UNIVERSIDAD CATÓLICA DE LA SANTÍSIMA CONCEPCIÓN
Facultad de Ingeniería
Ingeniería Civil Informática

INTERFAZ DE USUARIO GRÁFICA PARA EL ANÁLISIS DE RESULTADOS EN OPTIMIZACIÓN MULTI-OBJETIVO

BRAULIO PADILLA REYES

INFORME DE PROYECTO DE TÍTULO PARA OPTAR AL TÍTULO DE INGENIERO CIVIL INFORMÁTICO

Profesor Guía

Pedro Gómez

Concepción, Noviembre de 2015

Resumen

Optimizar quiere decir buscar mejores resultados, mayor eficiencia en el desempeño de alguna tarea. Para el caso de un problema multi-objetivo, se busca la eficiencia en más de un objetivo a la vez. A través del tiempo, la optimización se ha vuelto un aspecto importante en múltiples campos del mundo real, en especial su enfoque a la ingeniería.

A medida que el estudio de la optimización multi-objetivo ha ido en aumento, se han diseñado muchos algoritmos de resolución de problemas, para poder encontrar el conjunto de soluciones óptimas, así también de indicadores de calidad, métricas, para analizar los resultados.

El objetivo de este proyecto es el desarrollo de un sistema para el análisis de soluciones de problemas multi-objetivo, resaltando el uso de plugins, que hará al sistema escalable e incremental. Además, se desarrollará en Java, dando la propiedad de adaptarse a diferentes plataformas. Se creará un prototipo que pueda analizar problemas de manera visual y analítica, a través de métricas analizadas en este proyecto.

Abstract

Optimize means search better results, greater efficiency in carrying out a task. In the case of multi-objective problem, efficiency is sought in more than one target at the same time. Over time, optimization has become an important aspect in many fields of the real world, especially its approach to engineering.

As the study of multi-objective optimization has grown, many algorithms have been designed to solve problems, to find the optimal set of solutions, so quality indicators, metrics, to analyze the results.

The objective of this project is to develop a system for the analysis of problem solving multi-objective, highlighting the use of plugins, which will make the system scalable and incremental. It was also developed in Java, giving the property to adapt to different platforms. A prototype that can analyze problems of visual and analytical way, through metrics analyzed in this project will be created.

Agradecimiento

A Dios todopoderoso por otorgarme un triunfo personal a mi vida, además de la sabiduría y entendimiento para lograrlo.

A mi compañera de carrera y vida, Karina Pedreros, por ser la piedra angular de mi vida, apoyando mis decisiones y acompañándome en esta vida.

A mi madre y familia, por el apoyo a través de esta ardua tarea, por estar a mi lado y dándome ánimos para finalizar este proyecto.

A la familia de mi novia, por encontrar un segundo hogar en ellos, dando tanto por mí como a uno de los suyos, donde estaré eternamente agradecido por su incondicional apoyo.

A mi profesor guía Pedro Gómez, por entregar las herramientas necesarias para logar completar esta tarea exitosamente, su compromiso y su dedicación siempre lo tendré en mis recuerdos.

Gracias a todas las personas que creyeron y apoyaron en la realización de esta tesis.

Índices

Índice de Figuras

Nomenclaturas

MOP= Multi Objective Problems (Problemas Multi-Objetivo).

JVM= Java Virtual Machine, Máquina Virtual de Java.

NSGA-II= Non-dominated Sorting Genetic Algorithm II.

SPEA2= Strength Pareto Evolutionary Algorithm 2.

PAES= The Pareto Archived Evolution Strategy.

PESA-II=Pareto Envolope based Selección.

OMOPSO= Optimized Multi-Objective Particle Swarm Optimization.

MOCell = Cellular Genetic Algorithm for Multi-objective Optimization.

AbYSS= Archive-Based Hybrid Scatter Search.

PSO= Particle Swarm Optimization.

CMA-ES= Covariance Matrix Adaptation Evolution Strategy.

ZDT= Zitzler-Deb-Thiele Problem.

DTLZ= Deb, Thiele, Laumanns, & Zitzler, Scalable Test Problems for Evolutionary Multi-objective Optimization.

WFG= Walking Fish Group.

CEC2009= Congress on Evolutionary Computation problems of 2009.

LZ09= Li and Zhang. Multi-objective optimization problems.

MTSP= The Multi-objective Traveling Salesman Problem.

MQAP= The Multi-objective Quadratic Assignment Problem.

MOGA= Multi-Objective Genetic Algorithm.

IBEA= Indicator-Based Evolutionary Algorithm.

PLS= Partial Least Squares.

IBMOLS = Indicator-Based Multi-Objective Local Search.

DMLS = Dominance-Based Multi-Objective Local Search.

1 Capítulo: Introducción

1.1 Presentación del Tema

Guimoo es una aplicación desarrollada en lenguaje C, dedicada al análisis de resultados para problemas de optimización multi-objetivo. Dentro de sus funciones se destaca la visualización gráfica de las fronteras de Pareto para problemas con 2 y 3 objetivos (evaluación cualitativa). Además permite realizar una comparación cuantitativa a través de diferentes métricas. Este sistema fue creado en 2005, apoyado por el INRIA, de la abreviatura Instituto Nacional Francés para la Investigación en Informática y Automática (Inria, 2015). Dejó de ser mantenida el año 2009, a pesar de que hay un considerable número de científicos que aún utilizan esta herramienta. Como consecuencia a esta falta de soporte de esta aplicación, el sistema ha quedado obsoleto dejando de ser compatible con las nuevas plataformas (arquitecturas en 64 bits). Se desea hacer una re-ingeniería al sistema actual, dejando a un lado la incompatibilidad de los diferentes sistemas, además de crear una estructura basada en componentes, dando la capacidad de añadir nuevas métricas actuales o modificar métricas obsoletas.

1.2 Objetivo General

Desarrollar una aplicación de interfaz gráfica para el análisis de resultados en optimización multi-objetivo.

1.3 Objetivos Específicos

1. Estudiar conceptos relacionados con la optimización multi-objetivo y las herramientas para el desarrollo de la aplicación.
2. Analizar las funciones del sistema en las aplicaciones similares.
3. Diseñar y programar la aplicación.
4. Testear la aplicación.

1.4 Justificación del Problema

Esta aplicación todavía sigue siendo utilizada y referenciada en muchos trabajos académicos relacionados con la optimización multi-objetivo. Sin embargo, muchos de estos usuarios se han visto imposibilitado en seguir usado esta aplicación, debido a la actualización en que ellos han debido realizar en sus plataformas computacionales (a arquitecturas en 64 bits). Si a esto le sumamos, la carencia de actividad del equipo de desarrollo, asumiendo que ha sido dejado de actualizar.

Otra justificación es que el sistema está basado en C++, siendo complejo adaptarlo a las nuevas arquitectura (64 bits), debido a cierto problemas con las longitudes de los datos, lo que limita su uso en diferentes plataformas.

1.5 Delimitación del Problema

Como primera opción, para implementar esta aplicación, se va a usar el lenguaje de programación Java, bajo una metodología de desarrollo evolutivo, además de que Java tiene la cualidad de ejecutarse sobre una máquina virtual, lo que hace se comunique con los distintos sistemas operativos. Esta metodología se justifica en el requerimiento de poder agregar o quitar nuevas métricas para ser ofrecidas por esta aplicación.

La incorporación de nuevas métricas podría ser transparente al usuario a través de una programación automática de las fórmulas para ser evaluada, y de esta forma el usuario no requeriría poseer un vasto conocimiento en programación. Esta característica requiere un tiempo considerable en su desarrollo, por lo que en primera instancia, nos limitaremos en ofrecer un número limitado de métricas precargadas y como prototipo del proyecto título la posibilidad de agregar o eliminar métricas, dejando la creación automática de las métricas para un trabajo posterior.

Una de las contribuciones para estas nuevas aplicaciones, está en poder agregar o quitar métricas a través de componentes, lo que permitiría que este programa no quede obsoleto en el tiempo, dando la posibilidad de actualizar y agregar nuevas métricas que surgen para un mejor estudio de problemas de multi-objetivo, dado que esta metodología de programación no requerirá al usuario de un conocimiento avanzado sobre el lenguaje Java.

1.6 Metodología

Objetivo 1: Estudiar conceptos básicos relacionados con la optimización multi-objetivo y las herramientas para el desarrollo de la aplicación.

1. Estudio acerca de optimización multi-objetivo en sitios como Google Schoolar y las bases de datos suscritas en la Universidad.
2. Estudio sobre el lenguaje de programación Java, se recopilara información de libros autorizados y de la página oficial del lenguaje o en su caso del interprete que se use.

Objetivo 2: Analizar las funciones del sistema en las aplicaciones similares.

1. Revisión de las aplicaciones similares, a través de un análisis al sistema.
2. Detección de funcionalidades que se incorporan en mejora para la nueva aplicación.

Objetivo 3: Diseñar y programar la aplicación.

1. Desarrollo de estructura de plugins.
2. Implementación de la aplicación con Netbeans, un IDE (Integrated Development Environment, en castellano significa ambiente de desarrollo integrado) de Java.

Objetivo 4: Testear la aplicación.

1. Planificación de testeos.
2. Análisis de Resultados.

2 Capítulo II: Estado del Arte

En el siguiente capítulo se analiza los programas similares a la aplicación en desarrollo:

2.1 JMetal

Es un conjunto de librerías para la implementación de algoritmos meta-heurísticos desarrollado en Java. Está orientado en objetos, su análisis se enfoca en la resolución de problemas de optimización multi-objetivo a través de meta-heurística. Dentro de las características de JMetal (Antonio J. Nebro, 2014), están:

- Portabilidad, gracias a su lenguaje basado en Java.
- Implementación de algoritmos Multi-objetivos: NSGA-II, SPEA2, PAES, PESA-II, OMOPSO, MOCell, AbYSS, entre los más usados.
- Implementación de algoritmos mono-objetivos: algoritmos genéticos (generacional, estado estable, celular sincrónico, celular asincrónico), PSO, DE, CMA-ES.
- Problemas incluidos: problemas conocidos (ZDT, DTLZ, WFG, CEC2009, LZ09), problemas clásicos (Kursawe, Fonseca, Schaffer), problemas con restricciones (Srinivas, Tanaka, Osyczka2, Constr_Ex, Golinski, Water), problemas combinacionales (MTSP, MQAP).
- Indicadores de calidad: hipervolumen, propagación, distancia generacional, distancia generacional inversa, épsilon adictivo, R2*, WFG hipervolumen.
- Variables representativas: binario, real, binario codificado real, enteros, permutación, mezclas codificadas (real + binario, entero +real).

Para más información de los algoritmos, puede visitar la página Emoo Web page. (Coello, 2015).

2.2 ParadisEO-MOEO

Es un conjunto de librerías dedicadas al diseño, implementación y análisis de meta-heurística de optimización multi-objetivo. Está basado en un modelo conceptual que tiende a unificar un sustancial número de metodologías propuestas hasta ahora, siguiendo las principales cuestiones de la asignación de la aptitud, la preservación de diversidad y elitismo.

Paradiseo (Liefooghe Arnaud, 2010) es una framework de tipo caja blanca, esto quiere decir que contiene clases concretas definidas en interfaces, orientado a objeto, basado en el lenguaje de C++, portátil a través de diferentes tipos de sistemas de Unix (Linux, MacOS y Windows). Se rige por la licencia ceCILL(CEA CNRS INRIA Logiciel Libre (Inria, 2015)). El framework incorpora algunas de las características y técnicas para métodos de resolución y tiene como objetivo proporcionar un conjunto de clases que permiten facilitar y acelerar el desarrollo de cómputo eficiente. El gran conjunto de clases modulares se combinan para desarrollar meta-heurísticas Multi-objetivo. Esta sistema mantiene el código fuente relacionado y regularmente es actualizado por los desarrolladores. Además, el marco evoluciona constantemente de acuerdo a las necesidades y facilita el desarrollo de nuevos algoritmos, ya sea secuencial o en paralelo.

Dentro de las características principales estas:

- Portabilidad: Windows, Unix y MacOS, opcionalmente en arquitecturas paralelas y distribuidas.
- La descomposición de grano fino, en componentes más pequeños como:
    - Las relaciones de dominancia: Pareto, debilidad, épsilon-dominancia.
    - Esquemas de asignación de fitness: escalar, enfoques basados en indicadores base- dominancia.
    - Mecanismos de preservación de la diversidad: compartir, hacinamiento.
    - Elitismo: selección elitista, sustitución elitista, técnicas de archivo (sin límites, limitada, de tamaño fijo).
    - Herramientas estadísticas e indicadores de calidad: Hipervolumen, Épsilon aditiva y multiplicativa, Contribución, Entropía.
- Meta-heurísticas de multi-objetivo Fácil de usar (sobre la base de los componentes antes mencionados).
    - Los algoritmos evolutivos: Moga, NSGA, NSGA-II, SPEA2, IBEA.
    - Algoritmos de búsqueda local: PLS, IBMOLS, DMLS.
    - Versiones de los algoritmos Híbridos, paralelos y distribuidos.
- Representación de solución para problemas de optimización multi-objetivo de naturaleza continua y combinatoria.
- Variables de tipo: números enteros, permutaciones, representación-Real, codificado, definido por el usuario.

- Problemas de optimización multi-objetivo: Implementaciones de problemas para ser conectados a ParadisEO-MOEO están disponibles para su descarga como contribuciones:
    - Proceso de evaluación de funciones de pruebas continuas: ZDT, DTLZ.
    - Problemas combinatorios Objetivo de referencia: TS, QAP, línea de flujo.
    - Problemas del mundo real: el enrutamiento, la programación, la bioinformática.

2.3 Guimoo

Guimoo (Graphical User Interface for Multi Objective Optimization), es un software libre dedicado para el análisis de resultados en optimización multi-objetivo (Inria, 2015).

Dentro de las características que posee son,

- Visualización en línea de las fronteras de Pareto aproximados, esta información puede ser utilizada para analizar meta-heurísticas más exactamente. Un frente de Pareto se características por su continuidad y discontinuidad, convexidad, distancias entre los puntos, homogeneidad, entre otros.
- Indicadores para el análisis de manera cuantitativa y cualitativa de algunos aspectos de su rendimiento, algunos de los cuales se destacan R-metric, contribución, entropía, distancia generacional, espaciamiento, etc.

Dado que la optimización multi-objetivo evolutiva está creciendo a pasos gigantes y varios algoritmos exitosos han emergido recientemente, junto con el creciente interés en los problemas multi-objetivos. A través de GUIMOO se permite comparár soluciones dadas por estos algoritmos gracias a su visualización gráfica, así también de sus muchas métricas.

3 Capítulo: Marco Teórico

3.1 Optimización Multi-objetivo

En la vida cotidiana, siempre enfrentamos problemas donde debemos optimizar nuestros recursos, como por ejemplo buscar un auto que sea económico, rápido y cómodo; pero no es tan fácil conseguir todas las características que estamos buscando de manera sencilla. En este punto es donde nace la optimización multi-objetivo.se define la optimización multi-objetivo como el proceso donde debemos tomar decisiones basadas en una cantidad numerosa de variables con características y comportamientos diferentes y tomar decisiones, de manera de obtener los beneficios.

Según (Galeano S.E., 2008) se define optimización tradicional, las metodologías se centran en encontrar un conjunto de elementos de manera que se mejore el resultado dado por la función objetivo. No obstante, los problemas reales involucran otra serie de objetivos que pueden ser de tanto interés como el que se optimizó, e incluso ser tan importantes y conflictivas que harían inútil la solución definida. La optimización multi-objetivo tiene como fin, resolver este tipo de problemas consiguiendo el conjunto de mejores soluciones, cumpliendo los objetivos y restricciones respectivas. A parte se detalla los conceptos relacionados a la optimización multi-objetivo.

Conceptos Básicos.

Para entender los problemas de Optimización Multi-objetivo (MOP), hay que definir tres conceptos previos:

Funciones Objetivos, son la medida cuantitativa del funcionamiento del sistema que se desea optimizar, ya sea maximizar o minimizar.

Variables, representan las decisiones que se pueden tomar para modificar y alterar la función objetivo, estas se clasifican en variables independientes o de control y variables dependiente o de estado.

Restricciones, representan el conjunto de relaciones que algunas variables deben de satisfacer, se expresan en términos de ecuaciones e inecuaciones.

Un MOP general está compuesto de *n* variables de decisión, con un conjunto de *k* funciones objetivo y un conjunto de *m* restricciones. Las funciones objetivo y las restricciones son funciones de las variables de decisión.

Optimizar, $y = F(x) = (f_1(x) \dots f_k(x))$.

Sujeto a, $g(x) = (g_1(\mathrm{x}) \dots g_m(x)) \leq 0$.

Donde, $x = (x_1, \dots, x_n) \in X \subseteq R^n$.
$\mathrm{y} = (y_1, \dots, y_n) \in \mathrm{Y} \subseteq R^k$.

Sabemos que $\boldsymbol{x}$ es el vector de decisión e $\boldsymbol{y}$ es el vector objetivo. El espacio de decisión se denota por X y el espacio objetivo por Y. En el proceso de optimización, dependiendo del problema, se puede maximizar o minimizar.

Ahora, considerado un conjunto de restricciones $g(x) \leq 0$, se determinará un conjunto de soluciones factibles X_f y su correspondiente conjunto de vectores objetivo factible Y_f.

El conjunto de soluciones factibles X_f, se define como el conjunto de decisión *x*, que satisface las restricciones $g(x)$,

$$X_f = \{x \in X \mid g(x) \leq 0\}$$

La imagen de $\boldsymbol{X}_f$, es la región de factibilidad en el espacio objetivo y se denota por,

$$Y_f = F(X_f) = \cup_{x \in \boldsymbol{X}_f}\{F(x)\}$$

De estas definiciones se tiene que cada solución de MOP consiste de un vector de decisión $\boldsymbol{x}$ que conduce a un vector objetivo *y*, donde cada x debe cumplir con el conjunto de restricciones $g(x) \leq 0$, en este caso en particular. El problema de optimización es encontrar el x que tenga el mejor valor de $f(x)$. Pero en MOP, es difícil encontrar una única solución, dado que existe un conjunto de soluciones, donde ninguna es mejor que la otra, considerando todos los objetivos. En este punto, donde existen conflictos entre los diferentes objetivos que

componen el problema, se busca un *óptimo*, aquí es donde se introduce el concepto de óptimo de Pareto.

Optimalidad de Pareto

La teoría de optimalidad de Pareto proporciona definiciones matemáticas precisas de los conceptos multi-objetivo generales, como las soluciones no comparables y las soluciones dominadas. Dentro de las cuales definiremos como:

Dominancia de Pareto

Un vector $U = (u_1, \dots\dots, u_k)$ se dice que domina a otro vector $V = (v_1, \dots\dots, v_k)$ si y sólo si U es parcialmente mejor que *V*, esto es,

$$f_i(u) \leq f_i(v)\ \forall\ i \in \{1,2, \dots, b\}\ \wedge\ \exists\ \mathrm{j} \in \{1,2, \dots, \mathrm{b}\} \mid\ f_i(u) < f_i(v)$$

Con esto decimos que *U* domina a *V*, si *U* es mejor o igual que *V* en todos los objetivos y estrictamente mejor en algunos de ellos.

La dominancia de un vector *U* sobre un vector *V* se denota:

$$U \succ V\ , U\ \boldsymbol{domina\ a}\ V.$$

Nota: En el caso de que el vector U sea estrictamente mejor que *V* para todos los objetivos, se denominara dominancia de Pareto fuerte.

Dominancia de Pareto débil

Dado un vector $U = (u_1, \dots\dots, u_k)$, se considera que domina de forma débil a otro vector $V = (v_1, \dots\dots, v_k)$ si y solo si, *U* es parcialmente mejor o igual que *V*, es decir,

$$\forall i \in \{1, \dots, k\}, U_i\ es\ mejor\ o\ igual\ que\ V_i$$

Esto significa, $U \succ V$ o $U = V$. La dominancia débil de *U* sobre *V* se denota como $U \succcurlyeq V$.

Dominancia de Pareto Estricta

Un vector *U* se dice que domina de manera estricta a otro vector *V*, si y solo si, *U* es mucho

Mejor que *V* para todos los objetivos determinados, esto es

$$\forall i \in \{1, \dots, k\}, U_i\ es\ mejor\ que\ V_i$$

La dominancia estricta de *U* respecto a *V*, se denota como $U \succ\succ V$.

Soluciones no comparables (Relación de Nodominancia)

Dado los vector $U = (u_1, \ldots\ldots, u_k)$ y el vector $V = (v_1, \ldots\ldots, v_k)$, se considera soluciones no comparables si y solo si la relación U $\succ$ V no se cumple. La relación de *nodominancia* del vector *U* con respecto al vector *V* se escribe como $U \nsucc V$ y se lee ***U*** nodomina a ***V.***

Vectores no Comparables

Los vectores *U* y *V* son no comparables si y sólo si:

$$(U \nsucc V) \wedge (V \nsucc U) \wedge (V \neq U)$$

Para este punto, donde ni *U* es mejor que *V*, ni *V* puede ser tomada mejor que *U*. Aunque los vectores *U* y *V* no son estrictamente iguales, las dos soluciones pueden ser consideradas igualmente buenas debido a que ninguna es mejor que la otra cuando se consideran todos los objetivos. Se dice entonces que vector *V* es indiferente o no comparable al vector *U*, denotándose como $\boldsymbol{V \sim U}$.

Óptimo de Pareto

Se dice que una solución $x \in X_f$ es Pareto óptima con respecto a un conjunto $\Omega \subseteq X_f$ si y solo si:

$$\nexists x_m \in \Omega \, para \, el \, cual \, V_m = F(x_m) = \big(f_1(x_m), \ldots, f_k(x_m)\big) domina \, a$$

$$U = F(x) = (f_1(x), \ldots, f_k(x))$$

Es decir, la solución *x* es una solución Pareto óptima si es nodominada por alguna otra solución en el conjunto factible.

Conjunto de Pareto Óptimo

Para un MOP, el conjunto Pareto óptimo, denotado por *P**, se define como:

$$P^* = \{x \in X_f \, / \, \nexists x_m \in X_f para \, el \, cual \, F(x_m) domine \, a \, F(x)\}$$

Es decir, una solución *x*, pertenece al conjunto Pareto óptimo si no existe ningún otro vector en el espacio factible que domine al vector *x*.

Frente de Pareto Óptimo

Es el conjunto de vectores nodominados en el espacio objetivo. En términos formales, se define como:

$$PF^* = \{U = F(x) = \big(f_1(x)\big) \,/\, x \in \ P^*\}$$

Donde *F(x)* representa el problema de optimización Multi-objetivo, y *P** es el conjunto de Pareto óptimo.

3.2 Métricas

A través de las gráficas que representan los diferentes frentes de Pareto, podemos apreciar una manera visual sobre un MOP, pero no es posible tomar una decisión precisa y detalla solo mirando gráficos, se debe usar métricas, que nos ayuden a tomar una solución más precisa antes problemas donde no siempre se ve puede apreciar de manera visual cual solución es mejor que otra.

Cuando se considera una métrica para MOP debemos tomar en cuenta:

1.- Minimizar la distancia del frente de Pareto producido por el algoritmo seleccionado con respecto al verdadero frente de Pareto, en el caso que esta fuese conocida.

2.- Maximizar la distribución de las soluciones encontradas, de forma que se pueda tener una distribución de soluciones no dominadas suaves y uniformes.

3.- Maximizar la cantidad de elementos del conjunto de óptimo de Pareto encontrados.

Aunque tener un único valor de la métrica no indica el desempeño de un algoritmo. Se debe considerar que el problema de las métricas curiosamente también es un problema de multi-objetivo. En consecuencia, es necesario de ocupar varias métricas para poder determinar el desempeño del algoritmo en cuestión, dado que una métrica no mide todos los aspectos de una solución.

3.2.1 Cobertura

Propuesta por, esta métrica compara dos conjunto de vectores no dominados calculando la fracción de cada uno que es *cubierta* o dominada por el otro.

Supongamos que queremos compara los algoritmos A_1 y A_2, y deseamos comparar sus desempeños. En esta métrica, el conjunto de vectores no dominados resultantes de una ejecución del algoritmo A_1 y de otra del algoritmo A_2 y viceversa.

Consideremos que X', X'' ⊆ X sean dos conjuntos de variables de decisión. La función C mapea el par ordenado (X', X'') al intervalo [0,1]:

$$CM(X', X'') = \frac{|\{a'' \in X''; \ a' \in X' : a' \succcurlyeq a''\}|}{|X''|}$$

Si CM(X', X'') = 1 entonces significa que todos los puntos en X'' son dominados por o son iguales a los puntos en X'. En el caso que CM (X', X'') = 0, entonces significa que ninguno de los puntos en X'' están cubiertas por el conjunto X'. Esta métrica se puede usar para demostrar si el resultado de un algoritmo domina al resultado del otro sin señalar que tan bueno es. En consecuencia, esta técnica de comparación no comprueba la uniformidad de las soluciones a lo largo del frente de Pareto. Otro punto a considerar con esta métrica es que puede retornar un mejor valor para un algoritmo que produzca un solo vector no dominado, más cercano a PF_{true} que otro algoritmo que produzca varios vectores bien distribuidos, pero más alejado de PF_{true}. Esta métrica fue creada para complementar la métrica de espacio **cubierto.**

3.2.2 Tamaño del Espacio Dominado

Esta métrica fue introducida por (E. Zitzler, 1998), conocida como *size of the dominated space*. Esta métrica estima el tamaño del conjunto dominado global en el espacio de las funciones objetivo. La idea de esta métrica es calcular el área del espacio de las funciones objetivo cubiertas por los vectores no dominados generados por nuestro algoritmo.

Para problemas con dos funciones objetivo, cada vector dominado representa un rectángulo.

3.2.3 Distancia Generacional

La distancia generacional fue introducida por (D. Van Veldhuizen, 1998) en 1998, como una manera de estimar que tan lejos están los elementos de $PF_{current}$ de PF_{true}.

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n}$$

Donde n es el número de vectores no dominados en $PF_{current}$ y d_i es la distancia Euclidiana (medida en el espacio de las funciones objetivos) entre cada una y el miembro más cercano de PF_{true}. Hay que considerar que si el valor de GD = 0 indica que todos los elementos generados están PF_{true}. Por lo tanto, cualquier otro valor nos indica que tan alejado esta del verdadero frente de Pareto del problema.

3.2.4 Separación

Esta métrica fue propuesta por (Schott, 1995), define la variación de distancias de los vectores vecinos en PF_{know}. La fórmula es,

$$S \triangleq \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\bar{d} - d_i)^2}$$

Donde, d_i es la distancia Euclidiana, $\bar{d}$ es la media de todos los d_i y n es el número de vectores en PF_{know}. Cuando S = 0, todos los miembros del vector están separados uniformemente. Esta métrica no requiere el conocimiento de PF_{true}.

3.2.5 Contribución

Esta métrica fue creada por (H. Meunier, 2000), esta métrica define la contribución de un conjunto de soluciones PO_1 relacionados a otro conjunto de soluciones PO_2 es el radio de soluciones nodominadas producidas por PO_1.

