INSTITUTO TECNOLÓGICO DE CIUDAD MADERO

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

OPCIÓN I: TESIS PROFESIONAL

“Análisis de Algoritmos Metaheurísticos Vía Diagnóstico Estadístico”

PARA OBTENER EL GRADO DE:
Maestra en Ciencias en Ciencias de la Computación

PRESENTA:
I.S.C. Mercedes Pérez Villafuerte
G04070504

DIRECTOR DE TESIS:
Dra. Laura Cruz Reyes

CO-DIRECTOR DE TESIS:
Dra. Claudia Guadalupe Gómez Santillán

Ciudad Madero, Tamaulipas, México Junio 2014

"2014, Año de Octavio Paz"

Cd. Madero, Tamps. a 08 de Mayo de 2014

OFICIO No.: US.088/14
AREA: DIVISIÓN DE ESTUDIOS
DE POSGRADO E INVESTIGACIÓN
ASUNTO: ELABORACIÓN DE CERTIFICADO

C. ING. MERCEDES PÉREZ VILLAFUERTE
NO. DE CONTROL G04070504
P R E S E N T E

Me es grato comunicarle que después de la revisión realizada por el Jurado designado para su examen de grado de Maestría en Ciencias en Ciencias de la Computación, el cual está integrado por los siguientes catedráticos:

PRESIDENTE :	DR. RODOLFO ABRAHAM PAZOS RANGEL
SECRETARIO :	DR. HÉCTOR JOAQUÍN FRAIRE HUACUJA
VOCAL :	DRA. LAURA CRUZ REYES
SUPLENTE :	DR. JOSÉ ANTONIO MARTÍNEZ FLORES
DIRECTORA DE TESIS :	DRA. LAURA CRUZ REYES
CO-DIRECTORA :	DRA. CLAUDIA GUADALUPE GÓMEZ SANTILLÁN

Se acordó autorizar la impresión de su tesis titulada:

"ANÁLISIS DE ALGORITMOS METAHEURÍSTICOS VÍA DIAGNÓSTICO ESTADÍSTICO"

Es muy satisfactorio para la División de Estudios de Posgrado e Investigación compartir con Usted el logro de esta meta.

Espero que continúe con éxito su desarrollo profesional y dedique su experiencia e inteligencia en beneficio de México.

A T E N T A M E N T E
"Por mi patria y por mi bien"®

M. P. MARIA YOLANDA CHÁVEZ CINCO
JEFA DE LA DIVISIÓN

S.E.P.
DIVISION DE ESTUDIOS
DE POSGRADO E
INVESTIGACION
I T C M

c.c.p.- Minuta
Archivo
MYCHC.'NLCO.'jar

DECLARACIÓN DE ORIGINALIDAD

Declaro que este documento de tesis es un trabajo original, donde en las citas incluidas y referencias, se indica explícitamente los datos de las publicaciones así como sus autores, incluidos en las referencias bibliográficas.

Acepto la total responsabilidad en caso de infringir con las leyes de derechos de terceros, de cualquier reclamación relacionada con los derechos de propiedad intelectual, que se puedan derivar en este documento de tesis, exonerando de la responsabilidad a mi director de tesis, así como al Instituto Tecnológico de Ciudad Madero.

Junio de 2014, Cd. Madero, Tamps.

Ing. Mercedes Pérez Villafuerte

DEDICATORIA

Dedico este trabajo a mis padres, Roberto Pérez Villanueva y Mercedes Villafuerte Hilerio, a mi padre por ser mi ejemplo a seguir, a mi madre por ser un ejemplo de lucha y ser mi mejor amiga.

A mi hermana Guadalupe Pérez Villafuerte, por ser mi amiga y darle ese toque de alegría a nuestro hogar.

A mi esposo César Medina Trejo y mi hijo César Roberto Medina Pérez, por ser mi alegría y mi motor día a día, los amo.

AGRADECIMIENTOS

Reciban un profundo agradecimiento los integrantes del comité tutorial de esta tesis: Dra. Laura Cruz Reyes, Dra. Claudia Guadalupe Gómez Santillán, Dr. Héctor Fraire Huajuca, Dr. Rodolfo Pazos Rangel, y Dr. José Antonio Martínez Flores.

Agradezco a las instituciones que dieron todas las facilidades para que este trabajo se llevara a cabo: el Consejo Nacional de Ciencia y Tecnología (Conacyt), la Dirección General de Educación Superior Tecnológica (DGEST) y el Instituto Tecnológico de Ciudad Madero.

Agradezco a mis compañeros que me permitieron convivir y aprender con ellos.

Así mismo, agradezco a los colaboradores que aportaron información y experimentaciones para este trabajo.

Resumen

Muchos problemas del mundo real son NP-Duros, para estos problemas se cree que no existen algoritmos exactos de solución cuyo tiempo de ejecución no aumente exponencialmente con el tamaño del problema. Hay dos formas de atacar a los problemas NP Duros. La primera es usando métodos exactos que requieren tiempo computacional exponencial. La segunda, son los que se usan en la práctica. Para los problemas de gran tamaño se emplean los métodos no exactos llamados metaheurísticos, los cuales producen soluciones en un tiempo razonable pero no se puede garantizar que encuentren los resultados óptimos.

Cuando se resuelven problemas complejos, el desempeño de algoritmos metaheurísticos depende de muchos factores, por lo que un mal diseño puede conducir a un desempeño pobre. No existen reglas guías que nos indiquen como diseñar apropiadamente los metaheurísticos. Es por esto que los diseñadores de estos métodos se toman demasiado tiempo para ajustarlos, mucho más aún que implementar en sí el propio metaheurístico. Este trabajo se hace manualmente a base de prueba y error consumiendo demasiado tiempo.

Este proyecto pretende aportar una herramienta de análisis mediante un diagnóstico visual del desempeño del metaheurístico; se busca disminuir el tiempo que se toman los desarrolladores en hacer ajustes. En particular, se propone aplicar la herramienta en el análisis de factores que han dificultado la solución de instancias retadoras del problema de empacado de objetos en contenedores (Bin Packing Problem, BPP). Además se añaden pruebas estadísticas no-paramétricas que son de uso en la comunidad científica para comprobar la significancia de la eficiencia de algoritmos.

Summary

Many real-world problems are NP-Hard, for these problems is believed that there are no exact solution algorithms whose running time does not increase exponentially with the problem size. There are two ways to attack NP Hard problems. The first is using exact methods which require exponential computational time. The second are those used in practice. For large problems are used accurate methods called metaheuristics, which produce solutions in a reasonable time but can not guarantee to find the optimal results.

When solving complex problems, metaheuristics algorithms performance depends on many factors, so a bad design can lead to poor performance. There are no guidelines to tell us how to design metaheuristics properly. This is why the designers of these methods take too long to adjust, much more so than implementing metaheuristic itself. This work is done manually through trial and error spending too much time.

This project aims to provide an analysis tool using a visual diagnosis metaheuristic performance, the goal is to reduce the time it takes developers to make adjustments. In particular, it is proposed to apply the tool in the analysis of factors that have hindered the solution of the challenging Bin Packing Problem (BPP) instances. Also is added nonparametric statistical tests that are commonly used in the scientific community to test the significance of the efficiency of algorithms.

Tabla de contenido

Índice de Figuras

Índice de Tablas

CAPÍTULO 1

Introducción

Muchos problemas del mundo real son NP-Duros, para estos problemas se cree que no existen algoritmos exactos de solución cuyo tiempo de ejecución no aumente exponencialmente con el tamaño del problema. Hay dos formas de atacar a los problemas NP Duros. La primera es usando métodos exactos que requieren tiempo computacional exponencial. La segunda, son los que se usan en la práctica. Para los problemas de gran tamaño se emplean los métodos no exactos llamados metaheurísticos, los cuales producen soluciones en un tiempo razonable pero no se puede garantizar que encuentren los resultados óptimos.

En el problema clásico *Bin Packing* (BPP) se busca el menor número de contenedores para el almacenamiento de un conjunto de objetos dado (Goldberg, 2002). Este problema es NP-Duro, y para su solución se cuenta con un Algoritmo Híbrido basado en búsqueda Tabú y un Algoritmo Genético de agrupación [Alvim 2004, Nieto 2007, Quiroz 2009]. A pesar que ambos algoritmos son de alto rendimiento, aún existen instancias de BPP que no han sido resueltas. El rendimiento de estos algoritmos son comparados con las soluciones conocidas para un conjunto de instancias.

Cuando se resuelven problemas complejos como BPP, el desempeño de algoritmos metaheurísticos depende de muchos factores, por lo que un mal diseño puede conducir a un desempeño pobre. No existen reglas guías que nos indiquen como diseñar apropiadamente los metaheurísticos. Es por esto que los diseñadores de estos métodos se toman demasiado tiempo para ajustarlos, mucho más aún que implementar en sí el propio metaheurístico. Este trabajo se hace manualmente a base de prueba y error consumiendo demasiado tiempo. Si existiera una forma de hacer estos ajustes de manera más rápida se aprovecharía mucho más el tiempo del diseñador.

En trabajos previos del grupo al que está adscrito el proponente, se han utilizado métodos de visualización para identificar áreas de mejora de algoritmos metaheurísticos, pero éstos son de alcance limitado. Entre sus limitaciones destacan dos: no satisfacen todos los requerimientos de análisis y operan de manera independiente. Este proyecto pretende aportar una herramienta de análisis mediante un diagnóstico visual del desempeño del metaheurístico; se busca disminuir el tiempo que se toman los desarrolladores en hacer ajustes. En particular, se propone aplicar la herramienta en el análisis de factores que han dificultado la solución de instancias retadoras de BPP. Además se añaden pruebas estadísticas no-paramétricas que son de uso en la comunidad científica para comprobar significancia de la eficiencia de algoritmos.

Por lo antes expuesto, el presente proyecto quiere brindar herramientas de diagnóstico visual integradas en una sola arquitectura que permita a otros investigadores extender las características de esta herramienta para automatizar el desarrollo de metaheurísticos, complementando éstas actividades con el apoyo de pruebas estadísticas que son incorporadas en ésta herramienta.

1.1. Antecedentes del Proyecto

El grupo de investigación, del cual se derivó este proyecto, tiene dos objetivos de largo plazo: el primero es contribuir al entendimiento del funcionamiento de algoritmos de solución aproximada y el segundo es desarrollar algoritmos basados en híper-heurísticas que proporcionen mayor cobertura en la solución de problemas (por ejemplo, con capacidad para resolver diferentes tipos de problemas y diferentes tipos de instancias).

1.2 Descripción del Problema

Un metaheurístico es un algoritmo genérico que puede ser usado para encontrar soluciones de alta calidad para los problemas de optimización combinatoria. Para llegar a un algoritmo funcional, un metaheurístico necesita ser configurada: típicamente algunos módulos necesitan ser desarrollados y algunos parámetros necesitan ser afinados. A estos dos problemas se le llama ajuste “estructural” y “paramétrica” respectivamente, a la combinación

de estos dos problemas se les llama “afinación”. La afinación es crucial para la optimización de un metaheurístico, ya que solo así se pueden obtener resultados buenos o incluso óptimos, de otra manera, se obtienen resultados pobres o a lo mucho resultados promedio.

En muchos casos, el diseño de metaheurísticos se hace de manera manual y sin sustento teórico, las pruebas se hacen a prueba y error y esta actividad toma demasiado en el proceso de diseño y desarrollo de los algoritmos metaheurísicos. El reto de esta investigación es contribuir a la automatización del análisis experimental de algoritmos mediante una herramienta de diagnóstico visual. Particularmente, con la incorporación de métricas de pruebas estadísticas que permitan contrastar diferentes configuraciones

1.3 Justificación y Beneficios del Proyecto

A pesar de que el problema de Bin Packing es un clásico entre los problemas complejos, no ha dejado de ser investigado por muchos, debido a las muchas aplicaciones que se pueden hacer en el mundo real en muchas área: ingeniería, comercio y en el ámbito industrial. Sin embargo, debido a los grandes volúmenes de datos que se manejan en el ámbito industrial, la tarea de encontrar soluciones se hace casi imposible, es por esto que se vuelve necesaria la aplicación de algoritmos que permitan el tratamiento de tales cantidades de información y que sean capaces de resolver los problemas presentados.

A la fecha hay muchos metaheurísticos que se han desarrollado y muchas son ajustadas y realizadas a prueba y error que toman demasiado tiempo; el gran reto en la algoritmia metaheurística es establecer fundamentos sólidos y no los hay, solo existen construcciones artesanales, se desea que los diseños de los algoritmos se hagan con bases teóricas. Este proyecto no va solucionar este problema de manera inmediata, solo se va a contribuir a su solución en el largo plazo, mediante una herramienta que facilite la comprensión del comportamiento algorítmico.

1.4 Objetivos

Proporcionar una herramienta que permita hacer un análisis y diagnóstico soportado en caracterizaciones del proceso de optimización mediante visualización para el diseño de metaheurísticos, complementando éstas actividades con el apoyo de pruebas estadísticas que son incorporadas en ésta herramienta.

1.5 Alcances y Limitaciones del Proyecto

El proyecto está enfocado en desarrollar una herramienta que sea útil para el análisis experimental del desempeño de heurísticos y metaheurísticos mediante el diagnóstico visual enfocados solamente al problema de bin packing unidimensional, dejando como trabajo futuro el análisis de híper-heurísticos.

Este proyecto no contempla la composición de metaheurísticos como parte de las utilerías de la herramienta, tampoco la generación de código.

1.6 Descripción de la complejidad del problema

Muchos de los problemas de combinatorios son computacionalmente intratables y a menudo se satisfacen con algoritmos metaheurísticos. Pero dada la naturaleza heurística de estos métodos, existen dos consideraciones importantes al diseñar un metaheurístico.

- Elegir las heurísticas a emplear.
- Seleccionar los parámetros apropiados para dirigir las heurísticas.

Este problema de diseñar apropiadamente metaheurísticos para los problemas combinatorios se llama el problema de ajuste de metaheurísticos. Y las anécdotas que se han tenido al

diseñar metaheurísticos sugieren que se toma un mayor esfuerzo al afinarlos, es decir, el 90% del tiempo de diseño y prueba puede gastarse afinado y ajustando al algoritmo metaheurístico.

Aunado a lo anterior, el problema que se quiere resolver es el problema de empacado en contenedores (Bin Packing Problem), el cual se ha demostrado que es NP-Duro (Álvarez, 2006) (Ver anexo A).

CAPÍTULO 2

Marco Teórico

2.1 Métodos de Solución para Problemas NP-Duros

En esta sección se describen tres tipos de algoritmos de solución aproximada: heurísticos, metaheurísticos e híper-heurísticos. Los algoritmos híper-heurísticos son de reciente creación y en su composición participan los dos primeros.

2.1.1 Problemas de Optimización Combinatoria

Muchos de los problemas combinatorios son computacionalmente intratables, en el sentido de que requieren una cantidad alta de recursos principalmente de tiempo del procesador. Uno de estos problemas es el empacado de objetos en contenedores (Bin Packing Problem, BPP), y es uno de los casos de estudio en esta tesis. Este problema es un problema combinatorio NP-Duro, en el que un conjunto de objetos de diferentes volúmenes deben ser empacados en un número finito de contenedores de capacidad limitada de manera que el número de contenedores sea minimizado.

Los problemas NP-Duros a menudo se satisfacen con algoritmos metaheurísticos. Pero dada la naturaleza heurística de estos métodos, existen dos consideraciones importantes al diseñar un metaheurístico.

- Elegir las heurísticas a emplear.
- Seleccionar los parámetros apropiados para dirigir las heurísticas.

Este problema de diseñar apropiadamente metaheurísticos para los problemas combinatorios se llama el problema de ajuste de metaheurísticos. Experimentos preliminares sugieren que se toma un mayor esfuerzo al afinarlos, es decir, el 90% del tiempo de diseño y prueba puede gastarse ajustando el algoritmo metaheurístico.

Para evaluar el desempeño de los métodos heurísticos diseñados, existen compendios de instancias para diferentes problemas de optimización combinatoria. Cabe mencionar que para algunos problemas es elevado el número de instancias disponibles para la comunidad científica, pero para otros se carece de ellas. Además, para un determinado problema de optimización, existen diferentes formatos de estructuración del contenido de la instancia, es decir, no existe un estándar para la representación de instancias de optimización, resultados y opciones de solucionadores (Fourer R., 2008).

2.1.2 Heurísticos

Los métodos heurísticos son procedimientos basados en el sentido común, que ofrecen buenas soluciones a problemas difíciles, de un modo fácil y rápido (Díaz, y otros, 1996). Son varios los factores que pueden motivar a utilizar alguna heurística en la solución de problemas del tipo combinatorio, entre los cuales están:

- No existe un método exacto de resolución o este requiere el uso de recursos computacionales de una manera inaceptable.
- No se requiere una solución óptima. Si el beneficio de encontrar una solución óptima no representa una diferencia importante respecto a una solución sub-óptima.
- Cuando los datos son poco fiables.
- Existen limitaciones de recursos computacionales.
- Como pre-procesamiento en algún otro algoritmo.

Para BPP un algoritmo heurístico clásico es el FFD (First Fist Decreasing), el cual ordena descendentemente los objetos y los acomoda, en ese orden, en el contenedor más lleno que lo pueda contener. Esta estrategia voraz forma parte de la siguiente clasificación (Duarte, y otros, 2007):

Métodos constructivos: Procedimientos que son capaces de construir una solución a un problema dado. La forma de construir la solución depende fuertemente de la estrategia seguida. Las estrategias más comunes son:

Estrategia voraz: Partiendo de una semilla, se va construyendo paso a paso una solución factible. En cada paso se añade un elemento constituyente de dicha solución, que se caracteriza por ser el que produce una mejora más elevada en la solución parcial para ese paso concreto.

Estrategia de descomposición: Se divide sistemáticamente el problema en subproblemas más pequeños. Este proceso se repite (generalmente de forma recursiva) hasta que se tenga un tamaño de problema en el que la solución a dicho subproblema es trivial. Después el algoritmo combina las soluciones obtenidas hasta que se tenga la solución al problema original.

Métodos de reducción: Identifican características que contienen las soluciones buenas conocidas y se asume que la solución óptima también las tendrá. De esta forma se puede reducir drásticamente el espacio de búsqueda.

Métodos de manipulación del modelo: Consisten en simplificar el modelo del problema original para obtener una solución al problema simplificado. A partir de esta solución aproximada, se extrapola la solución al problema original.

Métodos de búsqueda: Parten de una solución factible dada y a partir de ella intentan mejorarla.

Estrategia de búsqueda local 1: Parte de una solución factible que la mejora progresivamente. Para ello examina su vecindad y selecciona el primer movimiento que produce una mejora en la solución actual (first improvement)

Estrategia de búsqueda local 2: Parte de una solución factible que la mejora progresivamente. Para ello examina su vecindad y todos los posibles movimientos seleccionando el mejor movimiento de todos los posibles, es decir aquel que produzca un incremento (en el caso de maximización) más elevado en la función objetivo (best improvement).

Estrategia aleatorizada: Para una solución factible dada y una vecindad asociada a esa solución, se seleccionan aleatoriamente soluciones vecinas de esa vecindad.

2.1.3 Metaheurísticos

El término metaheurístico fue introducido por primera vez por Fred Glover (Glover, 1986), con este término quería definir un "procedimiento maestro de alto nivel que guía y modifica otras heurísticas para explorar soluciones más allá de la simple optimalidad local".

El término metaheurístico se ha usado como referencia para una familia de estrategias de búsqueda bien conocidas. Las cualidades esenciales de cada estrategia están encaminadas a descubrir mejores soluciones en el espacio de búsqueda mediante un enfoque ajustado en buenas soluciones y mejorándolas (intensificación), y a encaminar la exploración del espacio de soluciones mediante un enfoque amplio de la búsqueda de nuevas áreas (diversificación), estas dos cualidades son complementarias y necesarias.

Una búsqueda basada puramente en la intensificación no permite aceptar malas soluciones y por lo tanto no puede escapar de un óptimo local y por otro lado, una búsqueda basada puramente en diversificación no tiene control de calidad por el cual se rechazan malas soluciones y se alcanzan buenos resultados. (O'Brien, 2008).

A continuación se muestra una de las formas que más comúnmente se ha utilizado para clasificar a los metaheurísticos (Blum & Roli, 2003), sin embargo hay muchas otras formas más de clasificarlas.

Atendiendo a la Inspiración:

Natural: algoritmos que se basan en un símil real, ya sea biológico, social, cultural, entre otros. Entre los más populares están los algoritmos genéticos.

El Algoritmo Genético es un heurístico de búsqueda que simula el proceso de evolución natural, que mezcla elementos de herencia, mutación, selección y cruza.

Sin inspiración: algoritmos que se obtienen directamente de sus propiedades matemáticas.

Atendiendo al número de soluciones:

Poblacionales: buscan el óptimo de un problema a través de un conjunto de soluciones.

Trayectoriales: trabajan exclusivamente con una solución que mejoran iterativamente.

Atendiendo a la función objetivo:

Estáticas: no hacen ninguna modificación sobre la función objetivo del problema.

Dinámicas: modifican la función objetivo durante la búsqueda.

Atendiendo a la vecindad:

Una vecindad: durante la búsqueda utilizan exclusivamente una estructura de vecindad.

Varias vecindades: durante la búsqueda modifican la estructura de la vecindad.

Atendiendo al uso de memoria:

Sin memoria: se basan exclusivamente en el estado anterior.

Con memoria: utilizan una estructura de memoria para recordar la historia pasada.

2.1.4 Híper-heurísticos

El término híper-heurístico denota un método que opera en un nivel superior de abstracción y puede ser pensado como un metaheurístico que es capaz de elegir inteligentemente una posible heurística para ser aplicada en cualquier tiempo (Kendall, y otros, 2005).

El término en si fue acuñado por Cowling (Cowling, y otros, 2000) el cual menciona "Los híper-heurísticos administran la elección de cual heurístico de nivel inferior debe ser aplicado en cualquier tiempo dado, dependiendo de las características de los heurísticos y la región del espacio de solución que está actualmente bajo exploración". En otras palabras es un heurístico que elige entre heurísticos operando a un nivel de abstracción por encima de los metaheurísticos y fue propuesto como un enfoque para incrementar el nivel de generalidad al cual los sistemas de optimización pueden operar.

