**CENTRO NACIONAL DE INVESTIGACIÓN
Y DESARROLLO TECNOLÓGICO**

cenidet

CLASIFICACIÓN DE ALGORITMOS HEURÍSTICOS PARA LA SOLUCIÓN DE PROBLEMAS DE BIN PACKING

T E S I S

QUE PARA OBTENER EL GRADO DE
DOCTOR EN CIENCIAS EN CIENCIAS
DE LA COMPUTACIÓN

P R E S E N T A :

LAURA CRUZ REYES

DIRECTORES DE TESIS :

DR. JOAQUÍN PÉREZ ORTEGA
DR. RODOLFO A. PAZOS RANGEL

CUERNAVACA MORELOS JULIO 2004

DEDICATORIA

Dedico esta tesis a mi esposo Enrique Gómez Benítez, y a mis hijos Laura y Enrique Gómez Cruz, por el apoyo constante, la paciencia enorme y sacrificada, y el gran amor que nos une sólidamente.

A la memoria de mis padres Román y María, y mi hermano Moisés, por sus enseñanzas de esfuerzo perpetuo, afectivo y respetuoso.

A mis hermanos Elvira, Noé y Graciela por el cariño incondicional que siempre me han mostrado.

A la pequeña Danna por iluminar nuestras vidas y permitirnos amarla.

A todos los miembros de la familias Cruz, Gómez, Andrade y De la Fuente por la voluntad de conservar los lazos familiares.

RECONOCIMIENTOS

Mi profundo agradecimiento a los miembros del comité tutorial de esta tesis: Dr. Cesar Guerra Salcedo, Dr. Crispín Zavala Díaz, Dr. David Romero, Dr. Guillermo Rodríguez Ortiz, Dr. Joaquín Pérez Ortega, Dr. Juan Frausto Solís, y Dr. Raúl Pinto Elías.

En especial, mi sincero aprecio al Dr. Joaquín Pérez Ortega y Dr. Rodolfo Pazos Rangel por haber dirigido esta tesis; y al Dr. Juan Frausto Solís, Dr. Cesar Guerra Salcedo y Dr. David Romero por sus valiosas sugerencias y críticas.

Reciban mi reconocimiento las instituciones educativas participantes. Estoy en deuda con el Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET), Instituto Tecnológico de Ciudad Madero (ITCM), Instituto Tecnológico y de Estudios Superiores de Monterrey (ITESM), Instituto de Investigaciones Eléctricas (IIE), y la Minnesota State University (MSU) quienes proporcionaron todas las facilidades necesarias para esta investigación.

Finalmente, doy gracias a mis compañeros del ITCM por su ayuda, soporte moral y amistad. Para ellos mi estimación.

RESUMEN

En este trabajo se abordó el problema de selección de algoritmos, esto es, dado un conjunto de casos de un problema resuelto con varios algoritmos, para un nuevo caso pronosticar cuál es el algoritmo que lo resuelve mejor. En particular, fue de interés su aplicación en la solución de problemas de *Bin-packing*.

El método clásico para la selección consiste en determinar el mejor algoritmo respecto a un criterio dado, y resolver los nuevos casos con el algoritmo seleccionado. Un método posterior consistió en desarrollar, para cada algoritmo, funciones que relacionan su desempeño con el tamaño del problema; para un nuevo caso se evalúan las funciones y se selecciona el algoritmo. En uno de los métodos más recientes, el usuario define una taxonomía de casos, y para cada clase se almacenan las estadísticas de desempeño de cada algoritmo con cada caso; para un nuevo caso se determina a qué clase de la taxonomía pertenece, y a partir de los datos estadísticos se pronostica cuál algoritmo lo resolverá mejor. Dichos métodos están limitados para hacer una buena selección ya que por una parte los criterios para la formación de grupos no son formales, y por otra, no definen sistemáticamente las características críticas y sus interrelaciones. Este último aspecto incide notablemente en el desempeño de los algoritmos.

En esta tesis se propone una metodología basada en el aprendizaje automático y la estadística, que permite identificar características críticas y sus interrelaciones, lo cual posibilita que para cada algoritmo se determine un patrón de agrupamiento de los casos que ha resuelto mejor. Para un nuevo caso se determina a qué patrón de agrupamiento es más afín y se selecciona el algoritmo. La metodología propuesta se validó con un conjunto de pruebas que incorporan la interrelación entre cinco características críticas y el desempeño de siete algoritmos. Los pronósticos acertaron el 77% de las veces y generaron un error en la solución del 3.8%, en contraste con el 33% de aciertos y un error del 41% de una selección aleatoria. Se considera que el método seguido puede ser aplicado a la solución de otros problemas y a la caracterización del desempeño de otros algoritmos.

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

Capítulo 1

INTRODUCCIÓN

En este trabajo se aborda el problema de selección de algoritmos heurísticos, en el cual, para un caso dado de un problema de optimización combinatoria, se selecciona el mejor algoritmo para su solución. La falta de herramientas analíticas para estudiar el desempeño de este tipo de algoritmos, ha dificultado el desarrollo de métodos formales para la evaluación y selección de los mismos [Papadimitriou 1998, Reeves 1993]. Para aportar en esta dirección, se propone una metodología para la construcción de modelos matemáticos que permiten elegir el algoritmo con el mejor desempeño estimado.

En las secciones de este capítulo se presenta un panorama general de la tesis; el cual se inicia con la descripción de los motivos que llevaron a esta investigación y se continúa con la definición del problema de investigación. Enseguida se explican los objetivos que se plantearon alcanzar. Se explica también el contexto en el que se desarrolló este trabajo, y la justificación de usar en los experimentos el problema de distribución de objetos en contenedores. Se da una breve introducción a los algoritmos heurísticos y se termina dando una descripción del contenido de cada capítulo de la tesis.

1.1 MOTIVACIONES

En trabajos previos se han desarrollado modelos de optimización combinatoria, y para su solución se encontró que no existe un algoritmo que supere a los demás. De manera natural, esto conduce a la selección de algoritmos, el cual es un problema abierto de la computación [Anderson 2003]. De acuerdo con [McGeoch 2002], no existen herramientas analíticas adecuadas para estudiar el desempeño de algoritmos heurísticos, las que existen son de poca utilidad práctica: unas, demasiado pesimistas, y otras, muy simples.

En esta tesis se planteó la construcción de modelos de predicción a partir de investigación experimental. Este tipo de modelos es importante para las ciencias computacionales [Hooker 1996]. Además de permitir la selección de algoritmos, desde el punto de vista teórico podría proporcionar directrices para entender el comportamiento de los algoritmos.

Por otro lado, actualmente existe un gran número de problemas reales de optimización combinatoria clasificados como NP-duros, los cuales por su complejidad requieren métodos heurísticos para su solución. Entre ellos se pueden citar los siguientes: la distribución de objetos de datos en bases de datos distribuidas, la distribución de objetos en contenedores, la asignación de tareas a máquinas dentro de líneas de producción grandes y la programación de horarios de universidades. La metodología y algoritmos que se proponen en esta tesis, podrían ayudar a mejorar los resultados computacionales en la solución de este tipo de problemas.

1.2 DESCRIPCIÓN DEL PROBLEMA DE INVESTIGACIÓN

Dados un conjunto de casos C de un problema de optimización combinatoria y un conjunto A de dos o más algoritmos de solución heurística.

¿Es posible agrupar todos los casos de C para los cuales un algoritmo de A tuvo un desempeño igual o mejor que los otros algoritmos de A? Esto es, para una función $d(a, c)$ que cuantifica el desempeño de un algoritmo $a \in A$ sobre un caso $c \in C$, encontrar un conjunto disjunto de grupos G tal que

a) todo grupo g tenga asociado un algoritmo dominante

$$\forall\, g \in G, \exists\, \{\, (a \in A,\ C_i \subseteq C) \mid \forall\, b \in (\mathrm{A} - a), \forall\, c \in C_i\ \ d(a, c) \geq d(b, c)\},$$

b) y todos los casos de C estén ubicados en algún grupo de G

$$C = \bigcup_{i=1}^{|G|} C_i.$$

¿Es posible proyectar un caso nuevo $x \notin C$, a una región de solución $g \in G$ dominada por un algoritmo $a \in A$? Esto es, encontrar una función

$$p\colon\ C^c \ \rightarrow G$$
$$x \rightarrow g = p(x)$$

tal que $\forall b \in (\mathrm{A} - a),\ \ d(a, x) \geq d(b, x).$

A continuación se muestra un ejemplo sencillo de una selección de algoritmos con los algoritmos heurísticos TA, ACO y FFD, los cuales se describen en las secciones 1.4.4. y 3.2.4.1. En la Figura 1.1, el universo es un conjunto de casos C de un problema de optimización combinatoria, G es un conjunto de grupos, y cada grupo tiene asociado un algoritmo y el conjunto de casos que mejor resuelve. Para un nuevo caso x cuya solución se desconoce, la función $p(x)$ proyecta el caso nuevo a una región dominada por un algoritmo. En el ejemplo, los casos del grupo g_1 tienen características similares al nuevo caso, por lo tanto, el algoritmo TA, asociado a este grupo, se selecciona para dar solución a x.

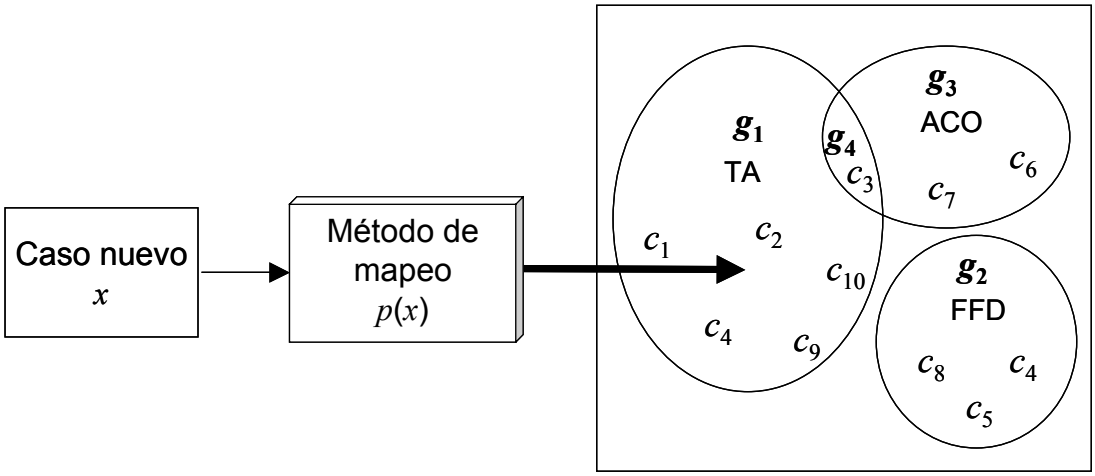


Figura 1.1 Proyección de un caso nuevo a una región

1.3 OBJETIVO DE LA TESIS

Dada la necesidad de contar con métodos de selección de algoritmos para resolver problemas *NP*-duros de una mejor manera, y debido a que los métodos propuestos están limitados para hacer una buena selección, en esta tesis se establecieron los siguientes objetivos:

a) Desarrollar una metodología de caracterización de algoritmos, que permita construir modelos de clasificación para seleccionar el algoritmo que mejor resuelve un caso de un problema dado.
b) Incorporar en el modelo de clasificación aspectos críticos para el desempeño de algoritmos, los cuales estén relacionados con las características de complejidad del problema. Esto con la finalidad de incrementar la precisión en la selección de algoritmos.
c) Explorar la implementación de la metodología en la caracterización y selección de algoritmos heurísticos. Esto con el propósito de resolver casos representativos del problema de distribución de Bases de Datos y del problema

de distribución de objetos en contenedores. Se justifica el uso de métodos heurísticos debido a que estos problemas de distribución tienen al menos la complejidad de un problema NP-duro [Pérez 2000a, Lin 1995].

Para cumplir los objetivos planteados, en este trabajo se propone una metodología, basada en aprendizaje automático y la estadística, para la creación de modelos matemáticos que permitan seleccionar de manera automática el algoritmo heurístico más adecuado a las condiciones especificadas del problema a resolver.

1.4 CONTEXTO DE LA INVESTIGACIÓN

En el Centro Nacional de Investigación y Desarrollo Tecnológico (Cenidet), se han venido desarrollando trabajos en el modelado del diseño de la distribución de bases de datos, y se estableció la necesidad de utilizar métodos heurísticos para su solución. Sin embargo, ninguno mostró superioridad sobre los demás [Pérez 1999]. Esto llevó a trabajar en el área de selección de algoritmos.

En esta sección se describe el problema de distribución de bases de datos, el problema de distribución de objetos en contenedores (*Bin-packing*) y una breve introducción a los algoritmos heurísticos. Se describe también la relación de complejidad que existe entre los problemas de distribución, y se justifica la validación con *Bin-packing* de la metodología de selección propuesta.

1.4.1 El Problema de Distribución de Bases de Datos Distribuidas: DFAR

La Figura 1.2 muestra un ejemplo sencillo de diseño de distribución de bases de datos distribuidas. En dicha figura se representa la red de comunicaciones para computadoras como una nube que permite a un conjunto de nodos estar interconectados. En este caso los nodos se identifican con la letra s_i, las

aplicaciones se identifican con la letra q_i, finalmente los fragmentos en que se divide una relación, se identifican con la letra f_i. La ubicación optimizada de estos fragmentos minimiza los costos de operación del sistema distribuido.

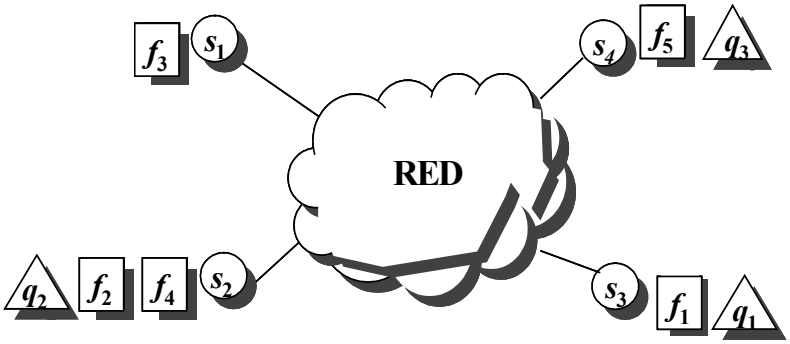


Figura 1.2 Ejemplo de un diseño de distribución

Tradicionalmente se ha considerado que el diseño de la distribución consta de dos fases consecutivas: fragmentación y ubicación. Contrario a esta creencia muy difundida, en [Pérez 1999, Pérez 2000b] se ha mostrado que es más sencillo combinar ambas fases. Para ello se formula un modelo matemático denominado DFAR (Distribution, Fragmentation, Allocation and Reallocation) que integra las dos fases, y se resuelve usando métodos heurísticos con buenos resultados. En este trabajo se hace una extensión al modelo DFAR. En particular se refina el modelado de la distribución replicada y se incorporan nuevos términos en la función objetivo. Los detalles de esta extensión se presentan en [Pérez 2003a, Pérez 2003b, Pérez 2003c].

1.4.2 El Problema de Distribución de Objetos en Contenedores: *Bin-packing*

El problema de distribución de objetos en contenedores (*Bin-packing*) es un problema clásico de optimización combinatoria NP-duro, en el cual hay una secuencia de n objetos $L = \{a_1, a_2, ..., a_n\}$, cada objeto con un tamaño dado $0 < s(a_i) \leq c$, y un número ilimitado de contenedores, cada uno de capacidad c. El objetivo

es determinar el menor número de contenedores m en los cuales todos los objetos pueden ser distribuidos. La expresión 1.1 enuncia este problema.

dado (1.1)

n = número de objetos a distribuir

c = capacidad del contenedor

L = secuencia de n objetos a_i

s_i (a_i)= tamaño de cada objeto a_i

encontrar

una partición de L mínima, $L = B_1 \cup B_2 \cup ..., \cup B_m$

tal que en cada conjunto B_j la sumatoria del tamaño de cada objeto $s(a_i)$ en B_j no exceda c,

$$\sum_{a_i \varepsilon B_j} s_i(a_i) \leq c \qquad \forall j,\ 1 \leq j \leq m.$$

En la versión discreta del problema *Bin-packing* de una-dimensión, la capacidad del contenedor es un entero c, el número de objetos es n, y por simplicidad, el tamaño de cada objeto es s_i, el cual es seleccionado del conjunto {1, 2, ..., c} [Coffman 2002].

1.4.3 Relación entre DFAR y *Bin-packing*

El problema DFAR es una generalización del problema *Bin-packing*. En trabajos previos se ha demostrado que el diseño de la distribución es al menos tan complejo como *Bin-packing* [Cruz 1999, Pérez 2000a]. En estos trabajos también se demostró que una versión simplificada de DFAR se puede reducir a este problema, y a su vez, *Bin-packing* a DFAR. Esto es, al resolver *Bin-packing*, se está

resolviendo un subconjunto de problemas de la distribución de bases de datos debido a la relación de equivalencia en la complejidad de los problemas.

Durante el avance de este trabajo, se desarrollaron métodos heurísticos para la solución de DFAR [Pérez 2003d, Pérez 2004b], y se realizó una experimentación preliminar en la cual se resolvieron casos medianos de 192 sitios en un tiempo promedio de 16752 segundos (5.31E-4 años). De manera simplificada, para la experimentación de esta tesis se consideró que se requeriría resolver al menos 150 casos medianos con 7 algoritmos heurísticos disponibles, y aplicar 30 veces cada algoritmo a cada caso para calcular su desempeño promedio. En estas condiciones, se estimó que la experimentación tomaría aproximadamente 17 años (5.31E-4$\times$150$\times$7$\times$30). Para propósitos de esta tesis resultaba prohibitivo el tiempo de procesamiento, por lo que se decidió utilizar casos de *Bin-packing*.

En resumen, como el propósito de esta investigación era la caracterización de algoritmos, se consideró razonable trabajar con *Bin-packing*, en particular con la versión discreta y de una dimensión, por las siguientes razones:

a) Es impráctico utilizar DFAR para validar experimentalmente la metodología de selección de algoritmos propuesta en esta tesis. La demostración de equivalencia de DFAR y *Bin-packing* permite asegurar que si se resuelve *Bin-packing* al menos se está resolviendo un subconjunto de DFAR.
b) Muchos problemas clásicos como *Bin-packing* cuentan con casos estándar reconocidos por la comunidad científica. El uso de este tipo de casos da formalidad a una investigación y permite contrastar los resultados obtenidos con los de otras investigaciones; los casos de DFAR no tienen esta ventaja.