- Se define PO_1 y PO_2 como dos conjuntos de soluciones.
- Se define C conjunto de soluciones en $PO_1 \cap PO_2$.
- W_1 (respectivamente, W_2) es el conjunto de soluciones en PO_1 (respectivamente, PO_2), donde dominan algunas soluciones de PO_2 (respectivamente, PO_1).
- L_1 (respectivamente, L_2) será el conjunto de soluciones en PO_1 (respectivamente, PO_2), donde están dominadas por algunas soluciones de PO_2 (respectivamente, PO_1).
- N_1 (respectivamente, N_2) serán las otras soluciones de PO_1 (respectivamente PO_2), donde:

$$N_i = PO_i / (C \cup W_i \cup L_i).$$

Se define el PO* como el conjunto de soluciones de $PO_1 \cup PO_2$, por lo tanto,

$$\|PO^*\| = \|C\| + \|W_1\| + \|N_1\| + \|N_2\| + \|W_2\|$$

Se define la contribución como,

$$C(PO_1/PO_2) = \frac{\|C\|/_2 + \|W_1\| + \|W_2\|}{\|PO^*\|}$$

3.2.6 Entropía

La entropía permite entender la divergencia del frente de Pareto logrado. Está basado en la definición del nicho de una solución s, la cual representa todas las soluciones que están cerca de la solución *s*.

- Se define PO_1 y PO_2 como dos conjuntos de soluciones.
- PO^* será el conjunto de soluciones optimas de Pareto de $PO_1 \cup PO_2$.
- N_i será la cardinalidad de las soluciones de $PO_1 \cup PO^*$, donde están en el nicho de la i^{ma} solución de $PO_1 \cup PO^*$.
- n_i será la cardinalidad de las soluciones de PO_1, donde están en el nicho de la i^{ma} solución de $PO_1 \cup PO^*$.
- C será la cardinalidad de las soluciones de $PO_1 \cup PO^*$.

Se define $\gamma = \sum_{i=1}^{C} 1/N_i$ como la suma de los coeficientes que afectan a cada solución. Mientras mas concentrado es la región del espacio de la solución, menor serán los coeficientes de su solución.

A continuación, se aplica la siguiente fórmula para evaluar la entropía E de PO_1 relativa al espacio ocupado por PO^*,

$$E(PO_1, PO_2) = -\frac{1}{\log \gamma} \sum_{i=1}^{C} \left(\frac{1}{N_i} \frac{n_i}{C} \log \frac{n_i}{C}\right)$$

Los valores de la entropía varían entre el 0 y el 1, mientras más cercano a 1 sea el valor mejor diversificado será el frente de Pareto.

3.2.7 Hipervolumen.

Este indicador aparece por primera vez en (E. Zitzler, 1998).Se dedica a medir el área que cubre la aproximación de la frontera de Pareto dado por algún algoritmo. Para delimitar el área, se utiliza un punto de referencia para cada objetivo. Mientras más grande sea el hipervolumen, este indicara mejor convergencia y un mejor cubrimiento de la frontera de Pareto óptima. Esta métrica calcula el volumen (en el espacio de objetivos) cubiertos por miembros de un conjunto Q, de soluciones no dominadas para el problema han de ser minimizados. Para cada $i \in Q$, se construye un hipercubo v_i con un punto de referencia W y la solución i que definen la diagonal de mismo. El punto W se puede obtener simplemente con los peores valores de las funciones objetivo. Entonces, la unión de todos los hipercubos es lo que define el hipervolumen (HV):

$$HV = Volumen\left(\bigcup_{i=1}^{|Q|} v_i\right)$$

3.3 JavaPlot

(Panayotis, 2007) Define Javaplot como una librería programada en Java para la comunicación con GNUplot (T. Williams, 2015). Puede ser usado como una manera de crear gráficos Gnu a través de comandos de Java. En contraste de librerías Gnuplots en Java comunes, esta librería usa estructuras que guardan varios parámetros del ploteo, incluyendo la información de la gráfica, en este caso llamaremos Data. Por otra parte, contiene flexibilidad para dar parámetros especiales a Gnuplot, incluso si la librería (aún) no es soportada. Además, se puede usar las excepciones de Java para informar al usuario si algo salió mal.

Los requerimientos son:

- Java 1.5 o mejor.
- GNUplot 4.2 o mejor.

Javaplot fue testeado en Linux, Windows, MacOS.

3.4 Plugins

Se define plugin (internetglosario, 2015) a una aplicación que tiene la capacidad de añadir nuevas funcionalidades o características al software. Su significado es Plug-in (del inglés enchufable o inserción), otros nombres son, add-ons, conector o extensión. Estos subprogramas se instalan al igual que cualquier otro software, pero no funcionan por su cuenta. Siempre necesitan trabajar dentro de un programa. La aplicación principal entrega métodos que el complemento pueda utilizar, incluyendo como se registran a sí mismos y un protocolo de intercambio de datos. Los complementos dependen de los servicios prestados por la aplicación de acogida y no suelen funcionar por sí mismos.

3.5 Poi Java

HSSF (Horrible SpreadSheet Format, formato de hoja horrible) es la implementación Java pura del Proyecto POI del formato '97 (2007) archivo de Excel.

HSSF (Apache, 2015) proporciona métodos de leer las hojas de cálculo: crear, modificar, leer y escribir hojas de cálculo XLS.

Sus principales características son:

- Estructuras de bajo nivel para las personas con necesidades especiales.
- Un api *eventmodel* para el acceso de sólo lectura eficiente.
- Un api *UserModel* completo para crear, leer y modificar archivos XLS.

HSSF permite leer variables numéricas, cadenas, fecha o teléfonos formula valores que se escriben o leen desde un archivo XLS. También se puede las dimensiones en filas y columnas, el estilo de celdas (negrita, cursiva, bordes, etc.). dispone de una API basada en eventos para la lectura y otra para la escritura de archivos XLS.

3.6 Netbeans

(Oracle, 2015), define netbeans una IDE (Integrated Development Environment, en castellano ambiente de desarrollo integrado) como programa para diseñar el sistema. Especializado en Java, aunque igual posee otros lenguajes para programar, tiene un desarrollo visual que lo hace más sencillo para la creación de las ventanas, botones, etc. Su fácil entendimiento hace que esta herramienta sea la más idónea para el desarrollo de la aplicación en cuestión.

3.7 Metodología evolutiva

Según (Sommerville, 2005) los métodos evolutivos son modelos iterativos, permiten desarrollar versiones cada vez más completas y complejas, hasta llegar al objetivo final deseado; incluso evolucionar más allá, durante la fase de operación. Los modelos "Iterativo Incremental" y "Espiral" (entre otros) son dos de los más conocidos y utilizados del tipo evolutivo.

La idea detrás de este modelo es el desarrollo de una implantación del sistema inicial, exponerla a los comentarios del usuario, refinarla en N versiones hasta que se desarrolle el sistema adecuado. Una ventaja de este modelo es que se obtiene una rápida realimentación del usuario, ya que las actividades de especificación, desarrollo y pruebas se ejecutan en cada iteración.

4 Capítulo: Especificación de requisitos del sistema

A continuación, se detallara la especificación relacionada al nuevo sistema, llamado *Asmop* (analysis system problems Multi-objective Optimization), cuyo significado en castellano es sistema de análisis de problemas de Optimización Multi-objetivo.

4.1 Introducción

4.1.1 Propósito

En el presente apartado se darán a conocer las funcionalidades y restricciones de la aplicación "Asmop", para la documentación de la presente especificación de requisitos se ha seguido el estándar IEEE-STD-830-1998.

4.1.2 Ámbito del sistema

El propósito de la aplicación "Asmop", cuyas funcionalidades se describen en el presente documento, es la descripción cuantitativa y cualitativamente sobre soluciones de problemas de multi-objetivo.

4.2 Descripción general

4.2.1 Características de los usuarios

Los usuarios de la aplicación corresponden a todas que usan programas para analizar soluciones de problemas de multi-objetivo.

4.2.2 Restricciones

La aplicación está diseñada para ser corrida en Windows, Linux y Mac.

4.3 Requisitos Funcionales

4.3.1 Funcionalidades Plot

Plot2D

Descripción: Permite graficar un arreglo de datos en 2 dimensiones de un arreglo de datos seleccionado en la lista cargada en el sistema.

Entradas: arreglo de datos, posición de archivo seleccionado en la lista, opción grilla, selección de opción de diseño, rango de los ejes, opción de los ejes.

Proceso:

- Verifica que opción de diseño fue seleccionada, para definir el aspecto de la línea del gráfico.

- Lee los datos de los ejes, define las leyendas de los ejes.
- Verifica la opción de la grilla, para agregarla en caso de que la opción este activada.
- Si la opción de los ejes es válida, delimita el rango de los ejes en el gráfico.

Salida: Pantalla de Gnuplot con el gráfico correspondiente.

Plot3D

Descripción: Permite graficar un arreglo de datos en 3 dimensiones de un arreglo de datos seleccionado en la lista cargada en el sistema.

Entradas: Arreglo de datos, posición de archivo seleccionado en la lista, opción grilla, selección de opción de diseño, rango de los ejes, opción activar rango de ejes en 3D.

Proceso:

- Verifica que opción de diseño fue seleccionada, para definir el aspecto de la línea del gráfico.
- Si esta la opción activar rango de ejes en 3D esta seleccionada, se activa la opción de límites de las ejes.
- Lee los datos, define las leyendas de los ejes.
- Verifica la opción de la grilla, para agregarla en caso de que la opción este activada.
- Si la opción de los ejes es válida, delimita el rango de los ejes en el gráfico.

Salida: Pantalla de Gnuplot con el gráfico correspondiente.

- **ploteo2DMul**

Entradas: Permite graficar un arreglo de datos en 2 dimensiones, seleccionado en la lista cargada en el sistema, dependiendo de las opciones de graficado.

Entradas: arreglo de datos, posición de archivo seleccionado en la lista, opción separar, opción grilla, selección de opción de diseño, rango de los ejes.

Proceso:

- Verifica que opción de diseño fue seleccionada, para definir el aspecto de la línea del gráfico.

- Si es verdadera la opción separar, se graficarán los datos en gráficos diferentes.
    - De caso contrario, se graficarán los datos en un solo gráfico.
- Lee los datos de los ejes, define las leyendas de los ejes.
- Si la opción de la grilla es válida, se agregará al gráfico la grilla.
- Si la opción de los ejes es válida, delimita el rango de los ejes en el gráfico.

Salida: Pantalla de Gnuplot con el gráfico correspondiente.

- **ploteo3DMul**

Descripción: Permite graficar varios arreglos de datos en 3D, seleccionados en la lista.

Entradas: arreglo de datos, posición de archivo seleccionado en la lista, opción grilla, selección de opción de diseño, rango de los ejes, opción activar rango de ejes en 3D.

Proceso:

- Verifica que opción de diseño fue seleccionada, para definir el aspecto de la línea del gráfico.
- Si es verdadera la opción Separar, se graficarán los datos en gráficos diferentes
    - De caso contrario, se graficarán los datos en un solo gráfico.
- Lee los datos de los ejes, define las leyendas de los ejes.
- Si la opción de la grilla es válida, se agregará al gráfico la grilla.
- Si esta la opción activar rango de ejes en 3D seleccionada, se activa la opción de límite de las ejes.
- Si la opción de los ejes es válida, delimita el rango de los ejes en el gráfico.

Salida: Pantalla de Gnuplot con el gráfico correspondiente.

- **ploteo2Dpng**

  Descripción: Permite guardar un gráfico 2D en un archivo .png de un arreglo de datos seleccionado en la lista.

  Entradas: arreglo de datos, posición de archivo seleccionado en la lista, opción grilla, selección de opción de diseño, rango de los ejes.

  Proceso:

  - Verifica que opción de diseño fue seleccionada, para definir el aspecto de la línea del gráfico.
  - Lee los datos de los ejes, define las leyendas de los ejes.
  - Si la opción de la grilla es válida, se agregará al gráfico la grilla.
  - Si la opción de los ejes es válida, delimita el rango de los ejes en el gráfico.
  - Se abre una ventana donde pide donde guardar el archivo.
    - Si el archivo existe, preguntara si desea sobrescribir.

  Salida: Archivo png.

- **ploteo3Dpng**

  Descripción: Permite guarda un gráfico 3D en un archivo .png de un arreglo de datos seleccionado en la lista.

  Entradas: arreglo de datos, posición de archivo seleccionado en la lista, opción grilla, selección de opción de diseño, rango de los ejes, opción activar rango de ejes en 3D.

  Proceso:

  - Verifica que opción de diseño fue seleccionada, para definir el aspecto de la línea del gráfico.
  - Si esta la opción activar rango de ejes en 3D seleccionada, se activa la opción límites de las ejes.
  - Lee los datos de los ejes, define las leyendas de los ejes.
  - Si la opción de la grilla es válida, se agregará al gráfico la grilla.
  - Si la opción de los ejes es válida, delimita el rango de los ejes en el gráfico.
  - Se abre una ventana donde pide donde guardar el archivo.
    - Si el archivo existe, preguntara si desea sobrescribir.

  Salida: Archivo png.

- **ploteo2DMulPng**

  Descripción: Permite guardar un multigráfico 2D en un archivo .png de un arreglo de datos seleccionado en la lista.

  Entradas: arreglo de datos, posición de archivo seleccionado en la lista, opción grilla, selección de opción de diseño, rango de los ejes.

  Proceso:

  - Verifica que opción de diseño fue seleccionada, para definir el aspecto de la línea del gráfico.
  - Si es verdadera la opción Separar, se graficarán los datos en gráficos diferentes
    - De caso contrario, se graficarán los datos en un solo gráfico.
  - Lee los datos de los ejes, define las leyendas de los ejes.
  - Si la opción de la grilla es válida, se agregará al gráfico la grilla.
  - Si la opción de los ejes es válida, delimita el rango de los ejes en el gráfico.
  - Se abre una ventana donde pide donde guardar el archivo.
    - Si el archivo existe, preguntara si desea sobrescribir.

  Salida: Archivo png.

- **ploteo3DMulPng**

  Descripción: Permite guardar un multigráfico 3D en un archivo .png de un arreglo de datos seleccionado en la lista.

  Entradas: arreglo de datos, posición de archivo seleccionado en la lista, opción grilla, selección de opción de diseño, rango de los ejes, opción activar rango de ejes en 3D.

  Proceso:

  - Verifica que opción de diseño fue seleccionada, para definir el aspecto de la línea del gráfico.
  - Si es verdadera la opción Separar, se graficarán los datos en gráficos diferentes
    - De caso contrario, se graficarán los datos en un solo gráfico.
  - Lee los datos de los ejes, define las leyendas de los ejes.
  - Si la opción de la grilla es válida, se agregará al gráfico la grilla.

- Si esta la opción activar rango de ejes en 3D seleccionada, se activa la opción de límites de las ejes.
- Si la opción rango de los ejes es válida, delimita el rango de los ejes en el gráfico.
- Se abre una ventana donde pide donde guardar el archivo.
    - Si el archivo existe, preguntara si desea sobrescribir.

Salida: Archivo png.

- **ploteo2DEps**

Descripción: Permite guardar un gráfico 2D en un archivo .eps de un arreglo de datos seleccionado en la lista.

Entradas: arreglo de datos, posición de archivo seleccionado en la lista, opción grilla, selección de opción de diseño, rango de los ejes.

Proceso:

- Verifica que opción de diseño fue seleccionada, para definir el aspecto de la línea del gráfico.
- Lee los datos de los ejes, define las leyendas de los ejes.
- Si la opción de la grilla es válida, se agregará al gráfico la grilla.
- Si la opción de los ejes es válida, delimita el rango de los ejes en el gráfico.
- Se abre una ventana donde pide donde guardar el archivo.
    - Si el archivo existe, preguntara si desea sobrescribir.

Salida: Archivo eps.

- **ploteo3DEps**

Descripción: Permite guardar un gráfico 3D en un archivo .eps de un arreglo de datos seleccionado en la lista.

Entradas: arreglo de datos, posición de archivo seleccionado en la lista, opción grilla, selección de opción de diseño, rango de los ejes.

Proceso:

- Verifica que opción de diseño fue seleccionada, para definir el aspecto de la línea del gráfico.
- Si esta la opción activar rango de ejes en 3D seleccionada, se activa la opción límites de las ejes.
- Lee los datos de los ejes, define las leyendas de los ejes.

- Si la opción de la grilla es válida, se agregará al gráfico la grilla.
- Si la opción de los ejes es válida, delimita el rango de los ejes en el gráfico.
- Se abre una ventana donde pide donde guardar el archivo.
    - Si el archivo existe, preguntara si desea sobrescribir.

Salida: Archivo eps.

- **ploteo2DMulEps**

Descripción: Permite guardar un multigráfico 2D en un archivo .eps de un arreglo de datos seleccionado en la lista.

Entradas: arreglo de datos, posición de archivo seleccionado en la lista, opción grilla, selección de opción de diseño, rango de los ejes.

Proceso:

- Verifica que opción de diseño fue seleccionada, para definir el aspecto de la línea del gráfico.
- Si es verdadera la opción Separar, se graficarán los datos en gráficos diferentes
    - De caso contrario, se graficarán los datos en un solo gráfico.
- Lee los datos de los ejes, define las leyendas de los ejes.
- Si la opción de la grilla es válida, se agregará al gráfico la grilla.
- Si la opción de los ejes es válida, delimita el rango de los ejes en el gráfico.
- Se abre una ventana donde pide donde guardar el archivo.
    - Si el archivo existe, preguntara si desea sobrescribir.

Salida: Archivo eps.

- **Ploteo3DMulEps**

Descripción: Permite guardar un multigráfico 3D en un archivo .eps de un arreglo de datos seleccionado en la lista.

Entradas: arreglo de datos, posición de archivo seleccionado en la lista, opción grilla, selección de opción de diseño, rango de los ejes, opción activar rango de ejes en 3D.

Proceso:

- Verifica que opción de diseño fue seleccionada, para definir el aspecto de la línea del gráfico.
- Si es verdadera la opción Separar, se graficarán los datos en gráficos diferentes
    - De caso contrario, se graficarán los datos en un solo gráfico.

- Lee los datos de los ejes, define las leyendas de los ejes.
- Si la opción de la grilla es válida, se agregará al gráfico la grilla.
- Si esta la opción activar rango de ejes en 3D seleccionada, se activa la opción de límite de las ejes.
- Si la opción de los ejes es válida, delimita el rango de los ejes en el gráfico.
- Se abre una ventana donde pide donde guardar el archivo.
    - Si el archivo existe, preguntara si desea sobrescribir.

Salida: Archivo eps.

4.3.2 Funcionalidades de LoadTxt

- **LargoTxt**

  Descripción: Calcula el largo de las filas de un archivo de texto.

  Entradas: FileInputStream.

  Proceso:

  - Calcula el tamaño del largo total del archivo.
    - Si una línea esta vacia, desplega el mensaje "Ocurrió un error en el largo".

  Salida: entero

- **AnchoTxt**

  Descripción: Calcula el ancho de las columnas de archivo de texto.

  Entradas: FileInputStream.

  Proceso:

  - Calcula el tamaño del ancho total del archivo.
    - Si alguno de los anchos no cuadra con los demás se desplegará el mensaje "Error Tamaño".

  Salida: entero.

- **FillTxt**

  Descripción: Procesa el archivo de texto a la estructura datos.

Entradas: Archivo, Largo, Ancho.

Proceso:

- Revisa el archivo de texto fila en fila, estos se cargan en una variable de tipo doublé para su almacenaje.
    - Verifica si algún dato es un carácter o símbolo, en caso de ser, envía un dialogo de error.

Salida: variable de tipo double.

4.3.3 Funcionalidades de LoadXls.

- **SizeX**

  Descripción: Calcula el largo de un archivo de excel.

  Entradas: List.

  Proceso:

  - Devuelve el valor del ancho del xls.

  Salida: entero.

- **SizeY**

  Descripción: Calcula el largo del archivo Excel.

  Entradas: variable list.

  Proceso:

  - Devuelve el valor del largo del archivo Excel.
    - Verifica si es existe algún espacio vacío, de ser así, se desplegará un dialogo de error.

  Salida: Variable de tipo entero.

- **ShowExelData**

  Descripción: Procesa el archivo de Excel.

Entradas: variable list.

Proceso:

- Carga la lista para ser validada y posteriormente se almacena en datos.

Salida: Datos.

- **CargarExcel**

  Descripción: carga el archivo Excel a la variable Sheetdata.

  Entradas: variable list, arreglo de datos.

  Proceso:

  - Lee el archivo seleccionado, pasando toda la información en una lista.

  Salida: variable list.

  Funcionalidades de DG

- **DistanceG**

  Descripción: Calcula la distancia generacional de dos datos.

  Entradas: arreglo de dato a, arreglo de dato b.

  Proceso:

  - Se calcula la distancia generacional del dato a con respecto al dato b.

  Salida: Dialogo con los resultados.

- **ErrorRatio**

  Descripción: Calcula el radio de error de dos arreglos de datos.

  Entradas: Arreglo de dato a, arreglo de dato b.

  Proceso:

  - Se calcula el radio de error del arreglo de dato a con respecto al arreglo de dato b.

  Salida: Dialogo con los resultados.

- **Spacing**

  Descripción: calcula el espacio de un arreglo de datos.

  Entradas: Arreglo de datos.

  Proceso:

  - Se calcula el espaciado de los datos de la lista.

  Salida: Dialogo con los resultados.

4.3.4 Funcionalidades de PluginDemo

- **GetPlugins**

  Descripción: Lee la carpeta de plugins y carga la lista

  Entradas: Enrutamiento del programa.

  Proceso:

  - Lee los archivos contenidos en la carpeta de plugins, para generar una lista con los plugins existentes.
  - Solo cargará archivos con la extensión .class.
    - En caso de error se desplegará un mensaje de error con su descripción.

  Salida: PluginFunction.

- **RunPlugins**

  Descripción: ejecuta un plugin que se ha seleccionado

  Entradas: Lista, variable Text, variable de tipo entero, arreglo de datos.

  Proceso:

  - Lee la lista en busca del plugin seleccionado.

- Ejecuta la función principal del plugin seleccionado
    - En caso de error, se desplegará un dialogo con el error relacionado.

Salida: Ventana del plugin.

4.3.5 Funcionalidades de Principal

- **Principal**

  Descripción: Carga la lista de los plugins al inicio del sistema.

  Entradas: Plugindemo.

  Proceso:

  - Se ejecuta la función getplugin.
  - Cada plugin leído sera añadido a la lista de ítem del botón plugins.

  Salida:

- **EscuchaItemMenu**

  Descripción: Ejecuta el plugin seleccionado.

  Entradas: Plugindemo.

  Proceso:

  - Se ejecuta la función runplugin.

  Salida: Dialogo con los resultados.

- **Abrir Archivo**

  Descripción: Abre un archivo dependiendo de la extensión.

  Entradas: Archivo.

Proceso:

- Se desplegará una ventana de selección, para escoger el archivo a procesar
    - Si el archivo es de extensión .txt, se cargaran las funciones de LoadTxt
        - Se verifica si el formato de los datos es válido, de lo contrario se desplegará un mensaje de error.
    - Si el archivo es de extensión .xls, se cargarán las funciones de LoadXls.
        - Se verifica si el formato de los datos es válido, de lo contrario se desplegará un mensaje de error.
    - Si el archivo tiene otra extensión que no sea la antes mencionada, se desplegará un mensaje de error.
- Una vez cargado el archivo, el nombre del archivo se agregará a las lista de datos en la pantalla principal.

Salida: Dialogo con los resultados.

- **Mostrar Datos**

  Descripción: Abre una ventana donde muestra toda información contenida dentro de un dato de la lista del sistema.

  Entradas: Dato seleccionado de la lista

  Proceso:

  - Se desplegará una ventana mostrando toda la información de dato seleccionado, el cual debe ser solo uno.

  Salida: Dialogo con los resultados.

5 Capítulo: Diseño

En este capítulo se verán los diseños de la aplicación en desarrollo.

5.1 Diagrama de Clases

En este diagrama (véase Figura 1) se pueden apreciar todas las clases, métodos y atributos que el sistema contiene.

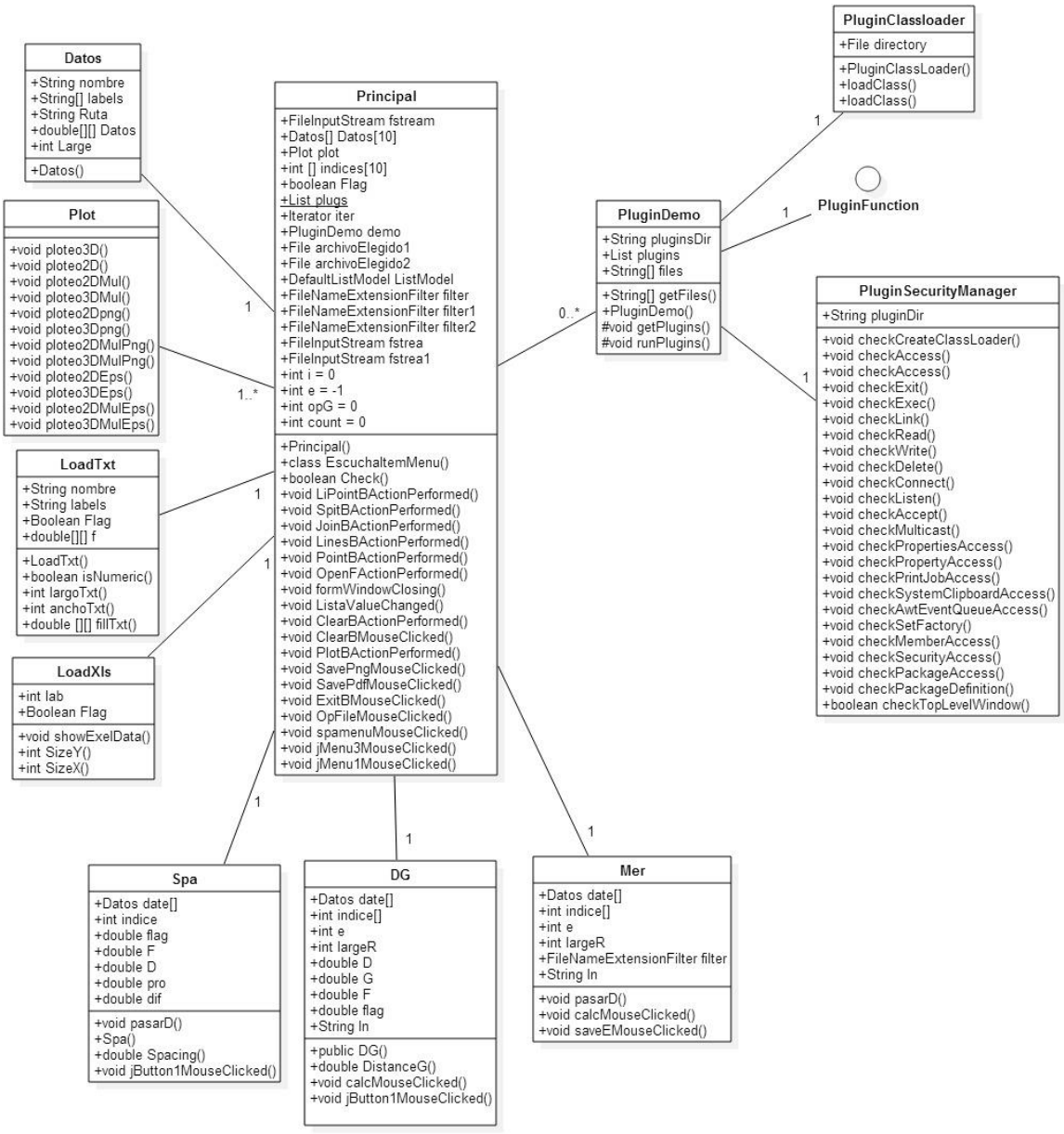

Figura 1: Diagrama de Clases.