Los híper-heurísticos son presentados como una alternativa para los metaheurísticos que son mayormente desarrollados para un problema en particular. Se espera que los híper-heurísticos puedan ser desarrollados y empleados por programadores no especializados con poca o experiencia nula en el dominio del problema. (Özcan, y otros, 2008)

2.2 Proceso de Optimización

2.2.1 Caracterización del Proceso de Optimización.

Para comprender el funcionamiento de un algoritmo frente a un problema, se debe realizar un estudio completo del proceso de optimización. El proceso de optimización se puede entender como la acción de resolver un problema de optimización (entrada) mediante un algoritmo (proceso), obteniendo una solución final (salida). La entrada es una instancia o caso particular del problema de optimización que se quiere resolver, compuesto por un conjunto de parámetros específicos que lo definen. El proceso incluye el conjunto de estrategias utilizadas para solucionar el problema. La salida proporciona la solución del problema, ya sea óptima o una aproximada. Este proceso y una posible caracterización de sus componentes es mostrado en la Figura 2.1

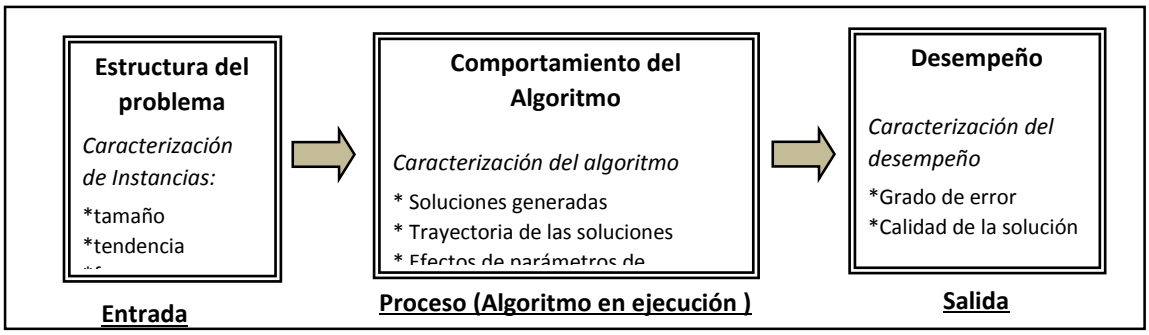

Figura 2.1- Proceso de optimización de un problema y su caracterización.

La caracterización del problema y del algoritmo es una parte esencial en el análisis del desempeño de los algoritmos, y permite identificar cuáles son las características (indicadores) que los describen adecuadamente. Con el modelo causal se provee una representación formal de las relaciones existentes entre los indicadores de la estructura del problema, y el comportamiento y desempeño del algoritmo.

El desempeño de un algoritmo puede ser afectado por la naturaleza del problema, es por esto que un algoritmo se comporta mejor con un determinado conjunto de instancias de un problema dado. En la Figura 2.2 se puede notar que el desempeño del algoritmo, depende en cierto grado de la trayectoria que sigue el algoritmo, y ésta a su vez, se relaciona con la naturaleza del problema que resuelve.

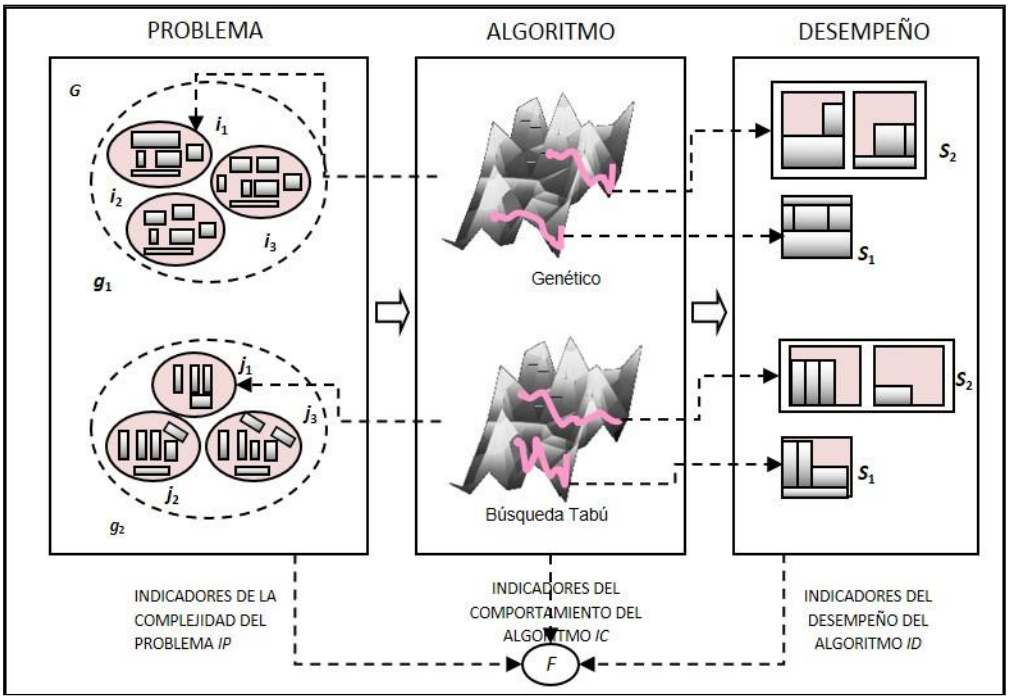

Figura 2.2 - Función de Relación de desempeño F.

2.2.3 Métricas de Desempeño para BPP

2.2.3.1 Caracterización del problema

La estructura de una instancia del problema de empacado de objetos es una característica importante para predecir el comportamiento que tendrá un algoritmo metaheurístico al momento de solucionarla. El tamaño de la instancia, la tendencia central de sus pesos y la forma en que se distribuyen, son indicadores que permiten conocer, de antemano, el posible grado de dificultad que la instancia puede tener para el algoritmo de solución.

Índices de Caracterización

La información descriptiva de cada instancia del problema se caracteriza por medio de índices que utilizan la información de los parámetros del problema de dicha instancia. A

continuación en las Tablas 2.1, 2.2 y 2.3 se muestran respectivamente los índices propuestos en los trabajos de Cruz, Álvarez y Quiroz (Cruz 2004, Álvarez 2006, Quiroz 2009) los cuales, según Quiroz son los que aportan una mayor cantidad de información sobre el problema de estudio.

Tabla 2.1- Índices propuestos por Cruz (Cruz, 2004)

Expresión	Descripción	Identificador de expresión
$p = \frac{n}{nmax}$	*p* es el índice del tamaño del caso, donde: *n* = número de objetos *nmax*=el tamaño máximo solucionado.	(2.1)
$t = \frac{\sum_{i=1}^{n} s_i / n}{c}$	*t* es el índice de capacidad ocupada por un objeto promedio, donde s_i = tamaño del objeto i, *c* = capacidad del contenedor.	(2.2)
$d = \sqrt{\frac{\sum_{i=1}^{n}\left[t - \left(\frac{s_i}{c}\right)\right]^2}{n}}$	El índice de dispersión *d* expresa el grado de la dispersión del cociente del tamaño de los objetos entre el tamaño del contenedor.	(2.3)
$f = \frac{\sum_{i=1}^{n} factor(c, s_i)}{n}$	El índice de factores *f* expresa la proporción de objetos cuyo tamaño s_i es factor de la capacidad del contenedor, donde $factor(c, s_i) = \begin{cases} 1 & \text{si } (c \bmod s_i) = 0 \\ 0 & \text{en caso contrario} \end{cases}$	(2.4)
$b = \begin{cases} 1 & \text{si } \ c \geq \sum_{i=1}^{n} s_i \\ \frac{c}{\sum_{i=1}^{n} s_i} & \text{en caso contrario} \end{cases}$	El uso de contenedor *b* expresa la proporción del tamaño total que se puede asignar en un contenedor de capacidad *c*.	(2.5)

Tabla 2.2 - Índices propuestos por Álvarez (Álvarez, 2006)

Expresión	Descripción	Identificador de expresión
$cv = \frac{\sqrt{1/n \sum_{i=1}^{n} (s_i - \bar{s})^2}}{\lvert \bar{s} \rvert}$	*cv* es el coeficiente de variación, donde *n* = número de objetos $\bar{s}$ = media de los tamaños de los objetos	(2.6)
$curtosis = \left(\frac{1}{n}\right) \frac{\sum_{i=1}^{n} (s_i - \bar{s})^4}{de^4}$	*curtosis* es la curtosis de la forma de la distribución de los tamaños de los objetos, donde *de* = desviación estándar de s	(2.7)
$asimetria = \left(\frac{1}{n}\right) \frac{\sum_{i=1}^{n} (s_i - \bar{s})^3}{de^3}$	*asimetría* es el coeficiente de asimetría de la distribución de los tamaños de los objetos, es decir, el sesgo del conjunto de pesos.	(2.8)

Tabla 2.3 - Índices propuestos por Quiroz (Quiroz, 2009)

Expresión	Descripción	Identificador de expresión
$menor(s) = \frac{\min(s)}{c}$	$menor(s)$ es el tamaño del objeto más pequeño, en relación al tamaño del contenedor. *s* = tamaños de objetos, *c* = capacidad del contenedor.	(2.9)
$mayor(s) = \frac{\max(s)}{c}$	$mayor(s)$ es el tamaño del objeto más grande, en relación al tamaño del contenedor. *s* = tamaños de objetos, *c* = capacidad del contenedor.	(2.10)
$Rango = mayor(s) - menor(s)$	*Rango* es el intervalo en el que se distribuyen los pesos de los objetos.	(2.11)

$M = \frac{\sum_{i=1}^{m} r_i}{m}$	M es la multiplicidad y es el promedio del número de repeticiones de cada tamaño de objeto, donde r_i = número de objetos con tamaño rs_i rs guarda los diferentes tamaños de los objetos.	(2.12)
$uniformidad = 1 - \frac{\sum_{j=1}^{4} \left\| \left(\frac{n}{4} - \left\| B_j \right\| \right) \right\|}{n}$	(ver sección de índice de uniformidad)	(2.13)

Índice de uniformidad

Mide el grado de uniformidad de la distribución de los tamaños de los objetos, a partir de la división del rango de datos en cuatro partes y las diferencias entre el número de objetos esperado en cada subrango y el número real.

Sea

n = número de objetos

$S = \{ s_i \mid s_i < s_{i+1} \}, 1 \le i \le n$, arreglo de tamaños de objetos en orden creciente

$R = \max(S) - \min(S)$, amplitud del rango de tamaños de los objetos

$e = \frac{n}{4}$, número esperado de objetos en una cuarta parte del rango

$r_0 = 0$, parte inicial del rango

$r_1 = \min(S) + \frac{R}{4}$, primera división del rango

$r_2 = \min(S) + \frac{R}{2}$, segunda división del rango

$r_3 = \min(S) + \frac{3R}{4}$, tercera división del rango

$r_4 = \max(S)$, parte final del rango

$B_j \subset S \left| \bigcup_{j=1}^{4} B_j = S \right.$, una cuarta parte del rango de distribución de tamaños

Donde

$$B_j = \left\{ s \in S \left| r_{j-1} < s \le r_j \right. \right\} 1 \le j \le 4$$

El grado de uniformidad del conjunto de tamaños es:

$$uniformidad = 1 - \frac{\sum_{j=1}^{4} \left| \left(\frac{n}{4} - \left| B_j \right| \right) \right|}{n}$$

Si el valor de uniformidad es igual a uno, se considera que la distribución del conjunto de tamaños es uniforme y mientras más cercano a uno más uniforme.

2.2.3.2 Caracterización del Comportamiento Algorítmico

Los algoritmos metaheurísticos incluyen estrategias que se ajustan al problema que resuelven, la medición de funciones de caracterización está ligada directamente con las heurísticas utilizadas. En esta sección se describe el algoritmo genético estudiado propuesto por Quiroz (Quiroz, 2009), así como los índices propuestos para caracterizar el comportamiento del algoritmo durante ejecución y la trayectoria de búsqueda trazada por éste.

Algoritmo Genético para el Problema de Empacado de Objetos: GGA- BP

La estrategia metaheurística considerada como uno de los objetivos de estudio de este trabajo de investigación es un algoritmo genético híbrido de agrupación. Este algoritmo incorpora estrategias heurísticas y criterios que al aplicarse en la creación y evolución de individuos, de generación en generación, permiten obtener buenas soluciones en tiempos cortos. El Algoritmo 2. 1 describe el procedimiento general.

ALGORITMO GENÉTICO GGA-BP

```
1   Inicio
2   inicializar parámetros
3   para cada generación
4   para cada individuo
5               new_binsol=generar individuo
6               Calcula aptitud del individuo
7               si random[0-1) <= 0.95
8                       si fitness(new_binsol)>fitness(old_binsol)
9               old_binsol = new_binsol
10              fin si
11              si no
12              old_binsol = new_binsol
13              fin si
14          fin para
15              Seleccionar individuos de forma proporcional
16              Cruzamiento por Agrupación
17              Mutación por Agrupación
18              Busca la mejor solución
19  fin para
20  fin procedimiento
```

Algoritmo 2. 1 Procedimiento general de AG

El proceso consta de generar una población para lo cual se crea un conjunto de individuos en forma no determinista. Después de la primera generación, la probabilidad de

mantener al individuo de la generación previa es proporcional a su aptitud y a la generación de un número aleatorio (4-14). Dado un porcentaje y con base a la aptitud (15), se eligen mediante selección proporcional un conjunto de individuos para aplicarles los operadores evolutivos de cruzamiento y mutación (16-17), posteriormente se guarda la mejor solución de la población para que después de un cierto número de generaciones se cuente con el individuo más apto.

Los principales parámetros del algoritmo son:

POPULATION indica el número de individuos (soluciones) que formaran parte de la población y evolucionarán para generar mejores soluciones. El tamaño de población utilizado es de 50 individuos.

RUNS (corridas) es el número de veces que se ejecutará el programa, para cuestiones de cálculos estadísticos y comprobación de estabilidad. Se recomienda fijarlo en 30 corridas.

NUMBER_GENERATION es el número de generaciones que el algoritmo iterará generando individuos y aplicando operadores de cruza y muta. Se ha encontrado que 100 generaciones es suficiente para obtener una buena solución.

CONSTANTE POSITIVA K que expresa la concentración de un buen llenado de contenedor, este parámetro es utilizado al calcular la aptitud de un individuo, se recomienda k = 2.

PERCENTAGE_CROSS_OVER es el porcentaje de individuos de la población que se cruzarán entre ellos en cada generación, para producir nuevos individuos; cruzar el 30% de los individuos ha mostrado buenos resultados.

PORC_MUTATION es el porcentaje de individuos de la población que mutarán en cada generación, para producir nuevos individuos con pequeñas modificaciones, en la configuración actual 50% de los individuos mutan en cada generación.

Índices de Caracterización

Para observar y caracterizar el comportamiento del algoritmo durante ejecución Quiroz (Quiroz, 2009) propone los índices descritos en la Tabla2.4.

Tabla 2.4 - Índices propuestos para caracterizar el comportamiento algorítmico

Expresión	Descripción	Identificador de expresión
$p\_ind = \frac{\sum_{gen=1}^{n\_gen} n\_ind_{gen}}{n\_gen}$	p_ind es el promedio de individuos que nacen en cada generación, donde n_ind_{gen} = número de individuos aceptados en la generación gen	(2.13)
$p\_gind = \frac{\sum_{gen=1}^{n\_gen} n\_gind_{gen}}{n\_gen}$	p_gind es el promedio de individuos buenos que nacen en cada generación, donde n_gind_{gen} = número de individuos buenos aceptados en la generación gen	(2.14)
$p\_bind = \frac{\sum_{gen=1}^{n\_gen} n\_bind_{gen}}{n\_gen}$	p_bind es el promedio de individuos malos que nacen en cada generación, donde n_bind_{gen} = número de individuos malos aceptados en la generación gen	(2.15)
$de\_ind = \frac{\sum_{gen=1}^{n\_gen} de\_ind_{gen}}{n\_gen}$	de_ind es el promedio de la desviación estándar de los individuos en n_gen generaciones, donde de_ind_{gen} = es la desviación estándar de la población en la generación gen	(2.16)
$prom\_gen = \frac{\sum_{corr=1}^{n\_corr} b\_gen_{corr}}{n\_corr}$	$prom\_gen$ es la generación promedio en la que se encuentra	(2.17)

	la mejor solución del algoritmo en n_corr corridas, donde b_gen_{corr} = generación en la que se encontró la mejor solución en la corrida corr	
$\overline{fb} = \frac{\sum_{i=1}^{n\_gen} fb_i}{n\_gen},\ n > 0$	$\overline{fb}$ es la aptitud promedio de la mejor solución en n_gen generaciones, donde fb_i = aptitud del mejor individuo de la generación i	(2.18)
$\rho_1 = \frac{\sum_{i=1}^{n\_gen-1} \left(fb_i - \overline{fb}\right)\left(fb_{i+1} - \overline{fb}\right)}{\sum_{i=1}^{n\_gen} \left(bf_i - \overline{fb}\right)^2}$	ρ_1 es la función de autocorrelación de las aptitudes de los mejores individuos	(2.19)
$best\_metod = met_i \,\vert\, rep_i = \max(rep)$	$best\_metod$ es la estrategia del algoritmo que permitió obtener el mayor número de mejores soluciones, donde met_i = estrategia i rep_i =número de veces que la estrategia i se repite	(2.20)

2.2.3.3 Caracterización del Desempeño Final

El desempeño final del algoritmo se mide por tres tipos de información: calidad de la solución, la consistencia de los resultados y el tiempo que le llevó en encontrar la mejor solución, los cuales se describen a continuación.

El radio teórico mide la calidad de la solución, éste es la razón de la mejor solución encontrada por el algoritmo (Zenc) y la solución óptima (Zopt). La función aptitud del mejor individuo está definida por la Expresión (2.21) de la Tabla 2.5.

Tabla 2.5 - Índices propuestos para caracterizar el desempeño final

Expresión	Descripción	Identificador de expresión
$radio\_teorico = \frac{Zenc}{Zopt}$	$radio\_teorico$ es la razón de la solución óptima y la solución mejor encontrada por el algoritmo.	(2.21)
$f(x) = \frac{\sum_{j=1}^{numbin} \left(F_j / c \right)^2}{numbin}$	*f(x)* es el valor de la función aptitud para la solución *x*, donde F_j= suma de los tamaños de los objetos que se encuentran en el contenedor *j*. *numbin*=Número de contenedores que tiene la solución. *x*= {{numbin},{F_1, F_2, F_3, …, F_{numbin}}}	(2.22)
$deZ = \sqrt{\frac{\sum_{i=1}^{n\_corr} \left(Zenc_i - \bar{Z}enc \right)^2}{n\_corr}}$	deZ es la desviación estándar de las soluciones encontradas en *n_corr* corridas del algoritmo, donde $Zenc_i$ = solución encontrada en la corrida *i* $\bar{Z}enc$ =solución promedio encontrada en *n_corr* corridas	(2.23)
$prom\_time = \frac{\sum_{corr=1}^{n\_corr} time_{corr}}{n\_corr}$	$prom\_time$ es el tiempo promedio, en segundos, que el algoritmo tarda en encontrar la solución, donde $time_{corr}$ = tiempo utilizado por el algoritmo en la corrida *n_corr*	(2.24)

El coeficiente de engaño se define en la expresión 2.25. Donde *E* es el valor esperado de aptitud final y los valores máximo y mínimo de *F* son $F_{mín}$ y $F_{máx}$. Si $F_{máx} - F_{mín} = 0$ entonces el valor de esta métrica es 0.

$$coef\_eng = 1 - \frac{E - F_{mín}}{F_{máx} - F_{mín}} \tag{2.25}$$

Este coeficiente se interpreta de la siguiente manera: si el valor obtenido muestra tendencia a 1 significa que el problema es engañoso, mientras que un valor cercano a 0 indica que el problema es amigable o sencillo de resolver.

2.3 Análisis Experimental

Tradicionalmente el análisis de complejidad de un algoritmo se ha efectuado de manera teórica. La naturaleza de muchos problemas del mundo real ha llevado al desarrollo de algoritmos cuyo análisis teórico no es suficiente o no es posible de efectuar con las herramientas matemáticas disponibles.

En el análisis teórico se determina matemáticamente la cantidad de recursos requeridos por cada algoritmo como una función cuya variable independiente es el tamaño de la entrada. Este análisis es independiente de los recursos de cómputo, implementación y programador, facilitando la generalización del comportamiento algorítmico.

En el análisis empírico, las implementaciones de los algoritmos se ejecutan para diferentes entradas y se comparan en base a los recursos consumidos. Los resultados obtenidos son específicos de las entradas consideradas en el estudio.

El desempeño de un algoritmo basado en heurísticas es determinado por la eficiencia y la efectividad mostradas por éste al resolver diferentes instancias de un problema. La efectividad de un algoritmo se refiere a la calidad de la solución encontrada o a su

confiabilidad en la tarea de encontrar soluciones adecuadas y es altamente dependiente de la estructura del problema. La eficiencia por otra parte, caracteriza el comportamiento del algoritmo en tiempo de ejecución (por ejemplo el tiempo computacional y los requerimientos de memoria) y está muy relacionada con el conocimiento que se tenga del dominio y la complejidad del problema (Pérez, 2007).

A pesar de la importancia del análisis de desempeño experimental, éste no ha sido suficientemente explotado en el estudio de algoritmos metaheurísticos. Una de la razones que dificultan este estudio es que, generalmente, los algoritmos son considerados como cajas negras cuyo funcionamiento interno es desconocido. Barr (Barr R., 1995) y McGeoch (McGeoch, 1992) intentan romper con el paradigma de las cajas negras y sugieren la existencia de tres categorías principales de factores que afectan el desempeño algorítmico: problema, algoritmo y ambiente.

- *Factores del problema.* Son los factores que definen la instancia del problema a resolver: dimensión, distribución de los parámetros, estructura del espacio de soluciones, entre otros.
- *Factores del algoritmo.* Incluyen las estrategias heurísticas utilizadas (procesos de construcción de solución inicial y parámetros de búsqueda asociados), códigos de computadoras empleados, configuración de control interno del algoritmo y comportamiento en la ejecución, entre otros.
- *Factores del ambiente.* Estos factores se refieren al ambiente físico en el que serán ejecutados los algoritmos como los son el software (sistema operativo, compilador) y hardware (velocidad del procesador, memoria). Así como también aquellos relacionados con el programador (por ejemplo, pericia, habilidad de afinación y lógica).

2.4 Análisis Visual

Visualizar es hacer un modelo mental o una imagen mental de algo. La visualización es una actividad cognitiva propia de los humanos, es algo que las computadoras no hacen.

La visualización de la información es el estudio de representaciones de colecciones a gran escala de información. La visualización de la información se enfoca en la creación de enfoques para llevar la información abstracta en una forma intuitiva. Las representaciones visuales y las técnicas de interacción toman ventaja de la facilidad de llegar a la mente por medio de los ojos y permite a los usuarios ver, explorar y entender grandes cantidades de información a la vez. Para la visualización de información se vale cualquier sentido, no sólo la vista, aunque es el sentido que es capaz de aportar más datos a la mente por unidad de tiempo.

