



DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN



**“Análisis de Algoritmos Metaheurísticos Vía Diagnóstico
Estadístico”**

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

PRESENTA:
ISC Mercedes Pérez Villafuerte
G04070504

DIRECTOR DE TESIS:
Dra. Laura Cruz Reyes

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



"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: AUTORIZACIÓN DE IMPRESIÓN DE TESIS

**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" ®

Mrs. Yolanda Chávez Cincó
M.P. MARÍA YOLANDA CHÁVEZ CINCO
JEFA DE LA DIVISIÓN



c.c.p.- Minuta
Archivo

MYCHC 'NLCO' jar
m



Ave. 1^{er} de Mayo y Sor Juana I. de la Cruz, Col. Los Mangos, CP. 89440 Cd. Madero, Tam.
Tel. (833) 357 48 20, Fax, Ext. 1002, e-mail: itcm@itcm.edu.mx
www.itcm.edu.mx



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.



Ing. Mercedes Pérez Villafuerte

13 de Junio de 2014, Cd. Madero, Tamps.

DEDICATORIA

Dedico este trabajo a mis padres

*Roberto Pérez Villanueva, a ti padre por ser mi ejemplo de disciplina, dedicación,
responsabilidad todo un ejemplo a seguir.*

*Mercedes Villafuerte Hilerio. A ti madre por ser un ejemplo de lucha, fortaleza, amor,
alegría y sobre todo, por ser mi mejor amiga.*

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

*Y por últimos pero no menos importantes... 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

Agradezco al Consejo Nacional de Ciencia y Tecnología (CONACYT) y a la Dirección General de Educación Tecnológica (DGEST) por el apoyo y facilidades recibidas para la conclusión de este trabajo.

De igual manera agradezco al Instituto Tecnológico de Cd. Madero por las facilidades proporcionadas para el desarrollo de este proyecto, así como a la planta de profesores que contribuyeron a mi formación académica.

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

Estoy infinitamente agradecida con mi Asesora de Tesis, Dra. Laura Cruz Reyes, porque ha sido para mí algo más que mi asesora, ha sido una amiga, le doy mi respeto y admiración por ser una mujer trabajadora, inteligente y con gran calidez. Así mismo manifiesto mi agradecimiento a la Dra. Claudia Guadalupe Gómez Santillán, por sus consejos y apoyo recibido. Sin duda, mujeres que son un ejemplo a seguir.

Gracias a los colaboradores que participaron en el desarrollo de este trabajo, en especial a Jesús Alviso Vargas, Jessica Márquez Mateos, Alan Gabriel Aguirre Lam, José Carlos Soto Monterrubio, y Andrés Bautista.

Agradezco a mis suegros, personas que han llegado a ser como mis segundos padres, César Medina de León y María Cristina Estela Trejo Salas, muchas gracias por su apoyo, confianza y amor manifestado hacia mi persona.

Gracias a toda mi familia, por su paciencia y comprensión, sus palabras de aliento y por todo el amor que me muestran, les amo.

Te agradezco a ti César Medina Trejo, empezamos este proyecto juntos, hemos pasado por dificultades y seguimos en pie, has sido mi mejor amigo y mi gran amor, gracias por tu apoyo, he aprendido mucho contigo y de ti, te amo.

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 cómo 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. Generalmente este trabajo se hace manualmente a base de prueba y error consumiendo demasiado tiempo.

Con la finalidad de apoyar al diseño de algoritmos metaheurísticos, recientemente se han propuesto herramientas orientadas al estudio de su desempeño. Hasta nuestro conocimiento, ninguna de estas herramientas aborda todo el proceso de optimización, que incluye: la estructura del problema, el comportamiento del algoritmo y el desempeño final. Además, son pocas las que integran un conjunto de pruebas estadísticas que permitan estudiar el desempeño final de manera confiable.

El presente proyecto aporta la arquitectura de una herramienta para el análisis de todo el proceso de optimización de algoritmos metaheurísticos. En la arquitectura se contempla la integración de técnicas de diversos campos, destacando estadística y visualización.

Debido a la amplitud de la arquitectura propuesta, la herramienta implementada sólo se enfoca al análisis comparativo de algoritmos. Se busca dar soporte estadístico a los estudios

para asegurar que los resultados tengan validez. Para ello, se proporciona un conjunto de pruebas estadísticas que pueden ser utilizadas de manera complementaria. Como caso de estudio se analizan algoritmos de solución del problema de empacado de objetos en contenedores.

Summary

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

When complex problems are solved, the performance of metaheuristic algorithms depends on many factors, a poor design can lead to poor performance. There are no guidelines that tell us how to properly design the metaheuristics. That is why the designers of these methods take too much time to adjust, even more time than implementing the metaheuristic itself. Usually this work is done manually through trial and error and consuming too much time.

In order to support the design of metaheuristic algorithms, there has been recently proposed tools aimed at the study of their performance. To our knowledge, none of these tools addresses the entire optimization process, including: the structure of the problem, the behavior of the algorithm and the final performance. In addition, few have integrated a set of statistical tests for the study of the final performance in a reliable way.

This project contributes with the architecture of a tool for analyzing the whole process of optimization for metaheuristic algorithms. In this architecture is contemplated the integration of techniques from diverse fields, emphasizing statistics and visualization.

Because of the scope of the proposed architecture, the implemented tool only focuses on the comparative analysis of algorithms. It seeks to give statistical support for the studies to secure the results are valid. For this, a set of statistical tests is provided, which can be used in a complementary manner. As a case study algorithms solving the bin packing problem on container are analyzed.

Tabla de contenido

Capítulo 1. Introducción	1
1.1. Antecedentes del proyecto.....	2
1.2 Descripción del problema.....	3
1.3 Justificación y beneficios del proyecto.....	3
1.4 Objetivos.....	4
1.5 Alcances y limitaciones del proyecto	5
1.6 Descripción de la complejidad del problema	5
1.7 Organización del documento	6
Capítulo 2. Problemas, algoritmos y su caracterización	8
2.1 Métodos de solución para problemas NP-duros	8
2.1.1 Problemas de optimización combinatoria.....	8
2.1.2 Heurísticos	9
2.1.3 Metaheurísticos.....	11
2.1.4 Hiperheurísticos.....	12
2.2 Problema de empacado de objetos en contenedores.....	13
2.3 Proceso de optimización y su caracterización	13
2.4 Métricas de desempeño para BPP	15
2.4.1 Caracterización del problema	15
2.4.2 Algoritmo genético para el problema de empacado de objetos: GGA-CGT.....	17
Capítulo 3. Estado del arte	21
3.1 Herramientas de análisis de algoritmos	21
3.1.1 ParadisEO	21
3.1.2 BPP Visualization Tool	22
3.1.3 OAT (Optimization Algorithm Toolkit).....	23
3.1.4 HeuristicLab Optimization Environment	25
3.1.5 Visualizer for Metaheuristics Development Framework.....	26
3.1.6 VIZ Visualization for Analyzing Trajectory-Based Metaheuristic Search Algorithms	27
3.2 Análisis estadístico de algoritmos aproximados.....	33

3.2.1 “A statistical test for comparing success rates”	33
3.2.2 “A study on the use of non-parametric tests for analyzing the evolutionary algorithms’ behaviour: a case study on the CEC’2005 Special Session on Real Parameter Optimization”	33
3.2.3 “Un tutorial sobre el uso de test estadísticos no paramétricos en comparaciones múltiples de metaheurísticas y algoritmos evolutivos”	34
3.2.4 “Knowledge Extraction based on Evolutionary Learning”	35
3.3 Análisis experimental del proceso de optimización	39
3.4 VisTHAA	42
3.5 Comentarios finales	42
Capítulo 4. Análisis estadístico	45
4.1 Prueba estadística	45
4.2 Pruebas estadísticas paramétricas y no paramétricas	47
4.2.1 Uso de pruebas no paramétricas para analizar algoritmos aproximados	48
4.3 Pruebas no paramétricas	49
4.3.1 Prueba de los signos	49
4.3.2 Prueba de signos múltiples	50
4.3.3 Estimación de contraste	51
4.3.4 Prueba de Wilcoxon de rangos con signos	52
4.3.5 Prueba de Friedman	54
4.3.6 Prueba de Quade	55
Capítulo 5. Arquitectura propuesta para VisTHAA	58
5.1 Planeación de la herramienta de diagnóstico visual	58
5.2 Módulos de VisTHAA	60
5.2.1 Módulo de caracterización y visualización	61
5.2.2 Módulo de modelación de desempeño algorítmico	62
5.2.3 Módulo de generación de instancias	63
5.2.4 Módulo de procesamiento de datos	63
5.2.5 Módulo de análisis comparativo de algoritmos	66
Capítulo 6. Experimentación y resultados	68
6.1 Diseño experimental	68
6.2 Análisis funcional de VisTHAA	68

6.2.1 Prueba de Friedman	69
6.2.2 Prueba de signos múltiples	72
6.2.3 Prueba estimación de contraste.....	76
6.2.4 Prueba de Wilcoxon de rangos con signos	79
6.3 Estudio comparativo de un algoritmo genético para el problema de empacado de objetos.....	81
6.3.1 Prueba Wilcoxon de rangos con signos a GGA-CGT	81
6.4 Estudio comparativo de GGA-CGT vs. algoritmos del estado del arte.....	83
6.4.1 Prueba de Friedman	83
6.4.2 Prueba de signos múltiples	85
6.4.3 Prueba de estimación de contraste	86
6.5 Comentarios finales	88
Capítulo 7. Conclusiones y trabajo futuro	89
Referencias	91
Anexo A Demostración de complejidad del Problema de Empacado de Objetos en Contenedores.....	97
Anexo B Tablas de valores críticos	99
Anexo C Ejemplos de procedimientos de pruebas no paramétricas	107
Anexo D Códigos en R para comprobar pruebas estadísticas no paramétricas usadas en experimentaciones.....	115
Anexo E Errores promedio obtenido en 25 funciones de referencia	118
Anexo F Funciones de Prueba.....	119
Anexo G Instancias usadas para las pruebas no paramétricas en VisTHAA	120

Índice de Figuras

Figura 2.1 Proceso de optimización de un problema y su caracterización.....	14
Figura 2.2 Función de relación de desempeño	15
Figura 3.1 BPP Simulator.....	22
Figura 3.2 OAT Explorer	25
Figura 3.3 Ejemplo de ejecución en HeuristicLab Optimizer.	26
Figura 3.4 Entorno VizSIMRA	29
Figura 3.5 Visualizaciones en VIZ específicas del algoritmo y del problema.	29
Figura 3.6 Entorno Viz EW	30
Figura 3.7 Ejemplo de archivo de configuración.	31
Figura 3.8 Casos de prueba resultantes del archivo de configuración.....	31
Figura 3.9 Resultados de la experimentación con diferentes configuraciones.....	32
Figura 3.10 Ventana del módulo “Data management”	35
Figura 3.11 Ventana del módulo “Design of experiments”.....	36
Figura 3.12 Ventana del módulo “Design of imbalanced experiments”	36
Figura 3.13 Ventana del módulo “Experimentation with multiple instance learning algorithms”.....	37
Figura 3.14 Ventana del módulo “Statistical tests”.....	38
Figura 3.15 Ventana del módulo “Educational experiments”.....	38
Figura 5.1 Esquema de módulos de VisTHAA	59
Figura 5.2 Herramientas para el Análisis Experimental del Proceso de Optimización de Algoritmos Heurísticos.	60
Figura 5.3 Archivos involucrados en el Procesamiento de Datos	65
Figura 5.4 Módulo de análisis comparativo de algoritmos	66
Figura 6.1 Rankings del test de Friedman [Derrac 2012].....	70
Figura 6.2 - Ejecución en R. Tabla de rankings y valor Rj para cada algoritmo.....	71
Figura 6.3 Resultados en VisTHAA de prueba de Friedman con datos de [Derrac 2012] ..	71
Figura 6.4 Prueba de signos múltiples [Derrac 2012]	73
Figura 6.5 Prueba de signos múltiples en R	74

Figura 6.6 Prueba de signos múltiples en VisTHAA.....	74
Figura 6.7 Prueba estimación de contraste [Derrac 2012].....	76
Figura 6.8 Estimador de contraste [Derrac 2012].....	77
Figura 6.9 Estimación de contraste. Tabla de diferencias y medianas en R.....	77
Figura 6.10 Estimación de contraste. Tabla de estimador de contraste en R.....	78
Figura 6.11 Resultados en VisTHAA de estimación de contraste con datos de [Derrac 2012].....	78
Figura 6.12 Tabla de resultados de prueba de Wilcoxon, G-CMA-ES vs otros algoritmos [García 2009].....	80
Figura 6.13 Prueba de Wilcoxon en VisTHAA, G-CMA-ES vs CoEVO.....	80
Figura 6.14 Datos de desempeño del algoritmo GGA-CGT.....	81
Figura 6.15 Prueba de Wilcoxon en VisTHAA a GGA-CGT.....	82
Figura 6.16 Instancia con errores de algoritmos que resuelven BPP.....	83
Figura 6.17 Resultados de la prueba de Friedman en VisTHAA.....	84
Figura 6.18 Prueba de signos múltiples en VisTHAA, selección de método de control.....	85
Figura 6.19 Resultados de la prueba de signos múltiples en VisTHAA	86
Figura 6.20 Resultados de la prueba de estimación de contraste en VisTHAA	87

Índice de Tablas

Tabla 2.1 Índices propuestos por Cruz [Cruz 2004].....	16
Tabla 3.1 Estado del arte en análisis de desempeño de algoritmos heurísticos	43
Tabla 3.2 Trabajos relacionados con el análisis estadístico de algoritmos aproximados.....	44
Tabla 5.1 Pruebas estadísticas no paramétricas que se incluyen a VisTHAA	67
Tabla A.1 Función de transformación de PP a BPP	98
Tabla B.2 Valores críticos en la prueba de los signos	99
Tabla B.3 Valores críticos de rj en la prueba de signos múltiples.	99
Tabla B.4 Tabla de la Distribución F	101
Tabla B.5 Tabla P de la significancia del índice T de Wilcoxon	106
Tabla C.1.1 Resultados de la prueba de signos múltiples	109
Tabla C.2.1 Resultados de desempeño de algoritmos de estudio	110
Tabla C.2.2 Resultados de diferencias en pares para cada algoritmo	111
Tabla C.3.1 Rankings para la prueba de Friedman para el ejemplo	113
Tabla E.1 Errores promedio obtenido en las 25 funciones de referencia.	118

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 empacado de objetos en contenedores (*Bin Packing Problem*,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 desempeño, aún existen instancias de BPP que no han sido resueltas. El desempeño de estos algoritmos es 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 aceptadas por la comunidad que nos indiquen cómo diseñar apropiadamente los metaheurísticos. Este trabajo generalmente se hace manualmente a base de prueba y error consumiendo demasiado tiempo.

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 esto no es suficiente. Como resultado de la revisión de la literatura, en esta tesis se propone la arquitectura de una herramienta para el análisis y diseño de algoritmos metaheurísticos. Debido al gran alcance de la arquitectura propuesta, en el proyecto sólo se aborda el desarrollo de una herramienta para el estudio comparativo de algoritmos; se busca dar soporte estadístico a los estudios para asegurar que los resultados de desempeño no ocurrieron por el azar.

Por lo antes expuesto, se espera contribuir al área de algoritmia experimental con una arquitectura que contempla la integración de diversas técnicas de análisis y diseño. La herramienta construida con la arquitectura propuesta, denominada VisTHAA, cuenta con diagnóstico estadístico y visual. Este último fue desarrollado en un trabajo previo [Castillo 2011].

1.1. Antecedentes del proyecto

El grupo de investigación, del cual se derivó este proyecto, tiene como objetivo de largo plazo el contribuir al entendimiento del funcionamiento de algoritmos de solución aproximada, como lo son los metaheurísticos. Previo al presente trabajo se han desarrollado tesis de licenciatura, maestría y doctorado encaminadas a este gran objetivo. En la sección 3.3 se da una revisión rápida de los trabajos más representativos del grupo. Cabe destacar que el problema de empacado de objetos en contenedores ha sido utilizado como caso de estudio en la mayoría de estos trabajos.

Comprender el funcionamiento de algoritmos metaheurísticos requiere mucho trabajo experimental, además del teórico. Esta tesis surge de la necesidad de contar con una herramienta integral que facilite el trabajo experimental sobre este tipo de algoritmos.

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 configurado: típicamente algunos módulos necesitan ser desarrollados y algunos parámetros necesitan ser afinados. A estos dos problemas se le llama ajuste “estructural” y “paramétrico” respectivamente, a la combinación de estos dos problemas se les llama “afinación”. La afinación es crucial para la optimización de un metaheurístico, ya que sólo así se pueden obtener resultados buenos o incluso óptimos, de otra manera, se obtienen resultados pobres o a lo mucho resultados promedio [Halim 2009].

En muchos casos, el diseño de metaheurísticos se hace de manera manual y sin sustento teórico, dificultando la comprensión de su buen o mal desempeño [Snodgrass 2010]. El reto de este proyecto es contribuir a la automatización del análisis experimental de algoritmos mediante una herramienta de diagnóstico estadístico. Particularmente, con la incorporación de pruebas estadísticas que permitan contrastar diferentes algoritmos y diferentes configuraciones para un mismo algoritmo. El desarrollo de la herramienta estadística se hace en el contexto de una arquitectura propuesta en esta tesis para el análisis y diseño de algoritmos metaheurísticos.

1.3 Justificación y beneficios del proyecto

El análisis estadístico es una herramienta muy útil en el diseño de experimentos, lo cual es una tarea de gran importancia en el desarrollo de nuevos propuestas de algoritmos. La validación de nuevos algoritmos requiere de un marco experimental que permita la inclusión de un amplio abanico de problemas y algoritmos del estado del arte. La parte crítica de estas comparaciones recae en la validación estadística de los resultados, contrastando las diferencias encontradas entre métodos. Esta misma herramienta puede ser útil para verificar si el rediseño de algoritmos realmente está aportando una mejora significativa.

Como ya se ha mencionado anteriormente, el principal caso de estudio de este trabajo es el problema de empacado de objetos en contenedores. Este 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 áreas: 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.

Para el problema de empacado y muchos otros problemas complejos, la fecha hay muchos metaheurísticos que se han desarrollado y muchas son ajustados y realizados a prueba y error que toman demasiado tiempo; el gran reto en la algoritmia metaheurística es establecer fundamentos sólidos y no los hay, sólo existen construcciones artesanales, se desea que los diseños de los algoritmos se hagan con bases teóricas [Morrison 2010]. Este proyecto no va solucionar este problema de manera inmediata, sólo se va a contribuir a su solución en el largo plazo, mediante una herramienta de análisis experimental que facilite la comprensión del comportamiento algorítmico.

1.4 Objetivos

Diseñar una arquitectura de software que permita la incorporación de pruebas estadísticas de significancia adecuadas para el análisis del desempeño de algoritmos aproximados, como los son los metaheurísticos.

Objetivo específicos

- Extender la herramienta VisTHAA con la incorporación de pruebas estadísticas no paramétricas para su aplicación al análisis del desempeño de algoritmos aproximados.
- Documentar las pruebas estadísticas seleccionadas para la herramienta VisTHAA.

- Elaborar casos de uso para mostrar la utilidad del análisis estadístico del desempeño algorítmico.

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 algoritmos aproximados mediante diagnóstico estadístico, aunque en esta tesis la herramienta sólo se aplica a la solución del problema de bin packing unidimensional, no está limitada a dicho problema.

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 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 [Halim 2006a].

- 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 experiencias que otros investigadores 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 el algoritmo metaheurístico [Adenso 2006].

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).

1.7 Organización del documento

La tesis está organizada de la siguiente manera:

Capítulo 2: *Problemas, algoritmos y su caracterización*, se muestran algunos fundamentos teóricos de los componentes del proceso de optimización y conceptos que están relacionados con éste trabajo.

Capítulo 3: *Estado del arte*, se hace una revisión de herramientas que tienen como apoyo la visualización de datos para encontrar áreas de mejora en algoritmos heurísticos. De igual manera se muestra el resultado de la revisión de trabajos que hacen uso del análisis estadístico para hacer comparaciones entre algoritmos y también para encontrar información significativa que pueda ser de utilidad para la mejora de algoritmos.

Capítulo 4: *Pruebas estadísticas*, en este capítulo se revisan las pruebas de hipótesis con un enfoque orientado a su implementación en la arquitectura propuesta para el análisis de algoritmos aproximados.

Capítulo 5: *Arquitectura propuesta para VisTHAA*, en este capítulo se describe la estructura propuesta y desarrollada para la herramienta VisTHAA, señalando la aportación del desarrollo del módulo de análisis estadístico.

Capítulo 6: *Experimentación y Resultados*, se presenta la aplicación de pruebas estadísticas no paramétricas al estudio comparativo de algoritmos.

Capítulo 7: *Conclusiones y Trabajos Futuros*, Se mencionan las conclusiones pertinentes al presente trabajo así como las recomendaciones para mejorarlo.

Anexo A: *Demostración de complejidad del Problema de Empacado de Objetos en Contenedores*

Anexo B: *Tablas de valores críticos*, se presentan las tablas de valores críticos que se necesitan para las pruebas estadísticas descritas en el capítulo 4.

Anexo C: *Ejemplos de procedimientos de pruebas no paramétricas*, se muestran ejemplos de los procedimientos para hacer las pruebas no paramétricas que se incorporaron a VisTHAA.

Anexo D: *Códigos en R para comprobar pruebas estadísticas no paramétricas usadas en experimentaciones*, se muestran los códigos en el lenguaje R que se hicieron para comprobar el funcionamiento de VisTHAA.

Anexo E: *Errores promedio obtenido en 25 funciones de referencia*. Se muestra una tabla con los errores promedio de las 25 funciones de prueba obtenidos por 9 algoritmos de estudio.

Anexo F: *Funciones de prueba*. Se da una breve descripción de las 25 funciones que son usadas en los experimentos del capítulo 6.

Anexo G: *Instancias usadas para las pruebas no paramétricas en VisTHAA*. Se muestran las instancias que se usaron para hacer los experimentos del capítulo 6.

Capítulo 2

Problemas, algoritmos y su caracterización

2.1 Métodos de solución para problemas NP-duros

En esta sección se describen tres tipos de algoritmos de solución aproximada: heurísticos, metaheurísticos e hiperheurísticos. Los algoritmos hiperheurísticos 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. 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 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 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 subóptima.
- Cuando los datos son poco fiables.
- Existen limitaciones de recursos computacionales.
- Como preprocesamiento en algún otro algoritmo.

Para BPP un algoritmo heurístico clásico es el FFD (First Fist Decreasing), el cual ordena descendenteamente los objetos y los acomoda en ese orden en el contenedor más lleno que los pueda contener. Esta estrategia voraz forma parte de la siguiente clasificación [Duarte 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 extrae la solución al problema original.

Métodos de búsqueda: Parten de una solución factible dada y a partir de ella intentan mejorarla. Los siguientes son algunos ejemplos de esta categoría:

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 aquél 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 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 cruce.

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 Hiperheurísticos

El término hiperheurístico 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 2005].

El término en sí fue acuñado por Cowling [Cowling 2000] el cual menciona “Los hiperheurísticos administran la elección de cuál 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 hiperheurísticos son presentados como una alternativa para los metaheurísticos que son mayormente desarrollados para un problema en particular. Se espera que los hiperheurísticos puedan ser desarrollados y empleados por programadores no especializados con poca o experiencia nula en el dominio del problema [Özcan 2008].

2.2 Problema de empacado de objetos en contenedores

El problema de empacado de objetos en contenedores (BPP, Bin Packing Problem) es un problema clásico de optimización combinatoria de los denominados NP-duro debido a que son considerados intratables pues demandan muchos recursos para su solución, y la cantidad requerida por éstos es parecida a una función polinomial de alto grado [Garey 1979].

El problema de Bin Packing se compone de una secuencia de n objetos L a distribuir y un número ilimitado de contenedores con una capacidad de carga c , donde:

- $L = \{a_1, a_2, \dots, a_n\}$ es el conjunto objetos a_i , donde cada objeto a_i tiene un peso dado $0 < s(a_i) \leq c$.
- $B = \{B_1, B_2, \dots, B_j\}$ es el conjunto de contenedores.

La tarea a resolver es encontrar una partición de L mínima, $L = B_1 \cup B_2 \cup \dots \cup B_m$, tal que en cada contenedor B_j la sumatoria del peso de cada objeto $s(a_i)$ en él no exceda c , esto es

$$\sum_{a_i \in B_j} s(a_i) \leq c, \quad 1 \leq j \leq m.$$

2.3 Proceso de optimización y su caracterización

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

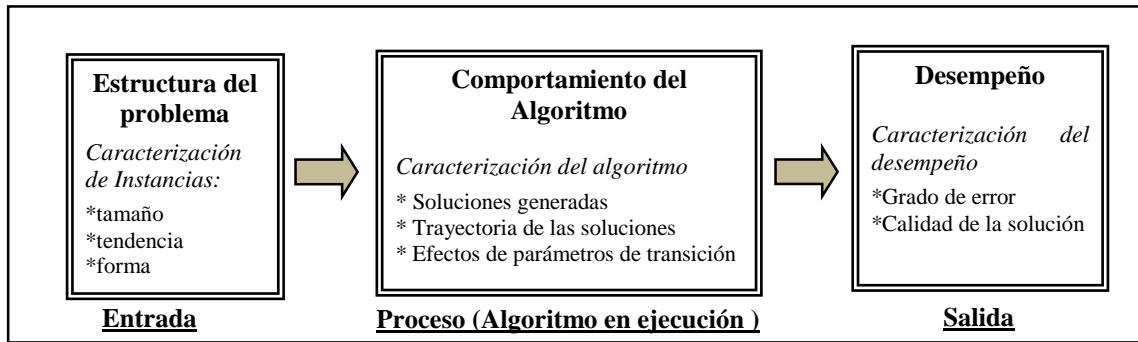


Figura 2.1 Proceso de optimización de un problema y su caracterización.

