las diferentes necesidades que cada usuarios tiene. Existen descripciones de servicios web dentro de internet como en servidores y en localidades llamados repositorios, la mayoría de estos repositorios ofrecen búsquedas de servicios con resultados no favorables para el usuario quien realiza la búsqueda.

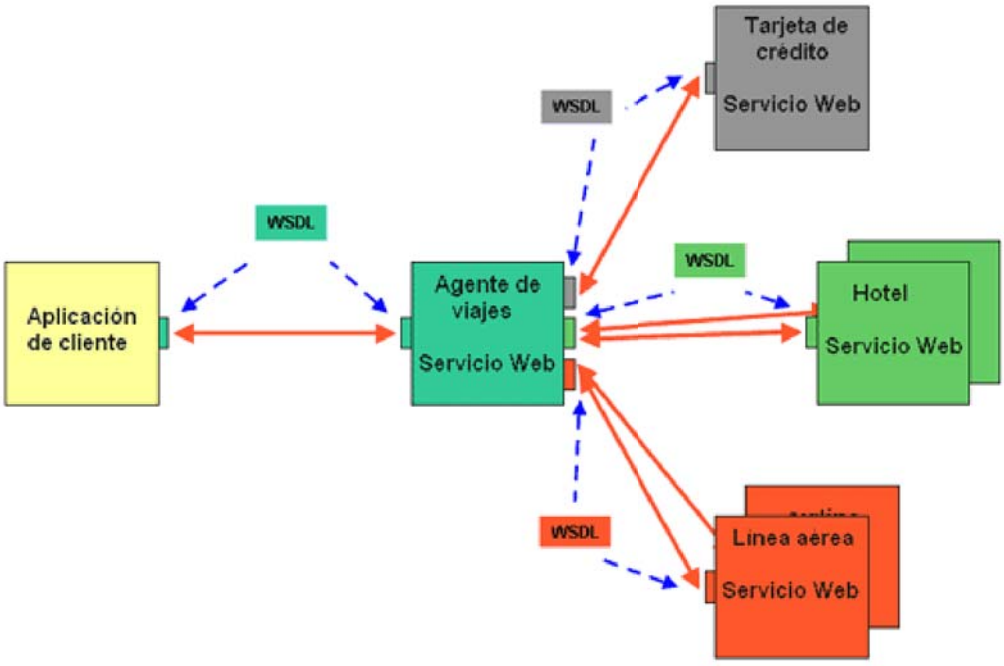


Figura 3. Diagrama de llamadas remotas a Servicios Web

3 Objetivo General

Adaptar e implementar un programa que permita clasificar servicios web mediante un hibrido[7] entre el algoritmo de Colonia Artificial de Abejas[8], K-Means[9] y Consenso.

3.1 Objetivos Específicos

1. Diseñar e implementar un módulo de adquisición y extracción de datos de descripciones de servicios Web.
2. Implementar un programa basado en el algoritmo Colonia Artificial de Abejas para la clasificación de servicios Web.
3. Implementar un repositorio que contenga la clasificación de los servicios web previamente analizados.
4. Evaluar mediante la comparación de los resultados obtenidos por medio del hibrido contra los resultados reportados en la literatura para estos servicios.

4 Justificación

Existen algunas formas de clasificar servicios web, en la actualidad los mismos repositorios en los que están contenidos los servicios web realizan dicha tarea, pero con resultados no satisfactorios en la obtención de la clasificación, ya que se requiere que el usuario proporcione información con el objeto de organizar dichos elementos [**7**].

Los avances en la tecnología son muchos actualmente y más en el ámbito de la informática y entre los cuales destacan los servicios web, desde los 80's se han implementado métodos basados en analogías biológicas para resolver problemas complejos de computación y optimización entre los que destacan algoritmos genéticos, estrategias evolutivas, comportamiento de algunas especies de animales, etcétera. Muestra de ello es la tendencia que ha aumentando y la inmensa fuente de inspiración que hay en la naturaleza para resolver problemas complejos y difíciles [1]. Los seres vivos como los animales y las plantas y el mismo clima exhiben extraordinarios, complejos y fascinantes fenómenos

[7] El termino hibrido se refiere a la combinación de dos o más técnicas heurísticas o metaheurísticas
[8] Colonia Artificial de Abejas, por sus siglas en ingles, Artificial Bee Colony (ABC) [6].
[9] K-MEANS, método de agrupamiento cuyo objetivo es la partición de un conjunto de *n* observaciones en *k* grupos en el que cada observación pertenece al grupo más cercano a la media.

naturales y uno de ellos es la forma de encontrar la mejor solución para resolver el problema y mantener el equilibrio perfecto entre sus componentes o el medio que los rodea.

Esta es la idea principal de la computación inspirada en la naturaleza, es decir, imitar los pasos que ha desarrollado la naturaleza y adaptarlos a un problema, convirtiéndolo así en un algoritmo bio-inspirado [4]**.** Son muchos los algoritmos bio-inspirados que imitan la naturaleza para resolver problemas de optimización [10].

En este Proyecto de Integración se implemento un programa que permite clasificar servicios web mediante el algoritmo Colonia Artificial de Abejas, para que un usuario realice la búsqueda de una manera más específica, con un tiempo de ejecución rápida y con un enfoque semántico, ya que actualmente en este proceso de búsqueda, los resultados son demasiados e irrelevantes ya que emplean palabras clave sin importar su significado.

El empleo de métodos convencionales para resolver problemas de optimización requiere enormes esfuerzos y recursos computacionales, que tienden a fallar en cuanto aumenta el tamaño del problema [1]. Esta es la razón principal de la implementación de un algoritmo que explote la característica de inteligencia colectiva[10], dicha característica la posee el algoritmo Colonia Artificial de Abejas como alternativa en la optimización de clasificar servicios web. Este algoritmo se basa en el mejoramiento iterativo de cualquier población de soluciones y en su mayoría emplear la aleatorización[11] y la búsqueda local como también búsqueda global para resolver un problema de optimización [10]. La Figura 4 muestra el diagrama básico que la gran mayoría de los algoritmos bio-inspirados siguen.

Figura 4. Diagrama básico de un Algoritmo bio-inspirado

[10] Inteligencia Colectiva también llamado Sistema Colectivo es capaz de resolver tareas difíciles en entornos dinámicos y variados sin ayuda o control externo y sin coordinación central.
[11] Aleatorización. Escoger de un conjunto de soluciones al azar cualquier solución sin hacer alguna distinción.

5 Antecedentes

5.1 Proyectos Terminales

1. *Clasificación de servicios web mediante ontologías* [2].
Éste Proyecto Terminal es muy similar al proyecto que se pretende realizar, ya que realiza un agrupamiento de servicios web, aunque hace uso de ontologías, aspecto del cual no se hará uso en el proyecto a realizar, la diferencia estriba en la implementación de un algoritmo bio-inspirado.

2. *Extracción automatizada y representación de servicios web mediante ontologías* [8].
Éste Proyecto Terminal se enfoca a servicios web mediante la extracción automatizada, cuyo contenedor son las ontologías, es similar al proyecto a realizar ya que también hará uso de la extracción de servicios web. La diferencia radica en la implementación de un algoritmo de optimización bio-inspirado.

5.2 Artículos

3. *A Hybrid Artificial Bee Colony Algorithm for the Service Selection Problem* [11].
Esta investigación que se realizo es muy similar a la propuesta, ya que hace uso del algoritmo ABC en combinación con otros algoritmos y lo mismo se pretenderá hacer en este proyecto, sin embargo la diferencia radica que en el artículo se enfoca a resolver problemas de selección de servicios web y en este proyecto solo se realizara la optimización en la clasificación de los servicios web.

4. *Web Services Selection Based on Discrete Artificial Bee Colony Algorithm* [9].
Esta investigación es similar al trabajo que se llevará a cabo, sin embargo una de las diferencias más grandes radica en el uso de un híbrido, que combina los algoritmos ABC con K-Medias y consenso. Además de las características de calidad de servicio (QoS, por sus siglas en inglés), mientras que en este proyecto no se emplearán esas características sino los parámetros de entrada y salida y los nombres de las operaciones de los servicios.

5.3 Tesis

5. *Adaptación del algoritmo de la colonia artificial de abejas para resolver problemas de diseño en ingeniería* [5].
Esta investigación es parecida al proyecto que se pretende realizar ya que también habrá una adaptación en el algoritmo ABC, pero enfocada en algo muy en especifico como son los servicios web y que sin en cambio en la tesis abarca temas más generales como son los problemas de diseño en ingeniería.

6 Desarrollo del Proyecto

El Proyecto de Integración desarrollado forma parte de un proyecto de investigación de mayor alcance cuyo objetivo es el de diseñar e implementar un directorio semántico de servicios web y recursos de programación utilizando métodos eficientes, totalmente automatizados, que permitan la organización, composición y clasificación de servicios web con un enfoque semántico. El cual está compuesto por módulos cada uno con un objetivo específico pero que sin duda cada modulo es auxiliar del modulo que le sigue, en particular, con este trabajo se contribuyo con el módulo de clasificación y agrupamiento de servicios web y para su funcionamiento hace uso de los módulos que le preceden (ver Figura 5).

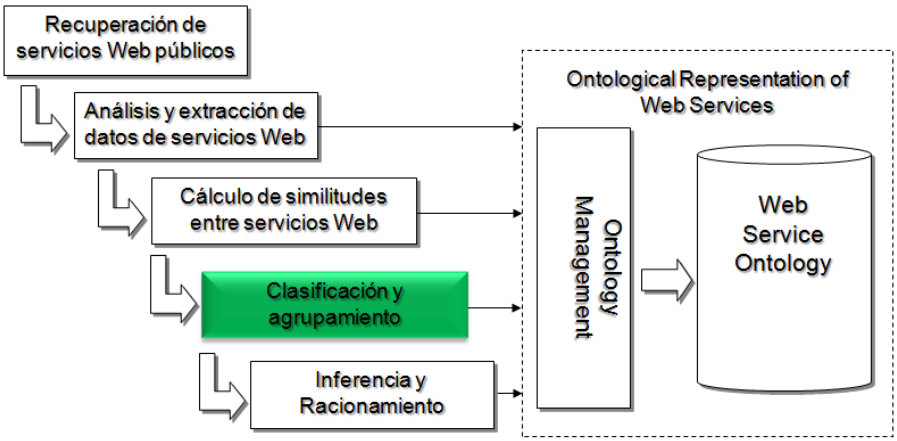


Figura 5. Proyecto de Investigación

Los servicios web pueden ser extraídos de la red mediante el uso de un crawler[12] de código abierto, para este Proyecto de Integración se uso un repositorio local el cual contiene colecciones de servicios web públicos, 1027 servicios web públicos en total fueron con los que se trabajo y con los cuales se realizaron pruebas y análisis para la clasificación.

Los datos específicos para poder realizar las mediciones de la colección de los servicios web se utilizo el modulo de *Análisis y extracción de datos de servicios web* que consistía de un parser[13] para poder extraer toda la información necesaria de cada uno de los servicios web cuyos valores son almacenados en una ontología, se crearon en particular 3 archivos OWL[14] de tamaño 50, 647 y 1027 de los servicios web respectivamente (ver Figura 6).

[12] Crawler, programa que inspecciona las páginas en Internet de forma metódica y automatizada.
[13] Parser, programa que convierte en tokens cada una de los parámetros que tiene como entrada.
[14] Web Ontology Language (OWL). Es un lenguaje de etiquetas para publicar y compartir datos usando ontologías en la web.

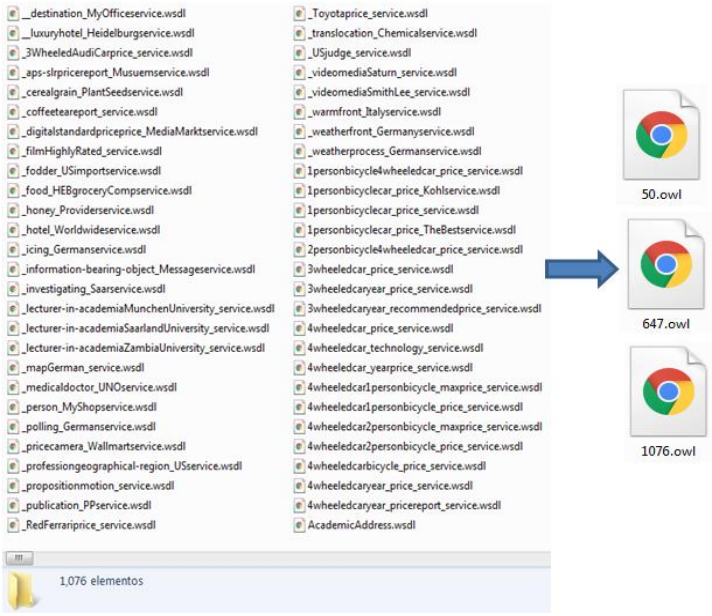


Figura 6. Extracción de la información hacia las ontologías

El modulo de *Cálculo de Similitudes entre servicios web* se utilizo para calcular la similitud de los datos que se obtuvieron de las ontologías, cabe resaltar que este modulo fue modificado y adaptado para este Proyecto de Integración ya que se incorporo nuevas métricas[15] para el cálculo de los datos de similitud sumando así ocho medidas de similitud (ver Anexo B), los cuales arrojaron nuevos datos para ser analizados y obtener un análisis más completo (ver Figura 7).

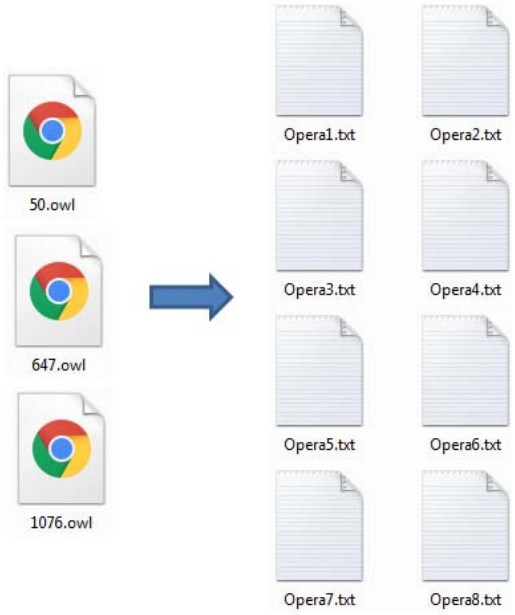


Figura 7. Obtención de las 8 Matrices de Similitud

[15] http://code.google.com/archive/p/ws4j/

La estructura general de este Proyecto de Integración, que es el modulo de Clasificación y Agrupamiento emplea un algoritmo hibrido y se muestra en la figura 8, cuyo funcionamiento como estructura se describe con más detalle en las siguientes secciones.

El Proyecto consta de los siguientes módulos:

Repositorio de los servicios web local.

- Este módulo tiene todo un conjunto de servicios web, que posteriormente serán agrupados, los servicios web ya han sido almacenados anteriormente por una búsqueda del usuario.

Modulo de adquisición y lectura de datos de los servicios web.

- Este módulo realiza la tarea de reunir toda la información de los servicios web para su posterior clasificación.

Modulo de clasificación de los servicios web mediante el hibrido

- Este módulo del hibrido permitirá:
  1. Determinar cuántos grupos deben generarse.
  2. Determinar los centroides de los grupos.
  3. Agrupar los servicios en función de los centros generados.

Modulo del repositorio para los servicios web clasificados.

- Este modulo albergara todo los servicios web una vez clasificados, el objetivo de este modulo es contener todo un agrupamiento optimo.

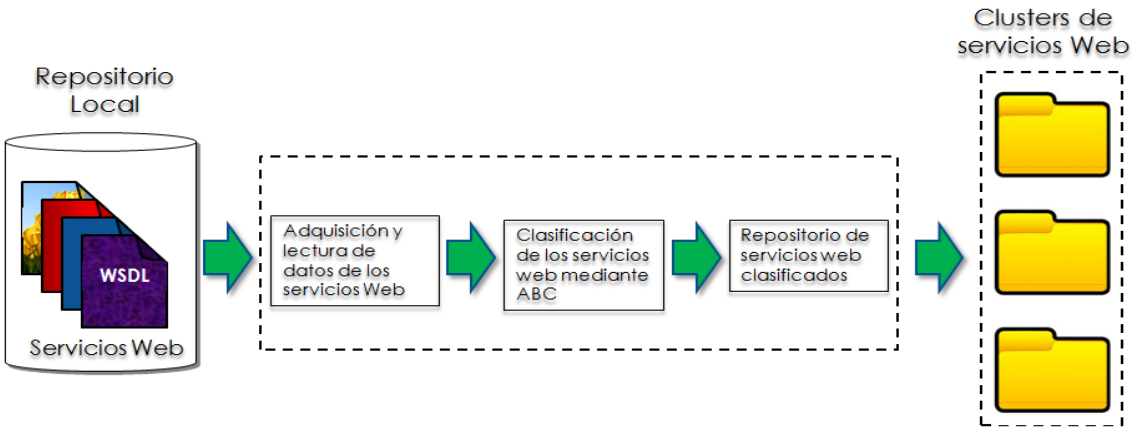


Figura 8. Esquema básico del Proyecto

6.1 Recursos

Para la realización del Proyecto de Integración se utilizo los siguientes recursos de libre distribución:

Hardware

- Computadora portátil Compaq Presario CQ40 Notebook con procesador Intel Celeron 900 2.20GHz, 2 GB de memoria, 150 Gb de Disco Duro, con sistema operativo Windows 7 Ultimate Service Pack 1

- Computadora portátil Toshiba con procesador Intel Core Duo T5550 1.83GHz, 4 GB de memoria, 150 de Disco Duro, con sistema operativo Windows 7 Ultimate Service Pack 1

- Servidor Dell con procesador Intel (R) Xeon (R) CPU 3.30 GHz dual, 64 GB de memoria, con sistema operativo Windows Server 2012 R2 Data center de 64 bits.

Software

- Lenguaje de programación Java[16].
  - ✓ IDE Eclipse[17].
- Parser de servicios Web.
- Servidor Tomcat 7.0.64[18].

6.2 Especificación Técnica

El programa creado tiene como parámetros de entrada los archivos que son generados de los módulos mencionados anteriormente los cuales hacen uso de servicios web almacenados en un repositorio local, a los cuales se extrajeron los datos de las operaciones de los servicios y de los parámetros de entrada y salida.

Para el proceso de comparar semánticamente las operaciones de entrada y de salida de cada servicio se utilizo las siguientes métricas de similitud semántica de libre distribución y que haces uso de WordNet[19]:

[16] Java, Lenguaje de programación informática creada por Sun Microsystems.
[17] Eclipse, Entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java.
[18] http://tomcat.apache.org/

- WUP[20]
- LIN[21]
- PATH[22]
- LESK[23]
- HSO[24]
- JCN[25]
- LCH[26]
- RES[27]

Las métricas de similitud regresan un valor entre 0 y 1 donde si el parentesco semántico entre los servicios es diferente se le asigna un valor aproximado a 0 y si hay alguna característica de parentesco semántico entre los servicios se le asigna un valor muy cercano a 1(ver Figura 9).

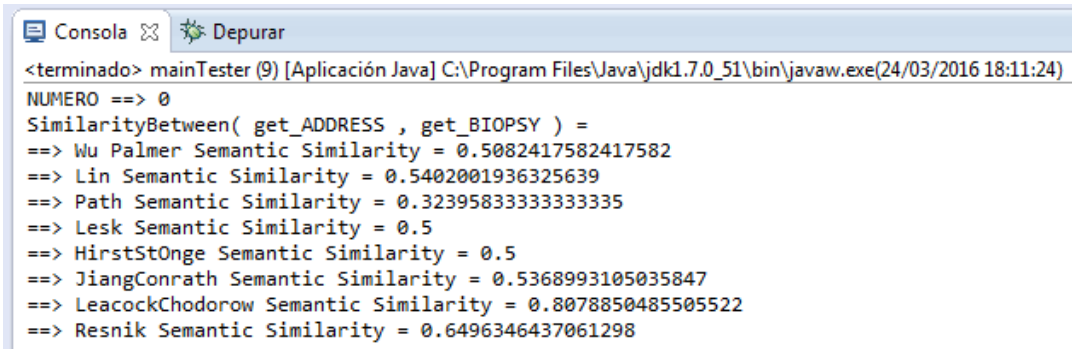


Figura 9. Ejecución del programa que calcula la similitud entre servicios

El valor de similitud de cada métrica es almacenado en una matriz, es decir cada archivo creado contiene la información de cada métrica que se utilizo para este proyecto y que es obtenido al ejecutar el programa.

El proceso de clasificación se efectúa una vez que se crearon los archivos que contienen la información de las similitudes de los servicios web mediante el algoritmo implementado en este Proyecto de Integración el cual se describe a continuación.

---

[19] Base de datos léxica del ingles, https://wordnet.princeton.edu/
[20] http://search.cpan.org/dist/WordNet-Similarity/lib/WordNet/Similarity/wup.pm
[21] http://search.cpan.org/dist/WordNet-Similarity/lib/WordNet/Similarity/lin.pm
[22] http://search.cpan.org/dist/WordNet-Similarity/lib/WordNet/Similarity/path.pm
[23] http://search.cpan.org/dist/WordNet-Similarity/lib/WordNet/Similarity/lesk.pm
[24] http://search.cpan.org/dist/WordNet-Similarity/lib/WordNet/Similarity/hso.pm
[25] http://search.cpan.org/dist/WordNet-Similarity/lib/WordNet/Similarity/jcn.pm
[26] http://search.cpan.org/dist/WordNet-Similarity/lib/WordNet/Similarity/lch.pm
[27] http://search.cpan.org/dist/WordNet-Similarity/lib/WordNet/Similarity/res.pm

6.3 Descripción del ABC

La idea principal del algoritmo Colonia Artificial de Abejas (ABC) es imitar el comportamiento de búsqueda y recolección de alimento haciendo uso de recursos muy específicos como son la exploración[28], explotación[29], reclutamiento[30] y abandono[31]. El algoritmo ABC está compuesto por dos grupos de abejas: las abejas obreras y las abejas en espera. Las abejas en espera se subdividen a su vez en: abejas exploradoras y en abejas espectadoras, estas dos clases de abejas (obrera y espectadora) tiene la características de elegir y ajustar mediante su experiencia (memoria) la fuente de alimento. En el algoritmo ABC se asume que cada posición de la fuente de alimento equivale a una solución factible y la cantidad de alimento que se recolecta de la fuente de alimento representa la calidad asociada a la solución [10], (ver Figura 11). La Figura 10 resume el algoritmo del ABC.