Existen muchas técnicas para crear representaciones de la información, a continuación se mencionan algunas:

Gráficas: son una representación de datos, generalmente numéricos, mediante líneas, superficies o símbolos, para ver la relación que esos datos guardan entre sí. También puede ser un conjunto de puntos, que se plasman en coordenadas cartesianas, y sirven para analizar el comportamiento de un proceso, o un conjunto de elementos o signos que permiten la interpretación de un fenómeno.

Grafos: es un conjunto de objetos llamados vértices o nodos unidos por enlaces llamados aristas o arcos, que permiten representar relaciones binarias entre elementos de un conjunto.

Los grafos permiten estudiar las interrelaciones entre unidades que interactúan unas con otras.

Diagramas: se utilizan generalmente para facilitar el entendimiento de largas cantidades de datos y la relación entre diferentes partes de los datos. Los diagramas pueden generalmente ser leídos más rápidamente que los datos en bruto de los que proceden. Se utilizan en una amplia variedad de campos.

CAPITULO 3

Estado del Arte

En esta revisión del estado del arte se analizan herramientas que tienen como apoyo la visualización de datos y resultados para encontrar áreas de mejora en algoritmos heurísticos. De igual manera se ha hecho una revisión de trabajos que hacen uso del análisis estadístico para encontrar información significativa que pueda ser de utilidad para la mejora de algoritmos. Cabe mencionar que la herramienta propuesta en esta tesis incorpora ambos enfoques, el análisis visual y el estadístico con la idea de que estas técnicas permitan encontrar áreas de mejora en el proceso de optimización.

3.1 Herramientas de Análisis de Algoritmos

3.1.1 GUIMOO

Guimoo (Graphical User Interface for Multi Objective Optimization) es un software libre dedicado al análisis de resultados en optimizaciones multiobjetivo, entre sus aportaciones disponibles se encuentran:

La visualización en línea de aproximaciones de frentes de Pareto, tal información puede ser usada por el experto para construir metaheurísticos más eficientes.

Algunas métricas para evaluación de desempeño cuantitativo y cualitativo (contribución, entropía, espaciado, tamaño del espacio dominado, etc.). Esta herramienta se encuentra disponible en (INRIA).

3.1.2 ParadisEO

ParadisEO (Liefooghe, y otros, 2009) es una arquitectura de caja blanca dedicada al diseño de metaheurísticos reusables, metaheurísticos híbridos, y metaheurísticos paralelos y

distribuidas. ParadisEO provee un amplio rango de utilidades incluyendo algoritmos evolutivos, búsquedas locales, mecanismos de hibridación, entre otros. ParadisEO separa los problemas que se tratan de resolver de aspectos conceptuales de los métodos de solución. Esta separación permite el reuso de código y facilita el diseño. Esta herramienta se encuentra disponible para descargar en (Europe, s.f.)

3.1.3 BPP Visualization Tool, Callan 2007

Este proyecto es un diseño de profesorado que se implementó para analizar y comparar visualmente la efectividad de diferentes algoritmos metaheurísticos para resolver dos problemas de optimización: El problema de empacado de objetos y el Problema de Asignación Generalizada. Los algoritmos de prueba usados para resolver estos problemas son: Algoritmo Genético, Recocido simulado y un Algoritmo Genético propuesto usando poblaciones factibles y no factibles. Esta herramienta no está disponible para uso público.

Las visualizaciones disponibles, se muestran en la Figura 3.1. Se visualiza la representación de la mejor solución encontrada para cada algoritmo, mediante contenedores. Otra visualización disponible es la gráfica del desempeño sobre el tiempo.

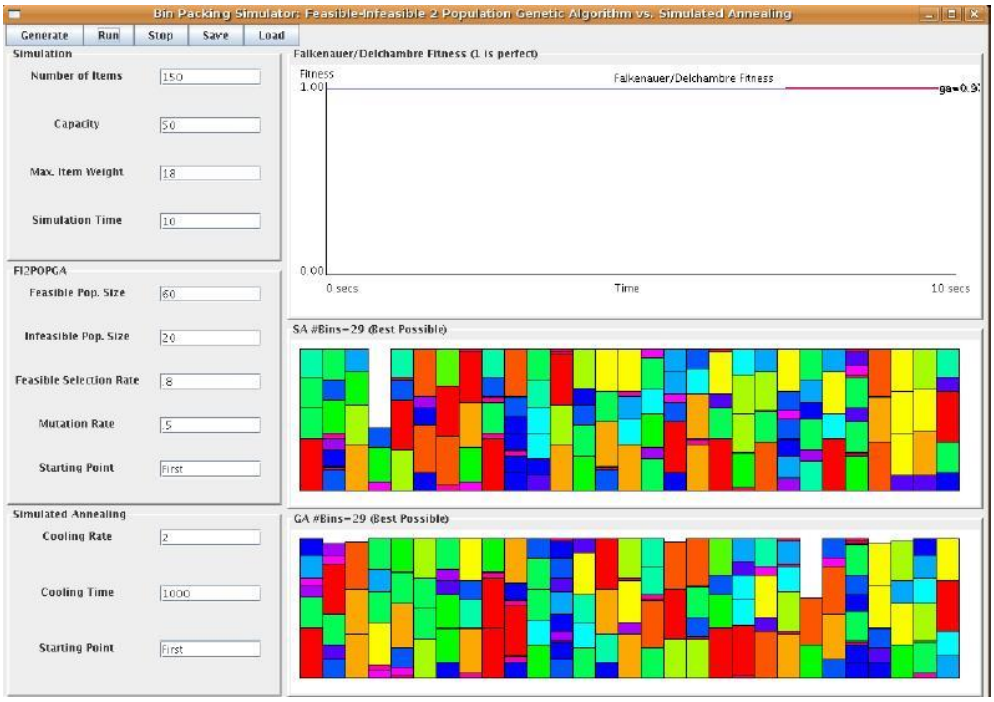

Figura 3.1 BPP Simulator

3.1.4 OAT (Optimization Algorithm Toolkit), Brownlee 2007

OAT cuenta con una sección llamada Experimenter, la cual es una interface muy sencilla, que permite configurar conjuntos de algoritmos y problemas para ejecutarse y hacer posteriormente análisis estadísticos de estas ejecuciones.

Esta herramienta está destinada a 3 tipos de usuarios:

Amateurs interesados que pueden o no tener entrenamiento en inteligencia artificial y ciencia computacional pero que están interesados en experimentar con algoritmos y problemas, apoyándose en la interfaz de exploración y experimentación gráfica.

Desarrolladores de Software que estén interesados en integrar algoritmos y problemas en su propio software, implementando sus propias técnicas o explotar la herramienta para nuevos dominios.

Científicos investigadores que puedan usar la herramienta para fines de investigación usando las interfaces gráficas de usuario e interfaces de programación aplicada.

Las visualizaciones que ofrece esta herramienta están limitadas a la gráfica de la evolución de las soluciones en la ejecución además de la representación de la solución actual. OAT es un banco de trabajo y herramienta para desarrollar, evaluar, experimentar y jugar con algoritmos de optimización clásicos y del estado del arte en problemas de dominio de referencia estándar. El proyecto incluye algoritmos implementados, graficación, visualizaciones y opciones adicionales adecuadas a cada problema.

Las herramientas estadísticas que se proveen para las pruebas estadísticas de hipótesis son: prueba de normalidad (prueba anderson-darling, prueba de criterio cramer-von-mises, y la prueba kolmogorov-smirnov), y prueba de comparación de población (análisis de varianza de una dirección (ANOVA), prueba kruskal-wallis, prueba de independencia T de Student y la Prueba-u de mann-whitney.

Esta herramienta resuelve dos problemas de optimización de inteligencia computacional (TSP y Coloreo de Grafos) con instancias, algoritmos de solución clásicos del estado del arte, y visualización básica del desempeño. Por el momento incluye los siguientes algoritmos, pero la herramienta no está limitada a estos: computación evolutiva, programación evolutiva, algoritmo genético, estrategias evolutivas, evolución diferencial, recocido simulado, algoritmo hill climbing, nube de partículas, colonia de hormigas, algoritmo voraz y optimización extrema.

La interface OAT Explorer que se muestra en la Figura 3.2, brinda un punto de entrada en el que los investigadores pueden experimentar con algoritmos e instancias que pueden ser seleccionadas, configuradas y ejecutadas. El enfoque en esta interface es la experimentación

informal exploratoria mediante la visualización de la ejecución y la colección de información generada durante la ejecución. Fue inspirado en la interface de WEKA.

OAT es un proyecto desarrollado en Java. Este proyecto pone a disposición el código fuente así como el archivo ejecutable de la herramienta. Esta herramienta aun presenta fallas en la ejecución. El autor ofrece la oportunidad de colaborar en el desarrollo de esta herramienta como trabajo futuro.

OAT incluye una librería para investigar algoritmos y problemas ya existentes, así como para implementar nuevos problemas y algoritmos. El objetivo de la librería es facilitar la mejor práctica del algoritmo, problema y diseño de experimentos e implementación, así como principios de ingeniería de software. La interfaz gráfica de usuario provee un acceso no técnico para configurar y visualizar técnicas ya existentes en instancias de problemas de referencia. Para el desarrollo de esta herramienta usaron las librerías JFreeChart, Open Source Physics, JUnit. Esta herramienta se encuentra disponible en (Brownlee, 2006 - 2008)

Figura 3. 2 OAT Explorer

3.1.5 HeuristicLab Optimization Environment, Wagner 2004

Wagner (Wagner, y otros, 2004) propone esta herramienta para romper con el paradigma de la dependencia del análisis del algoritmo con determinado problema. Este proyecto busca que se puedan ejecutar diferentes problemas con un algoritmo, y de la misma manera, resolver un problema con diferentes algoritmos. Para esta herramienta solo se encuentra disponible el archivo ejecutable en ((HEAL), s.f.).

Este proyecto, se enfoca más en la independencia, más que en la visualización del proceso algorítmico. Aun así, muestra la representación de las soluciones para algunos problemas, como se puede ver en la Figura 3.3

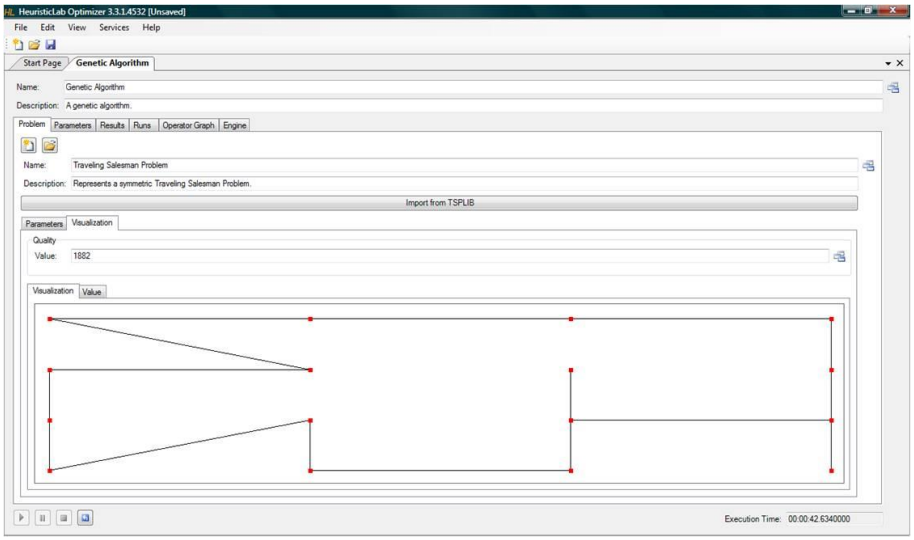

Figura 3.3. Ejemplo de ejecución en HeuristicLab Optimizer.

3.1.6 Visualizer for Metaheuristics Development Framework (V-MDF)

V-MDF es un proyecto desarrollado por Halim (Halim, y otros, 2005) busca capturar una vista pictorial de la búsqueda de trayectorias y reporta cualquier anomalía al usuario. Mediante la inspección visual de anomalías, el operador puede determinar los problemas

encontrados en la búsqueda y consecuentemente aplicar estrategias para remediarlos. Con V-MDF el diseñador del algoritmo comienza con una estrategia de búsqueda definida, y con el apoyo del visualizador, observa el comportamiento de la búsqueda y dinámicamente cambia las estrategias de búsqueda. V-MDF difiere de enfoques existentes para el ajuste de metaheurísticos en que los otros se enfocan en el diseño de un método eficiente para elegir el mejor parámetro o configuración, sin embargo, Halim extiende la idea de visualización y análisis del ajuste de trayectoria propuesto para ayudar a los usuarios a diseñar mejor los metaheurísticos al vuelo. V-MDF es útil específicamente para el diseño de metaheurísticos para nuevos problemas en los que las estrategias de búsqueda no han sido bien definidas. El demo de esta herramienta se puede descargar en (Halim, World of Seven, V-MDF, 2000 - 2013)

3.1.7 VIZ Visualization for Analyzing Trajectory-Based Metaheuristic Search Algorithms

Halim (Halim, y otros, 2007) extiende V-MDF y como resultado crea una herramienta de visualización con enfoque de caja blanca llamado VIZ. En este proyecto propone una herramienta de visualización interactiva. Combina la visualización genérica aplicable en algoritmos arbitrarios con visualizaciones del problema específico. VIZ puede ser usado para analizar visualmente algoritmos de Búsqueda Local Estocástica mientras atraviesan el espacio de soluciones de los problemas de optimización combinatoria NP-Duros.

VIZ consiste en: a) Viz Experiment Wizard VIZ (EW); y b) Viz Single Instance Multiple Runs Analyzer (SIMRA).

Entre las visualizaciones para la búsqueda local que se ofrecen en esta herramienta se encuentran las siguientes:

- Visualizaciones de búsqueda local genérica (Generic Local Search Visualizations)

a) Abstracción 2-D de la trayectoria de búsqueda (2-D Abstraction of Search Trajectory)

b) Valor objetivo sobre el tiempo (Objective Value over Time)

c) Correlación de ajuste de distancia (Fitness Distance Correlation)

d) Barra de eventos (Event Bar)

- Visualizaciones Específicas para el Algoritmo (Algorithm-Specific Visualizations)
- Visualizaciones Para el problema específico (Problem-Specific Visualizations)

Visualizaciones Genéricas de Búsqueda local. VIZ cuenta con 4 secciones principales para el análisis de búsqueda local:

a) *Fitness Ladscape and Search Trajectory*: Muestra una animación de la trayectoria de la búsqueda. Es la animación primaria en VIZ para destacar varios comportamientos que podrían estar ocultos.

b) *Objective Value:* En lugar de sólo graficar el valor objetivo sobre el tiempo/iteraciones, en VIZ mejoran esto con varia información estadística para entender todo el contexto de cómo va cambiando el valor objetivo. Esto incluye: máximo, mínimo, un histograma de frecuencia para resaltar el promedio y distribución del valor objetivo encontrado por la ejecución de la búsqueda local, una línea para indicar la mejor solución encontrada y un indicador de porcentaje que compara la solución actual contra la mejor encontrada durante toda la ejecución.

c) *Fitness Distance Correlation:* Este análisis está destinado a dar una gran medida de la dificultad del problema. Se quiere saber si existe una correlación entre la forma y la distancia de las soluciones con respecto a la mejor solución encontrada.

d) *Event Bar:* Como en un video, esta barra sirve para movernos a través de toda la ejecución, dando la posibilidad de regresar o avanzar a un momento deseado; empleando puntos o regiones claves; de esta manera evita que se muestren partes

donde no pasa algo importante. Esto se logra debido a que VIZ calcula puntos relevantes, por ejemplo, la mejor nueva solución encontrada, series de movimientos sin mejora.

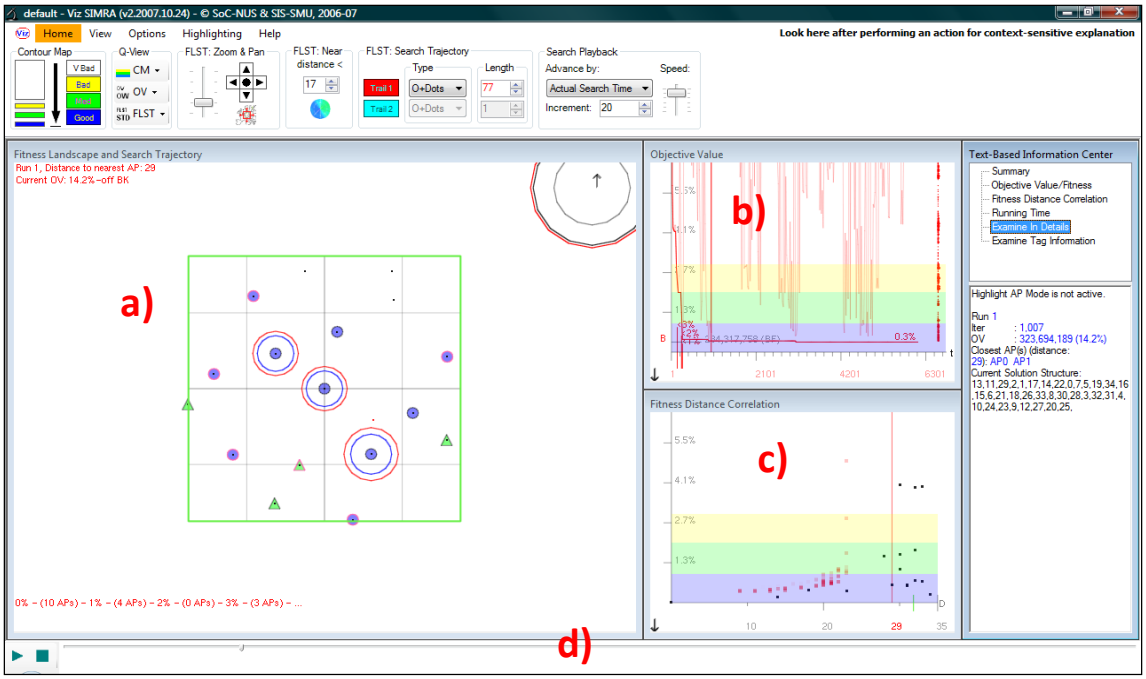

Figura 3.4 Entorno VizSIMRA

Visualización específica del algoritmo. Estas visualizaciones están relacionadas al algoritmo de búsqueda local que se está analizando. Por lo regular se visualiza el cambio de valores dinámicos en los parámetros sobre el tiempo.

Visualización específica del problema. Viz SIMRA también hace una visualización de la instancia, pero no se hacen cálculos de métricas.

Las visualizaciones del algoritmo y del problema se muestran en la Figura 3.5.

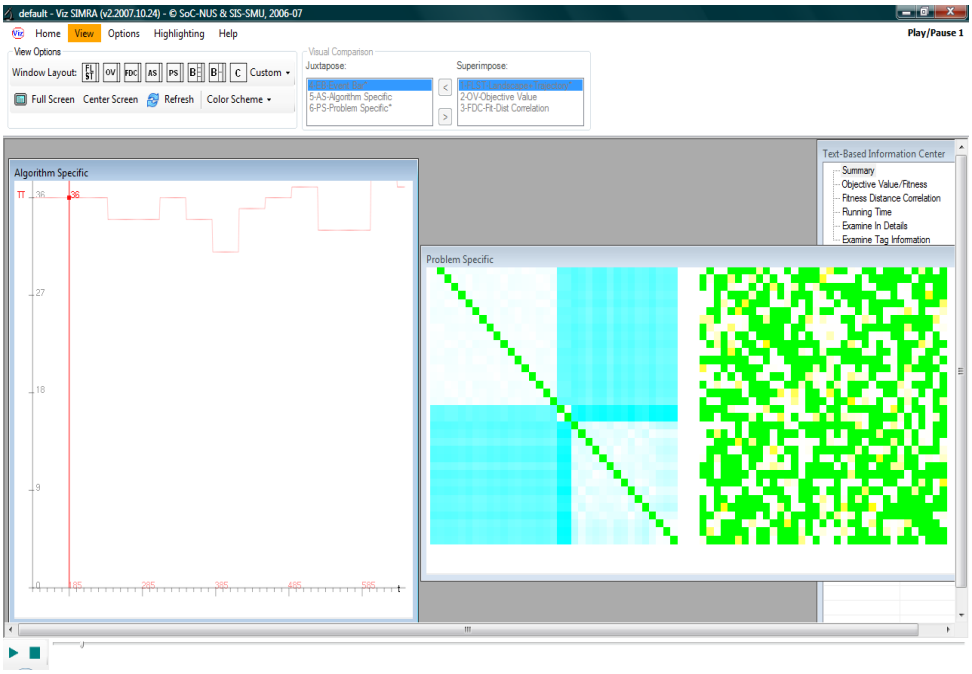

Figura 3.5 Visualizaciones en VIZ específicas del algoritmo y del problema.

VIZ se compone de 2 aplicaciones:

- Viz EW. Produce archivos de datos visuales (VDFs) que incluye la visualización de la trayectoria de la búsqueda.
- Viz SIMRA es la herramienta visual que utiliza el archivo creado en VizEW.

La Figura 3.6 muestra la interfaz gráfica de VizEW.