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. Existen métodos estadísticos que permiten establecer relaciones entre variables aleatorias, entre ellos destaca el análisis causal. Con la construcción de un modelo causal es posible representar formalmente las relaciones [Quiroz 2009, Pérez 2007]. 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.

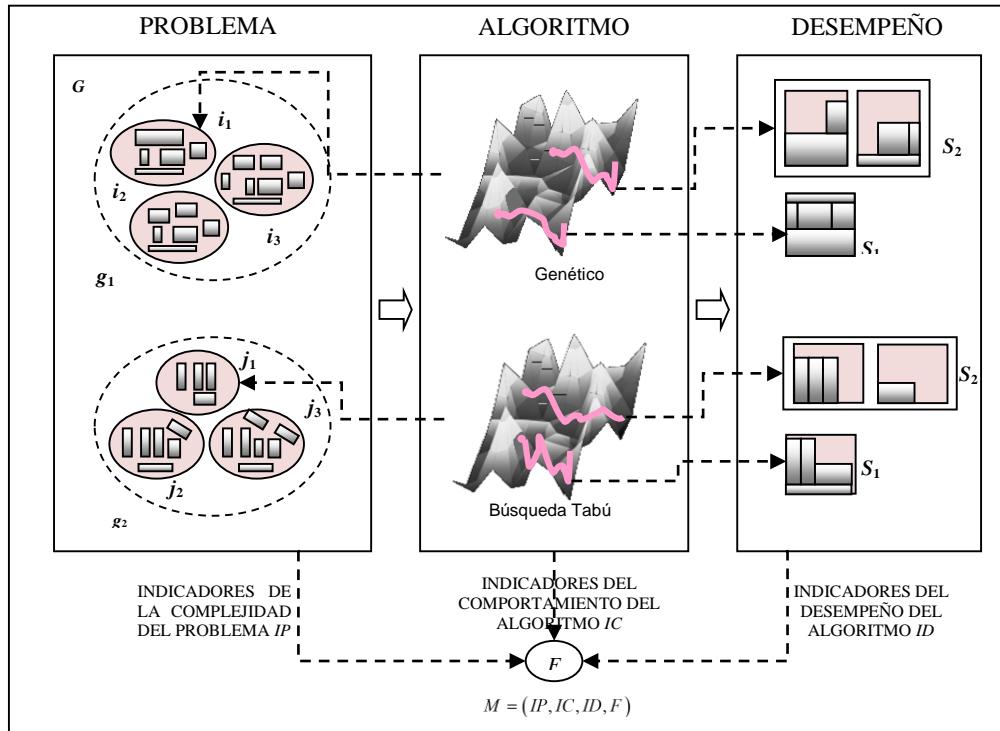


Figura 2.2 Función de relación de desempeño. Figura tomada de [Pérez 2007]

2.4 Métricas de desempeño para BPP

Muchos factores afectan el grado de dificultad de una instancia de BPP, por eso para entenderla y tratar de explicar las dificultades es necesario hacer caracterizaciones propias del problema y de la instancia. De esta forma, diferentes autores han propuesto conjuntos de índices para la caracterización de BPP.

2.4.1 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, estos índices aportan una mayor cantidad de información sobre el problema de estudio. Un ejemplo de estos, en el caso del problema de empacado de objetos (BPP) son los propuestos por Cruz [Cruz 2004], están descritos en la Tabla 2.1.

Tabla 2.1 - Índices propuestos por Cruz [Cruz 2004]

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

Tabla 2.2 - Índices propuestos por Cruz [Cruz 2004]. Continuación

Expresión	Descripción
$b = \begin{cases} 1 & \text{si } c \geq \sum_{i=1}^n s_i \\ \frac{c}{\sum_{i=1}^n s_i} & \text{en caso contrario} \end{cases}$	El uso de contenedor b expresa la proporción del tamaño total que se puede asignar en un contenedor de capacidad c .

2.4.2 Algoritmo genético para el problema de empacado de objetos: GGA-CGT

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 propuesto por Quiroz [Quiroz 2014]. Este algoritmo es denominado GGA-CGT (Grouping Genetic Algorithm with Controlled Gene Transmission), *Algoritmo Genético de Agrupación con Transmisión de Genes Controlada*.

GGA-CGT promueve la transmisión de los mejores genes en los cromosomas sin perder el equilibrio entre la presión selectiva y diversidad de la población. La transmisión de los mejores genes se realiza por medio de un nuevo conjunto de operadores genéticos de agrupación, mientras que el proceso evolutivo es equilibrado por medio de una nueva técnica de reproducción que controla la exploración del espacio de búsqueda y evita la convergencia prematura del algoritmo. El Algoritmo 2.1 describe el procedimiento general.

ALGORITMO GENÉTICO DE AGRUPACIÓN CON TRANSMISIÓN DE GENES CONTROLADA

Procedimiento GGA-CGT

- 1 Generar una población inicial P con la heurística FF- \tilde{n} ;
 - 2 **mientras** generación < max_gen y Valor(mejor_solución) > L_2
 - 3 Seleccionar n_c individuos por medio de Selección_Controlada para la cría;
 - 4 Aplicar Cruzamiento_Nivel_Gen y la heurística FFD a los n_c individuos seleccionados;
 - 5 Aplicar Reemplazo_Controlado para introducir las crías;
 - 6 Seleccionar n_m individuos por medio de Selección_Controlada para mutación y clonar;
 - 7 Aplicar Mutación_Adaptativa y la heurística RP a los n_m mejores individuos;
 - 8 Aplicar Reemplazo_Controlado para introducir los clones;
 - 9 actualizar mejor_solución_global;
 - 10 **fin mientras;**
 - fin** GGA-CGT
-

Algoritmo 2.1 Procedimiento general del algoritmo GGA-CGT

A continuación se describe el proceso general de GGA-CGT tomado de Quiroz [Quiroz 2014]. El proceso inicia generando una población P de individuos con la heurística de empacado FF- \tilde{n} (Línea 1). Luego, durante un máximo de max_gen generaciones, algunos individuos de P son seleccionados para ser recombinados y mutados en dos fases. En la primera fase (Líneas 3-5), n_c individuos son seleccionados con la estrategia Selección_Controlada para aplicarles el operador de cruce Cruzamiento_Nivel_Gen y la heurística FFD; en seguida, la estrategia Reemplazo_Controlado es utilizada para introducir la descendencia. En la segunda fase (Líneas 6-8), de acuerdo con una max_edad predefinida, los individuos del grupo élite B son clonados y los mejores n_m individuos son seleccionados para aplicarles el operador Mutación_Adaptativa y la heurística RP; posteriormente, los clones son introducidos a la población por medio de la estrategia Reemplazo_Controlado. El algoritmo se detendrá antes de alcanzar el número máximo de generaciones max_gen en caso de encontrar una solución cuyo tamaño coincide con el límite inferior L_2 de Martello y Toth [Martello 1990]. El resultado final del algoritmo es el individuo con la mayor aptitud de todo el proceso evolutivo.

A continuación se da una breve descripción de los componentes de GGA-CGT

Heurística FF. El algoritmo FF (First Fit) acomoda un objeto en el primer contenedor que tiene suficiente capacidad disponible para almacenarlo; si ninguno de los contenedores puede almacenarlo, el objeto se almacenará en un nuevo contenedor vacío.

Heurística FFD. Es similar a FF pero el acomodo se realiza sobre el conjunto de objetos ordenados de manera descendente.

Heuristica FF- \tilde{n} . Inicialmente, los \tilde{n} objetos grandes son empacados en contenedores separados. Enseguida, los $n - \tilde{n}$ objetos restantes son empacados usando la heurística de acomodo FF sobre una permutación aleatoria de los objetos.

Heurística RP. La heurística RP (Reacomodo por Pares) se compone de dos etapas: primero, se recorre cada contenedor en un intento de mejorar su llenado haciendo intercambios entre

pares de objetos empacados y objeto libres; segundo, los objetos libres se introducen en la solución utilizando la heurística FF.

Selección_Controlada para cruzamiento. Primero se genera el conjunto G que incluye $n_c/2$ individuos tomados al azar de los mejores n_c individuos de P . Enseguida se genera el conjunto R que incluye $n_c/2$ individuos elegidos al azar de $P \setminus B$, siendo B es el grupo elite. El operador de cruzamiento tomará estos elementos por pares, cruzando G_i con R_i ($0 \leq i < n_c/2$). Se prohíbe que un mismo individuo tenga la misma posición i en ambos conjuntos y que el cruzamiento sea entre los individuos del grupo élite B .

Cruzamiento_Nivel_Gen. Dadas dos soluciones padre p_1 y p_2 , se generan dos hijos, cruzando: p_1 con p_2 y p_2 con p_1 . Los contenedores de ambos padres son comparados en pares, el más lleno es el primero en ser heredado a la nueva solución, seguido por el otro contendor; si tienen el mismo llenado, se da preferencia al del primer parente. Al final del cruzamiento, si un parente aún tiene contenedores, éstos son heredados directamente a la nueva solución. Los contendores con objetos duplicados son eliminados de la nueva solución, y los objetos que queden fuera de la solución son reinsertados con la heurística FFD.

Reemplazo_Controlado para cruzamiento. El operador de cruzamiento genera un conjunto C de n_c hijos a partir de los conjuntos G (buenas soluciones) y R (soluciones aleatorias). Los primeros $n_c / 2$ hijos son introducidos en P sustituyendo a los individuos de R . Los otros $n_c / 2$ hijos se introducen en P para reemplazar soluciones de G que no son elite con las siguientes reglas: a) si hay individuos con aptitud duplicada, reemplazarlos con nuevas crías; b) si aún hay crías que no han sido insertadas en P , introducirlas reemplazando las peores soluciones.

Selección_Controlada para la mutación. El número de individuos para mutar n_m debe ser mayor que el tamaño del grupo elite y se seleccionan del conjunto de mejores individuos de P en orden decreciente de su aptitud. Antes de aplicar el operador de mutación se aplica una clonación a los individuos del grupo elite B que no han alcanzado una edad límite.

Mutación_Adaptativa. Dada una solución, los contenedores son considerados en orden descendente de su llenado, eliminando los contenedores menos llenos y reinsertando los objetos libres con una heurística reacomodo RP. Se calcula adaptivamente el número de contenedores a eliminar en función del tamaño del cromosoma y del número de contenedores incompletos.

Reemplazo_Controlado para la mutación. Antes de aplica el operador de mutación, algunos de los mejores individuos se clonian para preservar las mejores soluciones. Cada clon se puede introducir en la población de dos maneras: a) si hay soluciones con aptitud duplicada, entonces el clon reemplazará una de ellas; b) si la primera alternativa no es posible, entonces el clon reemplazará a la peor de las soluciones.

Límite L₂. Fue propuesto por Martello y Toth [Martello 1990] para estimar un límite inferior del número óptimo de contenedores.

Capítulo 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. Las herramientas cuyo código está disponible en la Web también se revisaron de manera práctica, además de documental. Cabe mencionar que la herramienta desarrollada 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. La parte visual fue desarrollada en un trabajo previo de Castillo [Castillo 2011].

3.1 Herramientas de análisis de algoritmos

3.1.1 ParadisEO [ParadisEO 2013]

ParadisEO [Liefooghe 2009] es una arquitectura de caja blanca dedicada al diseño de metaheurísticos reusables, metaheurísticos híbridos, metaheurísticos paralelos y distribuidos. 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 [ParadisEO 2013].

3.1.2 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.

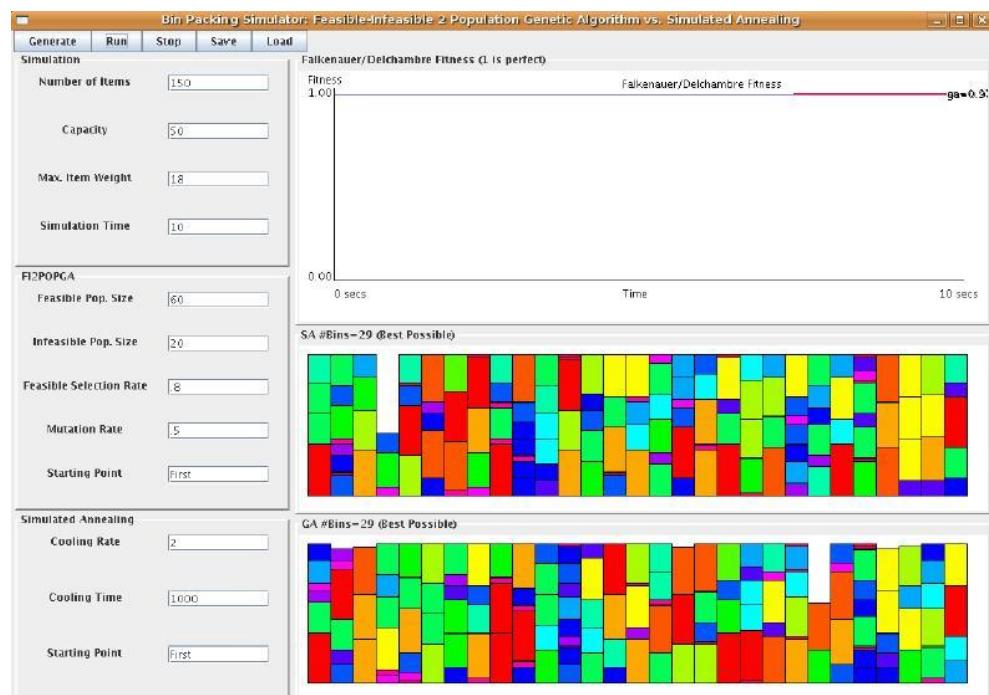


Figura 3.1 BPP Simulator

3.1.3 OAT (Optimization Algorithm Toolkit), [Brownlee 2007]

OAT cuenta con una sección llamada Experimenter, la cual es una interfaz 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 y visualizaciones.

Las herramientas estadísticas que se proveen para las pruebas estadísticas de hipótesis son: prueba de normalidad (prueba Anderson-Darling, prueba de criterio Cramér-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: 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 interfaz 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 interfaz es la experimentación informal exploratoria mediante la visualización de la ejecución y la colección de información generada durante la ejecución. Fue inspirado en la interfaz 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 [OAT 2013].

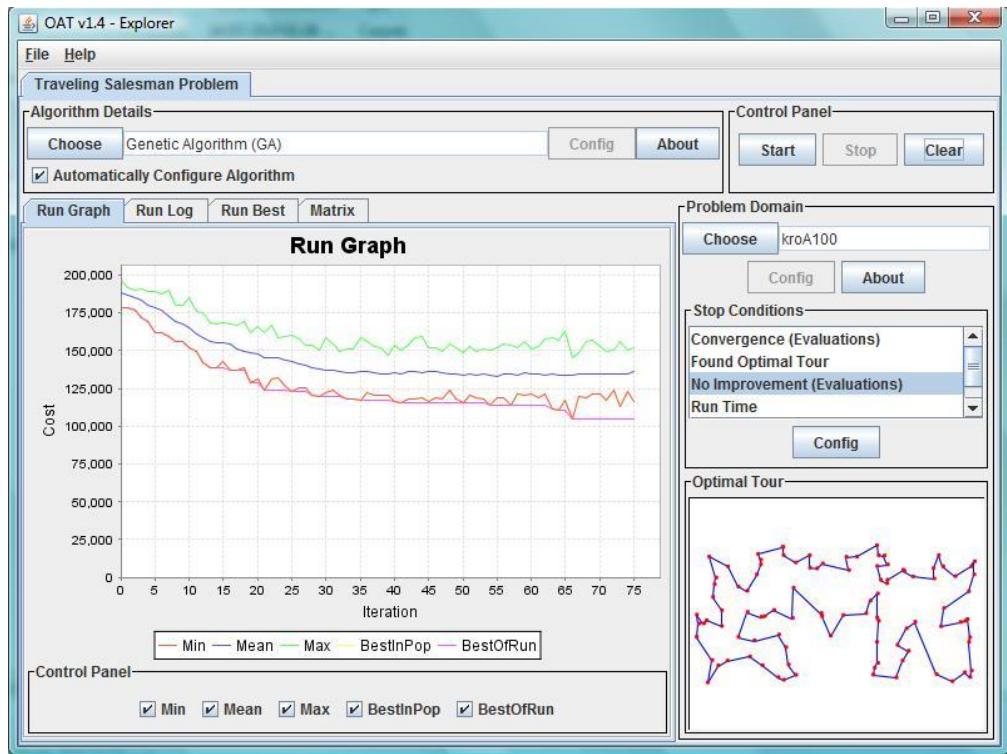


Figura 3.2 OAT Explorer

3.1.4 HeuristicLab Optimization Environment, [Wagner 2004]

Wagner 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 sólo se encuentra disponible el archivo ejecutable en [Heal 2014].

Este proyecto, se enfoca más en la independencia, 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

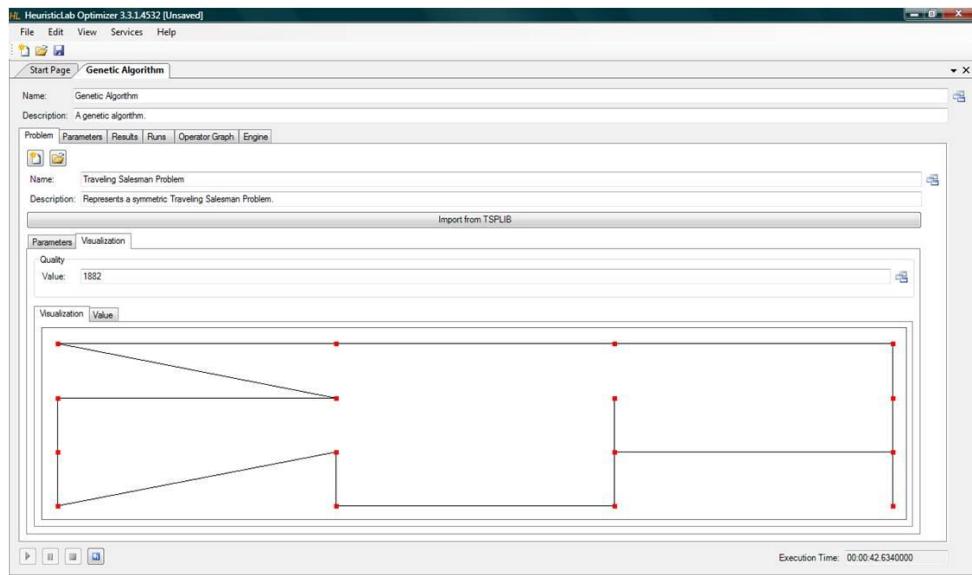


Figura 3.3 Ejemplo de ejecución en HeuristicLab Optimizer.

3.1.5 Visualizer for Metaheuristics Development Framework [V-MDF 2009]

V-MDF es un proyecto desarrollado por Halim [Halim 2005] que busca capturar una vista pictorial de la búsqueda de trayectorias y reportar 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 [V-MDF 2009].

3.1.6 VIZ Visualization for Analyzing Trajectory-Based Metaheuristic Search Algorithms, [VIZ 2010]

Halim [Halim 2006b, Halim 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. La Figura 3.4 muestra los cuatro componentes del entorno Viz SIMRA:

- a) *Fitness Landscape 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 y series de movimientos sin mejora.

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.

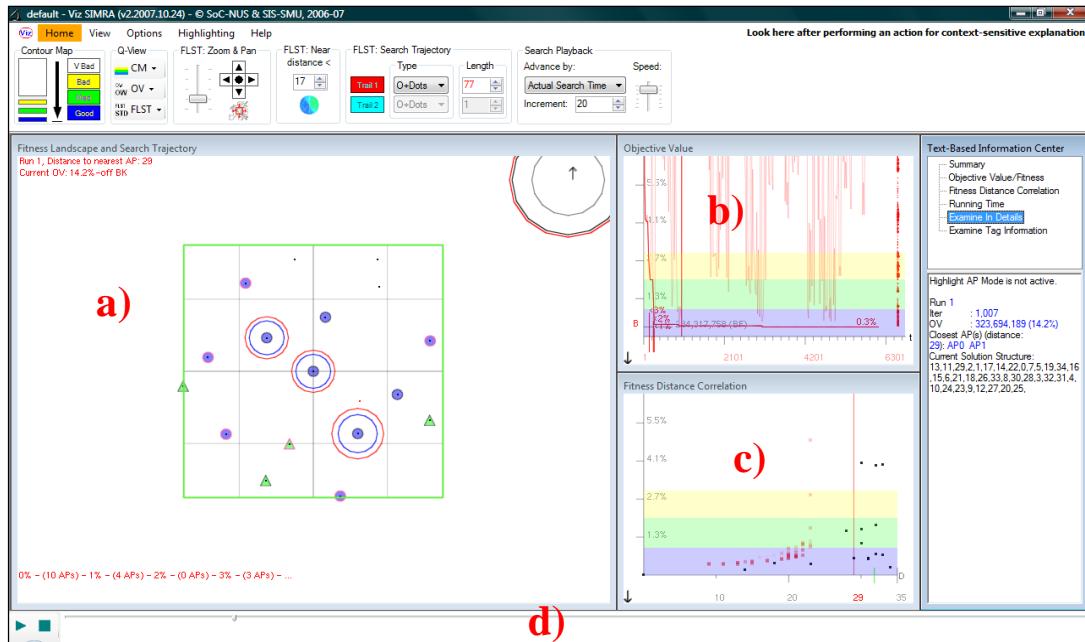


Figura 3.4 Entorno Viz SIMRA

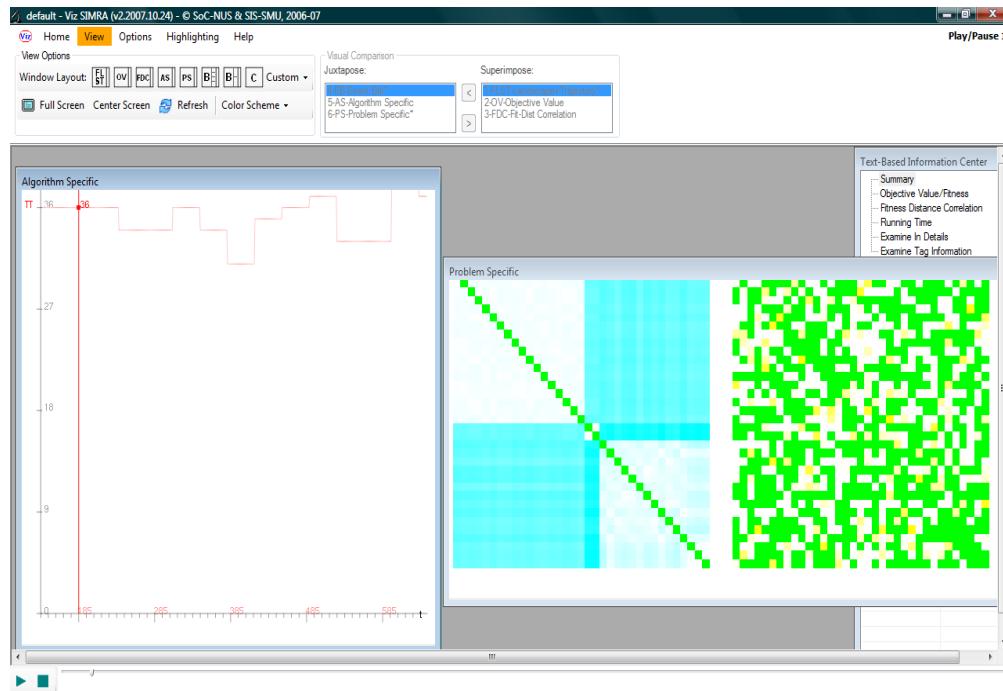


Figura 3.5 Visualizaciones en VIZ específicas del algoritmo y del problema.

VIZ se compone de 2 aplicaciones:

- Viz EW. Produce archivos de datos visuales (Visual Data Files, VDFs) que incluye la visualización de la trayectoria de la búsqueda. La Figura 3.6 muestra la interfaz gráfica de Viz EW.
- Viz SIMRA es la herramienta visual que utiliza el archivo creado en Viz EW.