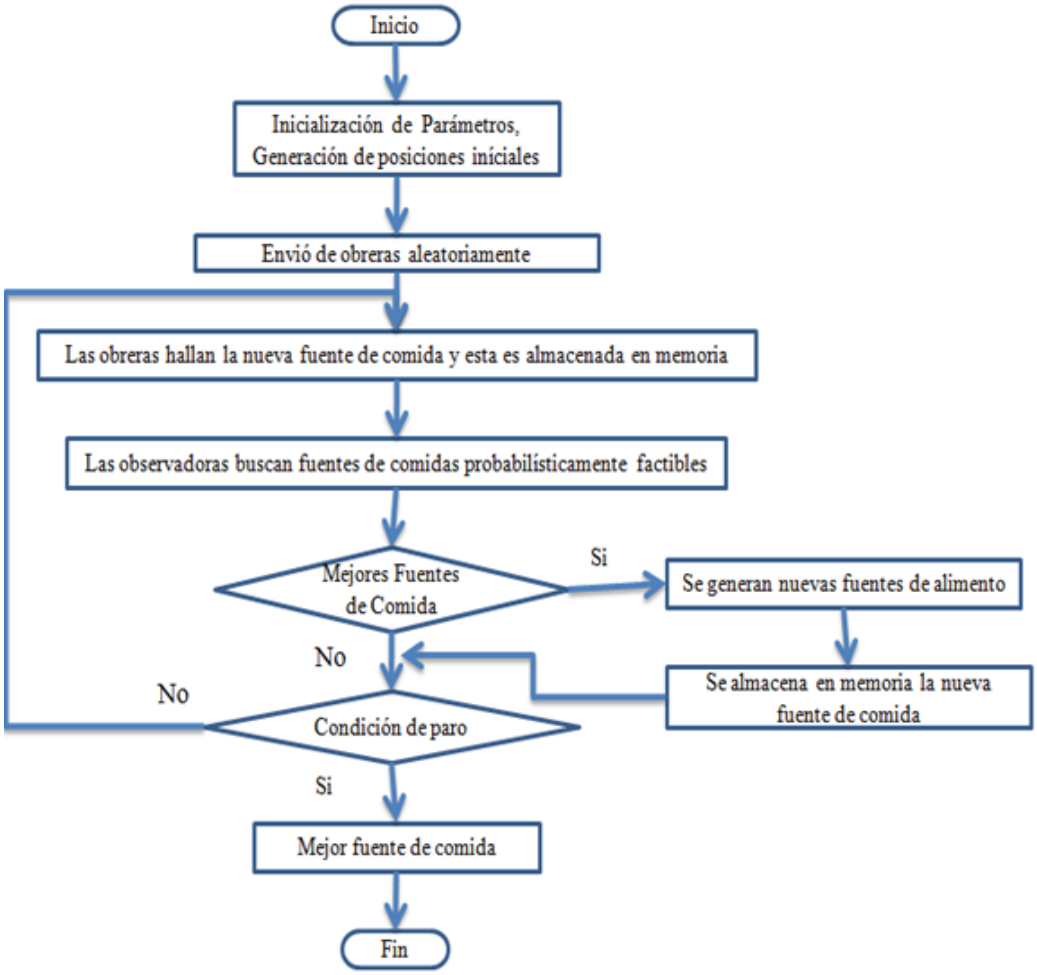
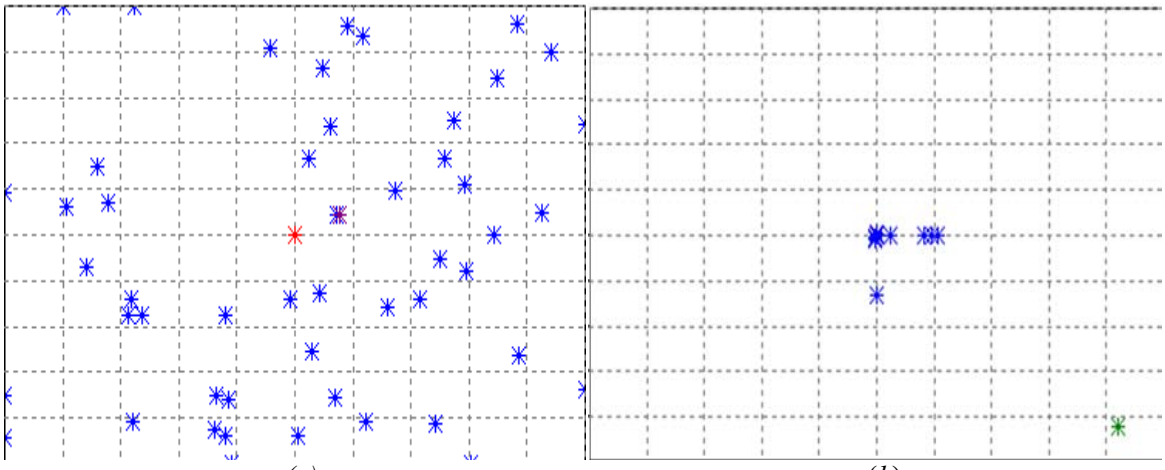


Figura 10. Diagrama general del algoritmo ABC

[28] Acción que realizan las abejas desempleadas para nuevas fuentes de alimento.
[29] Tarea que llevan a cabo las abejas obreras en una fuente de alimento.
[30] Acción que realizan las abejas desempleadas con las abejas obreras para explotar una fuente de alimento.
[31] Sucede cuando se encuentra una mejor fuente de alimento.



(a) (*b*)

Figura 11.Grafica del comportamiento del algoritmo ABC. Color Azul = Abeja Obrera, Color Verde = Abeja Exploradora, Color Purpura = Mejor solución obtenida, Color Rojo = Optimo deseado. (a) Comienza el algoritmo ABC posicionando a las obreras aleatoriamente. (b) Después de una serie de ejecuciones es alcanzada la solución factible.

La combinación de otros métodos de agrupamiento como es el caso del método de consenso, permite al algoritmo bio-inspirado la creación de grupos que se generan desde 2 grupos (valor mínimo de grupos permitidos) hasta el número de servicios web que se utilizaron entre 2 (valor máximo de grupos que se podrán generar). Para el caso del método K-Means, el algoritmo selecciona elementos dentro de los grupos generados anteriormente convirtiéndolos en centroides con los cuales se podrá medir la similitud, también el hibrido hace uso de métodos heurísticos al generar números aleatorios que el algoritmo utiliza para el cálculo en la generación de grupos también en la selección de los centroides utilizando las operaciones básicas matemáticas como complejas como probabilidades.

La solución se presenta como un vector de entrada de tamaño *n* (numero de servicios web a clasificar) donde el elemento en el vector, devolverá el grupo al que pertenece. La función objetivo que el híbrido emplea es:

$$\text{Min} \sum_{\substack{i=1 \\ x \in ci \\ y \in ci}}^{C} d(x_i, y_i)$$

donde:

d : Distancia

y_i : Centro del *i-ésimo* cluster.

x_i : Uno de los servicios incluidos en el *i-ésimo* cluster. Ningún cluster puede estar vacio y no puede existir intersección entre los clusters.

6.4 Funcionamiento del Hibrido

La primera etapa que realiza el hibrido es la filtración de las 8 matrices que contienen la información de similitudes de los servicios web mediante la ecuación (1), descartando valores que excedan los límites permitidos y establecidos por las mismas métricas, se generan nuevas matrices filtradas con un 95% de certeza en las mediciones.

$$\mathcal{X} - 1.96\ \frac{\sigma}{\sqrt{N}} \leq \mu \leq \mathcal{X} + 1.96\ \frac{\sigma}{\sqrt{N}} \qquad (1)$$

donde:

$\mathcal{X}$: Matriz Promedio.
1.96 : Constante.
σ : Desviación estándar.
N : Elemento de la matriz de similitud.
μ : Similitud Promedio.

Posteriormente filtradas las 8 matrices todos los datos obtenidos se almacenaran en una matriz promedio (fuentes de alimentos) descartando las posiciones que contengan información nula ó 0 es decir se promediaron todos aquellos valores que si fueron filtrados y aceptados como valores factibles cuya información si aporta gran ayuda al algoritmo.

En una etapa posterior se lee la matriz de similitud promedio para generar por primera vez una serie de arreglos (abejas) los cuales tendrán almacenada información, por ejemplo los grupos generados, los centroides seleccionados etcétera (ver Figura 12).

Estos datos son los que decidirán los cambios de grupos (conversión de una abeja obrera a exploradora) mediante la ecuación (2), ecuación que crea nuevos vectores para la localización de nuevas soluciones (nuevas fuente de alimentos), también indican la distancia que están con el centro de cada grupo generado (ver Figura 13), cuyo objetivo principal es minimizar la similitud de los centroides de cada grupo y por lo tanto el de maximizar la similitud entre miembros de cada grupo al centro correspondiente.

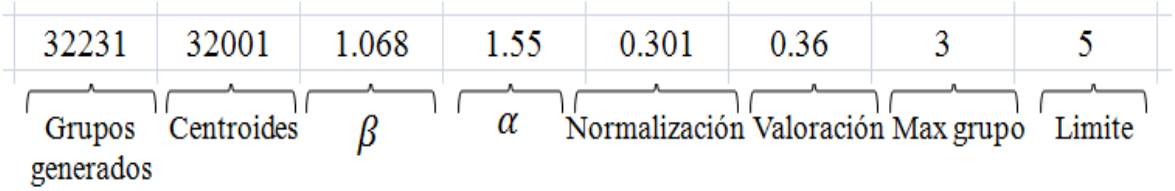


Figura 12. Composicion de un vector con información de los grupos generados, los centroides seleccionados aleatoriamente, β es la sumatoria de la similitud entre centroides de cada grupo, α es la sumatoria de la similitud entre miembros de cada grupo al centro correspondiente y que son necesarios para el cálculo de la Normalización y Validación cuyo valor es de vital importancia en la elección de nuevos vectores en el algoritmo ABC, valor máximo del grupo generado y un contador o limite.

$$X'_{new} = X'_i - \phi(X'_i - X'_s) \quad (2)$$

donde:

X'_{new} : Nuevo Vector generado.

X'_i : Primer vector que genera el algoritmo.

ϕ : Numero Aleatorio entre 0 y 1.

La siguiente etapa que sigue el hibrido es obtener el valor máximo de la valoración, para ello se ejecuta un ciclo de iteraciones con los cuales los vectores retoman las etapas posteriores, es decir generan nuevos grupos como centroides y calculan la Normalización y la Validación con ciertas restricciones como por ejemplo si no pueden mejorar la validación anterior el valor de limite que lo compone aumenta, en caso contrario este campo se vuelve a cero y la valoración con el valor obtenido es almacenado, en caso de alcanzar su valor máximo el campo de limite no solo se vuelve a cero, en este caso todo el vector es generado completamente como se menciono anteriormente, esto sucede en cada uno de los vectores como se muestra en las siguientes figuras 13 y 14.

322313	20310	0.4833	0.7132	0.0481	0.0699	3	0
221212	20010	0.5553	2.9279	0.1977	0.1515	2	0
211211	200010	0.4921	2.288	0.1545	0.1239	2	0
322313	200031	0.3072	0.5974	0.0432	0.0625	3	1
221212	20010	0.5553	2.9279	0.2118	0.1799	2	0
211211	1002	0.3644	1.9757	0.1429	0.12	2	1

Figura 13. Conjunto de vectores, Color Rojo: Conjunto de vectores inicio, Color Azul: Conjunto de vectores que se modifican en cada iteración en el algoritmo

322313	130200	0.3456	0.6598	0.0477	0.0698	3	2
221212	20010	0.5553	2.9279	0.212	0.1798	2	1
211211	21	0.3968	1.4853	0.1075	0.1065	2	3
322313	103200	0.3456	0.6598	0.0498	0.0688	3	3
221212	20010	0.5553	2.9279	0.221	0.181	2	0
211211	100002	0.2983	1.775	0.1339	0.1049	2	4
			·				
			·				
			·				
322313	102030	0.3585	0.811	0.0448	0.0674	3	10
221212	20010	0.5553	2.9279	0.1619	0.1506	2	1
211211	120000	0.4153	2.2007	0.1217	0.1129	2	0
332211	300210	0.6149	0.6403	0.0392	0.0937	3	0
221212	20010	0.5553	2.9279	0.1792	0.1566	2	1
211211	100020	0.3577	2.288	0.1401	0.1132	2	0
332211	20310	0.4833	0.7132	0.0481	0.0937	3	0
221212	20010	0.5553	2.9279	0.221	0.181	2	0
211211	20100	0.352	2.005	0.1144	0.1132	2	0

Figura 14. El conjunto de vectores que contiene solo puntos indican una secuencia de ejecuciones, Color Verde: Conjunto de vectores con los resultados finales

7 Implementación

Los parámetros con los que fue ejecutado el algoritmo fueron varios, por ejemplo las 8 matrices cuadradas que se filtraron fueron de tamaño 5, 50, 647 y 1024 respectivamente, la generación de grupos de vectores iníciales fue de 4 vectores para cuestiones de práctica y análisis en la construcción del algoritmo, los cuales se acordó en generar grupos de 10 vectores una vez construido el algoritmo, el valor de "phi" (φ) en un inicio fue de 0.2 calibrando el algoritmo se estableció en 0.8, el valor de la variable límite que conforma cada vector desde un principio se estableció con el valor de 10, el tamaño de ciclos con los que el programa se ejecuto también estuvo en contante cambio, la generación de números aleatorios también fue modificado ya que se utilizo una forma más compleja para la generación de los mismos, el programa está diseñado para poder mostrar el vector completo, es decir mostrar toda la información que se genera (grupos generados, centroides, β, α, normalización, validación, tamaño máximo de grupo y limite) pero para uso más práctico en las pruebas solo se muestran la valoración y el tamaño máximo de grupo que se genero, las siguientes tablas resumen los ejecuciones con los diferentes parámetros que se modificaron.

Tamaño Matriz	100 iteraciones	vectores	φ
50	0.4631	10	0.8
647	0.4457	10	0.8
1027	0.4414	10	0.8

Tabla 1. Configuración de instancias con 100 iteraciones.

Tamaño Matriz	200 iteraciones	vectores	φ
50	0.5011	10	0.8
647	0.4152	10	0.8
1027	0.4782	10	0.8

Tabla 2. Configuración de instancias con 200 iteraciones.

Tamaño Matriz	500 iteraciones	vectores	φ
50	0.5169	10	0.8
647	0.4228	10	0.8
1027	0.4542	10	0.8

Tabla 3. Configuración de instancias con 500 iteraciones.

Tamaño Matriz	Mejores Resultados
50	500 iteraciones
647	100 iteraciones
1027	200 iteraciones

Tabla 4. Resumen con las mejores configuraciones obtenidas.

8 Resultados

A continuación se muestran los resultados obtenidos en forma de tablas con las mejores configuraciones, también se muetran las graficas resultado de las mismas:

	100			200			500		
	Mejor	**Peor**	**Promedio**	**Mejor**	**Peor**	**Promedio**	**Mejor**	**Peor**	**Promedio**
	0.3322116	0.0563415	0.1009004	0.4981613	0.0439323	0.0982948	0.5205043	0.0411708	0.0989385
	0.5190820	0.0369711	0.1445253	0.4490453	0.0410601	0.0935569	0.5351554	0.0378401	0.0975018
	0.4334746	0.0480076	0.0991093	0.4902191	0.0398102	0.1050635	0.5364543	0.0397102	0.0983724
	0.5575567	0.0353303	0.1150988	0.4752771	0.0383082	0.1280018	0.5458992	0.0458110	0.1018580
	0.4106739	0.0371488	0.1083493	0.5020176	0.0370918	0.0948430	0.5014388	0.0384841	0.1075409
	0.2927498	0.0630970	0.1040203	0.5397763	0.0393485	0.1212832	0.5294111	0.0416599	0.1100818
	0.4846925	0.0466351	0.1124559	0.4612397	0.0409208	0.1066123	0.5365068	0.0454492	0.1163531
	0.5353784	0.0404557	0.1186004	0.5350821	0.0417565	0.1023592	0.5106629	0.0341039	0.0940143
	0.4945885	0.0437915	0.0956999	0.5485311	0.0410027	0.1002204	0.4655496	0.0334956	0.1357247
	0.4932416	0.0456645	0.1025432	0.5309521	0.0460585	0.0987517	0.5244522	0.0433163	0.0987347
	0.3808616	0.0486251	0.1036605	0.4792554	0.0415437	0.1017561	0.5046523	0.0351837	0.0982017
	0.5893620	0.0298631	0.0993391	0.4812489	0.0305092	0.0986736	0.4907689	0.0408167	0.1106888
	0.3816476	0.0400868	0.0935831	0.5023888	0.0460301	0.1032030	0.5696470	0.0395370	0.1019360
	0.4951235	0.0454474	0.1448746	0.5251207	0.0391945	0.1002726	0.5716353	0.0353065	0.0993421
	0.4467321	0.0479013	0.1022506	0.4646931	0.0427688	0.1192913	0.5063083	0.0412513	0.095093
	0.5101395	0.0404475	0.1030672	0.5296219	0.0446670	0.1034013	0.5090425	0.0381359	0.0990068
	0.4899370	0.0454912	0.1376305	0.4952296	0.0411123	0.0962395	0.4698080	0.0440028	0.1042793
	0.4698910	0.0432380	0.0953103	0.5336771	0.0429933	0.1017662	0.5434270	0.0347715	0.0984205
	0.4039364	0.0505077	0.1038958	0.4684326	0.0437512	0.1396249	0.4746606	0.0334925	0.1009078
	0.5418479	0.044463	0.1236261	0.5129643	0.0408044	0.1009319	0.4936065	0.0329317	0.0920672
Mejor	0.5893620	0.0630970	0.1448746	0.5485311	0.0460585	0.1396249	0.5716353	0.0458110	0.1357247
Peor	0.2927498	0.0298631	0.0935831	0.4490453	0.0305092	0.0935569	0.4655496	0.0329317	0.0920672
Promedio	0.4590148	0.0444764	0.1097323	0.5005247	0.0411505	0.1059587	0.5182097	0.0391336	0.1035261
Varianza	0.0059335	5.4433E-05	0.0002497	0.0008830	1.2008E-05	0.0001431	0.0009008	1.6939E-05	9.526E-05
D Estándar	0.0770295	0.0073778	0.0158017	0.0297154	0.0034653	0.0119655	0.0300134	0.0041156	0.0097601

Tabla 5. Resultados de la Similitud de las 50 Instancias con 100, 200 y 500 iteraciones

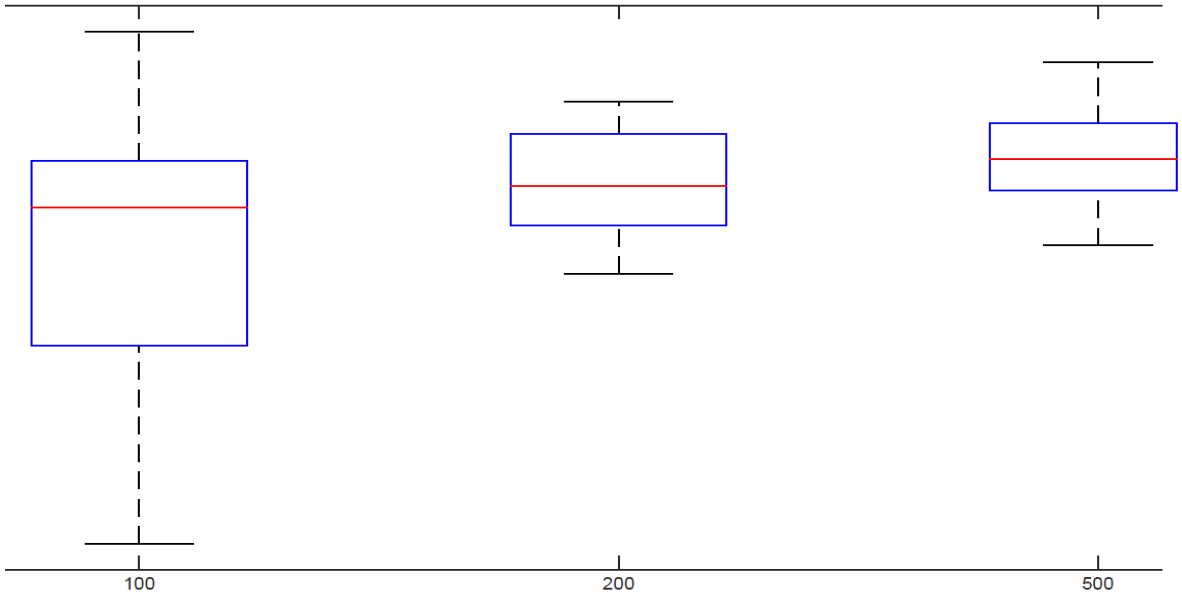


Figura 15. Comparativo de similitud entre las mejores soluciones encontradas con 50 servicios