5.2 Diagrama de Actividades

En el diagrama que se aprecia en la Figura 2, que representa el flujo de procesos del sistema.

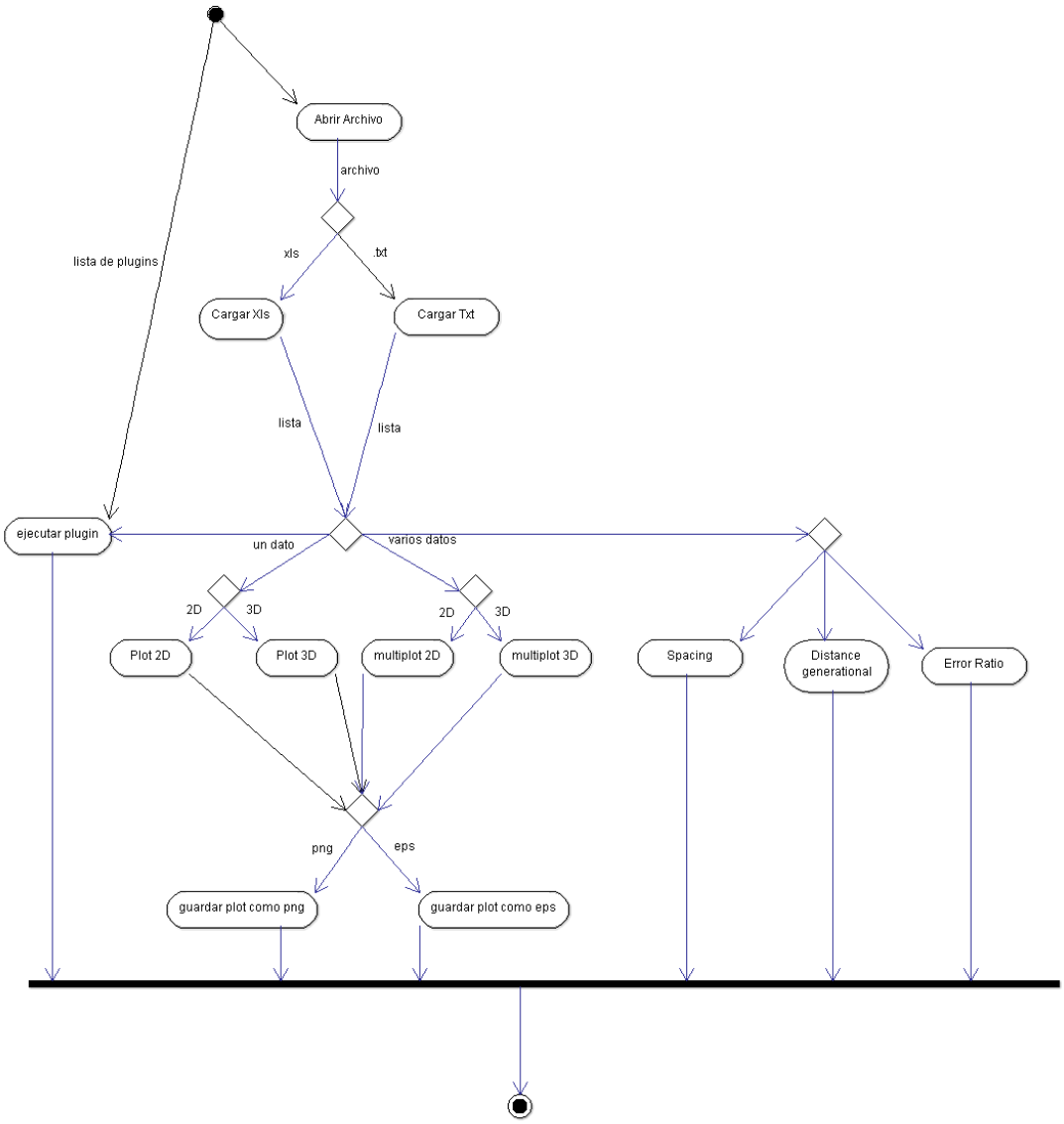

Figura 2: Diagrama de Actividades.

5.3 Interfaz de Usuario

A continuación, se describen las interfaces de usuarios que se ocuparon en el programa Asmop, descrito en la Figura 3.

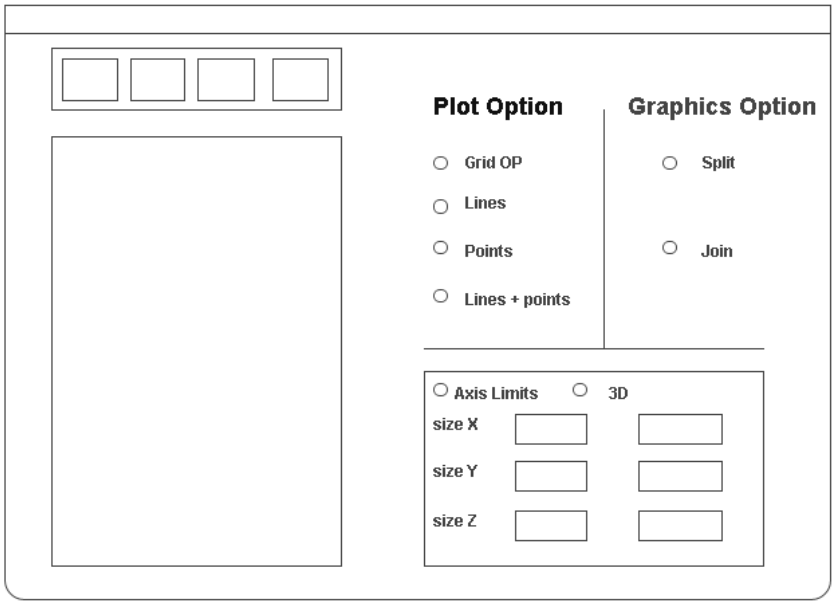

Figura 3: Wireframe Interfaz Principal.

En la Figura 3 se visualiza la pantalla principal del sistema, aquí es donde se maneja todas las funciones más importantes del sistema.

Dando una pantalla como se muestra en la Figura 4.

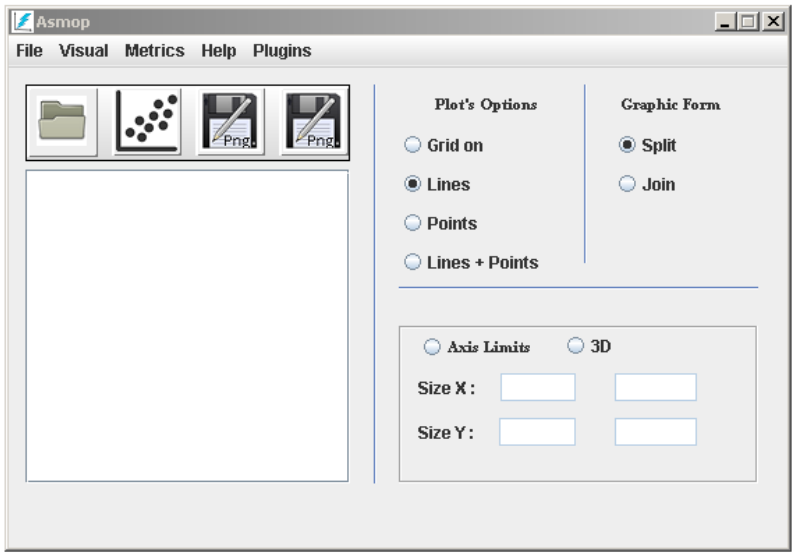

Figura 4: Pantalla Principal.

En la Figura 5 se visualiza la pantalla donde se desplegará la información de la métrica Spacing.

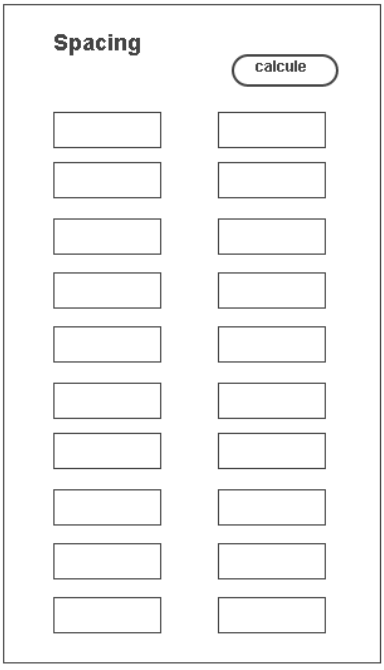

Figura 5: Wireframe Spacing.

La cual se verá como en la Figura 6.

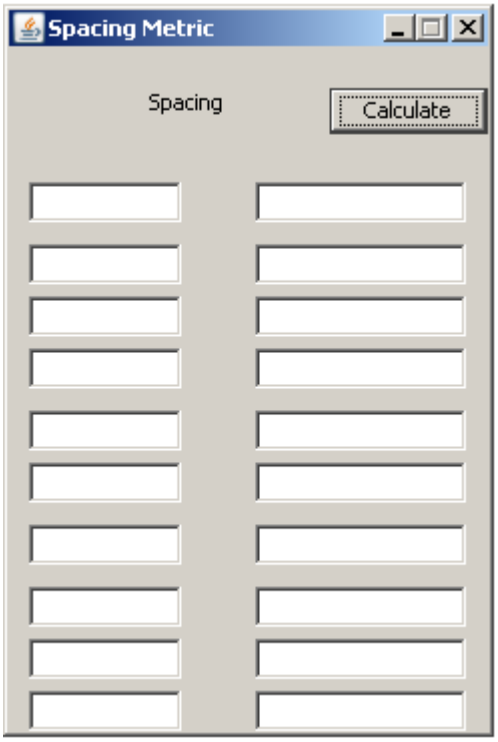

Figura 6: Pantalla Spacing.

En la Figura 7 se visualiza la pantalla donde se desplegará la información de la métrica Distance Generational.

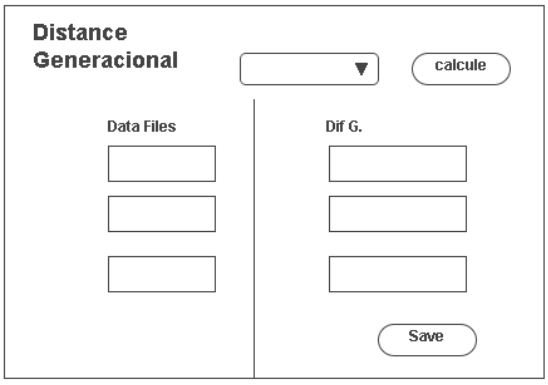

Figura 7: Wireframe Distance Generational.

La pantalla se visualizará como en la Figura 8.

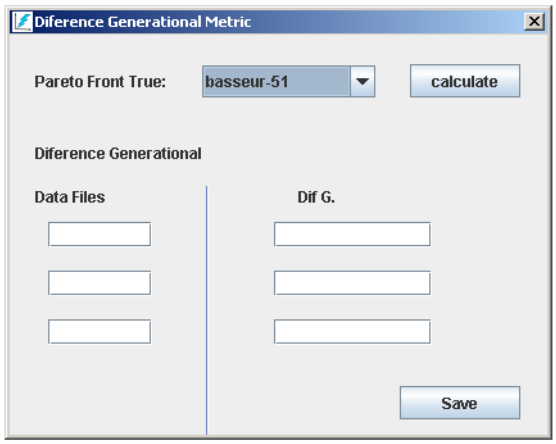

Figura 8: Pantalla Distance Generational.

En la Figura 9 se visualiza la pantalla donde se desplegará la información de la métrica Error Ratio.

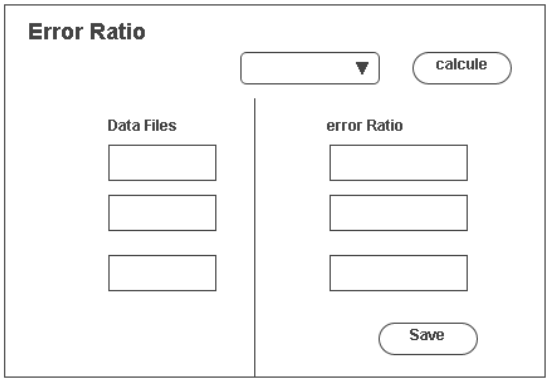

Figura 9: Wireframe Error Ratio.

La pantalla se visualizará como en la Figura 10.

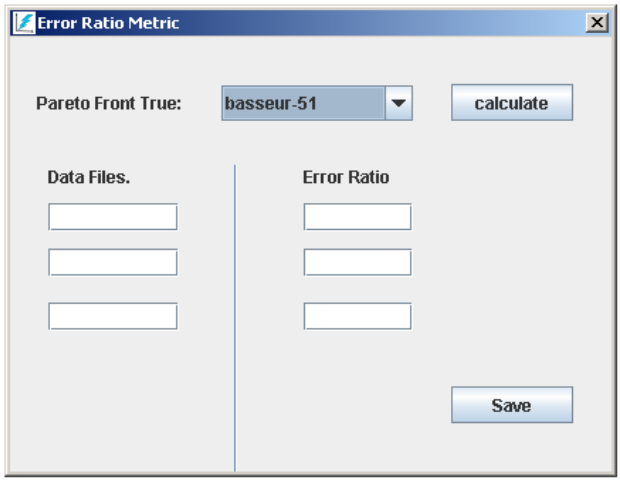

Figura 10: Pantalla Error Ratio.

En la Figura 11 se visualiza la pantalla donde se desplegará la pantalla correspondiente al registro de los nombres de los ejes en 2D.

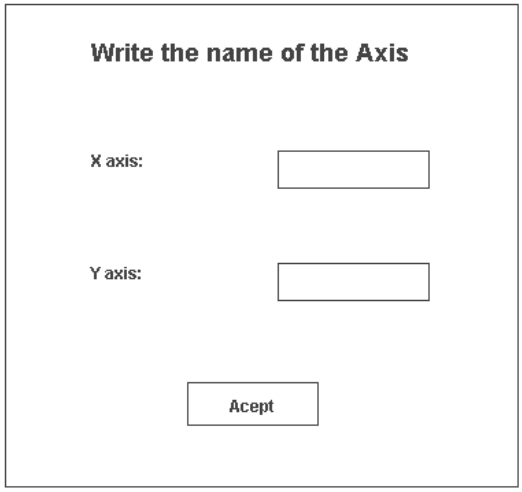

Figura 11: Wireframe Label 2D.

La pantalla se visualizará como en la Figura 12.

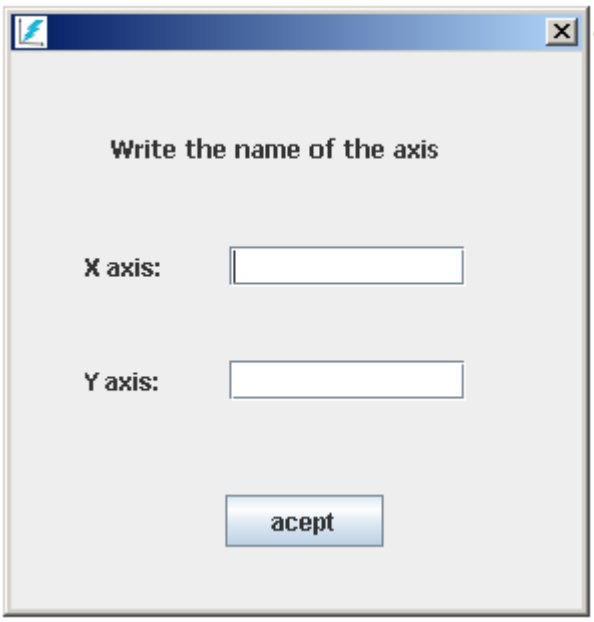

Figura 12: Pantalla Label 2D.

En la Figura 13 se visualiza la pantalla donde se desplegará la pantalla correspondiente al registro de los nombres de los ejes en 3D.

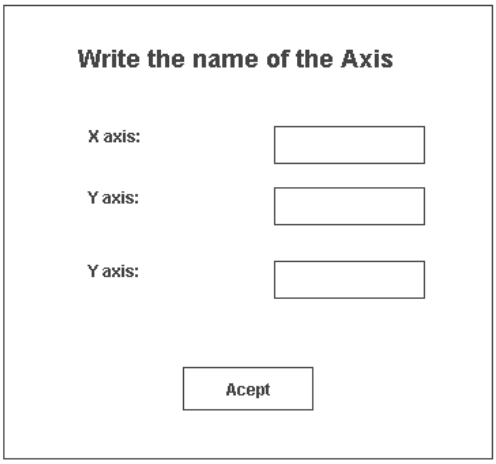

Figura 13: Wireframe Label 3D.

La pantalla se visualizará como en la Figura 14.

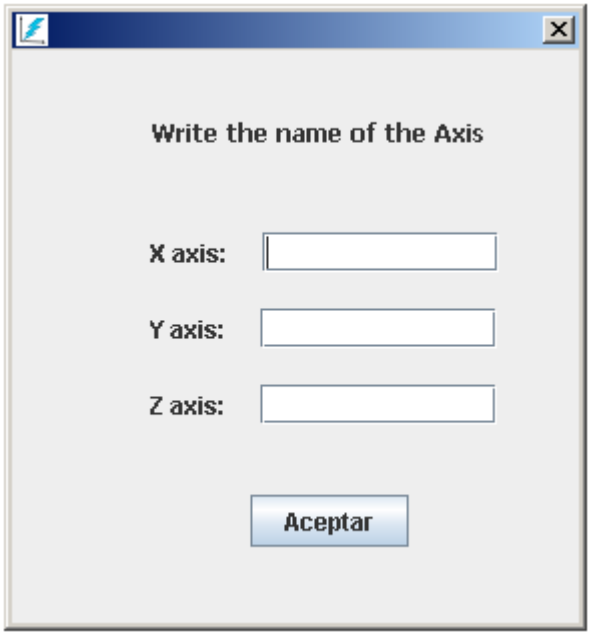

Figura 14: Pantalla Label 3D.

En la figura 15 y 16, se muestran el wireframe de Show y su respectiva pantalla.

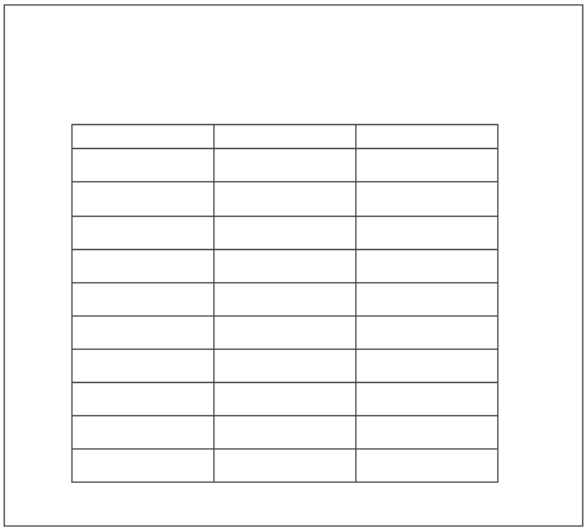

Figura 15: *Wireframe* Show.

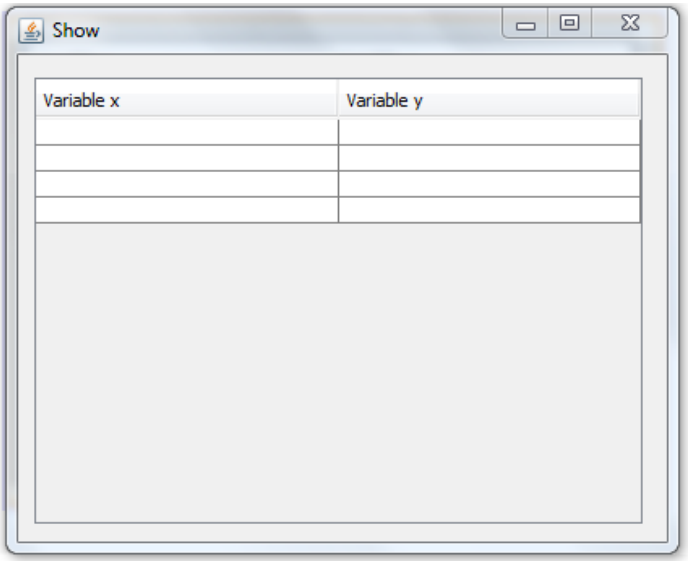

Figura 16: Pantalla Show.

En la figura 17 y 18, se muestran el wireframe de About y su respectiva pantalla.

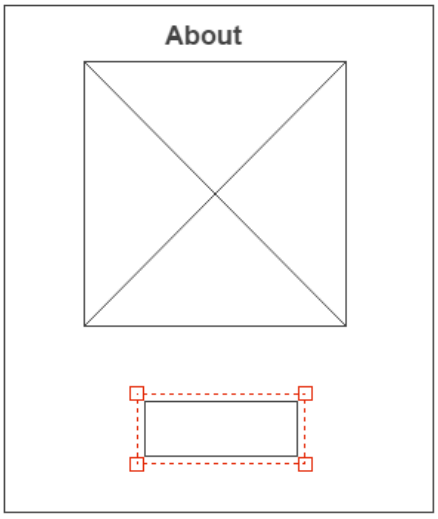

Figura *17*: *Wireframe About.*

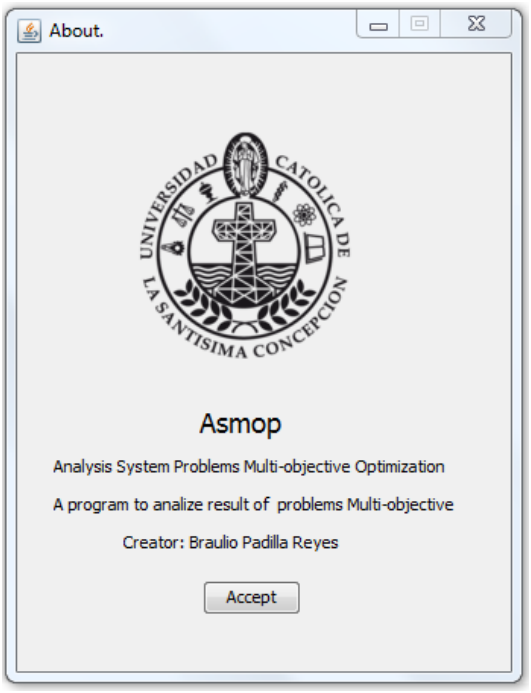

Figura 18: Pantalla About.

6 Capítulo: Desarrollo

En este capítulo se presenta una descripción de las principales componentes del sistema implementado.

Cabe destacar que se creó en base a una metodología evolutiva, donde se generó dos prototipos. El primero consta con las parte de carga de archivos y diseño de los gráficos. La segunda se desarrolló el sistema al punto que se describe en este documento.

6.1 Gnuplot

Para generar la comunicación con gnuplot es necesario de la librería Javaplot.jar. Para agregar librerías se debe seguir los pasos:

Clickeamos nuestro proyecto y luego vamos a propiedades, como se puede apreciar en la Figura 15.

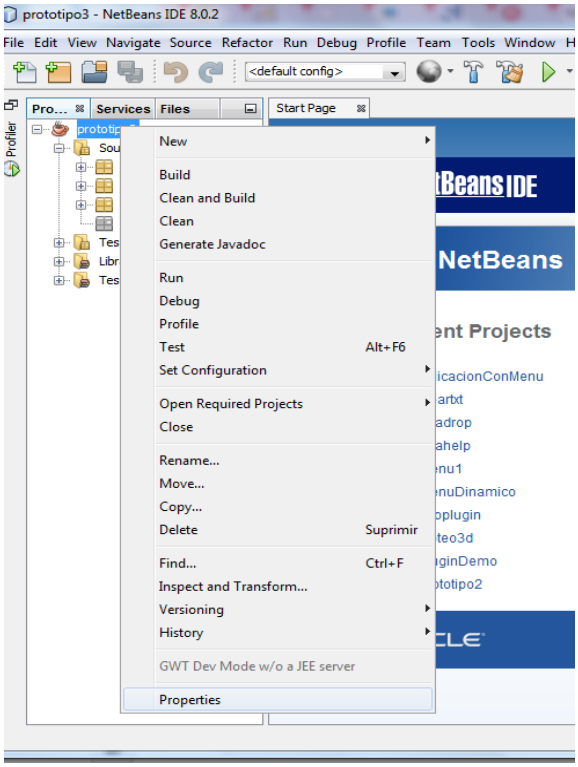

Figura 19: ConFiguración 1.

Luego, agregamos la librería .jar de Javadoc (descargado de http://www.Javaplot.panayotis.com) con el botón ADD JAR Folder,

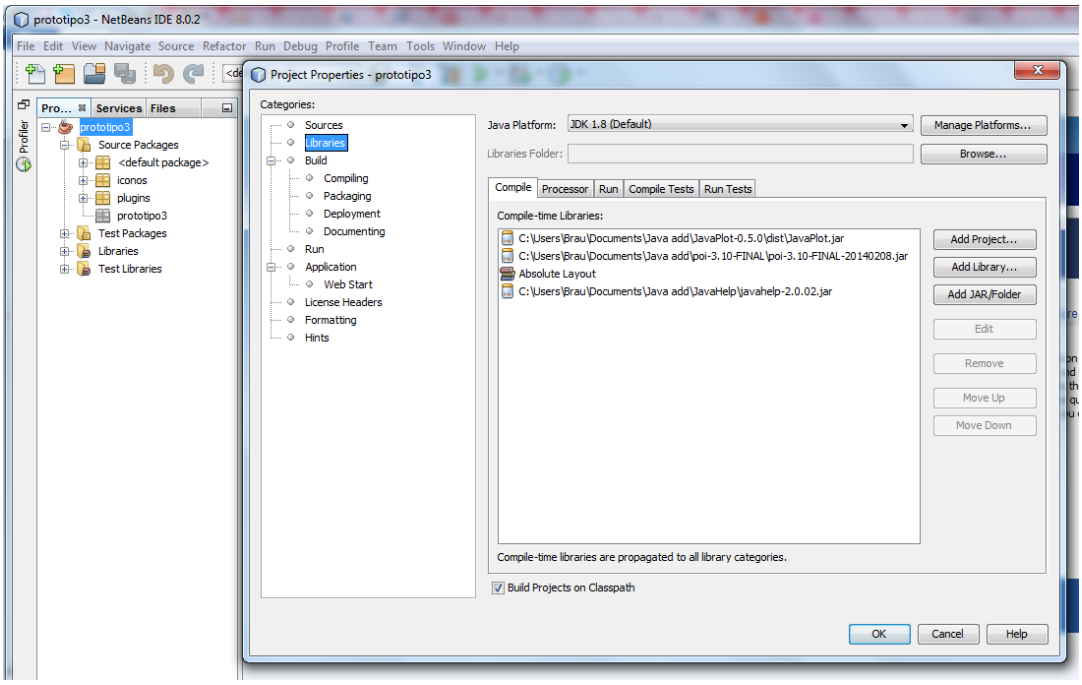

Figura 20: ConFiguración 2.