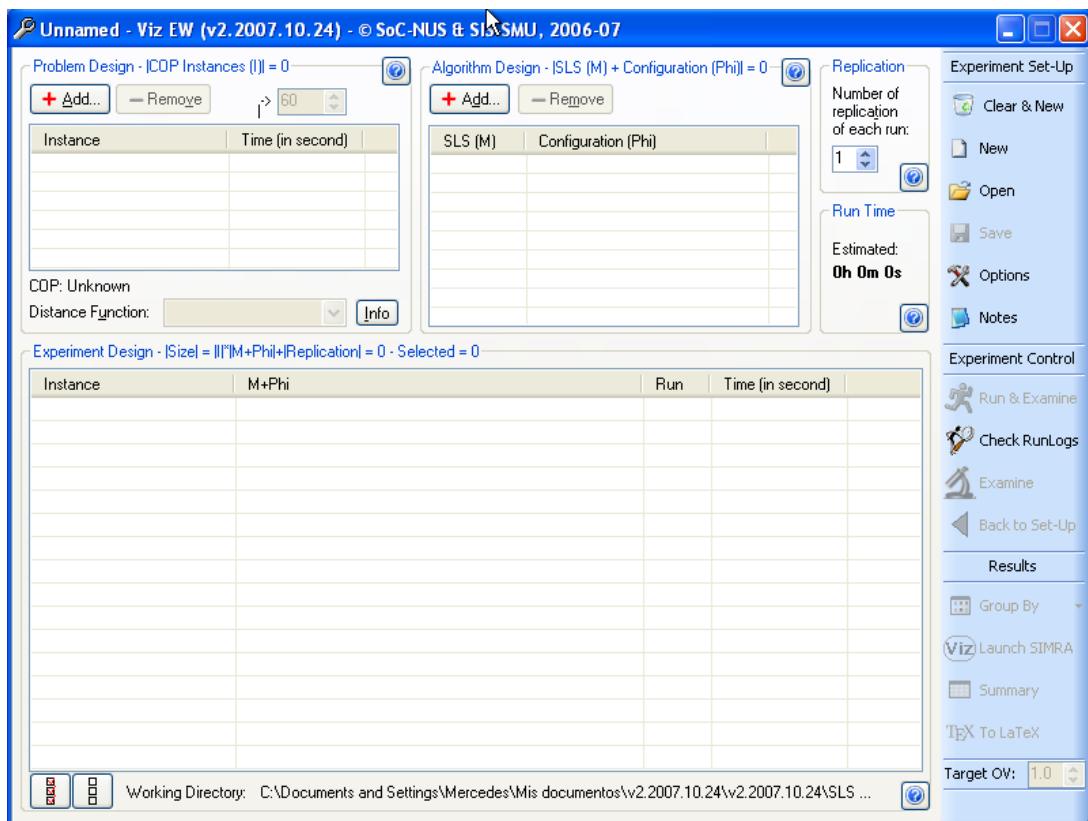


Figura 3.6 Entorno Viz EW

En Viz EW se agregan las instancias a resolver 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. Un archivo de ejemplo se muestra en la Figura 3.7.

```

100000      MAXIMUM_NUMBER_OF_ITERATIONS must always be the first line
500      UPDATE_RATE virtually no update
1      NON_IMPROVING_MOVES_TOLERANCE execute a strategy after
n*NON_IMPROVING_MOVES_TOLERANCE moves
3      NAVIGATION_DECISION 0-N/A, 1-Ro-TS-A (lower tabu tenure
range), 2-Ro-TS-B (RuinAndRecreate)
90      TTL w.r.t n (instance size)
20      TTD w.r.t n (instance size)
10;30 X    number of items w.r.t n (instance size)

```

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

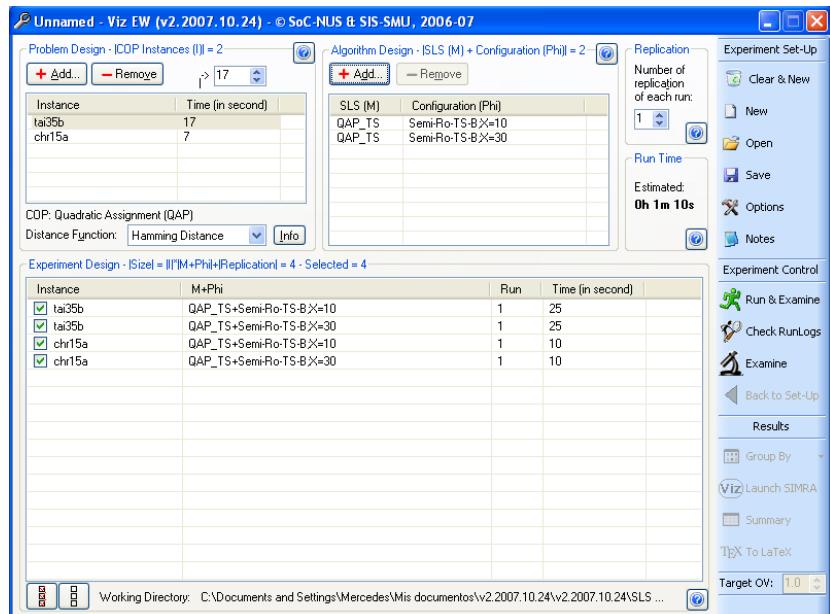


Figura 3.8 Casos de prueba resultantes del archivo de configuración.

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.

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.

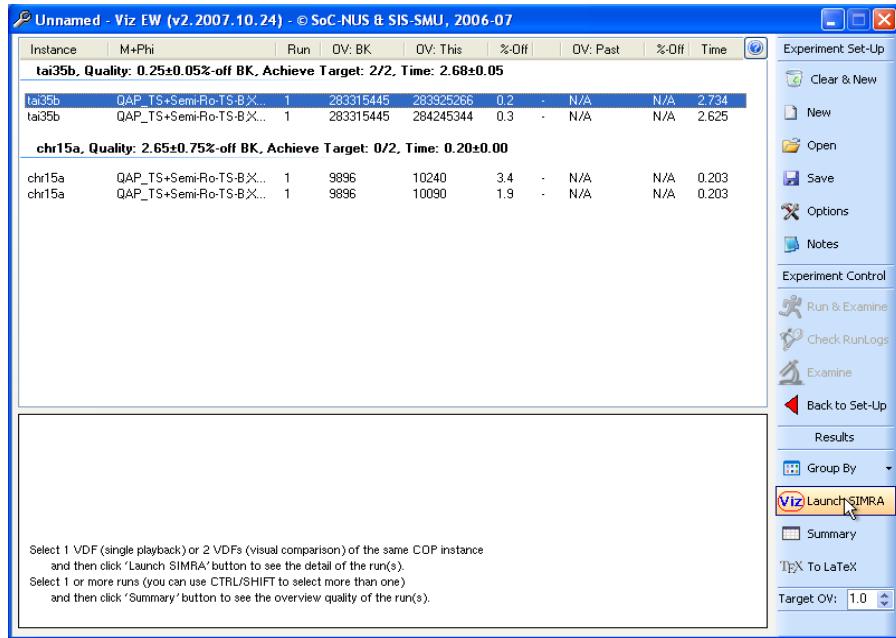


Figura 3.9 Resultados de la experimentación con diferentes configuraciones.

VIZ permite analizar los problemas de Asignación Cuadrática y el Problema del Agente Viajero. Para la solución de estos problemas, la herramienta usa algoritmos de búsqueda tabú, y hace caracterizaciones del problema (distance, bond distance, edge distance, etc.), el algoritmo (MaxIteration, Acceptance Criteria, Initial Solution, Perturbation, etc) y los resultados(average percentage-off, standard deviation, etc.), 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.

VIZ permite el análisis de nuestros propios algoritmos de búsqueda local, para hacer esto se deben seguir ciertas especificaciones. Esta herramienta se encuentra disponible [VIZ 2010].

3.2 Análisis estadístico de algoritmos aproximados

3.2.1 “A statistical test for comparing success rates” [Taillard 2003]

Este artículo presenta la prueba U, una prueba no-paramétrica que propone para dar respuesta a la pregunta que pueden hacer los investigadores: ¿Es una tasa de éxito de $a = n$ significativamente superior a una tasa de éxito de $b = m$? esto de interés para aquellos que deseen comparar diferentes algoritmos heurísticos 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.

3.2.2 “A study on the use of non-parametric tests for analyzing the evolutionary algorithms’ behaviour: a case study on the CEC’2005 Special Session on Real Parameter Optimization” [García 2009]

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 sólo 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 de 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* (Rönkkönen et al. 2005), *DMS-L-PSO* (Liang y Suganthan 2005), *EDA* (Yuan y

Gallagher 2005), *G-CMA-ES* (Auger y Hansen 2005^a), *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 los autores 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 sólo 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.

3.2.3 “Un tutorial sobre el uso de test estadísticos no paramétricos en comparaciones múltiples de metaheurísticas y algoritmos evolutivos” [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 los autores han 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 la prueba de Friedman como procedimiento para realizar comparaciones múltiples entre diferentes algoritmos. Así mismo se aplicaron las pruebas de Friedman Alineado y Quade como versiones avanzadas del mismo.

3.2.4 “Knowledge Extraction based on Evolutionary Learning” [KEEL 2014]

Knowledge Extraction based on Evolutionary Learning (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 [KEEL 2014]. La versión actual de KEEL está compuesta de las siguientes funcionalidades:

- Data management*: Esta parte está compuesta por un conjunto de herramientas que pueden ser usadas para construir nuevos datos, exportar e importar datos 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 Ventana del módulo “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 .

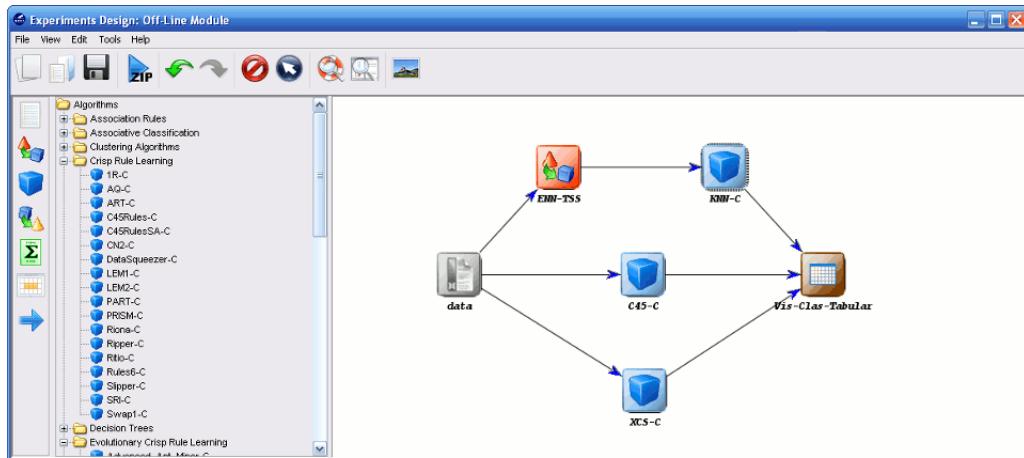


Figura 3.11 Ventana del módulo “Design of experiments”

c) *Design of imbalanced experiments*: El objetivo de esta funcionalidad es el diseño de la experimentación deseada sobre los conjuntos de datos seleccionados desequilibrados. La interfaz de esta sección se muestra en la Figura 3.12 .

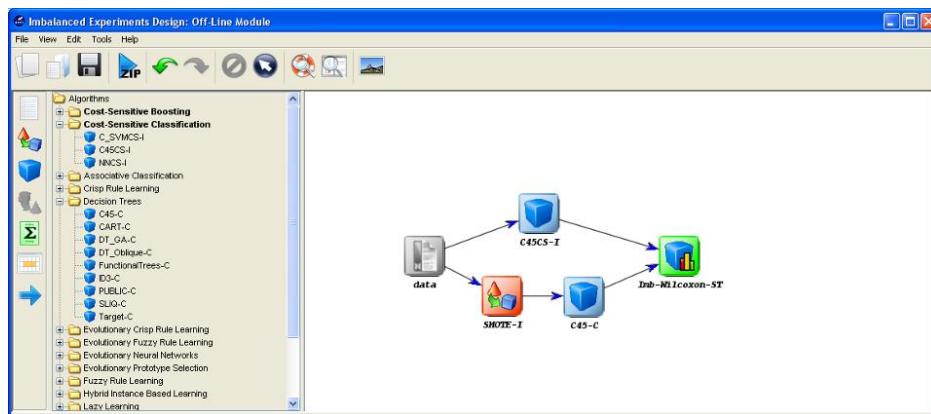


Figura 3.12 Ventana del módulo “Design of imbalanced experiments”

d) *Experimentation with multiple instance learning algorithms*: Esta funcionalidad permite al investigador 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 etiqueta 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.

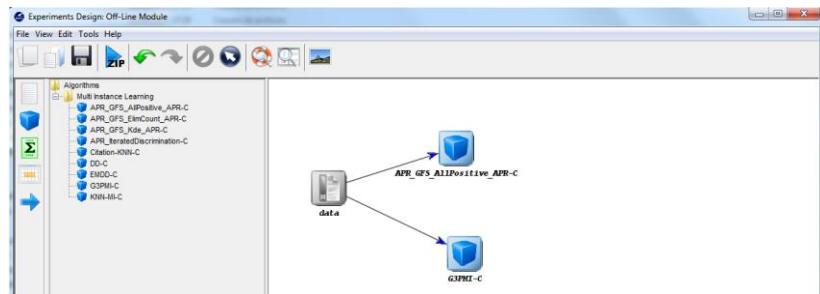


Figura 3.13 Ventana del módulo “Experimentation with multiple instance learning algorithms”

- 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 Figura 3.14.
- f) *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 .

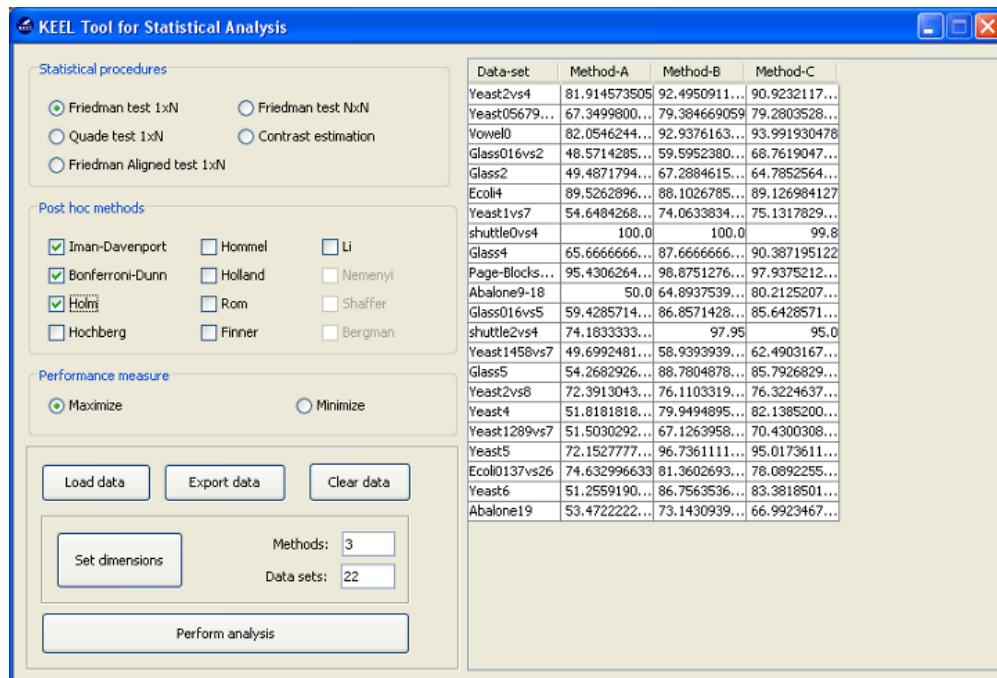


Figura 3.14 Ventana del módulo “Statistical tests”

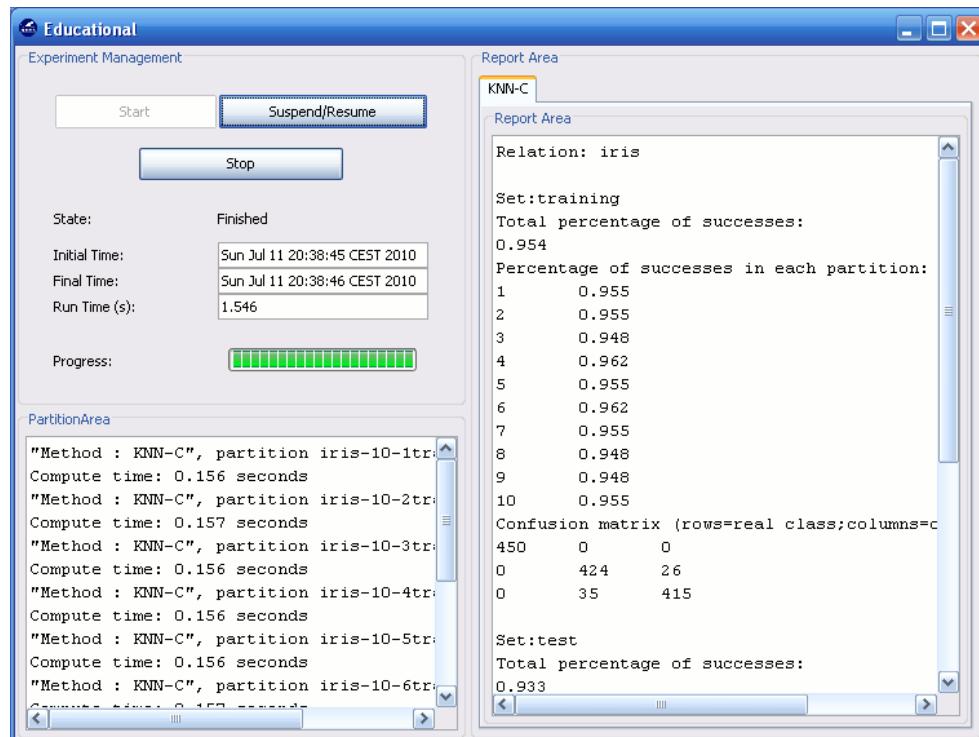


Figura 3.15 Ventana del módulo “Educational experiments”

3.3 Análisis experimental del proceso de optimización

En esta sección se describen los trabajos que han contribuido al proyecto general al cual está circunscrito el presente trabajo de tesis. Cabe mencionar, que todos están orientados al análisis experimental del proceso de optimización mencionado en la sección 2.2.

- 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, obtenida por regresión 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 2002].

- 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 2010].
- g) “Caracterización del proceso de optimización de algoritmos heurísticos aplicados al problema de empacado de objetos en contenedores”, en este trabajo se formula una metodología para el estudio integral del proceso de optimización, con la finalidad de identificar relaciones inherentes entre los factores que afectan al desempeño algorítmico.

La metodología propuesta combina métodos de análisis exploratorio de datos y de análisis causal en cuatro etapas: caracterización, refinación de características, estudio de relaciones y análisis de las características del proceso de optimización. El producto de este trabajo es un nuevo algoritmo de solución de BPP diseñado a partir del conocimiento obtenido del análisis experimental del proceso de optimización BPP. Este algoritmo es denominado GGA-CGT (Grouping Genetic Algorithm with Controlled Gene Transmission), *Algoritmo Genético de Agrupación con Transmisión de Genes Controlada* [Quiroz 2014].

3.4 VisTHAA [Castillo 2011]

“Evaluación de estrategias de mejora del desempeño de metaheurísticos aplicados a BPP vía diagnóstico visual”, en este trabajo se desarrolló una herramienta de visualización llamada VisTHAA, en la cual se implementaron técnicas visuales y descriptivas básicas que sirven de apoyo para el investigador en el proceso de diseño de algoritmos metaheurísticos.

Esta herramienta permite al usuario introducir instancias sin importar el formato de origen, el autor introduce y desarrolla la idea de un metalenguaje, el cual consta de un conjunto de palabras reservadas que permiten describir la instancia a través de la creación de un archivo denominado metainstance para indicar a VisTHAA cuántas y cuáles se van a introducir.

Otra de las funciones de VisTHAA es que permite hacer una caracterización de las instancias que puede ser llevada a cabo a través de índices así mismo permite visualizar las características de las instancias apoyado de gráficas. Este trabajo incorpora una visualización del comportamiento algorítmico mediante una gráfica donde el eje x representa el número de iteración y el eje y el valor de aptitud.

3.5 Comentarios finales

Después de hacer la revisión del estado del arte, se muestra un resumen de los trabajos revisados concentrado en las siguientes tablas haciendo una comparativa con el trabajo que se desarrolla en este proyecto.

La Tabla 3.1 muestra un resumen del estado del arte en análisis de desempeño de algoritmos heurísticos, donde se destacan las etapas del proceso de optimización algorítmica que han sido analizadas por los diferentes autores, también se indica si desarrollaron una herramienta para el análisis del proceso de optimización algorítmica y si hacen pruebas estadísticas del desempeño del algoritmo.

Tabla 3.1 – Estado del arte en análisis de desempeño de algoritmos heurísticos

Trabajo	Caracterización	Herramienta de análisis			Pruebas estadísticas
		Arquitectura	Entrada genérica	Caracterización visual	
Liefoghe 2009, ParadisEO	C, D	✓			
Callan 2007, BPP Visualization Tool	C	✓			
Halim 2005, V-MDF	P, C, D	✓			
Halim 2009, VIZ	P, C, D	✓		✓	
Brownlee 2007, OAT	P, D	✓			✓
Wagner 2004, HeuristicLab	C, D	✓			
Quiroz 2009	P, C, D, M				
Taillard 2003					✓
Derrac 2012					✓
Fernández 2004, KEEL		✓			✓
García 2009					✓
Castillo 2011 VisTHAA		✓	✓	✓	
Este trabajo, VisTHAA		✓	✓		✓

Caracterización: P=Estructura del problema, C=Conducta del algoritmo en ejecución, D=Desempeño final del algoritmo, M=modelo integral.

A continuación en la Tabla 3.2 se concentra la información de pruebas no paramétricas con una breve descripción de su uso, del tipo de comparaciones que se hacen con ellas, así mismo se indica qué trabajos revisados hacen uso de éstas pruebas no paramétricas.

Tabla 3.2 - Trabajos relacionados con el análisis estadístico de algoritmos aproximados

Prueba estadística	Propósito	Tipo de Comparación	Trabajos que la incluyen
Wilcoxon de rangos con signos	Responde lo siguiente: ¿Dos muestras representan dos diferentes poblaciones?	P	G, V
Prueba U	Determinar si una taza de éxito es realmente mejor que otra	P	T
Prueba de Signos Múltiples	Realizar un primer estudio preliminar de los datos, primer filtro para eliminar muestras que se encuentren fácilmente distinguibles como no óptimas	1xN	D
Estimación de Contraste	Calcular la estimación de las diferencias entre el desempeño de dos algoritmos, dando importancia a la mediana de sus resultados.	1xN	D
Prueba de Friedman	Considera que todos los problemas iguales en importancia, permite determinar si en un conjunto de pruebas existen diferencias significativas entre sí, y si las mismas se encuentran interrelacionadas.	1xN, NxN	G, D
Prueba de Friedman Alineado	Realizar comparaciones entre desempeños de algoritmos en un sólo conjunto, sin considerar las interrelaciones que puedan existir entre los conjuntos de la prueba completa.	1xN	D
Prueba de Quade	Saber si las diferencias entre las observaciones o datos registrados son significativamente grandes o no lo son, es decir, si son más difíciles.	1xN	D

Tipo de comparación: (**P**) Comparación por pares, (**1xN**) Múltiples comparaciones con método de control $1 \times N$, (**NxN**) Múltiples comparaciones ($N \times N$).

Trabajos: (**T**)Taillard 2003, (**G**) García et al 2009, (**D**) Derrac et. al 2012, (**K**) KEEL, (**V**) VisTHAA

Capítulo 4

Análisis estadístico

Los investigadores de diferentes áreas del conocimiento utilizan diversos recursos para el análisis multivariado de datos basado en pruebas de hipótesis. Para este propósito existen herramientas estadísticas comerciales (SPSS®, SAS®, Minitab®, Matlab, entre otras), así como versiones de uso libre (por ejemplo, DataPlot, R y Octave). Sin embargo, debido a la particularidad de cada campo de aplicación y a la naturaleza de los datos que se analizan, los investigadores generalmente hacen uso de más de una de estas herramientas de software para complementar el análisis.

En la literatura especializada se han encontrado recomendaciones sobre pruebas de hipótesis adecuadas para el análisis de desempeño de algoritmos aproximados (ver sección 3.2). Debido a la falta de literatura que las documente de manera integral, en este capítulo se revisan con un enfoque orientado a su implementación en la arquitectura propuesta para el análisis de algoritmos aproximados. En el Anexo C se da un ejemplo de uso de cada prueba de hipótesis reportada.

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” [Siegel 1995].

El interés de la inferencia estadística es el cómo obtener conclusiones acerca de grandes grupos de sujetos o de eventos, sobre la base de observaciones de pocos sujetos o de lo 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 si podemos inferir legítimamente que varios grupos difieren entre ellos.

A continuación se definen los elementos que intervienen en una prueba de hipótesis [Mendenhall 1997, Berenson 2001].

- **Hipótesis:** es una afirmación que se hace acerca de una o varias características de una población. Las características pueden ser los parámetros de una distribución de probabilidad predeterminada, seleccionada para la población.
- **Prueba de hipótesis:** es un procedimiento para decidir si una hipótesis se acepta como válida o se rechaza. Dos son las hipótesis que generalmente se contrastan: **hipótesis nula** (H_0) que es la hipótesis en la que se basa el procedimiento de contraste, y la **hipótesis alternativa** (H_a) que es la hipótesis que se acepta cuando se rechaza la nula y viceversa.
- **Estadístico de contraste:** Variable aleatoria usada para **estimar** un parámetro poblacional. Su distribución es conocida cuando la H_0 es verdadera.

- **Nivel de significancia (α):** Probabilidad de rechazar H_0 cuando es verdadera (margen de error permitido).
- **Región de aceptación y región crítica.** Conjunto de **valores del estadístico** que nos llevan a aceptar o rechazar H_0 respectivamente.
- **P-valor.** Probabilidad de que H_0 sea verdadera. Se acepta H_0 cuando $p\text{-valor} > \alpha$. En otras palabras es la menor α para el que la prueba rechazaría la hipótesis nula.

Una vez definidos los elementos de la prueba de hipótesis, a continuación se listan los pasos que se realizan para llevar a cabo una prueba de hipótesis.

Procedimiento general de una prueba de hipótesis

1. Formular hipótesis:
2. Fijar nivel de significación (generalmente $\alpha=0.05$)
3. Determinar el estadístico y su distribución
4. Determinar la región crítica usando α y la distribución esperada
5. Seleccionar una muestra y determinar el valor experimental del estadístico
6. Tomar la decisión: Rechazar H_0 si el valor experimental queda en la región crítica, en caso contrario no se rechaza H_0
7. Concluir

4.2 Pruebas estadísticas paramétricas y no paramétricas

Una *prueba estadística paramétrica* especifica 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 paramétrica* está basada en un modelo que especifica sólo condiciones muy generales y ninguna acerca de la forma específica de la distribución de la cual fue 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 la variable en estudio es continua; pero estas suposiciones son menores y más débiles que aquéllas 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.

4.2.1 Uso de pruebas no paramétricas para analizar algoritmos aproximados

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 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.

Es importante destacar que dado que los metaheurísticos son de origen aleatorio, se requiere del conjunto particular de ejecuciones para realizar una estimación estadística a través del análisis de las medias de las soluciones, errores u otros índices obtenidos en cada ejecución,

lo cual permite indicar en primera instancia qué algoritmo se comporta en general mucho mejor que otro. Dado que se desconoce la distribución de los datos obtenidos es fundamental la ejecución de pruebas estadísticas no-paramétricas, ya que éstas no presuponen una distribución de probabilidad para los datos. Estas pruebas permiten determinar sobre un conjunto de datos la posible relación que existen entre sus muestras, o en contraparte su independencia, comprobar si existen diferencias significativas, etc.