	100			200			500		
	Mejor	**Peor**	**Promedio**	**Mejor**	**Peor**	**Promedio**	**Mejor**	**Peor**	**Promedio**
	0.2153186	0.04708268	0.09768613	0.52395334	0.04048671	0.0960905	0.54881541	0.04213923	0.096217
	0.53929347	0.04579912	0.12377976	0.52888873	0.04850837	0.14610007	0.35860715	0.0532852	0.10652016
	0.50550715	0.04645278	0.09697993	0.54118636	0.04531317	0.09792119	0.48874034	0.04870091	0.09879442
	0.4107395	0.05308694	0.09942944	0.37001832	0.04109386	0.09420772	0.53679182	0.04537954	0.09742133
	0.34259407	0.05590719	0.09739855	0.53060795	0.04556273	0.09693626	0.24856467	0.06081254	0.09803991
	0.47327271	0.04779748	0.09801848	0.20257995	0.05318307	0.09698382	0.22262872	0.05463912	0.09020784
	0.52967807	0.04537187	0.09723884	0.3354368	0.0506458	0.09891759	0.32165086	0.04968344	0.09467435
	0.39152403	0.05129277	0.09625431	0.28910657	0.05180847	0.09819475	0.26415789	0.0484095	0.09565052
	0.48609226	0.04512111	0.10086034	0.54778745	0.04065674	0.0993225	0.5185978	0.04371084	0.0962829
	0.48910908	0.04495656	0.09921112	0.2701243	0.06075739	0.09343512	0.38194056	0.05043058	0.09584534
	0.40743852	0.05330588	0.09934789	0.5413542	0.04429324	0.09650627	0.54085513	0.04747848	0.09880171
	0.40391961	0.04893022	0.09544515	0.34075583	0.04955581	0.0912395	0.39855868	0.05031664	0.09854731
	0.5327554	0.04714912	0.09978197	0.54673284	0.04340013	0.09852777	0.53596094	0.0463246	0.09735382
	0.33850524	0.05323068	0.09843432	0.44614786	0.05095455	0.0969124	0.43354759	0.05097802	0.09800627
	0.2284805	0.05336027	0.08869007	0.49322173	0.04068958	0.09578279	0.50087956	0.04777307	0.09687251
	0.53247242	0.04228284	0.09687289	0.47561538	0.0482518	0.09385443	0.5342804	0.04614646	0.09763114
	0.54947584	0.04315878	0.09701713	0.30187468	0.04865296	0.09713459	0.23621277	0.05253534	0.09200647
	0.54677893	0.04274006	0.09864191	0.24936309	0.04922921	0.09391703	0.41286051	0.04950698	0.09643241
	0.51332023	0.04648238	0.09808696	0.24062337	0.05184011	0.09528061	0.47949394	0.04828251	0.09797936
	0.47848759	0.04622322	0.09893596	0.52927081	0.04622953	0.09928046	0.49449973	0.04611704	0.09779277
Mejor	0.54947584	0.05590719	0.12377976	0.54778745	0.06075739	0.14610007	0.54881541	0.06081254	0.10652016
Peor	0.2153186	0.04228284	0.08869007	0.20257995	0.04048671	0.0912395	0.22262872	0.04213923	0.09020784
Promedio	0.44401451	0.04807941	0.09890396	0.40923046	0.04762546	0.09880342	0.41911288	0.04929121	0.09701499
Varianza	0.01027133	1.6153E-05	4.0365E-05	0.01554565	2.6212E-05	0.0001285	0.01280278	1.72E-05	9.7324E-06
D Estándar	0.10134758	0.00401904	0.00635332	0.12468218	0.00511977	0.01133589	0.11314939	0.0041473	0.00311968

Tabla 6. Resultados de la Similitud de las 647 Instancias con 100, 200 y 500 iteraciones

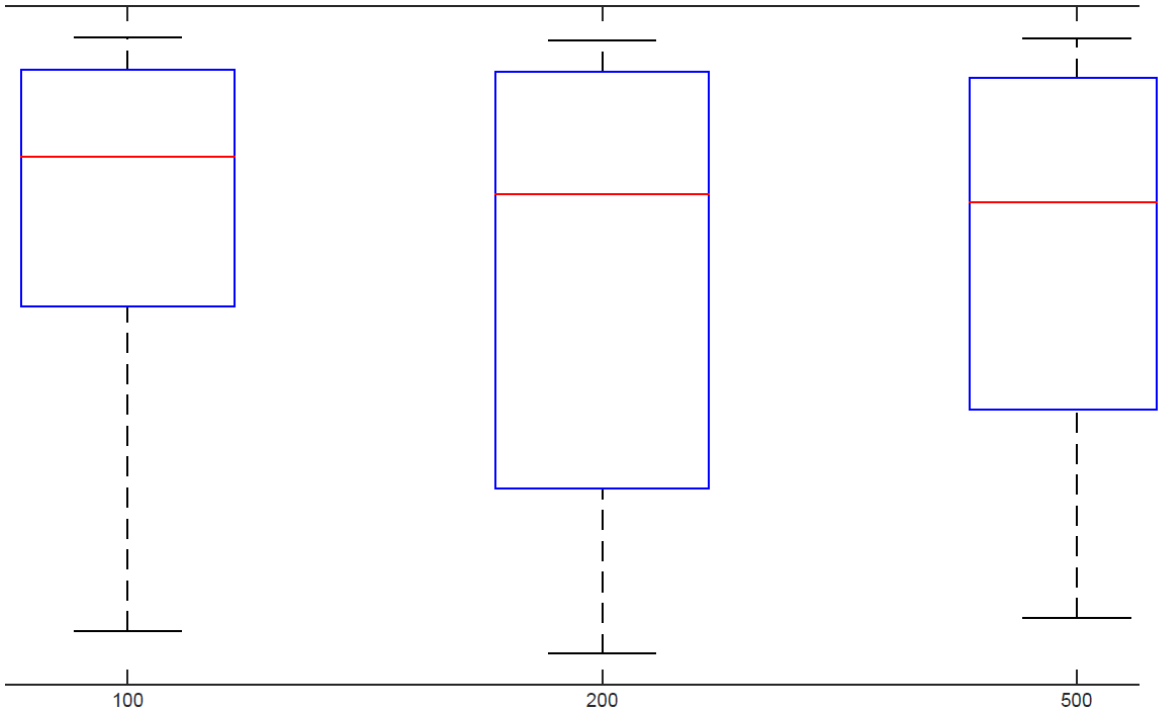
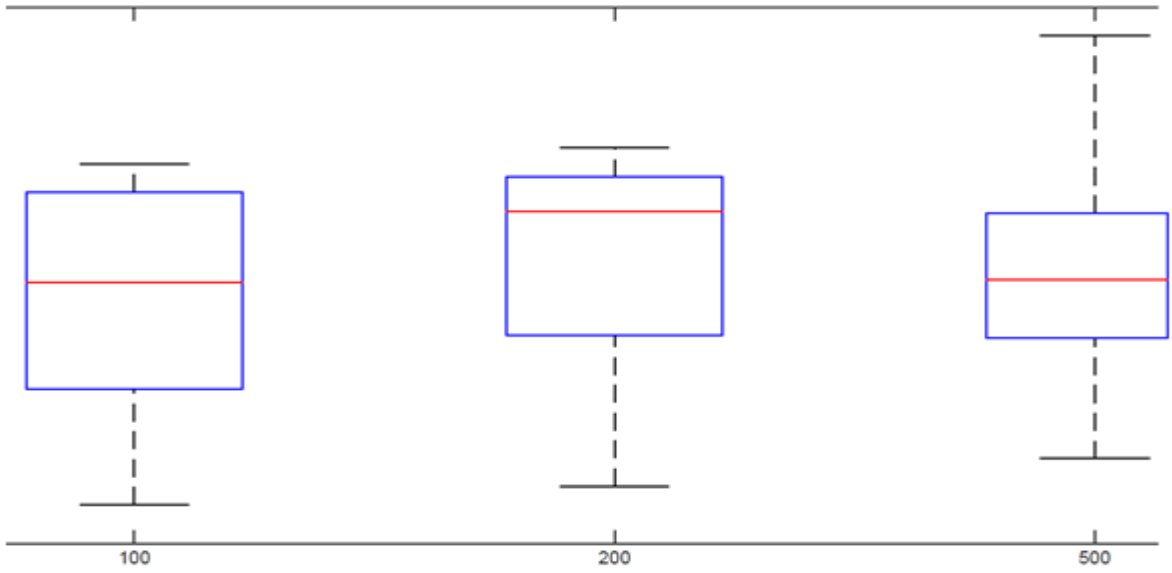


Figura 16. Comparativo de similitud entre las mejores soluciones encontradas con 647 servicios

	100			200			500		
	Mejor	**Peor**	**Promedio**	**Mejor**	**Peor**	**Promedio**	**Mejor**	**Peor**	**Promedio**
	0.49233249	0.04398484	0.09563675	0.54016378	0.04552551	0.09711782	0.51172364	0.04948446	0.0985225
	0.46859254	0.04777328	0.09618077	0.55123427	0.0384052	0.09697024	0.53356765	0.04488606	0.09691675
	0.42945425	0.0475179	0.09798991	0.53149176	0.04665506	0.09734941	0.44531389	0.04950754	0.09279642
	0.55240422	0.04097989	0.09729078	0.38417152	0.04812969	0.10039072	0.66112025	0.03447956	0.09866241
	0.40123004	0.05171223	0.13381499	0.47626708	0.04952645	0.10155157	0.50299292	0.04813204	0.09899422
	0.44550333	0.04483043	0.09346987	0.50966332	0.04777525	0.1000348	0.43481648	0.05130512	0.09544344
	0.31820053	0.05493566	0.09334663	0.56846422	0.0465486	0.09981139	0.35879715	0.05664075	0.09623362
	0.53571592	0.0459108	0.09799064	0.34828884	0.04983684	0.0974804	0.31200901	0.06174495	0.09950224
	0.48359442	0.05206164	0.0998819	0.56522096	0.03944667	0.09589788	0.51610158	0.04018082	0.09817397
	0.42974501	0.04601709	0.10426243	0.28919387	0.04876465	0.09345679	0.46252878	0.04754935	0.09641645
	0.32077948	0.04934677	0.10263097	0.47040663	0.04888471	0.09997775	0.45609362	0.04611206	0.09171088
	0.5300962	0.04479657	0.10333807	0.43566463	0.04957788	0.09385006	0.42201354	0.05206895	0.09737673
	0.53394262	0.04556478	0.09919599	0.54472464	0.04881704	0.10072275	0.5132191	0.04651397	0.09684895
	0.55531806	0.0355509	0.09615256	0.54860021	0.04667821	0.09944591	0.34764665	0.05208849	0.09103648
	0.47903273	0.05197162	0.1009656	0.54352817	0.04764264	0.09883381	0.47842453	0.04854434	0.09971215
	0.27378002	0.05473007	0.09524966	0.39113724	0.04350337	0.0975473	0.40012834	0.04885275	0.10021026
	0.36274032	0.05249036	0.09448587	0.52193753	0.04267194	0.09780841	0.55060552	0.04492041	0.09820568
	0.29213846	0.05257477	0.09836867	0.49920341	0.04157971	0.09893767	0.52759466	0.04771394	0.09861182
	0.5482962	0.04485832	0.09855088	0.3112439	0.05201766	0.09511087	0.44726509	0.05092111	0.09673483
	0.3757233	0.04785516	0.09323131	0.53475026	0.04837667	0.09898575	0.20302145	0.05144659	0.09079777
Mejor	0.55531806	0.05493566	0.13381499	0.56846422	0.05201766	0.10155157	0.66112025	0.06174495	0.10021026
Peor	0.27378002	0.0355509	0.09323131	0.28919387	0.0384052	0.09345679	0.20302145	0.03447956	0.09079777
Promedio	0.44488931	0.04776884	0.099937	0.47529505	0.04642037	0.09801555	0.4674717	0.04850772	0.09695315
Varianza	0.00847871	2.3496E-05	7.539E-05	0.00772078	1.3138E-05	5.0017E-06	0.00977301	3.128E-05	8.3771E-06
D Estándar	0.09207991	0.00484724	0.00868274	0.08786795	0.00362466	0.00223645	0.09885856	0.00559285	0.00289432

Tabla 7. Resultados de la Similitud de las 1027 Instancias con 100, 200 y 500 iteraciones



Numero de iteraciones

Figura 17 Comparativo de similitud entre las mejores soluciones encontradas con 1027 servicios

	100			200			500		
	Mejor	**Peor**	**Promedio**	**Mejor**	**Peor**	**Promedio**	**Mejor**	**Peor**	**Promedio**
	10	24	17.1	2	21	9.2	5	22	11.7
	4	24	12.5	4	23	10.9	6	25	15.6
	4	25	17.9	2	24	13.8	4	25	15.8
	2	24	13.7	2	25	12.8	5	23	16.9
	7	24	16.1	4	19	13.2	3	21	11.5
	9	25	17.7	2	25	13.3	2	23	14.2
	3	23	10.9	2	24	13.2	2	25	15.9
	2	24	14.6	3	25	14.9	6	25	15.9
	2	23	14.8	4	25	17	4	25	13.6
	2	24	13.9	4	24	18.1	5	21	13.6
	7	24	16.3	4	25	14.1	5	25	13.5
	2	25	12.9	2	23	11.2	2	25	13.5
	4	24	11.2	3	19	12.6	6	25	13.8
	3	23	12.8	4	21	14	5	25	13.6
	7	25	17.4	2	22	9.9	3	20	10.7
	3	20	11.6	4	25	14.9	2	21	11.2
	2	23	12.4	2	25	13.2	3	23	12.6
	3	21	10.7	4	25	15.9	5	22	11.8
	6	24	14.1	3	25	12.2	2	23	14.3
	4	25	13.9	4	25	18.4	6	18	11.7
Mejor	2	20	10.7	2	19	9.2	2	18	10.7
Peor	10	25	17.9	4	25	18.4	6	25	16.9
Promedio	4.31578947	23.6315789	14.1368421	3	23.4210526	13.3894737	3.94736842	23.3684211	13.6684211
Varianza	6.22105263	1.69473684	5.37565789	0.89210526	4.15789474	6.01094737	2.36578947	4.41052632	3.22326316
D Estándar	2.49420381	1.30182059	2.3185465	0.94451324	2.03909165	2.45172335	1.53811231	2.10012531	1.79534486

Tabla 8. Generación de los Grupos de las 50 Instancias con 100, 200 y 500 iteraciones

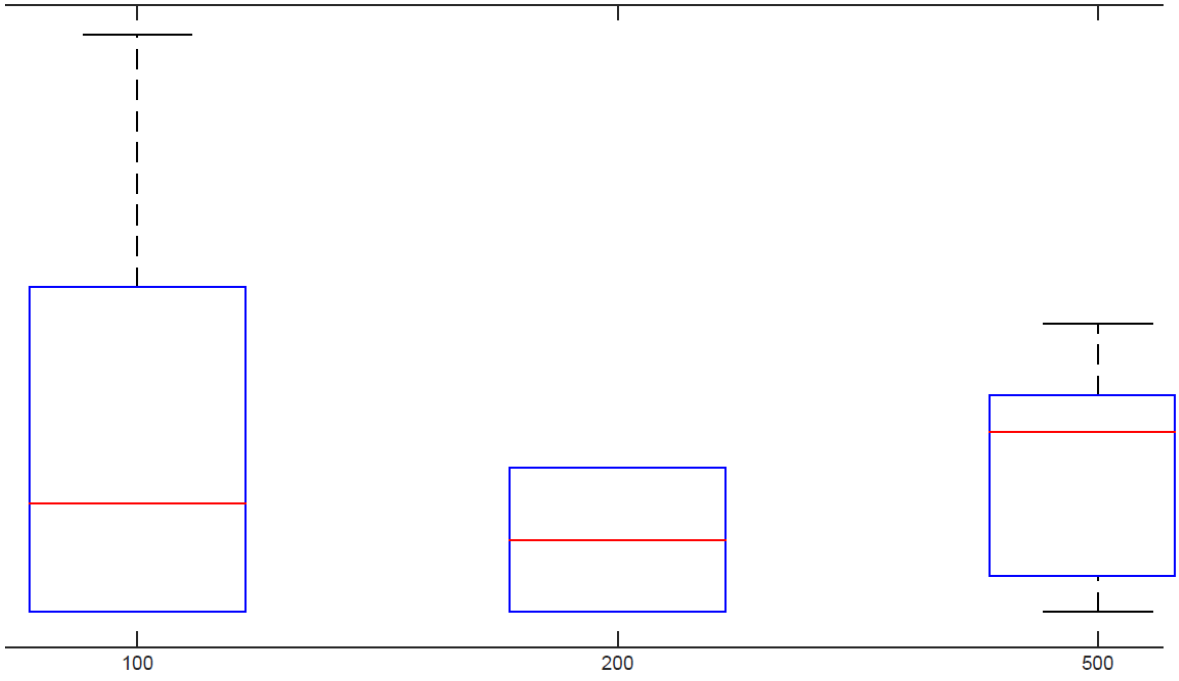
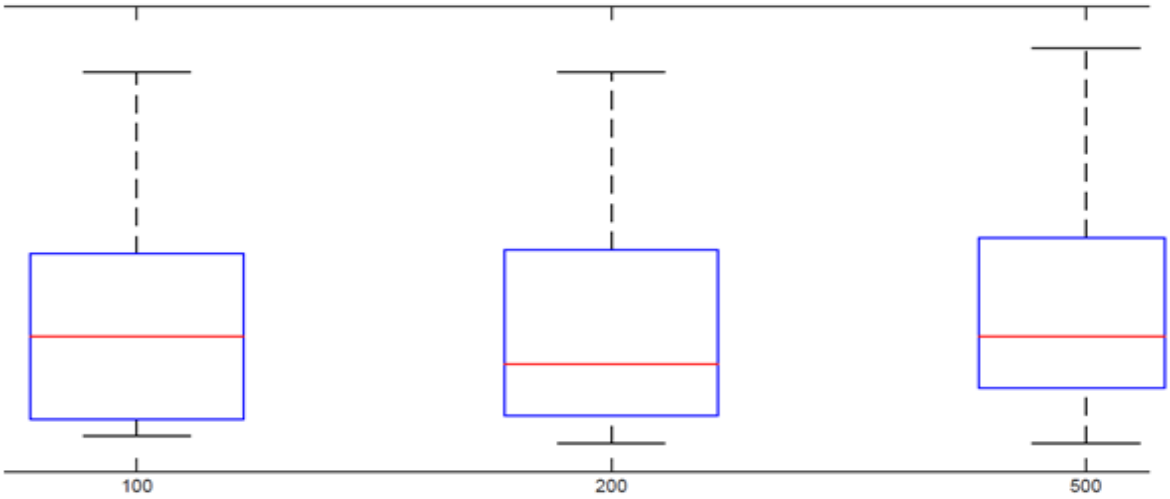


Figura 18. Comparativo del número de grupos en el mejor caso para 50 servicios

	100			200			500		
	Mejor	**Peor**	**Promedio**	**Mejor**	**Peor**	**Promedio**	**Mejor**	**Peor**	**Promedio**
	22	233	103.1	2	300	125	2	307	121.6
	4	294	121.3	7	320	164.8	52	317	215.6
	10	316	135	5	310	157.2	11	248	130.4
	39	278	204.1	5	296	162.5	7	303	174.5
	49	317	184.4	7	302	146	73	295	188.4
	7	234	107.9	49	322	149.4	65	277	173.3
	8	275	136.7	17	311	119.8	17	195	95.1
	20	300	157.4	29	289	122.8	35	313	161.2
	15	309	177.4	6	312	186.3	11	289	165.3
	18	314	204.6	67	315	203.7	18	278	103.1
	33	309	168.9	5	313	134.2	9	277	168.5
	18	318	197.7	24	256	143.5	16	259	117
	3	302	152.3	6	274	171.8	6	238	146.3
	30	230	115.4	16	276	162.6	19	256	130.3
	49	289	166	5	281	135.1	15	261	146.8
	6	295	185.1	12	323	212.1	8	296	152.8
	4	321	115.4	24	309	160.7	47	324	176
	4	291	123.7	38	249	163.4	9	291	152.3
	4	312	96.6	36	283	163.3	21	305	181.4
	16	323	197.2	12	294	147.4	9	291	152
Mejor	3	230	96.6	2	249	119.8	2	195	95.1
Peor	49	323	204.6	67	323	212.1	73	324	215.6
Promedio	18.0526316	291.421053	150.157895	18.9473684	296.894737	157.063158	23.2105263	280.473684	152.626316
Varianza	221.207895	867.473684	1329.45568	305.831579	455.671053	607.254316	426.368421	972.315789	898.558395
D Estándar	14.8730594	29.4529062	36.4617016	17.488041	21.3464529	24.6425306	20.6486905	31.1819786	29.9759636

Tabla 9. Generación de los Grupos de las 647 Instancias con 100, 200 y 500 iteraciones

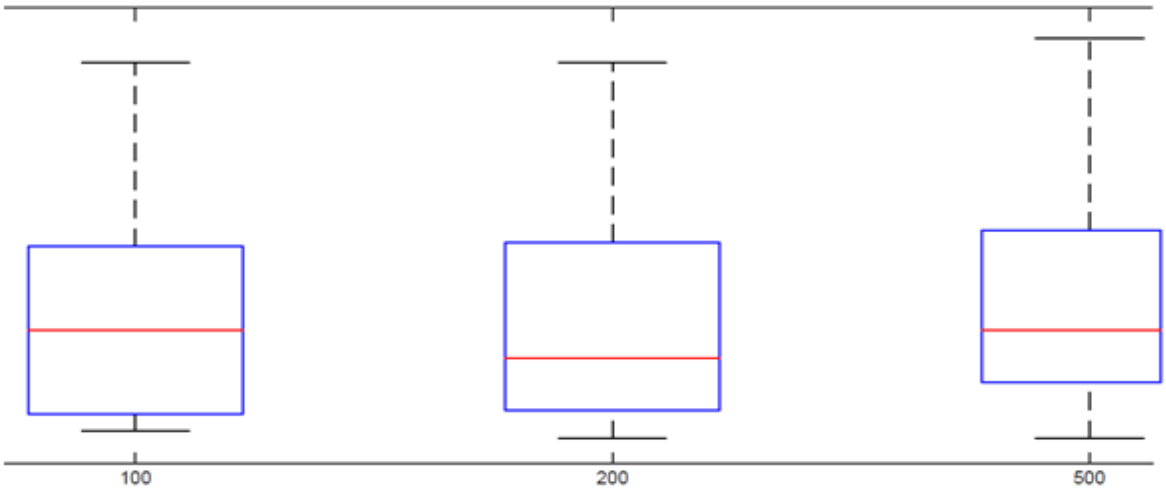


Número de iteraciones

Figura 19. Comparativo del número de grupos en el mejor caso para 647 servicios

	100			200			500		
	Mejor	**Peor**	**Promedio**	**Mejor**	**Peor**	**Promedio**	**Mejor**	**Peor**	**Promedio**
	11	486	220.7	11	527	213.1	30	527	319.7
	18	492	213.3	5	492	270.1	11	486	292.5
	14	530	239.1	17	513	265.6	10	520	316.6
	6	455	210.9	19	517	190.6	2	387	181.2
	26	487	255.2	36	463	269.6	16	525	311.2
	22	485	257.3	23	506	294	40	490	301.5
	110	510	378.8	5	389	229.4	72	452	306.1
	2	418	183.8	49	510	256.9	123	472	340.4
	42	460	309.1	4	498	187.9	10	517	247.8
	15	416	199.8	52	505	268.8	39	378	277.1
	38	511	283.9	24	530	244.8	20	466	248.5
	16	520	262.6	41	501	362.6	37	528	264.2
	2	463	258.4	8	532	274.1	19	513	270.7
	6	496	258.2	7	533	258.4	56	524	318.5
	27	518	223.1	6	494	246.3	27	514	303.1
	106	523	312.9	22	491	250.4	23	484	220.2
	75	522	315	12	531	178	3	517	208.8
	52	495	239.6	10	474	171.6	15	434	272.4
	6	534	278.6	69	499	253.3	36	505	279.1
	61	509	325.6	2	411	194.3	121	505	284.8
Mejor	2	416	183.8	2	389	171.6	2	378	181.2
Peor	110	534	378.8	69	533	362.6	123	528	340.4
Promedio	31.2631579	490.578947	257.910526	22.1052632	500.263158	246.605263	31	486.263158	277.873684
Varianza	1066.82895	1164.15789	2387.30471	355.884211	1444.37895	2085.88832	1179.21053	1980.8	1665.87432
D Estándar	32.6623475	34.1197581	48.8600523	18.8648936	38.0049858	45.6715263	34.3396349	44.5061793	40.8151236