1.4.4 Algoritmos Heurísticos

Cuando se tiene un problema clasificado como NP-duro y se desea resolver un caso grande del mismo, la única opción que queda, es dar una "buena solución" al

problema, pero no necesariamente la mejor. Para tal efecto se han desarrollado una serie de métodos denominados heurísticos, los cuales en general obtienen buenas soluciones, pero no garantizan una solución óptima.

Para el problema de *Bin-packing*, una solución óptima puede encontrarse considerando todas las formas de hacer una partición del conjunto de n objetos en un subconjunto de tamaño n o más pequeño; desafortunadamente, el número de posibles particiones es mayor que $(n/2)^{n/2}$ [Basse 1998]. Los algoritmos heurísticos abordados en esta tesis usan estrategias deterministas y no-deterministas para obtener soluciones subóptimas de *Bin-packing* con menos esfuerzo computacional que el probar todas las particiones.

El algoritmo FFD (First Fit Decreasing) es un ejemplo de algoritmo determinista, mientras que ACO (Ant Colony Optimization) es un ejemplo de algoritmo no determinista. Enseguida se da una descripción breve de estos dos tipos de algoritmos, y en la sección 3.2.4.1 se detallan los algoritmos heurísticos utilizados en esta tesis.

Los algoritmos deterministas siempre siguen la misma ruta para llegar a la solución. Por esta razón, obtienen la misma solución en diferentes ejecuciones. Los algoritmos deterministas aproximados para *Bin-packing* son sencillos y rápidos. Un análisis teórico de algoritmos de aproximación se presenta en [Coffman 1997, Coffman 1998, Lodi 2002].

Los algoritmos heurísticos de propósito general (meta-heurísticos) son no-deterministas, ya que generalmente no obtienen la misma solución en diferentes ejecuciones. Las metaheurísticas son métodos generales basados en búsqueda por entornos; parten de una solución inicial factible y mediante alteraciones de esa solución, van pasando a otras soluciones factibles de su entorno. Para la terminación del proceso se sigue algún criterio de parada, dando como resultado la mejor solución visitada. En particular, las metaheurísticas aplican estrategias inteligentes y aleatorias para evitar caer en mínimos locales.

1.5 ORGANIZACIÓN DEL DOCUMENTO

La tesis está organizada de la siguiente manera:

El Capítulo 2 aborda el problema de la selección de algoritmos. Como parte del marco teórico se presentan aspectos relacionados con la complejidad del problema y el análisis de desempeño de algoritmos, los cuales son esenciales para ubicar esta tesis en el contexto de otras investigaciones relacionadas. El capítulo finaliza con la revisión de otros trabajos enfocados a resolver el problema de la selección de algoritmos.

El Capítulo 3 presenta una metodología para caracterizar el desempeño de algoritmos heurísticos y aplicar dicha caracterización a la selección automática de los mismos. Expone la formulación de esta metodología en el contexto del aprendizaje automático, el cual tiene como objetivo el desarrollo de modelos de clasificación aprendidos a partir de datos históricos.

El Capítulo 4 muestra la experimentación realizada para validar la metodología propuesta para la selección de algoritmos heurísticos. Se describen los casos de prueba, el diseño del experimento y resultados.

El Capítulo 5 presenta las conclusiones a las que se llegó durante el desarrollo de esta investigación. El capítulo termina dando sugerencias de trabajos futuros.

Capítulo 2

CARACTERIZACIÓN Y SELECCIÓN DE ALGORITMOS

Los desarrollos en técnicas heurísticas para optimización combinatoria son alentadores, y el desempeño reportado para estos métodos sobre una variedad de tipos de problemas es excelente. Sin embargo, aún no se ha dado respuesta satisfactoria a la pregunta sobre qué tan bien se desempeñará un algoritmo heurístico, no sólo de manera general, sino también de manera particular en algún caso dado.

El análisis teórico y el análisis experimental están en constante evolución tratando de dar respuesta a la pregunta anterior y a otras interrogantes. Debido a la naturaleza aleatoria de los algoritmos heurísticos, el análisis experimental parece ser el más prometedor para ellos. En esta tesis se siguió este camino.

La sección 2.1 presenta el marco teórico de esta tesis. En esta sección se abordan conceptos relacionados con la complejidad del problema y el análisis de desempeño de algoritmos, los cuales son conceptos básicos para ubicar esta tesis en el contexto de otras investigaciones. En la sección 2.2 se describen los trabajos relacionados más relevantes que caracterizan experimentalmente el desempeño de los algoritmos.

2.1 MARCO TEÓRICO

En esta sección se presentan algunos conceptos básicos de la teoría de la NP-completez [Garey 1979, Papadimitriou 1998, Cormen 2001] y del análisis de algoritmos.

2.1.1 Teoría de Complejidad Computacional

Primero es importante entender la razón del desarrollo de la teoría de la NP-completez. Un problema fácil o tratable es aquél para el cual se puede construir un algoritmo determinista de tiempo polinomial que lo resuelve. La manera en que se prueba que un problema es tratable es presentando un algoritmo que resuelve el problema en un tiempo de ejecución que puede ser caracterizado como un polinomio. Los problemas de este tipo forman la clase P.

Para probar que un problema es intratable, se tiene que probar que no existe ningún algoritmo determinista que lo resuelva en tiempo polinomial o que todo algoritmo determinista que lo resuelve es de tiempo no polinomial. Probar cualquiera de estas proposiciones no es una tarea simple. Si para un problema dado después de múltiples intentos de resolverlo con algoritmos deterministas de tiempo polinomial, se tiene la sospecha de que es intratable y no se puede construir una prueba de su intratabilidad, ¿qué se puede hacer en tal situación?

La teoría de la NP-completez ofrece una alternativa. Los problemas NP-completos tienen la propiedad siguiente: Un problema NP-completo se resuelve en tiempo polinomial por un algoritmo determinista si y sólo si $P=NP$. Es decir basta resolver un solo problema NP-completo para dar solución a una conjetura que por varios decenios no se ha podido resolver [Garey 1979]. Esto, si bien no prueba que un problema NP-completo es intratable, indica que su solución es tan difícil como resolver la conjetura señalada. Por lo tanto, la respuesta a la cuestión planteada anteriormente es, probar que el problema es NP-completo, y con esto

justificar la posibilidad de resolver con un método exacto una versión simplificada del problema o utilizar un método de solución aproximada del problema original.

La teoría de la NP-completez se aplica únicamente a problemas de decisión, aquéllos que tienen una solución “sí” o “no”, en virtud de que su solución se define en función de la aceptación del lenguaje de casos codificados, por una máquina determinista de Turing (MDT). Como las máquinas de Turing sólo tienen la capacidad de aceptar o no una cadena, los únicos problemas que pueden resolver son los problemas de decisión.

Problema de decisión y algoritmo

Dos conceptos computacionales básicos que se formalizan son problema y algoritmo. Un problema de decisión se asocia con un lenguaje formal y un algoritmo con una máquina de Turing. Un problema de decisión Π consta de un conjunto de casos D, obtenidos a partir de un caso genérico, el cual se especifica en términos de varios componentes: conjuntos, grafos, funciones, números, etc. El conjunto D contiene un subconjunto de casos-sí $Y \subseteq D$. Un caso pertenece a Y si y sólo si, la respuesta para ese caso es “sí”.

Clase *P*

Es el conjunto de todos los problemas de decisión que pueden ser resueltos en tiempo polinomial por una MDT. A los problemas que pertenecen a esta clase se les denomina tratables.

Problemas intratables

Son problemas tan difíciles que ningún algoritmo de tiempo polinomial ha podido hasta ahora resolverlos, es decir, son todos los problemas que están en P^c.

Clase *NP*

Es el conjunto de todos los problemas de decisión que se resuelven en tiempo polinomial con una máquina no determinista de Turing (MNDT).

Relación entre *P* y *NP*

Como toda máquina determinista es una máquina no determinista, se tiene entonces que $P \subseteq NP$.

Transformación polinomial

Se dice que un problema $\Pi_1=(D_1, Y_1)$ se puede transformar polinomialmente al problema $\Pi_2=(D_2, Y_2)$, si existe una función $f: D_1 \rightarrow D_2$ que satisface las siguientes dos condiciones:

1. f es computable por un algoritmo determinista de tiempo polinomial.
2. Para todo caso $I \in D_1, I \in Y_1$ si y sólo si $f(I) \in Y_2$.

En tal caso se dice que hay una transformación polinomial de Π_1 a Π_2, y se escribe $\Pi_1 \leq_P \Pi_2$.

Equivalencia polinomial

Los problemas de decisión Π_1 y Π_2 son polinomialmente equivalentes ($\Pi_1 \approx \Pi_2$) si y sólo si $\Pi_1 \leq_P \Pi_2$ y $\Pi_2 \leq_P \Pi_1$.

Problemas *NP*-completos y *NP*-duros

Un problema Π es NP-completo si y sólo si,

1. $\Pi \in NP$ y
2. si $\Pi_1 \in NP-$completo implica que $\Pi_1 \leq_P \Pi$.

Esto es, que un problema $NP-$completo conocido se pueda transformar al problema de interés. Además, si se prueba la NP-completez del problema de decisión Π, entonces el problema de optimización asociado a Π es NP-duro.

2.1.2 Análisis de Algoritmos: Analítico vs. Experimental

Los métodos de caracterización del desempeño de los algoritmos se dividen en teóricos y experimentales. Los primeros son aquéllos en los que, para cada algoritmo, se determina matemáticamente la cantidad de recursos necesarios

como función del tamaño del caso considerado (mejor, peor o promedio). Los segundos son aquellos que se basan en la experimentación para realizar la caracterización.

El estudio teórico del desempeño de los algoritmos heurísticos basado en el análisis de complejidad de los casos medio y peor, con frecuencia es difícil de realizar y en muchas situaciones no ayuda a decidir qué tan bien se comportan para un caso específico [Lawler 85]. En particular, debido a que bajo este método se describe el comportamiento de los algoritmos en el límite del tamaño del problema, dicha caracterización podría incluir sólo tamaños de casos que están fuera del contexto de la aplicación [Hooker 1994, Moret 2003, Selman 1998].

Por otro lado, el análisis experimental sí es adecuado para casos específicos y es ampliamente reportado en publicaciones científicas. Sin embargo, se utiliza muy informalmente, no reúne estándares mínimos de reproducción y sólo ocasionalmente se reportan resultados analizados usando métodos estadísticos rigurosos [Hoos 1998, Hooker 1994, Hooker 1996].

Por lo anterior, la investigación sobre la metodología de experimentación computacional está creciendo rápidamente, y tiene como objetivo promover que los experimentos sean importantes, correctos, replicables y que produzcan conocimiento [Mc Geoch 2002].

2.2 TRABAJOS RELACIONADOS

Esta sección inicia con una reseña de la selección de algoritmos. Posteriormente se abordan cuatro métodos empíricos para caracterizar y seleccionar algoritmos. El método más reciente es el orientado a la construcción de modelos de desempeño algorítmico que incorporan características del problema que afectan al desempeño de una manera crítica, y es revisado con mayor profundidad en la sección 2.2.5.

2.2.1 Sinopsis

En diferentes trabajos de investigación se han mostrado las bondades y limitaciones de los métodos heurísticos para resolver formulaciones matemáticas de problemas de optimización complejos. Sin embargo, para resolver problemas prácticos no existen estudios que resuelvan el cuestionamiento de cuándo utilizar un método u otro. En este sentido, es importante predecir el comportamiento de los algoritmos heurísticos en cuanto a la calidad y tiempo de cómputo empleado en la solución. Sin embargo, la falta de metodologías matemáticas para caracterizar el desempeño de estos algoritmos, ha dificultado la evaluación y selección de los mismos [Papadimitriou 1998, Reeves 1993].

El problema de seleccionar el mejor algoritmo para un caso particular, es un problema abierto de investigación [Anderson 2003]. Sobre todo para algoritmos heurísticos que resuelven problemas NP-duros. Esta dificultad se debe a la influencia desconocida de múltiples factores que afectan el desempeño de los algoritmos: el tamaño del problema, la estructura del espacio de solución definida por las restricciones del problema, la estructura del espacio de búsqueda trazado por el algoritmo de solución, aspectos de implementación y ambiente computacional.

Los primeros trabajos de caracterización del desempeño de los algoritmos, se orientaron al análisis de complejidad de algoritmos exactos, como el método Simplex. Por lo general, se modelaban mediante una función matemática el desempeño optimista, pesimista y promedio, de los algoritmos. Con el surgimiento de los métodos heurísticos, que incorporan el azar en el proceso de búsqueda de la solución, ya no fue posible aplicar los anteriores criterios de complejidad [Moret 2003].

El nuevo criterio fue estimar la superioridad de un algoritmo sobre los demás en base a experimentos realizados sobre un conjunto reducido de casos, y se seleccionaba el que consistentemente obtenía mejores resultados. En sus

inicios, el análisis experimental fue muy limitado y sin el rigor suficiente para reproducir sus resultados. Los primeros trabajos serios dieron evidencia de que no hay garantía de que un algoritmo heurístico sea el mejor para todos los casos de un problema. Actualmente es muy conocido que, en situaciones de la vida real, ningún algoritmo es superior en todas las circunstancias [Bertsekas 1991].

Una reciente investigación teórica ha sugerido que los casos de un problema pueden agruparse en clases, y que existe un algoritmo para cada clase que resuelve los casos de esa clase de manera más eficiente [Wolpert 1997]. De lo anterior, surge la pregunta ¿de qué manera identificar las regiones de dominación de los algoritmos, de forma que puedan ser utilizadas para predecir la región a la que pertenece un caso dado y el algoritmo que mejor lo resuelva? De manera simplificada, se puede decir que si se contesta dicha pregunta se estará resolviendo el problema de la selección de algoritmos heurísticos. Sin embargo, éste no es un problema trivial.

Uno de los enfoques más abordado para identificar las regiones de dominación de los algoritmos heurísticos consiste en el desarrollo de modelos matemáticos que relacionan el desempeño con el tamaño del problema. Sin embargo, el desempeño se ve afectado por muchos factores, entre ellos las propiedades del caso que se resuelve [McGeoch, 2002]. Pocos investigadores han tratado de identificar las regiones de dominación considerando más de una característica del problema. Además, algunas investigaciones no identifican las que son críticas para el desempeño, y otras no las incorporan explícitamente en su modelo de desempeño.

2.2.2 Método Clásico

El método más común para comparar experimentalmente algoritmos, consiste en el uso complementario de un conjunto de pruebas estadísticas sencillas muy conocidas: la prueba de signo, la prueba de Wilcoxon y la prueba de Friedman,

entre otras. Estas pruebas se basan en la determinación de las diferencias en el desempeño promedio observado experimentalmente: si las diferencias entre los algoritmos son significativas estadísticamente, el algoritmo con mejores resultados se considera como superior.

Las pruebas estadísticas antes descritas son abordadas ampliamente por Golden, Stewart y Reeves en [Lawler 1985, Reeves 1993]. En esos trabajos se señala que con estas pruebas se compara la eficiencia promedio sin considerar su dispersión, es decir, si en promedio un algoritmo es mejor que los otros, en la mayoría de los casos estudiados, entonces se considera que es superior.

2.2.3 Muestreo del Desempeño en Línea

Los métodos de muestreo en línea se basan en la ejecución, por un periodo corto, de un algoritmo sobre un caso en particular, con el fin de obtener una muestra representativa de ciertos datos estadísticos de interés, que permitan predecir el comportamiento general del algoritmo sobre dicho caso. Donald Knuth es el precursor de esta técnica.

El método propuesto por Knuth, consiste en muestrear el árbol de búsqueda trazado por algoritmos de Retroceso Cronológico [Knuth 1975]. Una muestra se toma mediante la exploración de un camino desde la raíz del árbol hasta una hoja. El camino es elegido de manera aleatoria. Los estadísticos de interés para Knuth son el costo de procesar un nodo y el número de hijos del nodo. Si el número de hijos del nivel i es d_i entonces: $(1) + (d_1) + (d_1 \times d_2) + ... + (d_1 \times ... \times d_n)$ es un estimado del número de nodos en el árbol. Si el costo de procesamiento de cada nodo en el nivel i es c_i, entonces $c_1 + c_2\,(d_1) + c_3\,(d_1 \times d_2) + ... + c_{n+1}\,(d_1 \times ... \times d_n)$ es un estimado del costo de procesar todos los nodos del árbol.

A partir del trabajo de Knuth, el cual explora una sola ruta de la raíz hasta una hoja, otros investigadores han propuesto métodos más precisos. El método

propuesto por Purdom en [Purdom 1978] permite la exploración de más de una ruta, y recibe como entrada el número de hijos que se explorarán de cada nodo visitado. El método propuesto en [Sillito 2000] para algoritmos de Retroceso con Propagación de Restricciones, permite la exploración completa del árbol hasta cierto nivel de profundidad, y a partir de allí el número de nodos se estima aplicando un muestreo similar al de Purdom. En [Lobjois 1998] se aplica esta técnica a algoritmos de Ramificación y Acotamiento, y en [Allen 1996], a algoritmos de Retroceso con Propagación de Restricciones y Búsqueda Tabú.

Los resultados reportados en los trabajos antes descritos, fueron satisfactorios para algoritmos exactos (Retroceso y Ramificación y Acotamiento). Sin embargo para algoritmos heurísticos (Búsqueda Tabu) fueron desalentadores.

2.2.4 Modelos Basados en el Tamaño del Problema

Los modelos matemáticos que relacionan el desempeño al tamaño del problema, permiten establecer rangos en los cuales es preferible un algoritmo sobre otro. Sin embargo, es muy conocido que el desempeño también es afectado por otros factores. Algunos trabajos interesantes de este tipo se describen a continuación.

Gent y Walsh trabajaron con el problema de la satisfacción de expresiones lógicas (SAT, satisfability). Hacen un estudio empírico del algoritmo GSAT, el cual es un algoritmo de aproximación para SAT, y aplican análisis de regresión para modelar el crecimiento del costo de obtener la solución con el tamaño de problema [Gent 1993, Gent 1997].

Lagoudakis y Littam, en [Lagoudakis 2001], abordan la selección de algoritmos como un problema de minimización del tiempo total de ejecución, el cual es resuelto con un algoritmo de Aprendizaje Reforzado (RL, Reinforced Learning). Se enfocaron a dos problemas clásicos: selección y ordenamiento; y

determinaron, mediante entrenamiento con un cierto número de casos, el mejor algoritmo para cada conjunto de problemas agrupados por su tamaño.