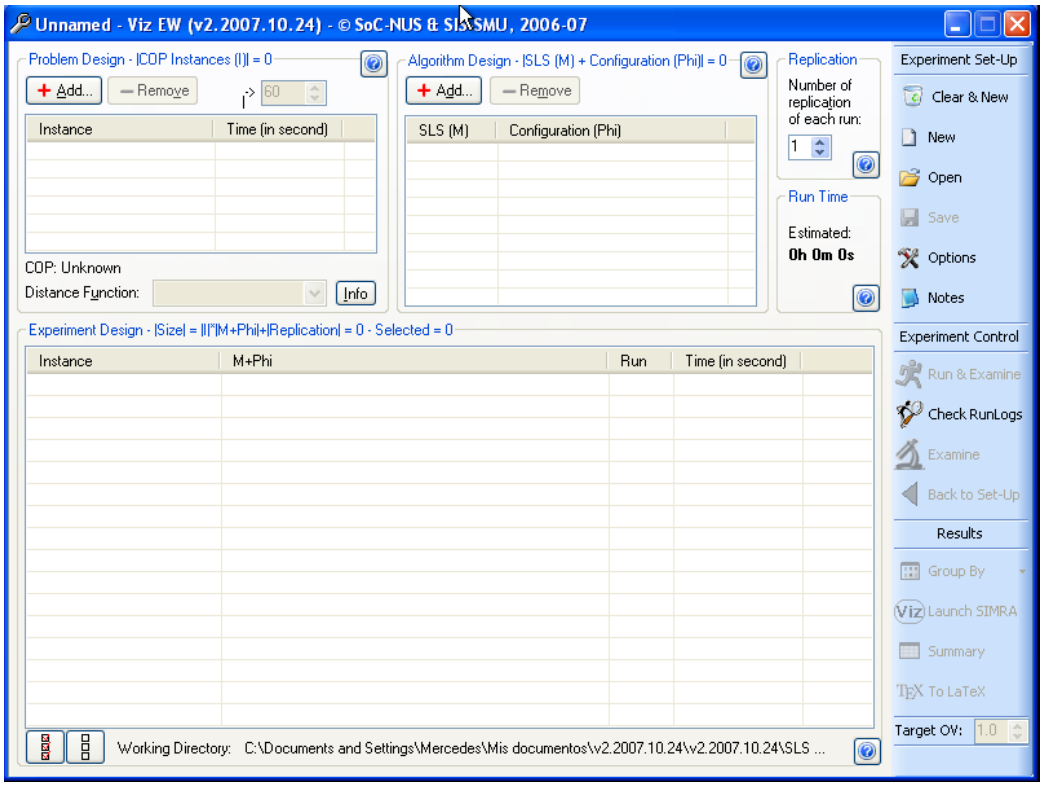

Figura 3.6 Entorno Viz EW

En Viz EW se agrega o agregan las instancias a analizar y después se elige el algoritmo de búsqueda local que se desea analizar, junto con un archivo de configuración para ejecutar el algoritmo de búsqueda local, este archivo contiene parámetros específicos para el algoritmo que se va analizar. Este archivo se muestra en la Figura 3.7.

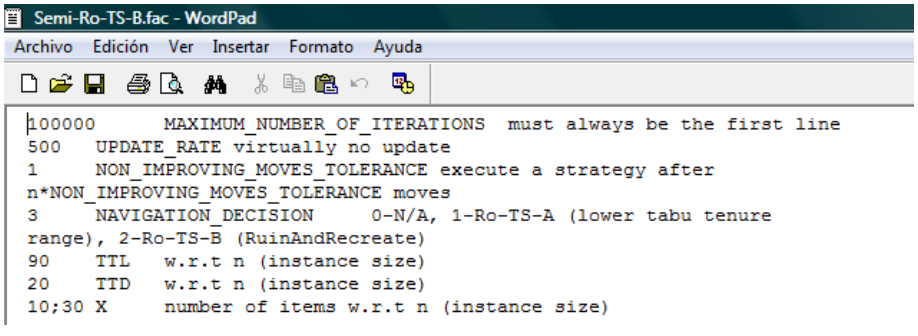

Figura 3.7 Ejemplo de archivo de configuración.

Para el análisis se crean diferentes casos de prueba tomando en cuenta las combinaciones posibles de parámetros con diferentes valores. Esto se puede ver en la Figura 3.8

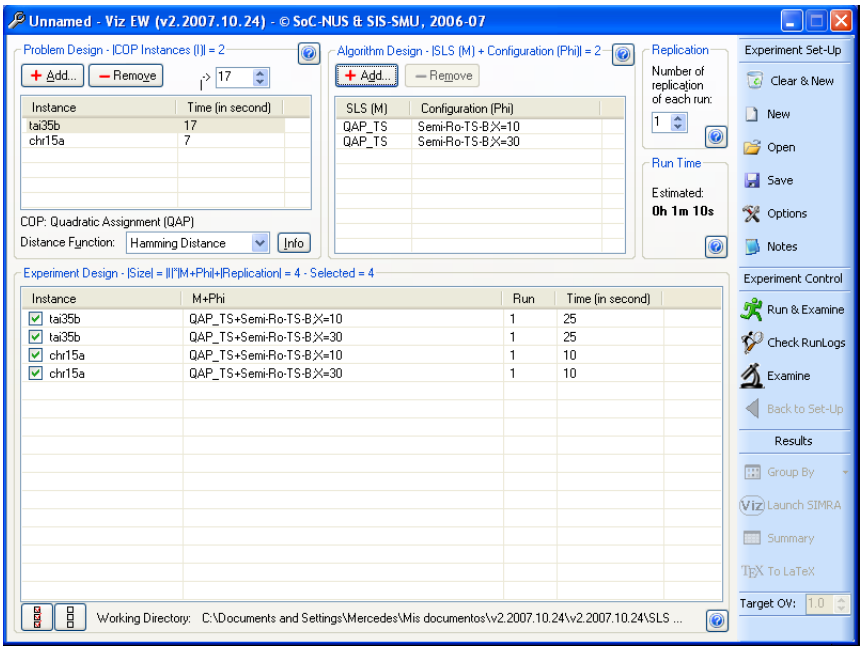

Figura 3.8 Casos de prueba resultantes del archivo de configuración.

Después de esto se ejecuta el experimento y se van creando los VDFs, al finalizar la ejecución se muestra una interfaz como la de la Figura 3.8, en la que para cada instancia con su respectiva configuración se da un resumen de los resultados obtenidos como ejemplo, el porcentaje de error.

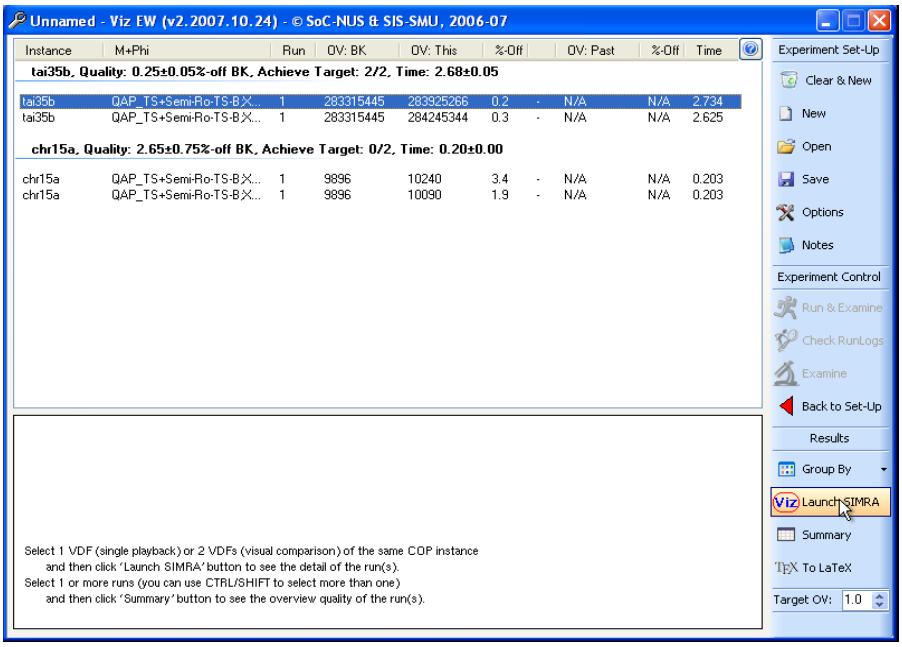

Figura 3.9 Resultados de la experimentación con diferentes configuraciones.

De los resultados obtenidos, se puede seleccionar uno o dos resultados de la misma y se pueden visualizar en Viz SIMRA, en la Figura 3.9 se muestra el entorno de esta aplicación, en esta figura se muestran las visualizaciones genéricas de la búsqueda local que son descritas a continuación.

VIZ permite analizar los problemas de Asignación Cuadrática y el Problema del Agente Viajero.

Esta herramienta está orientada a la solución de problemas como TSP y asignación cuadrática, usando algoritmos de búsqueda tabú, y hace caracterizaciones estadísticas para el problema, el algoritmo y los resultados, pero no establece una relación entre esas características.

Durante la puesta en marcha de VIZ se producía un error, el cual se solucionó ejecutando la aplicación en Windows XP.

Usando el algoritmo de Búsqueda Tabú. VIZ permite el análisis de nuestros propios algoritmos de búsqueda local, para hacer esto se deben seguir ciertas especificaciones. Esta herramienta se encuentra disponible en (Halim, World of Seven - Viz: SLS Engineering Suite, 2000 - 2013).

3.2 Análisis de Algoritmos Metaheurísticos mediante el uso de Métodos Estadísticos

En esta sección se describen investigaciones que abordan el estudio del análisis del comportamiento de algoritmos metaheurísticos mediante el uso de métodos estadísticos, en contraste con el tema de tesis no cuentan con un visualizador gráfico del comportamiento del algoritmo:

a) "Automatización del diseño de la Fragmentación Vertical y Ubicación en Bases de Datos Distribuidas usando Métodos Heurísticos y Exactos", este trabajo muestra que a partir de datos experimentales y mediante un tratamiento estadístico, se pueden obtener funciones de eficiencia y eficacia de los algoritmos con lo cual es posible predecir el comportamiento de estos, se mostró que se pudo obtener un algoritmo que dados como entrada el tamaño del problema y los parámetros de eficiencia y tolerancia deseados, recomienda el método de solución que cumple con los requerimientos, o bien, señala que la solución no es factible. Para esto, se incorporaron en el algoritmo las funciones de eficiencia y eficacia de los métodos. Con la función polinomial de eficiencia de un algoritmo se pretende describir la relación entre el tamaño del problema y el tiempo que tarda el algoritmo en obtener la solución de dicho problema, usa el análisis de regresión para obtener las funciones polinomiales que permiten estimar la eficiencia de los algoritmos cuando se conoce el tamaño del problema. De esta manera, el algoritmo de selección de métodos trata con problemas a gran escala, por lo que la solución de problemas de tamaño real es factible (Cruz, 1999).

b) "Predicción del Desempeño de Algoritmos Exactos y Heurísticos: un Enfoque Estadístico", en este artículo se muestra que es factible caracterizar diferentes algoritmos obteniendo sus funciones de desempeño lo que ayuda a comprender mejor el comportamiento de los algoritmos, utilizando pruebas estadísticas muy conocidas para relacionar significativamente el desempeño empírico de los algoritmos y concluye cuál de ellos es el

mejor, es decir, primero genera una muestra representativa del comportamiento de los algoritmos, posteriormente mediante análisis de regresión, determina las funciones de desempeño, las que incorpora finalmente a un mecanismo de selección de algoritmos [Pérez 02].

c) "Modelo para representar la complejidad del problema y el desempeño de algoritmos", se desarrollaron 21 índices de complejidad basados en estadística descriptiva para medir la influencia de las características estructurales del problema sobre el desempeño algorítmico y se desarrolló un método RPI (Regla de Pertenencia Informada) para validar el desempeño del agente de selección de algoritmos, basado en reglas de pertenencia que consideran el comportamiento de los algoritmos de prueba (Álvarez, 2006).

d) "Modelado Causal del Desempeño de Algoritmos Metaheurísticos en Problemas de distribución de objetos", se desarrolló una metodología para la construcción sistemática de modelos causales que representan las relaciones entre la complejidad del problema, el comportamiento del algoritmo y el desempeño obtenido por el mismo. El modelado causal aplicado al análisis de algoritmos permite explicar con rigor estadístico, cómo la naturaleza del problema y la estructura de diseño de los algoritmos afectan su desempeño. Además se presenta la formulación de indicadores que describen al problema, el comportamiento del algoritmo y su desempeño y una estrategia para generar e interpretar modelos causales a partir de indicadores del proceso algorítmico para identificar relaciones entre problema, algoritmo y desempeño (Pérez, 2007).

e) "Caracterización de factores de desempeño de algoritmos de solución de BPP", se desarrolló una metodología experimental para el análisis del desempeño de algoritmos metaheurísticos que permite comprender el comportamiento del algoritmo y mejorar su desempeño además se definió un conjunto de índices de caracterización para el comportamiento del algoritmo y se generaron modelos de desempeño que permitieron

obtener explicaciones del comportamiento del algoritmo genético HGGA-BP en la solución de instancias de BPP (Bin Packing Problem) con diferentes estructuras. (Quiroz, 2009).

f) "Caracterización del Proceso de Optimización de Algoritmos Heurísticos", en este trabajo se desarrolló una metodología experimental para el análisis del comportamiento de algoritmos metaheurísticos. Se aportó un conjunto de índices de caracterización de BPP (Bin Packing Problem) que impactan en el comportamiento y se mejoró el desempeño del algoritmo HGGA-BP [Cruz 10].

3.3 Herramientas de Análisis Estadístico

3.3.1 Taillard 2003

Este artículo presenta una prueba no-paramétrica que es de interés para aquellos que deseen comparar diferentes algoritmos heurístico que no necesariamente terminan con soluciones factibles (o satisfactorias). Esta prueba ha sido diseñada para trabajar con muestras de tamaño muy pequeño, lo que significa que se puede ahorrar esfuerzo computacional cuando se realicen experimentos numéricos.

3.3.2 García 2008

En éste artículo se enfocan en el estudio del uso de técnicas estadísticas en el análisis del comportamiento de algoritmos evolutivos en los problemas de optimización. El estudio es conducido de dos maneras: análisis de un solo problema y análisis de múltiples problemas. Los resultados obtenidos establecen que una prueba estadística paramétrica puede no ser apropiada especialmente cuando se trata con resultados múltiples problemas. En el análisis de múltiples-problemas, propone el uso de pruebas estadísticas no-paramétricas ya que son menos restrictivas que las pruebas paramétricas y pueden ser usadas en resultados de muestras de tamaño pequeño.

En éste artículo se comparó el desempeño de los siguientes algoritmos evolutivos:

BLXGL50 (García-Martínez y Lozano 2005), *BLX-MA* (Molina et al. 2005), *CoEVO* (Pošík 2005), *DE* (Rônkkônen et al. 2005), *DMS-L-PSO* (Liang y Suganthan 2005), *EDA* (Yuan y Gallagher 2005), *G-CMA-ES* (Auger y Hansen 2005a), *K-PCX* (Sinha et al. 2005), *L-CMA-ES* (Auger y Hansen 2005b), *L-SaDE* (Qin y Suganthan 2005), *SPC-PNX* (Ballester et al. 2005).

Para hacer las pruebas paramétricas se verificaron las siguientes condiciones:

- Independencia, en éste caso de estudio se obvia la independencia de los eventos.
- Normalidad: verificada con las pruebas de normalidad Komogorov-Smirnov, Shapiro-Wilk y D'Agostino-Pearson.
- Heteroscedasticidad: verificada con la prueba Levene.

Para el análisis de un solo problema se aplicó la prueba paramétrica T-pareada y la prueba no-paramétrica Wilcoxon; en el caso del análisis de múltiples problemas, se aplicaron las pruebas Friedman, Iman Davenport, Bonferroni-Dunn, Holm, Hochberg y Wilcoxon.

3.3.3 Derrac 2012

En este trabajo se revisan los métodos no paramétricos de comparaciones múltiples más representativos, aplicados a un caso de estudio. Para la revisión se ha seleccionado como caso de uso una comparación basada en los 25 problemas presentados en la Sesión Especial de Optimización de Parámetros Reales del Congreso IEEE sobre Computación Evolutiva de 2005 (CEC'2005). En la comparación, se han empleado 4 algoritmos clásicos: Un algoritmo de Optimización de Nube de Partículas (PSO), un algoritmo Genético Estacionario (SSGA), un algoritmo de Búsqueda Dispersa con operador de cruce BLX (SS-BLX) y un modelo de Evolución Diferencial con operador de cruce Rand/1/exp (DE-EXP).

Se usó el Test de Friedman como procedimiento para realizar comparaciones múltiples entre diferentes algoritmos. Así mismo se aplicaron los Tests de Friedman Alineado y Quade, como versiones avanzadas del mismo.

3.3.4 Knowledge Extraction based on Evolutionary Learning (KEEL)

KEEL es una herramienta software para evaluar algoritmos evolutivos para problemas de minería de datos incluyendo regresión, clasificación, agrupamiento, patrones de minería entre otros. Esta herramienta se encuentra disponible en (Fernández, García, & Derrac, 2004-2013). La versión actual de KEEL está compuesta de las siguientes funcionalidades:

a) *Data Management:* Esta parte está compuesta por un conjunto de herramientas que pueden ser usadas para construir nuevos datos, exportar e importar datos en otros formatos a formato KEEL, edición de datos y visualización, aplicar transformaciones y particionamientos a los datos. La interfaz de esta sección se muestra en la Figura 3. 10.

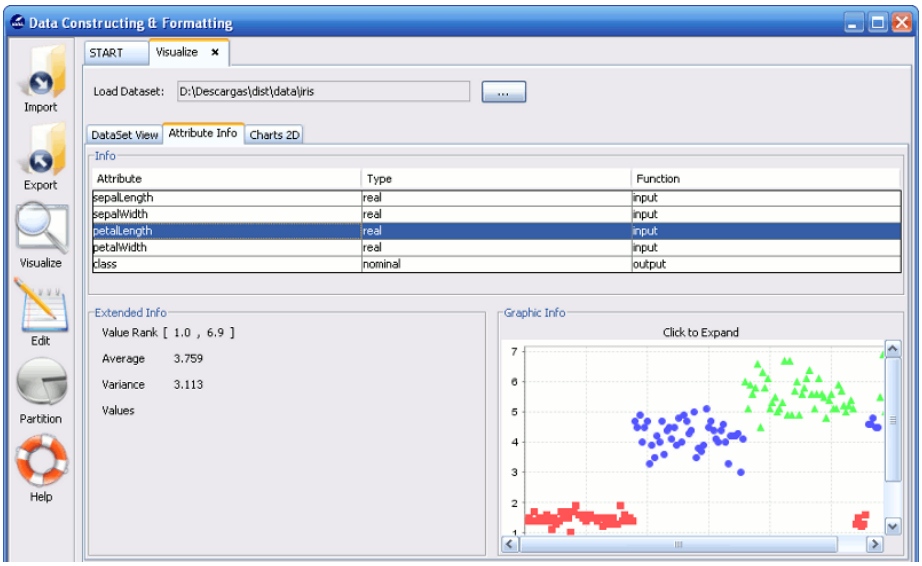

Figura 3. 10 Data Management

b) *Design of Experiments:* El objetivo de esta funcionalidad es el diseño de la experimentación deseada sobre los conjuntos de datos seleccionados. Provee opciones para muchas alternativas: tipo de validación, tipo de aprendizaje (clasificación, regresión, aprendizaje no supervisado, descubrimiento de subgrupos). La interfaz de esta sección se muestra en la Figura 3. 11.

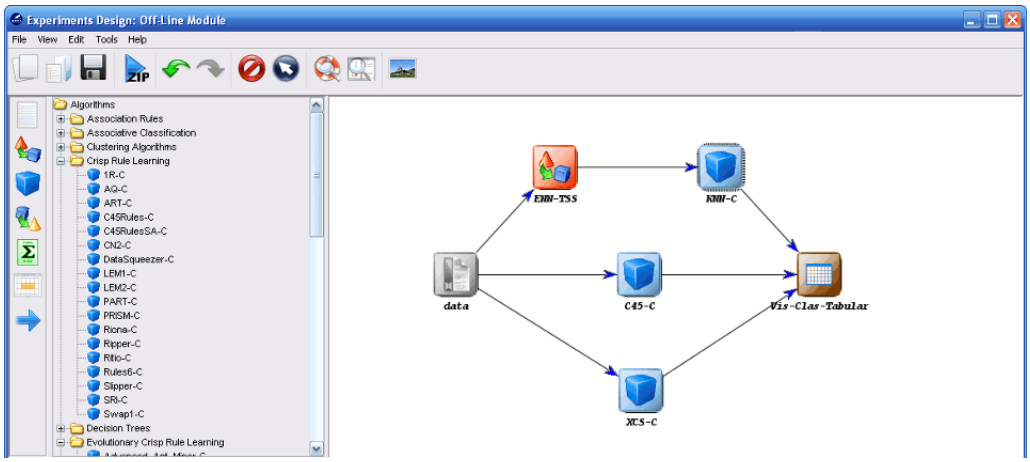

Figura 3. 11 Design of Experiments

c) *Design of Imbalanced Experiments:* El objetivo de esta funcionalidad es el diseño de la experimentación deseada sobre los conjuntos de datos seleccionados desequilibrados. Estos experimentos son creados para conjuntos de datos 5cfv e incluye algoritmos específicos para datos desequilibrados y algoritmos de clasificación generales. La interfaz de esta sección se muestra en la Figura 3. 12.

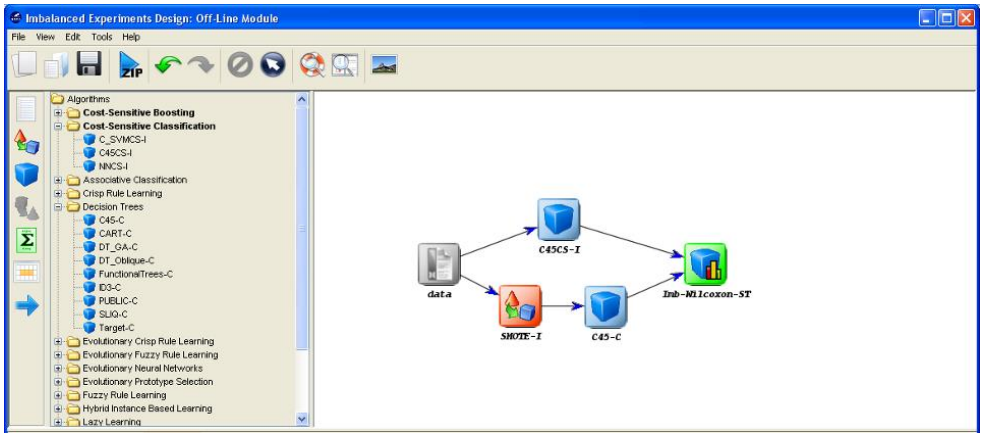

Figura 3. 12 Design of Imbalanced Experiments

d) *Experimentation with Multiple Instance Learning Algorithms:* En esta sección, cualquier investigador es capaz de hacer frente a la clasificación con múltiples instancias de conjuntos de datos. En este caso, en lugar de recibir un conjunto de instancias que están etiquetadas como positivas o negativas, el aprendiz recibe un conjunto de carteras con múltiples instancias, que son etiquetadas como positivas o

negativas. El supuesto más común es que una cartera es etiquetada negativa si todas las instancias que contiene son negativas. De otra manera, una cartera es etiquetada positiva si hay en ella al menos una instancia la cual es positiva. La interfaz de esta sección se muestra en la Figura 3. 13.