Tabla 10. Generación de los Grupos de las 1027 Instancias con 100, 200 y 500 iteraciones



Número de iteraciones

Figura 20. Comparativo del número de grupos en el mejor caso para 1027 servicios

	Mejor	Peor	Promedio	Varianza	Desviación
50-100	0.58936208	0.29274988	0.45901481	0.00593354	0.0770295
50-200	0.54853117	0.44904535	0.50052478	0.00088301	0.02971549
50-500	0.57163535	0.46554965	0.51820976	0.00090081	0.03001343
647-100	0.54947584	0.2153186	0.44401451	0.01027133	0.10134758
647-200	0.54778745	0.20257995	0.40923046	0.01554565	0.12468218
647-500	0.54881541	0.22262872	0.41911288	0.01280278	0.11314939
1027-100	0.55531806	0.27378002	0.44488931	0.00847871	0.09207991
1027-200	0.56846422	0.28919387	0.47529505	0.00772078	0.08786795
1027-500	0.66112025	0.20302145	0.4674717	0.00977301	0.09885856

Tabla 11. Resumen con los mejores resultados de la Similitud

	Mejor	Peor	Promedio	Varianza	Desviación
50-100	2	10	4.31578947	6.22105263	2.49420381
50-200	2	4	3	0.89210526	0.94451324
50-500	2	6	3.94736842	2.36578947	1.53811231
647-100	3	49	18.0526316	221.207895	14.8730594
647-200	2	67	18.9473684	305.831579	17.488041
647-500	2	73	23.2105263	426.368421	20.6486905
1027-100	2	110	31.2631579	1066.82895	32.6623475
1027-200	2	69	22.1052632	355.884211	18.8648936
1027-500	2	123	31	1179.21053	34.3396349

Tabla 12. Resumen con los mejores resultados de los Grupos

Todos los datos contenidos en las tablas anteriores son información que se obtuvieron de una experimentación exhausta la cual se puede analizar en el Anexo C con los datos completos para un análisis más detallado.

9 Conclusión

En este Proyecto de Integración se diseño e implemento un algoritmo hibrido basado en K-Means, Concenso y en ABC; el cual permitió la clasificación de servicios web.

Cabe mencionar que se diseño e implemento un modulo para adquisición, extracción, filtrado y manejo de datos, este modulo emplea pruebas de hipótesis para dichos propósitos.

El repositorio que contiene la clasificación de los servicios web sirvió para evaluar la comparción de los resultados obtenidos mediante el hibrido, dichos datos se muestran a continuación:

	50	647	1027
100	4.31578947	18.0526316	31.2631579
200	3	18.9473684	22.1052632
500	3.94736842	23.2105263	31

Tabla 13. Valores Promedios de Grupos generados

	50	647	1027
100	6.22105263	221.207895	1066.82895
200	0.89210526	305.831579	355.884211
500	2.36578947	426.368421	1179.21053

Tabla 14. Valores de Varianza

Con los valores de las Tablas 13 y Tabla 14 se realizaron pruebas de hipótesis para determinar si el número de grupos formados para cada instancia en base al número de corridas, estuvieran en el rango establecido, los resultados se muestran a continuación:

	50	647	1027
100vs200	0.93630521	-0.0106012	0.03641695
200vs500	-1.6756679	-0.036335	-0.0322945
100vs500	0.24755126	-0.0480222	0.00074009

Tabla 15. Hipotesis

	50	647	1027
100vs200	1	1	1
200vs500	1	1	1
100vs500	1	1	1

Tabla 16. Resultados de la Hipotesis

En base a la tabla 16 la generación de grupos con instancias de 50 servicios, el rango fue entre 3 y 4 grupos; para el caso de 647 instancias en el mejor de los casos la generación de grupos fue entre 18 y 23 grupos y para el último caso que fue de 1027 servicios el rango fue entre 22 a 31 grupos generados; esto quiere decir que si bien el algoritmo es bueno en la generación de instancias pequeñas en un tiempo corto, la generación de un número de grupos en rango no muy grande tendría que requerir más ejecuciones del programa para poder tener muy buenos resultados ya que el hibrido resulto ser un algoritmo rápido y eficiente, concluyendo que el algoritmo converge y puede ser adaptado para cualquier problema en el que se requiera realizar una clasificación en menor tiempo.

El algoritmo modificado adaptándolo a cualquier problema como sustento en este Proyecto puede dar pie a más investigación en muchas áreas del conocimiento incluso en pruebas Benchmarks.

10 Anexos

- **Anexo A**
  - **Clase Main**

```
/**
 *
 * @author Ivan Flores Sanchez
 *
 */

public class Main {

	public static void main(String[] args) {
		int n;
		LecturaDatos MatAux = new LecturaDatos();
		String f1 = "Opera1.txt", f2 = "Opera2.txt", f3 = "Opera3.txt", f4 = "Opera4.txt",
			f5 = "Opera5.txt", f6 = "Opera6.txt", f7 = "Opera7.txt", f8 = "Opera8.txt";

		n = MatAux.TamMatriz(f1);

		double[][] Mat1f1 = new double[n][n];
		double[][] Mat2f2 = new double[n][n];
		double[][] Mat3f3 = new double[n][n];
		double[][] Mat4f4 = new double[n][n];
		double[][] Mat5f5 = new double[n][n];
		double[][] Mat6f6 = new double[n][n];
		double[][] Mat7f7 = new double[n][n];
		double[][] Mat8f8 = new double[n][n];

		double[][] MatProm = new double[n][n];
		Calculo valores = new Calculo();
		//Filtro Mat1f1Filtrado = new Filtro();
		int GrupCent = (n * 2);
		int TAMANO = 10;
		int Limite = 10;
		double[][] AlfayBeta = new double[TAMANO][GrupCent + 6];
		double[][] AyB2 = new double[TAMANO][GrupCent + 6];
		double[] A2 = new double[TAMANO];
		double[] B2 = new double[TAMANO];

		Mat1f1 = MatAux.leerMatriz(f1, n);
		Mat2f2 = MatAux.leerMatriz(f2, n);
		Mat3f3 = MatAux.leerMatriz(f3, n);
		Mat4f4 = MatAux.leerMatriz(f4, n);
		Mat5f5 = MatAux.leerMatriz(f5, n);
		Mat6f6 = MatAux.leerMatriz(f6, n);
		Mat7f7 = MatAux.leerMatriz(f7, n);
		Mat8f8 = MatAux.leerMatriz(f8, n);

		//Mat1f1Filtrado.filtMat(Mat1f1, Mat2f2, Mat3f3,
			Mat4f4, Mat5f5, Mat6f6, Mat7f7, Mat8f8);
		MatProm = valores.Prom(Mat1f1, Mat2f2, Mat3f3, Mat4f4,
					Mat5f5, Mat6f6, Mat7f7, Mat8f8);

		//Generacion del Primer bloque de Abejas
		for (int i = 0; i < AlfayBeta.length; i++) {
			AlfayBeta[i] = valores.GeneraGrupos(MatProm);
			for (int j = 0; j < GrupCent + 6; j++)
				AyB2[i][j] = AlfayBeta[i][j];
			B2[i] = AlfayBeta[i][GrupCent];
			A2[i] = AlfayBeta[i][GrupCent + 1];
		}
		A2 = valores.Normalizar(A2);
		B2 = valores.Normalizar(B2);
		B2 = Media(B2, A2);
```

```
for (int i = 0; i < AlfayBeta.length; i++) {
        AlfayBeta[i][GrupCent + 2] = A2[i];
        AyB2[i][GrupCent + 2] = A2[i];
}
for (int i = 0; i < AlfayBeta.length; i++) {
        AlfayBeta[i][GrupCent + 3] = B2[i];
        AyB2[i][GrupCent + 3] = B2[i];
}

//Almacenamiento de la informacion
valores.DF2(AlfayBeta, n);

//Ciclo de las Abejas Obreras y en Espera
for (int I = 0; I < 100; I++) {
        for (int i = 0; i < TAMANO; i++) {
                double max = 0;
                for (int i2 = 0; i2 < TAMANO; i2++)
                        max = Math.max(max, AyB2[i2][GrupCent + 3]);
                if (max == (AyB2[i][GrupCent + 3]))
                        AyB2[i][GrupCent + 5] = 0;
                else
                        AyB2[i] = valores.MejorarV(AyB2[i],     SelecValoriza(AyB2,
n,((double) Math.random() * 32767 / ((double) 32767 + (double) (1)))),  MatProm);
        }
        for (int i = 0; i < AyB2.length; i++) {
                B2[i] = AyB2[i][GrupCent];
                A2[i] = AyB2[i][GrupCent + 1];
        }
        A2 = valores.Normalizar(A2);
        B2 = valores.Normalizar(B2);
        B2 = Media(B2, A2);
        for (int i = 0; i < AlfayBeta.length; i++)
                AyB2[i][GrupCent + 2] = A2[i];
        for (int i = 0; i < AlfayBeta.length; i++)
                AyB2[i][GrupCent + 3] = B2[i];
        for (int i = 0; i < TAMANO; i++) {
                if ((int) AyB2[i][GrupCent + 5] < Limite) {
                        if (AlfayBeta[i][GrupCent + 3] <= AyB2[i][GrupCent + 3])
                                AyB2[i][GrupCent + 5] = 0;
                        else
                                AyB2[i][GrupCent + 5] += 1;
                } else {
                        if (((double) Math.random() * 32767 / ((double) 32767 +
                        (double) (1))) <= 0.5) {
                                AyB2[i] = valores.GeneraGrupos(MatProm);
                                for (int z = 0; z < AyB2.length; z++) {
                                        B2[z] = AyB2[z][GrupCent];
                                        A2[z] = AyB2[z][GrupCent + 1];
                                }
                                A2 = valores.Normalizar(A2);
                                B2 = valores.Normalizar(B2);
                                B2 = Media(B2, A2);
                                for (int zz = 0; zz < AlfayBeta.length; zz++)
                                        AyB2[zz][GrupCent + 2] = A2[zz];
                                for (int zzz = 0; zzz < AlfayBeta.length; zzz++)
                                        AyB2[zzz][GrupCent + 3] = B2[zzz];
                        } else {
                                AyB2[i][GrupCent + 5] = 0;
                                for (int j = 0; j < GrupCent + 6; j++)
                                        AlfayBeta[i][j] = AyB2[i][j];
                        }
                }
        }
        double MAX = 0;
        for (int i = 0; i < TAMANO; i++)
                MAX = Math.max(MAX, AyB2[i][GrupCent + 3]);
        for (int i = 0; i < TAMANO; i++) {
                if (MAX == (AyB2[i][GrupCent + 3]))
```

```
                                        if (AlfayBeta[i][GrupCent + 3] < AyB2[i][GrupCent + 3])
                                                for (int j = 0; j < GrupCent + 6; j++)
                                                        AlfayBeta[i][j] = AyB2[i][j];
                        }
                        valores.DarFormat(AyB2, n);
                        valores.DF2(AlfayBeta, n);
                }
        }
        //Funcion que devuelve un vector dependiendo del valor aleatorio
        public static double[] SelecValoriza(double[][] A, int n, double R1) {
                double ValorAcum = 0;
                double VA = 0;
                int GrupCent2 = (n * 2);
                double[] ValorSelcet = new double[GrupCent2 + 6];

                for (int i = 0; i < A.length; i++) {
                        if (i == 0 || i == (A.length - 1)) {
                                if (i == 0)
                                        if (R1 <= VA + A[i][GrupCent2 + 3])
                                                ValorAcum = A[i][GrupCent2 + 3];
                                if (i == (A.length - 1))
                                        if (VA < R1)
                                                ValorAcum = A[i][GrupCent2 + 3];
                        } else if (VA < R1 && R1 <= VA + A[i][GrupCent2 + 3])
                                ValorAcum = A[i][GrupCent2 + 3];
                        VA = VA + A[i][GrupCent2 + 3];
                }

                for (int i = 0; i < A.length; i++)
                        if (A[i][GrupCent2 + 3] == ValorAcum) {
                                for (int j = 0; j < ValorSelcet.length; j++)
                                        ValorSelcet[j] = A[i][j];
                        }
                return ValorSelcet;
        }
        //Funcion que devuelve el reciproco de un valor
        public static double Inver(double A) {
                if (A == 0)
                        return 0.0;
                else
                        return 1 / A;
        }
        //Funcion que devuelve la Media de dos valores
        public static double[] Media(double[] A, double[] B) {
                double[] V = new double[A.length];

                for (int i = 0; i < A.length; i++)
                        V[i] = (A[i] + B[i]) / 2;

                return V;
        }
}
```

- **Clase calculo**

```
import java.io.FileWriter;
import java.io.PrintWriter;
import java.io.IOException;


public class Calculo {
        //Funcion que devuelve el promedio de las 8 matrices
        public double[][] Prom(double[][] mat1, double[][] mat2, double[][] mat3, double[][] mat4,
                        double[][] mat5, double[][] mat6, double[][] mat7, double[][] mat8) {
                int N = mat1.length;
                double[][] MatrixPom = new double[N][N];
                int[][] MatrixZEROS = new int[N][N];
                MatrixZEROS = ZEROS(mat1, mat2, mat3, mat4, mat5, mat6, mat7, mat8);
```

```
        for (int i = 0; i < N; i++)
                for (int j = 0; j < N; j++)
                        if (((8 - MatrixZEROS[i][j]) == 0))
                                MatrixPom[i][j] = 0;
                        else
                                MatrixPom[i][j] = (mat1[i][j] + mat2[i][j] + mat3[i][j] +
                                mat4[i][j] + mat5[i][j] + mat6[i][j] + mat7[i][j] +
                                mat8[i][j]) / (8 - MatrixZEROS[i][j]);
        return MatrixPom;
}
//Funcion que da formato a las matrices (Coloca zeros a la diagonal de la matriz)
public int[][] ZEROS(double[][] mat1, double[][] mat2, double[][] mat3, double[][] mat4,
                                        double[][] mat5, double[][] mat6, double[][] mat7,
double[][] mat8) {
        int N = mat1.length;
        int[][] Z = new int[N][N];
        int count = 0;

        for (int i = 0; i < N; i++)
                for (int j = 0; j < N; j++) {
                        if (mat1[i][j] == 0.0)
                                count++;
                        if (mat2[i][j] == 0.0)
                                count++;
                        if (mat3[i][j] == 0.0)
                                count++;
                        if (mat4[i][j] == 0.0)
                                count++;
                        if (mat5[i][j] == 0.0)
                                count++;
                        if (mat6[i][j] == 0.0)
                                count++;
                        if (mat7[i][j] == 0.0)
                                count++;
                        if (mat8[i][j] == 0.0)
                                count++;
                        Z[i][j] = count;
                        count = 0;
                }

        return Z;
}
//Funcion que calcula la varianza
public double[][] Var(double[][] mat1, double[][] mat2, double[][] mat3, double[][] mat4,
                  double[][] mat5, double[][] mat6, double[][] mat7, double[][] mat8) {
        int N = mat1.length;
        double[][] MatrixVar = new double[N][N];
        double[][] Media = new double[N][N];
        int M = 8;
        int[][] MatrixZEROS = new int[N][N];
        MatrixZEROS = ZEROS(mat1, mat2, mat3, mat4, mat5, mat6, mat7, mat8);
        Media = Prom(mat1, mat2, mat3, mat4, mat5, mat6, mat7, mat8);

        for (int i = 0; i < N; i++)
                for (int j = 0; j < N; j++)
                        MatrixVar[i][j] = (Math.pow((mat1[i][j] - Media[i][j]), 2)
                                        + Math.pow((mat2[i][j] - Media[i][j]), 2)
                                        + Math.pow((mat3[i][j] - Media[i][j]), 2)
                                        + Math.pow((mat4[i][j] - Media[i][j]), 2)
                                        + Math.pow((mat5[i][j] - Media[i][j]), 2)
                                        + Math.pow((mat6[i][j] - Media[i][j]), 2)
                                        + Math.pow((mat7[i][j] - Media[i][j]), 2)
                                        + Math.pow((mat8[i][j] - Media[i][j]), 2))
                                        / ((M - 1) - MatrixZEROS[i][j]);
        return MatrixVar;
}
//Funcion que Normalizacion el conjunto de vectores
public double[] Normalizar(double[] mat) {
        double Sumatoria = 0;
```

```
        double Normal[] = new double[mat.length];

        for (int i = 0; i < mat.length; i++)
                Sumatoria = Sumatoria + mat[i];
        for (int i = 0; i < mat.length; i++)
                Normal[i] = mat[i] / Sumatoria;

        return Normal;
}
//Funcion que genera los grupos de abejas los centroides alfa, beta, y el grupo maximo
public double[] GeneraGrupos(double[][] mat) {
        double V = mat.length;
        double aux = Math.ceil(V / 2);
                int NumElem = (int) (((double) Math.random() * 32767 / ((double) 32767 +
        (double) (1)))  * (((int) aux) - 2 + 1) + 2);
        int[] VectorN = new int[mat.length];
        int[] VectorAux = new int[mat.length];
        double Mu = 0;
        int[] C = new int[NumElem];
        double D = 0;
        double[] AyB = new double[(mat.length * 2) + 6];
        boolean flag = true;
        double v = 0;

        for (int i = 0; i < VectorN.length; i++)
                VectorN[i] = (int) (((double) Math.random() * 32767 / ((double) 32767 +
                (double) (1))) * NumElem + 1);

        int incremt = 0;
        for (int i = C.length; i > 0; i--) {
                VectorN[incremt] = i;
                incremt = incremt + 2;
        }
        for (int i = 0; i < VectorN.length; i++)
                AyB[i] = VectorN[i];

        for (int i = 0; i < NumElem; i++) {
                while (flag == true) {
                        int Aux = (int) (((double) Math.random() * 32767 / ((double) 32767 +
                        (double) (1)))  * (mat.length) + 0);
                        while ((i + 1) == VectorN[Aux] && flag == true) {
                                VectorAux[Aux] = VectorN[Aux];
                                flag = false;
                        }
                }
                flag = true;
                C[i] = i + 1;
        }

        for (int i = 0; i < mat.length; i++)
                AyB[mat.length + i] = VectorAux[i];
        AyB[(mat.length * 2) + 4] = MAX(AyB, VectorN.length);

        for (int i = 0; i < VectorAux.length; i++) {
                if (VectorAux[i] != 0)
                        for (int j = 0; j < VectorN.length; j++)
                                if (VectorAux[i] == VectorN[j]) {
                                        if (j == i)
                                                v = 0;
                                        else
                                                v = mat[i][j];
                                        Mu = Mu + v;
                                }
        }
        if (Mu == 0.0)
                AyB[mat.length * 2] = Mu;
        else
                AyB[mat.length * 2] = 1 / Mu;
        for (int i = 0; i < mat.length - 1; i++) {
```

```
                for (int j = i + 1; j < mat.length; j++) {
                        if (VectorAux[i] != 0)
                                if (VectorAux[j] != 0)
                                        D = D + Math.max(mat[i][j], mat[j][i]);
                }
        }
        if (D == 0.0)
                AyB[(mat.length * 2) + 1] = D;
        else
                AyB[(mat.length * 2) + 1] = 1 / D;
        return AyB;
}
//Funcion que detecta el grupo maximo generado
double MAX(double[] M, int l) {
        double Max = 0;
        for (int i = 0; i < l; i++)
                Max = Math.max(Max, M[i]);
        return Max;
}
//Funcion que genera la nuevas fuente de alimentos
double[] MejorarV(double[] Vi, double[] V2, double[][] Promedio) {
        double[] Mejorado = new double[V2.length];
        double[] Aux = new double[Promedio.length];
        int N = Promedio.length;
        double fi = 0.8;
        boolean flag1 = true;
        boolean flag3 = true;

        for (int i = 0; i < V2.length; i++)
                Mejorado[i] = Vi[i];

        for (int i = 0; i < Promedio.length; i++)
                Aux[i] = (Math.rint(Math.abs(Vi[Promedio.length] + fi * (Vi[Promedio.length]
                - V2[Promedio.length])))) % Vi[(Promedio.length * 2) + 4];

        for (int i = 0; i < Promedio.length - 1; i++) {
                for (int j = i + 1; j < Promedio.length; j++) {
                        if (Aux[i] == Aux[j])
                                Aux[j] = 0.0;
                }
        }

        double G = Vi[Vi.length - 2];

        while (flag1 == true) {
                int P = (int) (((double) Math.random() * 32767 / ((double) 32767 + (double)
                (1))) * ((Aux.length - 1) - 0 + 1) + 0);
                if (Aux[P] == 0.0) {
                        Aux[P] = G;
                        flag1 = false;
                }
        }

        double X = 1.0;
        while (X < G) {
                for (int i = 0; i < Aux.length; i++) {
                        if (Aux[i] == X)
                                flag3 = false;
                }
                while (flag3 == true) {
                        int P = (int) (((double) Math.random() * 32767 / ((double) 32767 +
                        (double) (1))) * ((Aux.length - 1) - 0 + 1) + 0);
                        if (Aux[P] == 0.0) {
                                Aux[P] = X;
                                flag3 = false;
                        }
                }
                flag3 = true;
                X = X + 1;
```

```
        }

        for (int i = 0; i < Aux.length; i++)
                Mejorado[N++] = Aux[i];

        double v = 0;
        double Mu = 0;
        for (int i = 0; i < Aux.length; i++) {
                if (Mejorado[Promedio.length + i] != 0)
                        for (int j = 0; j < Aux.length; j++)
                                if (Mejorado[Promedio.length + i] == Mejorado[j]) {
                                        if (j == i)
                                                v = 0;
                                        else
                                                v = Promedio[i][j];
                                        Mu = Mu + v;
                                }
        }
        if (Mu == 0.0)
                Mejorado[Promedio.length * 2] = Mu;
        else
                Mejorado[Promedio.length * 2] = 1 / Mu;

        double D = 0;
        for (int i = 0; i < Aux.length - 1; i++) {
                for (int j = i + 1; j < Aux.length; j++) {
                        if (Mejorado[Promedio.length + i] != 0)
                                if (Mejorado[Promedio.length + j] != 0)
                                        D = D + Math.max(Promedio[i][j], Promedio[j][i]);
                }
        }
        if (D == 0.0)
                Mejorado[(Promedio.length * 2) + 1] = D;
        else
                Mejorado[(Promedio.length * 2) + 1] = 1 / D;
        return Mejorado;
}
//Funcion para almacenar la similitud
public void DarFormat(double Vect[][], int n) {
        try {
                FileWriter w = new FileWriter("Grupos.txt", true);
                PrintWriter writer = new PrintWriter(w);

                for (int i = 0; i < Vect.length; i++) {
                        for (int j = 0; j < n; j++) { // 1er Vector
                                writer.print((int) Vect[i][j]);
                                System.out.print((int) Vect[i][j]);
                        }
                        writer.print(" ");
                        writer.print(" * ");
                        writer.print(" ");
                        System.out.print(" ");
                        System.out.print(" * ");
                        System.out.print(" ");

                        for (int j = 0; j < n; j++) {// 2do Vector
                                writer.print((int) Vect[i][j + n]);
                                System.out.print((int) Vect[i][j + n]);
                        }
                        writer.print(" ");
                        writer.print(" * ");
                        writer.print(" ");
                        System.out.print(" ");
                        System.out.print(" * ");
                        System.out.print(" ");

                        for (int j = 0; j < 1; j++) {// Beta
                                writer.print(Vect[i][j + (n * 2)]);
                                System.out.print(Vect[i][j + (n * 2)]);
```

```
                        }
                        writer.print(" ");
                        writer.print(" * ");
                        writer.print(" ");
                        System.out.print(" ");
                        System.out.print(" * ");
                        System.out.print(" ");

                        for (int j = 0; j < 1; j++) {// Alfa
                                writer.print(Vect[i][j + ((n * 2) + 1)]);
                                System.out.print(Vect[i][j + ((n * 2) + 1)]);
                        }
                        writer.print(" ");
                        writer.print(" * ");
                        writer.print(" ");
                        System.out.print(" ");
                        System.out.print(" * ");
                        System.out.print(" ");

                        for (int j = 0; j < 1; j++) {// Normalizacion
                                writer.print(Vect[i][j + ((n * 2) + 2)]);
                                System.out.print(Vect[i][j + ((n * 2) + 2)]);
                        }
                        writer.print(" ");
                        writer.print(" * ");
                        writer.print(" ");
                        System.out.print(" ");
                        System.out.print(" * ");
                        System.out.print(" ");

                        for (int j = 0; j < 1; j++) {// Valoracion
                                writer.print(Vect[i][j + ((n * 2) + 3)]);
                                System.out.print(Vect[i][j + ((n * 2) + 3)]);
                        }
                        writer.print(" ");
                        writer.print(" * ");
                        writer.print(" ");
                        System.out.print(" ");
                        System.out.print(" * ");
                        System.out.print(" ");

                        for (int j = 0; j < 1; j++) {// # Grupos
                                writer.print((int) Vect[i][j + ((n * 2) + 4)]);
                                System.out.print((int) Vect[i][j + ((n * 2) + 4)]);
                        }
                        writer.print(" ");
                        writer.print(" * ");
                        writer.print(" ");
                        System.out.print(" ");
                        System.out.print(" * ");
                        System.out.print(" ");

                        for (int j = 0; j < 1; j++) {// # Limite
                                writer.print((int) Vect[i][j + ((n * 2) + 5)]);
                                System.out.print((int) Vect[i][j + ((n * 2) + 5)]);
                        }
                        writer.println(" ");
                        System.out.println(" ");
                }
                writer.println(" ");
                writer.println(" ");
                System.out.println(" ");
                System.out.println(" ");

                writer.close();
        } catch (IOException e) {
                e.printStackTrace();
        }
}
```

```java
//Funcion para almacenar la similitud
public void DF2(double Vect[][], int n) {
	try {
		FileWriter w = new FileWriter("Grupos.txt", true);
		PrintWriter writer = new PrintWriter(w);

		for (int i = 0; i < Vect.length; i++) {
			for (int j = 0; j < n; j++) { // 1er Vector
				writer.print((int) Vect[i][j]);
				System.out.print((int) Vect[i][j]);
			}
			writer.print(" ");
			writer.print(" - ");
			writer.print(" ");
			System.out.print(" ");
			System.out.print(" - ");
			System.out.print(" ");

			for (int j = 0; j < n; j++) {// 2do Vector
				writer.print((int) Vect[i][j + n]);
				System.out.print((int) Vect[i][j + n]);
			}
			writer.print(" ");
			writer.print(" - ");
			writer.print(" ");
			System.out.print(" ");
			System.out.print(" - ");
			System.out.print(" ");

			for (int j = 0; j < 1; j++) {// Beta
				writer.print(Vect[i][j + (n * 2)]);
				System.out.print(Vect[i][j + (n * 2)]);
			}
			writer.print(" ");
			writer.print(" - ");
			writer.print(" ");
			System.out.print(" ");
			System.out.print(" - ");
			System.out.print(" ");

			for (int j = 0; j < 1; j++) {// Alfa
				writer.print(Vect[i][j + ((n * 2) + 1)]);
				System.out.print(Vect[i][j + ((n * 2) + 1)]);
			}
			writer.print(" ");
			writer.print(" - ");
			writer.print(" ");
			System.out.print(" ");
			System.out.print(" - ");
			System.out.print(" ");

			for (int j = 0; j < 1; j++) {// Normalizacion
				writer.print(Vect[i][j + ((n * 2) + 2)]);
				System.out.print(Vect[i][j + ((n * 2) + 2)]);
			}
			writer.print(" ");
			writer.print(" - ");
			writer.print(" ");
			System.out.print(" ");
			System.out.print(" - ");
			System.out.print(" ");

			for (int j = 0; j < 1; j++) {// Valoracion
				writer.print(Vect[i][j + ((n * 2) + 3)]);
				System.out.print(Vect[i][j + ((n * 2) + 3)]);
			}
			writer.print(" ");
			writer.print(" - ");
			writer.print(" ");
```

```
                                System.out.print(" ");
                                System.out.print(" - ");
                                System.out.print(" ");

                                for (int j = 0; j < 1; j++) {// # Grupos
                                        writer.print((int) Vect[i][j + ((n * 2) + 4)]);
                                        System.out.print((int) Vect[i][j + ((n * 2) + 4)]);
                                }
                                writer.print(" ");
                                writer.print(" - ");
                                writer.print(" ");
                                System.out.print(" ");
                                System.out.print(" - ");
                                System.out.print(" ");

                                for (int j = 0; j < 1; j++) {// # Limite
                                        writer.print((int) Vect[i][j + ((n * 2) + 5)]);
                                        System.out.print((int) Vect[i][j + ((n * 2) + 5)]);
                                }
                                writer.println(" ");
                                System.out.println(" ");
                        }
                        writer.println(" ");
                        writer.println(" ");
                        System.out.println(" ");
                        System.out.println(" ");

                        writer.close();
                } catch (IOException e) {
                        e.printStackTrace();
                }
        }
}
```