4.3 Pruebas no paramétricas

Para la descripción de las pruebas estadísticas presentadas en esta sección se consideró literatura enfocada al análisis estadístico de algoritmos [Derrac 2012]. En el Anexo C se ejemplifica cada prueba.

4.3.1 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 el valor $p < 0.05$.

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.

4.3.2 Prueba de signos múltiples

La prueba de signos múltiples es un sencillo procedimiento para realizar comparaciones $I \times N$ (método de control \times N métodos). Puede ser útil en casos en los que se desee realizar un primer estudio preliminar de los resultados. Se puede utilizar también como un primer filtro para eliminar muestras que se encuentren fácilmente distinguibles como no óptimas

Este test es una extensión de la prueba 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 desempeño del método de control en el conjunto i -ésimo. Representar con x_{ij} los valores de desempeño 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 desempeño, 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, (+) o (-), 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 respuesta 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.

4.3.3 Estimación de contraste

Es un procedimiento para realizar comparaciones $1 \times N$ (método de control \times N métodos). 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 desempeño de dos de ellos.

Un procedimiento que busca este propósito es la estimación de contraste, la cual supone que las diferencias esperadas entre desempeños de algoritmos son las mismas entre problemas. Asumimos que cada medición de desempeño se refleja como diferencias entre desempeños 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 (uv) de k algoritmos en el experimento, calcular la diferencia entre los desempeños de ambos en cada uno de los n problemas, $D_i(uv) = x_{iu} - x_{iv}$ donde $i = 1 \dots n$, $u = 1 \dots k$ y $v = 1 \dots k$. Se consideran sólo aquéllas diferencias donde $u < v$.
2. Buscar la mediana de cada conjunto de diferencias, Z_{uv} . Éste es un 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 :

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

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 desempeño 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.

4.3.4 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. Ésta es una prueba de comparación directa, es decir, sólo estudia un par de métodos.

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 (*rank*), 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 ésta prueba está usualmente incluido en paquetes de software estadísticos conocidos (SPSS, SAS, R, etc.).

La prueba de Wilcoxon de rangos con signos es más sensible que la prueba t. Asume commensurabilidad de diferencias, pero sólo 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.

4.3.5 Prueba de Friedman

En 1937 Milton Friedman 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 la universidad de Columbia durante la guerra resultó otro trabajo importante (1948) sobre muestreo secuencial: Friedman 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.

La prueba de Friedman es equivalente a una prueba ANOVA (Análisis de varianza) para dos factores y permite determinar si en un conjunto de pruebas existen diferencias significativas entre sí, y si las mismas se encuentran interrelacionadas.

La prueba 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).

La prueba 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 χ^2 con $k - 1$ 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 este 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. La Tabla B.3 permite consultar los valores críticos de este test.

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

4.3.6 Prueba de Quade

La prueba 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 desempeños de los algoritmos.

La prueba de Quade es una prueba no paramétrica, de $I \times N$ comparaciones estadísticas que lleva a cabo un análisis con rankings ponderados de las muestras de resultados. La prueba de Quade se utiliza para saber si las diferencias entre las observaciones o datos registrados son significativamente grandes o no lo son.

Para realizar esta prueba se recopilan datos de varios problemas para varios algoritmos y se les aplica la prueba de Friedman. La prueba de Friedman considera que todos los problemas son iguales en importancia. En cambio la prueba de Quade considera que algunos problemas son más difíciles o que las diferencias registradas en la ejecución de varios algoritmos sobre ellos son más distantes. Por lo tanto se supone que todos los problemas son diferentes en importancia.

El procedimiento para esta prueba es el siguiente:

- Encontrar los rankings r_{ij} de la misma forma que la prueba 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 desempeño 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, \dots, Q_n los rankings asignados a los problemas $1, 2, \dots, n$, 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 $j = 1, 2, \dots, k$.
- 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.

La prueba de Quade comprueba si la suma de rankings ponderados S_j son significativamente diferentes de 0.

Capítulo 5

Arquitectura propuesta para VisTHAA

En éste capítulo se describirá la estructura propuesta y desarrollada para la herramienta VisTHAA, señalando la aportación del desarrollo del módulo de análisis estadístico.

5.1 Planeación de la herramienta de diagnóstico visual

Para 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 éste proyecto se sentó las bases de lo que será VisTHAA, debido a que es un proyecto muy amplio en éste trabajo sólo se abordó un módulo. De manera específica, en éste proyecto se trabajó en un módulo para el análisis comparativo de algoritmos soportado en pruebas estadísticas no paramétricas.

En la Figura 5.1 se muestra un esquema de VisTHAA, con los módulos que ofrecerá al usuario y se puede observar resaltado el módulo que se desarrolló en este trabajo. 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, sólo se solicitará resultados de ejecuciones en archivos.

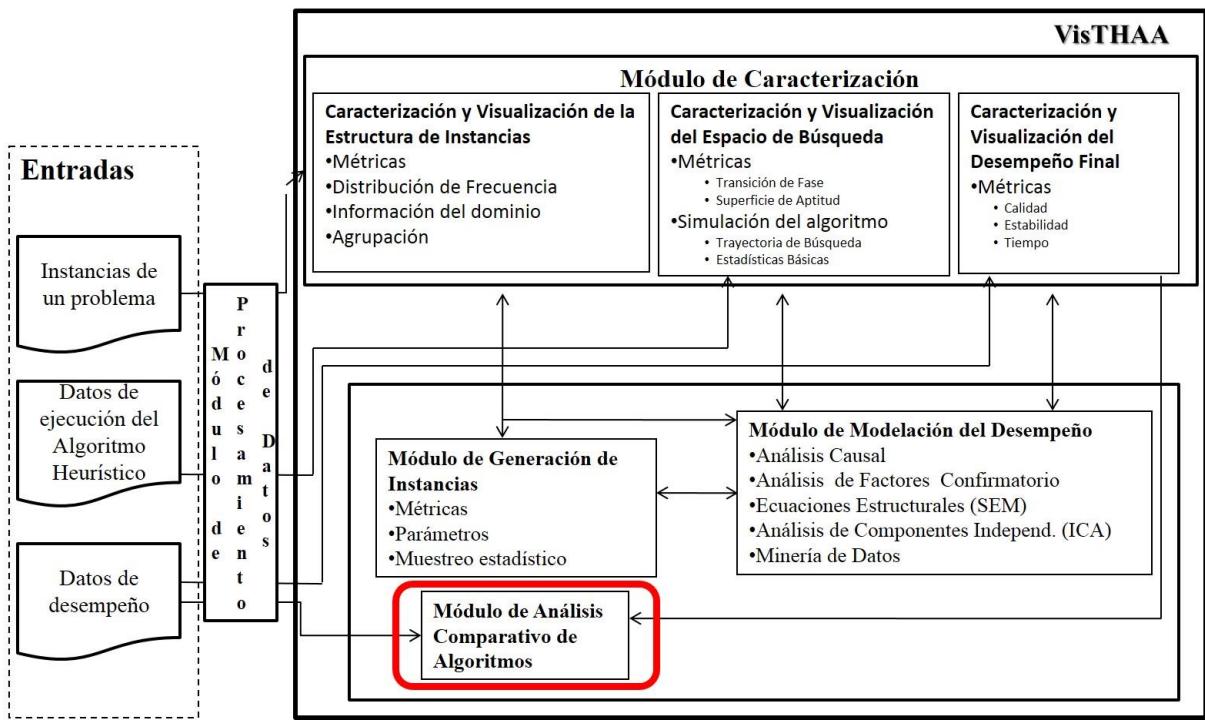


Figura 5.1 Esquema de módulos de VisTHAA

En la Figura 5.2 se muestra un diagrama de las interacciones entre datos que se tiene planeado para VistHAA, algunos de estos procesos ya se encuentran implementados y en funcionamiento. Para hacer la caracterización se necesitan las instancias e información del comportamiento algorítmico, para hacer este proceso la visualización de la información es una gran ayuda. Para hacer el análisis comparativo de algoritmos se necesita información referente al desempeño final del algoritmo, el análisis comparativo se vale de pruebas estadísticas.

Los procesos ya implementados se encuentran señalados. Estos procesos fueron desarrollados por Castillo [Castillo 2011] señalados con líneas rojas y este trabajo señalados con líneas azules.

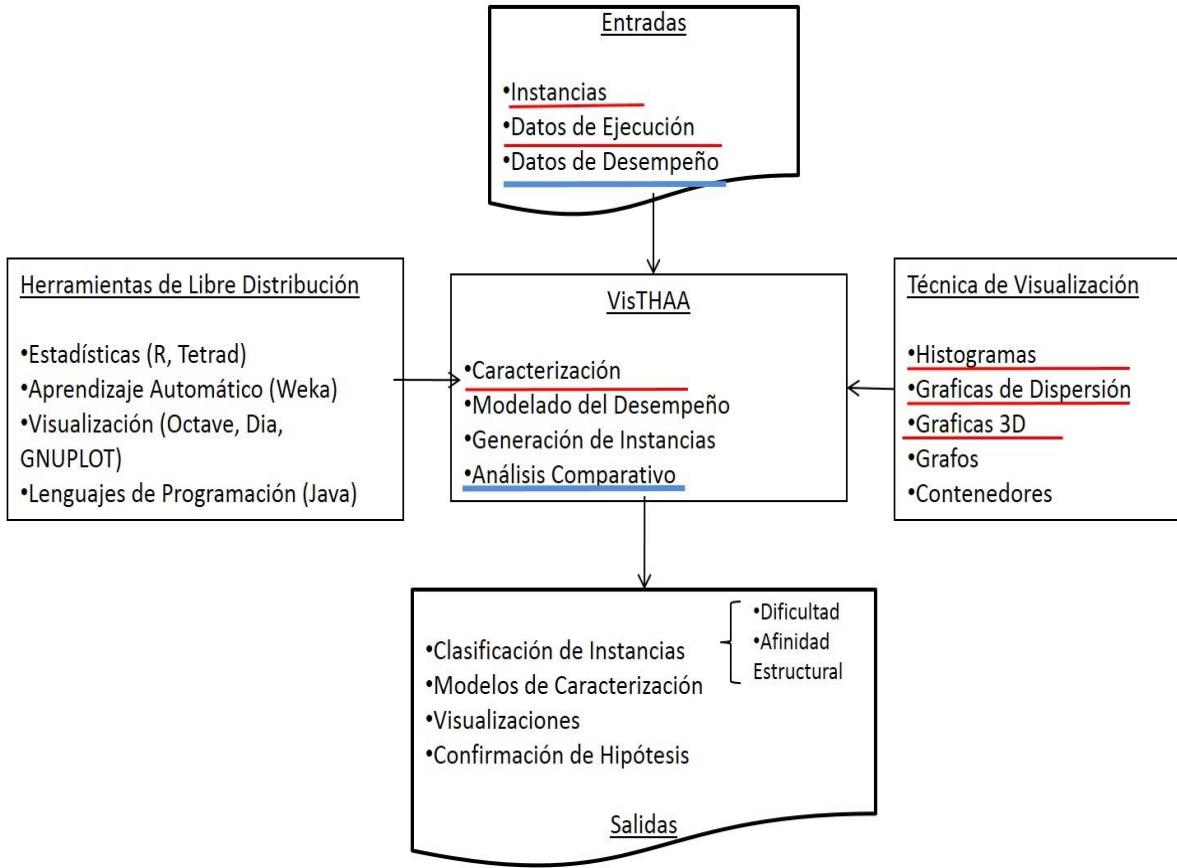


Figura 5.2 Herramientas para el Análisis Experimental del Proceso de Optimización de Algoritmos Heurísticos.

5.2 Módulos de VisTHAA

A partir de la arquitectura propuesta en esta tesis, se han desarrollado otros trabajos, entre ellos el presentado por Castillo [Castillo 2011], el cual desarrolló avances importantes en el módulo de entrada de datos, módulo de caracterización y visualización de la estructura de instancias del problema y en el módulo de caracterización y visualización del espacio de búsquedas. En esta sección se describen las características funcionales propuestas para VisTHAA y el estado actual de la herramienta, enfatizando el componente desarrollado para el análisis comparativo de algoritmos.

5.2.1 Módulo de caracterización y visualización

Caracterización y visualización de la estructura de instancias del problema

Este módulo se planteó para ayudar al investigador a obtener las características relevantes de los problemas de optimización NP-duros. Las características de interés se obtienen y visualizan en términos de: a) nuevas métricas o métricas tomadas de la literatura; b) la distribución de frecuencias de parámetros del problema, por ejemplo, puede ser de interés conocer y visualizar la distribución de los pesos de los objetos de una instancia de BPP en particular; c) información del dominio del problema proporcionada por el investigador; y d) su capacidad para asociar instancias afines mediante el uso de Minería de Datos o alguna otra técnica de agrupación.

Actualmente, el módulo le permite al usuario cargar un archivo el cual debe contener el nombre de las instancias o instancia que se desea analizar. Ya especificadas las instancias, el usuario puede seleccionar los índices o métricas que estén disponibles para las instancias leídas.

Caracterización del espacio de búsqueda

Este módulo se propone para caracterizar el proceso de búsqueda de un algoritmo de solución aproximada mediante métricas derivadas principalmente de dos tipos de estudios: a) análisis de transición de fase, para determinar que una instancia es difícil si la métrica la ubica en la zona de transición, que es el punto donde ocurre un cambio abrupto en el desempeño; y b) análisis de superficie de aptitud, para medir la variabilidad de las soluciones generadas durante la búsqueda. Además, se visualizará en forma simulada el proceso de solución, mostrando en el tiempo y en el espacio: a) el trazo de la trayectoria de búsqueda, por ejemplo expresada en desempeño; b) estadísticas básicas sobre los valores de las soluciones encontradas, por ejemplo la frecuencia, el mínimo, el máximo, y la cercanía a la mejor encontrada; y c) el valor de las métricas.

Actualmente, este módulo le permite al usuario cargar un archivo que contiene información del proceso de búsqueda. Otra funcionalidad incorporada en este módulo, es la visualización del comportamiento del algoritmo en el espacio de búsqueda, con esto se puede ver qué tan buena diversificación o intensificación posee el algoritmo, dando la posibilidad de rotar el gráfico de visualización para mostrar diferentes puntos del proceso de búsqueda.

Caracterización y visualización del desempeño final

Para la caracterización del desempeño de algoritmos heurísticos se consideró la incorporación de métricas que expresen tres tipos de información: calidad de la solución (la métrica más usual en bin packing es el cociente entre la solución obtenida y la mejor solución conocida), la consistencia de los resultados (generalmente se estima midiendo la desviación estándar de los resultados obtenidos) y el tiempo que le llevó al algoritmo en encontrar la mejor solución. Actualmente el módulo de caracterización tiene un componente parecido a una calculadora que permite incorporar nuevas métricas, lo cual da mayor flexibilidad al proceso de caracterización.

5.2.2 Módulo de modelación de desempeño algorítmico

La construcción de modelos de desempeño de metaheurísticos es una tarea compleja. Aunque este tipo de modelos tienen potencial para generar explicaciones del desempeño de metaheurísticas, raramente los estudios experimentales se han centrado en analizar las aportaciones e interacciones de las estrategias implicadas; siendo mayor el interés por los comparativos de eficiencia. Para este módulo se plantea la incorporación de técnicas exploratorias, tales como Análisis de Factores Confirmatorio, Ecuaciones Estructurales, Análisis de Componentes Independientes, Análisis Causal y Minería de Datos. De acuerdo con la literatura especializada, es posible visualizar, resumir y construir modelos de desempeño algorítmico con el uso de estas técnicas [Cohen 1995, Liu 1996, Feitelson 2006].

5.2.3 Módulo de generación de instancias

Problemas que han sido muy estudiados como el bin packing, requieren la reposición de instancias viejas por nuevas instancias que sean representativas del problema y de la gran escala. Algunas técnicas que han sido exploradas por investigadores del grupo al que se pertenece son: el uso de métricas de caracterización, especificación de rangos de parámetros y muestreo estadístico [Cruz 2004]. Estas técnicas están contempladas para su incorporación en la herramienta VisTHAA.

5.2.4 Módulo de procesamiento de datos

Lo que se describe en esta sección es el planteamiento y las bases que se sentaron para el módulo de procesamiento de datos, el cual fue desarrollado como parte de un trabajo de tesis relacionado [Castillo 2011]; se colaboró en el diseño de este módulo.

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 2008], se ha diseñado un metadato para poder describir en forma de texto la estructura 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 sirve 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.

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 es usada por el metalenguaje.

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.

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 sirve 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.

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.

Modelo de datos para Bin Packing Problem

En el ejemplo de la Figura 5.3, 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 5.3.a corresponde al archivo de procesamiento por lotes, al que denominamos *batch*. En éste se describe el número de instancias que serán procesadas con el archivo metadato de la Figura 5.3.b. El archivo metadato describe el formato de cómo se debe leer cada archivo de instancias de la Figura 5.3.c.

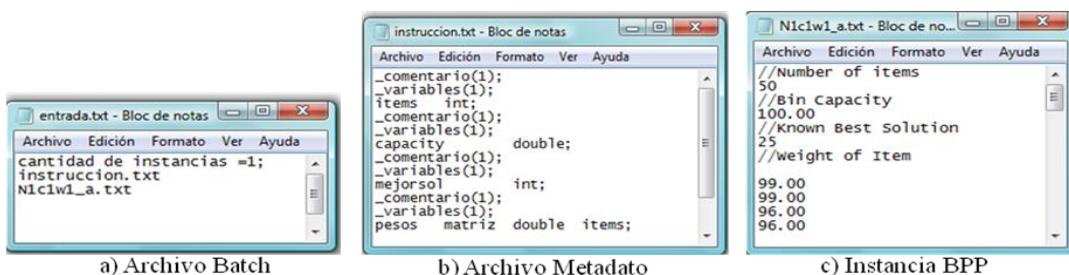


Figura 5.3 Archivos involucrados en el procesamiento de datos

5.2.5 Módulo de análisis comparativo de algoritmos

El análisis comparativo del desempeño de los algoritmos se puede hacer en este módulo, mediante las pruebas estadísticas no paramétricas que se describieron en la sección 4.3. A continuación se muestra la distribución de este módulo en la Figura 5.4. En ella se muestran las pruebas estadísticas no paramétricas que se añaden a este proyecto apoyando el módulo de análisis comparativo de algoritmos aproximados, cabe mencionar que las pruebas están implementadas para que el análisis de algoritmos no sea exclusivo.

En cada prueba se solicita un archivo que contiene resultados del desempeño de los algoritmos y en la interfaz se solicita información propia de cada prueba. Al final el usuario podrá ver el resultado de la prueba y si lo necesita, también se pueden visualizar los procedimientos intermedios de la prueba, de ser necesario, los resultados de las pruebas pueden ser almacenados en archivos de texto.

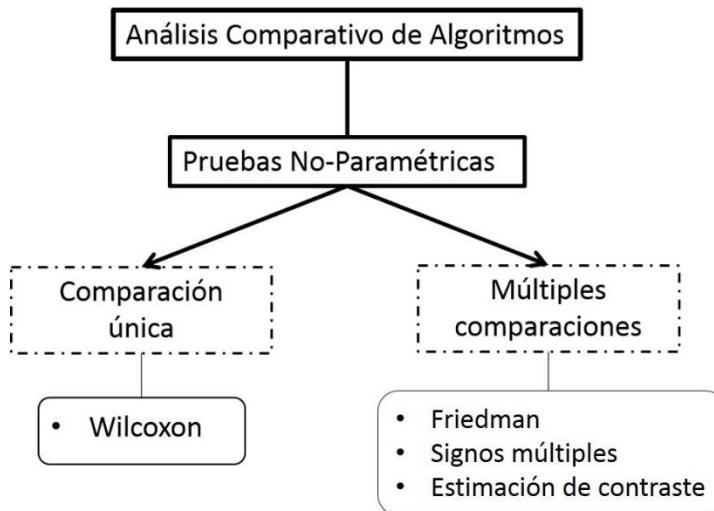


Figura 5.4 Módulo de análisis comparativo de algoritmos

En la Tabla 5.1 Se concentra la información de las pruebas estadísticas no paramétricas que se incluyen en VisTHAA, dando un resumen de lo ya descrito sobre estas pruebas en la sección 4.4

Tabla 5.1 - Pruebas estadísticas no-paramétricas que se incluyen a VisTHAA

Prueba estadística	Propósito	Tipo de Comparación
Wilcoxon	Responde lo siguiente: ¿dos muestras representan dos diferentes poblaciones?	P
Prueba de signos múltiples	Realiza un primer estudio preliminar de los datos, primer filtro para eliminar muestras que se encuentren fácilmente distinguibles como no óptimas	1×N
Estimación de contraste	Calcula la estimación de las diferencias entre el desempeño de dos algoritmos, dando importancia a la mediana de sus resultados.	1×N
Prueba de Friedman	Considera que todos los problemas son iguales en importancia, permite determinar si en un conjunto de pruebas existen diferencias significativas entre sí, y si las mismas se encuentran interrelacionadas.	1×N, N×N

Tipo de comparación: (**P**) Comparación por pares, (**1×N**) Múltiples comparaciones con método de control

1×N, (**N×N**) Múltiples comparaciones (**N×N**).

Capítulo 6

Experimentación y resultados

En esta sección se presenta la aplicación de pruebas estadísticas no paramétricas al estudio comparativo de algoritmos. Las pruebas estadísticas reportadas en el capítulo 4 se incorporaron en la herramienta VisTHAA de acuerdo a la arquitectura propuesta. Para verificar que la herramienta produce resultados correctos, se hizo la reproducción de pruebas no paramétricas reportadas en la literatura. Finalmente se muestran estudios comparativos entre algoritmos que resuelven problemas de empacado de objetos en contenedores.

6.1 Diseño experimental

Hardware

Todos los experimentos fueron realizados bajo las siguientes características de hardware: Procesador Intel (R) Core (TM) i5-3210M CPU a 2.50 GHz y 4 GB en memoria RAM.

Instancias

Las instancias se describirán en cada experimentación que se desarrolla en este capítulo.

6.2 Análisis funcional de VisTHAA

Estos experimentos se hicieron con el fin de demostrar que las pruebas hechas en VisTHAA se hicieran correctamente, para comprobarlo se comparó con los resultados de las pruebas descritas en Derrac [Derrac 2012]. Estas mismas pruebas se hicieron en el lenguaje y entorno de programación *R* para comprobar que los resultados coincidieran tanto con VisTHAA

como con lo reportado en el artículo. Los códigos de los procedimientos de los cálculos de las pruebas no paramétricas están descritos en el Anexo D.

Instancias

Para esta prueba, se usaron los datos de la Tabla E.1 en el Anexo E, la cual fue tomada de Derrac [Derrac 2012], donde se consideran los errores de 25 problemas presentados en la Sesión Especial de Optimización de Parámetros Reales del Congreso IEEE sobre Computación Evolutiva del 2005 (CEC'2005). La definición breve de estos 25 problemas se encuentra en el Anexo F, la descripción completa se encuentra en [Suganthan 2005].

Algoritmos

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). Como medida de desempeño, se ha recogido para cada problema el error medio obtenido en 50 ejecuciones de cada técnica.

Por lo anterior, tenemos para todas las pruebas que $k=4$ algoritmos y $n=25$ problemas

6.2.1 Prueba de Friedman

Objetivo del estudio

Se hizo la prueba de Friedman usando los datos del artículo de [Derrac 2012], para reproducir la prueba ahí descrita y de esta manera comparar los resultados obtenidos en VisTHAA y comprobar el correcto funcionamiento de VisTHAA. Ésta prueba también se reprodujo en el lenguaje de programación R. La instancia de entrada para esta prueba se encuentra en el Anexo G.1.

Resultados

En la Figura 6.1 se muestra una tabla que resume lo obtenido en el proceso de la prueba de Friedman, éste es el resultado del artículo. Se reprodujo la misma prueba en el lenguaje de programación R y los resultados se muestran en la Figura 6.2. En la Figura 6.3 se muestra el resultado de la prueba de Friedman obtenido por VisTHAA para estos mismos datos de entrada de Derrac.

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 6.1 Rankings de la prueba de Friedman, Derrac 2012

Algoritmo	PSO	SSGA	SS-BLX	DE-EXP
F1	3	2	4	1
F2	3	2	4	1
F3	2	3	4	1
F4	3	2	4	1
F5	4	3	2	1
F6	4	2	3	1
F7	1	2.5	4	2.5
F8	2.5	2.5	2.5	2.5
F9	4	1	3	2
F10	3	4	2	1
F11	4	3	2	1
F12	4	3	2	1
F13	4	3	1	2
F14	4	1	2	3
F15	4	2	1	3
F16	4	2	1	3
F17	4	2	1	3
F18	4	3	2	1
F19	3	4	2	1
F20	3	4	2	1
F21	4	3	1	2
F22	4	2	1	3
F23	4	3	1	2
F24	4	2	3	1
F25	1	3	4	2
T Ranking	84.5	64	58.5	43
Rj	3.38	2.56	2.34	1.72

Figura 6.2 Ejecución en R. Tabla de rankings y valor Rj para cada algoritmo.