En [Cruz 1999, Pérez 2004a], Pérez y Cruz presentan un método estadístico para construir modelos de desempeño de los algoritmos usando funciones polinomiales que relacionan el desempeño con el tamaño del problema. Este método genera primero una muestra representativa del desempeño de los algoritmos, entonces usando análisis de regresión determina las funciones de desempeño, las cuales son finalmente incorporadas en un mecanismo de selección de algoritmos. El algoritmo seleccionado es aquél cuyo desempeño, predicho por las funciones polinomiales, satisface mejor los requerimientos del usuario.

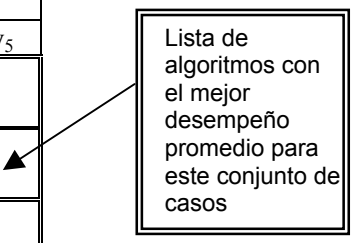
2.2.5 Modelos Basados en Características del Problema

Una técnica reciente para modelar el desempeño algorítmico, trata de incorporar más de una característica de complejidad del problema con la finalidad de obtener una mejor representación del mismo. El objetivo es incrementar la precisión del proceso de selección de algoritmos. Los trabajos descritos enseguida siguen esta técnica.

Los trabajos de Tsang y Frost se basan en la construcción de una tabla de asociación [Tsang 1995, Frost 1994]. En particular, Tsang proyectó el desempeño de diferentes algoritmos sobre un amplio rango de especificaciones de casos del Problema de Satisfacción de Restricciones (CSP, Constraint Satisfaction Problem). Los casos son caracterizados por los parámetros <w, x, y, z>, los cuales describen de manera simplificada la estructura del problema. El proceso consiste en generar conjuntos de casos de manera que, los que pertenecen al mismo conjunto, tengan valores similares de sus parámetros característicos. Con los resultados del desempeño se construye un mapa como el de la Tabla 2.1, el cual muestra, para cada conjunto de casos, el algoritmo de mejor desempeño promedio.

Tabla 2.1 Formato de un mapa de desempeño algorítmico, *w, x, y,* y *z* son parámetros del problema

w = valor fijo		*x* = valor fijo				
y/z		Valores de y				
		y_1	y_2	y_3	y_4	y_5
Valores de z	z_1					
	z_2					
	z_3					
	z_4					
	z_5					



Frost encuentra que el desempeño de algoritmos que resuelven casos de CSP, puede ser aproximado por dos familias estándares de funciones de distribución de probabilidad continuas [Frost 1997]. Los casos resolubles pueden ser modelados por la distribución de Weibull, y los no resolubles, por la distribución lognormal.

Hoos y Stuzle presentan un trabajo similar al de Frost. Ellos encontraron que el desempeño de algoritmos que resuelven casos de SAT, puede ser caracterizado por una distribución exponencial, [Hoos 1998, Hoos 2000]. La distribución del tiempo de ejecución se determina mediante la ejecución de un algoritmo k veces, sobre un conjunto de casos de la misma familia, con un tiempo de corte muy alto, y almacenando para cada corrida exitosa el tiempo de ejecución requerido para encontrar la solución. La distribución empírica de tiempo de ejecución es la distribución acumulada asociada con estas observaciones, y permite proyectar el tiempo de ejecución t, dado por el usuario, a la probabilidad de encontrar una solución en dicho tiempo. Una familia es un conjunto de casos con el mismo valor de los parámetros considerados críticos para el desempeño.

Borghetti utilizó gráficas de utilidad para correlacionar las características de complejidad de los casos con el desempeño algorítmico [Borghetti 1996]. El problema tratado fue razonamiento sobre una base de conocimiento Bayesiana

(BKB, Bayesian Knowledge Base), el cual fue resuelto con un Algoritmo Genético (GA, Genetic Algorithm) y un algoritmo de Búsqueda Primero el Mejor (BFS, Best First Search). Para el problema BKB, dos clases de características críticas fueron identificadas: topológicas y probabilísticas. Ejemplos de características topológicas contables son el número de nodos, número de arcos y número de variables aleatorias. Una desventaja importante de este método, es que no considera el efecto combinado de todas las características identificadas.

Minton propuso un método que permite especializar algoritmos genéticos para aplicaciones particulares [Minton 1996]. La entrada consiste de una descripción simplificada del problema y un conjunto de casos de entrenamiento, los cuales guían la búsqueda a través del espacio de diseños, constituido por heurísticas que compiten por ser incorporadas en el algoritmo genético. La salida es un programa ajustado al problema y a la distribución de los casos.

Fink desarrolló una técnica de selección de algoritmos para problemas de decisión, en la cual para un caso nuevo, estima la ganancia de los algoritmos a partir del análisis estadístico de un grupo de casos afines tomados de un registro histórico de casos resueltos [Fink 1998]. Aunque la estimación puede ser enriquecida con nuevas experiencias, su efectividad depende de la habilidad del usuario para definir una taxonomía de casos del problema. Además, las características del problema, dadas implícitamente en la taxonomía de casos, pudieran no tener ninguna relación con el desempeño de los algoritmos.

El equipo de investigación que desarrolla el proyecto METAL, propuso un método para seleccionar el algoritmo de clasificación más adecuado para un conjunto de casos relacionados [Soares 2000]. Ellos identifican el conjunto de casos antiguos que muestra características más similares a las del nuevo conjunto. El desempeño algorítmico de los casos viejos es conocido y es utilizado para predecir el mejor algoritmo para el nuevo conjunto de casos. La similitud entre conjuntos de casos se obtiene considerando tres tipos de características: generales, estadísticas y derivadas de la teoría de la información. Debido a que no proponen un modelo para

relacionar las características con el desempeño, la identificación de casos similares se repite con cada nuevo conjunto de datos, así que el tiempo de procesamiento para la selección de algoritmos puede ser elevado.

Rice introdujo el concepto de poli-algoritmo [Rice 1968] en el contexto de software numérico paralelo. Se propone el uso de funciones que permitan seleccionar, de entre un conjunto de algoritmos, el mejor para resolver una situación dada. Después del trabajo de Rice, otros investigadores han formulado diferentes funciones, por ejemplo las presentadas en [Li 1997, Brewer 1995]. La mayoría de las funciones propuestas son simples reglas heurísticas sobre las características estructurales de los parámetros del caso que se resuelve, o sobre el entorno computacional. La definición de las reglas requiere del experto humano.

2.2.6 Análisis Comparativo

La Tabla 2.2 presenta los trabajos más importantes que incorporan características del problema en su modelo de desempeño. La columna 3 indica si las características de complejidad del problema son modeladas formalmente. La columna 4 muestra si las características de los casos son utilizadas para agruparlos. La columna 5 se usa para indicar si la relación entre el desempeño de algoritmos y las características de complejidad se aprende en un modelo de desempeño a partir de experiencia pasada.

En la Tabla 2.2 se puede observar que ningún trabajo incluye todos los aspectos que en este trabajo se consideran necesarios para caracterizar el desempeño algorítmico. El uso de medidas de la complejidad de los casos es un área emergente y prometedora para la identificación de regiones de dominación de los algoritmos. Los trabajos de Borghetti y del grupo METAL son contribuciones importantes para esta área. Sin embargo, el primero no correlaciona las características de complejidad identificadas, dificultando la selección de algoritmos; y el segundo no es aplicable a la solución de un solo caso.

Tabla 2.2. Trabajos relacionados con caracterización del desempeño

Trabajo	Definición de características del problema	Modelado de características de complejidad	Agrupación de casos	Modelado del desempeño mediante aprendizaje
Fink	✓		✓ (informal)	
Borgethi	✓	✓		
METAL	✓	✓		
Propuesta de esta tesis	✓	✓	✓ (formal)	✓

En contraste con los trabajos previos, nuestra propuesta es la única que considera los cuatro aspectos de la Tabla 2.2, los cuales son relevantes para el problema de la selección de algoritmos.

De la revisión de trabajos relacionados, se concluye que aun cuando los métodos heurísticos han sido ampliamente estudiados, actualmente no se dispone de mecanismos de selección que puedan ser utilizados en forma general para sugerir los métodos más adecuados para un problema de optimización dado, y en forma particular para casos específicos.

Desarrollar modelos de desempeño algorítmico que integren todos los factores de influencia, no es una tarea fácil. Existe mucha incertidumbre en torno a ellos: cuáles son críticos, cómo afectan, y de qué manera están interrelacionados. Las técnicas de aprendizaje automático han mostrado su potencial en el manejo de conocimiento relacionado. Inspirados en estos resultados, en esta tesis se propone un procedimiento sistemático, basado en aprendizaje automático y la estadística, para crear modelos que incorporan y relacionan características del problema que son críticas para el desempeño de los algoritmos, esto con la finalidad de seleccionar el mejor para resolver un caso dado. En el siguiente capítulo se describe la metodología propuesta.

Capítulo 3

METODOLOGÍA PARA CARACTERIZACIÓN Y SELECCIÓN DE ALGORITMOS HEURÍSTICOS

Para la solución de problemas del tipo NP-duro, muchos investigadores coinciden en afirmar que los métodos heurísticos son una buena alternativa para casos reales [Johnson 2002, McGeoch 2002, Ahuja 2003]. Como resultado, muchos algoritmos heurísticos han sido propuestos para su solución. Sin embargo, no se ha formalizado ningún método adecuado para seleccionar el algoritmo más apropiado para resolverlos. De aquí nuestro interés en estudiar la caracterización de algoritmos heurísticos y su aplicación a la selección del algoritmo más adecuado para un caso dado de un problema.

En este capítulo se presenta una metodología que, basada en el aprendizaje automático, determina para cada algoritmo un patrón de agrupamiento de los casos que ha resuelto mejor. Para un nuevo caso, la selección del algoritmo que lo resuelve mejor, se realiza determinando a qué grupo es más afín y se selecciona el algoritmo adecuado a ese grupo. El capítulo está organizado como sigue: la sección 3.1 describe el método general de solución y las secciones 3.2 a 3.4 describen las fases que componen la metodología propuesta.

3.1 MÉTODO DE SOLUCIÓN

Esta sección describe el método general de solución que se siguió en el desarrollo de la metodología propuesta para la selección de algoritmos.

3.1.1 Estrategia General para la Selección de Algoritmos

Un factor que incide notablemente en el desempeño de los algoritmos, es el conjunto de propiedades asociadas a los parámetros que definen un caso. Debido a que se espera que ningún algoritmo domine a los demás en todos los escenarios posibles, una meta ambiciosa del análisis de algoritmos es descubrir las relaciones funcionales entre parámetros de casos y desempeño de algoritmos [McGeoch 2002]. Sin embargo, el uso de parámetros es muy limitado ya que no muestran sus interrelaciones. En este sentido es importante captar las propiedades asociadas a los parámetros.

La metodología propuesta se basa en el uso de índices de complejidad y el uso de técnicas estadísticas y de aprendizaje automático (ver Fig. 3.1). Los índices permiten cuantificar las interrelaciones entre los parámetros de un caso.

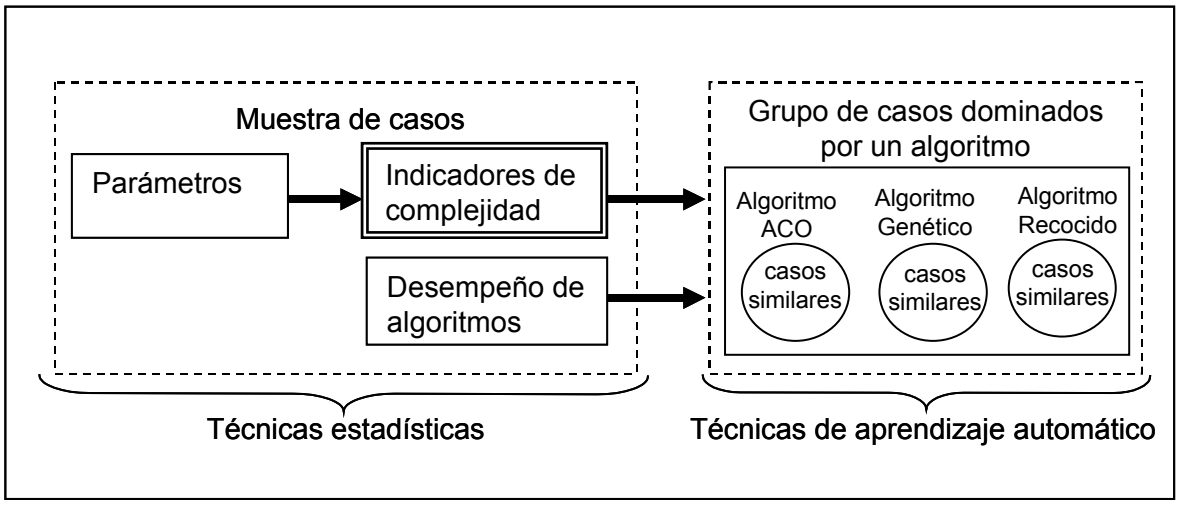


Figura 3.1. Estrategia general para la selección de algoritmos

3.1.2 Fases de la Metodología y el Sistema de Selección

La metodología propuesta está integrada por tres fases (ver Fig. 3.2). El objetivo de la Fase I es construir modelos de predicción del desempeño a partir de experiencia pasada, es decir, la relación entre el desempeño de los algoritmos y los indicadores de complejidad se aprende de una muestra inicial de casos resueltos con varios algoritmos. En la Fase II la relación aprendida se usa para predecir el mejor algoritmo para un nuevo caso dado. En la Fase III la capacidad de estimación de los predictores se mejora mediante la incorporación de nuevos casos resueltos.

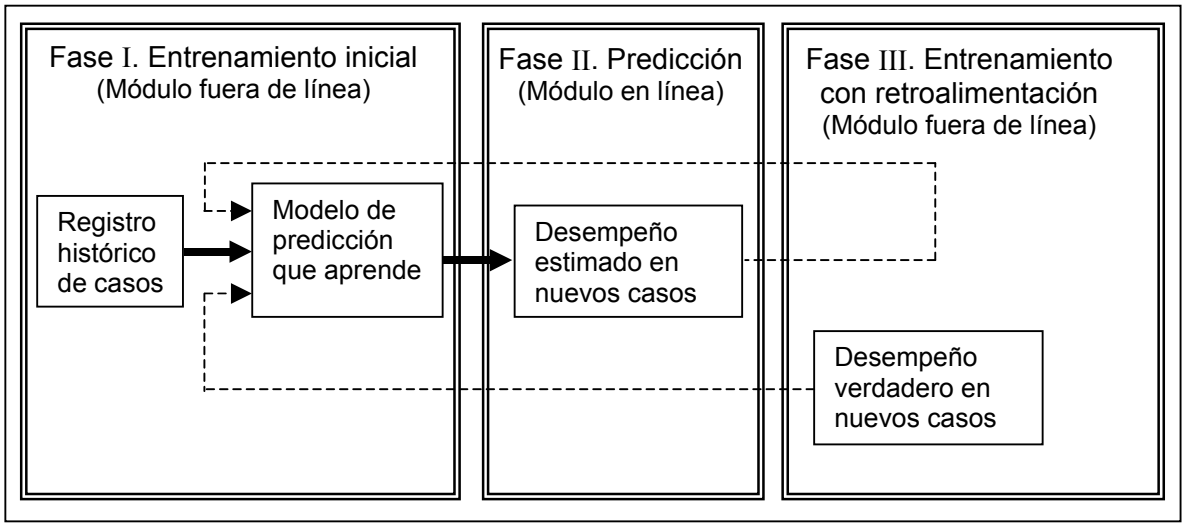


Figura 3.2. Fases de la metodología de selección de algoritmos

Con la implementación de la metodología obtenemos sistemas de software para la selección de algoritmos. Estos sistemas constan de tres módulos computacionales (ver Fig. 3.2). El módulo de predicción es el único que interactúa con el usuario del sistema de selección. Los módulos de entrenamiento se ejecutan fuera de línea, es decir, modifican el modelo de predicción desconectados del módulo de pronóstico. Por lo tanto, el tiempo de cómputo para efectuar la selección no es afectado por el tiempo de entrenamiento.

En las siguientes secciones se detalla cada una de las fases que integran la metodología de selección, y se ejemplifican con el problema de *Bin-packing*.

3.2 FASE DE ENTRENAMIENTO INICIAL

Los pasos de esta fase se muestran en la Fig. 3.3. En el paso 1 (Modelado de índices) se derivan expresiones indicadoras de la complejidad. El paso 2 (Muestreo estadístico) genera un conjunto de casos representativos del problema. En el paso 3 (Medición de índices), para cada caso se obtienen los valores de los indicadores a partir de los parámetros del caso. En el paso 4 (Solución de casos) los casos de la muestra se resuelven usando un conjunto de algoritmos heurísticos, y para cada caso se determinan los mejores algoritmos. El paso 5 (Agrupación de casos) integra para cada algoritmo un grupo de casos que han sido resueltos mejor por ese algoritmo. Finalmente, en el paso 6 (Construcción del selector) se aprende la agrupación, y se expresa en clasificadores formales, los cuales son predictores que modelan la relación entre indicadores y desempeño.