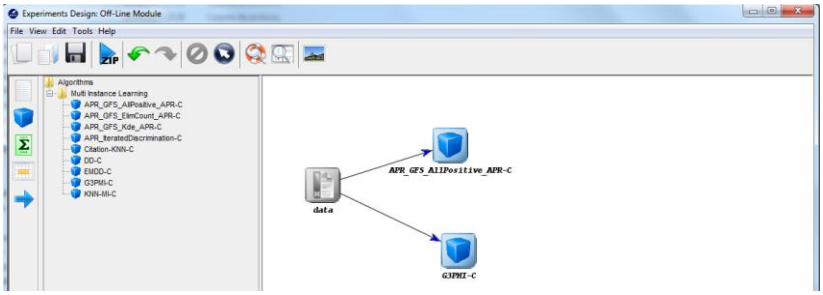

Figura 3. 13 Experimentation with Multiple Instance Learning Algorithms

e) *Statistical Tests:* KEEL es una de las pocas herramientas software de minería de datos que provee al investigador un conjunto completo de procedimientos estadísticos para comparaciones pareadas y múltiples. Dentro del ambiente KEEL se han codificado varios procedimientos paramétricos y no paramétricos, los cuales deberían ayudar a contrastar los resultados obtenidos en cualquier experimento realizado con la herramienta software. La interfaz de esta sección se muestra en la

f) Figura 3. *14*.

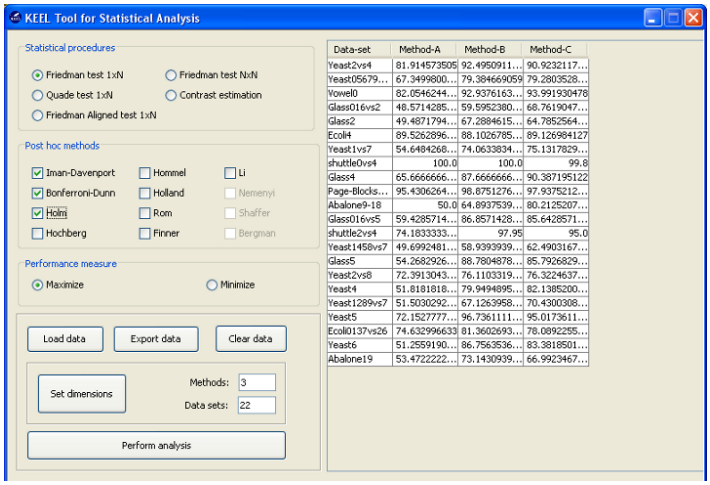

Figura 3. 14 Statistical Tests

g) *Educational Experiments:* Con una estructura similar a la parte de diseño de experimentos, KEEL permite diseñar un experimento el cual puede ser depurado paso

a paso para usarlo como una guía para mostrar el proceso de aprendizaje de un cierto modelo usando la plataforma con objetivos educacionales. La interfaz de esta sección se muestra en la Figura 3. 15.

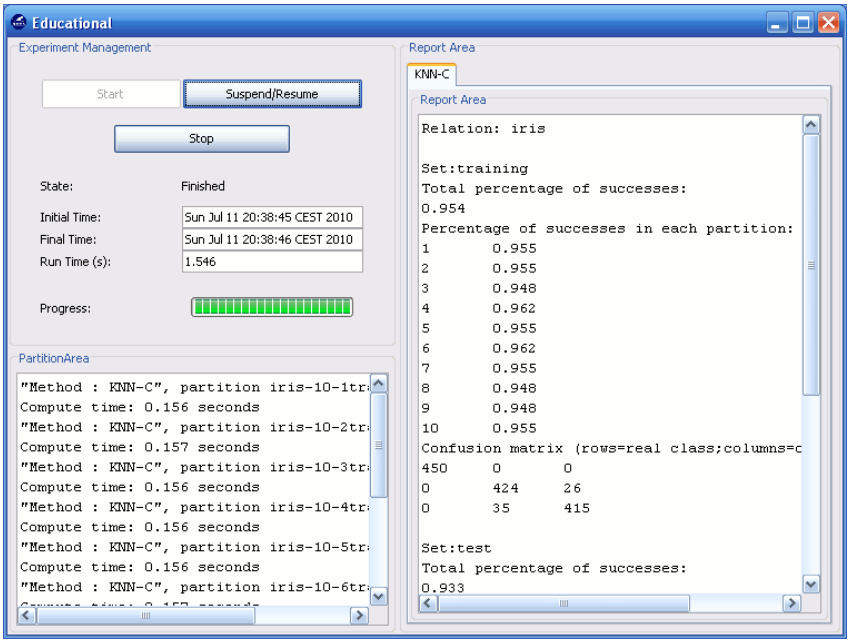

Figura 3. 15 Educational Experiments

CAPITULO 4

Análisis Estadístico

4.1 Prueba Estadística

Una función central de la estadística moderna es la inferencia estadística. La estadística inferencial está interesada en dos tipos de problemas: la estimación de los parámetros de la población y las pruebas de hipótesis.

El verbo inferir significa "obtener conclusiones como una consecuencia o una probabilidad".

El interés de la inferencia estadística es el cómo obtener conclusiones acerca de grandes grupos de sujetos o de eventos, sobre la base e observaciones de pocos sujetos o de los que ha ocurrido en el pasado. La estadística proporciona instrumentos que formalizan y estandarizan procedimientos para obtener tales conclusiones.

Un problema común para la inferencia estadística es determinar, en términos de probabilidad, si las diferencias observadas entre dos muestras significa que las poblaciones muestreadas son realmente diferentes. Los procedimientos de la inferencia estadística permiten determinar si las diferencias observadas están o no dentro del grado en que podrían haber ocurrido simplemente por azar. Otro problema común es determinar si una muestra de puntuaciones pertenece a alguna población específica. Un problema adicional consiste en decidir su podemos inferir legítimamente que varios grupos difieren entre ellos.

En el desarrollo de los métodos estadísticos modernos, las primeras técnicas de inferencia que aparecieron fueron aquellas que hicieron suposiciones acerca de la naturaleza de las poblaciones de las cuales se derivaron las observaciones y los datos. Estas técnicas estadísticas se laman paramétricas. Por ejemplo, una técnica de inferencia puede estar basada

en la suposición de que los datos se derivan de una población normalmente distribuida. Otra técnica de inferencia puede estar basada en la suposición de que dos conjuntos de datos se tomaron de poblaciones que tienen la misma varianza o dispersión de puntuaciones. Tales técnicas proporcionan conclusiones de la forma siguiente: “Si las suposiciones acerca e la forma de a distribución de la población son válidas, entonces podemos concluir que…”. Debido a las suposiciones comunes, tales se sistematizan fácilmente y son también muy fáciles de enseñar y aplicar.

Un poco más recientemente hemos presenciado el desarrollo de un gran número de técnicas de inferencia que no hacen suposiciones numerosas o rigurosas acerca de la población de la cual se han muestreado los datos. Estas técnicas de distribución libre o no paramétricas dan como resultado conclusiones que requieren menos calificaciones. Si hemos usado una de estas técnicas, seremos capaces de decir que: “Sin considerar la(s) forma(s) de la(s) población(es), podemos concluir que…”.

4.1.1 Pruebas Estadísticas Paramétricas y No Paramétricas

Una *prueba estadística paramétrica* específica ciertas condiciones acerca de la distribución de respuesta en la población de la cual se ha obtenido la muestra investigada. Ya que estas condiciones no son ordinariamente evaluadas, sólo se suponen. La significación de los resultados de la prueba paramétrica depende de la validez de estas suposiciones. Una adecuada interpretación de las pruebas paramétricas basadas en la distribución normal también supone que las puntuaciones que están siendo analizadas resultan de medidas en por lo menos una escala de intervalo.

Una *prueba estadística no par*amétrica está basada en un modelo que especifica sólo condiciones muy generales y ninguna acerca de la forma específica de la distribución de la cual fe obtenida la muestra. Ciertas suposiciones están asociadas con la mayoría de las pruebas no paramétricas, a saber: que las observaciones son independientes y quizá que a variable en estudio es continua; pero estas suposiciones son menores y más débiles que aquellas asociadas con las pruebas paramétricas. Los procedimientos no paramétricos

prueban diferentes hipótesis acerca de la población, que los procedimientos paramétricos no hacen. A diferencia de las pruebas paramétricas, existen pruebas no paramétricas que pueden aplicarse apropiadamente a datos medidos en una escala ordinal, y otras pruebas para datos en una escala nominal o categórica.

4.1.2 Uso De Pruebas No Paramétricas Para Analizar Algoritmos

Si el tamaño de la muestra es muy pequeño, puede no haber otra opción que usar una prueba estadística no paramétrica, a menos que la naturaleza de la distribución de la población se conozca con exactitud.

Las pruebas no paramétricas típicamente hacen menos suposiciones numéricas de los datos y pueden ser más relevantes a una situación particular.

Las pruebas estadísticas no paramétricas están disponibles para analizar datos que son inherentes a los rangos, así como datos cuyas puntuaciones numéricas tienen aparentemente a fuerza de los rangos.

Los métodos no paramétricos están disponibles para tratar datos que son simplemente clasificatorios o categóricos. Ninguna técnica paramétrica se aplica a tales datos.

Existen pruebas estadísticas no paramétricas que son adecuadas para tratar muestras obtenidas de observaciones de diferentes poblaciones.

Las pruebas no paramétricas típicamente son más fáciles de aprender y aplicar que las pruebas paramétricas. Además su interpretación suele ser más directa que la interpretación de las pruebas paramétricas.

4.2 Pruebas No-Paramétricas

4.2.1 Prueba de los signos

Una forma popular para comparar los desempeños generales de los algoritmos es contar el número de casos en el que el algoritmo es el ganador en general. Algunos autores también usan estos conteos en estadísticas inferenciales, con una forma de prueba binomial de dos colas que es conocida como la prueba de los Signos. Si ambos algoritmos comparados son, como se asume bajo la hipótesis nula, equivalentes, cada uno debería ganar en aproximadamente $n/2$ de los n problemas.

El número de éxitos es distribuido de acuerdo a una distribución binomial; para un mayor número de casos, el número de éxitos está bajo la hipótesis nula distribuida de acuerdo a $n(n/2, \sqrt{n}/2)$, lo cual permite el uso de la prueba z: si el número de victorias es al menos $n/2 + 1.96 \cdot \sqrt{n}/2$ (o, por regla general rápida, $n/2 + \sqrt{n}$), entonces el algoritmo es significativamente mejor con $p < 0.05$.

La Tabla B.1 del Anexo B muestra el número crítico de victorias necesarias para alcanzar los niveles de significancia $\alpha = 0.05$ y $\alpha = 0.1$. Hay que tomar en cuenta que aunque los empates dan soporte a la hipótesis nula, no deben dejar de contarse cuando se aplica este test, deben ser divididos en partes iguales entre los dos algoritmos; si hay un número impar de ellos, uno debe ser ignorado.

4.2.2 Test de Signos Múltiple

El Test de Signos Múltiple es un sencillo procedimiento para realizar comparaciones $1xN$. Puede ser útil en casos en los que se desee realizar un primer estudio preliminar de los resultados.

Este test es una extensión del Test de Signos convencional, capaz de comparar simultáneamente k métodos contra un método de control. Consiste en los siguientes pasos:

1. Representar con x_{i1} los valores de rendimiento del método control en el conjunto i-ésimo. Representar con x_{ij} los valores de rendimiento del resto de métodos (j-ésimo método en el conjunto i-ésimo).
2. Calcular los signos de las diferencias $d_{ij} = x_{i1} - x_{ij}$, asignando $(+)$ en el caso de que el método de control ofrezca un peor rendimiento, o $(-)$ en caso contrario. Las diferencias a 0 (empates, $(=)$) se descartan.
3. Calcular r_j como el número de diferencias d_{ij} con el signo menos frecuente, $(+)$ ó $(-)$, dentro de un emparejamiento del algoritmo j con el control.
4. Sea M_1 la respuesta mediana de una muestra de resultados del algoritmo de control y M_j la respuestas mediana de una muestra de resultados del algoritmo j-ésimo. El test de signos permite aplicar una de las dos reglas de decisión siguientes:
   - Para comprobar $H_0: M_j \geq M_1$ contra $H_0: M_j < M_1$, se rechaza H_0 si el número de signos $(+)$ es igual o inferior que el valor crítico de R_j que aparece en la Tabla B.2 del Anexo B para $k-1$ (número de algoritmos excluyendo el control), n (número de problemas) y el nivel de significancia escogido.
   - Para comprobar $H_0: M_j \leq M_1$ contra $H_0: M_j > M_1$, se rechaza H_0 si el número de signos $(-)$ es igual o inferior que el valor crítico de R_j que aparece en la Tabla B.2 del Anexo B para $k-1$ (número de algoritmos excluyendo el control), n (número de problemas) y el nivel de significancia escogido.

4.2.3 Estimación de Contraste

Utilizando los datos resultantes de la ejecución de varios algoritmos sobre múltiples problemas, un investigador podría estar interesado en la estimación de las diferencias entre el rendimiento de dos de ellos.

Un procedimiento que busca este propósito es la Estimación de Contraste, la cual supone que las diferencias esperadas entre rendimientos de algoritmos son las mismas entre problemas. Asumimos que cada medición de rendimiento se refleja como diferencias entre rendimientos de los algoritmos. Es decir, estamos interesados en estimar el contraste entre medianas de

muestras de resultados considerando todas las comparaciones por pares. Para ello, la Estimación de Contraste obtiene una diferencia cuantitativa calculada mediante medianas entre dos algoritmos.

Se procede de la siguiente manera:

1. Para cada pareja de k algoritmos en el experimento, calcular la diferencia entre los rendimientos de ambos en cada uno de los n problemas, $D_i(uv) = x_{iu} - x_{iv}$ donde $i = 1 \ldots n, u = 1 \ldots k$ y $v = 1 \ldots k$. Se consideran solo aquellas diferencias donde $u < v$.
2. Buscar la mediana de cada conjunto de diferencias, Z_{uv}. Z_{uv} es el estimador no ajustado de la diferencia entre medianas de u y v. Nótese que $Z_{uu} = 0$.
3. Calcular la media de cada conjunto de medianas no ajustadas que tienen el mismo prefijo u , m_u:

$$m_u = \frac{\sum_{j=1}^{k} Z_{uj}}{k}, u = 1, \ldots, k$$

4. El estimador $M_u - M_v$ es $m_u - m_v$, donde u y v están en el rango desde 1 hasta k. Por ejemplo, la diferencia entre $M_1 y\, M_2$ esta estimado por $m_1 - m_2$.

Estos estimadores pueden ser entendidos como una medida avanzada de rendimiento global. A pesar de que esta prueba no provee una probabilidad de error asociada con el rechazo de la hipótesis nula de igualdad, es especialmente usada para estimar qué tanto un algoritmo supera a otro.
Los algoritmos con valores negativos en los estimadores son los que presentan un menor porcentaje de error.

4.2.4 Prueba de Wilcoxon de rangos con signos

La prueba de Wilcoxon de rangos con signos es una prueba estadística no paramétrica de hipótesis para medidas repetidas de una sola muestra. Esta prueba estadística es usada para las muestras que no están distribuidas normalmente. Con esta prueba podemos saber si el

resultado de usar un algoritmo es significativamente diferente que usar otro algoritmo para resolver la misma muestra.

Esta prueba es usada para responder si dos muestras representan dos diferentes poblaciones. Es un procedimiento no paramétrico empleado en situaciones de pruebas de hipótesis, involucrando un diseño con dos muestras. Esta prueba es el análogo a la prueba t pareada en procedimientos estadísticos no paramétricos; por lo tanto, es una prueba de pares que tiene por objeto detectar diferencias significativas entre dos medias muestrales, el cual es el comportamiento de dos algoritmos.

La prueba de Wilcoxon está definida como sigue. Sea d_i la diferencia entre los resultados de desempeño de dos algoritmos en el i-ésimo de los n problemas (si estos resultados de desempeños son conocidos para ser representados en diferentes rangos, pueden ser normalizados en el intervalo [0,1], para no priorizar cualquier problema). Las diferencias son jerarquizadas de acuerdo a sus valores absolutos; en caso de empates, se recomienda el uso del promedio de los rangos con empates (por ejemplo, si dos diferencias están empatadas en la asignación de los rangos 1 y 2, asignar el rango 1.5 a ambas diferencias).

Sea $R+$ la suma de los rangos para los problemas en los cuales el primer algoritmo superó al segundo, y $R-$ sea la suma de los rangos para lo opuesto. Los rangos de $d_i = 0$ se dividen por igual entre las sumas; si hay un número impar en ellos, uno se pasa por alto:

$$R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i)$$

$$R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i)$$

Sea T la menor de las sumas, $T = min(R+, R-)$. Si T es menor o igual que el valor de la distribución de Wilcoxon por n grados de libertad (Ver Tabla B.4), la hipótesis nula de igualdad de medias es rechazada; esto significa que un algoritmo dado supera al otro, con el

p-value asociado. Dado su amplio uso, el cálculo del p-value para esta prueba esta usualmente incluida en paquetes de software estadísticos conocidos (SPSS, SAS, R, etc.).

La prueba Wilcoxon de rangos con signos es más sensible que la prueba t. Asume conmensurabilidad de diferencias, pero solo cualitativamente: diferencias mayores todavía cuentan por más, lo cual es probablemente deseado, pero las magnitudes absolutas son ignoradas. Desde el punto de vista estadístico, la prueba es segura dado que no asume distribuciones normales. Además, los valores atípicos (desempeños excepcionalmente buenos/malos de algunos problemas) tienen menos efecto en la prueba de Wilcoxon que en la prueba t. La prueba de Wilcoxon asume diferencias continuas d_i; por lo tanto, no deben estar redondeadas a uno o dos decimales, dado que esto disminuiría la potencia de la prueba en el caso de un alto número de empates.

4.2.5 Test de Friedman

En 1937 Milton Friedman se dejó tentar por la estadística y propuso un orden de rangos en el análisis de varianza, en un trabajo que aun hoy se recuerda como la aportación básica en el desarrollo de métodos no paramétricos para el análisis de varianza.

De los años de dedicación a la estadística en Columbia durante la guerrea resulto otro trabajo importante (1948) sobre muestreo secuencial: Friendman comprobó que, en la exploración de una muestra, el mismo proceso de verificación aportaba información sobre el grado de confianza alcanzado, de modo que se podía interrumpir el muestreo antes de alcanzar el tamaño muestral prefijado

El Test de Friedman trabaja asignando rankings r_{ij} a los resultados obtenidos por cada algoritmo j en cada problema i. Esto es, para cada problema, se asigna un ranking $1 \leq r_{ij} \leq k$, donde k es el número de algoritmos a comparar. Estos rankings se asignan de forma

ascendente, es decir, 1 al mejor resultado, 2 al segundo, etc. (en caso de haber empates, se asignan rankings medios).

El Test de Friedman requiere el cálculo de los rankings medios de los algoritmos sobre los n problemas,

$$R_j = \frac{\sum_{i=1}^{n} r_{ij}}{n}$$

La hipótesis nula indica que todos los algoritmos se comportan similarmente, por lo que sus rankings R_j deben ser similares. Siguiendo esta hipótesis, el estadístico de Friedman se distribuye de acuerdo a una distribución x^2con $k-1$grados de libertad.

$$F_F = \frac{12n}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right]$$

Este estadístico fue mejorado a su vez por Iman y Davenport, quienes mostraron que el estadístico de Friedman presenta un comportamiento demasiado conservativo. Para evitar éste problema, propusieron otro estadístico más ajustado que se distribuye de acuerdo a una distribución F con $k-1$ y $(k-1)(n-1)$ grados de libertad.

$$F_{ID} = \frac{(n-1)F_F}{n(k-1)-F_F}$$

4.2.6 Test de Quade

El Test de Friedman considera que todos los problemas son iguales en importancia. Una alternativa a esto podría tener en cuenta que algunos problemas son más difíciles o que las diferencias registradas en la ejecución de varios algoritmos sobre ellos son más distantes. Así, los rankings calculados en cada problema podrían escalarse dependiendo de las diferencias observadas en los rendimientos de los algoritmos.

El test de Quade lleva a cabo un análisis con rankings ponderados de las muestras de resultados. El procedimiento para esta prueba es el siguiente:

- Encontrar los rankings r_{ij} de la misma forma que el Test de Friedman. Esto es, por cada algoritmo *j* en cada problema i se asigna un ranking $1 \leq r_{ij} \leq k$.
- El siguiente paso requiere los valores originales del rendimiento de los algoritmos x_{ij}.
- Los rankings se asignan a los problemas de acuerdo al tamaño del rango de la muestra en cada uno. El rango de la muestra en un problema i es la diferencia entre la observación más alta y la más baja en dicho problema

$$\text{Rango en el problema } i = \max_j x_{ij} - \min_j x_{ij}$$

- Asignar el ranking 1 al conjunto i con menor rango y así sucesivamente hasta el mayor rango n, en caso de haber empates se asigna un ranking medio.
- Sean $Q_1, Q_2, \ldots, Q_n$ los rankings asignados a los problemas $1, 2, \ldots, n$, respectivamente. Calcular un estadístico S_{ij} que representa el tamaño relativo de cada observación dentro de cada problema, ajustado para reflejar la significancia relativa del problema en el que aparece.

$$S_{ij} = Q_i\left[r_{ij} - \frac{k+1}{2}\right]$$

- Para relacionarlo con Friedman, usar el ranking sin ajuste medio:

$$W_{ij} = Q_i[r_{ij}]$$

- S_j denota la suma para cada clasificador $S_j = \sum_{i=1}^{n} S_{ij}$ y $W_j = \sum_{i=1}^{n} W_{ij}$ para $j = 1,2,\ldots k$.
- Calcular los términos *A* y *B* para posteriormente calcular el estadístico F_Q, el cual está distribuido de acuerdo a una distribución F con $k-1$ y $(k-1)(n-1)$ grados de libertad, cuyos valores críticos se pueden consultar en la Tabla B.3 en el Anexo B.

$$A = n(n+1)(2n+1)k(k+1)(k-1)/72$$

$$B = \frac{1}{n}\sum_{j=1}^{k} S_j^2$$

$$F_Q = \frac{(n-1)B}{A-B}$$

- Calcular los rankings medios T_j para cada j. Los rangos medios T_j pueden ser comparados con los rankings $R_j$obtenidos por el test de Friedman clásico. T_j se calcula de la siguiente manera:

$$T_j = \frac{W_j}{n(n+1)/2}$$

- Calcular el p-value para el estadístico F_Q usando la distribución $F((k-1),(k-1)(n-1))$. Si la hipótesis nula resulta rechazada, esto quiere decir que la significancia de los clasificadores en cada conjunto es diferente.

El test de Quade comprueba si la suma de rankings ponderados S_j son significativamente diferentes de 0.