- **Clase Filtro**

```
import java.io.PrintWriter;
import java.io.IOException;

public class Filtro {
        //Funcion para Filtrar los archivos de entrada
        public void filtMat(double[][] mat1, double[][] mat2, double[][] mat3, double[][] mat4,
                                double[][] mat5, double[][] mat6, double[][] mat7, double[][]
mat8) {
                int N = 8;
                int n = mat1.length;
                double cte = 1.96;

                double[][] MatPom = new double[n][n];
                double[][] MatVar = new double[n][n];
                double[][] desEst = new double[n][n];

                for (int i = 0; i < n; i++)
                        for (int j = 0; j < n; j++) {
                                MatPom[i][j] = (mat1[i][j] + mat2[i][j] + mat3[i][j] + mat4[i][j] +
                                mat5[i][j] + mat6[i][j] + mat7[i][j] + mat8[i][j]) / N;

                                MatVar[i][j] = (Math.pow((mat1[i][j] - MatPom[i][j]), 2)
                                                + Math.pow((mat2[i][j] - MatPom[i][j]), 2)
                                                + Math.pow((mat3[i][j] - MatPom[i][j]), 2)
                                                + Math.pow((mat4[i][j] - MatPom[i][j]), 2)
                                                + Math.pow((mat5[i][j] - MatPom[i][j]), 2)
                                                + Math.pow((mat6[i][j] - MatPom[i][j]), 2)
                                                + Math.pow((mat7[i][j] - MatPom[i][j]), 2)
                                                + Math.pow((mat8[i][j] - MatPom[i][j]), 2))/(N - 1);

                                desEst[i][j] = Math.sqrt(MatVar[i][j]);
                        }
```

```
                try {
                        PrintWriter writer1 = new PrintWriter("Opera1.txt", "UTF-8");
                        for (int i = 0; i < n; i++) {
                                for (int j = 0; j < n; j++) {
                                        if (MatPom[i][j]
                                - (cte * (desEst[i][j]) / Math.sqrt(mat1.length)) <= mat1[i][j] &&
        mat1[i][j] <= MatPom[i][j] + (cte * (desEst[i][j]) / Math.sqrt(mat1.length))) {
                                                System.out.print(mat1[i][j]);
                                                System.out.print(" ");
                                                writer1.print(mat1[i][j]);
                                                writer1.print(" ");
                                        } else {
                                                System.out.print("*  CAMBIO");
                                                System.out.print(" ");
                                                writer1.print("0.0");
                                                writer1.print(" ");
                                        }
                                }
                                writer1.println(" ");
                                System.out.println();
                        }
                        writer1.close();
                        System.out.println();

                        PrintWriter writer2 = new PrintWriter("Opera2.txt", "UTF-8");
                        for (int i = 0; i < n; i++) {
                                for (int j = 0; j < n; j++) {
                                        if (MatPom[i][j]
                - (cte * (desEst[i][j]) / Math.sqrt(mat1.length)) <= mat2[i][j] && mat2[i][j] <=
MatPom[i][j] + (cte * (desEst[i][j]) / Math.sqrt(mat1.length))) {
                                                System.out.print(mat2[i][j]);
                                                System.out.print(" ");
                                                writer2.print(mat2[i][j]);
                                                writer2.print(" ");
                                        } else {
                                                System.out.print("*  CAMBIO");
                                                System.out.print(" ");
                                                writer2.print("0.0");
                                                writer2.print(" ");
                                        }
                                }
                                writer2.println(" ");
                                System.out.println();
                        }
                        writer2.close();
                        System.out.println();

                        PrintWriter writer3 = new PrintWriter("Opera3.txt", "UTF-8");
                        for (int i = 0; i < n; i++) {
                                for (int j = 0; j < n; j++) {
                                        if (MatPom[i][j]
                        - (cte * (desEst[i][j]) / Math.sqrt(mat1.length)) <= mat3[i][j] &&
mat3[i][j] <= MatPom[i][j] + (cte * (desEst[i][j]) / Math.sqrt(mat1.length))) {
                                                System.out.print(mat3[i][j]);
                                                System.out.print(" ");
                                                writer3.print(mat3[i][j]);
                                                writer3.print(" ");
                                        } else {
                                                System.out.print("*  CAMBIO");
                                                System.out.print(" ");
                                                writer3.print("0.0");
                                                writer3.print(" ");
                                        }
                                }
                                writer3.println(" ");
                                System.out.println();
                        }
                        writer3.close();
```

```
                        System.out.println();

                        PrintWriter writer4 = new PrintWriter("Opera4.txt", "UTF-8");
                        for (int i = 0; i < n; i++) {
                                for (int j = 0; j < n; j++) {
                                        if (MatPom[i][j]
                        - (cte * (desEst[i][j]) / Math.sqrt(mat1.length)) <= mat4[i][j] &&
mat4[i][j] <= MatPom[i][j] + (cte * (desEst[i][j]) / Math.sqrt(mat1.length))) {
                                                System.out.print(mat4[i][j]);
                                                System.out.print(" ");
                                                writer4.print(mat4[i][j]);
                                                writer4.print(" ");
                                        } else {
                                                System.out.print("*  CAMBIO");
                                                System.out.print(" ");
                                                writer4.print("0.0");
                                                writer4.print(" ");
                                        }
                                }
                                writer4.println(" ");
                                System.out.println();
                        }
                        writer4.close();
                        System.out.println();

                        PrintWriter writer5 = new PrintWriter("Opera5.txt", "UTF-8");
                        for (int i = 0; i < n; i++) {
                                for (int j = 0; j < n; j++) {
                                        if (MatPom[i][j]
                        - (cte * (desEst[i][j]) / Math.sqrt(mat1.length)) <= mat5[i][j] &&
mat5[i][j] <= MatPom[i][j] + (cte * (desEst[i][j]) / Math.sqrt(mat1.length))) {
                                                System.out.print(mat5[i][j]);
                                                System.out.print(" ");
                                                writer5.print(mat5[i][j]);
                                                writer5.print(" ");
                                        } else {
                                                System.out.print("*  CAMBIO");
                                                System.out.print(" ");
                                                writer5.print("0.0");
                                                writer5.print(" ");
                                        }
                                }
                                writer5.println(" ");
                                System.out.println();
                        }
                        writer5.close();
                        System.out.println();

                        PrintWriter writer6 = new PrintWriter("Opera6.txt", "UTF-8");
                        for (int i = 0; i < n; i++) {
                                for (int j = 0; j < n; j++) {
                                        if (MatPom[i][j]
                        - (cte * (desEst[i][j]) / Math.sqrt(mat1.length)) <= mat6[i][j] &&
mat6[i][j] <= MatPom[i][j] + (cte * (desEst[i][j]) / Math.sqrt(mat1.length))) {
                                                System.out.print(mat6[i][j]);
                                                System.out.print(" ");
                                                writer6.print(mat6[i][j]);
                                                writer6.print(" ");
                                        } else {
                                                System.out.print("*  CAMBIO");
                                                System.out.print(" ");
                                                writer6.print("0.0");
                                                writer6.print(" ");
                                        }
                                }
                                writer6.println(" ");
                                System.out.println();
                        }
                        writer6.close();
```

```
                        System.out.println();

                        PrintWriter writer7 = new PrintWriter("Opera7.txt", "UTF-8");
                        for (int i = 0; i < n; i++) {
                                for (int j = 0; j < n; j++) {
                                        if (MatPom[i][j]
                        - (cte * (desEst[i][j]) / Math.sqrt(mat1.length)) <= mat7[i][j] &&
mat7[i][j] <= MatPom[i][j] + (cte * (desEst[i][j]) / Math.sqrt(mat1.length))) {
                                                System.out.print(mat7[i][j]);
                                                System.out.print(" ");
                                                writer7.print(mat7[i][j]);
                                                writer7.print(" ");
                                        } else {
                                                System.out.print("*  CAMBIO");
                                                System.out.print(" ");
                                                writer7.print("0.0");
                                                writer7.print(" ");
                                        }
                                }
                                writer7.println(" ");
                                System.out.println();
                        }
                        writer7.close();
                        System.out.println();

                        PrintWriter writer8 = new PrintWriter("Opera8.txt", "UTF-8");
                        for (int i = 0; i < n; i++) {
                                for (int j = 0; j < n; j++) {
                                        if (MatPom[i][j]
                                        - (cte * (desEst[i][j]) / Math.sqrt(mat1.length)) <= mat8[i][j] &&
mat8[i][j] <= MatPom[i][j] + (cte * (desEst[i][j]) / Math.sqrt(mat1.length))) {
                                                System.out.print(mat8[i][j]);
                                                System.out.print(" ");
                                                writer8.print(mat8[i][j]);
                                                writer8.print(" ");
                                        } else {
                                                System.out.print("*  CAMBIO");
                                                System.out.print(" ");
                                                writer8.print("0.0");
                                                writer8.print(" ");
                                        }
                                }
                                writer8.println(" ");
                                System.out.println();
                        }
                        writer8.close();
                        System.out.println();
                } catch (IOException e) {
                        e.printStackTrace();
                }
        }
}
```

- **Clase LecturaDatos**

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Locale;
import java.util.Scanner;


public class LecturaDatos {
        //Funcion que devuelve el tamaño de las 8 Matrices
        public int TamMatriz(String Nom) {
                File f = new File(Nom);
                int count = 0;
                int T;
```

```
                Scanner S;
                try {
                        S = new Scanner(f);
                        S.useLocale(Locale.ENGLISH);
                        while (S.hasNextDouble()) {
                                double linea = S.nextDouble();
                                count++;
                        }
                        S.close();
                } catch (FileNotFoundException e) {
                        e.printStackTrace();
                }
                return T = (int) Math.sqrt(count);
        }
        //Funcion para obtener los datos de los archivos de entrada
        public double[][] leerMatriz(String Nom, int Tam) {
                File f = new File(Nom);
                double[][] matriz = new double[Tam][Tam];
                Scanner s;
                try {
                        s = new Scanner(f);
                        s.useLocale(Locale.ENGLISH);
                        int j = 0, k = 0;
                        while (s.hasNextDouble()) {
                                double linea = s.nextDouble();
                                if (k < matriz.length)
                                        matriz[j][k++] = linea;
                                else {
                                        j++;
                                        k = 0;
                                        matriz[j][k] = linea;
                                        k++;
                                }
                        }
                        s.close();
                } catch (FileNotFoundException e) {
                        e.printStackTrace();
                }
                return matriz;
        }
}
```

- **Anexo B**
  - **Clase Main**

```
public class Main {
        public static void main(String[] args) {
        String localDir = "C:/Users/ivan/workspace/Metricas";
        String ontologyFile = "C:/Users/ivan/workspace/Metricas/X.owl";
        String ontologyIRI = "http://www.semanticweb.org/ontologies/2012/8/templateWSDL10.owl";

                OntologyManagement manager = new OntologyManagement();
                Operation[] serviceOper = manager.LoadOntology(localDir, ontologyFile, ontologyIRI);

                Metrics med = new Metrics();

                int n = serviceOper.length;
                int arraySize = ((n * n) - n) / 2;

                SimilarityObject[] WuPalmerSimilarityArray = new SimilarityObject[arraySize];
                SimilarityObject[] LinSimilarityArray = new SimilarityObject[arraySize];
                SimilarityObject[] PathSimilarityArray = new SimilarityObject[arraySize];
                SimilarityObject[] LeskSimilarityArray = new SimilarityObject[arraySize];
                SimilarityObject[] HirstStOngemilarityArray = new SimilarityObject[arraySize];
                SimilarityObject[] JiangConrathmilarityArray = new SimilarityObject[arraySize];
                SimilarityObject[] LeacockChodorowmilarityArray = new SimilarityObject[arraySize];
                SimilarityObject[] ResnikmilarityArray = new SimilarityObject[arraySize];

                int cont = 0;
                for (int i = 0; i < serviceOper.length; i++)
                        for (int j = i + 1; j < serviceOper.length; j++) {
                                System.out.println("NUMERO ==> " + cont);
                                System.out.println("SimilarityBetween( "+
serviceOper[i].getOperationName() + " , "       + serviceOper[j].getOperationName() + " ) = ");

                                double[][] semSim = med.semSimWuPalmer(serviceOper[i],
serviceOper[j]);
                                double WuSim = med.matchAverage(semSim);
                                WuPalmerSimilarityArray[cont] = new
SimilarityObject(serviceOper[i].getOperationName(),     serviceOper[j].getOperationName(), WuSim,
"Undefined");

                                System.out.print("==> Wu Palmer Semantic Similarity = ");
                                System.out.println(med.matchAverage(semSim));

                                double[][] semSimL = med.semSimLin(serviceOper[i], serviceOper[j]);
                                double LinSim = med.matchAverage(semSimL);
                                LinSimilarityArray[cont] = new
SimilarityObject(serviceOper[i].getOperationName(), serviceOper[j].getOperationName(), LinSim,
"Undefined");

                                System.out.print("==> Lin Semantic Similarity = ");
                                System.out.println(med.matchAverage(semSimL));

                                double[][] semSimP = med.semSimPath(serviceOper[i],
        serviceOper[j]);
                                double PathSim = med.matchAverage(semSimP);
                                PathSimilarityArray[cont] = new
SimilarityObject(serviceOper[i].getOperationName(),     serviceOper[j].getOperationName(), PathSim,
"Undefined");

                                System.out.print("==> Path Semantic Similarity = ");
                                System.out.println(med.matchAverage(semSimP));

                                double[][] semSimLe = med.semSimLesk(serviceOper[i],
serviceOper[j]);
                                double LeskSim = med.matchAverage(semSimLe);
```

```
                                LeskSimilarityArray[cont] = new
SimilarityObject(serviceOper[i].getOperationName(),     serviceOper[j].getOperationName(), LeskSim,
"Undefined");

                                System.out.print("==> Lesk Semantic Similarity = ");
                                System.out.println(med.matchAverage(semSimLe));

                                double[][] semSimHSO = med.semSimHSO(serviceOper[i],
serviceOper[j]);
                                double HSO = med.matchAverage(semSimHSO);
                                HirstStOngemilarityArray[cont] = new
SimilarityObject(serviceOper[i].getOperationName(), serviceOper[j].getOperationName(), HSO,
"Undefined");