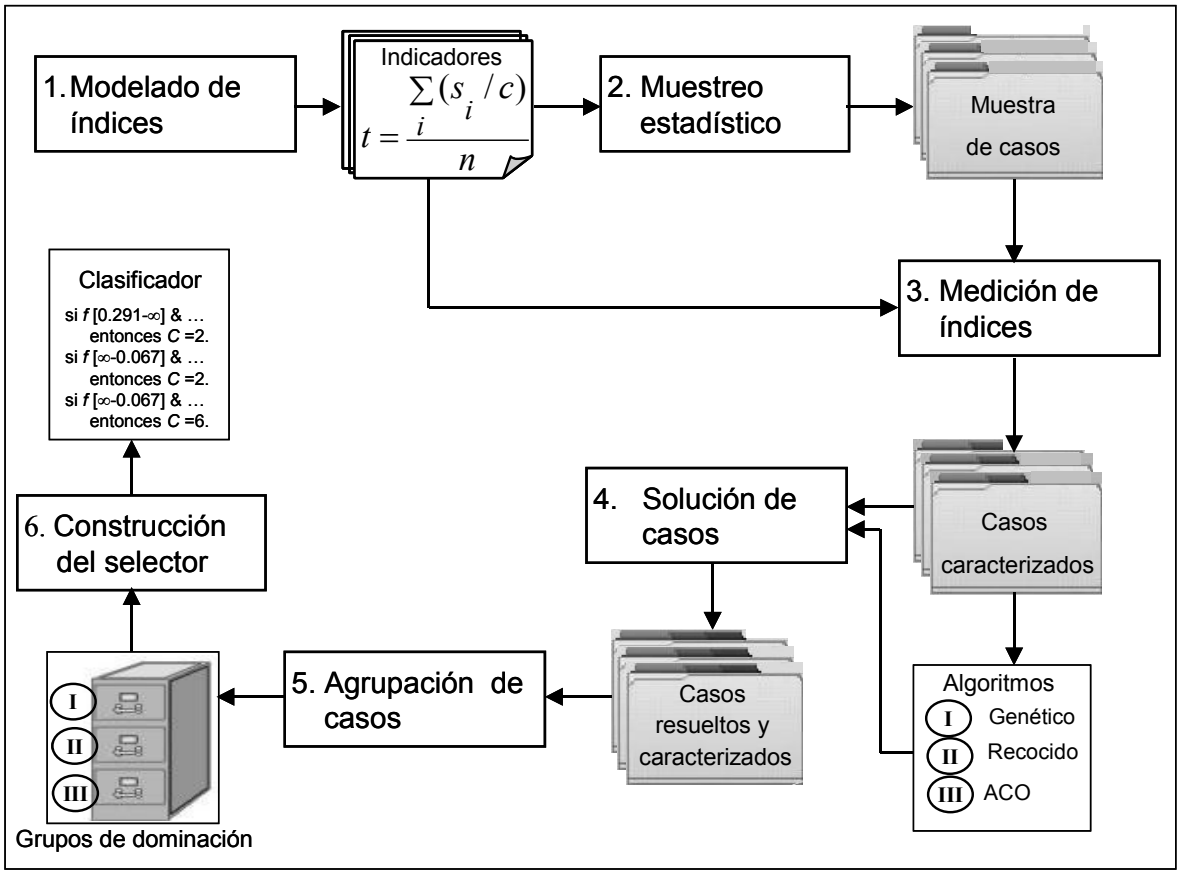


Figura 3.3. Pasos de la fase de entrenamiento inicial

3.2.1 Modelado de Índices de Complejidad del Problema

Se desarrolló analíticamente un conjunto de índices de complejidad del problema de *Bin-packing*, de los cuales cinco fueron los más relevantes (Tabla 3.1). Tres de ellos fueron inspirados en los trabajos reportados en [Borghetti 1996, Soares 2000, Coffman 2002]. Los otros dos se originaron en esta tesis: el índice de capacidad ocupada por un objeto y el índice de uso de contenedores.

Índice del tamaño del caso. El índice p en la expresión 3.1 expresa la relación entre el tamaño de un caso y el tamaño máximo resuelto. El tamaño de un caso es el número de objetos de un caso n, y el tamaño máximo resuelto es $nmax$. El valor de $nmax$ se asignó a 1000, el cual corresponde con el número de objetos del caso más grande resuelto reportado en la literatura especializada.

Índice de capacidad ocupada por un objeto promedio. El índice t en la expresión 3.2 expresa una relación entre el tamaño promedio de los objetos de un caso y el tamaño del contenedor. El tamaño del objeto i es s_i y el tamaño del contenedor es c.

Índice de dispersión del tamaño del objeto. El índice d en la expresión 3.3 expresa el grado de dispersión de los valores de los tamaños de los objetos de un caso.

Índice de factores. El índice f en la expresión 3.4 expresa la proporción de objetos cuyos tamaños son factores de la capacidad del contenedor; en donde se entiende que un objeto i es un factor cuando la capacidad del contenedor c es múltiplo de su correspondiente tamaño de objeto s_i. En general, los casos con muchos factores se consideran fáciles de resolver.

Índice del uso de contenedores. El índice b se muestra en la expresión 3.5. Este índice cuantifica la proporción de la suma de los tamaños de los objetos que pueden asignarse en un contenedor de capacidad $\boldsymbol{c}$, y expresa la cantidad ideal de

contenedores requeridos en la solución. Cuando este cociente toma un valor mayor o igual a 1 significa que todos los objetos caben en un contenedor y se le asigna un valor de 1 a ese índice.

Tabla 3.1 Tabla de índices de complejidad

Expresión	Descripción	Identificador de expresión
$p = \frac{n}{nmax}$	p es el índice del tamaño del caso, donde n = número de objetos, $nmax$ = el tamaño máximo solucionado (1000).	(3.1)
$t = \frac{\sum_{i=1}^{n} s_i / n}{c}$	t es el índice de capacidad ocupada por un objeto promedio, donde s_i = tamaño del objeto i, c = capacidad del contenedor.	(3.2)
$d = \sqrt{\frac{\sum_{i=1}^{n}\left[t - \left(\frac{s_i}{c}\right)\right]^2}{n}}$	El índice de dispersión d expresa el grado de la dispersión del cociente del tamaño de los objetos entre el tamaño del contenedor.	(3.3)
$f = \frac{\sum_{i=1}^{n} factor(c, s_i)}{n}$	El índice de factores f expresa la proporción de objetos cuyo tamaño s_i es factor de la capacidad del contenedor, donde $factor(c, s_i) = \begin{cases} 1 & \text{si } (c \bmod s_i) = 0 \\ 0 & \text{en caso contrario} \end{cases}$	(3.4)
$b = \begin{cases} 1 & \text{si } c \geq \sum_{i=1}^{n} s_i \\ \frac{c}{\sum_{i=1}^{n} s_i} & \text{en caso contrario} \end{cases}$	El uso de contenedor b expresa la proporción del tamaño total que se puede asignar en un contenedor de capacidad c.	(3.5)

El siguiente ejemplo ilustra el proceso de obtención de indicadores a partir de los valores de los parámetros. Considérese el siguiente caso de *Bin-packing*:

n = 48, c = 230 y s_i = (ver Tabla 3.2).

Tabla 3.2 Lista de tamaños de los objetos de un caso

I	s_i	i	s_i	i	s_i	i	s_i	i	s_i	i	s_i
1	17	9	16	17	16	25	16	33	17	41	15
2	16	10	15	18	17	26	15	34	15	42	17
3	17	11	16	19	17	27	16	35	15	43	15
4	17	12	15	20	17	28	16	36	15	44	17
5	17	13	15	21	17	29	16	37	17	45	16
6	17	14	17	22	15	30	17	38	17	46	15
7	17	15	15	23	15	31	17	39	15	47	16
8	16	16	17	24	16	32	16	40	15	48	16

Los valores de los parámetros se sustituyen en las expresiones indicadoras de la complejidad del caso (expresiones 3.1 a 3.5). Los resultados de aplicar las fórmulas a los parámetros del ejemplo son

p = 0.0480, t = 0.06969, d = 0.0036, f = 0 y b = 0.2979.

3.2.2 Muestreo Estadístico de Casos del Problema

Para la generación de casos representativos de un problema, se desarrolló un método en el cual se combinaron técnicas de muestreo de encuestas y reducción de la varianza.

La formación de estratos es una técnica que permite reducir la variabilidad de los resultados e incrementar la representatividad de la muestra. Esto puede ayudar en el aseguramiento de consistencia, especialmente en el manejo de datos agrupados [Micheals 2001]. Por otra parte, el muestreo de encuestas permite obtener estimados de la proporción de individuos que presentan características específicas [Thompson 2002].

3.2.2.1 Estrategia General para la Generación de Valores Aleatorios

Para realizar un análisis experimental del desempeño de algoritmos, se requiere contar con una muestra de casos del problema que se resuelve. En el método tradicional (ver Fig. 3.4), los casos de una muestra se obtienen generando al azar los valores de los parámetros dentro de rangos establecidos. La solución de los casos de esta muestra permite analizar el desempeño de algoritmos.

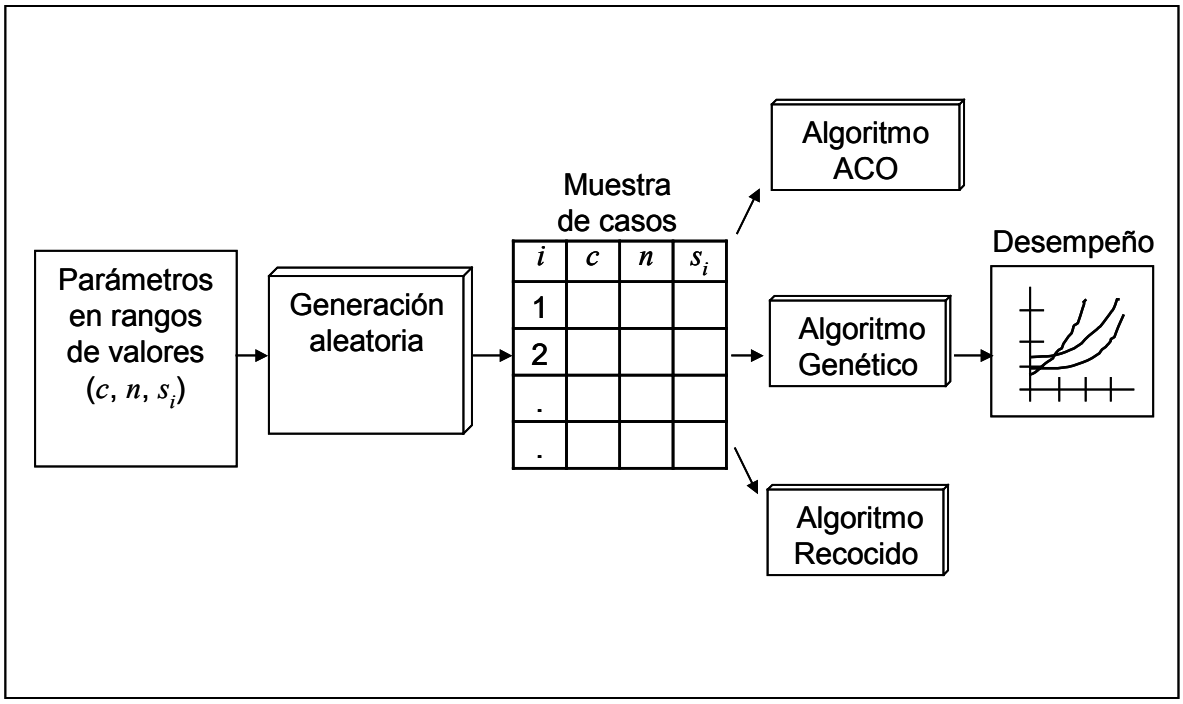


Figura 3.4 Método tradicional para generación de casos

El método tradicional de generación de casos no es adecuado para el objeto de estudio de esta tesis: por un lado, no conduce a una buena representación de la población, ya que los valores de los indicadores de complejidad del problema tienden a concentrarse en algunos puntos; y por otro lado, no permite identificar para cada algoritmo el conjunto de casos que mejor resuelve. En este sentido, los parámetros están muy limitados ya que no muestran sus interrelaciones, a diferencia de los indicadores de complejidad, los cuales enriquecen el conocimiento de la estructura de cada uno de los casos.

La estrategia que se propone aquí (Fig. 3.5), es que la generación al azar sea de los indicadores, y que sus valores sean obtenidos a partir de las especificaciones dadas en estratos y rangos. Aun cuando se obtienen estos, es necesario obtener los parámetros a partir de los valores de los indicadores. Esto nos permite contar con una muestra de casos resuelta con los algoritmos disponibles. Los resultados del desempeño de los algoritmos y los valores de complejidad del problema posibilitan la agrupación de casos dominados por un mismo algoritmo. Este resultado es esencial en la selección de algoritmos.

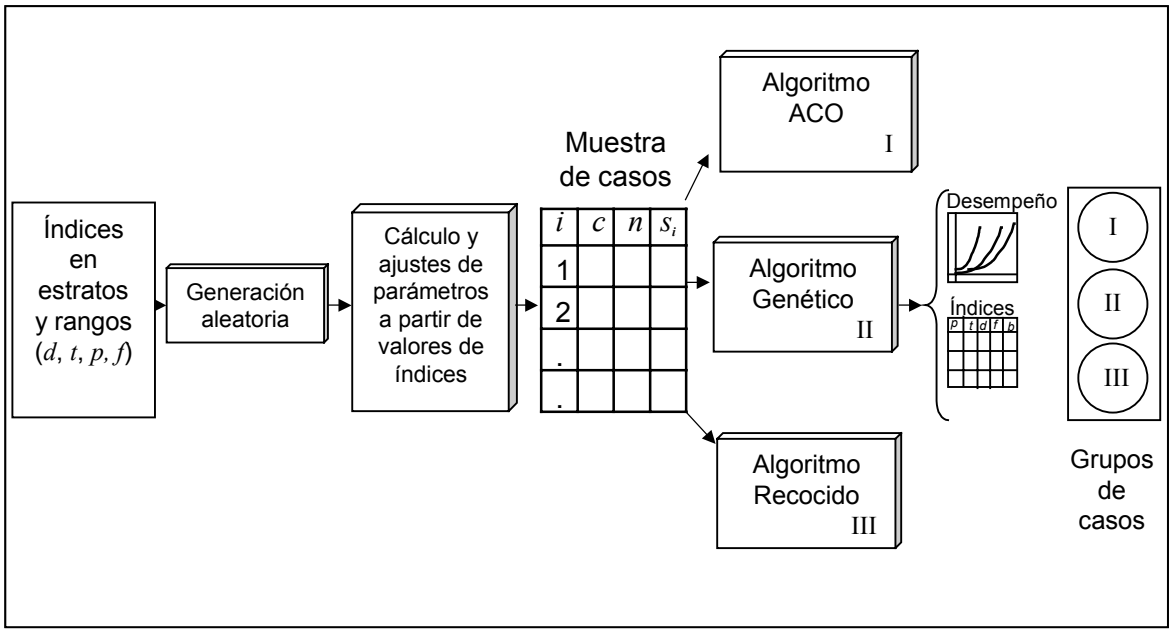


Figura 3.5 Enfoque propuesto para generación de casos

3.2.2.2 Método para Generar Casos Representativos de un Problema

El método propuesto consta de cinco pasos:

Paso 1. Desarrollo de mecanismos para obtener parámetros a partir de indicadores. Para generar un nuevo caso se obtienen los valores de sus parámetros a partir de los valores de sus características generadas en forma aleatoria. Para este proceso se desarrollaron cinco expresiones (3.6, 3.7, 3.8, 3.9 y, 3.10 de la Tabla 3.3) y un algoritmo.

Tabla 3.3 Expresiones para obtener parámetros a partir de indicadores

Expresión	Descripción	Identificador de expresión
$n = p \cdot nmax$	n = número de objetos	(3.6)
$\bar{s} = c \cdot t$	$\bar{s}$ = tamaño promedio de los objetos	(3.7)
$smin = \begin{cases} 1 & \text{si } \bar{s} - (\bar{s} \cdot 2d) < 1 \\ \bar{s} - (\bar{s} \cdot 2d) & \text{en caso contrario} \end{cases}$	$smin$ = tamaño mínimo de los objetos	(3.8)
$smax = \begin{cases} c & \text{si } \bar{s} + (\bar{s} \cdot 2d) > c \\ \bar{s} + (\bar{s} \cdot 2d) & \text{en caso contrario} \end{cases}$	$smax$ = tamaño máximo de los objetos	(3.9)
$s_i = \text{random}(smin, smax)$	s_i = tamaño del objeto i generado al azar en el intervalo entre $smin$ y $smax$, donde random es una función del lenguaje de cómputo	(3.10)

Capacidad del contenedor (c). Como se vió anteriormente, los indicadores de complejidad son valores normalizados, o sea en el rango de cero a uno. Para desnormalizar a la mayoría de estos valores, fue necesario que el valor de al menos uno de los parámetros se definiera de manera absoluta. En la literatura se identificó que los problemas de interés práctico utilizan un tamaño de contenedor que varía entre 100 y 100,000 unidades, por lo que se eligió este rango para asignar valores aleatorios al parámetro c.

Número de objetos (n). Para obtener el número de objetos n se definió la expresión 3.6. En esta expresión n es proporcional a p y $nmax$.

Tamaño del objeto i (s_i). Para obtener el tamaño s_i de cada objeto i en S, se derivaron cuatro expresiones: con la expresión 3.7 se obtiene el tamaño promedio de los objetos $\bar{s}$ y con las expresiones 3.8 y 3.9 se obtienen el tamaño mínimo para un objeto $smin$ y el tamaño máximo $smax$. Para propósitos de estudio se inicializó S utilizando una distribución normal con media $\bar{s}$ y desviación estándar d; los límites se ubicaron en $\pm$ 2d. Para obtener cada s_i se generó un número aleatorio entre $smin$ y $smax$ (ver expresión 3.10). La distribución inicial de S se modificó con el algoritmo AJUSTAR-FACTORES para incluir el índice f. No se realizó ajuste para b ya que se supuso que éste pudiera quedar representado por p y t. Se puede demostrar fácilmente que b varía inversamente con p y t.

Algoritmo AJUSTAR-FACTORES. Este algoritmo modifica el tamaño de los objetos necesarios para que S contenga el número de factores especificado en la proporción f. Primero se obtiene el conjunto F de divisores exactos de la capacidad del contenedor que se encuentran en el rango válido para S. Posteriormente, y utilizando F, se divide el conjunto S en dos subconjuntos: S_f contiene elementos que son factores de c, mientras que S_{nf} no tiene factores. Enseguida se completa el número de factores requeridos por f, convirtiendo elementos de S_{nf} a factores y moviéndolos a S_f. Este ajuste altera indirectamente el valor del índice t, el cual se restablece modificando algunos tamaños de S_{nf}, de manera que la suma de los tamaños de $S = S_f \cup S_{nf}$ recupere su valor inicial.

El siguiente ejemplo ilustra el proceso de obtención de parámetros a partir de indicadores. Dados los siguientes índices de un caso: p = 0.007, b = 0.1754, t = 0.8146, f = 0.4286 y d = 0.1640; y dados también los siguientes valores de referencia: c = 2467 y $nmax$ = 1000; los valores se sustituyen en las expresiones 3.6 a 3.10, y se obtiene lo siguiente: n = 7, $\bar{s}$ = 2009.5713, $smin$ = 1, $smax$ = 2467 y S = {1343, 1437, 1172, 1448, 1378, 1666, 1922}. Aplicando el algoritmo AJUSTAR-

FACTORES se modifica el tamaño de algunos objetos quedando S = {752, 846, 580, 2467, 787, 2467, 2467}. Con estos cambios, los parámetros del nuevo caso son, n = 7, c = 2467 y S = {752, 846, 580, 2467, 787, 2467, 2467}.