Y agregamos las librerías en el programa (véase Figura 16).

Con eso, está listo para agregar los comandos necesarios para el ploteo.

En este punto se definirán los comandos más importantes que fueron usados:

PlotStyle, este comando define el estilo de la gráfica, donde podemos definir qué tipo de estilo se pondrá para reflejar la gráfica, en este caso se optó para 3 opciones: línea, puntos y ambos. También usamos este comando para definir el color de la gráfica.

DataSetPlot, este comando genera la matriz de datos para el ploteo.

JavaPlot, Crear una nueva instancia de JavaPlot, con los parámetros por defecto. Si se define *true* como parámetro, se genera un plot en 3D, sin parámetros, se entiende como 2D.

JavaPlot.addPlot, este comando es donde agregamos los datos referentes que existan para generar el gráfico.

JavaPlot.getEjes, en este comando agregamos los nombres de los ejes.

JavaPlot.getEjes. *setBoundaries*, se define los límites de las ejes, es necesario el mínimo y máximo.

JavaPlot.plot aquí es donde se despliega la comunicación de la aplicación Java hacia el GnuPlot.

Se debe entender que como programa externo, solo puede hacer el llamado uno a la vez, esperando que se cierre para abrir otro gráfico.

Debido al modo en que se ejecuta, son necesarias varias funciones dependiendo de la selección de los datos, así como de las dimensiones. Para un dato en 2D se llama la función **ploteo2D**, para 3D se llama a **ploteo3D.** Para los casos en que se selecciona más de un elemento de la lista, para 2D se usa la función **ploteo2DMul**; para 3D se usa la función **ploteo3DMul**.

Ahora, para guardar gráficos los comandos más importantes son:

ImageTerminal, genera el tipo de archivo que guardara el gráfico.

JavaPlot.setTerminal, este comando agrega la información que donde se genera el gráfico, cuando se ejecuta *JavaPlot.plot* en vez de llamar a gnuplot como ventana externa, simplemente guarda el archivo descrito en *ImageTerminal.*

Como se comentó antes, dependiendo los datos se utiliza las funciones, en este punto se agrega otro aspecto, el tipo de archivo que se desea guardar, para dar más comodidad al usuario en el manejo de los archivos de salida. Existen dos tipos de archivos propuestos, el PNG (Portable Network Graphics) y el EPS (Encapsulated PostScript). Para el guardado de archivo png están las funciones **ploteo2Dpng, ploteo3Dpng, ploteo2DMulPng, ploteo3DMulPng.** Para archivos eps están **ploteo2DEps, ploteo3DEps, ploteo2DMulEps, ploteo3DMulEps.**

6.2 Carga de Archivos

Para la carga de archivos al sistema, hay dos tipos de archivos admitidos, XLS y TXT. Ambos siguen ciertos formatos para poder ser leídos respectivamente, que se describirán a continuación.

6.2.1 Carga de Archivos de Texto

Para esta carga archivos de texto se tiene las funciones: **largoTxt**, **anchoTxt, fillTxt.**

Dentro de estas funciones, los principales comandos se describen a continuación,

DataInputStream, este comando lee continuamente un dato de entrada, permite a una aplicación leer tipos de datos Java primitivos de un flujo de entrada subyacente de una manera independiente de la máquina.

BufferedReader, este comando lee la información de *DataInputStream*, creando un buffer para los caracteres, con el fin que la lectura de caracteres, líneas y arreglos sea eficiente.

BufferedReader.readLine, este comando lee línea por línea los datos almacenados en el buffer.

StringTokenizer, la clase tokenizer permite romper una cadena en tokens (pequeños grupos de datos que representan un conjunto de información mayor previamente establecida).

El formato de .txt de ser de la forma descrita en la Figura 17

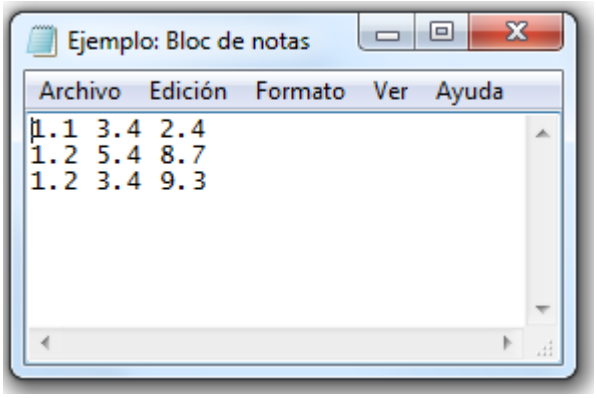

Figura 21: Formato de texto.

Espacio entre las columnas y para los decimales se tolera el punto y no comas. Las funcionalidades del carga txt se encarga de verificar el largo y ancho del block de notas, así de como la información, donde solo será aceptado dígitos y no caracteres. Las dimensiones se limitan hasta un máximo de 6 columnas. Para 2 y 3D, se desplegará una ventana de dialogo pidiendo los nombres de las ejes, para dimensiones mayores de 3D, se asignaran valores default para las dimensiones. La importancia para 2D y 3D es debido que estas pueden ser graficadas, donde tienen importancia.

6.2.2 Carga de Archivos Excel 2007

Para la carga de datos de Excel (2007), se describió anteriormente las funciones **SizeX, SizeY, showExelData, CargarExcel.** Como se explicó con el caso de gnuplot, la carga de archivos depende de la librería poi-3.10-FINAL-20140208.jar (descargado de http://poi.apache.org/).

Dentro de los comandos más relevantes, detallamos los siguientes,

HSSFWorkbook, este comando carga el libro de Excel.

HSSFSheet, carga una determinada hoja del libro de Excel seleccionado.

HSSFRow, carga una determinada fila de la hoja del libro.

HSSFCell, representa una determinada celda de una fila, Las celdas pueden ser numéricas, basadas en una fórmula o basadas en cadena (texto).

Para el formato de la hoja de Excel, hay dos formas, en la Figura 18 se muestra la primera.

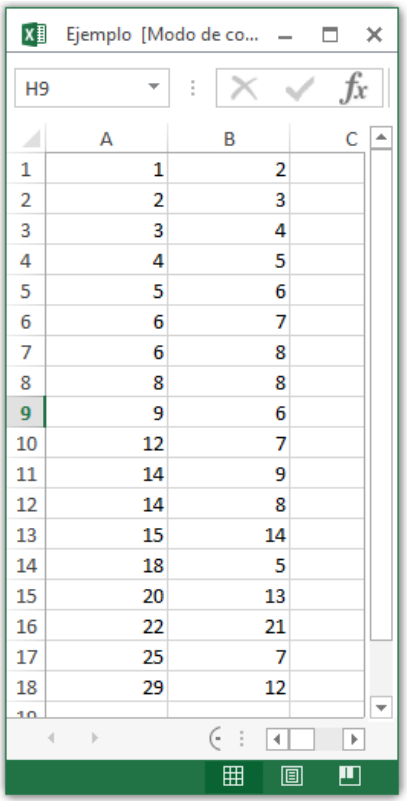

Figura 22: Formato de xls A.

Y en la Figura 19 la forma dos.

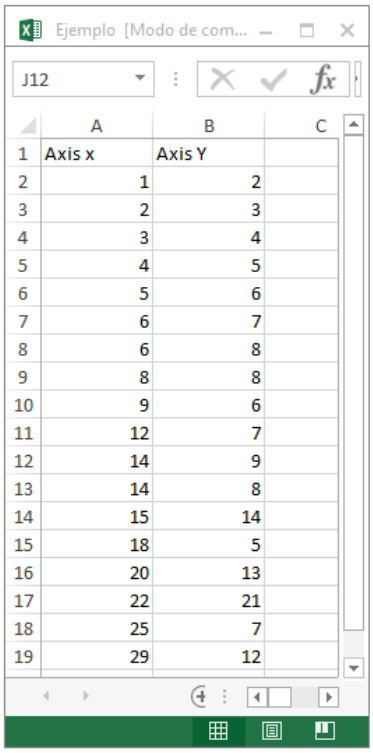

Figura 23: Formato de xls B.

Para el caso de carga de xls, con comandos es detectable cuando la primera línea es tipo carácter, así se guardan en las variables de los nombres de las ejes. Cuando no posea esa información en el xls, se desplegará la pantalla de registro de los nombres de los ejes. Se chequea que las dimensiones sean válidas, de lo contrario se desplegará un mensaje de error.

6.3 Métricas

Dentro del sistema se añadió 3 métricas estáticas de las cuales son:

6.3.1 Error Ratio

Se creó un algoritmo que calcula la métrica de radio de error entre dos datos dentro de la lista previamente cargada en el sistema. Las dimensiones de los datos genera una selección a través de sus dimensiones, con un máximo de 6 dimensiones especificado anteriormente. Si las dimensiones de la selección no son las mismas, desplegarán un mensaje de error.

Se desplegará una ventana de tipo dialogo con las soluciones, con la opción de poder guardar los resultados, que serán guardados en un block de notas. En la ventana el *listbox* representa cuál de los datos seleccionado será considerado como el PF_{true}.

La métrica se describe en la Figura 20.

```
public double ErrorRatio(Datos a, Datos b){
float e=0;boolean Flag=false;
switch(a.labels.length){
    case 2:
        for(int i=0; i<a.Large;i++){
            for(int j=0; j<b.Large;j++){
                if(a.Datos[i][0]==b.Datos[j][0]&&a.Datos[i][1]==b.Datos[j][1]){
                    Flag= true;
                    j= b.Large;
                    }else{
                          Flag= false;}}
       if(Flag==false){
       e=e+1;}}
        e=e/b.Large;
break;
    case 3:
        for(int i=0; i<a.Large;i++){
            for(int j=0; j<b.Large;j++){
                if(a.Datos[i][0]==b.Datos[j][0]&&a.Datos[i][1]==b.Datos[j][1]&&
                   a.Datos[i][2]==b.Datos[j][2]){
                    Flag= true;
                    j= b.Large;
                    }else{
                          Flag= false;}}

       if(Flag==false){
       e=e+1;}}
        e=e/b.Large;
break;
    case 4:
        for(int i=0; i<a.Large;i++){
            for(int j=0; j<b.Large;j++){
                if(a.Datos[i][0]==b.Datos[j][0]&&a.Datos[i][1]==b.Datos[j][1]&&
                   a.Datos[i][2]==b.Datos[j][3]&&
                   a.Datos[i][2]==b.Datos[j][3]){
                    Flag= true;
```

```
                    j= b.Large;
                    }else{
                        Flag= false;}}
      if(Flag==false){
      e=e+1;}}
       e=e/b.Large;
break;
   case 5:
       for(int i=0; i<a.Large;i++){
           for(int j=0; j<b.Large;j++){
               if(a.Datos[i][0]==b.Datos[j][0]&&a.Datos[i][1]==b.Datos[j][1]&&
                  a.Datos[i][2]==b.Datos[j][2]&&a.Datos[i][3]==b.Datos[j][3]&&
                  a.Datos[i][4]==b.Datos[j][4]){
                   Flag= true;
                   j= b.Large;
                   }else{
                        Flag= false;}}
      if(Flag==false){
      e=e+1;}}
       e=e/b.Large;
break;
   case 6:
     for(int i=0; i<a.Large;i++){
           for(int j=0; j<b.Large;j++){
               if(a.Datos[i][0]==b.Datos[j][0]&&a.Datos[i][1]==b.Datos[j][1]&&
                  a.Datos[i][2]==b.Datos[j][2]&&a.Datos[i][3]==b.Datos[j][3]&&
                  a.Datos[i][4]==b.Datos[j][4]&&a.Datos[i][5]==b.Datos[j][5]){
                   Flag= true;
                   j= b.Large;
                   }else{
                        Flag= false;}}
      if(Flag==false){
      e=e+1;}}
       e=e/b.Large;
break;}
return e;}
```

Figura 24: Código Métrica Error Ratio.

6.3.2 Spacing

Se creó un algoritmo que calcula la métrica de espaciado entre dos datos dentro de la lista previamente cargada en el sistema. Los casos del operador switch son las dimensiones de los datos seleccionados, con un máximo de 6 dimensiones especificado anteriormente. Si las dimensiones de la selección no son las mismas, desplegarán un mensaje de error.

Se desplegará una ventana de tipo dialogo con las soluciones de todos los datos de la lista, con la opción de poder guardar los resultados, que serán guardados en un block de notas. Cabe destacar, al apretar el botón de calcular, se mostrara el resultado de la métrica de la cantidad de la lista, ocultando las casillas vacías.

```
public double Spacing(Datos a){
    flag=0;D=0;pro=0;dif=0;
    double d[]= new double[a.Large];
    switch(a.labels.length){
     case 2:
         for(int i=0; i<a.Large;i++){
          for(int j=0; j<a.Large;j++){
              if(a.Datos[i][0]!=a.Datos[j][0]&&a.Datos[i][1]!=a.Datos[j][1]){
                 if(i!=j){
              D=(Math.pow((a.Datos[j][0]-a.Datos[i][0]),2) + Math.pow((a.Datos[j][1]-a.Datos[i][1]),2));
              D=Math.sqrt(D); }}
          if(flag==0)
         {  F=D;
        flag=1;}
        if(F>D ){
        F=D;}}
          d[i]=F;
          pro=pro+F;
          flag=0;}
         pro=pro/(a.Large);
         for(int i=0;i<a.Large;i++){
         dif=dif+Math.pow((pro-d[i]), 2);}
         dif=dif/(a.Large-1);
         dif=Math.sqrt(dif);
         dif=Math.rint(dif * 100) /100;
        return  dif;
     case 3:
             for(int i=0; i<a.Large;i++){
             for(int j=0; j<a.Large;j++){
              if(a.Datos[i][0]!=a.Datos[j][0]&&a.Datos[i][1]!=a.Datos[j][1]&&a.Datos[i][2]!=a.Datos[j][2]){
                 if(i!=j){
              D=(Math.pow((a.Datos[j][0]-a.Datos[i][0]),2) + Math.pow((a.Datos[j][1]-a.Datos[i][1]),2));
              D=Math.sqrt(D);}}
          if(flag==0)
         {F=D;
        flag=1;}

          if(F>D ){
          F=D;}}
            d[i]=F;
            pro=pro+F;
            flag=0;}
          pro=pro/(a.Large);
          for(int i=0;i<a.Large;i++){
          dif=dif+Math.pow((pro-d[i]), 2);}
          dif=dif/(a.Large-1);
          dif=Math.sqrt(dif);
          dif=Math.rint(dif * 100) /100;
         return  dif;
      case 4:
              for(int i=0; i<a.Large;i++){
              for(int j=0; j<a.Large;j++){
               if(a.Datos[i][0]!=a.Datos[j][0]&&a.Datos[i][1]!=a.Datos[j][1]&&a.Datos[i][2]!=a.Datos[j][2]&&
                  a.Datos[i][3]!=a.Datos[j][3]){
                    if(i!=j){
               D=(Math.pow((a.Datos[j][0]-a.Datos[i][0]),2) + Math.pow((a.Datos[j][1]-a.Datos[i][1]),2));
               D=Math.sqrt(D); }}
           if(flag==0)
          {F=D;
         flag=1;}
         if(F>D ){
         F=D;}}
           d[i]=F;
           pro=pro+F;
           flag=0;}
          pro=pro/(a.Large);
          for(int i=0;i<a.Large;i++){
          dif=dif+Math.pow((pro-d[i]), 2);}
          dif=dif/(a.Large-1);
          dif=Math.sqrt(dif);
          dif=Math.rint(dif * 100) /100;
         return  dif;
      case 5:
```

```
        if(F>D ){
        F=D;}}
          d[i]=F;
          pro=pro+F;
          flag=0;}
         pro=pro/(a.Large);
         for(int i=0;i<a.Large;i++){
         dif=dif+Math.pow((pro-d[i]), 2);}
         dif=dif/(a.Large-1);
         dif=Math.sqrt(dif);
         dif=Math.rint(dif * 100) /100;
         return dif;}
return 0;}
```

Figura 25: Código Métrica Spacing.

6.3.3 Distance Generational

Se creó un algoritmo que calcula la métrica de distancia generacional entre dos arreglos de datos dentro de la lista previamente cargada en el sistema. Los casos del operador switch son las dimensiones de los datos seleccionados, con un máximo de 6 dimensiones especificado anteriormente. Si las dimensiones de la selección no son las mismas, desplegarán un mensaje de error.

Se desplegará una ventana de tipo dialogo con las soluciones, con la opción de poder guardar los resultados, que serán guardados en un block de notas. En la ventana el listbox representa cuál de los datos seleccionado será considerado como el PF_{true}. En la Figura 21 se aprecia el código de la métrica.

```
public double DistanceG(Datos a, Datos b){
    D=0;G=0;F=-1;R=0;flag=0;
    switch(a.labels.length){
        case 2:
    for(int i=0; i<b.Large;i++){
      for(int j=0; j<a.Large;j++){
         D=( Math.pow(a.Datos[j][0]-b.Datos[i][0],2) + Math.pow((a.Datos[j][1]-b.Datos[i][1]),2));
         D=Math.sqrt(D);
         if(flag==0)
        {F=D;
         flag=1;}
        if(F>D ){
            F=D;}}
         G=(G+Math.pow(F, 2));
         flag=0;}
    G=(Math.pow(G,0.5));
    G=(G/b.Large);
    G=Math.rint(G * 100) /100;
        break;
        case 3:
            for(int i=0; i<a.Large;i++){
      for(int j=0; j<b.Large;j++){
          D=Math.sqrt(Math.pow((a.Datos[i][0]-b.Datos[j][0]), 2)+
           Math.pow((a.Datos[i][1]-b.Datos[j][1]), 2)+Math.pow((a.Datos[i][2]-b.Datos[j][2]), 2));
          D=Math.sqrt(D);
         if(flag==0)
        {  F=D;
         flag=1;}
        if(F>D ){
            F=D;}  }
         G=(G+Math.pow(F, 2));
         flag=0;}
    G=(Math.pow(G,0.33333333333));
    G=(G/b.Large);
    G=Math.rint(G * 100) /100;
    G=(Math.pow(G,0.3333333));

G=(G/b.Large);
G=Math.rint(G * 100) /100;
    break;
    case 4:
        for(int i=0; i<a.Large;i++){
  for(int j=0; j<b.Large;j++){
      D=Math.sqrt(Math.pow((a.Datos[i][0]-b.Datos[j][0]), 2)+
        Math.pow((a.Datos[i][1]-b.Datos[j][1]), 2)+Math.pow((a.Datos[i][2]-b.Datos[j][2]), 2)+
        Math.pow((a.Datos[i][0]-b.Datos[j][0]), 2)+Math.pow((a.Datos[i][3]-b.Datos[j][3]), 2));
      D=Math.sqrt(D);
     if(flag==0)
    {  F=D;
     flag=1;}
    if(F>D ){
        F=D;}}
     G=(G+Math.pow(F, 2));
     flag=0;}
G=(Math.pow(G,0.25));
G=(G/b.Large);
G=Math.rint(G * 100) /100;
    break;
    case 5:
        for(int i=0; i<a.Large;i++){
  for(int j=0; j<b.Large;j++){
      D=Math.sqrt(Math.pow((a.Datos[i][0]-b.Datos[j][0]), 2)+
        Math.pow((a.Datos[i][1]-b.Datos[j][1]), 2)+Math.pow((a.Datos[i][2]-b.Datos[j][2]), 2)+
        Math.pow((a.Datos[i][0]-b.Datos[j][0]), 2)+Math.pow((a.Datos[i][3]-b.Datos[j][3]), 2))+
        Math.pow((a.Datos[i][4]-b.Datos[j][4]), 2);
      D=Math.sqrt(D);
     if(flag==0)
    {  F=D;
     flag=1;}
    if(F>D ){
        F=D;}}
     G=(G+Math.pow(F, 2));
     flag=0;}
```

```
    G=(Math.pow(G,0.2));
    G=(G/b.Large);
    G=Math.rint(G * 100) /100;
        break;
        case 6:
      for(int i=0; i<a.Large;i++){
      for(int j=0; j<b.Large;j++){
          D=Math.sqrt(Math.pow((a.Datos[i][0]-b.Datos[j][0]), 2)+
            Math.pow((a.Datos[i][1]-b.Datos[j][1]), 2)+Math.pow((a.Datos[i][2]-b.Datos[j][2]), 2)+
            Math.pow((a.Datos[i][0]-b.Datos[j][0]), 2)+Math.pow((a.Datos[i][3]-b.Datos[j][3]), 2)+
            Math.pow((a.Datos[i][4]-b.Datos[j][4]), 2)+Math.pow((a.Datos[i][5]-b.Datos[j][5]), 2));
          System.out.print(D+"\n");
          if(F<D){
              F=D;}
          System.out.print("\n\n");}
        G=(G+(F*F));}
    G=(Math.pow(G,0.1666666666666667));
    G=(G/b.Large);
    G=Math.rint(G * 100) /100;
        break;
    }
  return  G;
}
```

Figura 26: Código Métrica Distance Generational.

6.4 Plugins

Dado que uno de los objetivo de la aplicación es que esta no quede obsoleta, se ha decido implementar un sistema con la capacidad de adquirir plugins. Primero, se define plugins como un programa que tiene la cualidad de añadir funcionalidades adicionales al sistema original, añadiendo nuevas características a este. En Java es muy común los plugins en API Applets de los navegadores web y la API Servlet en servidores web.

Para establecer un plugins, hay que saber que existen 4 procesos relacionados para su implementación:

1) Definir la interfaz gráfica, para implementar el plugins.
2) Determinar la forma en la aplicación host llega a saber que plugins están disponibles y donde encontrarlo.
3) Escribir un cargador de clases que carga las clases de plugins.
4) Escribir un gerente de seguridad que gobierna y que permite hacer a los plugins.

Una representación de los componentes que se relacionan con los plugins se describe en la Figura 23,

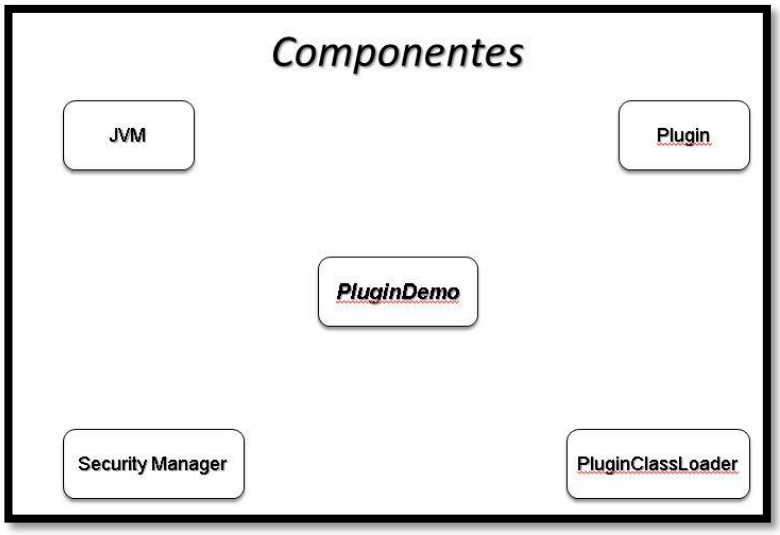

Figura 27: Componentes Plugins.

1) Definición del plugins
   Para que la aplicación host trate a todos los plugins de la misma manera, se define una interfaz donde los plugins se implementan. Los métodos de la interfaz pueden ser considerados como métodos-operaciones de *ciclo de vida*, la aplicación anfitrión llama para inicializar un plugins, los parámetros establecidos, ejecutarlo, obtener resultados y apagarlo. Para este sistema, se define *PluginFunction* interfaz, el cual está compuesto por 4 métodos:

   -public String getPluginName(), devuelve el nombre del plugins, por lo que se puede mostrar al usuario de alguna manera.
   -public setParameter(double [][]param), que permite establecer un parámetro para que el plugin pueda trabajar.
   -public int getResult(), a través del cual la aplicación pueda recuperar el resultado del plugin.
   -hasError public boolean(), que indica que el llamado anterior del plugin no tuvo éxito.

   En cuanto a las llamadas, es claro que los plugins son limitados, solo hay una llamada funcional, que toma el parámetro y devuelve un resultado. Para iniciar se llama al sistema de seguridad (Figura 25), pero se hablara de eso más adelante.

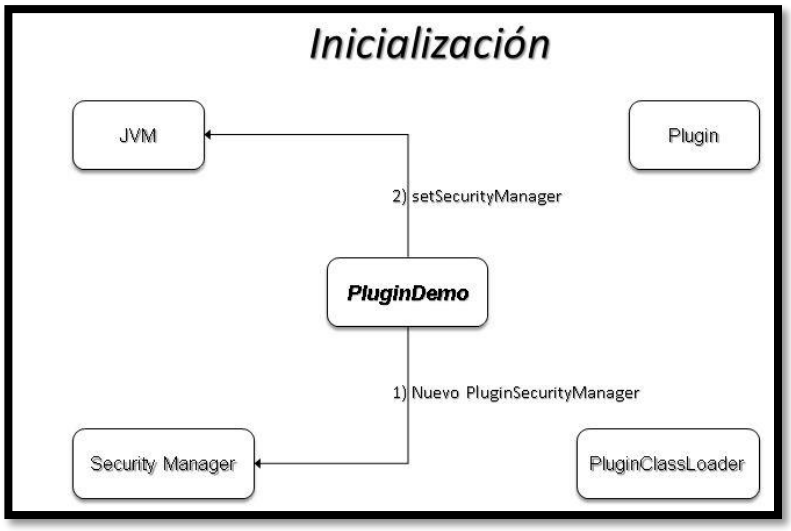

Figura 24: Inicialización.

2) Determinar la adquisición del Plugins.
   Es necesario definir como la aplicación adquiere o llega a saber de los plugins. En este sistema se limita a cargar los plugins desde un directorio en particular en el disco duro, una manera común para las aplicaciones de escritorio. El directorio se llamara *plugins* y estará situado dentro del directorio principal de la aplicación (en el caso de Windows, mientras que Linux será en documentos).
   Es importante considerar es el momento en que la aplicación escanea el directorio para descubrir los plugins. La forma más sencilla es hacerlo una vez en el inicio, momento en que los plugins no se han cargado. En la Figura 24 se muestra la secuencia del proceso *GetPlugin* que carga el nuevo *PluginClassLoader*.

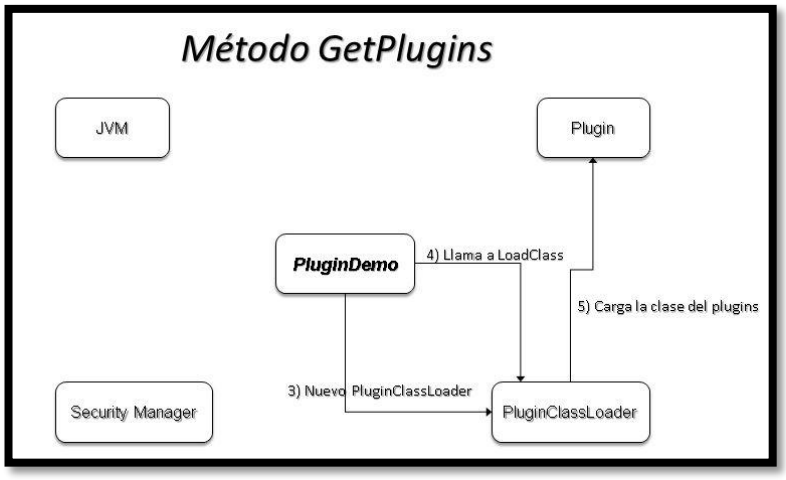

Figura 25: Método GetPlugins.