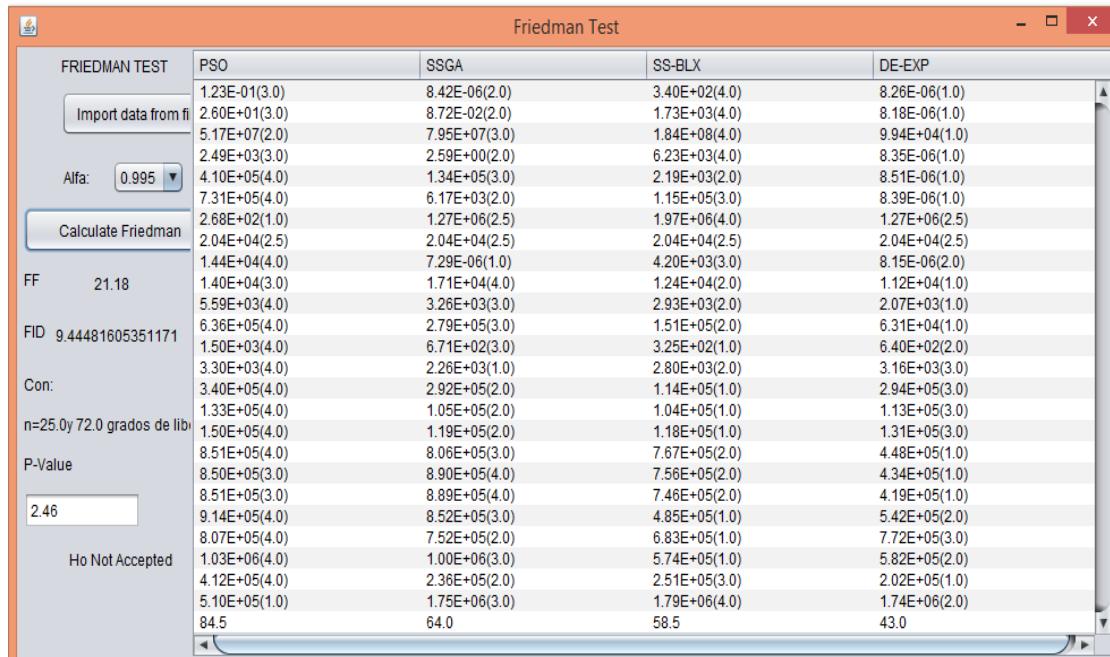


Figura 6.3 Resultados en VisTHAA de prueba de Friedman con datos de Derrac 2012

Análisis de resultados

Al comparar los resultados se puede ver que son iguales por lo que se puede comprobar que VisTHAA hace la prueba correctamente, el objetivo de la prueba de funcionalidad se cumple para ésta prueba.

La hipótesis nula para la prueba de Friedman indica que todos los algoritmos se comportan similarmente, por lo que sus rankings R_j deben ser similares. Se puede concluir para esta prueba que con cuatro algoritmos y 25 conjuntos, FID se distribuye de acuerdo a una distribución F con $4-1 = 3$ y $(4-1)(25-1) = 72$ grados de libertad. El p-valor calculado usando la distribución $F(3, 72)$ es 2,46E-05, por lo que la hipótesis nula se rechaza con una alta probabilidad.

6.2.2 Prueba de signos múltiples

Objetivo del estudio

Se hizo la prueba de signos múltiples usando los datos del artículo de [Derrac 2012], para reproducir la prueba ahí descrita y de esta manera comparar los resultados obtenidos en VisTHAA y así comprobar la correcta funcionalidad de la herramienta. Ésta prueba también se realizó en el lenguaje de programación R. La prueba se hace tomando al algoritmo DE-EXP como método de control así como lo hacen en el artículo de referencia. La instancia usada para ésta prueba se encuentra en el Anexo G.1.

Resultados

En la Figura 6.4 se muestra una tabla que resume lo obtenido en el proceso de la prueba de signos múltiples, éste es el resultado del artículo. Esta prueba también se reprodujo en el lenguaje de programación R y los resultados se muestran en la Figura 6.5. En la Figura 6.6 se muestra la reproducción de la prueba de signos múltiples en VisTHAA, de esta manera verificamos que la herramienta hace la prueba correctamente.

Algoritmo Orden	DE-EXP 1 (control)	PSO 2	SSGA 3	SS-BLX 4
F1	8,26E-06	1,23E-01 (-)	8,42E-06 (-)	3,40E+02 (-)
F2	8,18E-06	2,60E+01 (-)	8,72E-02 (-)	1,73E+03 (-)
F3	9,94E+04	5,17E+07 (-)	7,95E+07 (-)	1,84E+08 (-)
F4	8,35E-06	2,49E+03 (-)	2,59E+00 (-)	6,23E+03 (-)
F5	8,51E-06	4,10E+05 (-)	1,34E+05 (-)	2,19E+03 (-)
F6	8,39E-06	7,31E+05 (-)	6,17E+03 (-)	1,15E+05 (-)
F7	1,27E+06	2,68E+02 (+)	1,27E+06 (=)	1,97E+06 (-)
F8	2,04E+04	2,04E+04 (=)	2,04E+04 (=)	2,04E+04 (=)
F9	8,15E-06	1,44E+04 (-)	7,29E-06 (+)	4,20E+03 (-)
F10	1,12E+04	1,40E+04 (-)	1,71E+04 (-)	1,24E+04 (-)
F11	2,07E+03	5,59E+03 (-)	3,26E+03 (-)	2,93E+03 (-)
F12	6,31E+04	6,36E+05 (-)	2,79E+05 (-)	1,51E+05 (-)
F13	6,40E+02	1,50E+03 (-)	6,71E+02 (-)	3,25E+02 (+)
F14	3,16E+03	3,30E+03 (-)	2,26E+03 (+)	2,80E+03 (+)
F15	2,94E+05	3,40E+05 (-)	2,92E+05 (+)	1,14E+05 (+)
F16	1,13E+05	1,33E+05 (-)	1,05E+05 (+)	1,04E+05 (+)
F17	1,31E+05	1,50E+05 (-)	1,19E+05 (+)	1,18E+05 (+)
F18	4,48E+05	8,51E+05 (-)	8,06E+05 (-)	7,67E+05 (-)
F19	4,34E+05	8,50E+05 (-)	8,90E+05 (-)	7,56E+05 (-)
F20	4,19E+05	8,51E+05 (-)	8,89E+05 (-)	7,46E+05 (-)
F21	5,42E+05	9,14E+05 (-)	8,52E+05 (-)	4,85E+05 (+)
F22	7,72E+05	8,07E+05 (-)	7,52E+05 (+)	6,83E+05 (+)
F23	5,82E+05	1,03E+06 (-)	1,00E+06 (-)	5,74E+05 (+)
F24	2,02E+05	4,12E+05 (-)	2,36E+05 (-)	2,51E+05 (-)
F25	1,74E+06	5,10E+05 (+)	1,75E+06 (-)	1,79E+06 (-)
# (+)	-	2	6	8
# (-)	-	22	17	16
$R_j \alpha = 0,1$	-	7	6	7
$R_j \alpha = 0,05$	-	6	6	6

Figura 6.4 Prueba de signos múltiples, Derrac 2012

	CONTROL										
Algoritmo	DE-EXP	PSO	DIF	SIGNO	SSGA	DIF	SIGNO	SS-BLX	DIF	SIGNO	
F1	0.00000826	0.123	-0.12299174	-	0.00000842	-0.00000016	-	340	-339.999992	-	
F2	0.00000818	26	-25.99999182	-	0.0872	-0.08719182	-	1730	-1729.999999	-	
F3	99400	51700000	-51600600	-	79500000	-79400600	-	184000000	-183900600	-	
F4	0.00000835	2490	-2489.99999	-	2.59	-2.58999165	-	6230	-6229.99999	-	
F5	0.00000851	410000	-410000	-	134000	-134000	-	2190	-2189.99999	-	
F6	0.00000839	731000	-731000	-	6170	-6169.99999	-	115000	-115000	-	
F7	1270000	268	1269732	+	1270000	0	=	1970000	-700000	-	
F8	20400	20400	0	=	20400	0	=	20400	0	=	
F9	0.00000815	14400	-14400	-	0.00000729	0.00000086	+	4200	-4199.99999	-	
F10	11200	14000	-2800	-	17100	-5900	-	12400	-1200	-	
F11	2070	5590	-3520	-	3260	-1190	-	2930	-860	-	
F12	63100	636000	-572900	-	279000	-215900	-	151000	-87900	-	
F13	640	1500	-860	-	671	-31	-	325	315	+	
F14	3160	3300	-140	-	2260	900	+	2800	360	+	
F15	294000	340000	-46000	-	292000	2000	+	114000	180000	+	
F16	113000	133000	-20000	-	105000	8000	+	104000	9000	+	
F17	131000	150000	-19000	-	119000	12000	+	118000	13000	+	
F18	448000	851000	-403000	-	806000	-358000	-	767000	-319000	-	
F19	434000	850000	-416000	-	890000	-456000	-	756000	-322000	-	
F20	419000	851000	-432000	-	889000	-470000	-	746000	-327000	-	
F21	542000	914000	-372000	-	852000	-310000	-	485000	57000	+	
F22	772000	807000	-35000	-	752000	20000	+	683000	89000	+	
F23	582000	1030000	-448000	-	1000000	-418000	-	574000	8000	+	
F24	202000	412000	-210000	-	236000	-34000	-	251000	-49000	-	
F25	1740000	510000	1230000	+	1750000	-10000	-	1790000	-50000	-	
Rj	Sum(+)			2			6			8	
	Sum(-)			22			17			16	
	a=0.1			7			6			7	
	a=0.05			6			6			6	

Figura 6.5 Prueba de signos múltiples en R

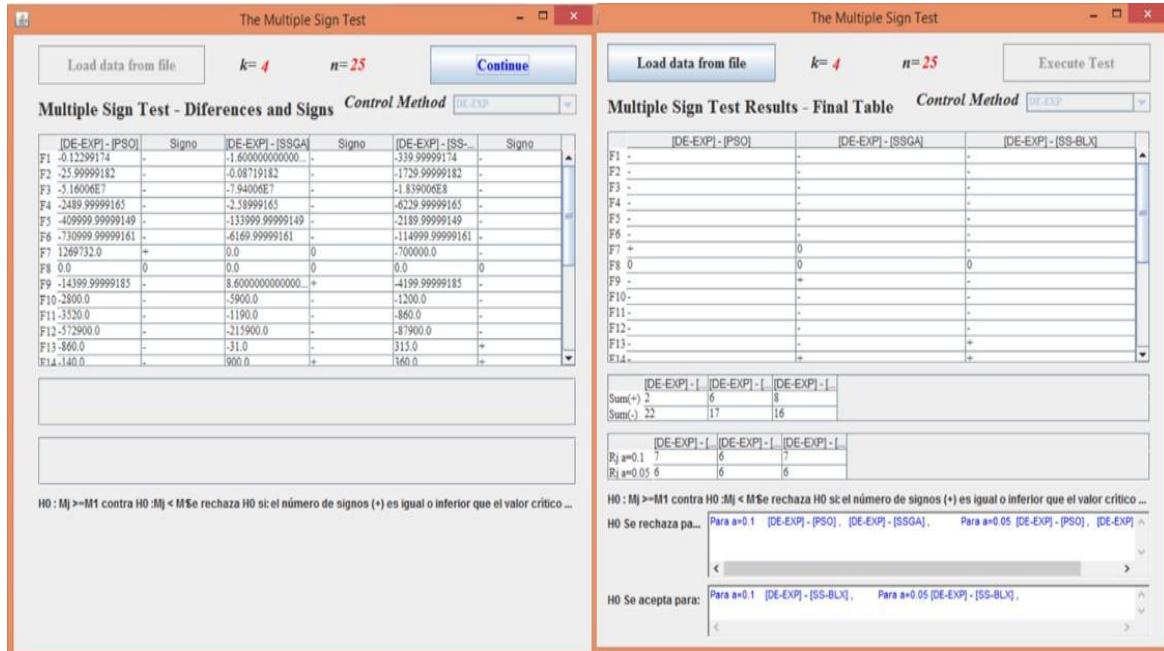


Figura 6.6 Prueba de signos múltiples en VisTHAA

Análisis de resultados

Los resultados obtenidos por visTHAA son iguales a los obtenidos en la ejecución en R y a los reportados en el artículo de referencia, por lo que el objetivo de la prueba de funcionalidad se cumple para ésta prueba.

La interpretación de esta prueba depende de lo siguiente. El punto crítico es definido por R_j , el cual representa el punto crítico de aceptación/rechazo de la hipótesis para la prueba del método de control con el método j . Se busca en la Tabla B.2 el valor crítico de R_j para cada método, donde:

- n es igual a la suma de signos (+) y (-) en el método
- m es el número de métodos a probar (sin contar el de control)

Aceptación y Rechazo:

Para $H_0: M_j \geq M_1$ Se aceptará H_0 si la suma de signos (+) del método j es superior al punto crítico

Para $H_0: M_j \leq M_1$ Se aceptará H_0 si la suma de signos (-) del método j es superior al punto crítico

Teniendo lo anterior podemos ver que:

$m=3$; para PSO $n=24$, para SSGA $n=23$ y para DE-EXP $n=24$

$H_0: M_j \geq M_1$

Para PSO y SSGA H_0 es rechazada, entonces concluimos que el método de control DE-EXP presenta un mejor desempeño que ellos.

Para SS-BLX H_0 es aceptada, entonces podemos ver que este metodo tiene un desempeño equivalente al método de control DE-EXP.

6.2.3 Prueba estimación de contraste

Objetivo del estudio

Se hizo la prueba estimación de contraste usando los datos del artículo de [Derrac 2012], para reproducir la prueba ahí descrita y de esta manera comparar los resultados obtenidos en VisTHAA para comprobar la correcta funcionalidad de VisTHAA. Esta prueba también se reprodujo en el lenguaje de programación R. Para tener mayor certeza de los resultados correctos para ésta prueba. La instancia usada para ésta prueba se encuentra en el Anexo G.1.

Resultados

Las Figuras 6.7 y 6.8 son tablas de los resultados obtenidos en el proceso de la prueba de estimación de contraste en el artículo. Los resultados de la prueba reproducida en R se muestran en la Figura 6.9 y 6.10. En la Figura 6.11 se muestra el resultado de la prueba de estimación de contraste obtenido por VisTHAA para estos mismos datos de entrada de Derrac. Al comparar los resultados se puede ver que son iguales por lo que se puede comprobar que VisTHAA hace la prueba correctamente.

Dif.	$D_{i(12)}$	$D_{i(13)}$	$D_{i(14)}$	$D_{i(23)}$	$D_{i(24)}$	$D_{i(34)}$
F1	1,23E-01	-3,40E+02	1,23E-01	-3,40E+02	1,60E-07	3,40E+02
F2	2,59E+01	-1,70E+03	2,59E+01	-1,73E+03	8,72E-02	1,73E+03
F3	-2,77E+07	-1,33E+08	5,16E+07	-1,05E+08	7,94E+07	1,84E+08
F4	2,49E+03	-3,74E+03	2,49E+03	-6,23E+03	2,58E+00	6,23E+03
F5	2,75E+05	4,07E+05	4,09E+05	1,32E+05	1,34E+05	2,18E+03
F6	7,25E+05	6,17E+05	7,31E+05	-1,08E+05	6,17E+03	1,14E+05
F7	-1,27E+06	-1,97E+06	-1,26E+06	-6,95E+05	6,00E+03	7,01E+05
F8	6,00E+01	8,00E+01	5,00E+01	2,00E+01	-1,00E+01	-3,00E+01
F9	1,44E+04	1,02E+04	1,44E+04	-4,19E+03	-8,65E-07	4,19E+03
F10	-3,08E+03	1,65E+03	2,86E+03	4,73E+03	5,94E+03	1,21E+03
F11	2,34E+03	2,66E+03	3,52E+03	3,26E+02	1,19E+03	8,62E+02
F12	3,57E+05	4,86E+05	5,73E+05	1,29E+05	2,16E+05	8,75E+04
F13	8,32E+02	1,18E+03	8,63E+02	3,47E+02	3,10E+01	-3,16E+02
F14	1,04E+03	5,08E+02	1,46E+02	-5,32E+02	-8,94E+02	-3,62E+02
F15	4,78E+04	2,26E+05	4,58E+04	1,78E+05	-2,00E+03	-1,80E+05
F16	2,80E+04	2,92E+04	2,08E+04	1,20E+03	-7,20E+03	-8,40E+03
F17	3,12E+04	3,14E+04	1,85E+04	2,00E+02	-1,27E+04	-1,29E+04
F18	4,49E+04	8,44E+04	4,03E+05	3,95E+04	3,58E+05	3,19E+05
F19	-4,02E+04	9,42E+04	4,16E+05	1,34E+05	4,56E+05	3,21E+05
F20	-3,84E+04	1,05E+05	4,32E+05	1,43E+05	4,71E+05	3,28E+05
F21	6,16E+04	4,29E+05	3,72E+05	3,67E+05	3,10E+05	-5,69E+04
F22	5,52E+04	1,24E+05	3,51E+04	6,91E+04	-2,01E+04	-8,92E+04
F23	2,40E+04	4,54E+05	4,46E+05	4,30E+05	4,22E+05	-8,40E+03
F24	1,76E+05	1,61E+05	2,10E+05	-1,53E+04	3,40E+04	4,93E+04
F25	-1,24E+06	-1,28E+06	-1,23E+06	-4,70E+04	5,00E+03	5,20E+04
Med.	2,49E+03	2,92E+04	2,08E+04	3,47E+02	1,19E+03	1,73E+03

Figura 6.7 Prueba estimación de contraste, Derrac 2012

	PSO	SSGA	SS BLX	DE-EXP
PSO	0	1,31E+04	1,98E+04	1,86E+04
SSGA	-1,31E+04	0	6,67E+03	5,49E+03
SS BLX	-1,98E+04	-6,67E+03	0	-1,17E+03
DE-EXP	-1,86E+04	-5,49E+03	1,17E+03	0

Figura 6.8 Estimador de contraste, Derrac 2012

Di(1, 2)	Di(1, 3)	Di(1, 4)	Di(2, 3)	Di(2, 4)	Di(3, 4)
-3.40E+02	1.23E-01	-3.40E+02	3.40E+02	1.23E-01	-3.40E+02
-1.73E+03	2.59E+01	-1.70E+03	1.76E+03	2.59E+01	-1.73E+03
-1.84E+08	-2.78E+07	-2.12E+08	1.56E+08	-2.78E+07	-1.84E+08
-6.23E+03	2.49E+03	-3.74E+03	8.72E+03	2.49E+03	-6.23E+03
-2.19E+03	2.76E+05	2.74E+05	2.78E+05	2.76E+05	-2.19E+03
-1.15E+05	7.25E+05	6.10E+05	8.40E+05	7.25E+05	-1.15E+05
-7.00E+05	-1.27E+06	-1.97E+06	-5.70E+05	-1.27E+06	-7.00E+05
0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
-4.20E+03	1.44E+04	1.02E+04	1.86E+04	1.44E+04	-4.20E+03
-1.20E+03	-3.10E+03	-4.30E+03	-1.90E+03	-3.10E+03	-1.20E+03
-8.60E+02	2.33E+03	1.47E+03	3.19E+03	2.33E+03	-8.60E+02
-8.79E+04	3.57E+05	2.69E+05	4.45E+05	3.57E+05	-8.79E+04
3.15E+02	8.29E+02	1.14E+03	5.14E+02	8.29E+02	3.15E+02
3.60E+02	1.04E+03	1.40E+03	6.80E+02	1.04E+03	3.60E+02
1.80E+05	4.80E+04	2.28E+05	-1.32E+05	4.80E+04	1.80E+05
9.00E+03	2.80E+04	3.70E+04	1.90E+04	2.80E+04	9.00E+03
1.30E+04	3.10E+04	4.40E+04	1.80E+04	3.10E+04	1.30E+04
-3.19E+05	4.50E+04	-2.74E+05	3.64E+05	4.50E+04	-3.19E+05
-3.22E+05	-4.00E+04	-3.62E+05	2.82E+05	-4.00E+04	-3.22E+05
-3.27E+05	-3.80E+04	-3.65E+05	2.89E+05	-3.80E+04	-3.27E+05
5.70E+04	6.20E+04	1.19E+05	5.00E+03	6.20E+04	5.70E+04
8.90E+04	5.50E+04	1.44E+05	-3.40E+04	5.50E+04	8.90E+04
8.00E+03	3.00E+04	3.80E+04	2.20E+04	3.00E+04	8.00E+03
-4.90E+04	1.76E+05	1.27E+05	2.25E+05	1.76E+05	-4.90E+04
-5.00E+04	-1.24E+06	-1.29E+06	-1.19E+06	-1.24E+06	-5.00E+04
-1.73E+03	2.49E+03	1.40E+03	8.72E+03	2.49E+03	-1.73E+03

Figura 6.9 Ejecución en R. Estimación de contraste. Tabla de diferencias y medianas en R

m1	1.29E+04				
m2	-2.38E+02				
m3	-6.90E+03				
m4	-5.73E+03				

Figura 6.10 Ejecución en R. Estimación de contraste. Tabla de estimador de contraste en R

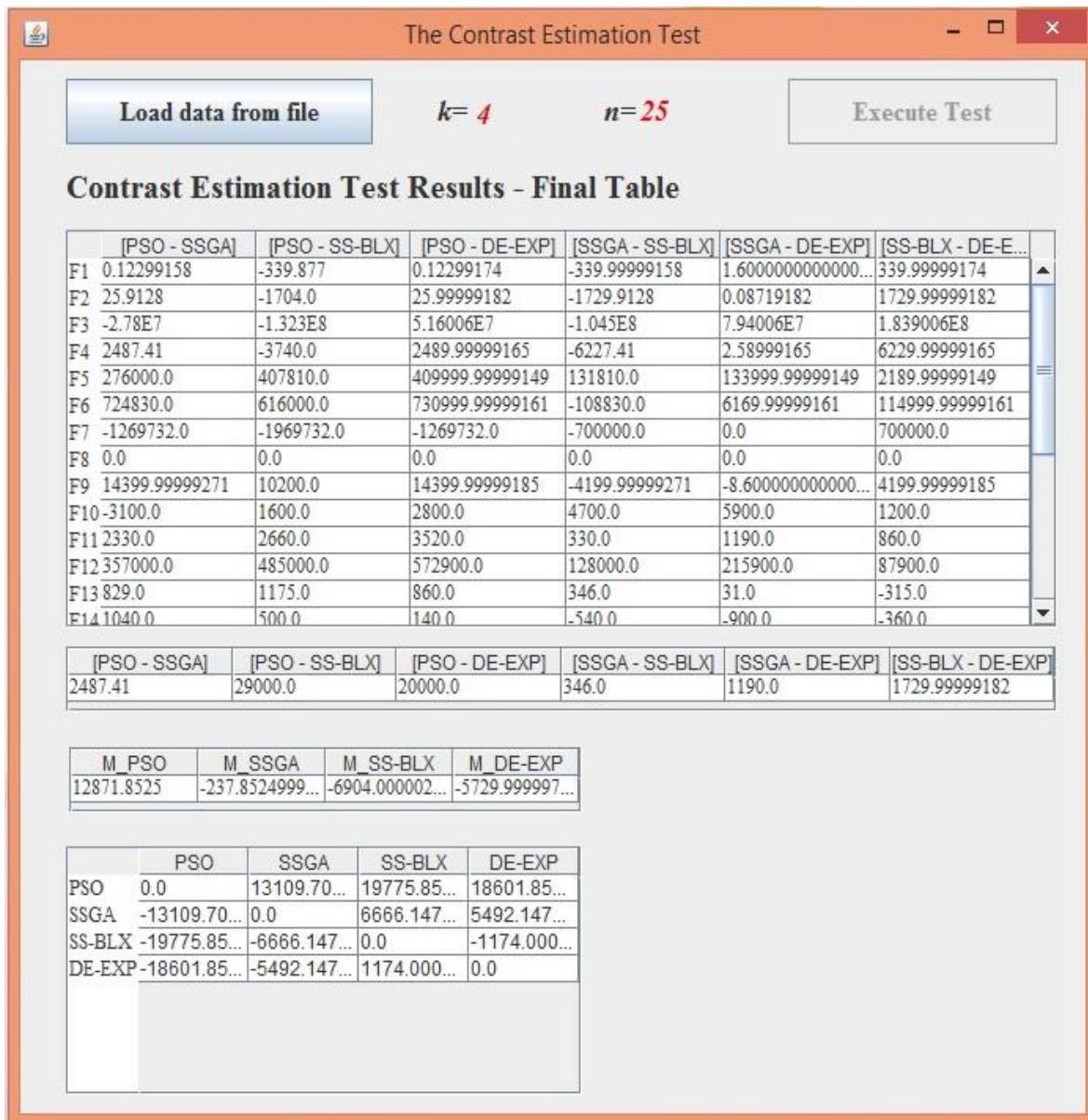


Figura 6.11 Resultados en VisTHAA de estimación de contraste con datos de Derrac 2012

Análisis de resultados

Comparando los resultados tanto en el artículo como en R, se puede concluir que VisTHAA cumple con el objetivo de correcta funcionalidad para ésta prueba. Es necesario mencionar, que en artículo [Derrac 2012] se encontró una operación en la página 7, la cual tiene un resultado erróneo, en el proceso de calcular el estimador de contraste. Se pudo verificar con los resultados obtenidos del código R, la herramienta VisTHAA y el desarrollo del cálculo manualmente; en los otros cálculos, los resultados de VisTHAA y la ejecución en R coinciden con los del artículo.

Con la interpretación de ésta prueba, se puede ver que el algoritmo SS-BLX presenta mejor desempeño, ya que toda su fila de estimadores tiene valores negativos, esto quiere decir que logra un nivel bajo de error. Al contrario de SS-BLX, PSO presenta el peor desempeño, ya que toda su fila de estimadores tiene valores positivos y altos.

6.2.4 Prueba de Wilcoxon de rangos con signos

Objetivo del estudio

Para comprobar el correcto funcionamiento de VisTHAA, para esta prueba se reprodujo la prueba de Wilcoxon de rangos con signos realizada en [García 2009], se tomaron los datos de éste artículo para reproducir la prueba ahí descrita y de ésta manera comparar los resultados obtenidos en VisTHAA. La instancia de ésta prueba se encuentra en el Anexo G.2.

Resultados

En la Figura 6.12 se muestra una tabla de resumen de la comparación del algoritmo G-CMA-ES contra todos los algoritmos estudiados en el artículo, en ésta tabla se resalta el resultado del par de algoritmos a los que se les aplicó la prueba de Wilcoxon en VisTHAA. En la Figura 6.13 se muestra el resultado en VisTHAA de la comparación G-CMA-ES vs CoEVO.