Paso 2. Formación de estratos. Con una muestra más pequeña que la requerida por muestro aleatorio simple, es posible reducir el error de las estimaciones dividiendo la población en grupos no traslapados llamados estratos. La elección de éstos debe asegurar que los grupos pequeños, raros o de interés sean explícitamente incluidos. Con muestreo aleatorio simple se requiere una muestra enorme para asegurar que todos los aspectos sean incluidos [Micheals 2001, Scheaffer 1987, Kalton 1983]. Para la selección de algoritmos se requiere que la muestra de casos sea representativa de los indicadores de complejidad. Para ello, primero la población se definió con cinco dimensiones: c, p, t, d y f. Después, para cada dimensión se especificaron tres rangos de valores (Tabla 3.4). Enseguida, utilizando la expresión 3.11 se calculó el número de estratos str. En la Tabla 3.5 se muestra un ejemplo de los 243 estratos formados y su configuración.

Tabla 3.4 Dimensiones de *Bin-packing* especificadas por categorías

Dimensión	Rango por categorías			Número de categorías (cat_i)
	Pequeño (P)	Mediano (M)	Grande (G)	
c	100 – 1000	1001 – 10000	10001 – 100000	3
p	0 – 0.33	0.34 – 0.66	0.67 – 1	3
t	0 – 0.33	0.34 – 0.66	0.67 – 1	3
d	0 – 0.080	0.0801-0.165	0.166 – 0.25	3
f	0 – 0.33	0.34 – 0.66	0.67 – 1	3

$$str = \prod^{i} cat_i \qquad \text{donde} \qquad (3.11)$$

str = número de particiones (o estratos) del espacio de casos

cat_i = número de categorías de la dimensión i

Tabla 3.5 Configuración de estratos de casos de *Bin-packing*

Estrato	Rango y categoría por dimensión				
	c	p	t	d	f
1	100 – 1000 (P)	0 - 0.33 (P)	0 - 0.33 (P)	0 - 0.080 (P)	0 - 0.33 (P)
2	100 – 1000 (P)	0 - 0.33 (P)	0 - 0.33 (P)	0 - 0.080 (P)	0.34 – 0.66 (M)
3	100 – 1000 (P)	0 - 0.33 (P)	0 - 0.33 (P)	0 - 0.080 (P)	0.67 – 1 (G)
: :	: :	: :	: :	: :	: :
243	10001 – 100000 (G)	0.67 – 1 (G)	0.67 – 1 (G)	0.166 - 0.25 (G)	0.67 – 1 (G)

Paso 3. Cálculo del tamaño de la muestra. El muestreo de encuestas propone estrategias, que permiten seleccionar un tamaño de muestra suficiente, para estimar proporciones con una precisión deseada a partir de alguna información sobre las condiciones de la población. En esta tesis se supone que al generar una muestra adecuada para estimar la proporción de casos que son resueltos de la mejor manera con un mismo algoritmo de solución, esa muestra también será adecuada para identificar patrones de desempeño que dan lugar a esas mismas proporciones en la población. Para incrementar el grado de certeza en los resultados, los cálculos se basaron siempre en condiciones del peor-caso, consecuentemente, el tamaño de la muestra es grande. El tamaño adecuado de la muestra para estimar una proporción, se determina por los siguientes factores:

Predominio ($\hat{p}$). Es la proporción estimada de la presencia de la variable de interés en la población (por ejemplo, la proporción en que un contaminante está presente en el agua de un sistema). En esta investigación, es de interés la proporción de casos mejor resueltos por un mismo algoritmo, es decir, la probabilidad de que un caso seleccionado al azar sea resuelto mejor por un algoritmo dado. En [Wild 2000] se demuestra que el tamaño de la muestra más grande se presenta con $\hat{p}$ =1/2.

Margen de error (ε). Establece un límite superior e inferior al valor de la proporción estimada; se espera que el error de la estimación no sea más grande que ε.

Nivel de confianza normalizado (z). Es el valor normalizado del nivel de confianza, el cual indica qué tan seguro se puede estar de la representatividad de la muestra. Un 95 % de nivel de confianza significa que se tiene un 0.95 de probabilidad de que las estimaciones realizadas tengan el margen de error especificado por ε. Los porcentajes más usados son 95% y 99%. Para un nivel de confianza del 99%, una tabla de la distribución normal señala z = 2.58.

El tamaño de muestra requerido para estimar una proporción, puede ser calculado con la expresión 3.12. Utilizando dicha expresión con z = 2.58 (para un nivel de confianza del 99%), $\hat{p}$ = 0.5, *y* ε = 0.03 se obtuvo un tamaño de muestra m = $(2.58)^2(0.5)(0.5)/(0.03)^2$ = 1849. Aplicando un factor de ajuste de 1.2 para obtener una aproximación del peor caso para varias proporciones [Thompson 2002], tenemos que m = (1849)(1.2) = 2219.

$$m = \left\lceil \frac{z^2(\hat{p})(1-\hat{p})}{\varepsilon^2} \right\rceil \quad \text{(3.12)}$$

donde

m = tamaño de la muestra

z = el valor crítico de la distribución normal en el nivel deseado de confianza

$\hat{p}$ = es la fracción estimada de la variable de interés (0.5 cuando es desconocida)

ε = es el margen de error deseado

Paso 4. Cálculo del número de casos por estrato. Se utilizó la expresión (3.13) para calcular el número de casos a generar para cada estrato, obteniéndose *mstr* =

$\lceil 2219 \div 243 \rceil$ = 10. Así, dado que el número de estratos str = 243 y el número de casos por estrato $mstr = 10$, el nuevo tamaño de muestra queda m = 2430.

$$mstr = \left\lceil \frac{m}{str} \right\rceil \qquad mstr = \text{número de casos generados para cada estrato} \qquad (3.13)$$

Paso 5. Generación aleatoria de los casos de cada estrato. Una vez definidos los estratos en que se divide la población y el número de casos por estrato, los valores de los indicadores de cada nuevo caso se generan aleatoriamente utilizando la configuración del estrato al que pertenece (ver tabla 3.5). Los valores de los parámetros se obtienen a partir de los valores de los indicadores usando las expresiones 3.6, 3.7, 3.8, 3.9, 3.10 y el algoritmo AJUSTAR-FACTORES.

3.2.3 Medición de Indicadores de Complejidad de la Muestra

Para cada caso de la muestra, los valores de sus parámetros fueron sustituidos en las expresiones indicadoras de complejidad (expresiones 3.1 a 3.5). La tabla 3.6 presenta los resultados obtenidos con un conjunto pequeño de casos, el cual fue seleccionado de una muestra con 2,430 casos aleatorios.

Table 3.6. Ejemplo de casos aleatorios con sus indicadores de complejidad

Caso	Indicadores de complejidad				
	p	b	t	f	d
e5i6	0.156	0.642	0.010	0.282	0.002
e14i4	0.001	1.000	0.498	0.000	0.000
e16i4	0.150	0.011	0.580	0.013	0.247
e24i5	0.003	0.351	0.951	0.667	0.070
e56i7	0.984	0.025	0.040	0.537	0.003
e111i7	0.390	0.241	0.011	1.000	0.000
e143i2	0.999	0.009	0.110	0.015	0.060
e155i2	0.987	0.001	0.974	0.449	0.030
e180i2	0.071	0.025	0.564	0.099	0.293
e182i6	0.047	0.022	0.988	0.277	0.012

3.2.4 Evaluación de Algoritmos

Los 2430 casos aleatorios fueron resueltos con siete algoritmos heurísticos. En la sección 3.2.4.1 se describen de manera general los algoritmos heurísticos utilizados, y en la sección 3.2.4.2 se explican los criterios aplicados para su evaluación.

3.2.4.1 Algoritmos Heurísticos Evaluados

Algoritmo de Aceptación por Umbral (TA)

El algoritmo de Aceptación por Umbral (Threshold Accepting, TA) es una variante de Recocido Simulado [Pérez 2002]. En este algoritmo para cada $\boldsymbol{x} \in \boldsymbol{X}$ (donde $\boldsymbol{X}$ representa el conjunto de todas las soluciones factibles) se asocia una vecindad $H(\boldsymbol{x}) \subset \boldsymbol{X}$. Así que, dada una solución actual factible $\boldsymbol{x}$, y un parámetro de control $\boldsymbol{T}$ (llamado temperatura) se genera una solución vecina factible $\boldsymbol{y} \in H(\boldsymbol{x})$; si $(z(\boldsymbol{y}) - z(\boldsymbol{x})) < T$, entonces se acepta $\boldsymbol{y}$ como la nueva solución actual, en caso contrario la solución actual permanece sin cambio. El valor de la temperatura T se decrementa cada vez que se alcanza el equilibrio térmico. Éste se prueba cada vez que se forme un conjunto de S soluciones factibles, El valor de T se reduce multiplicándolo repetidamente por un *factor de enfriamiento* $\mu < 1$ hasta que el sistema alcance el congelamiento.

Algoritmo de Optimización de Colonia de Hormigas (ACO)

El Algoritmo de Optimización de Colonia de Hormigas (Ant Colony Optimization, ACO) presentado en [Ducatelle 2001], está inspirado en la habilidad real de las hormigas de encontrar la ruta más corta entre su nido y la fuente de alimento usando una reserva de feromonas. Para cada hormiga se construye una solución inicial. Cada nueva solución se obtiene navegando en la vecindad de la solución anterior, de tal manera que los elementos modificados de la solución anterior se seleccionan estocásticamente usando principalmente la reserva de feromonas y la

conveniencia de ir a esa nueva solución. La reserva de feromonas se evapora un poco después de cada iteración y se refuerza con buenas soluciones.

Algoritmos heurísticos deterministas

El Algoritmo Primer Ajuste Decreciente (First Fit Decreasing, FFD). Con este algoritmo los objetos son colocados en una lista ordenada en forma decreciente de acuerdo con el valor del tamaño de los objetos. Entonces cada objeto se escoge ordenadamente de la lista y se coloca en el primer contenedor que tenga suficiente capacidad sin usar para almacenarlo. Si ningún contenedor parcialmente lleno tiene suficiente capacidad para almacenarlo, el objeto se coloca en un contenedor vacío.

El Algoritmo Mejor Ajuste Decreciente (Best Fit Decreasing, BFD). La única diferencia con FFD es que los objetos no son colocados en el primer contenedor que tenga espacio para almacenarlo, sino en el mejor contenedor parcialmente lleno que lo pueda almacenar.

El Algoritmo Apareamiento al Primer Ajuste (Match to First Fit, MFF). Es una variación de FFD. Pide al usuario que especifique el número de contenedores complementarios. Cada uno de esos contenedores auxiliares se configura para almacenar objetos en un único rango de tamaños. A medida que la lista es procesada, cada objeto es examinado para verificar si puede ser empacado en un nuevo contenedor, con objetos de un contenedor complementario apropiado; o empacado en un contenedor parcialmente lleno; o empacado solo en un contenedor complementario. Finalmente, los objetos que aún permanecen en los contenedores complementarios se empacan utilizando el algoritmo básico.

El Algoritmo Apareamiento al Mejor Ajuste (Match to Best Fit, MBF). Es una variación de BFD y es similar a MFF, excepto por el algoritmo básico utilizado.

El Algoritmo Ajuste Modificado Decreciente (Modified Best Fit Decreasing, MBFD). El algoritmo pide un valor de porcentaje. Éste es un porcentaje de la capacidad del

contenedor que puede quedar vacío y aún calificar como “buen ajuste”. Todos los objetos de tamaño mayor del 50% de la capacidad del contenedor son colocados en un contenedor propio. Con cada contenedor parcialmente lleno, se sigue un procedimiento especial para encontrar una combinación de objetos con un “buen ajuste”. Finalmente, todos los objetos restantes se empacan utilizando el algoritmo básico BFD.

3.2.4.2 Criterios de Evaluación

Para cada caso de la muestra, el desempeño promedio de cada algoritmo se calculó efectuando 30 experimentos. En experimentos de simulación es una práctica común utilizar 30 como el tamaño de muestra mínimo aceptable [Ross 1999]. Los resultados de desempeño obtenidos fueron radio teórico y tiempo de ejecución. El radio teórico es una de las métricas usuales para *Bin-packing*, y es la razón entre la solución obtenida y el óptimo teórico. Este último es un límite inferior del valor óptimo y es igual a la suma de los tamaños de todos los objetos dividido por la capacidad del contenedor.

Para cada caso se determinó la lista de algoritmos que mejor lo resuelven. Para ello se definieron los siguientes criterios de evaluación de algoritmos: el radio teórico tiene la más alta prioridad, dos valores son iguales si tienen una diferencia máxima de 0.0001. La tabla 3.7 muestra la lista de mejores algoritmos para un subconjunto de casos de la muestra.

Tabla 3.7. Ejemplo de casos aleatorios con su lista de mejores algoritmos

Caso	Mejores algoritmos
e44i2.txt	BFD
e147i10.txt	AU
e163i2.txt	ACO
e8i10.txt	MBF,MFF,ACO

La tarea de resolver 2,430 casos aleatorios requiere una gran cantidad de tiempo: tomó cinco días con cuatro estaciones de trabajo. Sin embargo, es

importante señalar que esta inversión de tiempo es necesaria solamente una vez, al inicio del proceso de caracterización de algoritmos, para crear una muestra de tamaño mínima. Se considera que este es un tiempo razonable para generar una muestra inicial cuyo tamaño validado estadísticamente, incrementa el nivel de confianza en los resultados

3.2.5 Agrupación de Casos Dominados por un Algoritmo

El objetivo de este paso es crear grupos de casos dominados, cada uno, por un algoritmo, o un conjunto de algoritmos. La similitud entre los miembros de cada grupo se determina a través de: indicadores de complejidad de los casos y el algoritmo con el mejor desempeño para todos los casos del grupo. La salida de este método es un conjunto de grupos, donde cada grupo está asociado con un conjunto de casos y el algoritmo o los algoritmos, con el mejor desempeño para el conjunto de casos del grupo.

Método de agrupación

Dado que la muestra de casos es representativa de todas las combinaciones de los valores de los indicadores de complejidad, los casos se encuentran dispersos en todo el espacio multidimensional delimitado por los indicadores. Para ejemplificar este punto asumiremos en forma simplificada que el problema de *Bin-packing* sólo se caracterizó con dos indicadores de complejidad: p y t.

Para la caracterización con *p*, y *t*, con el método de muestreo propuesto en esta tesis se generan casos en todo el espacio factible delimitado por *p* y *t* (ver Fig. 3.6a). Como se puede observar en la Fig. 3.6b, cada conjunto de algoritmos que mejor resuelve un conjunto de casos define de manera directa un grupo $g_i \in G$. En esta figura algunos casos del mismo grupo se encuentren muy distantes entre ellos, por lo que podría ser conveniente hacer una partición, por ejemplo el grupo g_1 dividirlo en g_1 y g_6 (ver Fig. 3.6c). Con base en estas observaciones, el método de agrupación propuesto, primero forma una agrupación inicial asignando

la misma clase a todos los casos que son mejor resueltos por el mismo conjunto de algoritmos. Enseguida se dividen los grupos cuyos casos están muy dispersos.

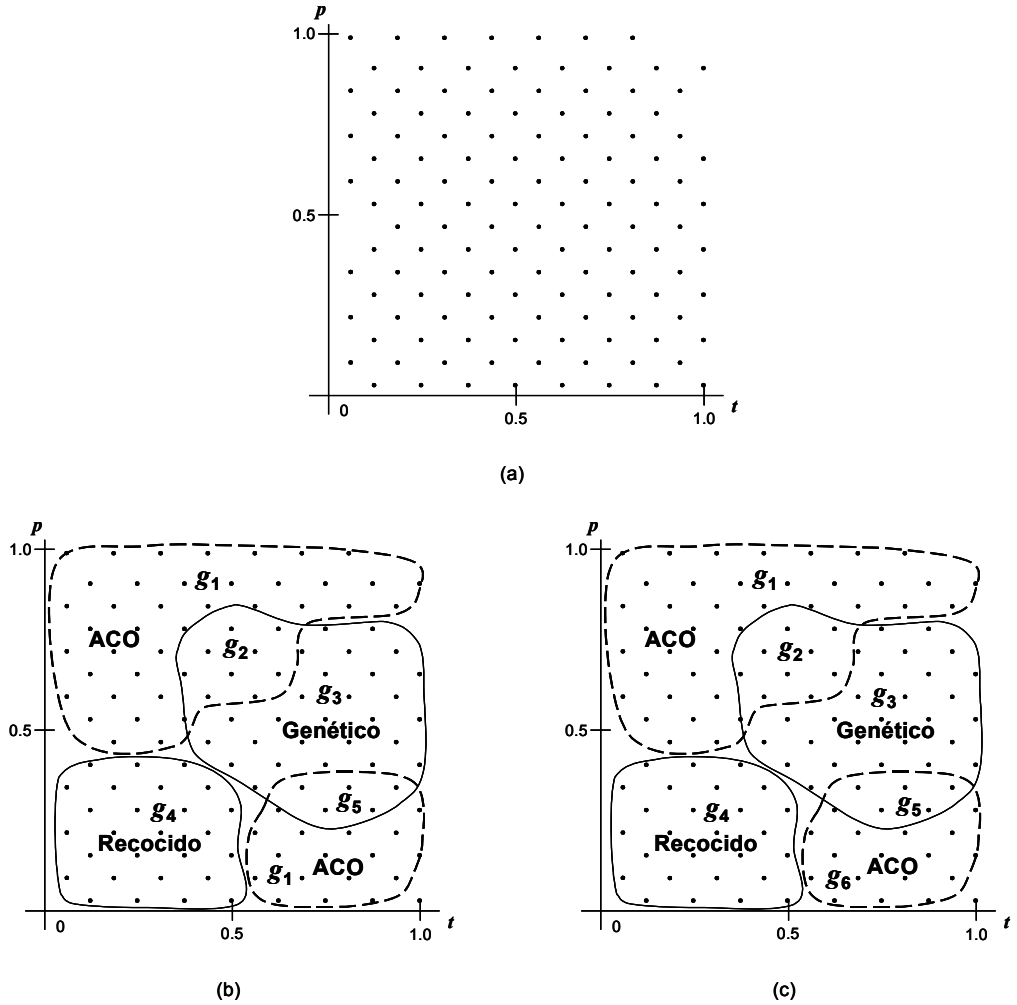


Figura 3.6. Estrategias de agrupación

Agrupación inicial

En este paso se considera que cada conjunto de algoritmos que son la mejor opción para un conjunto de casos, forma de manera natural un grupo, por lo que el objetivo del Algoritmo AGRUPACIÓN_INICIAL es asignar etiquetas de clase a cada uno de los casos. En las líneas 4-9 se asignan etiquetas nuevas evitando repeticiones, en caso contrario en las líneas 10-11 se busca la etiqueta que le corresponde al caso actual. La tabla 3.8 muestra los 16 grupos que se formaron con todos los casos de la muestra de *Bin-packing*.