3) Creación de un classloader.
   Al extender Java *lang.Classloader*, solo hay un único método de implementar. Se pasa una cadena con el nombre de la clase a cargar y maneja de vuelta el objeto de esa clase. A partir de ese objeto, la aplicación puede crear instancias de los objetos en tiempo de ejecución de la clase en el plugin.
   En su constructor, le decimos en que directorio están los plugins. Todo lo importante se ejecuta en el método *LoadClass*, acá hay que considerar tres casos: 1) si la clase está cargada, en ese caso, puede ser encontrada en el cache que mantiene todas las clases cargadas; 2) la clase que se va a cargar sea una clase del sistema, en ese caso la carga se delega al sistema classloader, 3) si no sucede ninguno de los dos casos anteriores, se examina el directorio de plugins, si contiene algún archivo class, con el nombre requerido. Si existe, entonces los bytes que componen físicamente la clase se cargan y se entregan al método *DefineClass* (implementado en *Java.lang.ClassLoader*). El resultado es el objeto de la clase deseada, que luego es devuelto por el método. Se describe el proceso en la Figura 25.

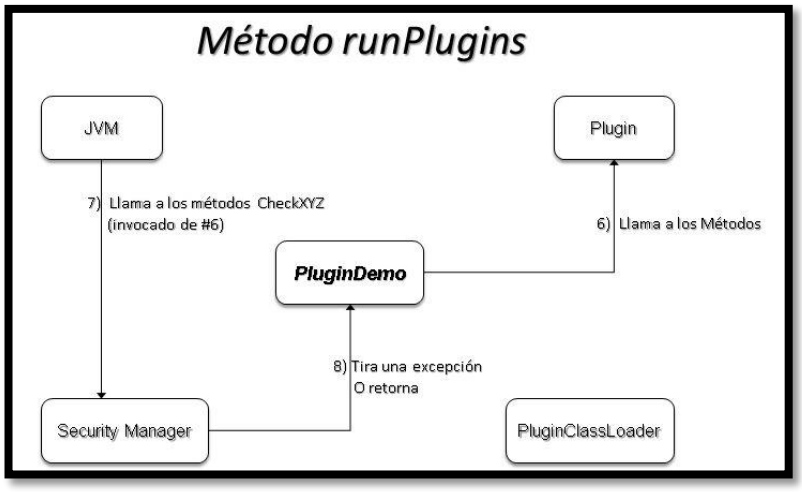

Figura 26: Método runPlugins.

4) Creación de un gerente de seguridad

Al igual que el classloader, escribir el gerente de seguridad se hace más fácil por el hecho de que podemos ampliar *Java.lang.SecurityManager*, y sólo es necesario reemplazar los métodos que especifican qué permisos a otorgar y que nieguen. Si nos fijamos en el código fuente de PluginSecurityManager, se verá un montón de métodos denominados *checkXYZ*. Cada uno de esos chequeos verifica el acceso a alguna característica específica del sistema debe ser permitido o denegado. Hay algunos métodos que tratan de archivo de entrada/ salida, unos pocos sobre la actividad de la red, y un número de otros como el acceso a la impresora, acceso portapapeles, ejecución de código externo, el acceso a las propiedades del sistema y salir de la aplicación. Cada uno de estos a su vez llama al método confiable, que simplemente comprueba si el código que se ejecuta se ejecuta dentro de un cargador de clases. Si produce una excepción, esto significa, que para el JVM la acción solicitada debe ser negado. En este programa se dejara a un lado el sistema de seguridad, debido a que se enfoca en el análisis del plugins y no de las resctricciones de este. En entregas posteriores, se enfocara en la seguridad.

Una vez entendido como se procesa el plugins, ahora veamos cómo se carga los plugins al sistema, como se aprecia en la Figura 27,

```
public Principal() {
    demo.getPlugins();
    initComponents();
    this.ListModel= new DefaultListModel();
    this.Lista.setModel(ListModel);
    JMenu jMenu3 = new javax.swing.JMenu();
    jMenu3.setText("Plugins");
    jMenuBar1.add(jMenu3);
    JMenuItem menuItem=null;
    EscuchaItemMenu eim = new EscuchaItemMenu() {};
    iter = demo.plugins.iterator();
    while (iter.hasNext())
    {
        PluginFunction pf = (PluginFunction) iter.next();
        menuItem =  new  JMenuItem (pf.getPluginName());
        menuItem.addActionListener(eim);
        jMenu3.add(menuItem);
        count++;}}
```

Figura 28: Código Carga de los Plugins.

A través de *iter* se genera iteraciones según la cantidad de plugins que hayan sido leídos en la carpeta de plugins, los cuales son agregados al *menuItem*. Además se le da un *ActionListener*, que es para generar el evento cuando se le hace click en el menú.

```
class EscuchaItemMenu implements ActionListener{
        public void actionPerformed(ActionEvent e){
            iter1=demo.plugins.iterator();
          if(archivoElegido==null){
              JOptionPane.showMessageDialog(null,"[1]  File not loaded","Error Message",JOptionPane.ERROR_MESSAGE);
          }else{
              if(indices.length!=0){
              String Text= e.getActionCommand();
           for(int i=0;i<count;i++){
              PluginFunction pf = (PluginFunction) iter1.next();
            if(Text==pf.getPluginName()){
               demo.runPlugins(demo.plugins,Text,i,Datos[indices[0]].Datos);
              }}}else{
              JOptionPane.showMessageDialog(null,"[1]  File not loaded","Error Message",JOptionPane.ERROR_MESSAGE);
              }}}}
```

Figura 29: Código EscuchaItemMenu.

Como se aprecia en la Figura 28, la función *EscuchaItemMenu*, se utiliza para la acción del evento click del botón dentro del menú, verificando previamente que existan datos en la lista y que se haya seleccionado algo.

7 Capítulo: Validación

En este capítulo, se describirá las pruebas que se realizaron al sistema desarrollado.

7.1 Pruebas de Compatibilidad

Uno de los principales punto del sistema, es la capacidad de compatibilizar con los diferentes sistemas operativos existentes. En este proyecto, se evaluó la compatibilidad hacia 3 sistemas operativos en particular: Windows, Linux y MacOS. Considerando que Java tiene JVM independiente de la arquitectura del sistema, nos otorga la característica de adaptarse a diferentes sistemas operativos.

7.1.1 Prueba en Windows

Se testeo el sistema en Windows 7 64 bits, con JDK 8 y Gnuplot 4.6, dando satisfactorios resultados.

Se aprecia en la Figura 29 corriendo el sistema.

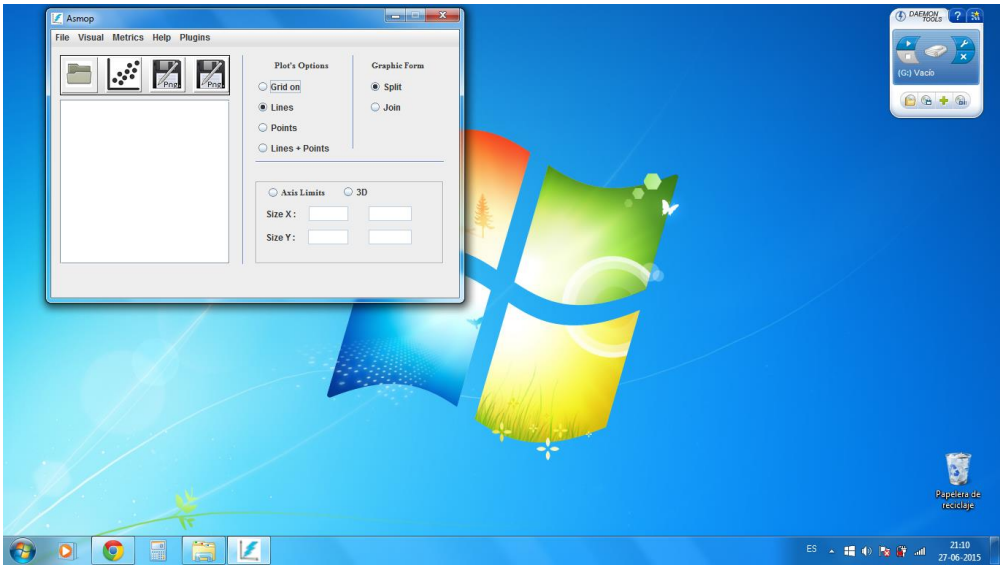

Figura 30: Test Windows 7.

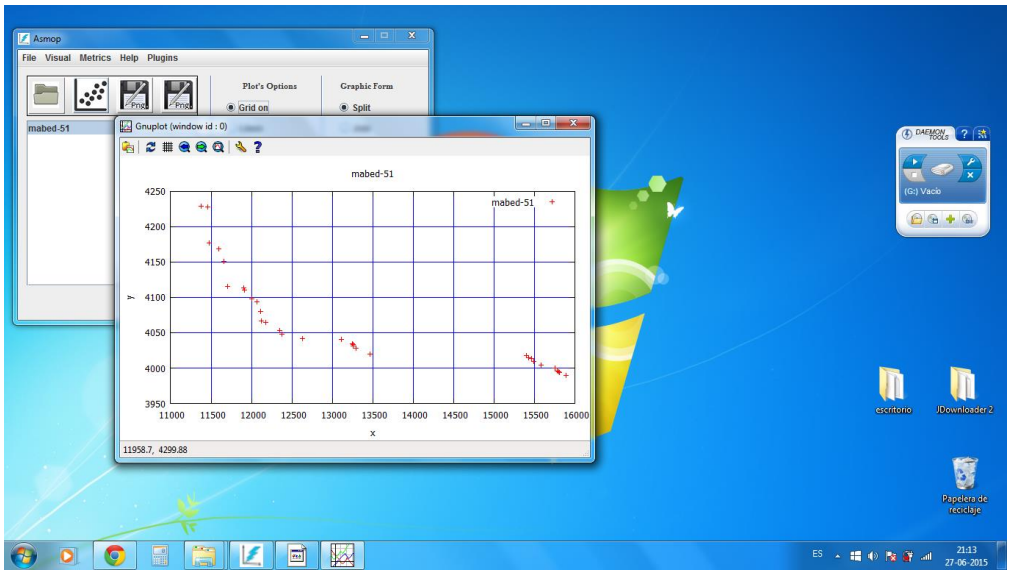

Figura 31: Test de Grafica Windows 7.

En la Figura 30, se muestra como grafica un conjunto de datos seleccionados.

7.1.2 Linux

Se testeo el sistema en Ubuntu 15.04 64x, con JDK 8 y Gnuplot 4.6, dando satisfactorios resultados, como se ven en las Figuras 31 y 32.

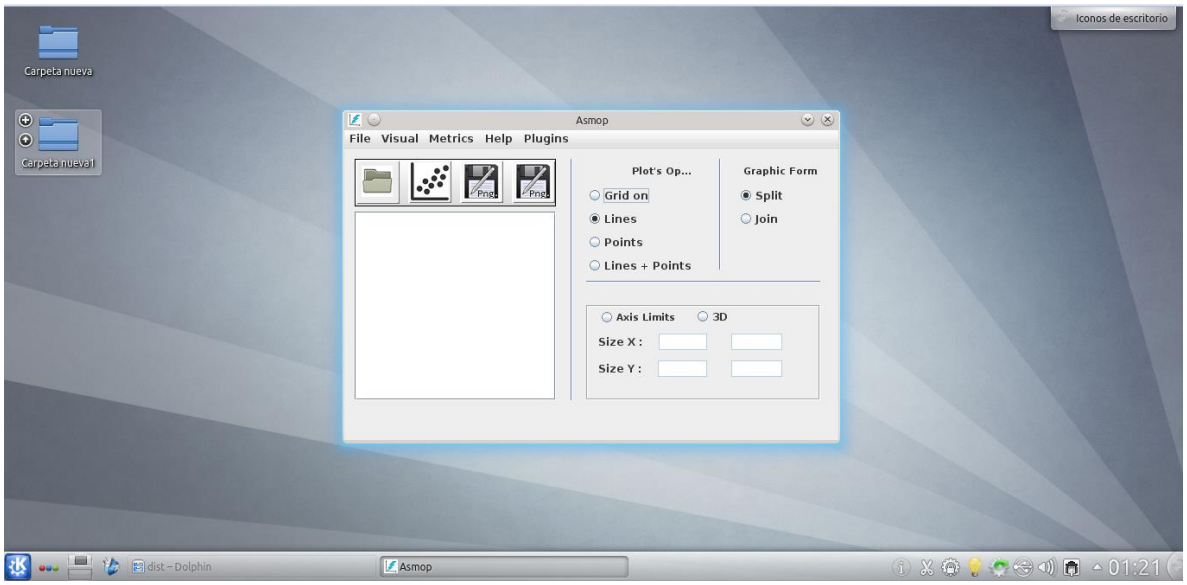

Figura 32: Test Kubuntu

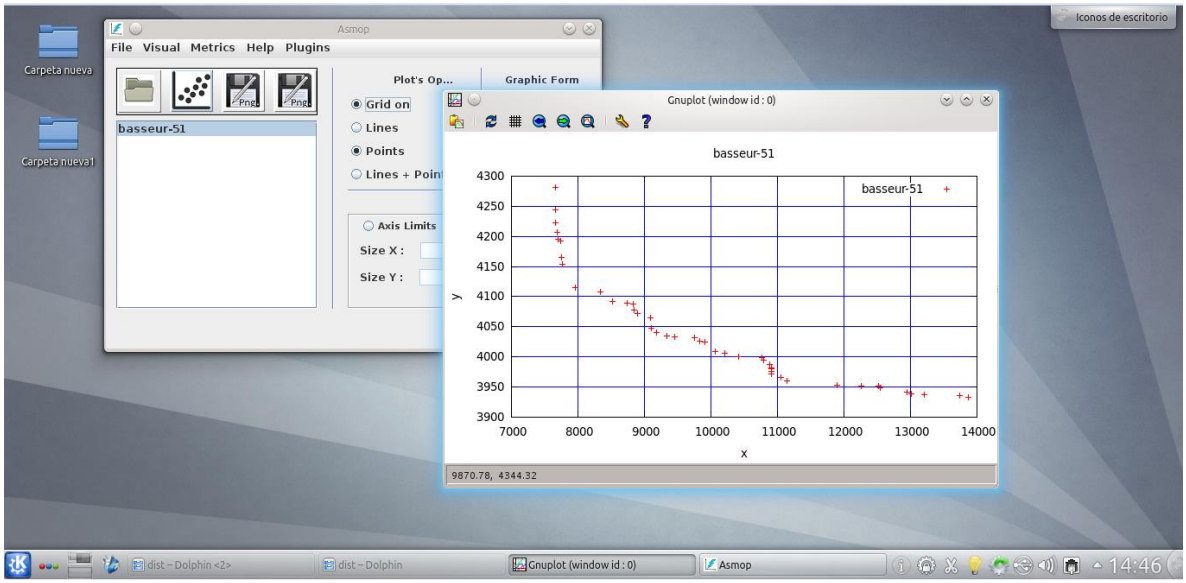

Figura 33: Test Grafica Kubuntu.

Cabe destacar que en Linux la ruta absoluta de donde se ejecuta el sistema, varia de cómo se ejecute, por lo tanto para que funcione correctamente se debe ejecutar desde consola

java –jar Asmop.jar.

7.1.3 Mac

Test en Mac, maverick 10.49, Gnuplot 4.4, JDK 8, como se aprecia en la Figura 33.

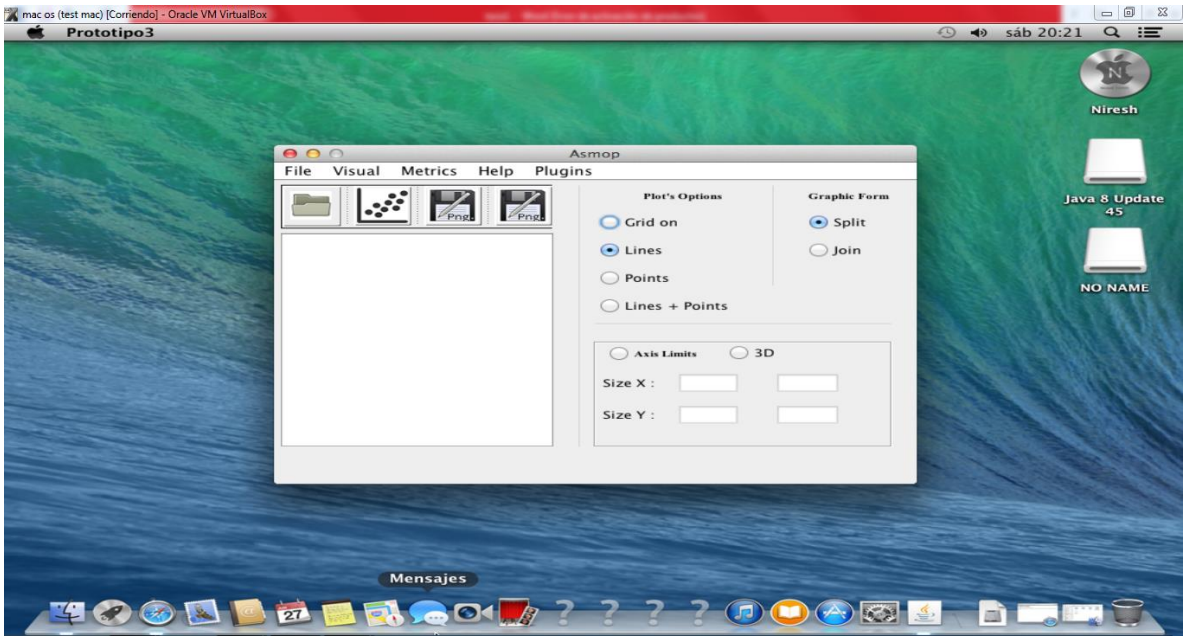

Figura 34: Test Maverick.

7.2 Casos de Prueba

Para el sistema creado existen una gran cantidad de pruebas a realizar, pero nos enfocaremos en casos de pruebas para caja negra, cuyo objetivo es verificar que las funciones cumplan con los resultados que se esperan.

7.2.1 Pruebas Datos de Entrada: Datos en Archivos de Texto.

Se enfocara en los datos de entrada, donde existen más probabilidades de error, cuyo objetivo es verificar si el sistema está listo para poder detectar errores en las entradas de datos, importantes para el graficado de estas.

Se hará un conjunto de archivos de entradas en archivos de texto, para ver su respuesta, y validar si la respuesta es la que se espera.

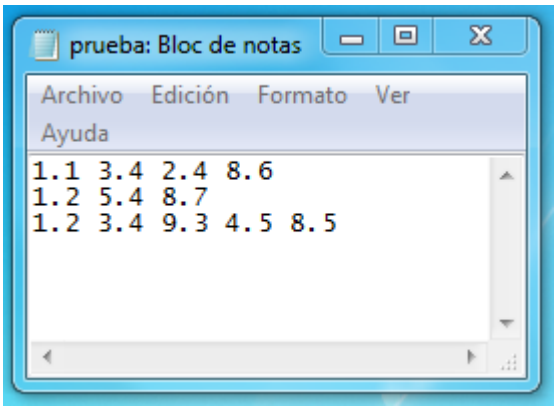

Figura 35. Test Txt 1.

Prueba 1: Fila de Datos incompletas (véase Figura 34).

Resultado:

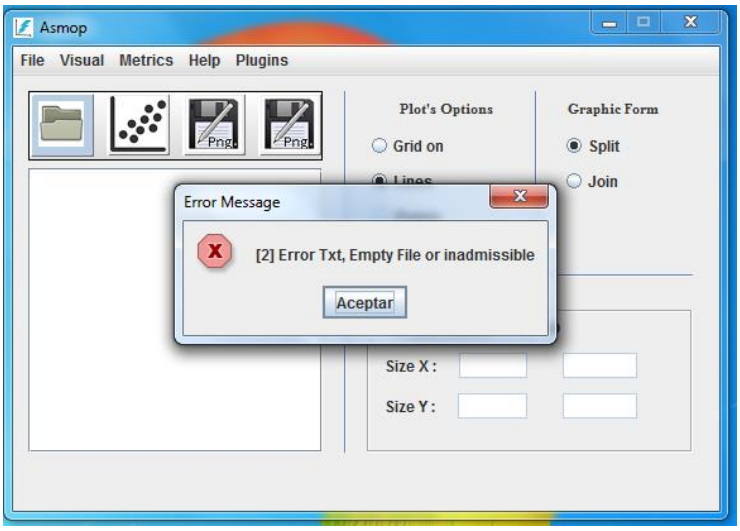

Figura 36: Resultado Test Txt 1.

Prueba 2: dimensiones de las columnas incompletas

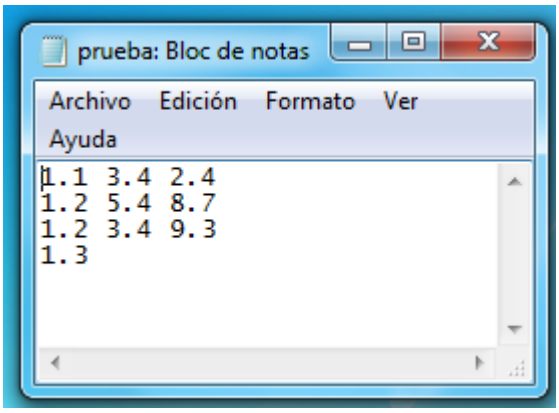

Figura 37: Test Txt 2.

Resultado:

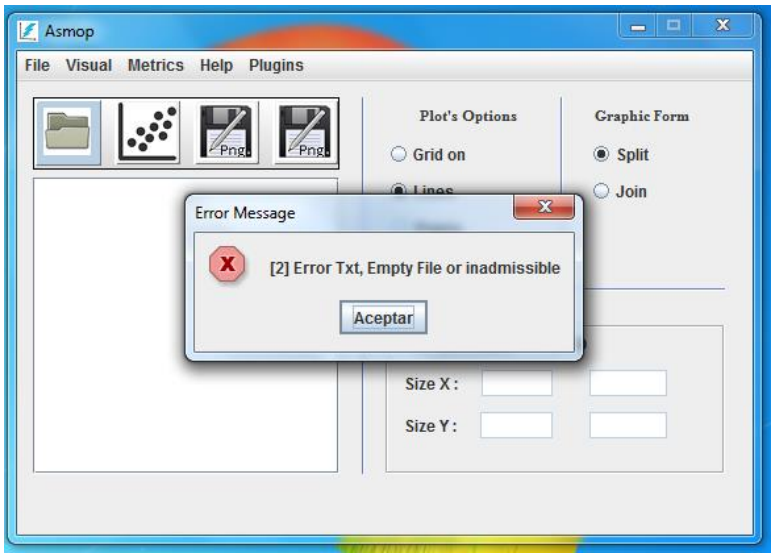

Figura 38: Resultado Test Txt 2.

Prueba 3: caracteres

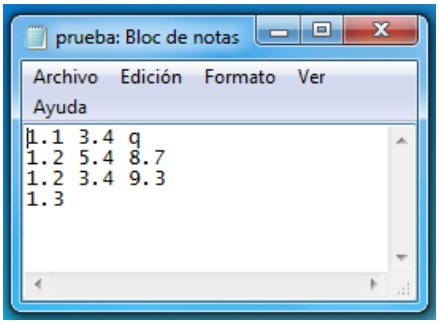

Figura 39: Test Txt 3.

Resultado:

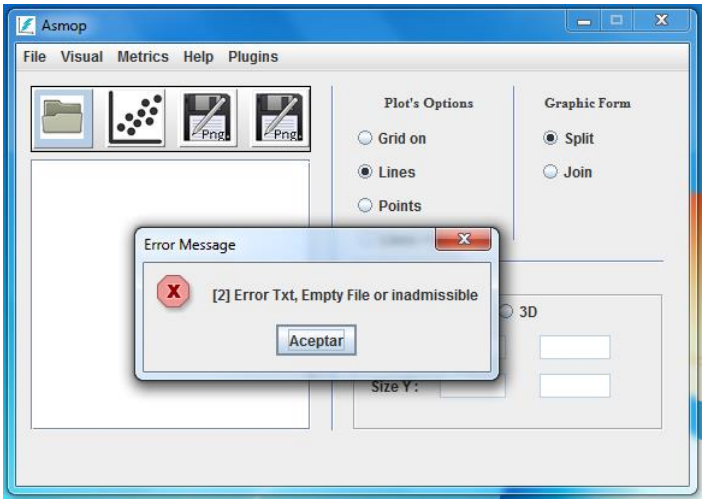

Figura 40: Resultados Test Txt 3.

7.2.2 Pruebas Datos de Entrada: Datos en Excel 2007.

Como el análisis anterior, se harán pruebas de entrada para ver si la respuesta es la esperada, en este caso a archivos de Excel 2007(xls). También experimentamos los mismos objetivos anteriores, es importante refinar la entrada de datos para que los gráficos se diseñen correctamente.

Prueba 1: Fila de Datos incompletas.

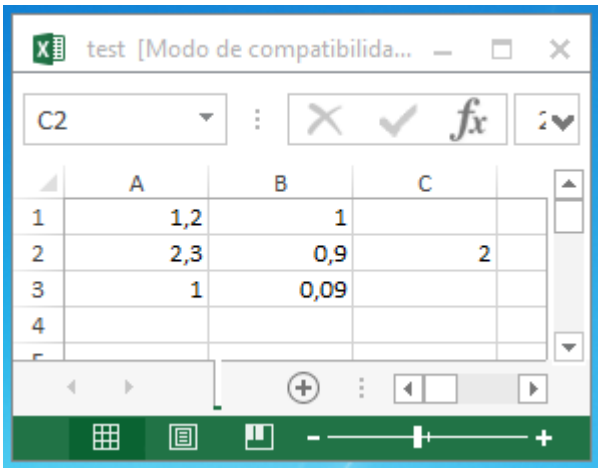

Figura 41: Test Xls 1.

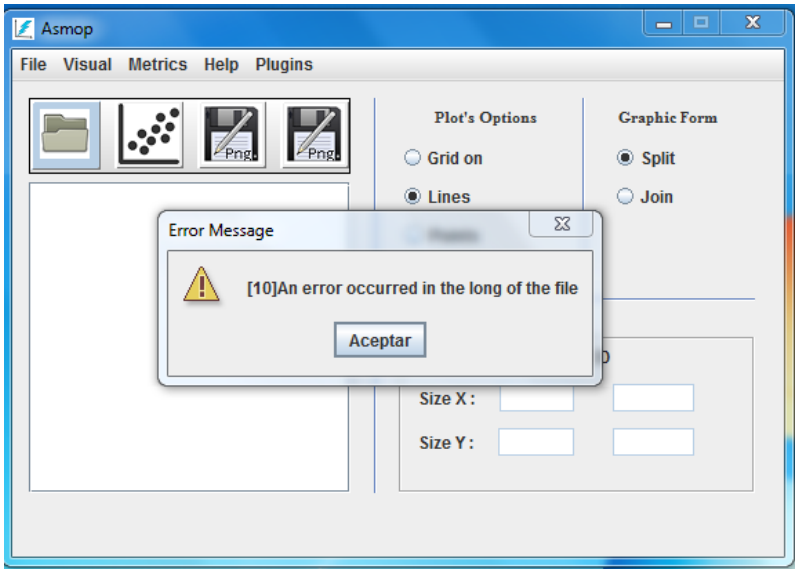

Figura 42: Resultado Test Xls 1.