Table 12 Wilcoxon test considering functions f1–f25

G-CMA-ES vs.	R ⁺	R ⁻	p-value
BLX-GL50	289.5	35.5	0.001
BLX-MA	295.5	29.5	0.001
CoEVO	301.0	24.0	0.000
DE	262.5	62.5	0.009
DMS-L-PSO	199.0	126.0	0.357
EDA	284.5	40.5	0.001
K-PCX	269.0	56.0	0.004
L-CMA-ES	273.0	52.0	0.003
L-SaDE	209.0	116.0	0.259
SPC-PNX	305.5	19.5	0.000

Figura 6.12 Tabla de resultados de prueba de Wilcoxon, G-CMA-ES vs otros algoritmos [García 2009].

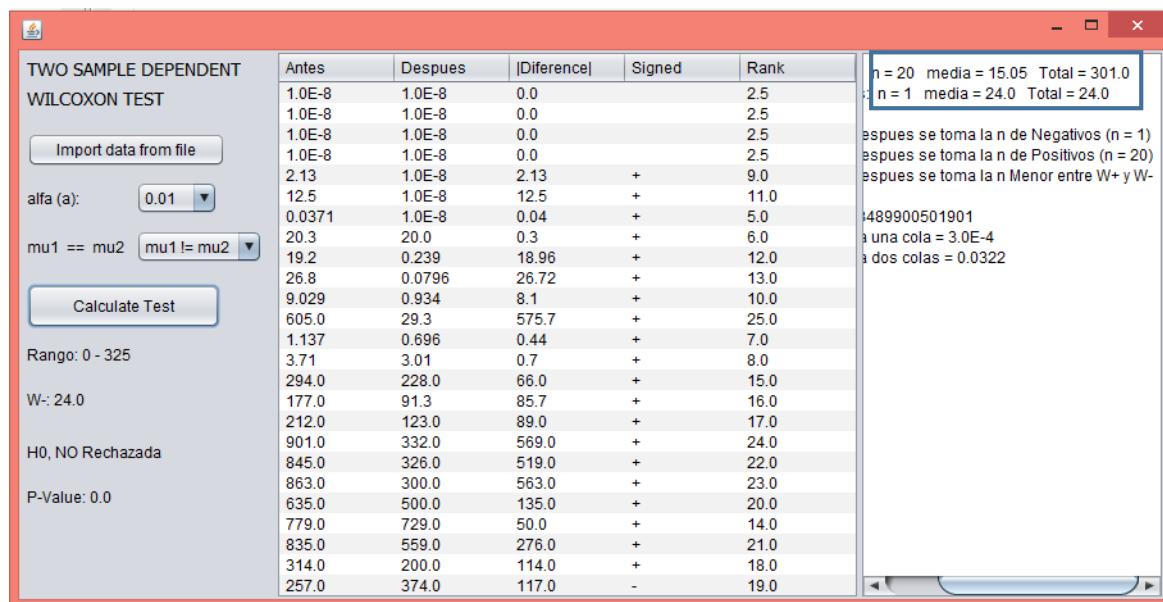


Figura 6.13 Prueba de Wilcoxon en VisTHAA, G-CMA-ES vs CoEVO.

Análisis de resultados

Ya que los resultados obtenidos en VisTHAA coinciden con el resultado obtenido en la comparación G-CMA-ES vs CoEVO de [García 2009], se puede concluir que VisTHAA realiza la prueba correctamente. El resultado de esta prueba concluye que dado el R+ y el p-value obtenido en la comparación se concluye que G-CMA-ES tiene mejor desempeño que CoEVO.

6.3 Estudio comparativo de un algoritmo genético para el problema de empacado de objetos.

6.3.1 Prueba Wilcoxon de rangos con signos a GGA-CGT

Objetivo del estudio

Esta prueba se aplicó a datos de desempeño de dos configuraciones del algoritmo genético *Grouping Genetic Algorithm with Controlled Gene Transmission* (GGA-CGT) desarrollado por Quiroz (Quiroz, 2014) para resolver el problema de empacado de objetos. Este algoritmo fue descrito brevemente en la sección 2.3.

Instancia

La instancia de este experimento está concentrada en la Figura 6.14, en ella se representa el porcentaje de error obtenido en 80 iteraciones de GGA-CGT cuando el tamaño del grupo elite b es $b=0.1$ y cuando es $b=0$.

Iter	Error_B=0	Error_B=.1	Iter	Error_B=0	Error_B=.1	Iter	Error_B=0	Error_B=.1	Iter	Error_B=0	Error_B=.1
1	0	0	21	0	0	41	0.01204819	0	61	0.01197605	0
2	0	0	22	0	0	42	0.01204819	0	62	0.01197605	0
3	0	0	23	0	0	43	0.01204819	0	63	0.00598802	0
4	0	0	24	0	0	44	0	0	64	0.01197605	0
5	0	0	25	0	0	45	0.01204819	0	65	0.00598802	0
6	0	0	26	0	0	46	0.01204819	0	66	0.01197605	0
7	0	0	27	0	0	47	0.01204819	0	67	0.00598802	0
8	0	0	28	0	0	48	0.01204819	0	68	0.01197605	0
9	0	0	29	0	0	49	0.01204819	0	69	0.00598802	0
10	0	0	30	0	0	50	0	0	70	0.01197605	0
11	0	0	31	0	0	51	0.01204819	0	71	0.01197605	0
12	0	0	32	0.025	0	52	0.01204819	0	72	0.01197605	0
13	0	0	33	0	0	53	0.01204819	0	73	0.01197605	0
14	0	0	34	0	0	54	0.01204819	0	74	0.00598802	0
15	0	0	35	0	0	55	0.01204819	0	75	0.00598802	0
16	0	0	36	0	0	56	0.01204819	0	76	0.01197605	0
17	0	0	37	0	0	57	0.01204819	0	77	0.01197605	0
18	0	0	38	0	0	58	0.01204819	0	78	0.00598802	0
19	0	0	39	0	0	59	0.01204819	0	79	0.00598802	0
20	0	0	40	0	0	60	0.01204819	0	80	0.00598802	0

Figura 6.14 Datos de desempeño del algoritmo GGA-CGT

Con esta prueba se quiere comprobar si el valor de b afecta al algoritmo y hace que tenga un mejor desempeño y que esta diferencia sea significativa. La H_0 dice que el desempeño de ambos son significativamente iguales.

Resultados

En la Figura 6.15 se muestran los resultados de la prueba en VisTHAA.

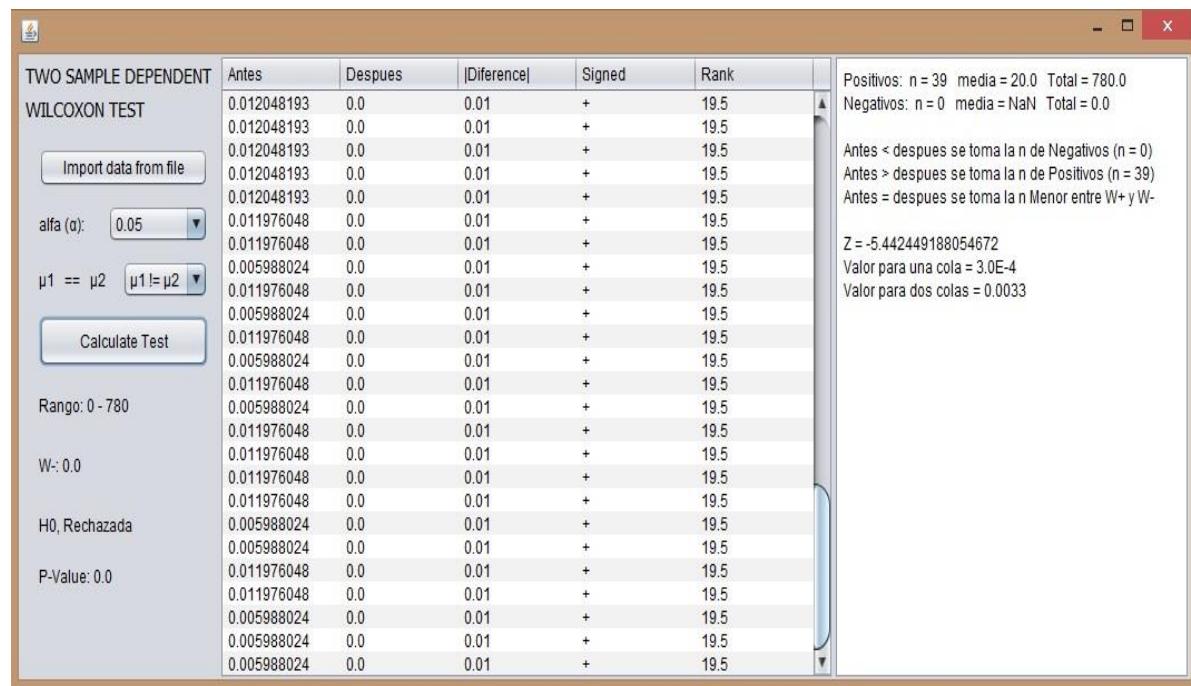


Figura 6.15 Prueba de Wilcoxon en VisTHAA a GGA-CGT

Análisis de resultados

Se muestra que la H_0 es rechazada, por lo tanto, se demuestra que el desempeño mejora significativamente cuando el valor de $b=0.1$. De esta manera se puede mejorar el algoritmo, y se puede afirmar con rigor científico que dicha configuración es la mejor.

6.4 Estudio comparativo de GGA-CGT vs. algoritmos del estado del arte

En este caso de estudio, se hizo un análisis del desempeño del algoritmo GGA-CGT comparado con otros algoritmos del estado del arte que resuelven el problema de empacado de objetos en contenedores; estos fueron HGGA-BP2 [Quiroz 2014], WABP Loh 2008], FF [Johnson 1974], BF [Johnson 1974], WF [Johnson 1974]. El análisis se hizo en base al error obtenido por cada algoritmo para resolver el conjunto de instancias TEST. En la Figura 6.16 se muestra la instancia usada para las pruebas y posteriormente se describen las pruebas realizadas para este estudio.

Instancias	Error_GGA-CGT	ErrorHGGA2	ErrorWABP	ErrorFF	ErrorBF	ErrorWF
TEST0022.txt	0	0	0	0	0	0
TEST0065.txt	0	0	0	0	0	0
TEST0097.txt	0	0	0.083333333	0.083333333	0.083333333	0.083333333
TEST0058.txt	0	0	0.05	0.05	0.05	0.05
TEST0055a.txt	0	0	0.066666667	0.066666667	0.066666667	0.066666667
TEST0049.txt	0	0	0.090909091	0.090909091	0.090909091	0.090909091
TEST0075.txt	0	0	0.076923077	0.076923077	0.076923077	0.076923077
TEST0054.txt	0	0	0.071428571	0.071428571	0.071428571	0.071428571
TEST0068.txt	0	0	0.083333333	0.083333333	0.083333333	0.083333333
TEST0014.txt	0.043478261	0.043478261	0.043478261	0.043478261	0.043478261	0.043478261
TEST0082.txt	0	0	0.041666667	0.041666667	0.041666667	0.041666667
TEST0044.txt	0	0	0.071428571	0.071428571	0.071428571	0.071428571
TEST0030.txt	0	0	0.037037037	0.037037037	0.037037037	0.03703704
TEST0005.txt	0	0	0.035714286	0.035714286	0.035714286	0.03571429
TEST0095.txt	0	0	0	0.0625	0.0625	0.0625
TEST0055b.txt	0	0	0	0.05	0.05	0.05
TEST0084.txt	0	0	0	0.0625	0.0625	0.0625

Figura 6.16 Instancia con errores de algoritmos que resuelven BPP

2.4.1 Prueba de Friedman

Objetivo del estudio

Esta prueba sirve para detectar diferencias significativas en el comportamientos de dos o más algoritmos, por lo que esta prueba se aplicó para determinar si los algoritmos comparados del estado del arte se comportan de manera similar

Resultados

A continuación se describe la prueba de Friedman así mismo los resultados obtenidos, para comparar el desempeño del algoritmo genético *Grouping Genetic Algorithm with Controlled Gene Transmission* (GGA-CGT) desarrollado por Quiroz [Quiroz, 2014] contra otros

algoritmos del estado del arte que resuelven BPP. La Figura 6.17 muestra los resultados de la prueba realizada.

Análisis de resultados

La hipótesis nula que indica que todos los algoritmos se comportan similarmente, por lo que sus rankings R_j deben ser similares.

En la Figura 6.17 se muestra el resultado de esta prueba, podemos ver que GGA-CGT y HGGA2 tienen el mejor ranking (35.5) seguido por WABP que muestra un desempeño similar (48.5) y por último vemos a FF, BF, WF que tienen el peor desempeño (78.5)

La prueba de Friedman procede comprobando si los rankings medios obtenidos son significativamente diferentes del ranking medio esperado bajo la hipótesis nula, $R_j = 59.5$

Con $FF = 38.01 >$ valor tabla Chi = 5.142 se rechaza la hipótesis nula, por lo que se concluye que los algoritmos no se comportan de manera similar.

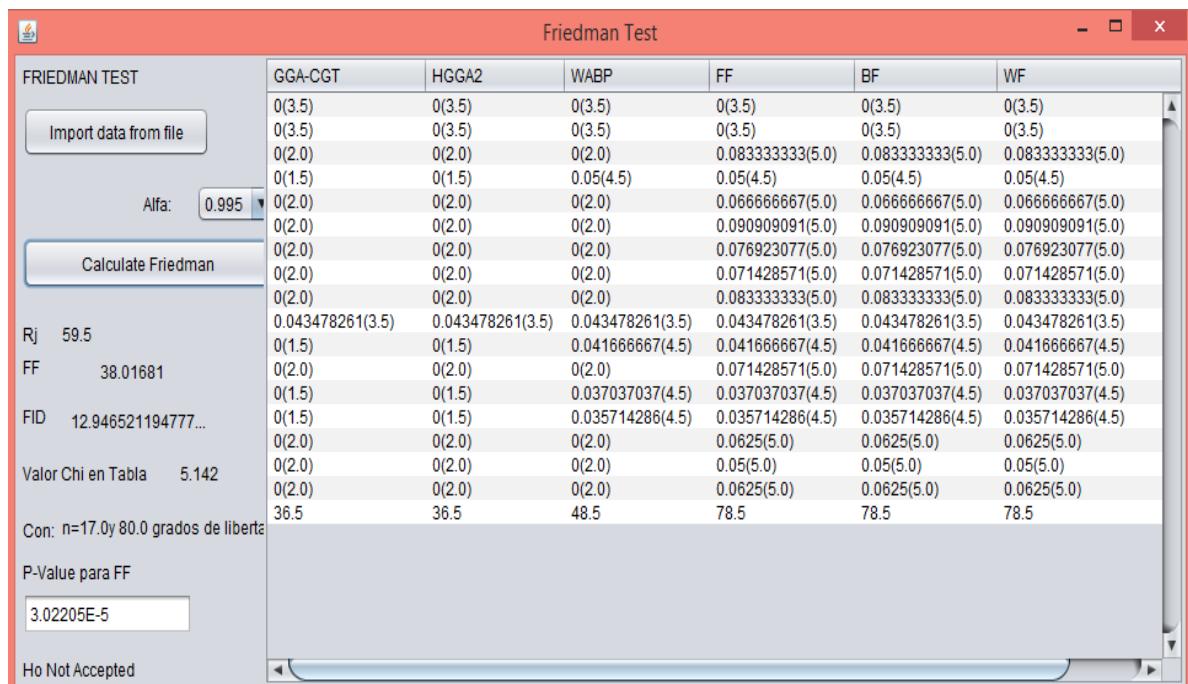


Figura 6.17 Resultados de la prueba de Friedman en VisTHAA

6.4.2 Prueba de signos múltiples

Objetivo del estudio

Hacer un análisis del comportamiento de los algoritmos para determinar si hay alguno que muestre mejor desempeño que GGA-CGT. Esta prueba se aplica para hacer un estudio preliminar de los resultados ya que permite destacar a los algoritmos a los cuales su desempeño es estadísticamente diferente cuando es comparado con el algoritmo de control.

Resultados

A continuación se describe la prueba de signos múltiples así mismo los resultados obtenidos, para comparar el desempeño del algoritmo genético *Grouping Genetic Algorithm with Controlled Gene Transmission* (GGA-CGT) desarrollado por Quiroz [Quiroz, 2014] contra otros algoritmos del estado del arte que resuelven BPP. Para hacer la comparación, se establece que el método de control es GGA-CGT como se muestra en la Figura 6.18.

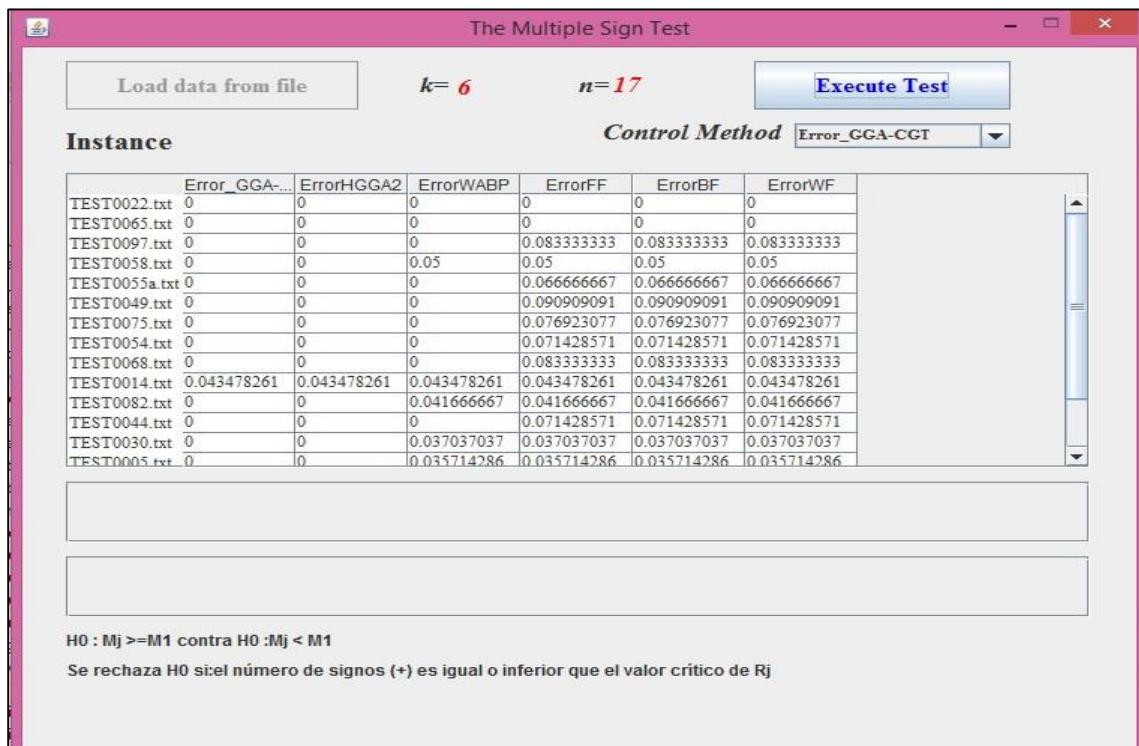


Figura 6.18 Prueba de signos múltiples en VisTHAA, selección de método de control

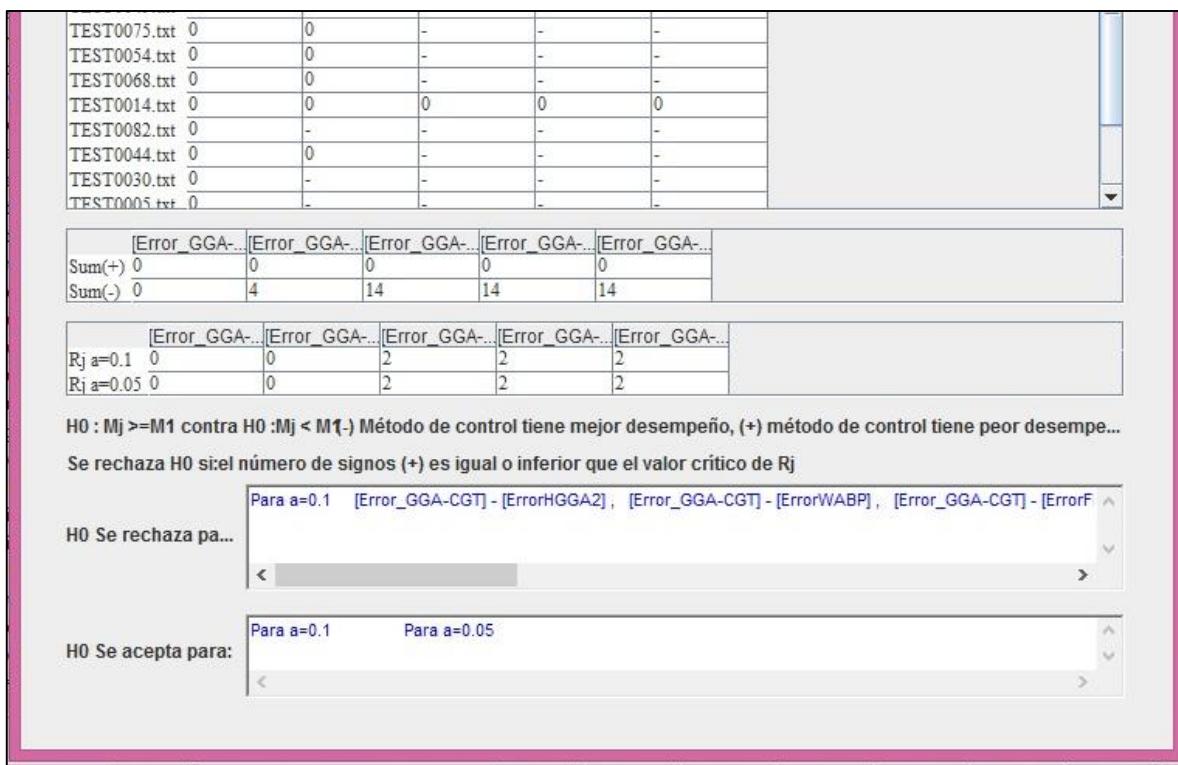


Figura 6.19 Resultados de la prueba de signos múltiples en VisTHAA

Análisis de resultados

Como se puede ver en la Figura 6.19, se muestra el resultado de esta prueba, tomando a GGA-CGT como método de control, en esta figura se puede observar que ningún algoritmo que se comparó con GGA-CGT se destaca estadísticamente y por esto se concluye que GGA-CGT tiene mejor desempeño para las instancias TEST.

6.4.3 Prueba de estimación de contraste

Objetivo del estudio

Esta prueba puede ser usada para estimar la diferencia entre el desempeño de dos algoritmos, la prueba asume que las diferencias esperadas entre el desempeño de los algoritmos es la misma para todos los problemas. Por lo que el desempeño del algoritmos es reflejado por magnitudes de la diferencia entre ellas en cada dominio. Esta prueba obtiene una diferencia

cuantitativa. El estimador de contraste puede ser visto como una medida de desempeño global avanzada, a pesar de que ésta prueba no provee una probabilidad de error asociada con el rechazo de la hipótesis nula de igualdad, el estimador es útil para estimar por qué tanto es mejor el desempeño de un algoritmo que otro.

Resultados

A continuación se describe la prueba de estimación de contraste así mismo los resultados obtenidos, para comparar el desempeño del algoritmo genético *Grouping Genetic Algorithm with Controlled Gene Transmission* (GGA-CGT) desarrollado por Quiroz [Quiroz, 2014] contra otros algoritmos del estado del arte que resuelven BPP.

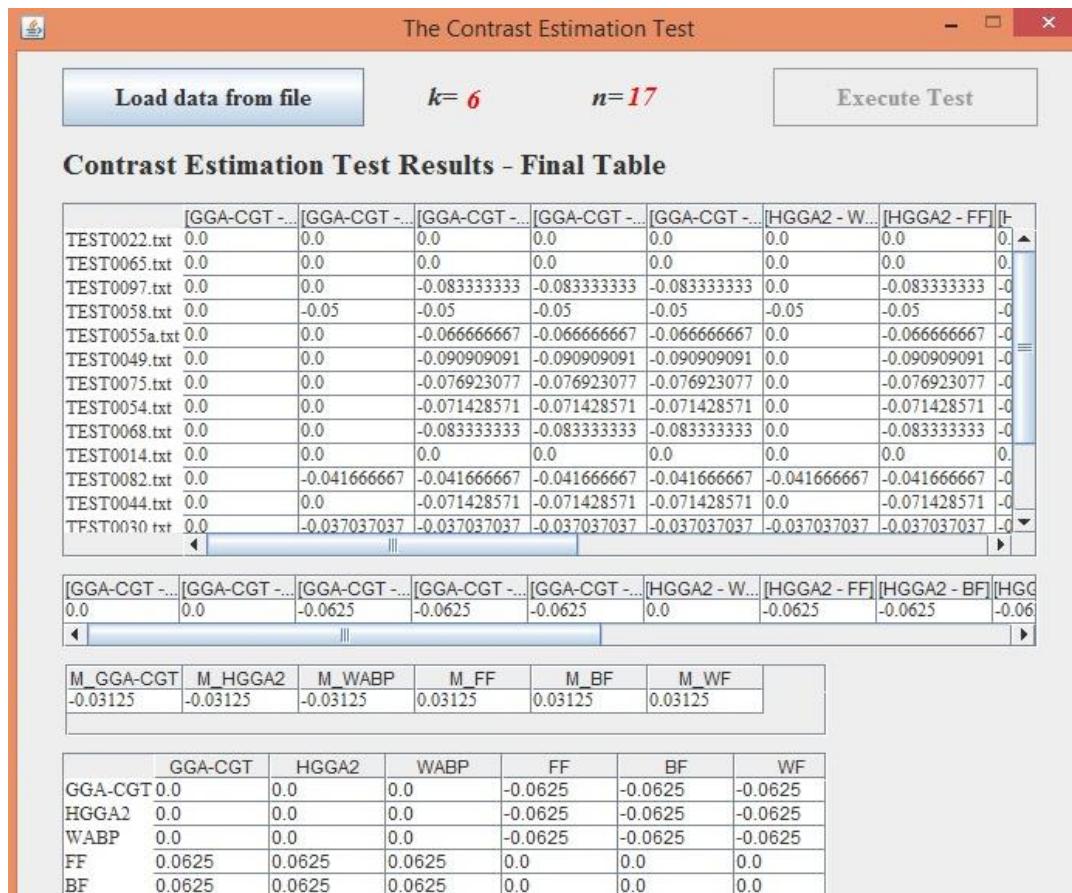


Figura 6.20 Resultados de la prueba de estimación de contraste en VisTHAA

Análisis de resultados

En la Figura 6.20 se puede ver en el estimador de contraste que los algoritmos GGA-CGT, HGGA2 y WABP se comportan igual y no hay diferencia en sus desempeños para la solución de las instancias TEST, éstos tres algoritmos tiene el mejor desempeño para la solución de estas instancias.