Algoritmo AGRUPACIÓN_INICIAL

(C: conjunto de casos de la muestra

$MEJORES$: lista de mejores algoritmos para cada caso)

Regresa la etiqueta de GRUPO de cada caso y

los grupos formados (G, D)

1 Sea G = Conjunto de etiquetas de los grupos de casos formados

2 Sea D = Conjunto de listas de algoritmos dominantes de cada grupo

3 j = 1

4 Para cada caso c_i que pertenece a C

5 si $mejores_i \notin D$ // lista de algoritmos dominantes no registrada

6 $g_j = j$ // registra nuevo grupo

7 $d_j \leftarrow mejores_i$ // asocia algoritmos al grupo

8 $grupo_i \leftarrow j$ // asigna etiqueta al caso

9 $j = j + 1$

10 sino

11 $grupo_i$ = ENCUENTRA_ETIQUETA_GRUPO($mejores_i$, D, G)

Tabla 3.8. Agrupación inicial

Grupo	Mejores algoritmos	Número de casos
1	1,2	83
2	1,2,7	8
3	1,2,3	2
4	1,2,3,4,5	55
5	1,2,3,5	1087
6	1,2,3,5,7	142
7	1,2,3,5,6	66
8	1,2,3,5,6,7	12
9	1,2,3,4,5,6,7	8
10	1,2,3,4,5,7	3
11	2	2
12	3,5,7	3
13	3	1
14	3,5	4
15	6	262
16	7	692

División de grupos dispersos

La técnica K-medias (K-means) es una técnica muy conocida de agrupación [Alsabti 1998], y en este trabajo se utilizó para dividir grupos de casos muy dispersos en subgrupos más cercanos. El análisis fue efectuado utilizando el software comercial SPSS versión 10.0 para Windows. La cercanía entre los miembros de cada subgrupo se determinó a través de los indicadores de complejidad de los casos.

La tabla 3.9 muestra los grupos con aparente mayor amplitud en el valor de sus indicadores de complejidad. Se aplicó K-medias para confirmar que estos grupos contienen casos muy alejados, y formar dos subgrupos de cada uno de ellos. En el siguiente capítulo se detalla este método. Como se verá, la división de grupos con esta estrategia no mejoró la exactitud de la selección; pero disminuyó el error en la solución del problema *Bin-packing*.

Tabla 3.9. Amplitud de los indicadores de complejidad

Grupo	Límite	Valor del indicador en un punto extremo				
		p	b	t	f	d
5	máximo	0.024	0.001	0.010	0.000	0.001
	mínimo	0.998	0.524	0.985	0.495	0.292
6	máximo	0.005	0.009	0.010	0.000	0.000
	mínimo	0.708	1.000	0.598	1.000	0.220
	mínimo	0.082	0.267	0.954	0.488	0.252
16	máximo	0.024	0.002	0.030	0.000	0.001
	mínimo	0.997	0.149	0.746	0.289	0.292

3.2.6 Construcción del Selector de Algoritmos

En esta investigación se utilizó la técnica de aprendizaje automático C4.5 para aprender patrones de agrupación de casos de *Bin-packing* dominados por un conjunto de algoritmos. Con la formalización de grupos se aprende la relación entre indicadores de complejidad y el desempeño de algoritmos. Como resultado de este paso se obtiene un clasificador de casos, el cual permite seleccionar el algoritmo asociado al grupo al que pertenece el nuevo caso.

3.2.6.1 Construcción del Clasificador de Casos con el Método C4.5

El algoritmo C4.5 construye un árbol de decisión para clasificar nuevos casos en base al valor de sus atributos. Los nodos de este árbol corresponden a los atributos y las ramas corresponden a los diferentes valores del atributo. A cada nodo se asigna el atributo que proporciona la mayor ganancia entre los atributos todavía no considerados desde el nodo raíz.

Tabla 3.10. Ejemplo de casos caracterizados y clasificados

	Atributos de agrupación					
i	*p*	*b*	*t*	*f*	*d*	Clase
1	(-∞-0.082]	(0.063-∞)	(0.034-0.094]	(-∞-0.067]	(0.011-0.138]	2
2	(-∞-0.082]	(0.063-∞)	(0.0999-0.48]	(-∞-0.067]	(0.011-0.138]	2
3	(0.082-∞)	(0.021-0.063]	(0.099-0.48]	(-∞-0.067]	(0.011-0.138	2
4	(-∞-0.082]	(0.063-∞)	(-∞-0.039]	(0.067-0.291]	(-∞-0.011]	2
5	(0.082-∞)	(0.063-∞)	(-∞-0.039]	(0.290-∞)	(0.011-0.138]	2
6	(-∞-0.082]	(0.063-∞)	(0.500-0.746]	(-∞-0.067]	(-∞-0.011]	2
7	(0.082-∞)	(0.003-0.021]	(0.099-0.48]	(-∞-0.067]	(0.011-0.138]	2
8	(-∞-0.082]	(0.021-0.063]	(0.099-0.48]	(-∞-0.067]	(0.011-0.138]	6
9	(0.082-∞)	(0.003-0.021]	(0.500-0.746]	(-∞-0.067]	(0.011-0.138]	6
10	(0.082-∞)	(0.021-0.063]	(0.500-0.746]	(-∞-0.067]	(0.011-0.138]	6
11	(-∞-0.082]	(0.063-∞)	(0.500-0.746]	(-∞-0.067]	(0.011-0.138]	6
12	(0.082-∞)	(0.003-0.021]	(0.500-0.746]	(-∞-0.067]	(0.011-0.138]	2
13	(0.082-∞)	(0.003-0.021]	(0.500-0.746]	(-∞-0.067]	(0.011-0.138]	2
14	(0.082-∞)	(0.003-0.021]	(0.099-0.48]	(-∞-0.067]	(0.011-0.138]	2
15	(-∞-0.082]	(0.021-0.063]	(0.500-0.746]	(-∞-0.067]	(0.011-0.138]	6
16	(0.082-∞)	(0.003-0.021]	(0.099-0.48]	(-∞-0.067]	(0.011-0.138]	7
17	(0.082-∞)	(0.003-0.021]	(0.099-0.48]	(-∞-0.067]	(-∞-0.011]	6
18	(-∞-0.082]	(0.021-0.063]	(0.099-0.48]	(-∞-0.067]	(0.011-0.138]	2
19	(0.082-∞)	(0.003-0.021]	(0.099-0.48]	(-∞-0.067]	(0.011-0.138]	7
20	(0.082-∞)	(0.003-0.021]	(0.099-0.48]	(-∞-0.067]	(0.011-0.138]	7
21	(0.082-∞)	(0.003-0.021]	(0.500-0.746]	(-∞-0.067]	(-∞-0.011]	2
22	(0.082-∞)	(0.003-0.021]	(0.500-0.746]	(-∞-0.067]	(0.011-0.138]	6
23	(-∞-0.082]	(0.021-0.063]	(0.500-0.746]	(-∞-0.067]	(0.011-0.138]	6
24	(-∞-0.082]	(0.063-∞)	(0.099-0.48]	(-∞-0.067]	(0.011-0.138]	2
25	(0.082-∞)	(0.021-0.063]	(0.099-0.48]	(-∞-0.067]	(0.011-0.138]	2
26	(0.082-∞)	(0.021-0.063]	(0.48-0.500]	(-∞-0.067]	(0.011-0.138]	2
27	(0.082-∞)	(0.003-0.021]	(0.500-0.746]	(-∞-0.067]	(0.011-0.138]	2
28	(0.082-∞)	(0.003-0.021]	(0.500-0.746]	(-∞-0.067]	(0.011-0.138]	2
29	(0.082-∞)	(0.003-0.021]	(0.099-0.48]	(-∞-0.067]	(0.011-0.138]	7
30	(-∞-0.082]	(0.021-0.063]	(0.099-0.48]	(-∞-0.067]	(0.011-0.138]	2

Enseguida se describe el concepto de ganancia en información y algunos términos relacionados [Rusell 1996]. Éstos se ejemplifican utilizando un subconjunto de la muestra de casos de *Bin-packing*, el cual se presenta en la tabla 3.10. En esta tabla, los valores de los atributos de cada caso fueron transformados en intervalos para aumentar la calidad de los resultados. A este proceso de preparación se le llama discretización y es muy utilizado en el área de aprendizaje automático. Se utilizó el método EMD (Entropy Minimization Discretization) [Yang 2002], el cual es uno de los más recomendados en preprocesamiento de datos.

Entropía de una distribución de probabilidad

En general, si se tiene una distribución de probabilidades P = (p_1, p_2, …, p_n), entonces la información proporcionada por esa distribución es llamada *entropía de* P, y se obtiene con la expresión 3.14.

$$I(P) = -(p_1 \cdot \log_2(p_1) + p_2 \cdot \log_2(p_2) + ... + p_n \cdot \log_2(p_n)) \qquad (3.14)$$

donde
P = una distribución de probabilidad

Por ejemplo, si P es (0.5, 0.5), entonces $I(P)$ es 1; si P es (0.67, 0.33), entonces $I(P)$ es 0.92; y si P es (1, 0), entonces $I(P)$=0. Cuanto más uniforme es la distribución de la probabilidad, mayor es la información que se debe proporcionar para localizar un elemento. Un valor de entropía alto significa que los valores de la muestra que siguen la distribución P están dispersos sobre todo el espacio poblacional. Un valor de entropía bajo significa que los valores de la muestra vienen de una distribución variada (picos y valles). En este último caso los valores de la muestra son más predecibles. En otras palabras, la entropía es otra forma diferente de medir la variabilidad de una muestra.

Los algoritmos que construyen árboles de decisión como C4.5, se basan en el concepto de entropía mínima para dividir los datos de entrenamiento en subgrupos que se asocian a las ramas del árbol.

Entropía de clase

Si un conjunto de T registros se divide en clases inconexas C_1, C_2, …, C_k, entonces la información necesaria para evaluar la clase de un elemento de T se obtiene con la expresión 3.15 y es conocida como entropía de clase.

$\text{Info}(T) = I(P)$ donde (3.15)

P = la distribución de probabilidades de la partición (C_1, C_2, …, C_k):

$$P = \left(\frac{|C_1|}{|T|}, \frac{|C_2|}{|T|}, \ldots \frac{|C_k|}{|T|} \right)$$

Se calcula la medida de información para las clases de *Bin-packing*, utilizando la expresión 3.15:

$\text{Info}(T)$ = -(15/30 $\log_2$(15/30) + 1/30 $\log_2$(1/14) + 8/30 $\log_2$(8/30) + 6/30 $\log_2$(6/30)),

= 1.636.

Entropía de un atributo

Si primero dividimos T en base al valor de un atributo X en conjuntos T_1, T_2,…,T_n, entonces, la información que necesitamos para identificar la clase de un elemento de T se calcula con la expresión 3.16.

$$\text{Info}(X, T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \cdot \text{Info}(T_i) \quad (3.16)$$

Esta medida representa la cantidad de información esperada que adicionalmente se necesitaría especificar para que un nuevo caso sea clasificado adecuadamente, dado que la construcción del árbol alcanzó ese nodo con el atributo X. Es decir, estima el tamaño del camino que faltaría por construir si el atributo X es el que se selecciona para incorporarse al nodo actual.

De acuerdo con la información de la tabla 3.10, si el valor del indicador p se asignara al nodo actual, se formarían dos particiones: T_1 y T_2. En la tabla 3.11 se muestran los valores del indicador p para cada partición (columna 2), el número de casos de cada partición que pertenecen a cada una de las clases (columnas 3 a 5), el total de casos de la partición (columna 6) y la entropía de cada partición (columna 7).

Tabla 3.11. Particiones del indicador *p*

i	p	Número de casos				Info(T_i)
		Clase 2	Clase 6	Clase 7	T_i	
1	(-∞-0.082]	7	4	0	11	0.946
2	(0.082-∞)	11	4	4	19	1.403

Usando la información de la tabla 3.11, obtenemos la entropía del indicador *p* como sigue:

$$\text{Info}(T_1) = -(7/11 \cdot \log_2(7/11) + 4/11 \cdot \log_2(4/11) + 0/11 \cdot \log_2(0/11)) = 0.946,$$

$$\text{Info}(T_2) = -(11/19 \cdot \log_2(11/19) + 4/19 \cdot \log_2(4/19) + 4/19 \cdot \log_2(4/19)) = 1.403,$$

$$\text{Info}(p,T) = 11/30 \cdot 0.946 + 19/30 \cdot 1.403 = 1.235.$$

Ganancia en información

La ganancia es la diferencia entre la información que se necesita para identificar un elemento de T y la información necesaria para identificar un elemento de T después de obtener el valor del atributo X, es decir, la ganancia en información debida al atributo X, y se calcula con la expresión 3.17.

$$\text{Ganancia}(X,T) = \text{Info}(T) - \text{Info}(X,T) \tag{3.17}$$

Utilizando la entropía del indicador p, se calcula la ganancia total, utilizando la expresión 3.17:

$$\text{Ganancia}(p, T) = \text{Info}(T) - \text{Info}(p, T) = 1.636 - 1.235 = 0.401.$$

Elección de atributos en base a ganancia

En la Figura 3.7 se muestran las particiones de f y p, y se puede observar que f obtiene mayor ganancia en información que p. Dos ramas de f no necesitan información adicional para determinar la clase (la entropía de las ramas vale cero), y en promedio la partición con f necesita menos información que con p, lo cual puede producir un árbol de menor profundidad si se selecciona a f como el atributo del nodo actual.

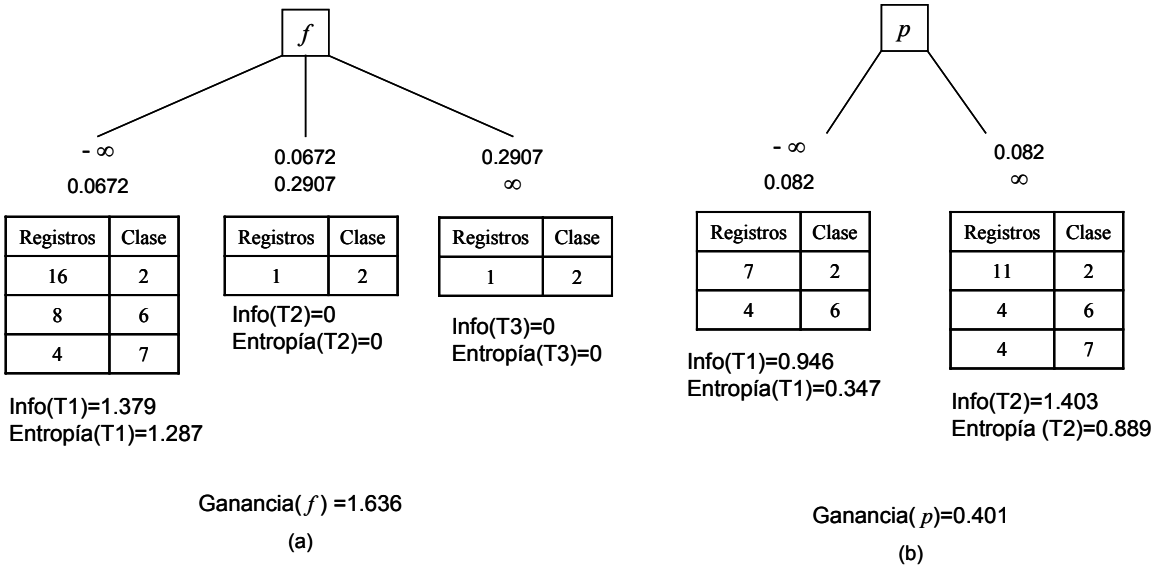


Figura 3.7. Ejemplo de ganancia en información

Algoritmo C4.5

C4.5 es un algoritmo avaro (greedy), el cual para construir un árbol de decisión primero selecciona un atributo de ganancia máxima y lo coloca en el nodo raíz (línea 4) y agrega una rama por cada posible valor del atributo (línea 5). Esto divide el registro de casos en subconjuntos, uno por cada valor del atributo (línea 6). El proceso se repite recursivamente para cada rama (línea 7), usando sólo aquellos casos correspondientes a la rama que se expande. Si todos los casos de un nodo pertenecen a la misma clase, se detiene el desarrollo de esa parte del árbol.

Algoritmo C4.5 (R: un conjunto de atributos no categóricos,
C: el atributo categórico,
S: un conjunto de entrenamiento) retorna un árbol de decisión;

1 Si S está vacío, retorna un solo nodo con valor Falla
2 Si S consta de registros todos con el mismo valor para el atributo categórico, retorna un solo nodo con ese valor;
3 Si R está vacío, entonces retorna un solo nodo como el valor más frecuente de los valores del atributo categórico que son encontrados en los registros de S; [nótese que entonces habrá errores, es decir, los registros serán clasificados incorrectamente];
4 Sea D el atributo con mayor Ganancia(D, S) entre los atributos en R;
5 Sean $\{d_j | j=1, 2, \ldots, m\}$ los valores del atributo D;
6 Sean $\{S_j | j=1, 2, \ldots, m\}$ los subconjuntos de S, cada S_j consta de registros con valor d_j para el atributo D;
7 Retorna un árbol con raíz etiquetada D y arcos etiquetados d_1, d_2, ..., d_m que se dirigen respectivamente a los siguientes árboles:
C4.5 (R-{D}, C, S_1), C4.5 (R-{D}, C, S_2), ..., C4.5 (R-{D}, C, S_m);

Aplicando el algoritmo C4.5 a los registros de la tabla 3.10 se obtiene el árbol de decisión de la Fig. 3.8. A partir de un recorrido del árbol se generan las siguientes reglas de clasificación, C es la clase con la que será etiquetado el caso:

Si f[0.067-0.291] entonces C=2.
Si f[0.291-∞] entonces C=2.
Si f[∞-0.067] & t=[0.039-0.099] entonces C=2.
Si f[∞-0.067] & t=[0.48-0.500] entonces C=2.
Si f[∞-0.067] & t=[0.099-0.48] & t[0.0630-∞] entonces C=2.
Si f[∞-0.067] & t=[0.099-0.48] & t[0.003-0.021] & t[∞-0.011] entonces C=6.
Si f[∞-0.067] & t=[0.500-0.746] & t[0.003-0.021] & t[∞-0.011] entonces C=2.
Si f[∞-0.067] & t=[0.500-0.746] & t[0.063-∞] & t[0.011-0.138] entonces C=6.
Si f[∞-0.067] & t=[0.500-0.746] & t[0.063-∞] & t[∞-0.011] entonces C=2.
Si f[∞-0.067] & t=[0.099-0.48] & t[0.003-0.021] & t[0.011-0.138] & p[0.082-∞] entonces C=7.
Si f[∞-0.067] & t=[0.099-0.48] & t[0.214-0.063] & t[0.011-0.138] & p[0.082-∞] entonces C=2.
Si f[∞-0.067] & t=[0.099-0.48] & t[0.214-0.063] & t[0.011-0.138] & p[∞-0.082] entonces C=2.
Si f[∞-0.067] & t=[0.500-0.746] & t[0.003-0.021] & t[0.011-0.138] & p[0.082-∞] entonces C=2.