CAPITULO 5

Arquitectura Propuesta para VisTHAA

Se plantea que la metodología de trabajo se aplique de manera creciente, y no secuencial. Por pasos se irán agregando componentes hasta llegar a integrar toda la herramienta, por ejemplo, no es necesario hacer el análisis de todas las caracterizaciones para empezar a integrar alguna de ellas en la herramienta final. La metodología propuesta consta de los siguientes pasos:

1. Revisión y estudio de fundamentos:
    - Caracterización del proceso de optimización.
    - Establecimiento de relaciones de desempeño.
    - Visualización de caracterizaciones y relaciones de desempeño.
2. Revisión y puesta en marcha de implementaciones del estado del arte (ejecución de las implementaciones).
3. Analizar, seleccionar y adaptar métodos de caracterización del proceso algorítmico
4. Desarrollar métodos de visualización de caracterizaciones del proceso algorítmico seleccionadas.
5. Construcción incremental de una herramienta de diagnóstico visual.
6. Diagnóstico visual del algoritmos GGA-BP aplicado a Bin Packing.
7. Rediseño de algoritmos.
8. Experimentación final.

En la figura 4.1 se muestra el diagrama del proceso de desarrollo, destacando el área en que se trabajará y la influencia que tendrá sobre el proceso de desarrollo de algoritmos.

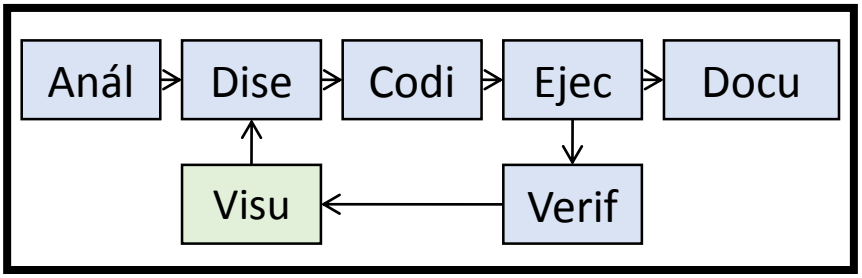

Figura 4.1 Diagrama del Proceso de Desarrollo

4.1 Construcción de la herramienta de diagnóstico visual

Para comenzar la construcción de la herramienta de diagnóstico visual, la cual se ha nombrado VisTHAA (Visualization Tool for Heuristic Algorithm Analysis), primero se hizo una planeación de lo que llegará a ser la herramienta, en la Figura 4.2 se muestra un esquema de VisTHAA, con sus módulos y de manera general se muestra lo que le ofrece al usuario. En este proyecto se sentará las bases de lo que será VisTHAA, debido a que es un proyecto muy amplio solo se abordaran algunos módulos básicos. De manera específica, en este proyecto se trabajará en los módulos de Caracterización Estructural de Instancias y Problemas, Caracterización del Espacio de Búsqueda, Caracterización del Desempeño y parte del Análisis comparativo de algoritmos. Cabe mencionar, que VisTHAA está planeado para ser una herramienta de análisis fuera de línea, es decir, no tendrá interacción directa con ejecuciones de los algoritmos, solo se solicitará resultados de ejecuciones en archivos.

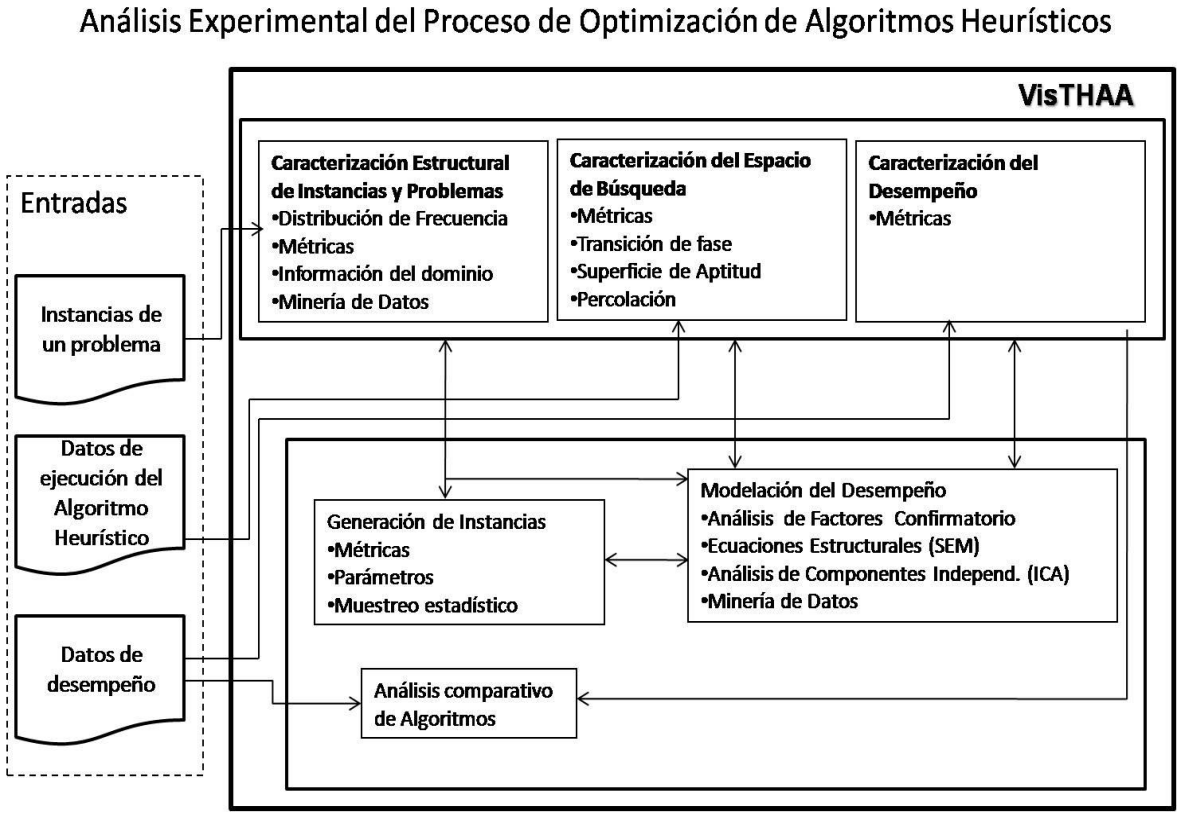

Figura 4.2 Esquema de módulos de VisTHAA

VisTHAA necesitará de datos de entradas, para ser procesados con ayuda de herramientas de libre distribución y se apoyará en técnicas de visualización para ofrecer al usuario visualizaciones e información que le servirán en el proceso del diseño de metaheurísticos. Esta interacción se muestra en la Figura 4.3.

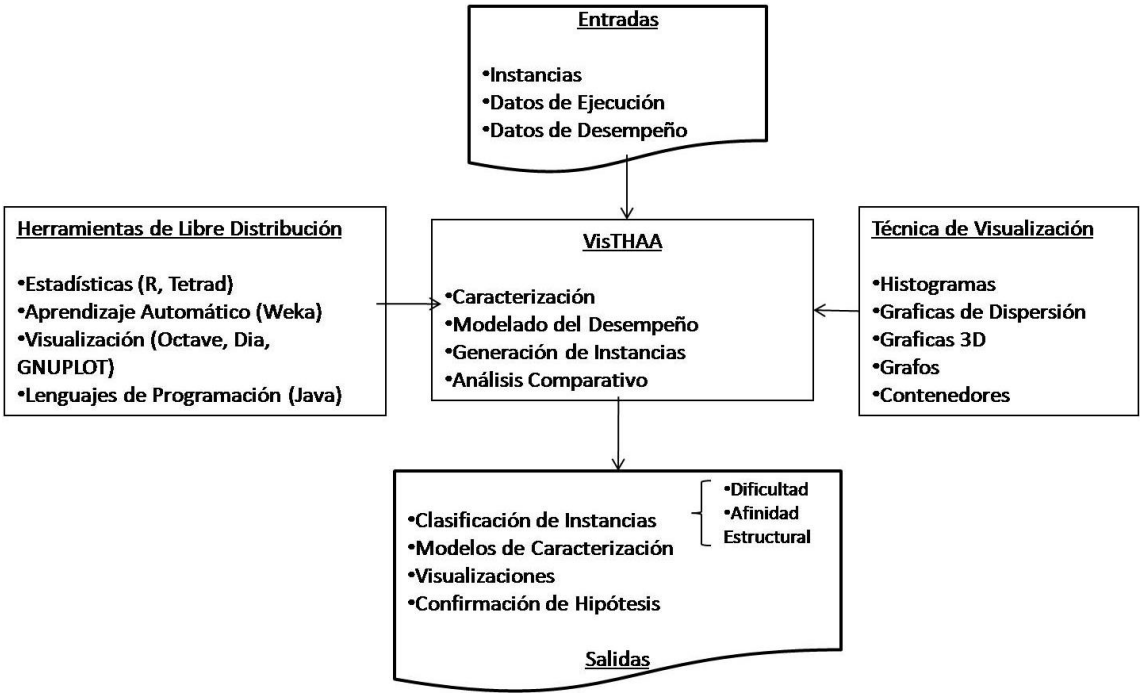

Figura 4.3 Herramientas para el Análisis Experimental del Proceso de Optimización de Algoritmos Heurísticos.

De manera inicial se ha hecho un diseño de la interfaz de usuario de VisTHAA, los módulos diseñados se describen a continuación.

Caracterización Estructural de Instancias y Problemas

Este módulo le permitirá al usuario cargar un archivo el cual contendrá el nombre de las instancias o instancia que se desea analizar. Ya especificadas las instancias, el usuario podrá seleccionar los índices o métricas que estén disponibles para el tipo de instancia, también tendrá la opción de seleccionar si desea visualizar la métrica seleccionada. El primer diseño de este modulo se ve en la Figura 4.4.

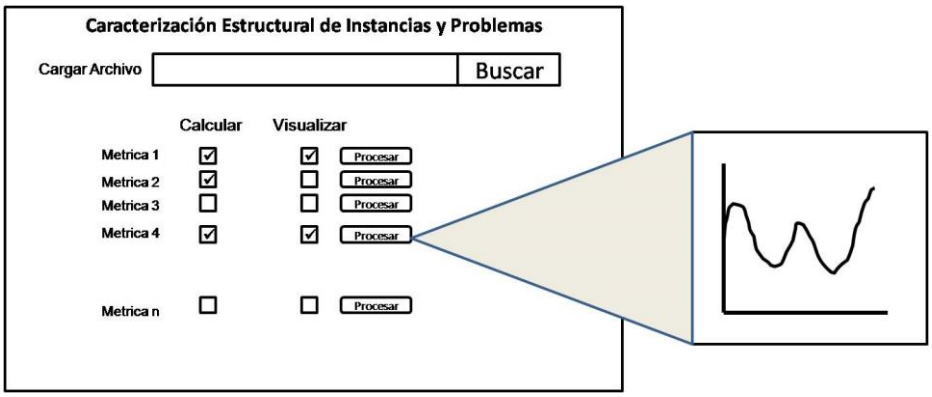

Figura 4. 4 Caracterización Estructural de Instancias y Problemas

Caracterización del Espacio de Búsqueda

Este módulo le permitirá al usuario cargar un archivo que contiene información del proceso de búsqueda, este archivo de la estructura de la entrada aun no se ha diseñado. De igual manera que en el módulo de instancias, el usuario podrá seleccionar índices que desee calcular y los que desee analizar. Otra funcionalidad que se desea incorporar a este módulo, es la visualización del comportamiento del algoritmo en el espacio de búsqueda, con esto se puede ver que tan buena diversificación o intensificación posee el algoritmo, dando la posibilidad de moverse en diferentes puntos del proceso de búsqueda. El diseño de este módulo se muestra en la Figura 4.5.

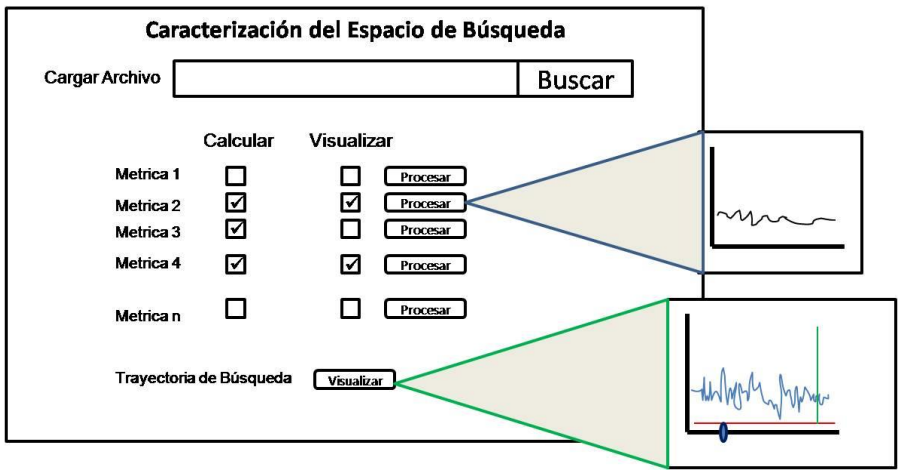

Figura 4.5 Caracterización del Espacio de Búsqueda

Caracterización del Desempeño

En este archivo se solicita al usuario que introduzca un archivo de datos relacionados con el desempeño del algoritmo, con estos datos se podrá calcular las métricas de desempeño que ofrecerá VisTHAA, aquí mismo se quiere introducir la disposición de pruebas no paramétricas. En la Figura 4.6 se muestra el diseño de este módulo.

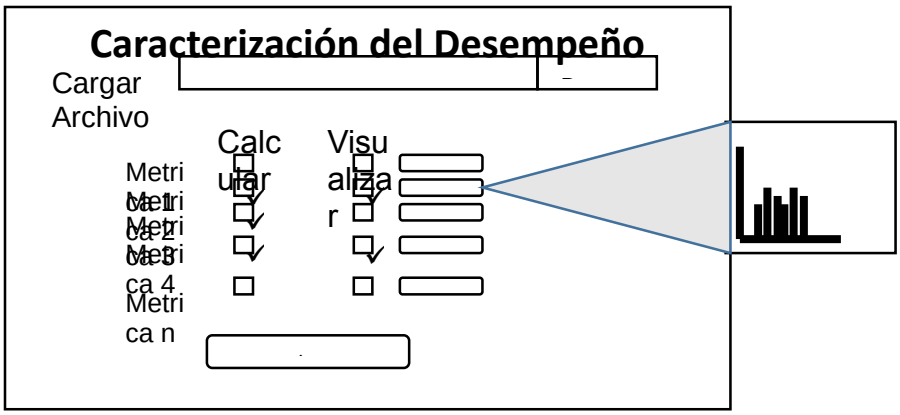

Figura 4.6 Caracterización del Desempeño

Se desea agregar la opción de editar las etiquetas que se presenten en las visualizaciones, para que el usuario lo maneje a su conveniencia.

De los módulos descritos anteriormente se ha empezado a desarrollar el de Caracterización Estructural de Instancias y Problemas. Para este módulo se ha hecho el cálculo de métricas y la lectura de instancias.

En la lectura de instancias es necesario mencionar que ya se codificó de tal manera que pueda leer todo tipo de instancias. Aun se tiene problemas en algunos casos, se está haciendo la corrección para que sea totalmente funcional.

Para iniciar el proceso de desarrollo de visualización se tiene que partir del cálculo de las métricas, es por esto que hasta este momento se han codificado métodos para el cálculo de métricas. Posteriormente se irán desarrollando las visualizaciones para cada métrica.

Para iniciar con el proceso de construcción de la herramienta se hizo una revisión de plataformas y complementos de desarrollo. Además se hizo el diseño de la interfaz de usuario y se inició con la implementación.

La interfaz gráfica de usuario, como base de la herramienta de análisis de algoritmos metaheurísticos se construyó utilizando el lenguaje Java. Se ha incorporado el uso de una librería de libre distribución para Java llamada JFreeChart. La cual contiene una amplia posibilidad de generación de gráficas y la utilidad de generar y guardar los resultados obtenidos.

Se ha hecho la selección de las métricas para la caracterización de la entrada y ya están codificadas de manera modular para que sean incorporadas posteriormente en la herramienta.

Para el proceso de visualización se hizo la revisión de tutoriales para aprender el manejo de la librería JFreeChart y se han hecho visualizaciones de algunas métricas. Cabe mencionar que todos los avances que se han hecho en cuestión de programación se están haciendo de manera modular, con el fin de que los módulos puedan ser reutilizados en otros procesos de cálculo o visualización.

Durante un rediseño de la herramienta VisTHAA, se ha propuesto que permita a futuros investigadores la incorporación de nuevos métodos, es por esto que se ha tomado un tiempo considerable para hacer un análisis de cómo se podrán incorporar nuevos métodos.

En la Figura 4.7 se muestra un primer prototipo de la interfaz de usuario, haciendo más énfasis en el aspecto funcional. En esta imagen se puede ver el menú principal, con los tres apartados: Caracterización del problema, Caracterización de la búsqueda y Caracterización del desempeño.

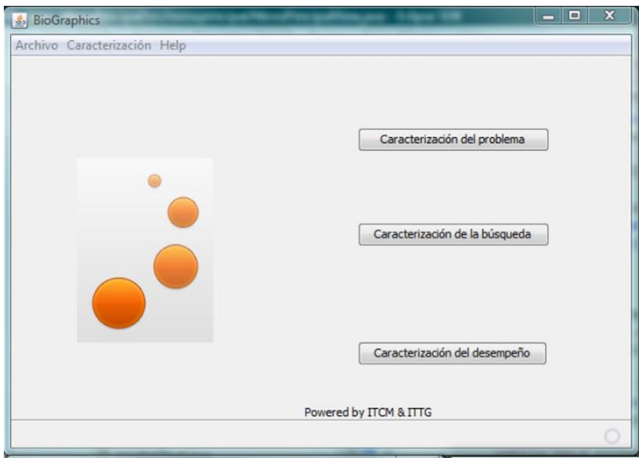

Figura 4.7 Prototipo del Menú Principal de VisTHAA

En la ventana de Caracterización del Problema , se le permite al usuario buscar el archivo de instancia que se va analizar, posteriormente puede seleccionar las métricas disponibles y posteriormente aparecerá un cuadro de diálogo donde se le permite al usuario activar si desea ver el resultado de los cálculos de la métrica así como la visualización de esa métrica.

En la Figura 4.8 se muestra un ejemplo del cálculo de una métrica y su gráfica. Se brinda al usuario la opción de guardar la información mostrada en el área de texto así como el gráfico generado por la herramienta, esto se hizo pensando en los usos que le puede dar a esta información posteriormente.

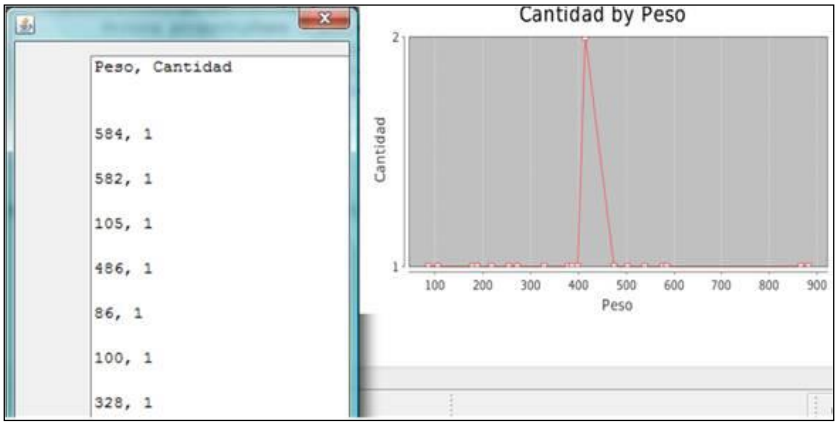

Figura 4.8 Ejemplo de la salida del cálculo de una métrica y su visualización.

4.2 Diseño de un metadato para describir las entradas del sistema

Este proyecto está planeado para que sea posible el análisis de diferentes instancias de problemas y de igual manera se desea que sea posible la lectura de los resultados del proceso de análisis de algoritmos metaheurísticos. Sin tener la necesidad de modificar el proceso de lectura de datos.

Debido a que no existe un estándar para la representación de instancias de optimización, resultados y opciones de solucionadores (Fourer R., 2008), se ha diseñado un metadato para poder describir en forma de texto la estructura de del contenido de instancias y datos que servirán para el proceso de análisis.

Metadato es un término utilizado para describir datos que dan el tipo y clase de la información, es decir, son datos acerca de datos, que proveen la información necesaria para que los datos puedan ser empleados ágilmente en diferentes aplicaciones (López, 2000).

Este diseño servirá para que el usuario describa el contenido de su archivo de instancias, dando la posibilidad de introducir sólo una instancia a la vez o un listado de todas las instancias que desea evaluar.

Esta idea se puede extender en trabajos futuros a la lectura de datos de otras entradas, por ejemplo, los datos correspondientes a la ejecución de algoritmos metaheurísticos.

4.3 Diseño de una estructura de datos genérica para las entradas del sistema

Con el diseño del metadato, se creó el diseño un metalenguaje para manejar el procesamiento de los datos.

Mediante un metalenguaje o sistema formal, se añade información o codificación a la forma digital de un documento, ya sea para controlar su procesamiento o para representar su significado (Lamarca, 2009)

Tomando esta idea, se propuso una estructura de datos para el almacenamiento eficiente de instancias de optimización genéricas, la cual será usada por el metalenguaje.

4.4 Lenguaje de Definición de Datos Propuesto para VisTHAA

Durante el desarrollo de la herramienta se observó que existe una gran cantidad de formatos para los archivos involucrados en el proceso de optimización (por ejemplo archivos de instancias). Considerando que los investigadores difícilmente podrían cambiar sus formatos, se decidió incorporar un procedimiento que permita la lectura de instancias con diferentes formatos y para diferentes problemas. Este procedimiento, denominado VisTHAA_Parser, que fue desarrollado de manera colaborativa, incluye: un metadato, un método de procesamiento y una estructura de datos. El procedimiento podría ser extendido para otro tipo de entradas.

4.4.1 Metadato

Metadato es un término utilizado para describir datos que dan el tipo y clase de la información, es decir, son datos acerca de datos, que proveen la información necesaria para que los datos puedan ser empleados ágilmente en diferentes aplicaciones.

El metadato diseñado para VisTHAA_Parser describe el formato de las instancias. El método de procesamiento de datos convierte las instancias introducidas con cualquier formato a un

formato único reconocido por VisTHAA, para ello utiliza el metadato asociado a la instancia. Este metadato servirá para que el usuario describa el contenido de su archivo de instancias, dando la posibilidad de introducir sólo una instancia a la vez o un listado de todas las instancias que desea evaluar.

4.4.2 Estructura de Datos

Para el almacenamiento de los datos fue necesario el diseño de una estructura de datos genérica que permita el acceso y almacenamiento eficiente de los datos leídos.