6.5 Comentarios finales

Después de aplicar las pruebas no paramétricas a GGA-CGT se concluye lo siguiente.

Con la prueba de Wilcoxon de rangos con signos se comprobó que una nueva configuración aporta un mejor desempeño al algoritmo.

Después se hizo un análisis comparativo de desempeño entre algoritmos del estado del arte para resolver las instancias TEST. Con la prueba de Friedman se pudo comprobar que los algoritmos comparados no tienen un comportamiento similar, dado esto se aplicó la prueba de signos múltiples para ver si los algoritmos comparados tenían un mejor comportamiento que GGA-CGT y se comprobó que no eran mejor. Posteriormente se aplicó la prueba de estimación de contraste, la cual sirvió para ver la diferencia cuantitativa entre el desempeño de los algoritmos. Con esta prueba se demostró que GGA-CGT, HGGA-BP2 y WABP tuvieron el mejor desempeño. La mayoría de las pruebas apuntan a que GGA-CGT tiene el mejor desempeño, sin embargo la prueba de estimación de contraste encuentra igual desempeño con HGGA-BP2 y WABP. Hay que recordar que estas pruebas solo fueron aplicadas para el conjunto de instancias TEST.

Capítulo 7

Conclusiones y trabajo futuro

En este proyecto se propone una arquitectura para el análisis experimental de algoritmos heurísticos que servirá de guía para el desarrollo de un sistema de software. A partir de la arquitectura, se construye una herramienta para el estudio comparativo de algoritmos heurísticos, la cual está basada en pruebas de hipótesis no-paramétricas.

En la literatura especializada se ha reportado que muchos estudios experimentales no cumplen las propiedades requeridas para aplicar las muy conocidas pruebas paramétricas (por ejemplo: t-student) [Derrac 2011]. Por lo que la herramienta (VisTHAA) ofrece un conjunto de pruebas de hipótesis complementarias para dar mayor significancia a los resultados. Hasta nuestro conocimiento, son pocos los sistemas de análisis experimental de algoritmos que integran este tipo de herramientas. La herramienta propuesta se integra con otra de análisis visual para proporcionar conocimiento que facilite la tarea de diseño.

Se validó la funcionalidad de la herramienta estadística reproduciendo exitosamente estudios importantes de la literatura. Como caso de estudio se analizan dos configuraciones del algoritmo GGA-CGT que da solución a problemas BPP. El experimento revela que una de las configuraciones es mejor que la otra de manera significativa. También se hizo un análisis comparativo de GGA-CGT contra algoritmos del estado del arte, las pruebas demuestran que GGA-CGT es mejor que éstos algoritmos.

Entre los trabajos futuros para enriquecer la herramienta es necesario extender el metalenguaje para el ingreso de datos a VisTHAA con parámetros de más de una dimensión, además incorporar a VisTHAA la capacidad para realizar el tratamiento de los datos con técnicas de análisis multivariado. Así mismo, implementar más pruebas de hipótesis, paramétricas y no paramétricas. Algo que se puede considerar necesario es conectar a VisTHAA con herramientas de libre distribución, para apoyar en cálculos necesarios en el análisis estadístico que en sí es laborioso realizar por medio de tablas. Otro objetivo que se quiere alcanzar en futuros trabajos es el proveer a VisTHAA la capacidad de que funcione de manera *on-line*.

Referencias

- [Adenso 2006]** Adenso-Diaz, B., and Laguna, M. “Fine-tuning of algorithms using fractional experimental design and local search”. Operations Research 54(1):99–114. 2006
- [Álvarez 2006]** Álvarez V. “Modelo para Representar la Complejidad del Problema y el Desempeño de Algoritmos”, Tesis de Maestría / Instituto Tecnológico de Ciudad Madero. - Ciudad Madero, Tam, 2006.
- [Alvim 2004]** Alvim A., Glover F., Ribeiro C., Aloise D. “A hybrid improvement heuristic for the one-dimensional bin packing problem”. Journal of Heuristics, 10: 205-229, 2004.
- [Barr 1995]** Barr R. Golden B., Kelly J., Resendez M., Stewart W. “Designing and Reporting on Computational Experiments with Heuristic Methods”. Journal of Heuristics. - 1995. - págs. 19-32.
- [Berenson 2001]** Berenson M., Levine D., Krehbiel T. “Estadística para administración”. Prentice Hall, 2da edición, México 2001
- [Blum 2003]** Blum C. y Roli A. “Metaheuristics in combinatorial optimization: Overview and conceptual comparison”. ACM Computing Surveys. págs. 268–308., 2003
- [Brownlee 2007]** Brownlee J. “OAT: The Optimization Algorithm Toolkit” Complex Intelligent Systems Laboratory (CIS), Centre for Information Technology Research (CITR), Faculty of Information and Communication Technologies (ICT), Swinburne University of Technology, Victoria, Australia, Technical Report. 2007
- [Callan 2007]** Callan R., Hsu R. “Design and Analysis of Meta-heuristics for Constrained Optimization Problems”. CSE 400/401 Senior Design Project, 2007.

- [Castillo 2011]** Castillo, N. “Evaluación de Estrategias de Mejora del Desempeño de Metaheurísticos Aplicados a BPP Vía Diagnóstico Visual”. Tesis de maestría Ciudad Madero, Tamaulipas. 2011.
- [Cohen 1995]** Cohen P. R. “Empirical Methods for Artificial Intelligence”. The MIT Press, Cambridge, Massachusetts, 1995.
- [Crowling 2000]** Cowling P., Kendall G. y Soubeiga E. “A Hyperheuristic Approach to Scheduling a Sales Summit.”, 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 1999]** Cruz L. “Automatización del Diseño de la Fragmentación Vertical y Ubicación en Bases de Datos Distribuidas usando Métodos Heurísticos y Exactos” Tesis de Maestría / Universidad Virtual del Campus Tampico del Instituto Tecnológico y de Estudios Superiores de Monterrey. - Altamira, Tam, 1999.
- [Cruz 2004]** Cruz L. “Clasificación de Algoritmos Heurísticos Para la Solución de Problemas de Bin Packing” Tesis Doctoral / Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET). - Cuernavaca, México. 2004.
- [Derrac 2011]** Derrac J. García S., Molina D., Herrera F. “A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms”. Swarm and Evolutionary Computation Volume 1, Issue 1, March 2011.
- [Derrac 2012]** Derrac J. García S., Molina D., Herrera F. “Un tutorial sobre el uso de test estadísticos no paramétricos en comparaciones múltiples de metaheurísticas y algoritmos evolutivos”. VIII Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB), 2012.
- [Díaz 1996]** Díaz A. y Glover F. “Optimización Heurística y Redes Neuronales” España : Paraninfo, 1996.
- [Duarte 2007]** Duarte A., Pantrigo J. y Gallego M. “Metaheurísticas” Universidad Rey Juan Carlos. - Madrid, España, Dykinson, 2007.

- [Feitelson 2006]** Feitelson D. G. "Experimental Computer Science: The Need for a Cultural Change". <http://www.cs.huji.ac.il/~feit/papers/exp05.pdf>, 2006.
- [Fourer 2008]** Fourer R., Ma J. "Optimization Services: A Framework for Distributed Optimization". Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois 60208, USA., 2008
- [García 2009]** García S. Molina D., Lozano M., Herrera F. "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization". Journal of Heuristics, December 2009
- [Garey 1979]** Garey M. R., Jonson D. S. "Computers and Intractability: A Guide to the Theory of NP-Completeness". W. H. Freeman and Company, 1979. A classic introduction to the field.
- [Glover 1986]** 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 2002]** The Design of Innovation (Genetic Algorithms and Evolutionary Computation) [Sección de libro]: Springer 1 ed, 2002. - 1 ed.
- [Halim 2005]** Halim S., Lau H. y Wan W. "Tuning Tabu Search Strategies via Visual Diagnosis"Metaheuristics International Conference (MIC 2005). - Vienna, Austria, 2005. - págs. 630-636.
- [Halim 2006a]** Halim Steven, R.H.C. Yap, H.C. Lau. "Visualization for Analyzing Trajectory-Based Metaheuristic Search Algorithms", SICS Technical Report T2006:11, Mayo 2006
- [Halim 2006b]** Halim Steven, R.H.C. Yap, H.C. Lau "Viz: A Visual Analysis Suite for Explaining Local Search Behavior" User Interface System and Technology Montreux, Switzerland October, 2006.
- [Halim 2007]** Halim S., Yap R. y Lau H. "An Integrated White+Black Box Approach for Designing and Tuning Stochastic Local Search", Principles and Practice of Constraint Programming. - Rhode Island, United States of America , 2007. - págs. 332-347.

- [Heal 2014]** Heuristic and Evolutionary Algorithms Laboratory HeuristicLab [En línea]. - <http://dev.heuristiclab.com/trac/hl/core/wiki>
- [Johnson 1974]** Johnson DS. “Fast algorithms for bin packing”. Journal of Computer and System Sciences 1974; 8(3):272-314.
- [KEEL 2014]** Fernández J., García S. y Derrac J. KEEL (Knowledge Extraction based on Evolutionary Learning). <http://www.keel.es/>
- [Kendall 2005]** Kendall G. y Mohd N. “An Investigation of a Tabu-Search-Based Hyper-Heuristic for Examination Timetabling”. Automated Scheduling, Optimisation and Planning (ASAP) Research Group ; School of Computer Science and Information Technology, Universiry of Nottingham. - Nottingham, 2005.
- [Lamarca 2009]** Lamarca M. “Hipertexto, el nuevo concepto de documento en la cultura de la imagen”. Tesis doctoral. Universidad Complutense de Madrid, 2009
http://www.hipertexto.info/documentos/lenguajes_h.htm
- [Liefooghe 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”, Institut National de Recherche en Informatique et en Automatique. 2009.
- [Liu 1996]** Liu X. “Intelligent Data Analysis: Issues and Challenges”. The Knowledge Engineering Review, 11: 365-371, 1996.
- [Loh 2008]** Loh K., Golden B., Wasil E. “Solving the one-dimensional bin packing problem with a weight annealing heuristic”. Computers & Operations Research 35: 2283-2291, 2008.
- [Lopez 2000]** López C. “Modelo para el Desarrollo de Bibliotecas Digitales Especializadas”2000.
<http://www.bibliodgsca.unam.mx/tesis/tes7cllg/tes7cllg.htm>
- [Martello 1990]** Martello S., Toth P. “Knapsack Problems: Algorithms and Computer Implementations”. Wiley & Sons Ltd. 1990.
- [McGeoch 1992]** McGeoch C. “Analysis of algorithms by simulation: variance reduction techniques and simulation speedups” ACM Computing Surveys. - 1992. - págs. 195- 212.

- [Mendenhall 1997]** Mendenhall W., Sincich T. "Probabilidad y estadística para ingeniería y ciencias". Prentice-Hall. Cuarta edición, México 1997.
- [Morrison 2010]** Morrison C., Snodgrass R. "Computer science can use more science", University of Arizona, October 27, 2010.
<http://www.cs.arizona.edu/~rts/pubs/CACM2011.pdf>
- [Nieto 2007]** Nieto D. "Hibridación de Algoritmos Metaheurísticos para Problemas de Bin Packing". Tesis de Maestría / Instituto Tecnológico de Ciudad Madero. - Ciudad Madero, Tam., 2007.
- [OAT 2013]** Brownlee J. Optimization Algorithm Toolkit, OAT
<http://optalgtoolkit.sourceforge.net/>
- [O'Brien 2008]** O'Brien R. "Ant Algorithm Hyperheuristic Approaches for Scheduling Problems", The University of Nottingham. - 2008.
- [Özcan 2008]** Özcan E., Bilgin B. y Korkmaz E. "A Comprehensive Analysis of Hyper-heuristics", Department of Computer Engineering, Yeditepe University.. - 2008.
- [ParadisEO 2013]** Europe Inria Lille - Nord ParadisEO A Software Framework for Metaheuristics
<http://paradiseo.gforge.inria.fr/index.php?n>Main.HomePage>
- [Pérez 2007]** Pérez V. "Modelado Causal del Desempeño de Algoritmos Metaheurísticos en Problemas de Distribución de Objetos". Tesis de maestría, Instituto Tecnológico de Cd. Madero, Tamaulipas, México, 2007.
- [Quiroz 2009]** Quiroz M. "Caracterización de Factores de Desempeño de Algoritmos de Solución de BPP" Tesis de Maestría. - Instituto Tecnológico de Ciudad Madero, 2009.
- [Quiroz 2013]** Quiroz, M. "A grouping genetic algorithm with controlled gene transmission for the bin packing problem". Elsevier Editorial System(tm) for Computers & Operations Research 2013.
- [Quiroz 2014]** Quiroz, M. "Caracterización del proceso de optimización de algoritmos heurísticos aplicados al problema de empacado de objetos en contenedores". Tesis Doctoral, Institutito Tecnológico de Ciudad Madero, México, 2014

- [Siegel 1995]** Siegel, S., & Castellan, N. “Estadística no paramétrica aplicada a las ciencias de la conducta” (Cuarta ed.). Trillas. 1995
- [Snodgrass 2010]** Snodgrass R. Ergalics. “A Natural Science of Computation”. February 2010. <http://www.cs.arizona.edu/projects/ergalics/whatis.html>
- [Suganthan 2005]** P., N. Hansen, J. Liang, K. Deb, Y. Chen, A. Auger, S. Tiwari. “Problem definitions and evaluation criteria for the CEC’2005 special sesión on real parameter optimization”, Nanyang Technological University, Tech. Rep., 2005.
- [Taillard 2003]** Taillard Éric D. “A Statistical Test for Comparing Success Rates”. MIC2003: The Fifth Metaheuristics International Conference.
- [VIZ 2010]** Halim S. World of Seven. “Viz: SLS Engineering Suite”. 2010 <http://www.comp.nus.edu.sg/~stevenha/viz/index.html>.
- [V-MDF 2009]** Halim S. World of Seven, V-MDF. <http://www.comp.nus.edu.sg/~stevenha/v-mdf/>.
- [Wagner 2004]** Wagner, S. & Affenzeller, M. The HeuristicLab Optimization Environment. Linz, Austria: Johannes Kepler University. 2004

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 un contenedor, lo cual se realiza en $\theta(n)$ pasos.
2. Despues 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 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, \dots, 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 $I \subseteq \{1, 2, \dots, m\}$, 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, \dots, 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, \dots, 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 de la suma de los tamaños de los objetos. Sea k igual a 2, hay una solución para BPP que use dos contenedores, si y sólo 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 $\theta(n)$ pasos.

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

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 1979].

Anexo B

Tablas de valores críticos

Tabla B.2 - Valores críticos en la prueba de los signos de dos colas en $\alpha = 0,05$ y $\alpha = 0,1$. Un algoritmo es significativamente mejor que otro si tiene un mejor desempeño en al menos los casos que se presentan en cada fila.

#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.3 - Valores críticos de r_j mínimo para la comparación de $m=k-1$ algoritmos contra un control en los conjuntos de datos n .

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

Tabla B.4 - Valores críticos de rj mínimo para la comparación de $m=k-1$ algoritmos contra un control en los conjuntos de datos n . Continuación...

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.5 - Tabla de la Distribución F

F Values for $\alpha = 0.10$

d_2		d_1								
	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	

Tabla B.6 - Tabla de la Distribución F. Continuación

		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	

Tabla B.7 - Tabla de la Distribución F. Continuación

		F Values for $\alpha = 0.05$									
		d_1									
d_2	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	

Tabla B.8-Tabla de la Distribución F. Continuación

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

Tabla B.9 - Tabla de la Distribución F. Continuación

F Values for $\alpha = 0.01$

d_2	10	12	15	20	24	30	40	60	120	inf
	d_1									
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.10 - Tabla P de la significancia del índice T de Wilcoxon

$n \backslash \alpha$	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

Anexo C

Ejemplos de procedimientos de pruebas no paramétricas

Anexo C.1 Prueba de Signos Múltiples

Supuestos

- ▶ Se tiene un método de control, llamado M_1 .
- ▶ Se tienen $(k-1)$ métodos de prueba, llamados M_j , donde $j=2,3\dots,k$.
- ▶ X_{ij} representa el i -ésimo elemento del conjunto M_j .

Se consideran dos posibles hipótesis nulas:

- ▶ $H_0: M_j \geq M_1$ Significa que el método a probar es igual o más efectivo que el método de control
- ▶ $H_0: M_j \leq M_1$ Significa que el método de control es igual o más efectivo que el método a probar

Parámetros a estimar

- ▶ M_j : Representa la respuesta mediana de una muestra de resultados del algoritmo j -ésimo
- ▶ M_1 : Representa la respuesta mediana de una muestra de resultados del algoritmo de control

El parámetro a estimar entonces es: $M_1 - M_j$

Estimador del parámetro

- ▶ Se evalúa el rendimiento de cada método contra el de control ($X_{1k} - X_{jk}$), y se coloca en cada método y por cada prueba:

- (+) Si el método de prueba es más eficiente
- (-) Si el método de control es más eficiente
- (=) Si los dos métodos son igual de eficientes
- Se contabiliza el número de signos (+) y (-) por cada método

Punto crítico

- ▶ El punto crítico es definido por R_j , el cual representa el punto crítico de aceptación/rechazo de la hipótesis para la prueba del método de control con el método j
- ▶ Se busca en la Tabla B.2 el valor crítico de R_j para cada método, donde:
 - n es igual a la suma de signos (+) y (-) en el método
 - m es el número de métodos a probar (sin contar el de control)

Procedimiento de aceptación y rechazo

- ▶ Para $H_0: M_j \geq M_1$
Se aceptará H_0 si la suma de signos (+) del método j es superior al punto crítico
- ▶ Para $H_0: M_j \leq M_1$
Se aceptará H_0 si la suma de signos (-) del método j es superior al punto crítico

La Tabla C.1.1 concentra el proceso de la prueba de signos múltiples, donde DE-EXP es el método de control.

Tomando $H_0: M_j \geq M_1$

Para PSO y SSGA H_0 es rechazada

Para SS-BLX: H_0 es aceptada

Tabla C.1.1 Resultados de la prueba de signos múltiples

Algoritmo Orden	DE-EXP 1 (control)	PSO 2	SSGA 3	SS-BLX 4
F1	8,26E-06	1,23E-01 (-)	8,42E-06 (-)	3,40E+02 (-)
F2	8,18E-06	2,60E+01 (-)	8,72E-02 (-)	1,73E+03 (-)
F3	9,94E+04	5,17E+07 (-)	7,95E+07 (-)	1,84E+08 (-)
F4	8,35E-06	2,49E+03 (-)	2,59E+00 (-)	6,23E+03 (-)
F5	8,51E-06	4,10E+05 (-)	1,34E+05 (-)	2,19E+03 (-)
F6	8,39E-06	7,31E+05 (-)	6,17E+03 (-)	1,15E+05 (-)
F7	1,27E+06	2,68E+02 (+)	1,27E+06 (=)	1,97E+06 (-)
F8	2,04E+04	2,04E+04 (=)	2,04E+04 (=)	2,04E+04 (=)
F9	8,15E-06	1,44E+04 (-)	7,29E-06 (+)	4,20E+03 (-)
F10	1,12E+04	1,40E+04 (-)	1,71E+04 (-)	1,24E+04 (-)
F11	2,07E+03	5,59E+03 (-)	3,26E+03 (-)	2,93E+03 (-)
F12	6,31E+04	6,36E+05 (-)	2,79E+05 (-)	1,51E+05 (-)
F13	6,40E+02	1,50E+03 (-)	6,71E+02 (-)	3,25E+02 (+)
F14	3,16E+03	3,30E+03 (-)	2,26E+03 (+)	2,80E+03 (+)
F15	2,94E+05	3,40E+05 (-)	2,92E+05 (+)	1,14E+05 (+)
F16	1,13E+05	1,33E+05 (-)	1,05E+05 (+)	1,04E+05 (+)
F17	1,31E+05	1,50E+05 (-)	1,19E+05 (+)	1,18E+05 (+)
F18	4,48E+05	8,51E+05 (-)	8,06E+05 (-)	7,67E+05 (-)
F19	4,34E+05	8,50E+05 (-)	8,90E+05 (-)	7,56E+05 (-)
F20	4,19E+05	8,51E+05 (-)	8,89E+05 (-)	7,46E+05 (-)
F21	5,42E+05	9,14E+05 (-)	8,52E+05 (-)	4,85E+05 (+)
F22	7,72E+05	8,07E+05 (-)	7,52E+05 (+)	6,83E+05 (+)
F23	5,82E+05	1,03E+06 (-)	1,00E+06 (-)	5,74E+05 (+)
F24	2,02E+05	4,12E+05 (-)	2,36E+05 (-)	2,51E+05 (-)
F25	1,74E+06	5,10E+05 (+)	1,75E+06 (-)	1,79E+06 (-)
# (+)	-	2	6	8
# (-)	-	22	17	16
$R_j \alpha = 0,1$	-	7	6	7
$R_j \alpha = 0,05$	-	6	6	6

Anexo C.2 Prueba de Estimación de Contraste

Tabla C.2.1 Resultados de desempeño de algoritmos de estudio

Algoritmo	PSO	SSGA	SS-BLX	DE-EXP
Orden	1	2	3	4
F1	1.23E-01	8.42E-06	3.40E+02	8.26E-06
F2	2.60E+01	8.72E-02	1.73E+03	8.18E-06
F3	5.17E+07	7.95E+07	1.84E+08	9.94E+04
F4	2.49E+03	2.59E+00	6.23E+03	8.35E-06
F5	4.10E+05	1.34E+05	2.19E+03	8.51E-06
F6	7.31E+05	6.17E+03	1.15E+05	8.39E-06
F7	2.68E+02	1.27E+06	1.97E+06	1.27E+06
F8	2.04E+04	2.04E+04	2.04E+04	2.04E+04
F9	1.44E+04	7.29E-06	4.20E+03	8.15E-06
F10	1.40E+04	1.71E+04	1.24E+04	1.12E+04
F11	5.59E+03	3.26E+03	2.93E+03	2.07E+03
F12	6.36E+05	2.79E+05	1.51E+05	6.31E+04
F13	1.50E+03	6.71E+02	3.25E+02	6.40E+02
F14	3.30E+03	2.26E+03	2.80E+03	3.16E+03
F15	3.40E+05	2.92E+05	1.14E+05	2.94E+05
F16	1.33E+05	1.05E+05	1.04E+05	1.13E+05
F17	1.50E+05	1.19E+05	1.18E+05	1.31E+05
F18	8.51E+05	8.06E+05	7.67E+05	4.48E+05
F19	8.50E+05	8.90E+05	7.56E+05	4.34E+05
F20	8.51E+05	8.89E+05	7.46E+05	4.19E+05
F21	9.14E+05	8.52E+05	4.85E+05	5.42E+05
F22	8.07E+05	7.52E+05	6.83E+05	7.72E+05
F23	1.03E+06	1.00E+06	5.74E+05	5.82E+05
F24	4.12E+05	2.36E+05	2.51E+05	2.02E+05
F25	5.10E+05	1.75E+06	1.79E+06	1.74E+06

Procedimiento de la prueba

- Para cada par de k algoritmos en el experimento, se calculan las diferencias entre desempeños de dos algoritmos (ver Tabla C.2.2) con cada uno de los n problemas.

$$D_{i(u, v)} = x_{iu} - x_{iv} \text{ donde:}$$

$$i = 1, \dots, n. u = 1, \dots, k. v$$

$= 1, \dots, k.$ Considerando únicamente donde los pares sean $u < v.$

Tabla C.2.2 Resultados de diferencias en pares para cada algoritmo

Di(1, 2)	Di(1, 3)	Di(1, 4)	Di(2, 3)	Di(2, 4)	Di(3, 4)
1.23E-01	-3.40E+02	1.23E-01	-3.40E+02	1.60E-07	3.40E+02
2.59E+01	-1.70E+03	2.60E+01	-1.73E+03	8.72E-02	1.73E+03
-2.78E+07	-1.32E+08	5.16E+07	-1.05E+08	7.94E+07	1.84E+08
2.49E+03	-3.74E+03	2.49E+03	-6.23E+03	2.59E+00	6.23E+03
2.76E+05	4.08E+05	4.10E+05	1.32E+05	1.34E+05	2.19E+03
7.25E+05	6.16E+05	7.31E+05	-1.09E+05	6.17E+03	1.15E+05
-1.27E+06	-1.97E+06	-1.27E+06	-7.00E+05	0.00E+00	7.00E+05
0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
1.44E+04	1.02E+04	1.44E+04	-4.20E+03	-8.60E-07	4.20E+03
-3.10E+03	1.60E+03	2.80E+03	4.70E+03	5.90E+03	1.20E+03
2.33E+03	2.66E+03	3.52E+03	3.30E+02	1.19E+03	8.60E+02
3.57E+05	4.85E+05	5.73E+05	1.28E+05	2.16E+05	8.79E+04
8.29E+02	1.18E+03	8.60E+02	3.46E+02	3.10E+01	-3.15E+02
1.04E+03	5.00E+02	1.40E+02	-5.40E+02	-9.00E+02	-3.60E+02
4.80E+04	2.26E+05	4.60E+04	1.78E+05	-2.00E+03	-1.80E+05
2.80E+04	2.90E+04	2.00E+04	1.00E+03	-8.00E+03	-9.00E+03
3.10E+04	3.20E+04	1.90E+04	1.00E+03	-1.20E+04	-1.30E+04
4.50E+04	8.40E+04	4.03E+05	3.90E+04	3.58E+05	3.19E+05
-4.00E+04	9.40E+04	4.16E+05	1.34E+05	4.56E+05	3.22E+05
-3.80E+04	1.05E+05	4.32E+05	1.43E+05	4.70E+05	3.27E+05
6.20E+04	4.29E+05	3.72E+05	3.67E+05	3.10E+05	-5.70E+04
5.50E+04	1.24E+05	3.50E+04	6.90E+04	-2.00E+04	-8.90E+04
3.00E+04	4.56E+05	4.48E+05	4.26E+05	4.18E+05	-8.00E+03
1.76E+05	1.61E+05	2.10E+05	-1.50E+04	3.40E+04	4.90E+04
-1.24E+06	-1.28E+06	-1.23E+06	-4.00E+04	1.00E+04	5.00E+04

2. Se encuentra la mediana de cada conjunto de diferencias (Z_{uv} , la cual puede considerarse como un estimador no ajustado de las medianas para los algoritmos u y v , $M_u - M_v$).