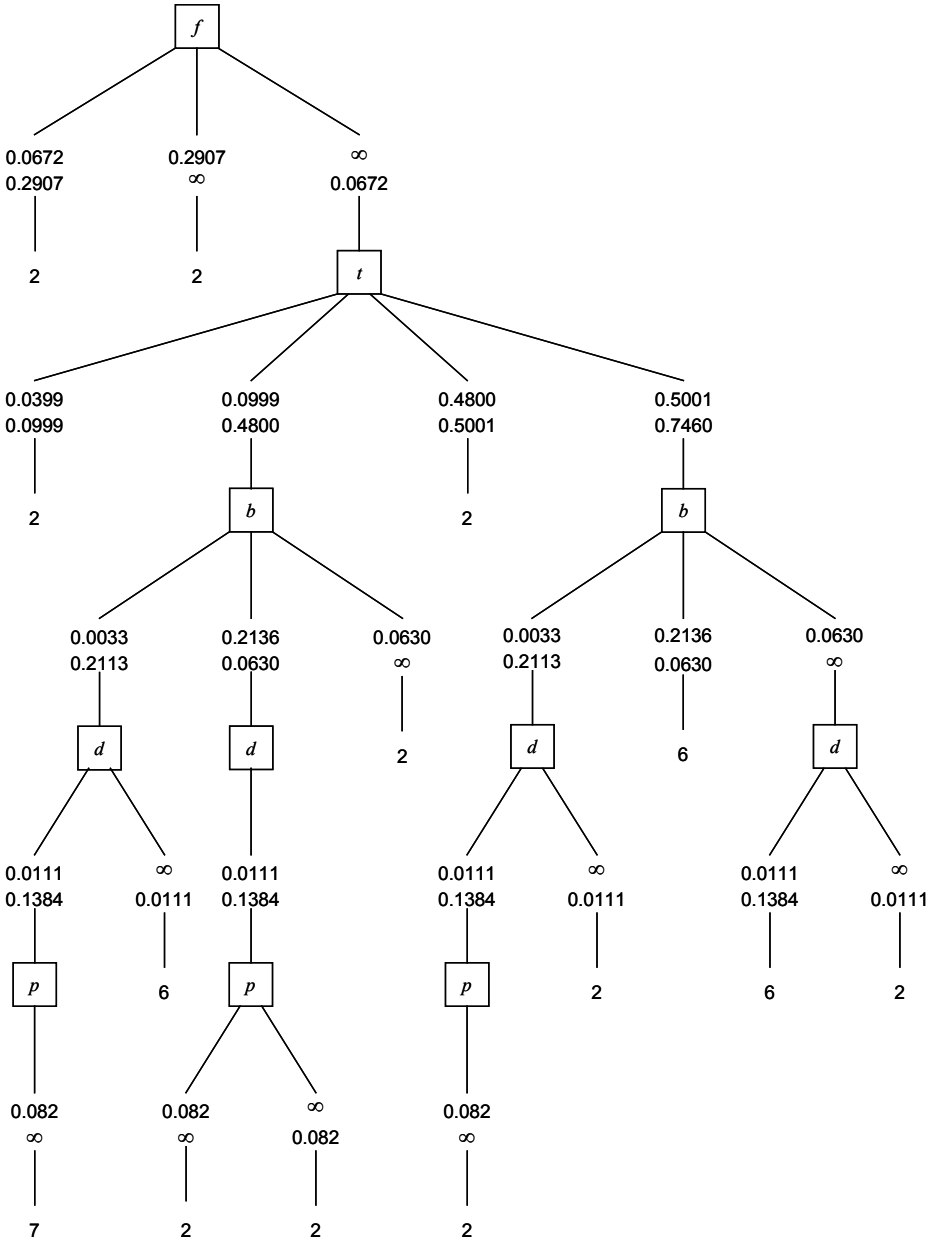


Figura 3.8. Árbol de decisión construido con C4.5

3.2.6.2 El Clasificador de Casos como un Selector de Algoritmos

Anteriormente se describió que el método C4.5 construye un árbol de decisión a partir del conjunto de datos de entrenamiento, y que posteriormente el árbol es convertido a un conjunto de reglas de clasificación. Las reglas son ordenadas por

exactitud y son aplicadas en secuencia para clasificar cada nuevo caso en el grupo correspondiente. El algoritmo seleccionado es el asociado al grupo al que pertenece el caso. El análisis de clasificación se realizó utilizando la implementación disponible de C4.5 en el Sistema Weka [Witten 2000].

El porcentaje de nuevas observaciones clasificadas correctamente es un indicador de la efectividad de las reglas de clasificación. Si éstas son efectivas sobre la muestra de entrenamiento, se espera que clasifiquen bien con nuevas observaciones cuyos grupos correspondientes son desconocidos.

Para obtener las reglas de clasificación para los casos de *Bin-packing*, se usaron cinco indicadores (ver sección 3.2.1) como variables independientes (o atributos), y el número asignado a la lista de mejores algoritmos como variable dependiente (o variable de clase). El clasificador fue entrenado con 2,430 casos de *Bin-packing* generados con el procedimiento descrito en la sección 3.2.2, y validado con la misma muestra de entrenamiento usando el método de validación cruzada (cross-validation) [Witten 2000]. Este método crea clasificadores con subconjuntos de la muestra y éstos se validan con los casos restantes, la exactitud media en la elección del algoritmo correcto fue de 70%, y la exactitud con toda la muestra fue de 84%

3.3 FASE DE PREDICCIÓN

En esta fase la relación aprendida indicadores-desempeño se usa para predecir el mejor algoritmo para un nuevo caso. Los pasos de esta fase se muestran en la Fig. 3.9. Para un nuevo caso, el paso 7 (Medición de indicadores), calcula los valores de los indicadores de complejidad del nuevo caso. El paso 8, usa el clasificador aprendido para determinar, a partir de los indicadores del nuevo caso, el grupo al que pertenece. El conjunto de algoritmos asociados a este grupo son los de mejor desempeño esperado para el nuevo caso.

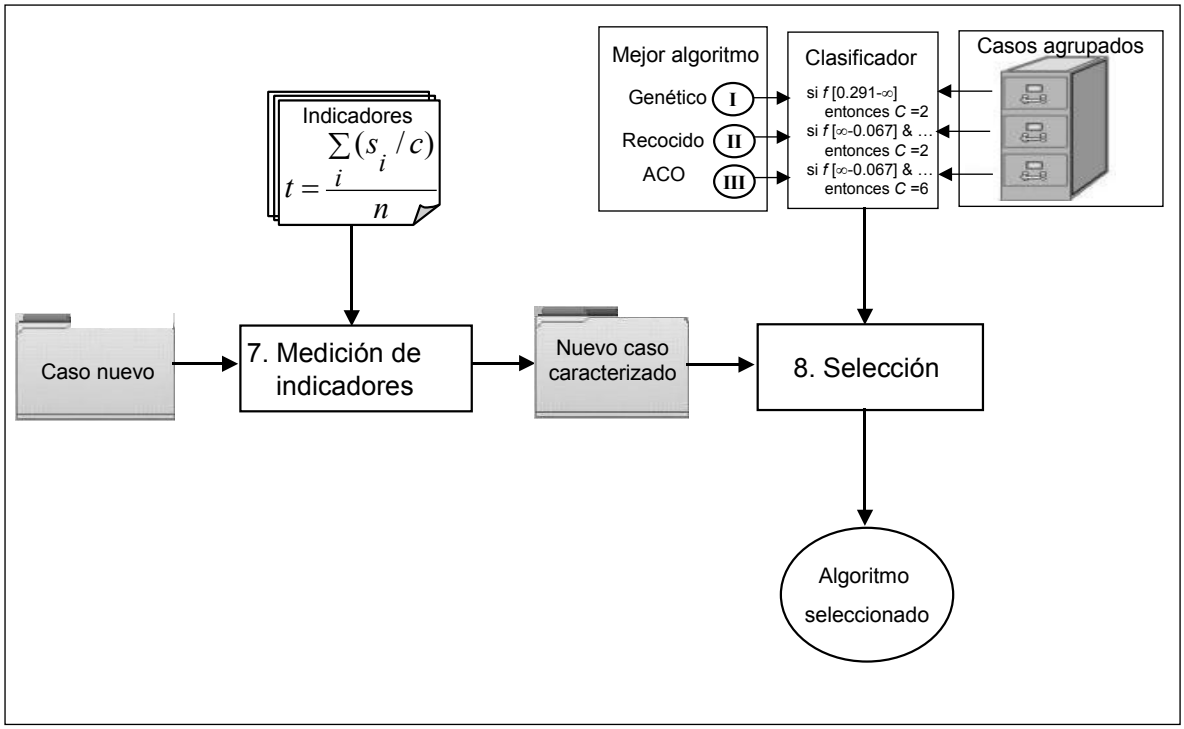


Figura 3.9. Pasos de la fase de predicción

Paso 7. Medición de indicadores. Para validar la efectividad del predictor del desempeño de algoritmos aprendido con C4.5, obtuvimos de Internet todos los casos que son aceptados por la comunidad de investigación del área. En este paso los usamos como nuevos casos. Para la mayoría de ellos, la solución óptima es conocida; para el resto se cuenta con la mejor solución encontrada (cuatro tipos de casos estándar de *Bin-packing* están disponibles en Internet). La biblioteca de Investigación de Operaciones de Beasley contiene dos tipos de problemas *Bin-packing*: casos u, y casos t; éstos fueron generados por Falkenauer [Falkenauer 1996]. La biblioteca de Investigación de Operaciones de la Universidad Tecnológica de Darmstadt contiene problemas de dos clases: casos N y casos Hard. Todos esos casos son ampliamente descritos en [Ross 2002].

La tabla 3.12 presenta una parte de los 1,369 casos estándar copiados. Para cada caso se obtuvieron los indicadores de complejidad como se explicó en la sección 3.2.1. Además, todos los casos fueron resueltos con el propósito de contrastar para cada caso, el algoritmo seleccionado contra el verdadero mejor algoritmo; la columna 7 muestra este último.

Tabla 3.12. Ejemplo de casos estándar con sus indicadores y mejores algoritmos

Caso	Indicador de complejidad					Mejores algoritmos
	p	*b*	*t*	*f*	*d*	
u120_00	0.120	0.025	0.333	0.016	0.074	ACO
u500_06	0.500	0.005	0.414	0.030	0.150	ACO
t60_00	0.060	0.050	0.333	0.000	0.073	ACO
t501_19	0.501	0.006	0.333	0.024	0.079	ACO
Hard0	0.200	0.018	0.272	0.000	0.042	ACO
Hard8	0.200	0.018	0.278	0.000	0.043	ACO
N1c2w1_o	0.050	0.068	0.294	0.300	0.210	BFD
N4c1w4_a	0.500	0.003	0.660	0.032	0.202	BFD
N1w4b3r0	0.050	0.193	0.104	0.000	0.060	BFD
N4w4b3r9	0.500	0.018	0.111	0.028	0.056	BFD

Paso 8. Selección. Se usó el clasificador aprendido para predecir el mejor algoritmo para cada uno de los 1369 casos estándar. La Tabla 3.13 presenta una parte de los resultados de la validación del selector. Para cada caso, la columna 2 muestra el verdadero mejor algoritmo y la columna 3 el algoritmo seleccionado por el predictor. Si el algoritmo predicho corresponde con el verdadero, se contabiliza el acierto. El selector predijo el algoritmo correcto para el 77% de los casos estándar.

Tabla 3.13. Resultados de la selección con 1,369 casos estándar

Caso	Mejores algoritmos	Algoritmo seleccionado	Acierto
Hard0.txt	ACO	ACO	1
N1c1w1_a.txt	FFD, BFD	FFD	1
N1c1w1_p.txt	FFD,BFD,MBF,MFF	BFD	1
u120_16.txt	FFD, BFD	ACO	0
. .	. .	. .	. .
N4w4b2r8.txt	ACO	MFF	0

3.4 FASE DE ENTRENAMIENTO CON RETROALIMENTACIÓN

Los resultados del nuevo caso, resuelto con todos los algoritmos, son utilizados para retroalimentar al sistema. Si la predicción es correcta, el grupo

correspondiente se refuerza, en caso contrario, se necesita un ajuste a la clasificación.

Los pasos de esta fase se muestran en la Figura 3.10. El objetivo es retroalimentar al sistema de selección y mantenerlo en continuo entrenamiento. Para cada nuevo caso caracterizado, cuyo mejor algoritmo fue seleccionado en la fase de predicción, el paso 9 (Solución del caso) lo resuelve y obtiene el algoritmo que verdaderamente es la mejor opción para él. Después de eso, el paso 10 (Verificación de patrones) compara el algoritmo seleccionado contra el verdadero mejor algoritmo. Si la predicción es incorrecta y la exactitud promedio rebasa un umbral de aceptación, entonces el selector debe ser reconstruido usando el conjunto de datos viejo y el nuevo; de otra manera el caso se almacena y el proceso termina.

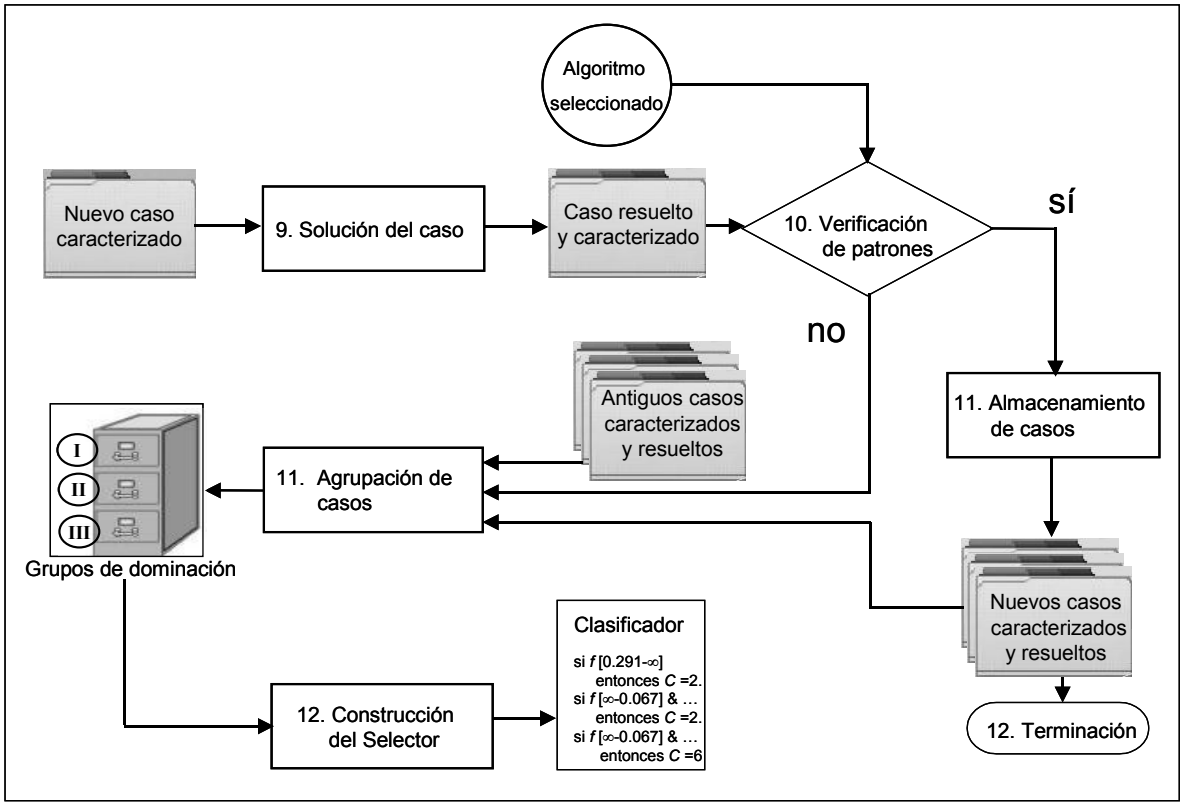


Figura 3.10. Pasos de la fase de entrenamiento con retroalimentación

Capítulo 4

VALIDACIÓN DE LA METODOLOGÍA PROPUESTA

En experimentos preliminares observamos que la calidad de un selector de algoritmos, construido con la metodología propuesta en esta tesis, puede ser afectada considerablemente por el conjunto de indicadores de complejidad desarrollado, y el método de agrupación de casos implementado. El desarrollo de un conjunto de indicadores que permita establecer diferencias entre los algoritmos evaluados y la integración de grupos de casos bien formados, son tareas que requieren un estudio a profundidad. En esta tesis se establecieron las bases teóricas y experimentales para iniciar este tipo de estudios. Experimentalmente exploramos un conjunto de indicadores y tres alternativas de agrupación de casos.

En este capítulo se muestra la experimentación realizada para validar la metodología de selección de algoritmos descrita en el capítulo 3. En particular se muestran los experimentos que permitieron, para el problema *Bin-packing* realizar lo siguiente: a) construir un selector de algoritmos básico, b) explorar métodos alternos de agrupación de casos, y c) contrastar nuestro selector con un selector aleatorio. En la sección 4.1 se describen los casos de prueba utilizados en los experimentos, y en la sección 4.2 se explica el diseño experimental, los resultados obtenidos y la interpretación de esos resultados. Los experimentos más importantes se publicaron en [Pérez 2004].