                                System.out.print("==> HirstStOnge Semantic Similarity = ");
                                System.out.println(med.matchAverage(semSimHSO));

                                double[][] semSimJiang = med.semSimJCN(serviceOper[i],
serviceOper[j]);
                                double Jiang = med.matchAverage(semSimJiang);
                                JiangConrathmilarityArray[cont] = new
SimilarityObject(serviceOper[i].getOperationName(), serviceOper[j].getOperationName(), Jiang,
"Undefined");

                                System.out.print("==> JiangConrath Semantic Similarity = ");
                                System.out.println(med.matchAverage(semSimJiang));

                                double[][] semSimLea = med.semSimLCH(serviceOper[i],
serviceOper[j]);
                                double LeaCho = med.matchAverage(semSimLea);
                                LeacockChodorowmilarityArray[cont] = new
SimilarityObject(serviceOper[i].getOperationName(), serviceOper[j].getOperationName(), LeaCho,
"Undefined");

                                System.out.print("==> LeacockChodorow Semantic Similarity = ");
                                System.out.println(med.matchAverage(semSimLea));

                                double[][] semSimREsnik = med.semSimResnik(serviceOper[i],
serviceOper[j]);
                                double REsnik = med.matchAverage(semSimREsnik);
                                ResnikmilarityArray[cont] = new
SimilarityObject(serviceOper[i].getOperationName(), serviceOper[j].getOperationName(), REsnik,
"Undefined");

                                System.out.print("==> Resnik Semantic Similarity = ");
                                System.out.println(med.matchAverage(semSimREsnik));
                                cont++;
                        }
                System.out.println("El tamaño de n es: " + n);
                System.out.println("El tamaño del array es: " + arraySize);

                med.setSimilarityDesition(WuPalmerSimilarityArray, "Opera1");// WuPalmer
                med.setSimilarityDesition(LinSimilarityArray, "Opera2");// Lin
                med.setSimilarityDesition(PathSimilarityArray, "Opera3");// Path
                med.setSimilarityDesition(LeskSimilarityArray, "Opera4");// Lesk
                med.setSimilarityDesition(HirstStOngemilarityArray, "Opera5");// HirstStOnge
                med.setSimilarityDesition(JiangConrathmilarityArray, "Opera6");// JiangConrath
                med.setSimilarityDesition(LeacockChodorowmilarityArray, "Opera7");// LeacockChodorow
                med.setSimilarityDesition(ResnikmilarityArray, "Opera8");// Resnik
                System.out.println("FINAL RESULTS");
        }

        public static double[] getSimArray(SimilarityObject[] array) {
                double[] res = new double[array.length];
                for (int i = 0; i < array.length; i++)
                        res[i] = array[i].getSim();
                return res;
        }
}
```

- **Clase Metricas**

```
import java.io.IOException;
import java.io.PrintWriter;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import edu.cmu.lti.lexical_db.ILexicalDatabase;
import edu.cmu.lti.lexical_db.NictWordNet;
import edu.cmu.lti.ws4j.RelatednessCalculator;
import edu.cmu.lti.ws4j.impl.HirstStOnge;
import edu.cmu.lti.ws4j.impl.JiangConrath;
import edu.cmu.lti.ws4j.impl.LeacockChodorow;
import edu.cmu.lti.ws4j.impl.Lesk;
import edu.cmu.lti.ws4j.impl.Lin;
import edu.cmu.lti.ws4j.impl.Path;
import edu.cmu.lti.ws4j.impl.Resnik;
import edu.cmu.lti.ws4j.impl.WuPalmer;
import edu.cmu.lti.ws4j.util.WS4JConfiguration;

public class Metrics {
        public double[][] semSimWuPalmer(Operation Op1, Operation Op2) {
                double res[][] = null;
                ILexicalDatabase db = new NictWordNet();
                RelatednessCalculator rcs = new WuPalmer(db);
                WS4JConfiguration.getInstance().setMFS(true);
                String[] operationNameOp1 =
lexicalDiscriminant(stringLexicalUnits(Op1.getOperationName()));
                String[] operationNameOp2 =
lexicalDiscriminant(stringLexicalUnits(Op2.getOperationName()));
                res = rcs.getNormalizedSimilarityMatrix(operationNameOp1, operationNameOp2);
                return res;
        }

        public double[][] semSimLin(Operation Op1, Operation Op2) {
                double res[][] = null;
                ILexicalDatabase db = new NictWordNet();
                RelatednessCalculator rcs = new Lin(db);
                WS4JConfiguration.getInstance().setMFS(true);
                String[] operationNameOp1 =
lexicalDiscriminant(stringLexicalUnits(Op1.getOperationName()));
                String[] operationNameOp2 =
lexicalDiscriminant(stringLexicalUnits(Op2.getOperationName()));
                res = rcs.getNormalizedSimilarityMatrix(operationNameOp1, operationNameOp2);
                return res;
        }

        public double[][] semSimPath(Operation Op1, Operation Op2) {
                double res[][] = null;
                ILexicalDatabase db = new NictWordNet();
                RelatednessCalculator rcs = new Path(db);
                WS4JConfiguration.getInstance().setMFS(true);
                String[] operationNameOp1 =
lexicalDiscriminant(stringLexicalUnits(Op1.getOperationName()));
                String[] operationNameOp2 =
lexicalDiscriminant(stringLexicalUnits(Op2.getOperationName()));
                res = rcs.getNormalizedSimilarityMatrix(operationNameOp1, operationNameOp2);
                return res;
        }

        public double[][] semSimLesk(Operation Op1, Operation Op2) {
                double res[][] = null;
                ILexicalDatabase db = new NictWordNet();
                RelatednessCalculator rcs = new Lesk(db);
                WS4JConfiguration.getInstance().setMFS(true);
                String[] operationNameOp1 =
lexicalDiscriminant(stringLexicalUnits(Op1.getOperationName()));
                String[] operationNameOp2 =
lexicalDiscriminant(stringLexicalUnits(Op2.getOperationName()));
```

```
                res = rcs.getNormalizedSimilarityMatrix(operationNameOp1, operationNameOp2);
                return res;
        }

        public double[][] semSimHSO(Operation Op1, Operation Op2) {
                double res[][] = null;
                ILexicalDatabase db = new NictWordNet();
                RelatednessCalculator rcs = new HirstStOnge(db);
                WS4JConfiguration.getInstance().setMFS(true);
                String[] operationNameOp1 =
lexicalDiscriminant(stringLexicalUnits(Op1.getOperationName()));
                String[] operationNameOp2 =
lexicalDiscriminant(stringLexicalUnits(Op2.getOperationName()));
                res = rcs.getNormalizedSimilarityMatrix(operationNameOp1, operationNameOp2);
                return res;
        }

        public double[][] semSimJCN(Operation Op1, Operation Op2) {
                double res[][] = null;
                ILexicalDatabase db = new NictWordNet();
                RelatednessCalculator rcs = new JiangConrath(db);
                WS4JConfiguration.getInstance().setMFS(true);
                String[] operationNameOp1 =
lexicalDiscriminant(stringLexicalUnits(Op1.getOperationName()));
                String[] operationNameOp2 =
lexicalDiscriminant(stringLexicalUnits(Op2.getOperationName()));
                res = rcs.getNormalizedSimilarityMatrix(operationNameOp1, operationNameOp2);
                return res;
        }

        public double[][] semSimLCH(Operation Op1, Operation Op2) {
                double res[][] = null;
                ILexicalDatabase db = new NictWordNet();
                RelatednessCalculator rcs = new LeacockChodorow(db);
                WS4JConfiguration.getInstance().setMFS(true);
                String[] operationNameOp1 =
lexicalDiscriminant(stringLexicalUnits(Op1.getOperationName()));
                String[] operationNameOp2 =
lexicalDiscriminant(stringLexicalUnits(Op2.getOperationName()));
                res = rcs.getNormalizedSimilarityMatrix(operationNameOp1, operationNameOp2);
                return res;
        }

        public double[][] semSimResnik(Operation Op1, Operation Op2) {
                double res[][] = null;
                ILexicalDatabase db = new NictWordNet();
                RelatednessCalculator rcs = new Resnik(db);
                WS4JConfiguration.getInstance().setMFS(true);
                String[] operationNameOp1 =
lexicalDiscriminant(stringLexicalUnits(Op1.getOperationName()));
                String[] operationNameOp2 =
lexicalDiscriminant(stringLexicalUnits(Op2.getOperationName()));
                res = rcs.getNormalizedSimilarityMatrix(operationNameOp1, operationNameOp2);
                return res;
        }

        private String[] stringLexicalUnits(String cad) {
                String[] result;
                Pattern p = Pattern.compile("_\\p{Lower}");
                Matcher m = p.matcher(cad);
                StringBuffer sb = new StringBuffer();
                while (m.find()) {
                        m.appendReplacement(sb, m.group().toUpperCase());
                }
                m.appendTail(sb);
                cad = sb.toString().trim().replace("_", "");
                p = Pattern.compile("\\p{Lower}\\p{Lu}");
                m = p.matcher(cad);
                sb = new StringBuffer();
```

```
                while (m.find()) {
                        m.appendReplacement(sb, m.group().charAt(0) + " " + m.group().substring(1));
                }
                m.appendTail(sb);
                result = sb.toString().trim().split(" ");
                return result;
        }

        private String[] lexicalDiscriminant(String[] cad) {
                int k = 0;
                for (int i = 0; i < cad.length; i++) {
                        int com1 = cad[i].compareToIgnoreCase("http");
                        int com2 = cad[i].compareToIgnoreCase("for");
                        int com3 = cad[i].compareToIgnoreCase("Return");
                        int com4 = cad[i].compareToIgnoreCase("Result");
                        int com5 = cad[i].compareToIgnoreCase("_");
                        int com6 = cad[i].compareToIgnoreCase("");
                        int com7 = cad[i].compareToIgnoreCase("soap");

                        if ((com1 != 0) && (com2 != 0) && (com3 != 0) && (com4 != 0) && (com5 != 0)
&& (com6 != 0) && (com7 != 0)) {
                                k++;
                        }
                }

                String cadRet[] = new String[k];
                int j = 0;

                for (int i = 0; i < cad.length; i++) {
                        int com1 = cad[i].compareToIgnoreCase("http");
                        int com2 = cad[i].compareToIgnoreCase("for");
                        int com3 = cad[i].compareToIgnoreCase("Return");
                        int com4 = cad[i].compareToIgnoreCase("Result");
                        int com5 = cad[i].compareToIgnoreCase("_");
                        int com6 = cad[i].compareToIgnoreCase("");
                        int com7 = cad[i].compareToIgnoreCase("soap");

                        if ((com1 != 0) && (com2 != 0) && (com3 != 0) && (com4 != 0) && (com5 != 0)
&& (com6 != 0) && (com7 != 0)) {
                                cadRet[j] = cad[i];
                                j++;
                        }
                }
                return cadRet;
        }

        public double matchAverage(double[][] m) {
                double res = 0.0;
                double acumT = 0.0;

                for (int i = 0; i < m.length; i++) {
                        double acumR = 0.0;
                        double contR = 0.0;
                        double promR = 0.0;

                        for (int j = 0; j < m[i].length; j++) {
                                if (m[i][j] != 0.0) {
                                        acumR = acumR + m[i][j];
                                        contR++;
                                }
                        }

                        if (acumR != 0.0)
                                promR = (double) acumR / contR;
                        acumT = acumT + promR;
                }

                int sumTokens = m.length + m[0].length;
                res = (2 * acumT) / sumTokens;
```

```
                return res;
        }

        public SimilarityObject[] setSimilarityDesition(SimilarityObject[] array, String methodName)
{
                SimilarityObject[] res = null;
                String fileName = methodName + ".txt";
                try {
                        PrintWriter writer = new PrintWriter(fileName, "UTF-8");

                        float cf1 = 1, cf2 = 8, cf3 = 2;
                        int r = (int) ((cf1 + Math.sqrt(cf1 + (cf2 * array.length))) / cf3);

                        double[][] Matriz = new double[r][r];
                        int k = 0;
                        for (int i = 0; i < r; i++) {
                                for (int j = i; j < r; j++) {
                                        if (i == j)
                                                Matriz[i][j] = 1;
                                        else {
                                                Matriz[i][j] = array[k++].getSim();
                                                Matriz[j][i] = Matriz[i][j];
                                        }
                                }
                        }
                        for (int i = 0; i < r; i++) {
                                for (int j = 0; j < r; j++) {
                                        writer.print(Matriz[i][j]);
                                        writer.print(" ");
                                }
                                writer.println("");
                        }
                        writer.close();
                } catch (IOException e) {
                        e.printStackTrace();
                }
                res = array;
                return res;
        }

}
```

- **Clase OntologyManagement**

```
import java.io.File;
import java.util.Set;
import org.semanticweb.owlapi.apibinding.OWLManager;
import org.semanticweb.owlapi.model.IRI;
import org.semanticweb.owlapi.model.OWLClass;
import org.semanticweb.owlapi.model.OWLDataFactory;
import org.semanticweb.owlapi.model.OWLIndividual;
import org.semanticweb.owlapi.model.OWLObjectProperty;
import org.semanticweb.owlapi.model.OWLOntology;
import org.semanticweb.owlapi.model.OWLOntologyCreationException;
import org.semanticweb.owlapi.model.OWLOntologyIRIMapper;
import org.semanticweb.owlapi.model.OWLOntologyManager;
import org.semanticweb.owlapi.util.AutoIRIMapper;


public class OntologyManagement {
        private IRI ontoIri;

        public Operation[] LoadOntology(String localDir, String ontologyFile, String ontoloIRI) {
                Operation[] operations = null;
                try {
                        File dir = new File(localDir);
```

```
                        OWLOntologyManager man = OWLManager.createOWLOntologyManager();
                        OWLOntologyIRIMapper autoIRIMapper = new AutoIRIMapper(dir, false);
                        man.addIRIMapper(autoIRIMapper);

                        File ontoFile = new File(ontologyFile);
                        ontoIri = IRI.create(ontoFile);

                        OWLOntology ontology = man.loadOntologyFromOntologyDocument(ontoIri);
                        IRI ontologyIRI = IRI.create(ontoloIRI);
                        IRI OperationClass = IRI.create(ontologyIRI.toString() + "#" + "Operation");
                        IRI InputParameter = IRI.create(ontologyIRI.toString() + "#" +
"hasParameterInput");
                        IRI OutputParameter = IRI.create(ontologyIRI.toString() + "#" +
"hasParameterOutput");
                        IRI ParameterPart = IRI.create(ontologyIRI.toString() + "#" + "hasPart");
                        OWLDataFactory df = OWLManager.getOWLDataFactory();
                        OWLClass operationClass = df.getOWLClass(OperationClass);
                        OWLObjectProperty hasInputParameter =
df.getOWLObjectProperty(InputParameter);
                        OWLObjectProperty hasOutputParameter =
df.getOWLObjectProperty(OutputParameter);
                        OWLObjectProperty hasParameterPart = df.getOWLObjectProperty(ParameterPart);
                        Set<OWLIndividual> operationIndividuals =
operationClass.getIndividuals(ontology);
                        operations = new Operation[operationIndividuals.size()];
                        int i = 0;

                        for (OWLIndividual singleOperation : operationIndividuals) {

                                String operation = "";
                                String inputPar = "";
                                String inputParPart = "";
                                String outputPar = "";
                                String outputParPart = "";

                                operation = getIndividualNameString(singleOperation);

                                Set<OWLIndividual> inputParams =
singleOperation.getObjectPropertyValues(hasInputParameter, ontology);
                                for (OWLIndividual inputParam : inputParams) {
                                        inputPar = getIndividualNameString(inputParam);

                                        Set<OWLIndividual> inputParts =
inputParam.getObjectPropertyValues(hasParameterPart, ontology);
                                        for (OWLIndividual paramPart : inputParts) {
                                                inputParPart = getParameterPartName(paramPart);
                                        }
                                }

                                Set<OWLIndividual> outputParams =
singleOperation.getObjectPropertyValues(hasOutputParameter, ontology);
                                for (OWLIndividual outputParam : outputParams) {
                                        outputPar = getIndividualNameString(outputParam);

                                        Set<OWLIndividual> outputParts =
outputParam.getObjectPropertyValues(hasParameterPart, ontology);
                                        for (OWLIndividual paramPart : outputParts) {
                                                outputParPart = getParameterPartName(paramPart);
                                        }
                                }

                                Operation op = new Operation(operation, inputPar, inputParPart,
        outputPar, outputParPart);
                                operations[i] = op;
                                i++;
                        }
                } catch (OWLOntologyCreationException e) {
                        System.out.println("No se pudo cargar la ontología: " + e.getMessage());
                }
```

```
                return operations;
        }

        String getIndividualNameString(OWLIndividual ind) {
                String res = "";
                String aux = ind.toString();
                int i = aux.indexOf("#");
                int j = aux.indexOf("$");
                res = aux.substring(i + 1, j);
                return res;
        }

        String getParameterPartName(OWLIndividual ind) {
                String res = "";
                String aux = ind.toString();
                int i = aux.lastIndexOf("$");
                int j = aux.indexOf(">");
                res = aux.substring(i + 1, j);
                return res;
        }
}
```

- **Clase Operation**

```
public class Operation {
        private String operationName;
        private String inputMessage;
        private String outputMessage;
        private String inputMessagePart;
        private String outputMessagePart;

        public Operation(String operationName, String inputMessage,     String inputMessagePart,
String outputMessage, String outputMessagePart) {
                this.operationName = operationName;
                this.inputMessage = inputMessage;
                this.outputMessage = outputMessage;
                this.inputMessagePart = inputMessagePart;
                this.outputMessagePart = outputMessagePart;
        }

        public String getOperationName() {
                return operationName;}

        public String getInputMessage() {
                return inputMessage;}

        public String getOutputMessage() {
                return outputMessage;}

        public String getInputMessagePart() {
                return inputMessagePart;}

        public String getOutputMessagePart() {
                return outputMessagePart;}

        @Override
        public String toString() {
                String s = "";
                s = s + "[Operation: " + operationName;
                s = s + "   - Input: " + inputMessage;
                s = s + "   - Output: " + outputMessage + "]";
                return s;
        }
}
```

- **Clase SimilarityObject**

```
public class SimilarityObject {
	String Operation1;
	String Operation2;
	Double sim;
	String Desition;

	public SimilarityObject(String Operation1, String Operation2, Double sim, String Desition) {
		this.Operation1 = Operation1;
		this.Operation2 = Operation2;
		this.sim = sim;
		this.Desition = Desition;
	}