Mientras se hace el procesamiento de los datos, de acuerdo a las instrucciones dadas en el archivo metadato, por cada variable especificada, se crea un espacio exacto de memoria para el contenido de esa variable. Debido a que la estructura de datos es dinámica y de acceso directo, ésta se considera eficiente.

4.4.3 Modelo de Datos para Bin Packing Problem

En el ejemplo de la Figura 4.9, se muestran los archivos involucrados en el procesamiento de datos requerido para posterior solución de instancias del problema clásico de empacado de objetos en contenedores (Bin Packing Problem, BPP). La Figura 5.1.a corresponde al archivo de procesamiento por lotes, al que denominaremos *batch*. En éste se describe el número de instancias que serán procesadas con el archivo metadato de la Figura 5.1.b. El archivo metadato describe el formato de cómo se debe leer cada archivo de instancias de la Figura 5.1.c.

a) Archivo Batch b) Archivo Metadato c) Instancia BPP

Figura 4.9 Archivos involucrados en el Procesamiento de Datos

CAPITULO 6

Experimentación y Resultados

5.1 Prueba de Friedman

Se hizo la prueba de Friedman usando los datos del artículo de Derrac [Derrac 2012], para comparar los resultados obtenidos en VisTHAA y verificar que la herramienta está haciendo correctamente la prueba. En la imagen 5.1, se muestran los datos de entrada tomados de Derrac 2012, en esta imagen se muestra una tabla que resume lo obtenido en el proceso de la prueba.

Algoritmo	PSO	SSGA	SS-BLX	DE-EXP
F1	1,23E-01 (3)	8,42E-06 (2)	3,40E+02 (4)	8,26E-06 (1)
F2	2,60E+01 (3)	8,72E-02 (2)	1,73E+03 (4)	8,18E-06 (1)
F3	5,17E+07 (2)	7,95E+07 (3)	1,84E+08 (4)	9,94E+04 (1)
F4	2,49E+03 (3)	2,59E+00 (2)	6,23E+03 (4)	8,35E-06 (1)
F5	4,10E+05 (4)	1,34E+05 (3)	2,19E+03 (2)	8,51E-06 (1)
F6	7,31E+05 (4)	6,17E+03 (2)	1,15E+05 (3)	8,39E-06 (1)
F7	2,68E+02 (1)	1,27E+06 (2,5)	1,97E+06 (4)	1,27E+06 (2,5)
F8	2,04E+04 (2,5)	2,04E+04 (2,5)	2,04E+04 (2,5)	2,04E+04 (2,5)
F9	1,44E+04 (4)	7,29E-06 (1)	4,20E+03 (3)	8,15E-06 (2)
F10	1,40E+04 (3)	1,71E+04 (4)	1,24E+04 (2)	1,12E+04 (1)
F11	5,59E+03 (4)	3,26E+03 (3)	2,93E+03 (2)	2,07E+03 (1)
F12	6,36E+05 (4)	2,79E+05 (3)	1,51E+05 (2)	6,31E+04 (1)
F13	1,50E+03 (4)	6,71E+02 (3)	3,25E+02 (1)	6,40E+02 (2)
F14	3,30E+03 (4)	2,26E+03 (1)	2,80E+03 (2)	3,16E+03 (3)
F15	3,40E+05 (4)	2,92E+05 (2)	1,14E+05 (1)	2,94E+05 (3)
F16	1,33E+05 (4)	1,05E+05 (2)	1,04E+05 (1)	1,13E+05 (3)
F17	1,50E+05 (4)	1,19E+05 (2)	1,18E+05 (1)	1,31E+05 (3)
F18	8,51E+05 (4)	8,06E+05 (3)	7,67E+05 (2)	4,48E+05 (1)
F19	8,50E+05 (3)	8,90E+05 (4)	7,56E+05 (2)	4,34E+05 (1)
F20	8,51E+05 (3)	8,89E+05 (4)	7,46E+05 (2)	4,19E+05 (1)
F21	9,14E+05 (4)	8,52E+05 (3)	4,85E+05 (1)	5,42E+05 (2)
F22	8,07E+05 (4)	7,52E+05 (2)	6,83E+05 (1)	7,72E+05 (3)
F23	1,03E+06 (4)	1,00E+06 (3)	5,74E+05 (1)	5,82E+05 (2)
F24	4,12E+05 (4)	2,36E+05 (2)	2,51E+05 (3)	2,02E+05 (1)
F25	5,10E+05 (1)	1,75E+06 (3)	1,79E+06 (4)	1,74E+06 (2)
R_j	3,38	2,56	2,34	1,72

Figura 5.1 Rankings del test de Friedman, Derrac 2012

A continuación, en la Figura 5.2 se muestra el resultado obtenido por VisTHAA para estos mismos datos:

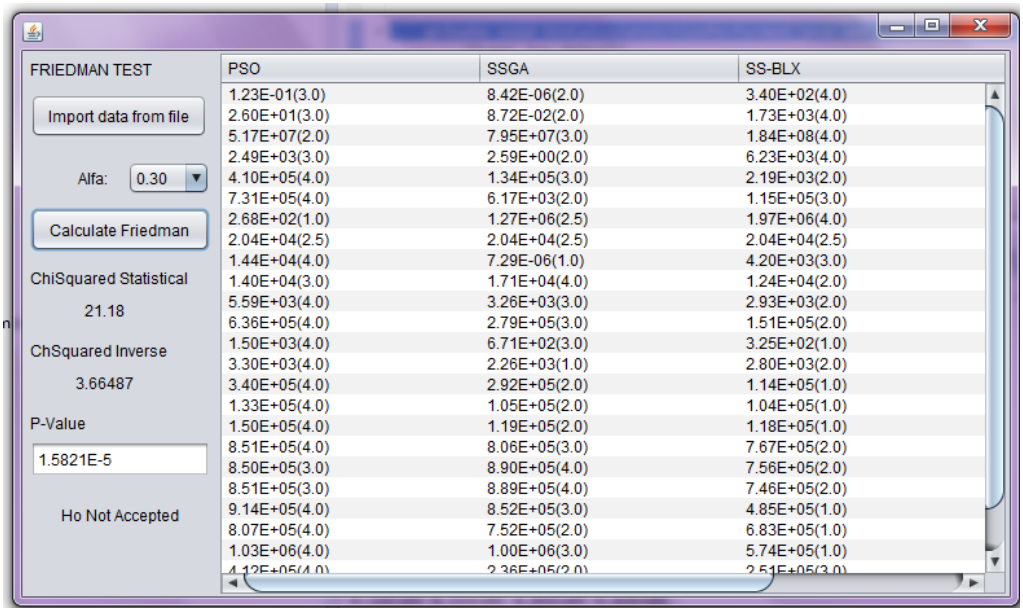

Figura 5. 2 Resultados en VisTHAA de prueba de Friedman con datos de Derrac 2012

Con esto se puede comprobar que el rankeo coincide y los resultados también.

5.2 Prueba Wilcoxon

Esta prueba se aplicó a datos obtenidos de dos algoritmos para resolver el probema de empacado de objetos, a continuación, en la Figura 5.3 se muestran los resultados de la prueba:

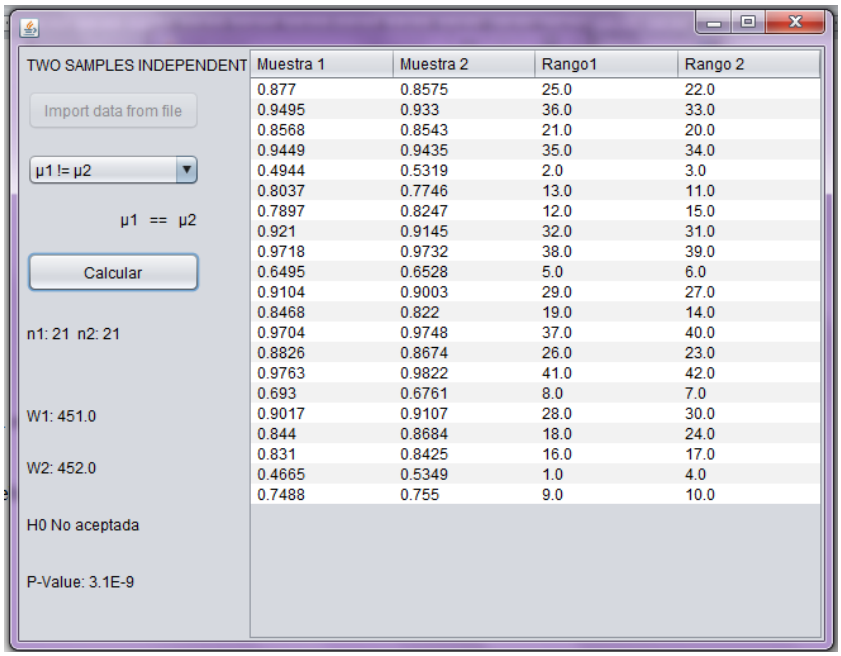

Figura 5.3 Prueba de Wilcoxon en VisTHAA para 2 algoritmos que resuelven BPP

CAPÍTULO 7

Conclusiones y Trabajo Futuro

VisTHAA es una herramienta que se encuentra en crecimiento, hasta éste momento provee herramientas de análisis para el desempeño durante el proceso de optimización; la herramienta facilita el manejo de instancias genéricas, estadísticas y caracterización visual del proceso de estrategias heurísticas. En el proceso de crecimiento de VisTHAA, la aportación de este trabajo se centra en el área de análisis comparativo de algoritmos, para ésto se ha introducido la automatización del uso de pruebas no-paramétricas dada la naturaleza de los datos.

Como trabajo futuro para enriquecer esta herramienta, es necesario mejorar la apariencia visual para hacerla más accesible para el usuario, también es necesario añadir secciones de ayuda.

Referencias

(HEAL) Heuristic and Evolutionary Algorithms Laboratory HeuristicLab [En línea]. - http://dev.heuristiclab.com/trac/hl/core/wiki.

Álvarez V. "Modelo para Representar la Complejidad del Problema y el Desempeño de Algoritmos" [Informe] : Tesis de Maestría / Instituto Tecnológico de Ciudad Madero. - Ciudad Madero, Tam : [s.n.], 2006.

Alvim A. [y otros] "A Hybrid Improvement Heuristic for the one-dimensional Bin Packing Problem" [Publicación periódica] // Journal of Heuristics 10. - 2004. - págs. 205-229.

Barr R. Golden B., Kelly J., Resendez M., Stewart W. Designing and Reporting on Computational Experiments with Heurístic Methods [Publicación periódica] // Journal of Heuristics. - 1995. - págs. 19-32.

Blum C. y Roli A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. [Publicación periódica] // ACM Computing Surveys. - 2003. - págs. 268–308.

Brownlee J. Optimization Algorithm Toolkit [En línea]. - 2006 - 2008. - http://optalgtoolkit.sourceforge.net/.

Cowling P., Kendall G. y Soubeiga E. A Hyperheuristic Approach to Scheduling a Sales Summit. [Conferencia] // Practice and Theory of Automated Timetabling III : Third International Conference. - Konstanz, Germany : Springer, Lecture Notes in Computer Science, 2000. - Vol. 2079. - págs. 176-190.

Cruz L. Automatización del Diseño de la Fragmentación Vertical y Ubicación en Bases de Datos Distribuidas usando Métodos Heurísticos y Exactos [Informe] : Tesis de Maestría / Universidad Virtual del Campus Tampico del Instituto Tecnológico y de Estudios Superiores de Monterrey. - Altamira, Tam : [s.n.], 1999.

Cruz L. Clasificación de Algoritmos Heurísticos Para la Solución de Problemas de Bin Packing [Informe] : Tesis Doctoral / Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET). - Cuernavaca, México. : [s.n.], 2004.

Díaz A. y Glover F. Optimización Heurística y Redes Neuronales [Informe]. - España : Paraninfo, 1996.

Duarte A., Pantrigo J. y Gallego M. Metaheurísticas [Informe] / Universidad Rey Juan Carlos. - Madrid, España. : Dykinson, 2007. - págs. 25-44.

Europe Inria Lille - Nord ParadisEO A Software Framework for Metaheuristics [En línea]. - http://paradiseo.gforge.inria.fr/index.php?n=Main.HomePage.

Fernández J., García S. y Derrac J. KEEL (Knowledge Extraction based on Evolutionary Learning) [En línea]. - 2004-2013. - http://www.keel.es/.

Fourer R. Ma J. Optimization Services: A Framework for Distributed Optimization [Informe] / Department of Industrial Engineering and Management Sciences ; Northwestern University. - Evanston, Illinois, USA. : [s.n.], 2008.

Garey M. R. y Jonson D. S. Computers and Intractability, a Guide to the Theory of NP-completeness [Informe]. - New York, USA. : W. H. Freeman and Company, 1979.

Glover F. Future Paths for Integer Programming and Links to Artificial Intelligence [Informe] / University of Colorado. - [s.l.] : Center for Applied Artificial Intelligence, Graduate School of Business, 1986.

Goldberg D. The Design of Innovation (Genetic Algorithms and Evolutionary Computation) [Sección de libro]. - [s.l.] : Springer 1 ed, 2002. - 1 ed.

Halim S. World of Seven - Viz: SLS Engineering Suite [En línea]. - 2000 - 2013. - http://www.comp.nus.edu.sg/~stevenha/viz/index.html.

Halim S. World of Seven, V-MDF [En línea]. - 2000 - 2013. - http://www.comp.nus.edu.sg/~stevenha/v-mdf/.

Halim S., Lau H. y Wan W. Tuning Tabu Search Strategies via Visual Diagnosis [Conferencia] // Metaheuristics International Conference (MIC 2005). - Vienna, Austria : [s.n.], 2005. - págs. 630-636.

Halim S., Yap R. y Lau H. An Integrated White+Black Box Approach for Designing and Tuning Stochastic Local Search [Conferencia] // Principles and Practice of Constraint Programming. - Rhode Island, United States of America : [s.n.], 2007. - págs. 332-347.

INRIA Guimoo (a Graphical User Interface for Multi Objective Optimization) [En línea] / ed. (INRIA) French National Institute for Research in Computer Science and Control. - 2005. - http://guimoo.gforge.inria.fr/.

Kendall G. y Mohd N. An Investigation of a Tabu-Search-Based Hyper-Heuristic for Examination Timetabling [Informe] / Automated Scheduling, Optimisation and Planning (ASAP) Research Group ; School of Computer Science and Information Technology, Universiry of Nottingham. - Nottingham : [s.n.], 2005.

Lamarca M. Hipertexto, el nuevo concepto de documento en la cultura de la imagen [Informe] : Tesis Doctoral / Universidad Complutense de Madrid. - España : [s.n.], 2009.

Liefooghe A., Jourdan L. y Talbi E. A Unified Model for Evolutionary Multiobjective Optimization and its Implementation in a General Purpose Software Framework: ParadisEO-MOEO [Informe] / Institut National de Recherche en Informatique et en Automatique. - 2009.

López C. Modelo para el Desarrollo de Bibliotecas Digitales Especializadas [Informe]. - 2000.

McGeoch C. Analysis of algorithms by simulation: variance reduction techniques and simulation speedups [Publicación periódica] // ACM Computing Surveys. - 1992. - págs. 195- 212.

Nieto D. "Hibridación de Algoritmos Metaheurísticos para Problemas de Bin Packing" [Informe] : Tesis de Maestría / Instituto Tecnológico de Ciudad Madero. - Ciudad Madero, Tam : [s.n.], 2007.

O'Brien R. Ant Algorithm Hyperheuristic Approaches for Scheduling Problems [Publicación periódica] // The University of Nottingham. - 2008.

Özcan E., Bilgin B. y Korkmaz E. A Comprehensive Analysis of Hyper-heuristics [Informe] / Department of Computer Engineering, Yeditepe University.. - 2008.

Pérez V Modelado causal del desempeño de algoritmos metaheurísticos en problemas de distribución de objetos [Informe] : Tesis de maestría / Instituto Tecnológico de Ciudad Madero. - Ciudad Madero, Tam., México : [s.n.], 2007.

Quiroz M. "Caracterización de Factores de Desempeño de Algorítmos de Solución de BPP" [Informe] : Tesis de Maestría. - Instituto Tecnológico de Ciudad Madero : [s.n.], 2009.

Anexo A

Demostración de complejidad del Problema de Empacado de Objetos en Contenedores

Para demostrar que el BPP es NP-duro, primeramente se verifica que es NP:

1. Se debe construir una máquina no deterministas, que genere aleatoriamente, en tiempo polinomial, una solución candidata en tiempo polinomial.

Una solución candidata, para el BPP, es cualquier asignación aleatoria a cada objeto de uno contenedor, lo cual se realiza en $\theta(n)$ pasos.

2. Después se debe validar la factibilidad de esta solución, en tiempo polinomial.

La verificación de la restricción de capacidad de los contenedores se realiza en $\theta(n)$ pasos, sumando los pesos de los objetos de cada contenedor

Con lo anterior queda demostrado que el BPP pertenece a la clase de problemas NP, ahora se demostrará que es NP-completo por medio de la reducción a BPP del problema de Partición, el cual en (Garey, y otros, 1979) se demuestra que es NP-completo. Para esto primeramente definiremos el problema de Partición (PP) como sigue:

Dado una secuencia de enteros $a_1, a_2, \ldots, a_m$. Determinar si hay una partición de enteros en dos subconjuntos, tal que la suma de los elementos de un subconjunto es igual a la suma del

otro subconjunto. Formalmente, determinar si existe $I \subseteq \{1, 2,..., m\}$, tal que $\sum_{i \in I} a_i = \sum_{i=1}^{m} a_i / 2$

.

La versión de decisión del BBP es: Dado un conjunto de objetos de tamaño w_1, w_2,…w_n, un número ilimitado de contenedores de tamaño *c*, y un parámetro *k*. Determinar si es posible empacar todos los objetos en *k* contenedores.

La reducción de cualquier caso del PP a BPP, cuya función de transformación se muestra en la Tabla A.1, consiste en: Dado un conjunto de objetos de tamaño a_1, a_2,…a_m, construir un caso de BPP con objetos (w_i) del mismo tamaño a los de PP (a_i) y un contenedor de tamaño igual a la mitad del la suma de los tamaños de los objetos. Sea *k* igual a 2, hay una solución para BPP que use dos contenedores, si y solo si, hay una solución para el problema de Partición, es decir Un caso "Si" de PP es un caso "Si" de BPP, ya que $\sum_{i \in I} a_i = \sum_{i \in B_1} w_i = c$ y $\sum_{i \in I'} a_i = \sum_{i \in B_2} w_i = c$. Esta reducción se realiza en θ(*n*) pasos.

Función de Reducción	Tiempo
$n = m$	1
$W = A$	n
$c = \sum_{i=1}^{m} a_i / 2$	n
$k = 2$	1
$B_1 = I$	n
$B_2 = I' = \{1, 2, ..., m\} - I$	

Tabla A.1 Función de transformación de PP a BPP

Con lo anterior, se demuestra que el BPP es NP-duro, ya que su problema de decisión asociado es NP-Completo (Garey, y otros, 1979).

Anexo B

Tabla B.1 Valores Críticos para la prueba de signos de dos colas en el que α=0.05 y α=0.1

#Cases	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$\alpha = 0.05$	5	6	7	7	8	9	9	10	10	11	12	12	13	13	14	15	15	16	17	18	18
$\alpha = 0.1$	5	6	6	7	7	8	9	9	10	10	11	12	12	13	13	14	14	15	16	16	17

Tabla B.2 Valores críticos de Rj mínimo para comparaciones de m=k-1 algoritmos contra un control en n conjunto de datos.

n	Level of significance	*m* = 2	*m* = 3	*m* = 4	*m* = 5	*m* = 6	*m* = 7	*m* = 8	*m* = 9
5	0.1	0	0	–	–	–	–	–	–
	0.05	–	–	–	–	–	–	–	–
6	0.1	0	0	0	0	0	–	–	–
	0.05	0	0	–	–	–	–	–	–
7	0.1	0	0	0	0	0	0	0	0
	0.05	0	0	0	0	–	–	–	–
8	0.1	1	1	0	0	0	0	0	0
	0.05	0	0	0	0	0	0	0	0
9	0.1	1	1	1	1	0	0	0	0
	0.05	1	0	0	0	0	0	0	0
10	0.1	1	1	1	1	1	1	1	1
	0.05	1	1	1	0	0	0	0	0
11	0.1	2	2	1	1	1	1	1	1
	0.05	1	1	1	1	1	1	0	0
12	0.1	2	2	2	2	1	1	1	1
	0.05	2	1	1	1	1	1	1	1
13	0.1	3	2	2	2	2	2	2	2
	0.05	2	2	2	1	1	1	1	1
14	0.1	3	3	2	2	2	2	2	2
	0.05	2	2	2	2	2	2	1	1
15	0.1	3	3	3	3	3	2	2	2
	0.05	3	3	2	2	2	2	2	2
16	0.1	4	3	3	3	3	3	3	3
	0.05	3	3	3	3	2	2	2	2
17	0.1	4	4	4	3	3	3	3	3
	0.05	4	3	3	3	3	3	2	2
18	0.1	5	4	4	4	4	4	3	3
	0.05	4	4	3	3	3	3	3	3
19	0.1	5	5	4	4	4	4	4	4
	0.05	4	4	4	4	3	3	3	3
20	0.1	5	5	5	5	4	4	4	4
	0.05	5	4	4	4	4	4	3	3
21	0.1	6	5	5	5	5	5	5	5
	0.05	5	5	5	4	4	4	4	4
22	0.1	6	6	6	5	5	5	5	5
	0.05	6	5	5	5	4	4	4	4
23	0.1	7	6	6	6	6	5	5	5
	0.05	6	6	5	5	5	5	5	5
24	0.1	7	7	6	6	6	6	6	6
	0.05	6	6	6	5	5	5	5	5
25	0.1	7	7	7	7	6	6	6	6
	0.05	7	6	6	6	6	6	5	5
30	0.1	10	9	9	9	8	8	8	8
	0.05	9	8	8	8	8	8	7	7
35	0.1	12	11	11	11	10	10	10	10
	0.05	11	10	10	10	10	9	9	9
40	0.1	14	13	13	13	13	12	12	12
	0.05	13	12	12	12	12	11	11	11
45	0.1	16	16	15	15	15	14	14	14
	0.05	15	14	14	14	14	13	13	13
50	0.1	18	18	17	17	17	17	16	16
	0.05	17	17	16	16	16	16	15	15