Prueba 1: Columnas de Datos incompletas.

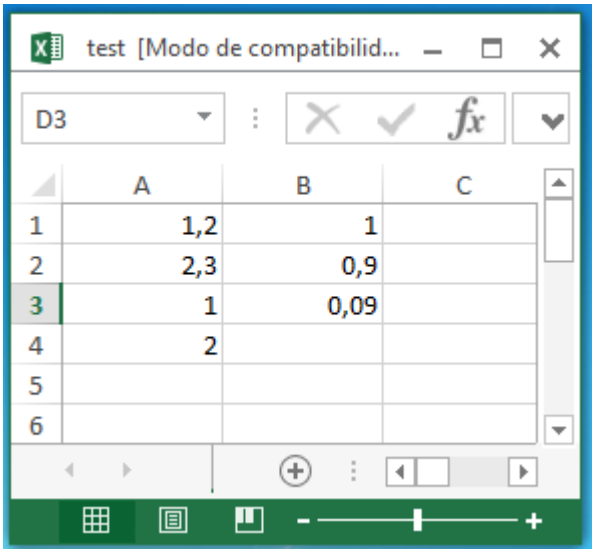

Figura 43: Test Xls 2.

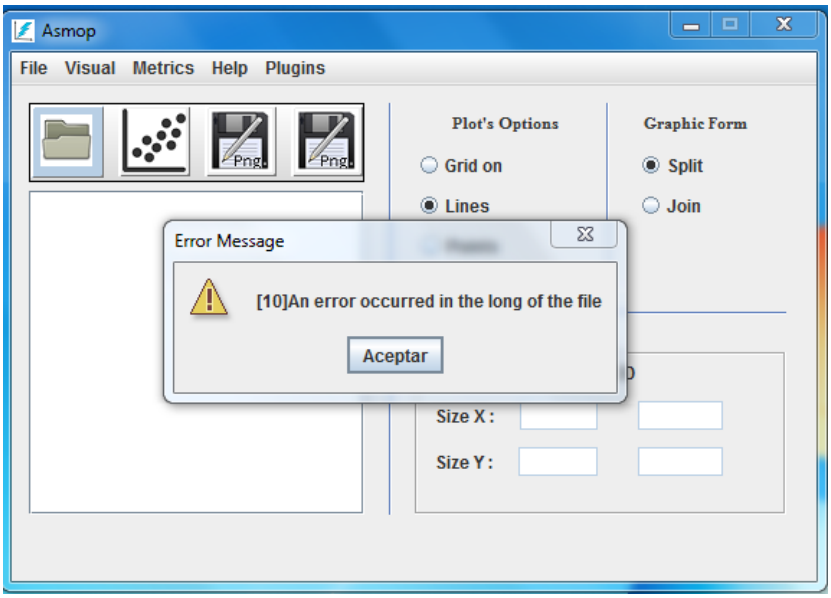

Figura 44: Resultado Test Xls 2.

Prueba 1: Datos incorrectos.

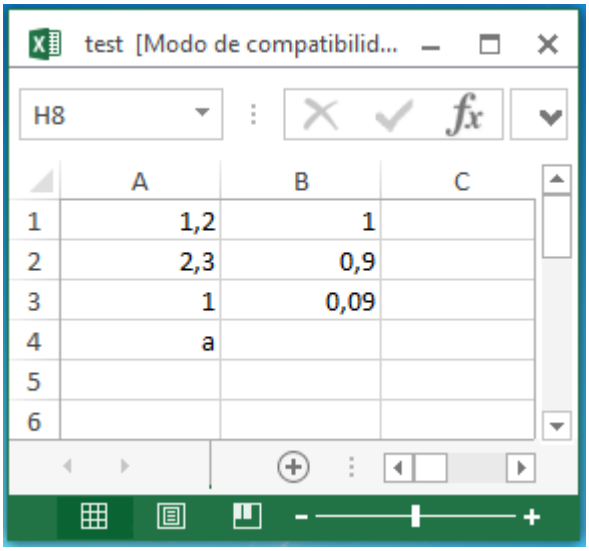

Figura 45: Test Xls 3.

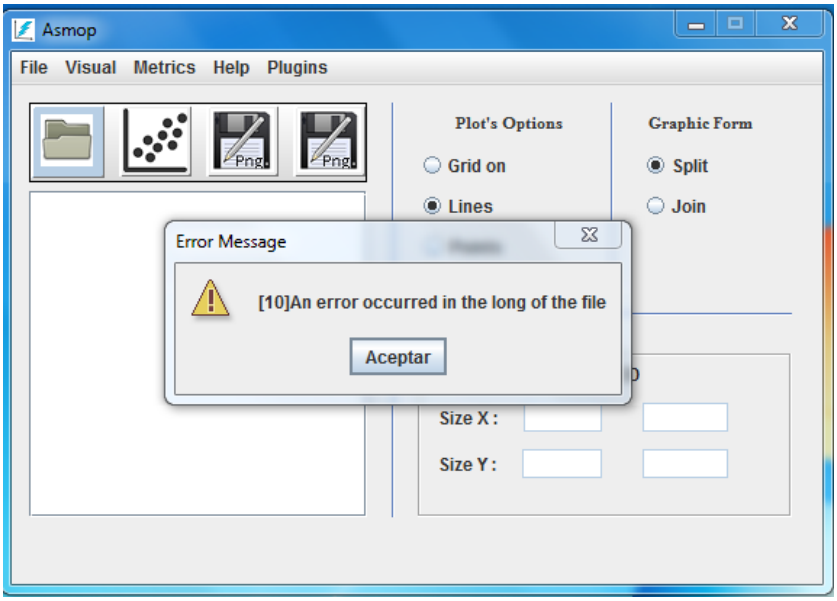

Figura 46: Resultado Test Xls 3.

8 Capítulo: Conclusiones.

La optimización siempre será un tema importante para la vida cotidiana ya que en todo sentido siempre se está requiriendo mejores formas de realizar ciertas tareas y cuyo objetivo siempre será hacerlo de la forma más óptima ya sea maximizando ganancias o minimizando costos, será un objetivo importante a perseguir. A través de Asmop se ha desarrollado una herramienta para el análisis de soluciones, de forma visual y cuantitativa.

El objetivo Principal de este proyecto fue el desarrollo de una aplicación capaz de analizar y graficar soluciones de problemas multi-objetivo, ya sea cualitativa o cuantitativamente, con la cualidad de agregar nuevas métricas al sistema a diferencia de las herramientas actuales.

De acuerdo a los objetivos propuestos en la sección 1, se puede concluir lo siguiente:

- Objetivo Especifico 1: Se estudiaron y definieron los puntos más importantes sobre la optimización multi-objetivo, entendiendo la importancia del frente de Pareto, como análisis gráfico y matemático de los problemas. Además, se estudiaron las métricas para el desarrollo de estas para el sistema.

- Objetivo Especifico 2: Se definieron las principales funciones del sistema y luego se hizo una especificación de requisitos donde se enfocó principalmente la entrada de datos en formato Excel y archivo de textos.
  La función de ploteo: se diseñó para graficas en 2D o 3D para uno o varios datos, unidos o separados. Las gráficas que se generan del ploteo pueden ser guardadas en 2 formatos como formato simple PNG (no escalar) y EPS (vectorial) para aquellos que utilicen documentos vectoriales como Latex o PDF.
  La carga de archivos: se definió la carga de archivos como formato admisible de tipo Excel 2007(XLS) y archivos de texto (TXT).
  La carga de los plugins: Se definió las funciones que leen y cargan los plugins y cómo interactúan con el sistema principal.
  Además se añadieron 3 métricas estáticas la distancia generacional el radio de error y el espaciado.

- Objetivo Especifico 3: Se diseñaron a través de varios diagramas de diseño el esquema general del sistema. Como son diagrama de clases, actividades, wireframe, interfaz de usuario de cada pantalla con el fin definir claramente el sistema.
  Se definió el desarrollo a través de la herramienta de desarrollo Netbeans hecho principalmente para programación Java.

Se analizaron las librerías JavaPlot que se encargaba de la comunicación entre Java y Gnuplot y la Poi-Java se encargaba de la lectura de los archivos de Excel (xls).
Se incorporaron plugins al sistema como poderosa propiedad que lo diferencia Guimoo que incorpora nuevas propiedades al sistema actual para que el sistema se mantenga actualizado a través del tiempo.
Se definió la arquitectura de los plugins para que en trabajos futuros como la creación de nuevos plugins o mejorar los ya existentes sea más rápido entender para su desarrollo.

- Objetivo Especifico 4: Testeo

  Se validó que el sistema fuera multiplaforma, testeándolo en los 3 principales sistemas como Windows 7, Ubuntu 15.04 y Maverick 10.49.

  Se debe considerar que para el buen funcionamiento es necesario ciertos requerimientos del sistema tales como: JDK 8 o superior, Gnuplot 4.0 o superior.

  Se realizó un testeo para los archivos de entrada, en este se ingresaron archivos con errores para probar que el programa los clasificará como archivo errores, dando los mensajes de errores correspondientes.

8.1 Recomendaciones.

A pesar de que en el capítulo 6 de desarrollo del sistema, se habló de la seguridad de los plugins, este no se desarrolló dado que no era un objetivo del proyecto y al profundizar en el sería un tema muy amplio el cual da para proyectos futuros en donde se puede tratar este tema junto con realizar un sistema creador de plugins y hacer crecer la aplicación sin tanto conocimiento en programación.

Dado que el sistema funciona con Xls que es de Excel 2007 como mejora sería que soporte con formatos Xlsx.

Bibliografía.

Antonio J. Nebro, J. J. (2014). *jMetal 4.5 User Manual.*

Apache. (03 de Agosto de 2015). Obtenido de http://poi.apache.org/

Arnaud Liefooghe, M. B.-G. (2007). *ParadisEO-MOEO: A Framework for Evolutionary Multi-objective Optimization.* France: Springer.

Coello, D. C. (16 de Septiembre de 2015). *EMOO Web Page*. Obtenido de http://delta.cs.cinvestav.mx/~ccoello/EMOO/

D. Van Veldhuizen, A. L. (1998). *Evolutionary computation and convergence to a pareto front.* San Francisco, California: Morgan Kaufmann Publicaciones.

Dittmer, U. (03 de agosto de 2015). *Adding Plugins to a Java Application*. Obtenido de http://www.javaranch.com/journal/200607/Plugins.html

E. Zitzler, L. T. (1998). *An evolutionary algorithm for multiobjective.* Zurich,Suiza.: Instituto Federal Suizo de Tecnología.

Galeano S.E., M. C. (2008). *Optimización Multi-objetivo de la Operación en Sistemas Automatizados de Distribución de Energía Eléctrica.* Universidad de Pereira, Pereira: Colombia.

H. Meunier, E. G. (2000). *A multiobjective genetic algorithm for radio network optimization.* New Jersey: CEC.

Inria. (20 de Agosto de 2015). *CeCILL and Free Software*. Obtenido de http://www.cecill.info/index.en.html

Inria. (20 de agosto de 2015). *Guimoo*. Obtenido de http://guimoo.gforge.inria.fr/

Isidro Ramos Salavert, M. L.-C. (2000). *Ingeniería del Software y Bases de Datos I.* Albacete, España: Ediciones de la Universidad de Castilla- La Mancha.

Liefooghe Arnaud, J. L.-G. (2010). *ParadisEO-MOEO: A Software Framework for Evolutionary Multi-Objective Optimization.* Université Lille 1, Bâtiment M3, 59655, Villeneuve d'Ascq cedex, France: Springer.

Madrid, U. P. (2011). *Optimización Multi-objetivo Basada en Meta-heurísticas.* Madrid.

Oracle. (Agosto de 2015). *Netbeans Homepage*. Obtenido de https://netbeans.org/

panayotis. (2007). *JavaPlot*. Obtenido de http://javaplot.panayotis.com/

Ramos I., L. M. (2000). *Ingeniería del Software y Base de Datos: Tendencias Actuales.* España: Albacete.

Schott, J. (1995). *Fault tolerant design using single and multictiteria gentetic algorithm optimization.* Massachusetts: Massachusetts Institute of Technology.

Sommerville, I. (2005). *Ingeniería del Software.* Madrid, España: Pearson Educación.

T. Williams, C. K. (20 de agosto de 2015). *gnuplot homepage*. Obtenido de http://www.gnuplot.info/

Zitzler, E. (1999). *Evolutionary Algorithms for Multiobjective Optimization: Methods.* Shaker Verlag, Aachen, Alemania: Instituto Federal Suizo de Tecnología.