	public String getOperation1() {
		return Operation1;
	}

	public void setOperation1(String operation1) {
		Operation1 = operation1;
	}

	public String getOperation2() {
		return Operation2;
	}

	public void setOperation2(String operation2) {
		Operation2 = operation2;
	}

	public Double getSim() {
		return sim;
	}

	public void setSim(Double sim) {
		this.sim = sim;
	}

	public String getDesition() {
		return Desition;
	}

	public void setDesition(String desition) {
		Desition = desition;
	}
}
```

- **Anexo C**

	50 X 100	1		2		3		4		5		6		7		8		9		10
	0.062275	19	0.036971	16	0.071038	21	0.03533	22	0.047654	15	0.066567	23	0.054335	10	0.043938	24	0.052194	15	0.091784	7
	0.079296	20	0.068841	11	0.048008	24	0.040841	21	0.038173	20	0.063097	25	0.062114	7	0.052989	23	0.051274	22	0.045665	18
	0.103946	19	0.056994	10	0.05835	25	0.249172	2	0.037911	22	0.133304	11	0.050855	13	0.235161	2	0.053447	16	0.056431	10
	0.056341	24	0.070368	7	0.433475	4	0.557557	5	0.307037	7	0.071234	23	0.049869	23	0.056216	12	0.044382	23	0.053092	24
	0.082565	16	0.519082	5	0.086116	10	0.037938	24	0.053141	16	0.09981	14	0.205867	3	0.056337	14	0.047408	13	0.051898	16
	0.073763	16	0.042537	19	0.07327	19	0.059607	19	0.410674	8	0.066633	18	0.046635	18	0.053759	17	0.064102	9	0.06598	23
	0.088312	12	0.054494	13	0.049217	23	0.038776	12	0.052827	11	0.094709	16	0.059136	7	0.060188	10	0.055697	13	0.056771	13
	0.072577	15	0.050825	24	0.056861	24	0.046084	9	0.054753	15	0.29275	9	0.055384	8	0.535378	4	0.043792	13	0.046084	21
	0.057716	20	0.497122	4	0.057699	14	0.050181	10	0.037149	24	0.081973	17	0.484693	4	0.051582	22	0.050118	22	0.064487	2
	0.332212	10	0.04802	16	0.057061	15	0.035503	13	0.044175	23	0.070125	21	0.055673	16	0.040456	18	0.494589	2	0.493242	5
Mejor	0.332212	10	0.519082	4	0.433475	4	0.557557	2	0.410674	7	0.29275	9	0.484693	3	0.535378	2	0.494589	2	0.493242	2
Peor	0.056341	24	0.036971	24	0.048008	25	0.03533	24	0.037149	24	0.063097	25	0.046635	23	0.040456	24	0.043792	23	0.045665	24
X	0.1009	17	0.144525	13	0.099109	18	0.115099	14	0.108349	16	0.10402	18	0.112456	11	0.1186	15	0.0957	15	0.102543	14
Varianza	0.006817	17	0.036853	41	0.013939	49	0.028421	56	0.018071	37	0.004863	29	0.019396	42	0.024795	59	0.019678	42	0.019022	60

		11		12		13		14		15		16		17		18		19		20		
	0.0514	20	0.029863	19	0.381648	5	0.067159	11	0.064746	19	0.040448	18	0.055194	16	0.047325	16	0.403936	6	0.541848	4		
	0.096958	10	0.031389	25	0.05661	8	0.081336	19	0.05754	25	0.51014	5	0.047497	20	0.051271	13	0.050508	24	0.047296	24		
	0.068335	16	0.058994	2	0.060696	20	0.495124	3	0.058599	19	0.053542	11	0.489937	2	0.057059	7	0.132025	9	0.044464	13		
	0.065038	15	0.034761	14	0.066237	4	0.050844	7	0.084056	13	0.054071	20	0.045491	11	0.048742	21	0.052439	11	0.239418	4		
	0.380862	7	0.589362	10	0.102278	4	0.047955	23	0.062467	23	0.095659	3	0.487127	2	0.043238	10	0.062366	17	0.053427	23		
	0.048625	23	0.040036	17	0.06075	6	0.461945	5	0.058893	19	0.056046	13	0.051356	23	0.061025	8	0.086431	13	0.07258	7		
	0.150559	9	0.030609	25	0.040087	16	0.073079	17	0.047901	22	0.062562	14	0.047066	10	0.057815	12	0.05934	18	0.076118	8		
	0.054452	22	0.041536	9	0.050817	24	0.076885	14	0.06352	15	0.04346	6	0.049501	16	0.469891	3	0.06391	12	0.052542	18		
	0.054504	24	0.063466	5	0.070032	4	0.045447	19	0.446732	7	0.051456	6	0.052218	11	0.056087	10	0.059486	19	0.052956	25		
	0.065873	17	0.073376	3	0.046676	21	0.048973	10	0.078053	12	0.063288	20	0.050917	13	0.060651	7	0.068518	12	0.055613	13		
Mejor	0.380862	7	0.589362	2	0.381648	4	0.495124	3	0.446732	7	0.51014	3	0.489937	2	0.469891	3	0.403936	6	0.541848	4	0.463156	4.3
Peor	0.048625	24	0.029863	25	0.040087	24	0.045447	23	0.047901	25	0.040448	20	0.045491	23	0.043238	21	0.050508	24	0.044464	25	0.044476	24
X	0.103661	16	0.099339	13	0.093583	11	0.144875	13	0.102251	17	0.103067	12	0.137631	12	0.09531	11	0.103896	14	0.123626	14		
Varianza	0.010428	37	0.029879	72	0.010531	66	0.031153	45	0.014757	31	0.020688	41	0.034212	47	0.017357	26	0.011683	29	0.024986	67		

	50 X 200																			
	0.046679	19	0.062553	8	0.050712	22	0.048897	10	0.045201	18	0.06667	19	0.200602	2	0.044575	25	0.04103	25	0.048577	24
	0.057328	21	0.049973	23	0.044754	11	0.042185	24	0.046617	11	0.053967	18	0.043292	18	0.044679	20	0.548531	4	0.04812	23
	0.047174	10	0.051258	13	0.043428	19	0.337091	4	0.053439	15	0.043269	20	0.062416	20	0.049175	25	0.041192	24	0.530952	4
	0.498161	4	0.058566	7	0.043616	13	0.038308	21	0.502018	4	0.046436	14	0.46124	5	0.04955	9	0.041003	18	0.046524	21
	0.075274	2	0.045816	7	0.044143	22	0.475277	6	0.050466	9	0.539776	4	0.048719	20	0.041757	15	0.069107	23	0.048403	18
	0.05347	6	0.449045	5	0.03981	24	0.159243	2	0.047883	16	0.059169	7	0.053152	11	0.081524	3	0.042568	14	0.057603	21
	0.046661	9	0.052899	6	0.490219	4	0.043746	14	0.037092	15	0.039349	21	0.040921	18	0.535082	4	0.045713	18	0.049893	24
	0.043932	10	0.043523	19	0.075286	5	0.047621	25	0.061763	10	0.047902	25	0.051876	24	0.058726	20	0.074087	25	0.051632	23
	0.064521	2	0.04106	17	0.167622	2	0.0477	2	0.051413	19	0.145265	3	0.055794	6	0.051206	10	0.053531	7	0.046059	17
	0.049747	9	0.080876	4	0.051046	16	0.03995	20	0.05254	15	0.171029	2	0.048111	8	0.067318	18	0.045443	12	0.059753	6
Mejor	0.498161	2	0.449045	4	0.490219	2	0.475277	2	0.502018	4	0.539776	2	0.46124	2	0.535082	3	0.548531	4	0.530952	4
Peor	0.043932	21	0.04106	23	0.03981	24	0.038308	25	0.037092	19	0.039349	25	0.040921	24	0.041757	25	0.041003	25	0.046059	24
X	0.098295	9.2	0.093557	11	0.105064	14	0.128002	13	0.094843	13	0.121283	13	0.106612	13	0.102359	15	0.10022	17	0.098752	18
Varianza	0.019834	42	0.015733	44	0.0198	66	0.023858	84	0.020508	21	0.023705	73	0.017787	59	0.023265	65	0.024956	58	0.023082	53

	0.052504	19	0.051614	20	0.049421	14	0.099686	4	0.042769	11	0.056688	18	0.051086	10	0.042993	25	0.468433	5	0.512964	4		
	0.049948	24	0.052796	2	0.070849	18	0.050218	21	0.055795	11	0.089446	4	0.04338	11	0.049301	12	0.435065	5	0.04749	23		
	0.479255	5	0.061568	4	0.057611	19	0.047958	19	0.05464	9	0.049314	17	0.044201	18	0.056774	20	0.142574	3	0.061858	20		
	0.053229	13	0.032578	23	0.502389	3	0.059366	7	0.056329	8	0.529622	5	0.045147	7	0.045671	18	0.048118	20	0.074189	13		
	0.051371	10	0.481249	6	0.055118	13	0.043666	15	0.464693	3	0.044706	20	0.083976	2	0.06587	8	0.043751	25	0.048367	22		
	0.041544	20	0.064468	10	0.05259	19	0.039195	14	0.306544	3	0.053548	14	0.071661	25	0.06145	23	0.069213	7	0.048415	25		
	0.054958	14	0.033951	19	0.090531	8	0.039735	16	0.067594	2	0.068694	13	0.49523	3	0.533677	4	0.046734	7	0.040804	20		
	0.056149	7	0.145794	3	0.048787	10	0.049972	21	0.048506	22	0.044667	14	0.041861	24	0.046465	12	0.051963	12	0.061633	24		
	0.051197	25	0.030509	19	0.058703	12	0.525121	5	0.047463	15	0.048523	25	0.044742	12	0.047544	13	0.045716	18	0.0643	9		
	0.127404	4	0.03221	6	0.04603	10	0.047811	18	0.048581	15	0.048805	19	0.041112	20	0.067918	24	0.044682	20	0.049298	24		
Mejor	0.479255	4	0.481249	2	0.502389	3	0.525121	4	0.464693	2	0.529622	4	0.49523	2	0.533677	4	0.468433	3	0.512964	4	0.501147	3.1
Peor	0.041544	25	0.030509	23	0.04603	19	0.039195	21	0.042769	22	0.044667	25	0.041112	25	0.042993	25	0.043751	25	0.040804	25	0.041133	24
X	0.101756	14	0.098674	11	0.103203	13	0.100273	14	0.119291	9.9	0.103401	15	0.09624	13	0.101766	16	0.139625	12	0.100932	18		
Varianza	0.01818	59	0.01923	66	0.019846	27	0.022588	42	0.021138	40	0.022616	42	0.019861	68	0.023109	51	0.028009	62	0.021062	52		

	50 X 500																			
	0.059255	9	0.046192	13	0.055262	9	0.048615	18	0.041302	18	0.048724	16	0.051858	16	0.052854	7	0.058814	11	0.045512	21
	0.041171	15	0.048648	16	0.046225	10	0.047526	22	0.038484	16	0.04166	20	0.049047	24	0.038684	22	0.045479	21	0.0443	20
	0.071678	6	0.535155	6	0.062735	19	0.045811	19	0.041493	19	0.177012	2	0.047693	23	0.046209	15	0.045781	10	0.056417	13
	0.05782	11	0.044599	25	0.04418	25	0.049286	23	0.043511	21	0.044446	23	0.047045	25	0.051716	9	0.085215	11	0.044685	16
	0.520504	5	0.041594	17	0.05453	9	0.545899	5	0.112415	3	0.052507	14	0.236763	2	0.050592	25	0.46555	7	0.048712	11
	0.043287	22	0.055699	6	0.536454	4	0.060667	9	0.119251	3	0.067295	16	0.045449	9	0.510663	6	0.064136	4	0.043316	15
	0.046853	14	0.043599	19	0.03971	21	0.050838	20	0.045528	12	0.529411	5	0.049401	17	0.046872	22	0.459929	5	0.054463	9
	0.048234	10	0.044892	17	0.050774	17	0.060137	21	0.501439	4	0.042632	9	0.052646	17	0.034104	12	0.046502	17	0.076439	5
	0.050792	15	0.03784	13	0.049133	23	0.05348	14	0.051175	14	0.046177	15	0.047124	23	0.061137	24	0.033496	25	0.524452	6
	0.049791	10	0.0768	24	0.044722	21	0.05632	18	0.080811	5	0.050954	22	0.536507	3	0.047314	17	0.052346	25	0.049051	20
Mejor	0.520504	5	0.535155	6	0.536454	4	0.545899	5	0.501439	3	0.529411	2	0.536507	2	0.510663	6	0.46555	4	0.524452	5
Peor	0.041171	22	0.03784	25	0.03971	25	0.045811	23	0.038484	21	0.04166	23	0.045449	25	0.034104	25	0.033496	25	0.043316	21
X	0.098939	12	0.097502	16	0.098372	16	0.101858	17	0.107541	12	0.110082	14	0.116353	16	0.094014	16	0.135725	14	0.098735	14
Varianza	0.02202	25	0.023766	41	0.023737	52	0.024369	34	0.020075	51	0.023372	49	0.025289	73	0.021487	52	0.029898	62	0.022471	34

	0.035184	17	0.040817	21	0.041163	22	0.040988	24	0.047193	9	0.046337	13	0.469808	5	0.078123	21	0.033492	19	0.041243	14		
	0.057847	9	0.058013	19	0.049374	12	0.041625	25	0.049219	8	0.042808	10	0.047136	20	0.049082	6	0.474661	7	0.044065	10		
	0.047386	22	0.206404	2	0.039537	16	0.06084	8	0.044619	20	0.087216	3	0.059383	23	0.046371	12	0.040521	14	0.067916	6		
	0.053896	10	0.046994	25	0.04008	21	0.039548	22	0.041251	11	0.043746	8	0.044943	21	0.041189	14	0.059611	23	0.05016	14		
	0.063839	7	0.042148	19	0.05873	7	0.048677	11	0.047493	20	0.038136	18	0.044003	12	0.034772	12	0.040329	20	0.051391	18		
	0.067264	8	0.050049	25	0.063839	11	0.035307	13	0.044435	11	0.058532	21	0.146817	3	0.077538	5	0.148973	2	0.055054	10		
	0.052441	25	0.066154	6	0.068741	6	0.571635	5	0.061021	3	0.047944	16	0.045483	16	0.037349	9	0.052226	11	0.032932	17		
	0.049962	9	0.047889	8	0.569647	6	0.047169	11	0.060865	6	0.072862	2	0.05546	12	0.038216	22	0.039553	12	0.047822	11		
	0.049547	23	0.057651	7	0.043396	25	0.05038	9	0.506308	4	0.043446	14	0.080811	4	0.543427	7	0.063835	22	0.036485	10		
	0.504652	5	0.490769	3	0.044853	12	0.057253	8	0.048525	15	0.509042	7	0.04895	10	0.038139	10	0.055877	13	0.493607	7		
Mejor	0.504652	5	0.490769	2	0.569647	6	0.571635	5	0.506308	3	0.509042	2	0.469808	3	0.543427	5	0.474661	2	0.493607	6	0.51698	4.1
Peor	0.035184	25	0.040817	25	0.039537	25	0.035307	25	0.041251	20	0.038136	21	0.044003	23	0.034772	22	0.033492	23	0.032932	18	0.038824	23
X	0.098202	14	0.110689	14	0.101936	14	0.099342	14	0.095093	11	0.099007	11	0.104279	13	0.098421	12	0.100908	14	0.092067	12		
Varianza	0.020475	56	0.020272	84	0.027115	48	0.027601	53	0.02092	36	0.020997	40	0.017479	53	0.024704	34	0.018345	46	0.020003	16		

647 X 100	1		2		3		4		5		6		7		8		9		10		11		12		13		14		15		16		17		18		19		20			
	0.11506	31	0.539293	4	0.505507	10	0.05592	278	0.055907	317	0.049825	234	0.054812	73	0.07129	74	0.099226	41	0.057926	124	0.407439	33	0.06955	71	0.050441	160	0.070724	109	0.147374	63	0.04646	160	0.047752	116	0.05005	165	0.049037	62	0.051108	323		
	0.052627	166	0.047165	38	0.054996	59	0.41074	39	0.087344	126	0.050935	90	0.047003	275	0.063886	174	0.486092	15	0.080004	66	0.065795	134	0.051121	303	0.049979	259	0.067134	119	0.055654	285	0.04549	151	0.05038	17	0.058648	25	0.049887	312	0.053239	317		
	0.184252	23	0.050076	294	0.066292	278	0.096905	95	0.091226	120	0.047797	231	0.06204	262	0.053046	300	0.050847	255	0.052387	230	0.05815	212	0.086534	53	0.051073	108	0.054247	230	0.054707	263	0.051803	31	0.048465	166	0.046675	119	0.048944	75	0.478488	16		
	0.141141	27	0.050084	219	0.048436	97	0.059675	229	0.06274	233	0.049244	136	0.04819	135	0.051293	270	0.050141	267	0.489109	18	0.069341	124	0.052682	172	0.049886	302	0.073263	98	0.067123	144	0.050049	294	0.04422	47	0.049622	189	0.050232	31	0.054001	154		
	0.060222	65	0.05063	172	0.046453	316	0.064965	278	0.342594	49	0.050506	148	0.529678	8	0.391524	20	0.052186	171	0.05253	226	0.06393	141	0.050645	305	0.067979	13	0.09797	69	0.05336	289	0.532472	6	0.043159	59	0.05025	291	0.05311	127	0.049624	263		
	0.051576	233	0.050494	153	0.051151	69	0.061229	207	0.070863	172	0.473273	7	0.047513	73	0.063997	94	0.060345	201	0.057919	301	0.063092	152	0.067927	285	0.048735	190	0.061309	144	0.228481	49	0.055687	249	0.549476	4	0.045423	104	0.04852	77	0.046223	290		
	0.051954	182	0.045853	45	0.048806	104	0.061569	197	0.068883	187	0.04825	116	0.045372	271	0.063315	153	0.066298	130	0.053625	184	0.083424	96	0.40392	18	0.532755	3	0.053231	189	0.060797	158	0.049016	295	0.045886	107	0.546779	4	0.046482	155	0.052282	212		
	0.057626	70	0.045799	47	0.046566	176	0.064184	166	0.077764	143	0.04844	51	0.045614	87	0.064325	196	0.049789	168	0.051504	314	0.072349	309	0.051119	281	0.047149	290	0.088614	76	0.070459	127	0.049972	274	0.043704	75	0.04274	58	0.074854	14	0.094343	46		
	0.047083	212	0.063594	237	0.054569	54	0.06602	275	0.058068	257	0.048563	51	0.046388	84	0.051706	238	0.048558	217	0.052152	299	0.053306	282	0.04893	171	0.049923	150	0.079346	90	0.060914	189	0.045497	281	0.048885	242	0.046429	122	0.046482	109	0.059785	100		
	0.215319	22	0.294808	4	0.047023	187	0.053087	277	0.058596	240	0.113351	15	0.045777	99	0.088161	55	0.045121	309	0.044957	284	0.056654	206	0.072022	318	0.0499	48	0.338505	30	0.088031	93	0.042283	110	0.048245	321	0.049802	160	0.51332	4	0.050268	251		
Mejor	0.215319	22	0.539293	4	0.505507	10	0.41074	39	0.342594	49	0.473273	7	0.529678	8	0.391524	20	0.486092	15	0.489109	18	0.407439	33	0.40392	18	0.532755	3	0.338505	30	0.228481	49	0.532472	6	0.549476	4	0.546779	4	0.51332	4	0.478488	16	0.445738	17.95
Peor	0.047083	233	0.045799	294	0.046453	316	0.053087	278	0.055907	317	0.047797	234	0.045372	275	0.051293	300	0.045121	309	0.044957	314	0.053306	309	0.04893	318	0.047149	302	0.053231	230	0.05336	289	0.042283	295	0.043159	321	0.04274	291	0.046482	312	0.046223	323	0.047987	293
X	0.097686	103.1	0.12378	121.3	0.09698	135	0.099429	204.1	0.097399	184.4	0.098018	107.9	0.097239	136.7	0.096254	157.4	0.10086	177.4	0.099211	204.6	0.099348	168.9	0.095445	197.7	0.099782	152.3	0.098434	115.4	0.08869	166	0.096873	185.1	0.097017	115.4	0.098642	123.7	0.098087	96.6	0.098936	197.2		
Varianza	0.003926	7272	0.027237	11348	0.020641	10258	0.012109	6951	0.007571	6230	0.017792	6569	0.023115	9357	0.010882	9070	0.018571	8983	0.018854	10948	0.011793	7082	0.011905	13588	0.023178	12086	0.007317	3485	0.003203	7880	0.02344	12042	0.02528	10279	0.024811	7120	0.021355	8070	0.017974	12582		

647 X 200																																									
	0.068407	6	0.050369	320	0.541186	5	0.049493	242	0.049904	248	0.20258	49	0.079326	47	0.075251	84	0.050851	99	0.07702	187	0.049809	313	0.060649	123	0.047967	180	0.056241	143	0.049983	281	0.050876	257	0.052279	219	0.066615	123	0.056866	163	0.529271	12		
	0.049873	276	0.521538	12	0.049038	234	0.370018	5	0.046037	97	0.134163	62	0.051524	209	0.101309	59	0.040657	130	0.075952	315	0.044722	73	0.340756	24	0.050213	214	0.050955	272	0.047808	188	0.475615	12	0.05408	129	0.058755	171	0.053183	228	0.050372	197		
	0.050202	193	0.049652	217	0.050139	87	0.049641	222	0.04666	112	0.089832	87	0.054838	101	0.112334	55	0.04906	277	0.065385	224	0.049743	288	0.051326	256	0.046005	164	0.056948	257	0.090236	13	0.052413	131	0.052406	223	0.053611	249	0.052429	271	0.05539	78		
	0.043479	12	0.051537	210	0.047841	87	0.041094	62	0.050635	302	0.065155	136	0.050764	295	0.069691	92	0.051646	253	0.060757	266	0.541354	5	0.053433	177	0.0434	114	0.052984	138	0.05052	133	0.050645	247	0.052157	251	0.055577	203	0.065676	113	0.053903	115		
	0.040487	60	0.050422	244	0.050806	193	0.050078	296	0.050068	45	0.157813	55	0.158152	26	0.054377	210	0.060268	309	0.086082	160	0.050227	201	0.052244	207	0.050283	224	0.070339	276	0.058545	254	0.050133	197	0.048653	309	0.115832	123	0.240623	36	0.051684	168		
	0.046343	106	0.048686	216	0.049477	167	0.175875	6	0.530608	7	0.053183	322	0.057679	87	0.054718	203	0.547787	6	0.270124	67	0.045096	82	0.080468	62	0.04967	274	0.071491	58	0.043503	276	0.050447	315	0.301875	24	0.249363	38	0.05184	283	0.050106	187		
	0.046104	223	0.054721	124	0.045313	310	0.050016	214	0.046164	94	0.097246	78	0.050646	311	0.111591	53	0.046606	312	0.061363	232	0.046637	109	0.049556	240	0.05096	142	0.056643	102	0.041422	63	0.05413	232	0.237006	26	0.049229	225	0.1623	43	0.049879	294		
	0.05049	300	0.048508	198	0.046023	243	0.04719	160	0.045563	86	0.053395	290	0.335437	17	0.061762	154	0.047126	208	0.112386	115	0.048401	133	0.051023	187	0.049716	192	0.05117	254	0.04069	55	0.055692	115	0.051984	253	0.055084	216	0.067614	264	0.05334	117		
	0.041567	72	0.056678	100	0.049423	156	0.049238	157	0.053647	196	0.06078	176	0.077475	51	0.289107	29	0.052554	60	0.062314	221	0.04478	73	0.056045	122	0.546733	6	0.056206	110	0.041899	83	0.05034	323	0.057857	95	0.163134	49	0.1466	46	0.052631	117		
	0.523953	2	0.528889	7	0.049967	90	0.059434	261	0.050077	273	0.05569	239	0.073336	54	0.051808	289	0.04667	209	0.062967	250	0.044293	65	0.116896	37	0.05033	208	0.446148	16	0.493222	5	0.048252	292	0.063048	78	0.071969	237	0.055676	186	0.04623	189		
Mejor	0.523953	2	0.528889	7	0.541186	5	0.370018	5	0.530608	7	0.20258	49	0.335437	17	0.289107	29	0.547787	6	0.270124	67	0.541354	5	0.340756	24	0.546733	6	0.446148	16	0.493222	5	0.475615	12	0.301875	24	0.249363	38	0.240623	36	0.529271	12	0.415232	18.6
Peor	0.040487	300	0.048508	320	0.045313	310	0.041094	296	0.045563	302	0.053183	322	0.050646	311	0.051808	289	0.040657	312	0.060757	315	0.044293	313	0.049556	256	0.0434	274	0.050955	276	0.04069	281	0.048252	323	0.048653	309	0.049229	249	0.05184	283	0.04623	294	0.047556	296.8