Tabla B.3 Tabla de la Distribución F

F Values for $\alpha = 0.10$

	d_1								
d_2	1	2	3	4	5	6	7	8	9
1	39.86	49.5	53.59	55.83	57.24	58.2	58.91	59.44	59.86
2	8.53	9.00	9.16	9.24	9.29	9.33	9.35	9.37	9.38
3	5.54	5.46	5.39	5.34	5.31	5.28	5.27	5.25	5.24
4	4.54	4.32	4.19	4.11	4.05	4.01	3.98	3.95	3.94
5	4.06	3.78	3.62	3.52	3.45	3.40	3.37	3.34	3.32
6	3.78	3.46	3.29	3.18	3.11	3.05	3.01	2.98	2.96
7	3.59	3.26	3.07	2.96	2.88	2.83	2.78	2.75	2.72
8	3.46	3.11	2.92	2.81	2.73	2.67	2.62	2.59	2.56
9	3.36	3.01	2.81	2.69	2.61	2.55	2.51	2.47	2.44
10	3.29	2.92	2.73	2.61	2.52	2.46	2.41	2.38	2.35
11	3.23	2.86	2.66	2.54	2.45	2.39	2.34	2.3	2.27
12	3.18	2.81	2.61	2.48	2.39	2.33	2.28	2.24	2.21
13	3.14	2.76	2.56	2.43	2.35	2.28	2.23	2.20	2.16
14	3.10	2.73	2.52	2.39	2.31	2.24	2.19	2.15	2.12
15	3.07	2.70	2.49	2.36	2.27	2.21	2.16	2.12	2.09
16	3.05	2.67	2.46	2.33	2.24	2.18	2.13	2.09	2.06
17	3.03	2.64	2.44	2.31	2.22	2.15	2.10	2.06	2.03
18	3.01	2.62	2.42	2.29	2.20	2.13	2.08	2.04	2.00
19	2.99	2.61	2.40	2.27	2.18	2.11	2.06	2.02	1.98
20	2.97	2.59	2.38	2.25	2.16	2.09	2.04	2.00	1.96
21	2.96	2.57	2.36	2.23	2.14	2.08	2.02	1.98	1.95
22	2.95	2.56	2.35	2.22	2.13	2.06	2.01	1.97	1.93
23	2.94	2.55	2.34	2.21	2.11	2.05	1.99	1.95	1.92
24	2.93	2.54	2.33	2.19	2.10	2.04	1.98	1.94	1.91
25	2.92	2.53	2.32	2.18	2.09	2.02	1.97	1.93	1.89
26	2.91	2.52	2.31	2.17	2.08	2.01	1.96	1.92	1.88
27	2.90	2.51	2.30	2.17	2.07	2.00	1.95	1.91	1.87
28	2.89	2.50	2.29	2.16	2.06	2.00	1.94	1.90	1.87
29	2.89	2.50	2.28	2.15	2.06	1.99	1.93	1.89	1.86
30	2.88	2.49	2.28	2.14	2.05	1.98	1.93	1.88	1.85
40	2.84	2.44	2.23	2.09	2.00	1.93	1.87	1.83	1.79
60	2.79	2.39	2.18	2.04	1.95	1.87	1.82	1.77	1.74
120	2.75	2.35	2.13	1.99	1.90	1.82	1.77	1.72	1.68
inf	2.71	2.30	2.08	1.94	1.85	1.77	1.72	1.67	1.63

F Value for $\alpha = 0.10$

	d_1									
d_2	10	12	15	20	24	30	40	60	120	inf
1	60.19	60.71	61.22	61.74	62	62.26	62.53	62.79	63.06	63.33
2	9.39	9.41	9.42	9.44	9.45	9.46	9.47	9.47	9.48	9.49
3	5.23	5.22	5.20	5.18	5.18	5.17	5.16	5.15	5.14	5.13
4	3.92	3.90	3.87	3.84	3.83	3.82	3.80	3.79	3.78	3.76
5	3.30	3.27	3.24	3.21	3.19	3.17	3.16	3.14	3.12	3.10
6	2.94	2.90	2.87	2.84	2.82	2.80	2.78	2.76	2.74	2.72
7	2.70	2.67	2.63	2.59	2.58	2.56	2.54	2.51	2.49	2.47
8	2.54	2.50	2.46	2.42	2.40	2.38	2.36	2.34	2.32	2.29
9	2.42	2.38	2.34	2.30	2.28	2.25	2.23	2.21	2.18	2.16
10	2.32	2.28	2.24	2.20	2.18	2.16	2.13	2.11	2.08	2.06
11	2.25	2.21	2.17	2.12	2.10	2.08	2.05	2.03	2.00	1.97
12	2.19	2.15	2.10	2.06	2.04	2.01	1.99	1.96	1.93	1.90
13	2.40	2.10	2.05	2.01	1.98	1.96	1.93	1.90	1.88	1.85
14	2.10	2.05	2.01	1.96	1.94	1.91	1.89	1.86	1.83	1.80
15	2.06	2.02	1.97	1.92	1.90	1.87	1.85	1.82	1.79	1.76
16	2.03	1.99	1.94	1.89	1.87	1.84	1.81	1.78	1.75	1.72
17	2.00	1.96	1.91	1.86	1.84	1.81	1.78	1.75	1.72	1.69
18	1.98	1.93	1.89	1.84	1.81	1.78	1.75	1.72	1.69	1.66
19	1.96	1.91	1.86	1.81	1.79	1.76	1.73	1.70	1.67	1.63
20	1.94	1.89	1.84	1.79	1.77	1.74	1.71	1.68	1.64	1.61
21	1.92	1.87	1.83	1.78	1.75	1.72	1.69	1.66	1.62	1.59
22	1.90	1.86	1.81	1.76	1.73	1.70	1.67	1.64	1.60	1.57
23	1.89	1.84	1.80	1.74	1.72	1.69	1.66	1.62	1.59	1.55
24	1.88	1.83	1.78	1.73	1.70	1.67	1.64	1.61	1.57	1.53
25	1.87	1.82	1.77	1.72	1.69	1.66	1.63	1.59	1.56	1.52
26	1.86	1.81	1.76	1.71	1.80	1.65	1.61	1.58	1.54	1.50
27	1.85	1.80	1.75	1.70	1.67	1.64	1.60	1.57	1.53	1.49
28	1.84	1.79	1.74	1.69	1.66	1.63	1.59	1.56	1.52	1.48
29	1.83	1.78	1.73	1.68	1.65	1.62	1.58	1.55	1.51	1.47
30	1.82	1.77	1.72	1.67	1.64	1.61	1.57	1.54	1.50	1.46
40	1.76	1.71	1.66	1.61	1.57	1.54	1.51	1.47	1.42	1.38
60	1.71	1.66	1.60	1.54	1.51	1.48	1.44	1.40	1.35	1.29
120	1.65	1.60	1.55	1.48	1.45	1.41	1.37	1.32	1.26	1.19
inf	1.60	1.55	1.49	1.42	1.38	1.34	1.30	1.24	1.17	1.00

F Values for $\alpha = 0.05$

	d_1								
d_2	1	2	3	4	5	6	7	8	9
1	161.4	199.5	215.7	224.6	230.2	234.0	236.8	238.9	240.5
2	18.51	19.00	19.16	19.25	19.3	19.33	19.35	19.37	19.38
3	10.13	9.55	9.28	9.12	9.01	8.94	8.89	8.85	8.81
4	7.71	6.94	6.59	6.39	6.26	6.16	6.09	6.04	6.00
5	6.61	5.79	5.41	5.19	5.05	4.95	4.88	4.82	4.77
6	5.99	5.14	4.76	4.53	4.39	4.28	4.21	4.15	4.10
7	5.59	4.74	4.35	4.12	3.97	3.87	3.79	3.73	3.68
8	5.32	4.46	4.07	3.84	3.69	3.58	3.50	3.44	3.39
9	5.12	4.26	3.86	3.63	3.48	3.37	3.29	3.23	3.18
10	4.96	4.10	3.71	3.48	3.33	3.22	3.14	3.07	3.02
11	4.84	3.98	3.59	3.36	3.20	3.09	3.01	2.95	2.90
12	4.75	3.89	3.49	3.26	3.11	3.00	2.91	2.85	2.80
13	4.67	3.81	3.41	3.18	3.03	2.92	2.83	2.77	2.71
14	4.60	3.74	3.34	3.11	2.96	2.85	2.76	2.70	2.65
15	4.54	3.68	3.29	3.06	2.90	2.79	2.71	2.64	2.59
16	4.49	3.63	3.24	3.01	2.85	2.74	2.66	2.59	2.54
17	4.45	3.59	3.20	2.96	2.81	2.70	2.61	2.55	2.49
18	4.41	3.55	3.16	2.93	2.77	2.66	2.58	2.51	2.46
19	4.38	3.52	3.13	2.90	2.74	2.63	2.54	2.48	2.42
20	4.35	3.49	3.10	2.87	2.71	2.60	2.51	2.45	2.39
21	4.32	3.47	3.07	2.84	2.68	2.57	2.49	2.42	2.37
22	4.30	3.44	3.05	2.82	2.66	2.55	2.46	2.40	2.34
23	4.28	3.42	3.03	2.80	2.64	2.53	2.44	2.37	2.32
24	4.26	3.40	3.01	2.78	2.62	2.51	2.42	2.36	2.30
25	4.24	3.39	2.99	2.76	2.60	2.49	2.40	2.34	2.28
26	4.23	3.37	2.98	2.74	2.59	2.47	2.39	2.32	2.27
27	4.21	3.35	2.96	2.73	2.57	2.46	2.37	2.31	2.25
28	4.20	3.34	2.95	2.71	2.56	2.45	2.36	2.29	2.24
29	4.18	3.33	2.93	2.70	2.55	2.43	2.35	2.28	2.22
30	4.17	3.32	2.92	2.69	2.53	2.42	2.33	2.27	2.21
40	4.08	3.23	2.84	2.61	2.45	2.34	2.25	2.18	2.12
60	4.00	3.15	2.76	2.53	2.37	2.25	2.17	2.10	2.04
120	3.92	3.07	2.68	2.45	2.29	2.17	2.09	2.02	1.96
inf	3.84	3.00	2.60	2.37	2.21	2.10	2.01	1.94	1.88

F Values for $\alpha = 0.05$

d_2	d_1 10	12	15	20	24	30	40	60	120	inf
1	241.9	243.9	245.9	248.0	249.1	250.1	251.1	252.2	253.3	254.3
2	19.4	19.41	19.43	19.45	19.45	19.46	19.47	19.48	19.49	19.5
3	8.79	8.74	8.70	8.66	8.64	8.62	8.59	8.57	8.55	8.53
4	5.96	5.91	5.86	5.80	5.77	5.75	5.72	5.69	5.66	5.63
5	4.74	4.68	4.62	4.56	4.53	4.50	4.46	4.43	4.40	4.36
6	4.06	4.00	3.94	3.87	3.84	3.81	3.77	3.74	3.70	3.67
7	3.64	3.57	3.51	3.44	3.41	3.38	3.34	3.30	3.27	3.23
8	3.35	3.28	3.22	3.15	3.12	3.08	3.04	3.01	2.97	2.93
9	3.14	3.07	3.01	2.94	2.90	2.86	2.83	2.79	2.75	2.71
10	2.98	2.91	2.85	2.77	2.74	2.70	2.66	2.62	2.58	2.54
11	2.85	2.79	2.72	2.65	2.61	2.57	2.53	2.49	2.45	2.40
12	2.75	2.69	2.62	2.54	2.51	2.47	2.43	2.38	2.34	2.30
13	2.67	2.60	2.53	2.46	2.42	2.38	2.34	2.30	2.25	2.21
14	2.60	2.53	2.46	2.39	2.35	2.31	2.27	2.22	2.18	2.13
15	2.54	2.48	2.40	2.33	2.29	2.25	2.20	2.16	2.11	2.07
16	2.49	2.42	2.35	2.28	2.24	2.19	2.15	2.11	2.06	2.01
17	2.45	2.38	2.31	2.23	2.19	2.15	2.10	2.06	2.01	1.96
18	2.41	2.34	2.27	2.19	2.15	2.11	2.06	2.02	1.97	1.92
19	2.38	2.31	2.23	2.16	2.11	2.07	2.03	1.98	1.93	1.88
20	2.35	2.28	2.20	2.12	2.08	2.04	1.99	1.95	1.90	1.84
21	2.32	2.25	2.18	2.10	2.05	2.01	1.96	1.92	1.87	1.81
22	2.30	2.23	2.15	2.07	2.03	1.98	1.94	1.89	1.84	1.78
23	2.27	2.20	2.13	2.05	2.01	1.96	1.91	1.86	1.81	1.76
24	2.25	2.18	2.11	2.03	1.98	1.94	1.89	1.84	1.79	1.73
25	2.24	2.16	2.09	2.01	1.96	1.92	1.87	1.82	1.77	1.71
26	2.22	2.15	2.07	1.99	1.95	1.90	1.85	1.80	1.75	1.69
27	2.20	2.13	2.06	1.97	1.93	1.88	1.84	1.79	1.73	1.67
28	2.19	2.12	2.04	1.96	1.91	1.87	1.82	1.77	1.71	1.65
29	2.18	2.10	2.03	1.94	1.90	1.85	1.81	1.75	1.70	1.64
30	2.16	2.09	2.01	1.93	1.89	1.84	1.79	1.74	1.68	1.62
40	2.08	2.00	1.92	1.84	1.79	1.74	1.69	1.64	1.58	1.51
60	1.99	1.92	1.84	1.75	1.70	1.65	1.59	1.53	1.47	1.39
120	1.91	1.83	1.75	1.66	1.10	1.55	1.50	1.43	1.35	1.25
inf	1.83	1.75	1.67	1.57	1.52	1.46	1.39	1.32	1.22	1.00

F Values for $\alpha = 0.01$

d_2	d_1 1	2	3	4	5	6	7	8	9
1	4052	4999.5	5403	5625	5764	5859	5928	5982	6022
2	98.50	99.00	99.17	99.25	99.30	99.33	99.36	99.37	99.39
3	34.12	30.82	29.46	28.71	28.24	27.91	27.67	27.49	27.35
4	21.20	18.00	16.69	15.98	15.52	15.21	14.98	14.80	14.66
5	16.26	13.27	12.06	11.39	10.97	10.67	10.46	10.29	10.16
6	13.75	10.92	9.78	9.15	8.75	8.47	8.26	8.10	7.98
7	12.25	9.55	8.45	7.85	7.46	7.19	6.99	6.84	6.72
8	11.26	8.65	7.59	7.01	6.63	6.37	6.18	6.03	5.91
9	10.56	8.02	6.99	6.42	6.06	5.80	5.61	5.47	5.35
10	10.04	7.56	6.55	5.99	5.64	5.39	5.2	5.06	4.94
11	9.65	7.21	6.22	5.67	5.32	5.07	4.89	4.74	4.63
12	9.33	6.93	5.95	5.41	5.06	4.82	4.64	4.50	4.39
13	9.07	6.70	5.74	5.21	4.86	4.62	4.44	4.30	4.14
14	8.86	6.51	5.56	5.04	4.69	4.46	4.28	4.14	4.03
15	8.68	6.36	5.42	4.89	4.56	4.32	4.14	4.00	3.89
16	8.53	6.23	5.29	4.77	4.44	4.20	4.03	3.89	3.78
17	8.40	6.11	5.18	4.67	4.34	4.10	3.93	3.79	3.68
18	8.29	6.01	5.09	4.58	4.25	4.01	3.84	3.71	3.60
19	8.18	5.93	5.01	4.50	4.17	3.94	3.77	3.63	3.52
20	8.10	5.85	4.94	4.43	4.10	3.87	3.70	3.56	3.46
21	8.02	5.78	4.87	4.37	4.04	3.81	3.64	3.51	3.40
22	7.95	5.72	4.82	4.31	3.99	3.76	3.59	3.45	3.35
23	7.88	5.66	4.76	4.26	3.94	3.71	3.54	3.41	3.30
24	7.82	5.61	4.72	4.22	3.90	3.67	3.50	3.36	3.26
25	7.77	5.57	4.68	4.18	3.85	3.63	3.46	3.32	3.22
26	7.72	5.53	4.64	4.14	3.82	3.59	3.42	3.29	3.18
27	7.68	5.49	4.60	4.11	3.78	3.56	3.39	3.26	3.15
28	7.64	5.45	4.57	4.07	3.75	3.53	3.36	3.23	3.12
29	7.60	5.42	4.54	4.04	3.73	3.50	3.33	3.20	3.09
30	7.56	5.39	4.51	4.02	3.70	3.47	3.30	3.17	3.07
40	7.31	5.18	4.31	3.83	3.51	3.29	3.12	2.99	2.89
60	7.08	4.98	4.13	3.65	3.34	3.12	2.95	2.82	2.72
120	6.85	4.79	3.95	3.48	3.17	2.96	2.79	2.66	2.56
inf	6.63	4.61	3.78	3.32	3.02	2.80	2.64	2.51	2.41

F Values for $\alpha = 0.01$

	d_1									
d_2	10	12	15	20	24	30	40	60	120	inf
1	6056	6106	6157	6209	6235	6261	6287	6313	6339	6366
2	99.40	99.42	99.43	99.45	99.46	99.47	99.47	99.48	99.49	99.50
3	27.23	27.05	26.87	26.69	26.60	26.50	26.41	26.32	26.22	26.13
4	14.55	14.37	14.20	14.02	13.93	13.84	13.75	13.65	13.56	13.46
5	10.05	9.89	9.72	9.55	9.47	9.38	9.29	9.20	9.11	9.02
6	7.87	7.72	7.56	7.40	7.31	7.23	7.14	7.06	6.97	6.88
7	6.62	6.47	6.31	6.16	6.07	5.99	5.91	5.82	5.74	5.65
8	5.81	5.67	5.52	5.36	5.28	5.20	5.12	5.03	4.95	4.86
9	5.26	5.11	4.96	4.81	4.73	4.65	4.57	4.48	4.40	4.31
10	4.85	4.71	4.56	4.41	4.33	4.25	4.17	4.08	4.00	3.91
11	4.54	4.40	4.25	4.10	4.02	3.94	3.86	3.78	3.69	3.60
12	4.30	4.16	4.01	3.86	3.78	3.70	3.62	3.54	3.45	3.36
13	4.10	3.96	3.82	3.66	3.59	3.51	3.43	3.34	3.25	3.17
14	3.94	3.80	3.66	3.51	3.43	3.35	3.27	3.18	3.09	3.00
15	3.80	3.67	3.52	3.37	3.29	3.21	3.13	3.05	2.96	2.87
16	3.69	3.55	3.41	3.26	3.18	3.10	3.02	2.93	2.84	2.75
17	3.59	3.46	3.31	3.16	3.08	3.00	2.92	2.83	2.75	2.65
18	3.51	3.37	3.23	3.08	3.00	2.92	2.84	2.75	2.66	2.57
19	3.43	3.30	3.15	3.00	2.92	2.84	2.76	2.67	2.58	2.49
20	3.37	3.23	3.09	2.94	2.86	2.78	2.69	2.61	2.52	2.42
21	3.31	3.17	3.03	2.88	2.80	2.72	2.64	2.55	2.46	2.36
22	3.26	3.12	2.98	2.83	2.75	2.67	2.58	2.50	2.40	2.31
23	3.21	3.07	2.93	2.78	2.70	2.62	2.54	2.45	2.35	2.26
24	3.17	3.03	2.89	2.74	2.66	2.58	2.49	2.40	2.31	2.21
25	3.13	2.99	2.85	2.70	2.62	2.54	2.45	2.36	2.27	2.17
26	3.09	2.96	2.81	2.66	2.58	2.50	2.42	2.33	2.23	2.13
27	3.06	2.93	2.78	2.63	2.55	2.47	2.38	2.29	2.20	2.10
28	3.03	2.90	2.75	2.60	2.52	2.44	2.35	2.26	2.17	2.06
29	3.00	2.87	2.73	2.57	2.49	2.41	2.33	2.23	2.14	2.03
30	2.98	2.84	2.70	2.55	2.47	2.39	2.30	2.21	2.11	2.01
40	2.80	2.66	2.52	2.37	2.29	2.20	2.11	2.02	1.92	1.80
60	2.63	2.50	2.35	2.20	2.12	2.03	1.94	1.84	1.73	1.60
120	2.47	2.34	2.19	2.03	1.95	1.86	1.76	1.66	1.53	1.38
inf	2.32	2.18	2.04	1.88	1.79	1.70	1.59	1.47	1.32	1.00

Tabla B.4 Tabla P de la significancia del índice T de Wilcoxon

n \ α	0.10	0.05	0.02	0.01	0.005	0.001
5	0- 15	-	-	-	-	-
6	2- 19	0- 21	-	-	-	-
7	3- 25	2- 26	0- 28	-	-	-
8	5- 31	3- 33	1- 35	0- 36	-	-
9	8- 37	5- 40	3- 42	1- 44	0- 45	-
10	10- 45	8- 47	5- 50	3- 52	1- 54	-
11	13- 53	10- 56	7- 59	5- 61	3- 63	0- 66
12	17- 61	13- 65	9- 69	7- 71	5- 73	1- 77
13	21- 70	17- 74	12- 79	9- 82	7- 84	2- 89
14	25- 80	21- 84	15- 90	12- 93	9- 96	4-101
15	30- 90	25- 95	19-101	15-105	12-108	6-114
16	35-101	29-107	23-113	19-117	15-121	8-128
17	41-112	34-119	27-126	23-130	19-134	11-142
18	47-124	40-131	32-139	27-144	23-148	14-157
19	53-137	46-144	37-153	32-158	27-163	18-172
20	60-150	52-158	43-167	37-173	32-178	21-189
21	67-164	58-173	49-182	42-189	37-194	25-206
22	75-178	65-188	55-198	48-205	42-211	30-223
23	83-193	73-203	62-214	54-222	48-228	35-241
24	91-209	81-219	69-231	61-239	54-246	40-260
25	100-225	89-236	76-249	68-257	60-265	45-280
26	110-241	98-253	84-267	75-276	67-284	51-300
27	119-259	107-271	92-286	83-295	74-304	57-321
28	130-276	116-290	101-305	91-315	82-324	64-342
29	140-295	126-309	110-325	100-335	90-345	71-364
30	151-314	137-328	120-345	109-356	98-367	78-387
31	163-333	147-349	130-366	118-378	107-389	86-410
32	175-353	159-369	140-388	128-400	116-412	94-434
33	187-374	170-391	151-410	138-423	126-435	102-459
34	200-395	182-413	162-433	148-447	136-459	111-484
35	213-417	195-435	173-457	159-471	146-484	120-510
36	227-439	208-458	185-481	171-495	157-509	130-536
37	241-462	221-482	198-505	182-521	168-535	140-563
38	256-485	235-506	211-530	194-547	180-561	150-591
39	271-509	249-531	224-556	207-573	192-588	161-619
40	286-534	264-556	238-582	220-600	204-616	172-648