	Med 1,2	Med 1,3	Med 1,4	Med 2,3	Med 2,4	Med 3,4
Mediana	2.49E+03	2.90E+04	2.00E+04	3.46E+02	1.19E+03	1.73E+03

3. Se calcula la media de cada conjunto ajustado de medianas donde se tenga el mismo primer subscriptor, m_u :

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

m1	1.29E+04
m2	-2.38E+02
m3	-6.90E+03
m4	-5.73E+03

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 $m_1 - m_2$.

	PSO	SSGA	SS BLX	DE - EXP
PSO	0	1.31E+04	1.98E+04	1.86E+04
SSGA	-1.31E+04	0	6.67E+03	5.49E+03
SS BLX	-1.98E+04	-6.67E+03	0	-1.17E+03
DE-EXP	-1.86E+04	-5.49E+03	1.17E+03	0

Anexo C.3 Prueba de Friedman

Tabla C.3.1 Rankings para la prueba de Friedman para el ejemplo.

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

Para cada fila se calcula el ranking y se hace la sumatoria por cada columna (algoritmo)

	Total Ranking	R_j
• PSO	84.5	3.38
• SSGA	64	2.56
• SSBL	58.5	2.34
• DE-EXP	43	1.72

Los rankings medios proporcionan una comparación interesante de los algoritmos. En media, DE-EXP obtuvo el mejor ranking (1,72), seguido por SS-BLX y SSGA ,mientras que PSO ofrece el peor resultado.

En este punto, la prueba de Friedman procede comprobando si los rankings medios obtenidos son significativamente diferentes del ranking medio esperado bajo la hipótesis nula,

- $Rj = 2, 5$

$$F_F = \frac{12 \cdot 25}{4(4+1)} \cdot$$

$$\left[(3,38)^2 + (2,56)^2 + (2,34)^2 + (1,72)^2 - \frac{4(4+1)^2}{4} \right] = 21,18$$

$$F_{ID} = \frac{(25-1)21,18}{25(4-1)-21,18} = 9,44$$

Con cuatro algoritmos y 25 conjuntos, FID se distribuye de acuerdo a una distribución F con $4-1 = 3$ y $(4-1)(25-1) = 72$ grados de libertad.

El p-valor calculado usando la distribución $F(3, 72)$ es $2,46E- 05$, por lo que la hipótesis nula se rechaza con una alta probabilidad.

Anexo D

Códigos en R para comprobar pruebas estadísticas no paramétricas usadas en experimentaciones

Código D.1 Código en R para prueba Estimación de contraste

```
#Estimación de contaste

#Se enumera cada algoritmo:
# PSO - 1
# SSGA - 2
# SS-BLX - 3
# DE-Exp - 4

algoritmo1 <- c(1.23E-01, 2.60E+01, 5.17E+07, 2.49E+03, 4.10E+05, 7.31E+05, 2.68E+02, 2.04E+04, 1.44E+04,
1.40E+04, 5.59E+03, 6.36E+05, 1.50E+03, 3.30E+03, 3.40E+05, 1.33E+05, 1.50E+05, 8.51E+05, 8.50E+05,
8.51E+05, 9.14E+05, 8.07E+05, 1.03E+06, 4.12E+05, 5.10E+05)
algoritmo2 <- c(8.42E-06, 8.72E-02, 7.95E+07, 2.59E+00, 1.34E+05, 6.17E+03, 1.27E+06, 2.04E+04, 7.29E-06,
1.71E+04, 3.26E+03, 2.79E+05, 6.71E+02, 2.26E+03, 2.92E+05, 1.05E+05, 1.19E+05, 8.06E+05, 8.90E+05,
8.89E+05, 8.52E+05, 7.52E+05, 1.00E+06, 2.36E+05, 1.75E+06)
algoritmo3 <- c(3.40E+02, 1.73E+03, 1.84E+08, 6.23E+03, 2.19E+03, 1.15E+05, 1.97E+06, 2.04E+04, 4.20E+03,
1.24E+04, 2.93E+03, 1.51E+05, 3.25E+02, 2.80E+03, 1.14E+05, 1.04E+05, 1.18E+05, 7.67E+05, 7.56E+05,
7.46E+05, 4.85E+05, 6.83E+05, 5.74E+05, 2.51E+05, 1.79E+06)
algoritmo4 <- c(8.26E-06, 8.18E-06, 9.94E+04, 8.35E-06, 8.51E-06, 8.39E-06, 1.27E+06, 2.04E+04, 8.15E-06,
1.12E+04, 2.07E+03, 6.31E+04, 6.40E+02, 3.16E+03, 2.94E+05, 1.13E+05, 1.31E+05, 4.48E+05,
4.34E+05, 4.19E+05, 5.42E+05, 7.72E+05, 5.82E+05, 2.02E+05, 1.74E+06)

# Se calcula la diferencia entre cada algoritmo donde u < v
# difu_v
dif1_2 <- algoritmo1 - algoritmo2; dif1_2
dif1_3 <- algoritmo1 - algoritmo3; dif1_3
dif1_4 <- algoritmo1 - algoritmo4; dif1_4
dif2_3 <- algoritmo2 - algoritmo3; dif2_3
dif2_4 <- algoritmo2 - algoritmo4; dif2_4
dif3_4 <- algoritmo3 - algoritmo4; dif3_4

# Se obtiene la mediana de cada diferencia.
# Zuv
Z12 <- median(dif1_2); Z12
Z13 <- median(dif1_3); Z13
Z14 <- median(dif1_4); Z14
Z23 <- median(dif2_3); Z23
Z24 <- median(dif2_4); Z24
Z34 <- median(dif3_4); Z34

# Media de medianas donde se tiene el mismo prefijo u en mu.
m1 <- (Z12 + Z13 + Z14) / 4; m1
m2 <- (-Z12 + Z23 + Z24) / 4; m2
```

```

m3 <- (-Z13 - Z23 + Z34) / 4; m3
m4 <- (-Z14 - Z24 - Z34) / 4; m4

m <- matrix(0, 4, 4)
m[1, 2] <- m1 - m2
m[1, 3] <- m1 - m3
m[1, 4] <- m1 - m4
m[2, 3] <- m2 - m3
m[2, 4] <- m2 - m4
m[3, 4] <- m3 - m4

m[2, 1] <- m2 - m1
m[3, 1] <- m3 - m1
m[4, 1] <- m4 - m1
m[3, 2] <- m3 - m2
m[4, 2] <- m4 - m2
m[4, 3] <- m4 - m3
m

```

Código D.2 Código en R para prueba de Friedman

#Algoritmo	PSO	SSGA	SS-BLX	DE-EXP
#F1	1.23E-01	8.42E-06	3.40E+02	8.26E-06
#F2	2.60E+01	8.72E-02	1.73E+03	8.18E-06
#F3	5.17E+07	7.95E+07	1.84E+08	9.94E+04
#F4	2.49E+03	2.59E+00	6.23E+03	8.35E-06
#F5	4.10E+05	1.34E+05	2.19E+03	8.51E-06
#F6	7.31E+05	6.17E+03	1.15E+05	8.39E-06
#F7	2.68E+02	1.27E+06	1.97E+06	1.27E+06
#F8	2.04E+04	2.04E+04	2.04E+04	2.04E+04
#F9	1.44E+04	7.29E-06	4.20E+03	8.15E-06
#F10	1.40E+04	1.71E+04	1.24E+04	1.12E+04
#F11	5.59E+03	3.26E+03	2.93E+03	2.07E+03
#F12	6.36E+05	2.79E+05	1.51E+05	6.31E+04
#F13	1.50E+03	6.71E+02	3.25E+02	6.40E+02
#F14	3.30E+03	2.26E+03	2.80E+03	3.16E+03
#F15	3.40E+05	2.92E+05	1.14E+05	2.94E+05
#F16	1.33E+05	1.05E+05	1.04E+05	1.13E+05
#F17	1.50E+05	1.19E+05	1.18E+05	1.31E+05
#F18	8.51E+05	8.06E+05	7.67E+05	4.48E+05
#F19	8.50E+05	8.90E+05	7.56E+05	4.34E+05
#F20	8.51E+05	8.89E+05	7.46E+05	4.19E+05
#F21	9.14E+05	8.52E+05	4.85E+05	5.42E+05
#F22	8.07E+05	7.52E+05	6.83E+05	7.72E+05
#F23	1.03E+06	1.00E+06	5.74E+05	5.82E+05
#F24	4.12E+05	2.36E+05	2.51E+05	2.02E+05
#F25	5.10E+05	1.75E+06	1.79E+06	1.74E+06
#Datos				
n=25 #numero de pruebas				
k=4 #algoritmos a comparar				

```

#1 De la tabla principal se comparan cada una de las pruebas
#y se les asigna un ranking en forma ascendente (al mejor resultado un 1.. y asi
#consecutivamente) si en algun resultado se obtiene la misma respuesta de colocan
#un numero decimal, como a continuacion se observa

PSO<-c(3,3,2,3,4,4,1,2,5,4,3,4,4,4,4,4,4,4,3,3,4,4,4,4,1)
SSGA<-c(2,2,3,2,3,2,2,5,2,5,1,4,3,3,3,1,2,2,2,3,4,4,3,2,3,2,3)
SSBL<-c(4,4,4,4,2,3,4,2,5,3,2,2,2,1,2,1,1,1,2,2,2,1,1,1,3,4)
DEXP<-c(1,1,1,1,1,1,2,5,2,5,2,1,1,1,2,3,3,3,1,1,1,2,3,2,1,2)

#se calcula el ranking total de cada una de las pruebas

TotalRankingPSO<-sum(PSO);TotalRankingPSO
TotalRankingSSGA<-sum(SSGA);TotalRankingSSGA
TotalRankingSSBL<-sum(SSBL);TotalRankingSSBL
TotalRankingDEXP<-sum(DEXP);TotalRankingDEXP

#Se obtiene Rj que es la sumatoria total de cada una de las pruebas entre n
RjPSO<-TotalRankingPSO/n;RjPSO
RjSSGA<-TotalRankingSSGA/n;RjSSGA
RjSSBL<-TotalRankingSSBL/n;RjSSBL
RjDEXP<-TotalRankingDEXP/n;RjDEXP

#Una vez obtenidos estos datos es necesario realizar el calculo de FF
FF1<-(12*n)/(k*(k+1));FF1
SumiRj<-(RjPSO^2)+(RjSSGA^2)+(RjSSBL^2)+(RjDEXP^2))-((k*((k+1)^2))/4);SumiRj
FF<-FF1*SumiRj;FF
FID<- ((n-1)*FF)/ ((n*(k-1))-FF);FID

pf(FID,3,72, lower.tail=FALSE)
#####
Ri<-RjDEXP
RjPSO
RjSSGA
RjSSBL

zPSO<-(Ri-RjPSO)/sqrt((k*(k+1))/(6*n)); zPSO
zSSGA<-(Ri-RjSSGA)/sqrt((k*(k+1))/(6*n)); zSSGA
zSSBL<-(Ri-RjSSBL)/sqrt((k*(k+1))/(6*n)); zSSBL

#calcular el p-value no ajustado
pnorm(zPSO)*2
pnorm(zSSGA)*2
pnorm(zSSBL)*2

```

Anexo E

Tabla E.11 – Errores promedio obtenido en las 25 funciones de referencia.

Function	PSO	IPOP-CMA-ES	CHC	SSGA	SS-BLX	SS-Arit	DE-Bin	DE-Exp	SaDE
F1	$1.234 \cdot 10^{-4}$	0.000	2.464	$8.420 \cdot 10^{-9}$	3.402 · 10	1.064	$7.716 \cdot 10^{-9}$	$8.260 \cdot 10^{-9}$	$8.416 \cdot 10^{-9}$
F2	$2.595 \cdot 10^{-2}$	0.000	$1.180 \cdot 10^2$	$8.719 \cdot 10^{-5}$	1.730	5.282	$8.342 \cdot 10^{-9}$	$8.181 \cdot 10^{-9}$	$8.208 \cdot 10^{-9}$
F3	$5.174 \cdot 10^4$	0.000	$2.699 \cdot 10^5$	$7.948 \cdot 10^4$	$1.844 \cdot 10^5$	$2.535 \cdot 10^5$	$4.233 \cdot 10$	$9.935 \cdot 10$	$6.560 \cdot 10^3$
F4	2.488	$2.932 \cdot 10^3$	$9.190 \cdot 10$	$2.585 \cdot 10^{-3}$	6.228	5.755	$7.686 \cdot 10^{-9}$	$8.350 \cdot 10^{-9}$	$8.087 \cdot 10^{-9}$
F5	$4.095 \cdot 10^2$	$8.104 \cdot 10^{-10}$	$2.641 \cdot 10^2$	$1.343 \cdot 10^2$	2.185	$1.443 \cdot 10$	$8.608 \cdot 10^{-9}$	$8.514 \cdot 10^{-9}$	$8.640 \cdot 10^{-9}$
F6	$7.310 \cdot 10^2$	0.000	$1.416 \cdot 10^6$	6.171	$1.145 \cdot 10^2$	$4.945 \cdot 10^2$	$7.956 \cdot 10^{-9}$	$8.391 \cdot 10^{-9}$	$1.612 \cdot 10^{-2}$
F7	$2.678 \cdot 10$	$1.267 \cdot 10^3$	$1.269 \cdot 10^3$	$1.271 \cdot 10^3$	$1.966 \cdot 10^3$	$1.908 \cdot 10^3$	$1.266 \cdot 10^3$	$1.265 \cdot 10^3$	$1.263 \cdot 10^3$
F8	$2.043 \cdot 10$	$2.001 \cdot 10$	$2.034 \cdot 10$	$2.037 \cdot 10$	$2.035 \cdot 10$	$2.036 \cdot 10$	$2.033 \cdot 10$	$2.038 \cdot 10$	$2.032 \cdot 10$
F9	$1.438 \cdot 10$	$2.841 \cdot 10$	5.886	$7.286 \cdot 10^{-9}$	4.195	5.960	4.546	$8.151 \cdot 10^{-9}$	$8.330 \cdot 10^{-9}$
F10	$1.404 \cdot 10$	$2.327 \cdot 10$	7.123	$1.712 \cdot 10$	$1.239 \cdot 10$	$2.179 \cdot 10$	$1.228 \cdot 10$	$1.118 \cdot 10$	$1.548 \cdot 10$
F11	5.590	1.343	1.599	3.255	2.929	2.858	2.434	2.067	6.796
F12	$6.362 \cdot 10^2$	$2.127 \cdot 10^2$	$7.062 \cdot 10^2$	$2.794 \cdot 10^2$	$1.506 \cdot 10^2$	$2.411 \cdot 10^2$	$1.061 \cdot 10^2$	$6.309 \cdot 10$	$5.634 \cdot 10$
F13	1.503	1.134	$8.297 \cdot 10$	6.713 · 10	$3.245 \cdot 10$	$5.479 \cdot 10$	1.573	$6.403 \cdot 10$	$7.070 \cdot 10$
F14	3.304	3.775	2.073	2.264	2.796	2.970	3.073	3.158	3.415
F15	$3.398 \cdot 10^2$	$1.934 \cdot 10^2$	$2.751 \cdot 10^2$	$2.920 \cdot 10^2$	$1.136 \cdot 10^2$	$1.288 \cdot 10^2$	$3.722 \cdot 10^2$	$2.940 \cdot 10^2$	$8.423 \cdot 10$
F16	$1.333 \cdot 10^2$	$1.170 \cdot 10^2$	$9.729 \cdot 10$	$1.053 \cdot 10^2$	$1.041 \cdot 10^2$	$1.134 \cdot 10^2$	$1.117 \cdot 10^2$	$1.125 \cdot 10^2$	$1.227 \cdot 10^2$
F17	$1.497 \cdot 10^2$	$3.389 \cdot 10^2$	$1.045 \cdot 10^2$	$1.185 \cdot 10^2$	$1.183 \cdot 10^2$	$1.279 \cdot 10^2$	$1.421 \cdot 10^2$	$1.312 \cdot 10^2$	$1.387 \cdot 10^2$
F18	$8.512 \cdot 10^2$	$5.570 \cdot 10^2$	$8.799 \cdot 10^2$	$8.063 \cdot 10^2$	$7.668 \cdot 10^2$	$6.578 \cdot 10^2$	$5.097 \cdot 10^2$	$4.482 \cdot 10^2$	$5.320 \cdot 10^2$
F19	$8.497 \cdot 10^2$	$5.292 \cdot 10^2$	$8.798 \cdot 10^2$	$8.899 \cdot 10^2$	$7.555 \cdot 10^2$	$7.010 \cdot 10^2$	$5.012 \cdot 10^2$	$4.341 \cdot 10^2$	$5.195 \cdot 10^2$
F20	$8.509 \cdot 10^2$	$5.264 \cdot 10^2$	$8.960 \cdot 10^2$	$8.893 \cdot 10^2$	$7.463 \cdot 10^2$	$6.411 \cdot 10^2$	$4.928 \cdot 10^2$	$4.188 \cdot 10^2$	$4.767 \cdot 10^2$
F21	$9.138 \cdot 10^2$	$4.420 \cdot 10^2$	$8.158 \cdot 10^2$	$8.522 \cdot 10^2$	$4.851 \cdot 10^2$	$5.005 \cdot 10^2$	$5.240 \cdot 10^2$	$5.420 \cdot 10^2$	$5.140 \cdot 10^2$
F22	$8.071 \cdot 10^2$	$7.647 \cdot 10^2$	$7.742 \cdot 10^2$	$7.519 \cdot 10^2$	$6.828 \cdot 10^2$	$6.941 \cdot 10^2$	$7.715 \cdot 10^2$	$7.720 \cdot 10^2$	$7.655 \cdot 10^2$
F23	$1.028 \cdot 10^3$	$8.539 \cdot 10^2$	$1.075 \cdot 10^3$	$1.004 \cdot 10^3$	$5.740 \cdot 10^2$	$5.828 \cdot 10^2$	$6.337 \cdot 10^2$	$5.824 \cdot 10^2$	$6.509 \cdot 10^2$
F24	$4.120 \cdot 10^2$	$6.101 \cdot 10^2$	$2.959 \cdot 10^2$	$2.360 \cdot 10^2$	$2.513 \cdot 10^2$	$2.011 \cdot 10^2$	$2.060 \cdot 10^2$	$2.020 \cdot 10^2$	$2.000 \cdot 10^2$
F25	$5.099 \cdot 10^2$	$1.818 \cdot 10^3$	$1.764 \cdot 10^3$	$1.747 \cdot 10^3$	$1.794 \cdot 10^3$	$1.804 \cdot 10^3$	$1.744 \cdot 10^3$	$1.742 \cdot 10^3$	$1.738 \cdot 10^3$

Anexo F

Funciones de Prueba

Se puede consultar en [Suganthan 2005] la descripción completa de las funciones. El conjunto de funciones de prueba está compuesto por las siguientes funciones:

5 Funciones Unimodales

- Función Esfera desplazada.
- Problema 1.2 de Schwefel desplazado.
- Función Elíptica rotada ampliamente condicionada.
- Problema desplazado Schwefel 1.2 con ruido en el Fitness.
- Problema de Schwefel 2.6 con el óptimo global en la frontera.

20 Funciones Multimodales

- 7 Funciones básicas
 - Función Rosenbrock desplazada.
 - Función Griewank desplazada y rotada sin fronteras.
 - Función Ackley desplazada y rotada con óptimo global en la frontera.
 - Función Rastrigin desplazada.
 - Función Rastrigin desplazada y rotada.
 - Función Wieierstrass desplazada y rotada.
 - Problema 2.12 de Schwefel.
- 2 Funciones Expandidas.
- 11 Funciones híbridas. Cada una de éstas se han definido mediante composición de 10 de las 14 funciones anteriores (distintas en cada caso).

Anexo G

Instancias usadas para las pruebas no paramétricas en VisTHAA

Anexo G.1 Instancia con datos del artículo [Derrac 2012] para: prueba de signos múltiples, estimación de contraste y prueba de Friedman.

	4	25		
	Algoritmo	PSO	SSGA	SS-BLXDE-EXP
F1	1.23E-01	8.42E-06	3.40E+02	8.26E-06
F2	2.60E+01	8.72E-02	1.73E+03	8.18E-06
F3	5.17E+07	7.95E+07	1.84E+08	9.94E+04
F4	2.49E+03	2.59E+00	6.23E+03	8.35E-06
F5	4.10E+05	1.34E+05	2.19E+03	8.51E-06
F6	7.31E+05	6.17E+03	1.15E+05	8.39E-06
F7	2.68E+02	1.27E+06	1.97E+06	1.27E+06
F8	2.04E+04	2.04E+04	2.04E+04	2.04E+04
F9	1.44E+04	7.29E-06	4.20E+03	8.15E-06
F10	1.40E+04	1.71E+04	1.24E+04	1.12E+04
F11	5.59E+03	3.26E+03	2.93E+03	2.07E+03
F12	6.36E+05	2.79E+05	1.51E+05	6.31E+04
F13	1.50E+03	6.71E+02	3.25E+02	6.40E+02
F14	3.30E+03	2.26E+03	2.80E+03	3.16E+03
F15	3.40E+05	2.92E+05	1.14E+05	2.94E+05
F16	1.33E+05	1.05E+05	1.04E+05	1.13E+05
F17	1.50E+05	1.19E+05	1.18E+05	1.31E+05
F18	8.51E+05	8.06E+05	7.67E+05	4.48E+05
F19	8.50E+05	8.90E+05	7.56E+05	4.34E+05
F20	8.51E+05	8.89E+05	7.46E+05	4.19E+05
F21	9.14E+05	8.52E+05	4.85E+05	5.42E+05
F22	8.07E+05	7.52E+05	6.83E+05	7.72E+05
F23	1.03E+06	1.00E+06	5.74E+05	5.82E+05
F24	4.12E+05	2.36E+05	2.51E+05	2.02E+05
F25	5.10E+05	1.75E+06	1.79E+06	1.74E+06

Anexo G.2 Instancia con datos del artículo [García 2009] para la prueba de Wilcoxon de signos con rangos

25

G-CMA-ED	CoEVO
1.00E-08	1.00E-08
1.00E-08	2.13E+00
1.00E-08	1.25E+01
1.00E-08	3.71E-02
2.00E+01	2.03E+01
2.39E-01	1.92E+01
7.96E-02	2.68E+01
9.34E-01	9.029
2.93E+01	6.05E+02
6.96E-01	1.137
3.01E+00	3.71E+00
2.28E+02	2.94E+02
9.13E+01	1.77E+02
1.23E+02	2.12E+02
3.32E+02	9.01E+02
3.26E+02	8.45E+02
3.00E+02	8.63E+02
5.00E+02	6.35E+02
7.29E+02	7.79E+02
5.59E+02	8.35E+02
2.00E+02	3.14E+02
3.74E+02	2.57E+02

Anexo G.3 Instancia para hacer prueba de signos múltiples, estimación de contraste y prueba de Friedman para comparar GGA-CGT con otros algoritmos que resuelven BPP

6	17	Instancias	GGA-CGT	HGGA2	WABP	FF	BF	WF
TEST0022.txt			0	0	0	0	0	0
TEST0065.txt			0	0	0	0	0	0
TEST0097.txt			0	0	0	0.08333333	0.08333333	0.08333333
TEST0058.txt			0	0	0.05	0.05	0.05	0.05
TEST0055a.txt			0	0	0	0.06666667	0.06666667	0.06666667
TEST0049.txt			0	0	0	0.09090909	0.09090909	0.09090909
TEST0075.txt			0	0	0	0.07692308	0.07692308	0.07692308
TEST0054.txt			0	0	0	0.07142857	0.07142857	0.07142857
TEST0068.txt			0	0	0	0.08333333	0.08333333	0.08333333
TEST0014.txt	0.04347826	0.04347826	0.04347826	0.04347826	0.04347826	0.04347826	0.04347826	0.04347826
TEST0082.txt			0	0.04166667	0.04166667	0.04166667	0.04166667	0.04166667
TEST0044.txt			0	0	0	0.07142857	0.07142857	0.07142857
TEST0030.txt			0	0	0.03703704	0.03703704	0.03703704	0.03703704
TEST0005.txt			0	0	0.03571429	0.03571429	0.03571429	0.03571429
TEST0095.txt			0	0	0	0.0625	0.0625	0.0625
TEST0055b.txt			0	0	0	0.05	0.05	0.05
TEST0084.txt			0	0	0	0.0625	0.0625	0.0625