Anexos

Anexo I: Código Principal.Java.

```
public class Principal extends javax.swing.JFrame {
    public  File archivoElegido,archivoElegido2;
    private final DefaultListModel ListModel;
    private final FileNameExtensionFilter filter= new FileNameExtensionFilter("excel 97 y text files","xls","txt");
    private final FileNameExtensionFilter filter1= new FileNameExtensionFilter("Png Files","png");
    private final FileNameExtensionFilter filter2= new FileNameExtensionFilter("Eps Files","eps");
    FileInputStream fstream = null,fstrea = null,fstrea1;
    public Datos[] Datos= new Datos[10];
    public Plot plot= new Plot();
    int i=0,e=-1,opG=0,count=0;
    public int [] indices;
    public boolean Flag;
    public static List plugs;
    public  Iterator iter,iter1,iter2;
    PluginDemo demo = new PluginDemo();
    int [] axis=new int[6];

    public Principal() {
        demo.getPlugins();
        initComponents();
        this.ListModel= new DefaultListModel();
        this.Lista.setModel(ListModel);
        JMenu jMenu3 = new javax.swing.JMenu();
        jMenu3.setText("Plugins");
        jMenuBar1.add(jMenu3);
        JMenuItem menuItem=null;
        EscuchaItemMenu eim = new EscuchaItemMenu() {};
        iter = demo.plugins.iterator();
        while (iter.hasNext()) //Se leen los plugin que estan almacenados en el directorio especifico y son cargados en el menu
        {
            PluginFunction pf = (PluginFunction) iter.next();
             menuItem =  new  JMenuItem (pf.getPluginName());
             menuItem.addActionListener(eim);
             jMenu3.add(menuItem);
             count++;}}
    //esta clase esta encargada de leer las acciones de cuando se selecciona un plugin de la lista
    //busca el plugin y lo ejecuta
```

```
class EscuchaItemMenu implements ActionListener{
        public void actionPerformed(ActionEvent e){
            iter1=demo.plugins.iterator();
          if(archivoElegido==null){
               JOptionPane.showMessageDialog(null,"[1]  File not loaded","Error Message",JOptionPane.ERROR_MESSAGE);
          }else{
              if(indices.length!=0){
              String Text= e.getActionCommand();
           for(int i=0;i<count;i++){
              PluginFunction pf = (PluginFunction) iter1.next();
            if(Text==pf.getPluginName()){
                demo.runPlugins(demo.plugins,Text,i,Datos[indices[0]].Datos);
               }}}else{
               JOptionPane.showMessageDialog(null,"[1]  File not loaded","Error Message",JOptionPane.ERROR_MESSAGE);
              }}}}

    public Image getIconImagen(){//carga el icono
        Image retValue= Toolkit.getDefaultToolkit().getImage(ClassLoader.getSystemResource("iconos/icono.png"));
        return retValue;}

    public void goToURL() {
         File fichero = new File(System.getProperty("user.dir")+"//"+"help"+"//"+"html"+"//"+"main.html");
        try {
            Desktop.getDesktop().open(fichero);
        } catch (IOException ex) {
            Logger.getLogger(Principal.class.getName()).log(Level.SEVERE, null, ex);}}


    @SuppressWarnings("unchecked")
    Generated Code
    private void LiPointBActionPerformed(java.awt.event.ActionEvent evt) {
        PointB.setSelected(false);
        LinesB.setSelected(false);
        opG=2;
    }
    private void SpitBActionPerformed(java.awt.event.ActionEvent evt) {
        JoinB.setSelected(false);
```

```
    }
    private void JoinBActionPerformed(java.awt.event.ActionEvent evt) {
        SpitB.setSelected(false);
    }
    private void LinesBActionPerformed(java.awt.event.ActionEvent evt) {
        PointB.setSelected(false);
        LiPointB.setSelected(false);
        opG=0;
    }
    private void PointBActionPerformed(java.awt.event.ActionEvent evt) {
        LinesB.setSelected(false);
        LiPointB.setSelected(false);
        opG=1;
    }

    private void OpenFActionPerformed(java.awt.event.ActionEvent evt) {
            if(i<10){
            JFileChooser fc = new JFileChooser();
            fc.setFileFilter(filter);
            int respuesta = fc.showOpenDialog(this);
        //Comprobar si se ha pulsado Aceptar
        if (respuesta == JFileChooser.APPROVE_OPTION)
        {archivoElegido = fc.getSelectedFile();
            archivoElegido2 = fc.getSelectedFile();
                try {
                    fstream = new FileInputStream(archivoElegido.getAbsolutePath());
                    fstrea = new FileInputStream(archivoElegido.getAbsolutePath());
                    fstrea1 = new FileInputStream(archivoElegido.getAbsolutePath());
                } catch (FileNotFoundException ex) {
                    Logger.getLogger(Principal.class.getName()).log(Level.SEVERE, null, ex);}
         //revisa la extension del archivo, en este caso para archivos de texto
       if(archivoElegido.getAbsolutePath().endsWith(".txt")==true){
       LoadTxt load= new LoadTxt();
       int weight=load.anchoTxt(fstrea);
                    if(weight==0){
                       JOptionPane.showMessageDialog(null,"[2] Error Txt, Empty File or inadmissible","Error Message",
                       JOptionPane.ERROR_MESSAGE);
                        }else{
                          Datos[i]= new Datos();
                          if(weight<=6){
                          Datos[i].Large=load.largoTxt(fstrea1);
                         StringTokenizer name;
                         name = new StringTokenizer(archivoElegido.getName(),".");
                         Datos[i].nombre=name.nextToken();
            try {
                Datos[i].Datos=load.fillTxt(archivoElegido,load.largoTxt(fstream),weight);
            } catch (IOException ex) {
                Logger.getLogger(Principal.class.getName()).log(Level.SEVERE, null, ex);
            }
                         if(load.Flag==true){
                         this.ListModel.addElement(Datos[i].nombre);
                         switch ( weight ) {
                             case 2:
                             Label2d obj= new Label2d(this, true);
                             obj.setVisible(true);
                             Datos[i].labels= new String[weight];
                             Datos[i].labels[0]=obj.jTextField1.getText();
                             Datos[i].labels[1]=obj.jTextField2.getText();
                             i++;
                             break;
                             case 3:
                             Label3d obj1= new Label3d(this, true);
                             obj1.setVisible(true);
                             Datos[i].labels= new String[weight];
                             Datos[i].labels[0]=obj1.jTextField1.getText();
                             Datos[i].labels[1]=obj1.jTextField2.getText();
                             Datos[i].labels[2]=obj1.jTextField3.getText();
                             i++;
                                 break;
                             default:
                                  if(weight==4){
                                 Datos[i].labels= new String[weight];
                                 Datos[i].labels[0]="x";
                                 Datos[i].labels[1]="y";
```

```
                        Datos[i].labels[2]="z";
                        Datos[i].labels[3]="k";
                        }else{
                         if(weight==5){
                        Datos[i].labels= new String[weight];
                        Datos[i].labels[0]="x";
                        Datos[i].labels[1]="y";
                        Datos[i].labels[2]="z";
                        Datos[i].labels[3]="k";
                        Datos[i].labels[4]="l";
                         }else{
                          if(weight==6){
                        Datos[i].labels= new String[weight];
                        Datos[i].labels[0]="x";
                        Datos[i].labels[1]="y";
                        Datos[i].labels[2]="z";
                        Datos[i].labels[3]="k";
                        Datos[i].labels[4]="l";
                        Datos[i].labels[5]="m";
                          }}}
                        break;}
                    }}else{
                        JOptionPane.showMessageDialog(null,"[6]  You have exceeded the maximum of axis","Error Message",
                        JOptionPane.WARNING_MESSAGE);
                    }}}else{
    //en caso de que sea .xls la extension
if(archivoElegido.getAbsolutePath().endsWith(".xls")==true){
            LoadXls xls= new LoadXls();
            List sheetData = new ArrayList();
            try {
                 sheetData=xls.CargarExcel(archivoElegido.getAbsolutePath());
                } catch (IOException ex) {
                 Logger.getLogger(Principal.class.getName()).log(Level.SEVERE, null, ex);}
                Datos[i]= new Datos();
                if(xls.SizeY(sheetData)<=6){
                Datos[i].Large=xls.SizeX(sheetData);
                StringTokenizer name;
                name = new StringTokenizer(archivoElegido.getName(),".");
                Datos[i].nombre=name.nextToken();
                xls.showExelData(sheetData,Datos[i]);
                if(Datos[i].labels[0]==null){
                   if(xls.Flag==true){
                  switch ( xls.SizeY(sheetData) ) {
                       case 2:
                       Label2d obj= new Label2d(this, true);
                       obj.setVisible(true);
                       Datos[i].labels= new String[xls.SizeY(sheetData)];
                       Datos[i].labels[0]=obj.jTextField1.getText();
                       Datos[i].labels[1]=obj.jTextField2.getText();
                       this.ListModel.addElement(Datos[i].nombre);
                       i++;
                       break;
                       case 3:
                       Label3d obj1= new Label3d(this, true);
                       obj1.setVisible(true);
                       Datos[i].labels= new String[xls.SizeY(sheetData)];
                       Datos[i].labels[0]=obj1.jTextField1.getText();
                       Datos[i].labels[1]=obj1.jTextField2.getText();
                       Datos[i].labels[2]=obj1.jTextField3.getText();
                       this.ListModel.addElement(archivoElegido.getName());
                       i++;
                           break;
                       default:
                           if(xls.SizeY(sheetData)==4){
                           Datos[i].labels= new String[xls.SizeY(sheetData)];
                           Datos[i].labels[0]="x";
                           Datos[i].labels[1]="y";
                           Datos[i].labels[2]="z";
                           Datos[i].labels[3]="k";
                           }else{
                            if(xls.SizeY(sheetData)==5){
                           Datos[i].labels= new String[xls.SizeY(sheetData)];
                           Datos[i].labels[0]="x";
                           Datos[i].labels[1]="y";
```

```
                                Datos[i].labels[2]="z";
                                Datos[i].labels[3]="k";
                                Datos[i].labels[4]="l";
                                 }else{
                                  if(xls.SizeY(sheetData)==6){
                                Datos[i].labels= new String[xls.SizeY(sheetData)];
                                Datos[i].labels[0]="x";
                                Datos[i].labels[1]="y";
                                Datos[i].labels[2]="z";
                                Datos[i].labels[3]="k";
                                Datos[i].labels[4]="l";
                                Datos[i].labels[4]="m";
                                  }}}
                               break;}
                        }}else{
                        if(xls.lab<2){
                        JOptionPane.showMessageDialog(null,"[3]  Empty File or inadmissible","Error Message",
                        JOptionPane.ERROR_MESSAGE);
                        }else{
                        this.ListModel.addElement(archivoElegido.getName());}
                        }}else{
                         JOptionPane.showMessageDialog(null,"[6]  You have exceeded the maximum of axis","Error Message",
                         JOptionPane.ERROR_MESSAGE);}
                                                        }else{// en caso de poner otro tipo de archivos
                                                        System.out.println("[5]  Other formats are not compatible");}}}
            }else{
               JOptionPane.showMessageDialog(null,"[8]  Limit reached Files","Error Message",JOptionPane.ERROR_MESSAGE);}
        }
        private void formWindowClosing(java.awt.event.WindowEvent evt) {
            Object [] opciones ={"Aceptar","Cancelar"};
            int eleccion = JOptionPane.showOptionDialog(rootPane,"Do you want to Exit?","Confimation Message",
            JOptionPane.YES_NO_OPTION,
            JOptionPane.QUESTION_MESSAGE,null,opciones,"Aceptar");
            if (eleccion == JOptionPane.YES_OPTION)
            {
            System.exit(0);
            }else{}
        }

        private void ListaValueChanged(javax.swing.event.ListSelectionEvent evt) {
            indices=Lista.getSelectedIndices();
        }

        private void ClearBActionPerformed(java.awt.event.ActionEvent evt) {
           ListModel.clear();
           i=0;
           archivoElegido=null;
        }

        private void ClearBMouseClicked(java.awt.event.MouseEvent evt) {
             ListModel.clear();
                i=0;
        }

        private void PlotBActionPerformed(java.awt.event.ActionEvent evt) {

            if(archivoElegido==null||indices==null){
               JOptionPane.showMessageDialog(null,"[1]  File not loaded","Error Message",JOptionPane.ERROR_MESSAGE);}
           else{
            if(indices.length==1){
                if(Datos[indices[0]].labels.length==2){
                 Plot.ploteo2D(Datos,indices[0],GridB.isSelected(),opG,axisOp.isSelected(),axis);
                }else{
                    if(Datos[indices[0]].labels.length==3){
                Plot.ploteo3D(Datos,indices[0],GridB.isSelected(),opG,axisOp.isSelected(),axis,triD.isSelected());
                }
                 else{
                    JOptionPane.showMessageDialog(null,"[4]  Plot error exceeds more than 3 dimensions","Error Message",
                    JOptionPane.ERROR_MESSAGE);
                    }}
            }else{
               for(int i=1;i<indices.length;i++){
                    if(Datos[indices[i-1]].labels.length==Datos[indices[i]].labels.length){
                     Flag=true;
```

```
                }else{
                   JOptionPane.showMessageDialog(null,"[5]  The dimensions of the files does not match","Error Message",
                   JOptionPane.ERROR_MESSAGE);
                   i=indices.length;
                   Flag=false;}}
               if(Flag==true){
                   if(Datos[indices[0]].labels.length==2){
               Plot.ploteo2DMul(Datos,indices,GridB.isSelected(),opG,SpitB.isSelected(),axisOp.isSelected(),axis);
           }else{
                 if(Datos[indices[0]].labels.length==3){
               Plot.ploteo3DMul(Datos,indices,GridB.isSelected(),opG,SpitB.isSelected(),axisOp.isSelected(),axis,
               triD.isSelected());
                   }else{
                 JOptionPane.showMessageDialog(null,"[4]  Plot error exceeds more than 3 dimensions","Error Message",
                 JOptionPane.ERROR_MESSAGE);
                 }}}}}
    }

    private void SavePngMouseClicked(java.awt.event.MouseEvent evt) {
        //opcion donde guarda el plot, en un archivo .png
        if(archivoElegido==null||indices==null){
           JOptionPane.showMessageDialog(null,"[1] File not loaded","Error Message",JOptionPane.ERROR_MESSAGE);}
       else{
        JFileChooser fc=new JFileChooser();
        fc.setFileFilter(filter1);
        int respuesta=fc.showSaveDialog(this); //Muestra el diálogo
        if (respuesta == JFileChooser.APPROVE_OPTION){
            File Guardar =fc.getSelectedFile();
            if(Guardar.exists()){
        Object [] opciones ={"Aceptar","Cancelar"};
        int eleccion = JOptionPane.showOptionDialog(rootPane,"OverWrite the Files?","Mensaje de Confirmacion",
        JOptionPane.YES_NO_OPTION,
        JOptionPane.QUESTION_MESSAGE,null,opciones,"Aceptar");
        if (eleccion == JOptionPane.YES_OPTION)
        {
        String Path=Guardar.getAbsolutePath();
             if(indices.length==1){// para un dato
            if(Datos[indices[0]].labels.length==2){// 2D plot
             Plot.ploteo2Dpng(Datos,indices[0],GridB.isSelected(),opG,Path,axisOp.isSelected(),axis);
            }else{//3D plot
            Plot.ploteo3Dpng(Datos,indices[0],GridB.isSelected(),opG,Path,axisOp.isSelected(),axis,triD.isSelected());}
        }else{//varios Plots
           for(int i=1;i<indices.length;i++){
                 if(Datos[indices[i-1]].labels.length==Datos[indices[i]].labels.length){
                  Flag=true;
                 }else{
                    JOptionPane.showMessageDialog(null,"[5]  The dimensions of the files does not match","Error Message",
                    JOptionPane.ERROR_MESSAGE);
                    i=indices.length;
                    Flag=false;
                 }
            }
                if(Flag==true){
                    if(Datos[indices[0]].labels.length==2){
                Plot.ploteo2DMulPng(Datos,indices,GridB.isSelected(),opG,SpitB.isSelected(),Path,axisOp.isSelected(),axis);
            }else{
                Plot.ploteo3DMulPng(Datos,indices,GridB.isSelected(),opG,SpitB.isSelected(),Path,axisOp.isSelected(),axis,
                triD.isSelected());
                    }}

        }
        }} else{
               String Path=Guardar.getAbsolutePath()+""+".png";
             if(indices.length==1){
            if(Datos[indices[0]].labels.length==2){
             Plot.ploteo2Dpng(Datos,indices[0],GridB.isSelected(),opG,Path,axisOp.isSelected(),axis);
            }else{
            Plot.ploteo3D(Datos,indices[0],GridB.isSelected(),opG,axisOp.isSelected(),axis,triD.isSelected());
            }
        }else{
           for(int i=1;i<indices.length;i++){
                 if(Datos[indices[i-1]].labels.length==Datos[indices[i]].labels.length){
                  Flag=true;
                 }else{
```

```
    JOptionPane.YES_NO_OPTION,
    JOptionPane.QUESTION_MESSAGE,null,opciones,"Aceptar");
    if (eleccion == JOptionPane.YES_OPTION)
    {
    String Path=Guardar.getAbsolutePath();
         if(indices.length==1){
        if(Datos[indices[0]].labels.length==2){
         Plot.ploteo2DEps(Datos,indices[0],GridB.isSelected(),opG,Path,axisOp.isSelected(),axis);
        }else{
        Plot.ploteo3DEps(Datos,indices[0],GridB.isSelected(),opG,Path,axisOp.isSelected(),axis,triD.isSelected());
        }
    }else{
       for(int i=1;i<indices.length;i++){//corrobora que todos los datos sean de la misma dimension
            if(Datos[indices[i-1]].labels.length==Datos[indices[i]].labels.length){
             Flag=true;
            }else{
               JOptionPane.showMessageDialog(null,"[5]  The dimensions of the files does not match","Error Message",
               JOptionPane.ERROR_MESSAGE);
               i=indices.length;
               Flag=false;
            }
        }
            if(Flag==true){
                if(Datos[indices[0]].labels.length==2){
           Plot.ploteo2DMulEps(Datos,indices,GridB.isSelected(),opG,SpitB.isSelected(),Path,axisOp.isSelected(),axis);
        }else{
            Plot.ploteo3DMulEps(Datos,indices,GridB.isSelected(),opG,SpitB.isSelected(),Path,axisOp.isSelected(),axis,
            triD.isSelected());
                }}

    }
    }} else{
          String Path=Guardar.getAbsolutePath()+""+".eps";
         if(indices.length==1){
        if(Datos[indices[0]].labels.length==2){
         Plot.ploteo2DEps(Datos,indices[0],GridB.isSelected(),opG,Path,axisOp.isSelected(),axis);
        }else{
                    JOptionPane.showMessageDialog(null,"[5]  The dimensions of the files does not match","Error Message",
                    JOptionPane.ERROR_MESSAGE);
                    i=indices.length;
                    Flag=false;
                }
            }
            if(Flag==true){
                if(Datos[indices[0]].labels.length==2){
               Plot.ploteo2DMulPng(Datos,indices,GridB.isSelected(),opG,SpitB.isSelected(),Path,axisOp.isSelected(),axis);
            }else{
                Plot.ploteo3DMulPng(Datos,indices,GridB.isSelected(),opG,SpitB.isSelected(),Path,axisOp.isSelected(),axis,
                triD.isSelected());
                    }}

        }

        }
        }else{
             JOptionPane.showMessageDialog(null,"[1]  File not loaded","Error Message",JOptionPane.ERROR_MESSAGE);
        }

        }

    }

    private void SavePdfMouseClicked(java.awt.event.MouseEvent evt) {
         if(archivoElegido==null||indices==null){
           JOptionPane.showMessageDialog(null,"[1]  File not loaded","Error Message",JOptionPane.ERROR_MESSAGE);}
       else{
        JFileChooser fc=new JFileChooser();
        fc.setFileFilter(filter2);
        int respuesta=fc.showSaveDialog(this); //Muestra el diálogo
        if (respuesta == JFileChooser.APPROVE_OPTION){
            File Guardar =fc.getSelectedFile();
            if(Guardar.exists()){
        Object [] opciones ={"Aceptar","Cancelar"};
        int eleccion = JOptionPane.showOptionDialog(rootPane,"Desea sobreescribir el archivo?","Mensaje de Confirmacion",
```

```
            Plot.ploteo3DEps(Datos,indices[0],GridB.isSelected(),opG,Path,axisOp.isSelected(),axis,triD.isSelected());
            }
        }else{
           for(int i=1;i<indices.length;i++){
                if(Datos[indices[i-1]].labels.length==Datos[indices[i]].labels.length){
                 Flag=true;
                }else{
                   JOptionPane.showMessageDialog(null,"[5]  The dimensions of the files does not match","Error Message",
                   JOptionPane.ERROR_MESSAGE);
                   i=indices.length;
                   Flag=false;
                }
           }
               if(Flag==true){
                   if(Datos[indices[0]].labels.length==2){
                  Plot.ploteo2DMulEps(Datos,indices,GridB.isSelected(),opG,SpitB.isSelected(),Path,axisOp.isSelected(),axis);
           }else{
               Plot.ploteo3DMulEps(Datos,indices,GridB.isSelected(),opG,SpitB.isSelected(),Path,axisOp.isSelected(),axis,
               triD.isSelected());
                   }}

        }

        }

        }else{
             JOptionPane.showMessageDialog(null,"[1]  File not loaded","Error Message",JOptionPane.ERROR_MESSAGE);
        }

        }
    }

    private void ExitBMouseClicked(java.awt.event.MouseEvent evt) {
        Object [] opciones ={"Aceptar","Cancelar"};
        int eleccion = JOptionPane.showOptionDialog(rootPane,"Do you want to Exit?","Confirmation message",
        JOptionPane.YES_NO_OPTION,
        JOptionPane.QUESTION_MESSAGE,null,opciones,"Aceptar");
         if (eleccion == JOptionPane.YES_OPTION)
         {
         System.exit(0);
         }else{
         }
    }

    @SuppressWarnings("empty-statement")
    private void OpFileMouseClicked(java.awt.event.MouseEvent evt) {
                        if(i<10){
            JFileChooser fc = new JFileChooser();
            fc.setFileFilter(filter);
            int respuesta = fc.showOpenDialog(this);

        //Comprobar si se ha pulsado Aceptar
        if (respuesta == JFileChooser.APPROVE_OPTION)
        {archivoElegido = fc.getSelectedFile();
            archivoElegido2 = fc.getSelectedFile();
                 try {
                     fstream = new FileInputStream(archivoElegido.getAbsolutePath());
                     fstrea = new FileInputStream(archivoElegido.getAbsolutePath());
                     fstrea1 = new FileInputStream(archivoElegido.getAbsolutePath());
                 } catch (FileNotFoundException ex) {
                     Logger.getLogger(Principal.class.getName()).log(Level.SEVERE, null, ex);
                 }

       if(archivoElegido.getAbsolutePath().endsWith(".txt")==true){
       LoadTxt load= new LoadTxt();
       int weight=load.anchoTxt(fstrea);
                     if(weight==0){
                        JOptionPane.showMessageDialog(null,"[2]  Error Txt, Empty File or inadmissible","Error Message",
                        JOptionPane.ERROR_MESSAGE);
                     }else{
                       Datos[i]= new Datos();
                       if(weight<=6){
                       Datos[i].Large=load.largoTxt(fstrea1);
                        StringTokenizer name;
```

```
                    name = new StringTokenizer(archivoElegido.getName(),".");
                    Datos[i].nombre=name.nextToken();
            try {
                Datos[i].Datos=load.fillTxt(archivoElegido,load.largoTxt(fstream),weight);
            } catch (IOException ex) {
                Logger.getLogger(Principal.class.getName()).log(Level.SEVERE, null, ex);
            }
                    if(load.Flag==true){
                    this.ListModel.addElement(Datos[i].nombre);
                    switch ( weight ) {
                        case 2:
                        Label2d obj= new Label2d(this, true);
                        obj.setVisible(true);
                        Datos[i].labels= new String[weight];
                        Datos[i].labels[0]=obj.jTextField1.getText();
                        Datos[i].labels[1]=obj.jTextField2.getText();
                        i++;
                        break;
                        case 3:
                        Label3d obj1= new Label3d(this, true);
                        obj1.setVisible(true);
                        Datos[i].labels= new String[weight];
                        Datos[i].labels[0]=obj1.jTextField1.getText();
                        Datos[i].labels[1]=obj1.jTextField2.getText();
                        Datos[i].labels[2]=obj1.jTextField3.getText();
                        i++;
                            break;
                        default:
                        JOptionPane.showMessageDialog(null,"[4]  Plot error exceeds more than 3 dimensions","Error Message",
                        JOptionPane.ERROR_MESSAGE);
                             if(weight==4){
                            Datos[i].labels= new String[weight];
                            Datos[i].labels[0]="x";
                            Datos[i].labels[1]="y";
                            Datos[i].labels[2]="z";
                            Datos[i].labels[3]="k";
                            }else{
                             if(weight==5){
                            Datos[i].labels= new String[weight];
                            Datos[i].labels[0]="x";
                            Datos[i].labels[1]="y";
                            Datos[i].labels[2]="z";
                            Datos[i].labels[3]="k";
                            Datos[i].labels[4]="l";
                             }else{
                              if(weight==6){
                            Datos[i].labels= new String[weight];
                            Datos[i].labels[0]="x";
                            Datos[i].labels[1]="y";
                            Datos[i].labels[2]="z";
                            Datos[i].labels[3]="k";
                            Datos[i].labels[4]="l";
                            Datos[i].labels[5]="m";
                              }
                             }

                            }
                            break;
                    }
                    }}else{
                        JOptionPane.showMessageDialog(null,"[6]  You have exceeded the maximum of axis","Error Message",
                        JOptionPane.ERROR_MESSAGE);
                    }}
        }else{
        if(archivoElegido.getAbsolutePath().endsWith(".xls")==true){
                LoadXls xls= new LoadXls();
                List sheetData = new ArrayList();
                try {
                     sheetData=xls.CargarExcel(archivoElegido.getAbsolutePath());
                    } catch (IOException ex) {
                     Logger.getLogger(Principal.class.getName()).log(Level.SEVERE, null, ex);
                                        }
                    Datos[i]= new Datos();
                    if(xls.SizeY(sheetData)<=6){
```

```
Datos[i].Large=xls.SizeX(sheetData);
StringTokenizer name;
name = new StringTokenizer(archivoElegido.getName(),".");
Datos[i].nombre=name.nextToken();
xls.showExelData(sheetData,Datos[i]);
if(Datos[i].labels[0]==null){
   if(xls.Flag==true){
  switch ( xls.SizeY(sheetData) ) {
       case 2:
       Label2d obj= new Label2d(this, true);
       obj.setVisible(true);
       Datos[i].labels= new String[xls.SizeY(sheetData)];
       Datos[i].labels[0]=obj.jTextField1.getText();
       Datos[i].labels[1]=obj.jTextField2.getText();
       this.ListModel.addElement(Datos[i].nombre);
       i++;
       break;
       case 3:
       Label3d obj1= new Label3d(this, true);
       obj1.setVisible(true);
       Datos[i].labels= new String[xls.SizeY(sheetData)];
       Datos[i].labels[0]=obj1.jTextField1.getText();
       Datos[i].labels[1]=obj1.jTextField2.getText();
       Datos[i].labels[2]=obj1.jTextField3.getText();
       this.ListModel.addElement(archivoElegido.getName());
       i++;
           break;
       default:
       JOptionPane.showMessageDialog(null,"[4]  Plot error exceeds more than 3 dimensions","Error Message",
       JOptionPane.ERROR_MESSAGE);
           if(xls.SizeY(sheetData)==4){
           Datos[i].labels= new String[xls.SizeY(sheetData)];
           Datos[i].labels[0]="x";
           Datos[i].labels[1]="y";
           Datos[i].labels[2]="z";
           Datos[i].labels[3]="k";
           }else{
            if(xls.SizeY(sheetData)==5){
           Datos[i].labels= new String[xls.SizeY(sheetData)];
           Datos[i].labels[0]="x";
           Datos[i].labels[1]="y";
           Datos[i].labels[2]="z";
           Datos[i].labels[3]="k";
           Datos[i].labels[4]="l";
            }else{
             if(xls.SizeY(sheetData)==6){
           Datos[i].labels= new String[xls.SizeY(sheetData)];
           Datos[i].labels[0]="x";
           Datos[i].labels[1]="y";
           Datos[i].labels[2]="z";
           Datos[i].labels[3]="k";
           Datos[i].labels[4]="l";
           Datos[i].labels[4]="m";
             }
            }

           }

           break;
   }
}}else{
if(xls.lab<2){
JOptionPane.showMessageDialog(null,"[3]  Error Xls, Empty File or inadmissible","Error Message",
JOptionPane.ERROR_MESSAGE);
}else{
this.ListModel.addElement(archivoElegido.getName());
}
}}else{
 JOptionPane.showMessageDialog(null,"[6]  You have exceeded the maximum of axis","Error Message",
 JOptionPane.ERROR_MESSAGE);
 }
                                                    }else{
                                                    System.out.println("otros formatos no es compatibles");
                                                        }
```

```
                                    }
                }
                }else{
                   JOptionPane.showMessageDialog(null,"[8]  Limit reached Files","Error Message",JOptionPane.ERROR_MESSAGE);
                   }
    }

    private void MetricsMouseClicked(java.awt.event.MouseEvent evt) {

    }

    private void spamenuMouseClicked(java.awt.event.MouseEvent evt) {
        if(archivoElegido==null){
            JOptionPane.showMessageDialog(null,"[1] File not loaded","Error Message",JOptionPane.ERROR_MESSAGE);}
        else{
            Spa obj= new Spa(this,true);
            obj.pasarD(Datos, i);// arreglar esto
            obj.setVisible(true);
        }

    }

    private void jMenu3MouseClicked(java.awt.event.MouseEvent evt) {
        if(archivoElegido==null||indices==null){
            JOptionPane.showMessageDialog(null,"[1]  File not loaded","Error Message",JOptionPane.ERROR_MESSAGE);}
        else{
            if(indices.length<4 && indices.length>=2){
                DG obj= new DG(this, true);
                obj.pasarD(Datos,indices);
                for(int i=0;i<indices.length;i++){
                    obj.Com1.addItem(Datos[indices[i]].nombre);
                }
                obj.setVisible(true);
            }else{
                JOptionPane.showMessageDialog(null,"[7]  Data error Amount","Error Message",JOptionPane.ERROR_MESSAGE);
            }
        }
    }

    private void jMenu1MouseClicked(java.awt.event.MouseEvent evt) {
        if(archivoElegido==null||indices==null){
            JOptionPane.showMessageDialog(null,"[1]  File not loaded","Error Message",JOptionPane.ERROR_MESSAGE);}
        else{
            if(indices.length<4 && indices.length>=2){
                MER obj= new MER(this, true);
                obj.pasarD(Datos,indices);
                for(int i=0;i<indices.length;i++){
                    obj.Com1.addItem(Datos[indices[i]].nombre);
                }
                obj.setVisible(true);
            }else{
                JOptionPane.showMessageDialog(null,"[7]  Data error Amount ","Error Message",JOptionPane.ERROR_MESSAGE);
            }
        }
    }

    private void SaveBMouseClicked(java.awt.event.MouseEvent evt) {
if(archivoElegido==null||indices==null){
            JOptionPane.showMessageDialog(null,"[1]  File not loaded","Error Message",JOptionPane.ERROR_MESSAGE);}
       else{
        JFileChooser fc=new JFileChooser();
        fc.setFileFilter(filter1);
        int respuesta=fc.showSaveDialog(this); //Muestra el diálogo
        if (respuesta == JFileChooser.APPROVE_OPTION){
            File Guardar =fc.getSelectedFile();
            if(Guardar.exists()){
        Object [] opciones ={"Aceptar","Cancelar"};
        int eleccion = JOptionPane.showOptionDialog(rootPane,"OverWrite the file?","Mensaje de Confirmacion",
        JOptionPane.YES_NO_OPTION,
        JOptionPane.QUESTION_MESSAGE,null,opciones,"Aceptar");
        if (eleccion == JOptionPane.YES_OPTION)
        {
        String Path=Guardar.getAbsolutePath();
            if(indices.length==1){
```

```
            if(Datos[indices[0]].labels.length==2){
             Plot.ploteo2Dpng(Datos,indices[0],GridB.isSelected(),opG,Path,axisOp.isSelected(),axis);
            }else{
            Plot.ploteo3Dpng(Datos,indices[0],GridB.isSelected(),opG,Path,axisOp.isSelected(),axis,triD.isSelected());
            }
        }else{
           for(int i=1;i<indices.length;i++){
                if(Datos[indices[i-1]].labels.length==Datos[indices[i]].labels.length){
                 Flag=true;
                }else{
                   JOptionPane.showMessageDialog(null,"[5]  The dimensions of the files does not match","Error Message",
                   JOptionPane.ERROR_MESSAGE);
                   i=indices.length;
                   Flag=false;
                }
           }
               if(Flag==true){
                   if(Datos[indices[0]].labels.length==2){
                  Plot.ploteo2DMulPng(Datos,indices,GridB.isSelected(),opG,SpitB.isSelected(),Path,axisOp.isSelected(),axis);
           }else{
               Plot.ploteo3DMulPng(Datos,indices,GridB.isSelected(),opG,SpitB.isSelected(),Path,axisOp.isSelected(),axis,
               triD.isSelected());
                   }}

        }
        }} else{
               String Path=Guardar.getAbsolutePath()+""+".png";
            if(indices.length==1){
            if(Datos[indices[0]].labels.length==2){
             Plot.ploteo2Dpng(Datos,indices[0],GridB.isSelected(),opG,Path,axisOp.isSelected(),axis);
            }else{
            Plot.ploteo3D(Datos,indices[0],GridB.isSelected(),opG,axisOp.isSelected(),axis,triD.isSelected());
            }
        }else{
           for(int i=1;i<indices.length;i++){
                if(Datos[indices[i-1]].labels.length==Datos[indices[i]].labels.length){
                 Flag=true;
                }else{
                   JOptionPane.showMessageDialog(null,"[5]  The dimensions of the files does not match","Error Message",
                   JOptionPane.ERROR_MESSAGE);
                   i=indices.length;
                   Flag=false;
                }
           }
               if(Flag==true){
                   if(Datos[indices[0]].labels.length==2){
                  Plot.ploteo2DMulPng(Datos,indices,GridB.isSelected(),opG,SpitB.isSelected(),Path,axisOp.isSelected(),axis);
           }else{
               Plot.ploteo3DMulPng(Datos,indices,GridB.isSelected(),opG,SpitB.isSelected(),Path,axisOp.isSelected(),axis,
               triD.isSelected());
                   }}

        }

        }

        }else{
             JOptionPane.showMessageDialog(null,"[1]  File not loaded","Error Message",JOptionPane.ERROR_MESSAGE);
        }
        }

    }

    private void jButton2MouseClicked(java.awt.event.MouseEvent evt) {
if(archivoElegido==null||indices==null){
            JOptionPane.showMessageDialog(null,"[1]  File not loaded","Error Message",JOptionPane.ERROR_MESSAGE);}
        else{
        JFileChooser fc=new JFileChooser();
        fc.setFileFilter(filter2);
        int respuesta=fc.showSaveDialog(this); //Muestra el diálogo
        if (respuesta == JFileChooser.APPROVE_OPTION){
            File Guardar =fc.getSelectedFile();
            if(Guardar.exists()){
        Object [] opciones ={"Aceptar","Cancelar"};
```

```
    int eleccion = JOptionPane.showOptionDialog(rootPane,"Overwrite the File?","Mensaje de Confirmacion",
    JOptionPane.YES_NO_OPTION,
    JOptionPane.QUESTION_MESSAGE,null,opciones,"Aceptar");
    if (eleccion == JOptionPane.YES_OPTION)
    {
    String Path=Guardar.getAbsolutePath();
         if(indices.length==1){
        if(Datos[indices[0]].labels.length==2){
         Plot.ploteo2DEps(Datos,indices[0],GridB.isSelected(),opG,Path,axisOp.isSelected(),axis);
        }else{
        Plot.ploteo3DEps(Datos,indices[0],GridB.isSelected(),opG,Path,axisOp.isSelected(),axis,triD.isSelected());
        }
    }else{
       for(int i=1;i<indices.length;i++){
            if(Datos[indices[i-1]].labels.length==Datos[indices[i]].labels.length){
             Flag=true;
            }else{
               JOptionPane.showMessageDialog(null,"[5]   The dimensions of the files does not match","Error Message",
               JOptionPane.ERROR_MESSAGE);
               i=indices.length;
               Flag=false;
            }
        }
            if(Flag==true){
               if(Datos[indices[0]].labels.length==2){
           Plot.ploteo2DMulEps(Datos,indices,GridB.isSelected(),opG,SpitB.isSelected(),Path,axisOp.isSelected(),axis);
        }else{
            Plot.ploteo3DMulEps(Datos,indices,GridB.isSelected(),opG,SpitB.isSelected(),Path,axisOp.isSelected(),axis,
            triD.isSelected());
                }}

    }
    }} else{
           String Path=Guardar.getAbsolutePath()+""+".eps";
         if(indices.length==1){
        if(Datos[indices[0]].labels.length==2){
         Plot.ploteo2DEps(Datos,indices[0],GridB.isSelected(),opG,Path,axisOp.isSelected(),axis);
        }else{
        Plot.ploteo3DEps(Datos,indices[0],GridB.isSelected(),opG,Path,axisOp.isSelected(),axis,triD.isSelected());
        }
    }else{
       for(int i=1;i<indices.length;i++){
            if(Datos[indices[i-1]].labels.length==Datos[indices[i]].labels.length){
             Flag=true;
            }else{
               JOptionPane.showMessageDialog(null,"[5]  The dimensions of the files does not match","Error Message",
               JOptionPane.ERROR_MESSAGE);
               i=indices.length;
               Flag=false;
            }
        }
            if(Flag==true){
               if(Datos[indices[0]].labels.length==2){
           Plot.ploteo2DMulEps(Datos,indices,GridB.isSelected(),opG,SpitB.isSelected(),Path,axisOp.isSelected(),axis);
        }else{
            Plot.ploteo3DMulEps(Datos,indices,GridB.isSelected(),opG,SpitB.isSelected(),Path,axisOp.isSelected(),axis,
            triD.isSelected());
                }}

    }

    }

    }else{
         JOptionPane.showMessageDialog(null,"[1]  File not loaded","Error Message",JOptionPane.ERROR_MESSAGE);
    }

    }        // TODO add your handling code here:
}

private void axisOpActionPerformed(java.awt.event.ActionEvent evt) {
    if(axisOp.isSelected()==true){
        minX.enable();
        minY.enable();
```

```
            maxX.enable();
            maxY.enable();
            if(triD.isSelected()==true)
                minZ.enable();
                maxZ.enable();

        }else{
        minX.disable();
        minY.disable();
        maxX.disable();
        maxY.disable();
        if(triD.isSelected()==true){
            minZ.disable();
            maxZ.disable();
        }
        }
    }

    private void minXKeyTyped(java.awt.event.KeyEvent evt) {
        char car = evt.getKeyChar();
        if((car<'0' || car>'9')) evt.consume();
    }

    private void maxXKeyTyped(java.awt.event.KeyEvent evt) {
         char car = evt.getKeyChar();
         if((car<'0' || car>'9')) evt.consume();
    }

    private void minYKeyTyped(java.awt.event.KeyEvent evt) {
         char car = evt.getKeyChar();
         if((car<'0' || car>'9')) evt.consume();
    }

    private void maxYKeyTyped(java.awt.event.KeyEvent evt) {
        char car = evt.getKeyChar();
        if((car<'0' || car>'9')) evt.consume();
    }
    private void maxXMouseExited(java.awt.event.MouseEvent evt) {
      if(maxX.getText().length()==0){
      }else{
          if(minX.getText().length()!=0&&Integer.parseInt(maxX.getText())>=Integer.parseInt(minX.getText())){
      axis[0]=Integer.parseInt(maxX.getText());
      }else{
        JOptionPane.showMessageDialog(null,"[17]  The minimum may not exceed maximum","Error Message",
        JOptionPane.ERROR_MESSAGE); }      }
    }

    private void minXMouseExited(java.awt.event.MouseEvent evt) {
      if(minX.getText().length()==0){
      }else{
      axis[1]=Integer.parseInt(minX.getText());}
    }

    private void minYMouseExited(java.awt.event.MouseEvent evt) {
       if(minY.getText().length()==0){
       }else{
      axis[2]=Integer.parseInt(minY.getText());}

    }

    private void maxYMouseExited(java.awt.event.MouseEvent evt) {
      if(maxY.getText().length()==0){
      }else{
          if(minY.getText().length()!=0&&Integer.parseInt(maxY.getText())>=Integer.parseInt(minY.getText())){
      axis[3]=Integer.parseInt(maxY.getText());
      }else{
        JOptionPane.showMessageDialog(null,"[17]  The minimum may not exceed maximum","Error Message",
        JOptionPane.ERROR_MESSAGE); }      }
      if(maxY.getText().length()==0||minY.getText().length()==0||maxX.getText().length()==0||minX.getText().length()==0){
        JOptionPane.showMessageDialog(null,"[9] Missing information ","Error Message",JOptionPane.ERROR_MESSAGE);
      }
    }

    private void triDActionPerformed(java.awt.event.ActionEvent evt) {
```

```
        if(triD.isSelected()==true){
        jLabel5.setVisible(true);
        minZ.setVisible(true);
        maxZ.setVisible(true);
        if(axisOp.isSelected()==true){
                minZ.enable();
                maxZ.enable();
        }
        }

        if(triD.isSelected()==false){
        jLabel5.setVisible(false);
        minZ.setVisible(false);
        maxZ.setVisible(false);
        }


    }

    private void minZKeyTyped(java.awt.event.KeyEvent evt) {
        char car = evt.getKeyChar();
        if((car<'0' || car>'9')) evt.consume();
    }

    private void maxZKeyTyped(java.awt.event.KeyEvent evt) {
        char car = evt.getKeyChar();
        if((car<'0' || car>'9')) evt.consume();
    }

    private void minZMouseExited(java.awt.event.MouseEvent evt) {
        if(minZ.getText().length()==0){
        }else{
        axis[4]=Integer.parseInt(minZ.getText());}
    }

    private void maxZMouseExited(java.awt.event.MouseEvent evt) {
         if(maxZ.getText().length()==0){
     }else{
         if(minZ.getText().length()!=0&&Integer.parseInt(maxZ.getText())>=Integer.parseInt(minZ.getText())){
     axis[5]=Integer.parseInt(maxZ.getText());
     }else{
         JOptionPane.showMessageDialog(null,"[17]  The minimum may not exceed maximum","Error Message",
         JOptionPane.ERROR_MESSAGE); }}
      if(maxY.getText().length()==0||minY.getText().length()==0||maxX.getText().length()==0||minX.getText().length()==0||
         maxZ.getText().length()==0||minZ.getText().length()==0){
         JOptionPane.showMessageDialog(null,"[9] Missing information ","Error Message",JOptionPane.ERROR_MESSAGE);
     }
    }

    private void HelpActionPerformed(java.awt.event.ActionEvent evt) {

    }

    private void HelpMouseClicked(java.awt.event.MouseEvent evt) {
        goToURL();
    }
    /**
     * @param args the command line arguments
     */
    @SuppressWarnings("ResultOfObjectAllocationIgnored")
    public  void main(String args[]) {
        /* Set the Nimbus look and feel */
        Look and feel setting code (optional)
    }
    // Variables declaration - do not modify
    private javax.swing.JMenu ClearB;
    private javax.swing.JMenu ExitB;
    private javax.swing.JMenu File;
    public javax.swing.JRadioButton GridB;
    private javax.swing.JMenu Help;
    public javax.swing.JRadioButton JoinB;
    public javax.swing.JRadioButton LiPointB;
    public javax.swing.JRadioButton LinesB;
    public javax.swing.JList Lista;
```

```
    private javax.swing.JMenu Metrics;
    private javax.swing.JMenu OpFile;
    public javax.swing.JButton OpenF;
    public javax.swing.JButton PlotB;
    public javax.swing.JRadioButton PointB;
    public javax.swing.JButton SaveB;
    private javax.swing.JMenu SavePdf;
    private javax.swing.JMenu SavePng;
    public javax.swing.JRadioButton SpitB;
    private javax.swing.JToolBar ToolsBar;
    private javax.swing.JMenu Visual;
    private javax.swing.JRadioButton axisOp;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JMenu jMenu1;
    private javax.swing.JMenu jMenu3;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JPopupMenu.Separator jSeparator1;
    private javax.swing.JToolBar.Separator jSeparator2;
    private javax.swing.JToolBar.Separator jSeparator3;
    private javax.swing.JSeparator jSeparator4;
    private javax.swing.JToolBar.Separator jSeparator6;
    private javax.swing.JSeparator jSeparator7;
    private javax.swing.JSeparator jSeparator8;
    private javax.swing.JTextField maxX;
    private javax.swing.JTextField maxY;
    private javax.swing.JTextField maxZ;
    private javax.swing.JTextField minX;
    private javax.swing.JTextField minY;
    private javax.swing.JTextField minZ;
    private javax.swing.JMenu spamenu;
    private javax.swing.JRadioButton triD;
    // End of variables declaration
}
```