X	0.09609	125	0.1461	164.8	0.097921	157.2	0.094208	162.5	0.096936	146	0.096984	149.4	0.098918	119.8	0.098195	122.8	0.099323	186.3	0.093435	203.7	0.096506	134.2	0.09124	143.5	0.098528	171.8	0.096912	162.6	0.095783	135.1	0.093854	212.1	0.097135	160.7	0.093917	163.4	0.095281	163.3	0.09928	147.4		
Varianza	0.022662	13012	0.039934	10329	0.02426	8392	0.010988	11053	0.023225	10306	0.002681	10455	0.007943	12263	0.005052	7492	0.024856	11530	0.00411	5408	0.024436	10269	0.008123	6907	0.024807	5391	0.015109	9106	0.019717	11583	0.017997	9908	0.008496	10549	0.004278	5890	0.00427	9748	0.022833	5969		

647 X 500																																									
	0.548815	2	0.066493	182	0.051632	162	0.047613	157	0.060813	295	0.06298	200	0.053316	154	0.064446	131	0.045388	115	0.381941	18	0.049873	207	0.060451	81	0.050496	223	0.433548	19	0.052548	138	0.047823	115	0.236213	47	0.052388	186	0.052601	277	0.066316	291		
	0.044625	58	0.057183	282	0.050364	224	0.050352	277	0.070909	217	0.063986	209	0.056993	93	0.051045	313	0.060133	273	0.057668	124	0.047821	241	0.059576	91	0.050496	236	0.069601	71	0.051398	261	0.048441	74	0.091527	90	0.049706	130	0.076437	80	0.4945	9		
	0.049512	307	0.058128	228	0.052033	142	0.050369	196	0.06702	241	0.222629	65	0.075343	52	0.117287	56	0.050671	248	0.056055	124	0.050469	277	0.063394	73	0.535961	6	0.051395	256	0.50088	15	0.050028	138	0.057756	209	0.049776	124	0.04871	290	0.046194	96		
	0.042139	28	0.358607	52	0.05414	90	0.050511	273	0.083795	167	0.060079	225	0.051742	195	0.051991	258	0.046249	138	0.063066	74	0.051095	180	0.110858	34	0.047865	127	0.055803	153	0.052138	102	0.050137	236	0.059065	218	0.050736	102	0.048283	305	0.050197	278		
	0.04661	187	0.053285	280	0.05029	106	0.536792	7	0.06185	280	0.103508	97	0.059536	175	0.119979	56	0.050322	289	0.083862	55	0.047478	165	0.053976	128	0.04791	49	0.06975	73	0.052491	101	0.048071	73	0.052535	324	0.050816	291	0.052765	159	0.049651	266		
	0.045784	58	0.055311	316	0.090516	34	0.049519	185	0.248565	73	0.067464	210	0.076008	49	0.05806	133	0.048683	153	0.050431	278	0.047857	138	0.05152	208	0.050655	176	0.076086	66	0.052125	137	0.050417	137	0.055198	192	0.050532	271	0.050693	197	0.074989	30		
	0.046529	266	0.054564	317	0.051065	201	0.04538	129	0.074252	195	0.134143	78	0.321651	17	0.049046	270	0.043711	274	0.076819	54	0.05156	147	0.051066	233	0.047183	180	0.057969	117	0.051629	144	0.050743	169	0.054032	307	0.412861	9	0.059646	131	0.046117	86		
	0.044031	38	0.24779	54	0.050463	248	0.04792	164	0.120629	114	0.054639	277	0.049683	139	0.264158	35	0.053291	56	0.052453	172	0.540855	9	0.050317	259	0.046325	97	0.058093	162	0.052075	203	0.53428	8	0.059853	162	0.049507	239	0.057851	122	0.051079	54		
	0.045104	150	0.058711	204	0.048701	86	0.047394	54	0.112187	122	0.071965	151	0.077559	48	0.04841	308	0.518598	11	0.066232	69	0.050393	228	0.398559	16	0.048514	131	0.050978	254	0.05567	186	0.050224	296	0.059306	162	0.147529	15	0.053315	232	0.048111	141		
	0.049023	122	0.05513	241	0.48874	11	0.048365	303	0.08038	180	0.060684	221	0.124912	29	0.132085	52	0.045783	96	0.069925	63	0.050615	93	0.085756	47	0.048133	238	0.05684	132	0.047773	181	0.046146	282	0.194579	49	0.050474	156	0.479494	21	0.050774	269		
Mejor	0.548815	2	0.358607	52	0.48874	11	0.536792	7	0.248565	73	0.222629	65	0.321651	17	0.264158	35	0.518598	11	0.381941	18	0.540855	9	0.398559	16	0.535961	6	0.433548	19	0.50088	15	0.53428	8	0.236213	47	0.412861	9	0.479494	21	0.4945	9	0.422882	22.5
Peor	0.042139	307	0.053285	317	0.048701	248	0.04538	303	0.060813	295	0.054639	277	0.049683	195	0.04841	313	0.043711	289	0.050431	278	0.047478	277	0.050317	259	0.046325	238	0.050978	256	0.047773	261	0.046146	296	0.052535	324	0.049507	291	0.048283	305	0.046117	291	0.049133	281
X	0.096217	121.6	0.10652	215.6	0.098794	130.4	0.097421	174.5	0.09804	188.4	0.090208	173.3	0.094674	95.1	0.095651	161.2	0.096283	165.3	0.095845	103.1	0.098802	168.5	0.098547	117	0.097354	146.3	0.098006	130.3	0.096873	146.8	0.097631	152.8	0.092006	176	0.096432	152.3	0.097979	181.4	0.097793	152		
Varianza	0.025294	10939	0.011441	9318	0.018928	6273	0.023836	9151	0.003205	5241	0.002774	5130	0.006853	4271	0.004608	13055	0.022041	9949	0.010218	5755	0.024127	6081	0.011479	7547	0.023752	6283	0.01397	6236	0.020154	4507	0.023541	8870	0.004448	9192	0.013292	9468	0.018037	9066	0.019518	12715		

1027 X 100	1		2		3		4		5		6		7		8		9		10		11		12		13		14		15		16		17		18		19		20		
	0.051132	404	0.052823	97	0.051774	108	0.049381	215	0.08679	126	0.050949	205	0.066926	325	0.052623	189	0.057479	306	0.429745	15	0.054118	487	0.530096	16	0.046152	73	0.039161	126	0.053629	254	0.097645	190	0.065686	228	0.163819	78	0.05047	534	0.053745	413	
	0.047657	150	0.050717	206	0.064067	76	0.04098	62	0.05256	414	0.050716	218	0.063532	463	0.045911	152	0.060221	226	0.055537	253	0.05141	445	0.083462	47	0.049028	463	0.048472	459	0.479033	27	0.069492	341	0.36274	75	0.053328	469	0.048713	372	0.077287	163	
	0.043985	94	0.050856	492	0.429454	14	0.047534	267	0.398029	26	0.057365	328	0.071143	496	0.049974	418	0.052062	460	0.197139	18	0.13523	66	0.051108	520	0.054336	97	0.045333	303	0.058333	148	0.098721	198	0.072244	522	0.052575	495	0.044858	364	0.053611	469	
	0.492332	11	0.051551	140	0.052463	451	0.047709	299	0.051712	462	0.082863	57	0.318201	110	0.047164	18	0.483594	42	0.0505	182	0.054768	400	0.057845	434	0.533943	2	0.045861	327	0.060313	127	0.067833	342	0.066083	496	0.124163	83	0.0499	157	0.06654	488	
	0.046959	233	0.049725	413	0.127136	33	0.048199	166	0.056203	381	0.04483	434	0.073762	468	0.051045	121	0.054731	372	0.046017	338	0.163115	56	0.052307	207	0.058869	327	0.051765	391	0.051972	321	0.055491	523	0.05249	506	0.060135	260	0.04707	337	0.060934	255	
	0.079666	35	0.086479	50	0.053266	432	0.044961	196	0.068539	487	0.445503	22	0.074819	330	0.052716	354	0.056732	281	0.059722	66	0.320779	38	0.054882	478	0.0517	367	0.035551	71	0.05914	279	0.27378	106	0.056163	400	0.070668	136	0.548296	6	0.063548	411	
	0.051557	459	0.468593	18	0.051895	410	0.049876	243	0.068252	173	0.053059	237	0.054936	510	0.535716	2	0.059693	217	0.052556	207	0.049347	511	0.059271	89	0.050955	294	0.046238	83	0.074623	518	0.05473	481	0.06515	227	0.057971	206	0.04508	123	0.047855	509	
	0.046271	240	0.051423	273	0.054291	168	0.552404	6	0.056147	315	0.047092	336	0.072049	341	0.046533	320	0.05688	324	0.050938	281	0.059535	201	0.044797	255	0.045565	61	0.044888	320	0.05523	169	0.089421	226	0.066632	303	0.05445	327	0.050893	118	0.375723	61	
	0.050211	486	0.051869	180	0.048034	169	0.046775	455	0.098687	121	0.053398	251	0.077524	296	0.046338	207	0.054906	424	0.049905	416	0.056661	302	0.04881	273	0.050042	453	0.048938	496	0.062544	131	0.086073	249	0.067838	209	0.292138	52	0.050599	316	0.06955	187	
	0.046598	95	0.047773	264	0.047518	530	0.04509	200	0.40123	47	0.048922	485	0.060576	449	0.051887	57	0.06252	439	0.050565	222	0.081346	333	0.050803	307	0.05137	447	0.555318	6	0.05484	257	0.05931	473	0.069832	184	0.054439	290	0.049628	459	0.063519	300	
Mejor	0.492332	11	0.468593	18	0.429454	14	0.552404	6	0.40123	26	0.445503	22	0.318201	110	0.535716	2	0.483594	42	0.429745	15	0.320779	38	0.530096	16	0.533943	2	0.555318	6	0.479033	27	0.27378	106	0.36274	75	0.292138	52	0.548296	6	0.375723	61	0.441431	32.75
Peor	0.043985	486	0.047773	492	0.047518	530	0.04098	455	0.051712	487	0.04483	485	0.054936	510	0.045911	418	0.052062	460	0.046017	416	0.049347	511	0.044797	520	0.045565	463	0.035551	496	0.051972	518	0.05473	523	0.05249	522	0.052575	495	0.044858	534	0.047855	509	0.047773	491.5
X	0.095637	220.7	0.096181	213.3	0.09799	239.1	0.097291	210.9	0.133815	255.2	0.09347	257.3	0.093347	378.8	0.097991	183.8	0.099882	309.1	0.104262	199.8	0.102631	283.9	0.103338	262.6	0.099196	258.4	0.096153	258.2	0.100966	223.1	0.09525	312.9	0.094486	315	0.098369	239.6	0.098551	278.6	0.093231	325.6		
Varianza	0.019532	30734	0.01725	23173	0.014128	38094	0.025578	15339	0.019858	30940	0.015413	21609	0.006291	15193	0.023662	20501	0.018186	16062	0.015173	17915	0.007427	33528	0.022597	31632	0.023349	33133	0.026051	30210	0.017687	18502	0.004215	20269	0.00892	24530	0.006038	25127	0.024977	28633	0.009925	24090		

1027 X 200																																								
	0.045693	116	0.045254	275	0.051101	233	0.04813	103	0.053347	333	0.048145	390	0.568464	5	0.054605	284	0.565221	4	0.053452	339	0.05417	130	0.049578	494	0.060483	196	0.050307	126	0.051345	128	0.156286	36	0.057126	312	0.048954	71	0.073324	192	0.050804	206	
	0.047661	95	0.049	486	0.048562	295	0.384172	19	0.057151	237	0.054018	236	0.046638	248	0.054278	351	0.045303	244	0.289194	55	0.048885	319	0.050376	501	0.544725	8	0.049086	192	0.04926	42	0.391137	22	0.056039	46	0.049949	474	0.062283	274	0.048377	411	
	0.047599	527	0.040186	37	0.051916	286	0.110661	39	0.05281	361	0.053347	203	0.048091	313	0.053702	203	0.047476	498	0.058751	320	0.049132	302	0.053152	427	0.049785	367	0.049344	330	0.048611	165	0.050136	331	0.05529	62	0.046562	131	0.106915	125	0.052638	326	
	0.048366	456	0.049871	417	0.047282	397	0.049386	517	0.103579	88	0.050375	506	0.048113	139	0.173125	55	0.042996	141	0.173633	52	0.049795	365	0.435665	41	0.050713	249	0.049186	394	0.049449	190	0.051221	491	0.054613	225	0.0502	227	0.095278	131	0.049937	121	
	0.048029	498	0.551234	5	0.049258	327	0.152621	29	0.050711	463	0.047775	489	0.047474	389	0.052029	410	0.04802	476	0.07084	132	0.051535	175	0.054965	294	0.051036	279	0.049747	237	0.050502	488	0.052211	350	0.04721	305	0.499203	10	0.062392	271	0.048479	59	
	0.045756	94	0.046845	492	0.047436	140	0.050501	255	0.476267	36	0.081058	78	0.046549	132	0.075145	119	0.039447	148	0.056431	227	0.053388	345	0.05281	501	0.050872	165	0.052709	201	0.543528	6	0.050386	425	0.042672	97	0.04158	88	0.311244	69	0.05097	166	
	0.053447	69	0.048409	138	0.050605	289	0.049445	292	0.049526	420	0.050893	428	0.04919	138	0.051398	510	0.039984	50	0.084901	111	0.049212	217	0.062344	381	0.049671	525	0.049201	440	0.049414	494	0.057607	134	0.047698	531	0.049882	308	0.055789	499	0.050615	233	
	0.045526	122	0.050347	380	0.046655	159	0.052033	223	0.066553	138	0.051393	386	0.048806	290	0.062396	160	0.042808	40	0.04959	472	0.470407	24	0.052155	430	0.048817	176	0.046678	533	0.047643	213	0.05285	256	0.047873	113	0.046216	126	0.052018	462	0.050944	56	
	0.048938	143	0.038405	80	0.531492	17	0.049127	354	0.052939	323	0.509663	23	0.046903	334	0.348289	49	0.039621	33	0.048765	475	0.12328	41	0.05203	423	0.051925	244	0.5486	7	0.0498	333	0.070134	96	0.521938	12	0.048351	259	0.075442	169	0.052344	363	
	0.540164	11	0.050151	391	0.049188	513	0.057832	75	0.052631	297	0.053681	201	0.047885	306	0.049837	428	0.048104	245	0.049011	505	0.049973	530	0.075427	134	0.0492	532	0.049601	124	0.048787	404	0.043503	363	0.047626	77	0.108478	22	0.056425	341	0.53475	2	
Mejor	0.540164	11	0.551234	5	0.531492	17	0.384172	19	0.476267	36	0.509663	23	0.568464	5	0.348289	49	0.565221	4	0.289194	52	0.470407	24	0.435665	41	0.544725	8	0.5486	7	0.543528	6	0.391137	22	0.521938	12	0.499203	10	0.311244	69	0.53475	2	0.478268	21.1
Peor	0.045526	527	0.038405	492	0.046655	513	0.04813	517	0.049526	463	0.047775	506	0.046549	389	0.049837	510	0.039447	498	0.048765	505	0.048885	530	0.049578	501	0.048817	532	0.046678	533	0.047643	494	0.043503	491	0.042672	531	0.04158	474	0.052018	499	0.048377	411	0.046518	495.8
X	0.097118	213.1	0.09697	270.1	0.097349	265.6	0.100391	190.6	0.101552	269.6	0.100035	294	0.099811	229.4	0.09748	256.9	0.095898	187.9	0.093457	268.8	0.099978	244.8	0.09385	362.6	0.100723	274.1	0.099446	258.4	0.098834	246.3	0.097547	250.4	0.097808	178	0.098938	171.6	0.095111	253.3	0.098986	194.3		
Varianza	0.024239	39009	0.025493	35817	0.023272	19342	0.011173	27588	0.017595	20214	0.020808	28777	0.027116	14406	0.00916	26851	0.027205	32007	0.006155	31692	0.017463	24857	0.014484	25405	0.024349	26454	0.024908	26539	0.024415	30725	0.011732	28114	0.022231	26703	0.020151	21121	0.006085	20956	0.023445	19443		

1027 X 500																																								
	0.511724	30	0.050947	419	0.061954	443	0.035119	126	0.053516	142	0.059925	302	0.056656	452	0.071531	374	0.100213	24	0.0528	326	0.053277	345	0.084892	105	0.048685	219	0.05729	337	0.051639	458	0.048853	484	0.048112	405	0.049815	143	0.050921	505	0.098263	181	
	0.049484	408	0.045901	298	0.054894	242	0.035931	264	0.06722	492	0.051305	435	0.069693	275	0.067492	410	0.04558	232	0.057897	222	0.456094	20	0.053505	369	0.056495	141	0.062578	327	0.048544	399	0.055596	151	0.04717	159	0.049165	262	0.054441	267	0.054152	501	
	0.054592	308	0.05186	250	0.052785	520	0.037601	387	0.049611	398	0.068693	164	0.068931	345	0.069732	396	0.042246	72	0.462529	39	0.047844	466	0.052978	229	0.513219	19	0.05425	441	0.049085	373	0.400128	23	0.045779	139	0.050043	403	0.054242	269	0.069153	418	
	0.052232	328	0.044886	180	0.049508	429	0.037969	15	0.048145	238	0.051396	490	0.056641	422	0.101877	248	0.516102	10	0.057925	293	0.048026	246	0.079175	121	0.055773	296	0.347647	56	0.478425	27	0.050794	227	0.04492	105	0.048779	283	0.05124	470	0.100386	173	
	0.051155	520	0.050615	368	0.445314	10	0.036112	381	0.049587	525	0.059606	229	0.074555	256	0.099756	228	0.048994	481	0.052067	326	0.046112	101	0.422014	37	0.04929	211	0.076532	167	0.104151	61	0.171998	31	0.049471	314	0.527595	15	0.051765	445	0.070176	298	
	0.050938	323	0.044972	486	0.050865	474	0.03448	38	0.502993	16	0.060155	322	0.358797	72	0.065872	423	0.040181	116	0.052772	349	0.051735	325	0.052069	517	0.048847	435	0.052088	455	0.049945	514	0.053058	133	0.550606	3	0.047714	347	0.447265	36	0.051447	505	
	0.053809	219	0.053009	69	0.049815	319	0.034518	189	0.065716	448	0.054193	435	0.064121	369	0.312009	123	0.046737	517	0.052876	371	0.046879	194	0.069059	136	0.048904	222	0.054529	524	0.061764	138	0.055811	122	0.049501	23	0.067541	169	0.0561	314	0.104796	167	
	0.05173	368	0.533568	11	0.052618	249	0.03758	380	0.048132	427	0.05605	263	0.061115	384	0.061745	472	0.04871	320	0.047549	345	0.048519	384	0.052801	528	0.049217	513	0.053943	336	0.049027	485	0.051381	328	0.047282	41	0.048614	434	0.056766	234	0.090703	185	
	0.051012	527	0.044948	438	0.059027	74	0.66112	2	0.055816	86	0.434816	40	0.086123	186	0.078272	331	0.044773	505	0.052893	378	0.071717	52	0.053083	413	0.051546	247	0.08276	131	0.053582	245	0.049558	257	0.048602	517	0.048575	322	0.07103	122	0.203021	121	
	0.05855	166	0.048462	406	0.051184	406	0.036194	30	0.049206	340	0.058294	335	0.065703	300	0.066737	399	0.048204	201	0.074856	122	0.046905	352	0.054191	187	0.046514	404	0.068748	411	0.050959	331	0.064924	446	0.050614	382	0.048278	346	0.073577	129	0.065881	299	
Mejor	0.511724	30	0.533568	11	0.445314	10	0.66112	2	0.502993	16	0.434816	40	0.358797	72	0.312009	123	0.516102	10	0.462529	39	0.456094	20	0.422014	37	0.513219	19	0.347647	56	0.478425	27	0.400128	23	0.550606	3	0.527595	15	0.447265	36	0.203021	121	0.454249	35.5
Peor	0.049484	527	0.044886	486	0.049508	520	0.03448	387	0.048132	525	0.051305	490	0.056641	452	0.061745	472	0.040181	517	0.047549	378	0.046112	466	0.052069	528	0.046514	513	0.052088	524	0.048544	514	0.048853	484	0.04492	517	0.047714	434	0.050921	505	0.051447	505	0.048655	487.2
X	0.098522	319.7	0.096917	292.5	0.092796	316.6	0.098662	181.2	0.098994	311.2	0.095443	301.5	0.096234	306.1	0.099502	340.4	0.098174	247.8	0.096416	277.1	0.091711	248.5	0.097377	264.2	0.096849	270.7	0.091036	318.5	0.099712	303.1	0.10021	220.2	0.098206	208.8	0.098612	272.4	0.096735	279.1	0.090798	284.8		
Varianza	0.021085	23459	0.023548	26321	0.015358	29528	0.039058	26096	0.0202	32304	0.014245	18527	0.008588	13222	0.005771	11685	0.021865	39508	0.016602	13066	0.016449	23011	0.01316	31970	0.021413	21495	0.008239	23573	0.017992	31322	0.012507	25591	0.02527	33078	0.022754	16725	0.015234	24978	0.001927	20810		

11 Bibliografía

[1] S. Binitha, S S. Sathya, *A Survey of Bio inspired Optimization Algorithms*, International Journal of Soft Computing and Engineering, Volume-2, Issue-2, May 2012.

[2] E. S. Estrada, *Clasificación de servicios web semánticos mediante ontologías*, Proyecto Terminal, Universidad Autónoma Metropolitana Azcapotzalco, D.F., México, 2012.

[3] C. Ethan, Web Services Essentials *Distributed Applications with XML-RPC, SOAP, UDDI & WSD.*O'Reilly, February 2002.

[4] D. Floreano, C. Mattiussi, *Bio-Inspired Artificial Intelligence Theories, Methods and Technologies,* The MIT Press Cambridge, Massachusetts London, Englan, 2008

[5] S. J. Jorge, *Adaptación del algoritmo de la colonia artificial de abejas para resolver problemas de diseño en ingeniería*, Tesis, Universidad Veracruzana Facultad de Estadística e Informática, Xalapa, Ver., Marzo de 2009.

[6] D. Karaboga, Artificial Bee Colony (ABC) Algorithm HomePage, 2005, http://mf.erciyes.edu.tr/abc/

[7] R. Mohanty, *Web services classification using intelligent techniques*, Expert Systems with Applications. Volume 37, Issue 7, July 2010.

[8] J. P. Martínez. *Extracción automatizada y representación de servicios Web mediante ontologías,* Proyecto Terminal, Universidad Autónoma Metropolitana Azcapotzalco, D.F., México, 2011.

[9] Z. Wang, B. Wan, *Web Services Selection Based on Discrete Artificial Bee Colony Algorithm*, International Journal of Digital Content Technology and its Applications Volume6, Number21, November 2012

[10] B. Xing, J. Wen, Innovative Computational Intelligence: *A Rough Guide to 134 Clever Algorithms*, Springer, Volume 62, 2014.

[11] C. Zhang, B. Zhang, *A Hybrid Artificial Bee Colony Algorithm for the Service Selection Problem*, Discrete Dynamics in Nature and Society Volume 2014.

12 Entregables

Los entregables del Proyecto de Integración son los que se listan:

- Cd con todos los componentes para hacer funcional el proyecto.
- Código fuente documentado.