Anexo II: Código Datos.Java.

```
public class Datos extends Object {
    public String nombre;
    public String[] labels;
    public String Ruta;
    public double[][] Datos;
    public int Large;
   Datos() {
    }
}
```

Anexo III: Código LoadXls.Java.

```
public class LoadXls extends Object{
public Boolean Flag=true;
public int lab;
LoadXls(){}

// calcula el ancho de datos
  public static int SizeX(List SheetData){
       return SheetData.size();
    }
  //calcula el largo de los datos
  public static int SizeY(List SheetData){
           List list = (List) SheetData.get(1);
           return list.size();
    }
public  void showExelData(List sheetData,Datos datos) {
       datos.labels= new String[SizeY(sheetData)];// sizeX me define cuantas variables poseo
       datos.Datos = new double[SizeX(sheetData)][SizeY(sheetData)];//sizeY define el largo de los datos que poseo
       for (int i = 0; i < sheetData.size(); i++) {
           List list = (List) sheetData.get(i);
           for (int j = 0; j < list.size(); j++) {
               Cell cell = (Cell) list.get(j);
               if (cell.getCellType() == Cell.CELL_TYPE_NUMERIC) {// se encarga de verificar si los valores son numericos
                   datos.Datos[i][j]=cell.getNumericCellValue();
               } else if (cell.getCellType() == Cell.CELL_TYPE_STRING) {
                   if(i>1){// acepta que la primera linea sea con nombre de las axis
                   JOptionPane.showMessageDialog(null,"[12]  Error in xls format, please check the data","Error Message",
                   JOptionPane.ERROR_MESSAGE);
                    Flag=false;
                   }else{
                   datos.labels[j]=cell.getRichStringCellValue().getString();
                   lab++;
                   Flag= true;}
               } else if( cell.getCellType()==cell.CELL_TYPE_BLANK){// verifica si hay espacios vacios
               JOptionPane.showMessageDialog(null, "[13] no empty spaces are allowed ");}
               if (j < list.size() - 1) {
                   System.out.print("  ");
               }


           }
       }
}
//carga el archivo xls y lo convierte en una lista
   public List CargarExcel(String Archivo) throws IOException{
       String filename = Archivo;
       List sheetData = new ArrayList();
       FileInputStream fis = null;
         try {
           fis = new FileInputStream(filename);
           HSSFWorkbook workbook = new HSSFWorkbook(fis);
           HSSFSheet sheet = workbook.getSheetAt(0);
           Iterator rows = sheet.rowIterator();
           while (rows.hasNext()) {
               HSSFRow row = (HSSFRow) rows.next();
               Iterator cells = row.cellIterator();
               List data = new ArrayList();
               while (cells.hasNext()) {
                   HSSFCell cell = (HSSFCell) cells.next();
                   data.add(cell);
               }
               sheetData.add(data);
           }
       } catch (IOException e) {
           e.printStackTrace();
       } finally {
           if (fis != null) {
                fis.close();
           }
       }
         return sheetData;
}
}
```

Anexo IV: Código LoadTxt.Java.

```
public class LoadTxt extends Object {
    public String nombre;
    public String labels;
    public Boolean Flag;
    public double[][] f;
    LoadTxt() {
    }
    public static boolean isNumeric(String str) {// funcion que verifica si es un caracter
        return (str.matches("[+-]?\\d*(\\.\\d+)?") && str.equals("")==false);
    }

public int largoTxt(FileInputStream fstream){
    int i = 0;
     try{
            DataInputStream entrada = new DataInputStream(fstream);
            BufferedReader buffer = new BufferedReader(new InputStreamReader(entrada));
            String strLinea = buffer.readLine();
            do{
                if(strLinea.isEmpty()!=true){
                i++;}else{
                            }
              }while ((strLinea = buffer.readLine()) != null);
            entrada.close();
        }catch (IOException e){
            System.err.println("[10]An error occurred in the long of the file: " + e.getMessage());
        }
    return i;
        }
   public int anchoTxt(FileInputStream fstream){//verificar en caso de un dato faltante
    int i = 0;
     try{
            DataInputStream entrada = new DataInputStream(fstream);
            // Creamos el Buffer de Lectura
            BufferedReader buffer = new BufferedReader(new InputStreamReader(entrada));
            String strLinea = buffer.readLine();
            StringTokenizer st = new StringTokenizer(strLinea);
            i=st.countTokens();
```

```
        do{
            st = new StringTokenizer(strLinea);
            if(i!=st.countTokens()){
                System.err.println("[11] Size error");
                entrada.close();
                return 0;
            }
           }while ((strLinea = buffer.readLine()) != null);
           entrada.close();
       }catch (Exception e){
           System.err.println("[10]An error occurred in the long of the file: " + e.getMessage());
       }
   return i; }


   @SuppressWarnings("empty-statement")
   //funcion se encarga de leer las filas del txt, corroborar si es un valor y almacenarlas en un double
  public double [][] fillTxt(File archivo, int i, int k) throws IOException{
                   int x=0;
                   FileInputStream fstream = new FileInputStream(archivo.getAbsolutePath());
                   DataInputStream entrada = new DataInputStream(fstream);
                   BufferedReader buffer = new BufferedReader(new InputStreamReader(entrada));
                   String strLinea = buffer.readLine();
                   StringTokenizer st;
                   String check;
                   double [][] f1 = new double [i][k];
                   try {
                       do{
                           st = new StringTokenizer(strLinea," ");
                           for(int j=0; j<k;j++){
                             check=st.nextToken();
                              if (isNumeric(check) == true) {// corrobora si el dato no es caracter
                               f1[x][j]= Double.parseDouble(check);
                           }else{
                          JOptionPane.showMessageDialog(null,"[12] Error in txt format, please check the data","Mensaje de Error",
                          JOptionPane.ERROR_MESSAGE);
                           Flag=false;
                                   return f1;
                                     }}
                                   x++;
                           }while ((strLinea = buffer.readLine()) != null);
                       } catch (IOException ex) {
                           Logger.getLogger(Principal.class.getName()).log(Level.SEVERE, null, ex);
                       }
                       entrada.close();
           Flag=true;
           return f1;
       }


}
```

Anexo V: Código Plot.Java.

```
public class Plot extends Object{
    Plot(){}

   public static void ploteo3D(Datos[] Datos, int i, boolean grid, int opcion, boolean axis,int[] range,boolean trid){
        PlotStyle styleDeleted = new PlotStyle();
        switch(opcion){
            case 0:
                styleDeleted.setStyle(Style.LINES);
            break;
            case 1:
                styleDeleted.setStyle(Style.POINTS);
                break;
            case 2:
                styleDeleted.setStyle(Style.LINESPOINTS);
                break;}
        styleDeleted.setLineType(NamedPlotColor.RED);
        DataSetPlot setDeleted = new DataSetPlot(Datos[i].Datos);
        setDeleted.setPlotStyle(styleDeleted);
        setDeleted.setTitle(Datos[i].nombre);
        JavaPlot p = new JavaPlot(true);
        p.setTitle(Datos[i].nombre);
        p.getAxis("x").setLabel(Datos[i].labels[0]);
        p.getAxis("y").setLabel(Datos[i].labels[1]);
        p.getAxis("z").setLabel(Datos[i].labels[2]);
        if(grid==true){
         p.set("grid", "back ls 12");}
        if(axis==true&&trid==true){
            p.getAxis("x").setBoundaries(range[0],range[1]);
            p.getAxis("y").setBoundaries(range[2],range[3]);
            p.getAxis("z").setBoundaries(range[4],range[5]);}
        p.addPlot(setDeleted);
        p.plot();}

    public static void ploteo2D(Datos[] Datos, int sel, boolean grid,int opcion, boolean axis,int[] range){
        PlotStyle styleDeleted = new PlotStyle();
        switch(opcion){
            case 0:
                styleDeleted.setStyle(Style.LINES);
           break;
           case 1:
               styleDeleted.setStyle(Style.POINTS);
               break;
           case 2:
               styleDeleted.setStyle(Style.LINESPOINTS);
               break;}
       styleDeleted.setLineType(NamedPlotColor.RED);
       DataSetPlot setDeleted = new DataSetPlot(Datos[sel].Datos);
       setDeleted.setPlotStyle(styleDeleted);
       setDeleted.setTitle(Datos[sel].nombre);
       JavaPlot p = new JavaPlot();
       System.out.print(Datos[sel].nombre);
       p.setTitle(Datos[sel].nombre);
       p.getAxis("x").setLabel(Datos[sel].labels[0]);
       p.getAxis("y").setLabel(Datos[sel].labels[1]);
       if(grid==true){
        p.set("grid", "back ls 12");}
       if(axis==true){
           p.getAxis("x").setBoundaries(range[0],range[1]);
           p.getAxis("y").setBoundaries(range[2],range[3]);}
       p.addPlot(setDeleted);
       p.plot();}

       public static void ploteo2DMul(Datos[] Datos, int[] i, boolean grid,int opcion,boolean split, boolean axis,int[] range){
       PlotStyle styleDeleted = new PlotStyle();
       DataSetPlot Plotdata[] = new DataSetPlot[3];
       switch(opcion){
           case 0:
               styleDeleted.setStyle(Style.LINES);
           break;
           case 1:
               styleDeleted.setStyle(Style.POINTS);
               break;
           case 2:
               styleDeleted.setStyle(Style.LINESPOINTS);
```

```
            break;}
        for(int e=0; e<i.length;e++){
        Plotdata[e] = new DataSetPlot(Datos[i[e]].Datos);
        Plotdata[e].setPlotStyle(styleDeleted);
        Plotdata[e].setTitle(Datos[i[e]].nombre);}
        JavaPlot p = new JavaPlot();
        if(split==true){
        for(int e=0; e<i.length;e++){
         p.addPlot(Plotdata[e]);
         if(e!=i.length-1){
         p.newGraph();}}
        }else{
        for(int e=0; e<i.length;e++){
        p.addPlot(Plotdata[e]);}
        if(axis==true){
            p.getAxis("x").setBoundaries(range[0],range[1]);
            p.getAxis("y").setBoundaries(range[2],range[3]);}}
         if(grid==true){
         p.set("grid", "back ls 12");}
         p.plot();}

        public static void ploteo3DMul(Datos[] Datos, int[] i, boolean grid,
        int opcion,boolean split, boolean axis,int[] range,boolean trid){
        PlotStyle styleDeleted = new PlotStyle();
        DataSetPlot Plotdata[] = new DataSetPlot[3];
        switch(opcion){
            case 0:
                styleDeleted.setStyle(Style.LINES);
            break;
            case 1:
                styleDeleted.setStyle(Style.POINTS);
                break;
            case 2:
                styleDeleted.setStyle(Style.LINESPOINTS);
                break;}
        for(int e=0; e<i.length;e++){
        Plotdata[e] = new DataSetPlot(Datos[i[e]].Datos);
        Plotdata[e].setPlotStyle(styleDeleted);
        Plotdata[e].setTitle(Datos[i[e]].nombre);}
        JavaPlot p = new JavaPlot(true);
        if(split==true){
        for(int e=0; e<i.length;e++){
         p.addPlot(Plotdata[e]);
         if(e!=i.length-1){
         p.newGraph3D();}}
        }else{
        for(int e=0; e<i.length;e++){
        p.addPlot(Plotdata[e]);}
        if(axis==true&&trid==true){
            p.getAxis("x").setBoundaries(range[0],range[1]);
            p.getAxis("y").setBoundaries(range[2],range[3]);
            p.getAxis("z").setBoundaries(range[4],range[5]);}}
         if(grid==true){
         p.set("grid", "back ls 12");}
        p.plot();}

    public static void ploteo2Dpng(Datos[] Datos, int i, boolean grid,int opcion,String path, boolean axis,int[] range){
        PlotStyle styleDeleted = new PlotStyle();
        switch(opcion){
            case 0:
                styleDeleted.setStyle(Style.LINES);
            break;
            case 1:
                styleDeleted.setStyle(Style.POINTS);
                break;
            case 2:
                styleDeleted.setStyle(Style.LINESPOINTS);
                break;}
        styleDeleted.setLineType(NamedPlotColor.RED);
        DataSetPlot setDeleted = new DataSetPlot(Datos[i].Datos);
        setDeleted.setPlotStyle(styleDeleted);
        JavaPlot p = new JavaPlot();
        p.setTitle(Datos[i].nombre);
        ImageTerminal png = new ImageTerminal();
```

```
        File file = new File(path);
        p.getAxis("x").setLabel(Datos[i].labels[0]); //nombre coor
        p.getAxis("y").setLabel(Datos[i].labels[1]);
        if(grid==true){
         p.set("grid", "back ls 12");
        }if(axis==true){
            p.getAxis("x").setBoundaries(range[0],range[1]);
            p.getAxis("y").setBoundaries(range[2],range[3]);}
        p.setTerminal(png);
        p.addPlot(setDeleted);
        p.plot();
        try {
        ImageIO.write(png.getImage(), "png", file);
      } catch (IOException ex) {
        System.err.print(ex);}}

     public static void ploteo3Dpng(Datos[] Datos, int i, boolean grid,int opcion,String path, boolean axis,
     int[] range,boolean trid){
        PlotStyle styleDeleted = new PlotStyle();
        switch(opcion){
            case 0:
                styleDeleted.setStyle(Style.LINES);
            break;
            case 1:
                styleDeleted.setStyle(Style.POINTS);
                break;
            case 2:
                styleDeleted.setStyle(Style.LINESPOINTS);
                break;}
        styleDeleted.setLineType(NamedPlotColor.RED);
        DataSetPlot setDeleted = new DataSetPlot(Datos[i].Datos);
        setDeleted.setPlotStyle(styleDeleted);
        JavaPlot p = new JavaPlot(true);
        p.setTitle(Datos[i].nombre);
        ImageTerminal png = new ImageTerminal();
        File file = new File(path);
        p.getAxis("x").setLabel(Datos[i].labels[0]);
        p.getAxis("y").setLabel(Datos[i].labels[1]);
        if(grid==true){
         p.set("grid", "back ls 12");}
        if(axis==true&&trid==true){
            p.getAxis("x").setBoundaries(range[0],range[1]);
            p.getAxis("y").setBoundaries(range[2],range[3]);
            p.getAxis("z").setBoundaries(range[4],range[5]);}
        p.setTerminal(png);
        p.addPlot(setDeleted);
        p.plot();
        try {
        ImageIO.write(png.getImage(), "png", file);
      } catch (IOException ex) {
        System.err.print(ex);}}

    public static void ploteo2DMulPng(Datos[] Datos, int[] i, boolean grid,int opcion,boolean split,String path,
    boolean axis,int[] range){
        PlotStyle styleDeleted = new PlotStyle();
        DataSetPlot Plotdata[] = new DataSetPlot[3];
        switch(opcion){
            case 0:
                styleDeleted.setStyle(Style.LINES);
            break;
            case 1:
                styleDeleted.setStyle(Style.POINTS);
                break;
            case 2:
                styleDeleted.setStyle(Style.LINESPOINTS);
                break;}
        for(int e=0; e<i.length;e++){
        Plotdata[e] = new DataSetPlot(Datos[i[e]].Datos);
        Plotdata[e].setPlotStyle(styleDeleted);
        Plotdata[e].setTitle(Datos[i[e]].nombre);}
        JavaPlot p = new JavaPlot();
        ImageTerminal png = new ImageTerminal();
        File file = new File(path);
        if(split==true){
```

```
        p.setTerminal(png);
        for(int e=0; e<i.length;e++){
         p.addPlot(Plotdata[e]);
         if(e!=i.length-1){
         p.newGraph();}}
        }else{
        p.setTerminal(png);
        for(int e=0; e<i.length;e++){
        p.addPlot(Plotdata[e]);}
        if(axis==true){
            p.getAxis("x").setBoundaries(range[0],range[1]);
            p.getAxis("y").setBoundaries(range[2],range[3]);}}
         if(grid==true){
         p.set("grid", "back ls 12");}
        p.plot();
        try {
        ImageIO.write(png.getImage(), "png", file);
    } catch (IOException ex) {
        System.err.print(ex);}}

public static void ploteo3DMulPng(Datos[] Datos, int[] i, boolean grid,int opcion,boolean split,String path,
boolean axis,int[] range,boolean trid){
        PlotStyle styleDeleted = new PlotStyle();
        DataSetPlot Plotdata[] = new DataSetPlot[3];
        switch(opcion){
            case 0:
                styleDeleted.setStyle(Style.LINES);
            break;
            case 1:
                styleDeleted.setStyle(Style.POINTS);
                break;
            case 2:
                styleDeleted.setStyle(Style.LINESPOINTS);
                break;}
        for(int e=0; e<i.length;e++){
        Plotdata[e] = new DataSetPlot(Datos[i[e]].Datos);
        Plotdata[e].setPlotStyle(styleDeleted);
        Plotdata[e].setTitle(Datos[i[e]].nombre);}
        JavaPlot p = new JavaPlot(true);
        ImageTerminal png = new ImageTerminal();
        File file = new File(path);
        if(split==true){
         p.setTerminal(png);
        for(int e=0; e<i.length;e++){
         p.addPlot(Plotdata[e]);
         if(e!=i.length-1){
         p.newGraph3D();}}
        }else{
        p.setTerminal(png);
        for(int e=0; e<i.length;e++){
        p.addPlot(Plotdata[e]);}
        if(axis==true&&trid==true){
            p.getAxis("x").setBoundaries(range[0],range[1]);
            p.getAxis("y").setBoundaries(range[2],range[3]);
            p.getAxis("z").setBoundaries(range[4],range[5]);}}
         if(grid==true){
         p.set("grid", "back ls 12");}
        p.plot();
        try {
        ImageIO.write(png.getImage(), "png", file);
        } catch (IOException ex) {
        System.err.print(ex);}}

public static void ploteo2DEps(Datos[] Datos, int i, boolean grid,int opcion,String path, boolean axis,int[] range){
        PostscriptTerminal eps = new PostscriptTerminal(path);
        PlotStyle styleDeleted = new PlotStyle();
        switch(opcion){
            case 0:
                styleDeleted.setStyle(Style.LINES);
            break;
            case 1:
                styleDeleted.setStyle(Style.POINTS);
                break;
            case 2:
```

```
            styleDeleted.setStyle(Style.LINESPOINTS);
            break;}
        styleDeleted.setLineType(NamedPlotColor.RED);
        DataSetPlot setDeleted = new DataSetPlot(Datos[i].Datos);
        setDeleted.setPlotStyle(styleDeleted);
        JavaPlot p = new JavaPlot();
        p.setTitle(Datos[i].nombre);
        p.getAxis("x").setLabel(Datos[i].labels[0]);
        p.getAxis("y").setLabel(Datos[i].labels[1]);
        if(grid==true){
         p.set("grid", "back ls 12");}
        if(axis==true){
            p.getAxis("x").setBoundaries(range[0],range[1]);
            p.getAxis("y").setBoundaries(range[2],range[3]);}
        p.setTerminal(eps);
        p.addPlot(setDeleted);
        p.plot();}

    public static void ploteo3DEps(Datos[] Datos, int i, boolean grid,int opcion,String path, boolean axis,
    int[] range,boolean trid){
        PostscriptTerminal eps = new PostscriptTerminal(path);
        PlotStyle styleDeleted = new PlotStyle();
        switch(opcion){
            case 0:
                styleDeleted.setStyle(Style.LINES);
            break;
            case 1:
                styleDeleted.setStyle(Style.POINTS);
                break;
            case 2:
                styleDeleted.setStyle(Style.LINESPOINTS);
                break;}
        styleDeleted.setLineType(NamedPlotColor.RED);
        DataSetPlot setDeleted = new DataSetPlot(Datos[i].Datos);
        setDeleted.setPlotStyle(styleDeleted);
        JavaPlot p = new JavaPlot(true);
        p.setTitle(Datos[i].nombre);
        p.getAxis("x").setLabel(Datos[i].labels[0]);
        p.getAxis("y").setLabel(Datos[i].labels[1]);
        if(grid==true){
         p.set("grid", "back ls 12");}
        if(axis==true&&trid==true){
            p.getAxis("x").setBoundaries(range[0],range[1]);
            p.getAxis("y").setBoundaries(range[2],range[3]);
            p.getAxis("z").setBoundaries(range[4],range[5]);}
        p.setTerminal(eps);
        p.addPlot(setDeleted);
        p.plot();}
    public static void ploteo2DMulEps(Datos[] Datos, int[] i, boolean grid,int opcion,boolean split,String path,
    boolean axis,int[] range){
        PostscriptTerminal eps = new PostscriptTerminal(path);
        PlotStyle styleDeleted = new PlotStyle();
        DataSetPlot Plotdata[] = new DataSetPlot[3];
        switch(opcion){
            case 0:
                styleDeleted.setStyle(Style.LINES);
            break;
            case 1:
                styleDeleted.setStyle(Style.POINTS);
                break;
            case 2:
                styleDeleted.setStyle(Style.LINESPOINTS);
                break;}
        for(int e=0; e<i.length;e++){
        Plotdata[e] = new DataSetPlot(Datos[i[e]].Datos);
        Plotdata[e].setPlotStyle(styleDeleted);
        Plotdata[e].setTitle(Datos[i[e]].nombre);}
        JavaPlot p = new JavaPlot();
        if(split==true){
         p.setTerminal(eps);
        for(int e=0; e<i.length;e++){
         p.addPlot(Plotdata[e]);
         if(e!=i.length-1){
         p.newGraph();}}
```

```
    }else{
    p.setTerminal(eps);
    for(int e=0; e<i.length;e++){
    p.addPlot(Plotdata[e]);}
    if(axis==true){
        p.getAxis("x").setBoundaries(range[0],range[1]);
        p.getAxis("y").setBoundaries(range[2],range[3]);}}
     if(grid==true){
     p.set("grid", "back ls 12");}
    p.plot();}

public static void ploteo3DMulEps(Datos[] Datos, int[] i, boolean grid,int opcion,boolean split,String path,
boolean axis,int[] range,boolean trid){
    PostscriptTerminal eps = new PostscriptTerminal(path);
    PlotStyle styleDeleted = new PlotStyle();
    DataSetPlot Plotdata[] = new DataSetPlot[3];
    switch(opcion){
        case 0:
            styleDeleted.setStyle(Style.LINES);
        break;
        case 1:
            styleDeleted.setStyle(Style.POINTS);
            break;
        case 2:
            styleDeleted.setStyle(Style.LINESPOINTS);
            break;}
    for(int e=0; e<i.length;e++){
    Plotdata[e] = new DataSetPlot(Datos[i[e]].Datos);
    Plotdata[e].setPlotStyle(styleDeleted);
    Plotdata[e].setTitle(Datos[i[e]].nombre);}
    JavaPlot p = new JavaPlot(true);
    if(split==true){
    p.setTerminal(eps);
    for(int e=0; e<i.length;e++){
     p.addPlot(Plotdata[e]);
     if(e!=i.length-1){
     p.newGraph3D();}}
    }else{
    p.setTerminal(eps);
    for(int e=0; e<i.length;e++){
    p.addPlot(Plotdata[e]);}
    if(axis==true&&trid==true){
        p.getAxis("x").setBoundaries(range[0],range[1]);
        p.getAxis("y").setBoundaries(range[2],range[3]);
        p.getAxis("z").setBoundaries(range[4],range[5]);}}
     if(grid==true){
     p.set("grid", "back ls 12");}
    p.plot();}}
```

Anexo VI: Código PluginFunction.Java.

```
public interface PluginFunction {

        // let the application pass in a parameter
        public void setParameter (double [][]param);

        // retrieve a result from the plugin
        public int getResult();

        // return the name of this plugin
        public String getPluginName();

        // can be called to determine whether the plugin
        // aborted execution due to an error condition
        public boolean hasError();
}
```