4.1 DESCRIPCIÓN DE CASOS DE PRUEBA

Dos tipos de casos de prueba se utilizaron como entrada de todos los experimentos: casos aleatorios para entrenar el sistema de selección de algoritmos y casos estándar para verificar la calidad del selector.

4.1.1 Casos Aleatorios para la Fase de Entrenamiento

Un conjunto de 2,430 casos de *Bin-packing* aleatorios se generó con el procedimiento descrito en la sección 3.2.2. Los casos aleatorios fueron utilizados para entrenar al selector de algoritmos. El generador de muestras de casos aleatorios fue codificado en el compilador Borland C++ versión 5.01.

En la tabla 4.1 se presenta un subconjunto de la muestra de casos aleatorios. La primera columna indica el número de caso, la segunda, tercera, cuarta y última columnas corresponden respectivamente al nombre del caso, al número de objetos (n), la capacidad del contenedor (c) y los tamaños de los objetos (S).

Tabla 4.1 Ejemplo de casos aleatorios de una muestra

Caso	n	c	S
e70i2	820	217	163, 44,12, 35, 80, 1...
e100i9	1858	319	1712, 1858, 1749, 1858...
e101i8	23	2436	2024, 1969, 1991, 1904...
e103i2	245	8096	8096, 8096, 7821, 8096...
e104i8	64	8781	6866, 6342, 5217, 6484....
e109i10	536	1148	351, 271, 273, 287, 287...
e112i3	579	9780	978, 1956, 326, 2445, 1956…
e144i4	724	1896	948, 948, 948, 158, 474...
e168i6	112	15628	4, 3907, 3907, 3907, 4, 4...
e194i1	542	24735	1455, 4947, 485, 1649, 1649...
e220i7	948	24599	3557, 1107, 6918, 2220...
e30i6	526	804	134, 134, 134, 201, 67...
e42i7	404	846	576, 423, 423, 282, 282...
e122i4	422	7959	2653, 4993, 1918, 1137...
e92i7	199	1952	1248, 1253, 1227, 1265...

4.1.2 Casos Estándar para la Fase de Pronóstico

Para probar el modelo de desempeño aprendido, se obtuvieron casos estándar de los dos sitios de Internet más reconocidos: La biblioteca de Investigación de Operaciones de Beasley [Library_BOR] y la biblioteca de Investigación de Operaciones de la Universidad Tecnológica de Darmstadt [Library_DOR]. Se obtuvieron 1,369 casos estándar, y en la tabla 4.2 se presenta un subconjunto de esta muestra.

Tabla 4.2 Ejemplo de casos estándar de una muestra

Caso	n	c	S
N1c1w2_b	50	100	99, 96, 95, 95, 91, 91…
N1w1b2r2	50	1000	492, 489, 483, 482, 481…
N2c2w2_t	100	120	100, 100, 100, 99, 99, 99…
N2c3w2_c	100	150	100, 99, 99, 98, 97, 97…
N2c3w4_t	100	150	100, 100, 99, 99, 97, 97…
N2w1b3r5	100	1000	621, 620, 617, 607, 602…
N2w4b2r9	100	1000	163, 162, 157, 157, 156…
N3c3w1_c	200	150	100, 100, 100, 100, 99, 99…
N3c3w4_t	200	150	100, 100, 99, 99, 99, 98…
N4c1w1_l	500	100	100, 100, 100, 100, 100…
N4c3w2_s	500	150	100, 100, 100, 100, 100…
N4w3b1r2	500	1000	168, 168, 168, 168, 168…
t120_09	120	100	39, 28, 32 37, 28, 34…
u120_02	120	150	38, 100, 60, 42, 20, 69…
u1000_11	1000	150	75, 39, 46, 32, 92, 91…

4.2 EXPERIMENTACIÓN

En todos los experimentos se utilizaron cuatro servidores DELL PowerEdge 2600, cada uno con las siguientes características: CPU Xeon a 3.06 GHz, RAM de 3.87 GB, sistema operativo Microsoft Windows Server 2003 para pequeños negocios. Cuando fue necesaria la técnica K-medias, se utilizó la implementación disponible en el software comercial SPSS versión 10.0 para Windows. De la misma manera, cuando C4.5 fue requerido, se empleó la implementación disponible en el Sistema Weka. Para simplificar la descripción de los experimentos, se usará la siguiente

notación para los algoritmos: 1= FFD, 2= BFD, 3 = MBF, 4 = MBFD, 5 = MFF, 6 = TA, 7 = ACO.

4.2.1 Experimento1: Construcción de un Selector Básico

Objetivo

Aplicar la metodología propuesta en esta tesis para construir un selector de algoritmos heurísticos que dan solución al problema *Bin-packing*.

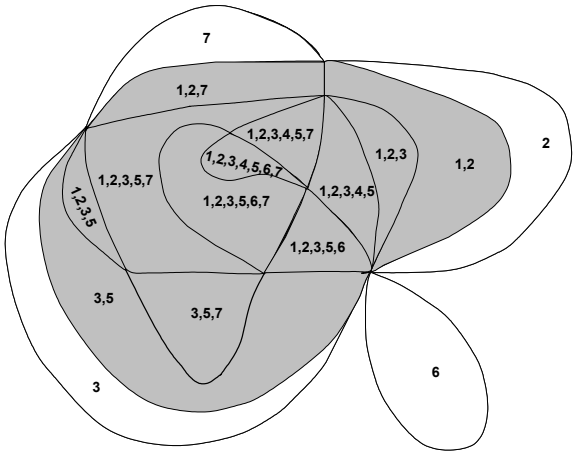
Procedimiento

Para construir el selector de algoritmos se utilizó el procedimiento de la sección 3.2. Primero se desarrollaron los indicadores de complejidad p, b, t, d, f (sección 3.2.1). Enseguida se generó una muestra representativa de 2430 casos del problema *Bin-packing* (sec. 3.2.2). Todos los casos de la muestra fueron resueltos con los siete algoritmos heurísticos descritos en la sección 3.2.4.1. Para cada caso resuelto se obtuvo la lista de los algoritmos que son su mejor opción, para ello se utilizaron los criterios de evaluación descritos en la sección 3.2.4.2. Los casos de la muestra se agruparon utilizando el método CIGI, el cual se describe líneas abajo. A partir de los grupos formados se obtuvo un selector de algoritmos utilizando el método C4.5 descrito en la sección 3.2.6.1.

Método de Agrupación CIGI

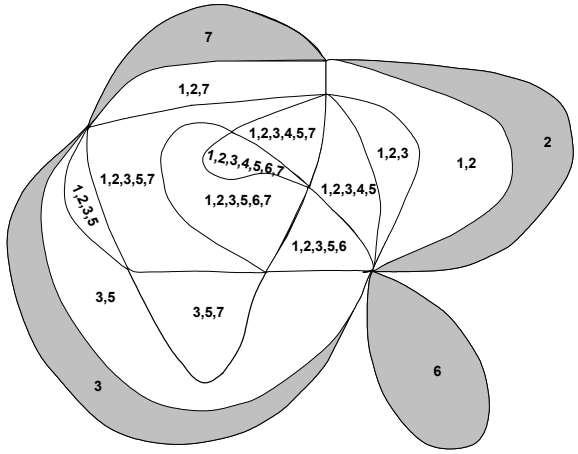
El método CIGI (Clustering by Initial Grouping and Integration) primero identifica todas las regiones múltiples, es decir, aquéllas dominadas por varios algoritmos (Fig. 4.1a). Enseguida identifica las regiones simples, aquéllas cuya mejor opción es un solo algoritmo (Fig. 4.1b). Cada región simple se constituye en un grupo, cuya etiqueta es el número del algoritmo dominante. En el ejemplo se forman los grupos 2, 3, 6 y 7 (Fig. 4.1c). Posteriormente los casos de las regiones múltiples se integran al primero de los grupos factibles. Por ejemplo, los casos dominados por los algoritmos 1,2, y 3 se integran al grupo 2, mientras que los casos dominados por 3, 5, y 7 se integran al grupo 3 (Fig. 4.1d y Fig. 4.1e).

Regiones múltiples	Algoritmos dominantes	No. de casos
1	1,2	83
2	1,2,7	8
3	1,2,3	2
4	1,2,3,4,5	55
5	1,2,3,5	1087
6	1,2,3,5,7	142
7	1,2,3,5,6	66
8	1,2,3,5,6,7	12
9	1,2,3,4,5,6,7	8
10	1,2,3,4,5,7	3
12	3,5,7	3
14	3,5	4



(a)

Regiones simples	Algoritmo dominante	No. de casos
11	2	2
13	3	1
15	6	262
16	7	692



(b)

Figura 4.1 Método de Agrupación CIGI

Regiones	No. casos	Algoritmos dominantes							Grupo
		1	2	3	4	5	6	7	
1	83	X	X						
2	8	X	X						
3	2	X	X	X					
4	55	X	X	X	X	X			
5	1087	X	X	X		X			
6	142	X	X	X		X			
7	66	X	X	X		X			
8	12	X	X	X		X			
9	8	X	X	X	X	X	X	X	
10	3	X	X	X	X		X	X	
11	2		D						2
12	3			X		X		X	
13	1			D					3
14	4			X		X			
15	262						D		6
16	692							D	7

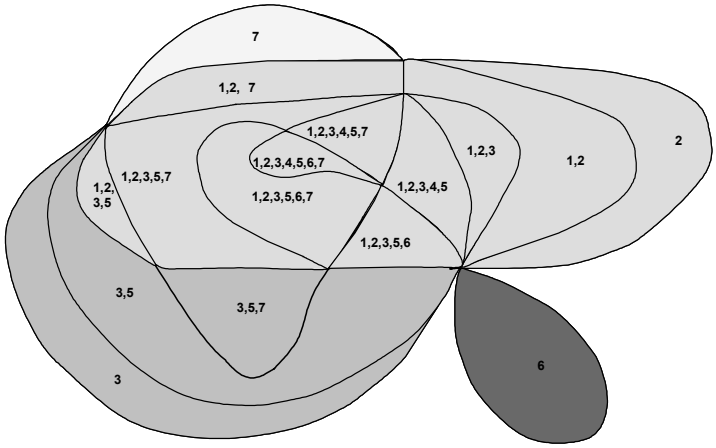
D: define grupo
X: no define grupo

(c)

Regiones	No. casos	Algoritmos dominantes							Grupo
		1	2	3	4	5	6	7	
1	83		I						2
2	8	X	I						2
3	2	X	I	X					2
4	55	X	I	X	X	X			2
5	1087	X	I	X		X			2
6	142	X	I	X		X			2
7	66	X	I	X		X			2
8	12	X	I	X		X			2
9	8	X	I	X	X	X	X	X	2
10	3	X	I	X	X		X	X	2
11	2		D						2
12	3			I		X		X	3
13	1			D					3
14	4			I		X			3
15	262						D		6
16	692							D	7

D: define grupo
I: integra al grupo
X: no define grupo

(d)



(e)

Figura 4.1 Método de agrupación CIGI (continuación)

Resultados

Para validar la calidad del selector construido, éste se aplicó en la selección de los mismos 2430 casos de la muestra de entrenamiento y en 1369 casos estándar. Con la muestra de entrenamiento se encontró que en el 86% de las veces el selector eligió el algoritmo correcto, mientras que con la muestra de casos estándar la exactitud fue de 67%.

Análisis de Resultados

De acuerdo con los resultados obtenidos, un error de pronóstico del 14% con los mismos casos que se usaron para el aprendizaje, muestra la conveniencia de refinar el conjunto de indicadores de complejidad o el método de agrupación. El primero debe establecer una fuerte diferenciación entre los algoritmos, y el segundo debe sacar ventaja de esa diferenciación.

Por otro lado, de acuerdo con lo reportado en la literatura especializada, es de esperarse que la calidad del pronóstico disminuya con nuevos casos, muy probablemente debido al conjunto de indicadores que requiere mayor capacidad de diferenciación, y cuya carencia se acentúa en la fase de pronóstico. Otra causa puede ser que en los casos estándar existan patrones que no están presentes en la muestra de entrenamiento. En tal situación, con la implementación de la fase III de la metodología sería posible aprender nuevos patrones a partir de nuevos casos resueltos.

4.2.2 Experimento2: Desarrollo de un Método de Agrupación de Casos

Objetivo

Desarrollar métodos alternos de agrupación de casos para incrementar la calidad del selector básico construido en el experimento 1.

Procedimiento

El procedimiento seguido para construir los selectores de este experimento es similar al procedimiento del experimento 1, la diferencia está en el método de agrupación de casos utilizado. Se desarrollaron tres métodos de agrupación y con cada uno de ellos se construyó un selector.

Método de Agrupación CIGIV

El método CIGIV (Clustering by Initial Grouping, Integration and Validation) es una extensión del método CIGI descrito en el experimento 1. En el método CIGI las regiones simples definen los grupos de la muestra (paso de agrupación inicial), y los casos de las regiones múltiples se incorporan a uno de los grupos definidos por las regiones simples (paso de integración). En el método CIGIV, se adiciona un paso de validación para confirmar que los casos de las regiones múltiples se integraron al grupo más afín, en caso contrario, se explora la integración a otro grupo. La validación se realiza con K-medias, la cual es una técnica muy conocida.

En la Fig. 4.2 se ejemplifica el método CIGIV. Primero, a los casos dominados por un solo algoritmo, se asigna como etiqueta de grupo el identificador del algoritmo dominante correspondiente (Fig. 4.2a). Se obtienen k = 4 grupos con etiquetas 2, 3, 6 y 7. Enseguida, cada uno de los casos dominados por más de un algoritmo, se asigna a uno de los k grupos (Fig. 4.2b). La asignación se valida usando K-medias con k = 4, la cual, para cada caso de la muestra, toma como entrada sus indicadores de complejidad y el algoritmo asociado al grupo asignado inicialmente. Como resultado se obtiene una nueva agrupación con etiquetas 1 a k (Fig. 4.2c). A cada nuevo grupo se asocia el algoritmo que la mayoría de los casos de ese grupo tiene como su mejor opción (Fig. 4.2d). Después, para cada caso se valida si el algoritmo asociado al grupo inicial es el mismo que el algoritmo asociado al nuevo grupo. En el ejemplo, uno de los casos fue incorrectamente agrupado (Fig. 4.2d), por lo que se inicia nuevamente el proceso de integración-validación para explorar otros grupos, entre ellos el grupo sugerido por K-medias.

i	Algoritmos dominantes	Grupo inicial	Algoritmo del grupo inicial
1	1,2,3		
2	1,2,3,5		
3	3,5,7		
4	6	6	6
5	7	7	7
6	7	7	7
7	2	2	2
8	7	7	7
9	3,5		
10	1,2,3,5		
11	3	3	3
:	:	:	:
2430	6	6	6

(a) Agrupación inicial

i	Algoritmos dominantes	Grupo inicial	Algoritmo del grupo inicial
1	1,2,3	2	2
2	1,2,3,5	2	2
3	3,5,7	3	3
4	6	6	6
5	7	7	7
6	7	7	7
7	2	2	2
8	7	7	7
9	3,5	3	3
10	1,2,3,5	2	2
11	3	3	3
:	:	:	:
2430	6	6	6

(b) Integración a grupos

i	*p*	*b*	*t*	*d*	*f*	Algoritmo del grupo inicial
1	0.156	0.642	0.010	0.282	0.002	2
2	0.001	1.000	0.498	0.000	0.000	2
3	0.150	0.011	0.580	0.013	0.247	3
4	0.003	0.351	0.951	0.667	0.070	6
5	0.984	0.025	0.040	0.537	0.003	7
6	0.390	0.241	0.011	1.000	0.000	7
7	0.999	0.009	0.110	0.015	0.060	2
8	0.987	0.001	0.974	0.449	0.030	7
9	0.071	0.025	0.564	0.099	0.293	3
10	0.047	0.002	0.998	0.277	0.012	2
11	0.072	0.024	0.600	0.100	0.200	3
:	:	:	:	:	:	:
2430	0.006	0.428	0.942	0.774	0.066	6

→

i	Grupo inicial	Algoritmo del grupo inicial	Grupo nuevo
1	2	2	1
2	2	2	1
3	3	3	2
4	6	6	3
5	7	7	4
6	7	7	4
7	2	2	1
8	7	7	4
9	3	3	2
10	2	2	2
11	3	3	2
:	:	:	:
2430	6	6	3

(c) Agrupación nueva con K-medias

Grupo nuevo	No. de casos dominados por un algoritmo				Algoritmo del grupo nuevo
	2	3	6	7	
1	920	0	548	0	2
2	0	692	0	0	3
3	0	0	5	0	6
4	0	0	0	262	7

(d) Algoritmo asociado al grupo nuevo

i	Grupo inicial	Algoritmo del grupo inicial	Grupo nuevo	Algoritmo del grupo nuevo	Agrupación correcta
1	2	2	1	2	✓
2	2	2	1	2	✓
3	3	3	2	3	✓
4	6	6	3	6	✓
5	7	7	4	7	✓
6	7	7	4	7	✓
7	2	2	1	2	✓
8	7	7	4	7	✓
9	3	3	2	3	✓
10	2	2	2	3	X
11	3	3	2	3	✓
:	:	:	:	:	:
2430	6	6	3	6	✓

(e) Correspondencia entre algoritmos de los grupos inicial y nuevo

Figura 4.2 Método de agrupación CIGIV

Método de Agrupación CIG

En el Método de Agrupación CIG (Clustering by Initial Grouping), se considera que cada conjunto de algoritmos que son la mejor opción para un conjunto de casos, forma de manera natural un grupo, por lo que el objetivo del Algoritmo AGRUPACIÓN_INICIAL es asignar etiquetas de clase a cada uno de los casos. Este algoritmo se detalla en la sección 3.2.5. En la Fig. 4.3 se muestran en forma tabular y gráfica los 16 grupos que se formaron con los casos aleatorios. A diferencia de los métodos anteriores, cada grupo tiene asociado un conjunto de casos y un conjunto de algoritmos que son la mejor opción para esos casos.

Grupo	Algoritmos dominantes	No. de casos
1	1,2	83
2	1,2,7	8
3	1,2,3	2
4	1,2,3,4,5	55
5	1,2,3,5	1087
6	1,2,3,5,7	142
7	1,2,3,5,6	66
8	1,2,3,5,6,7	12
9	1,2,3,4,5,6,7	8
10	1,2,3,4,5,7	3
11	2	2
12	3,5,7	3
13	3	1
14	3,5	4
15	6	262
16	7	692

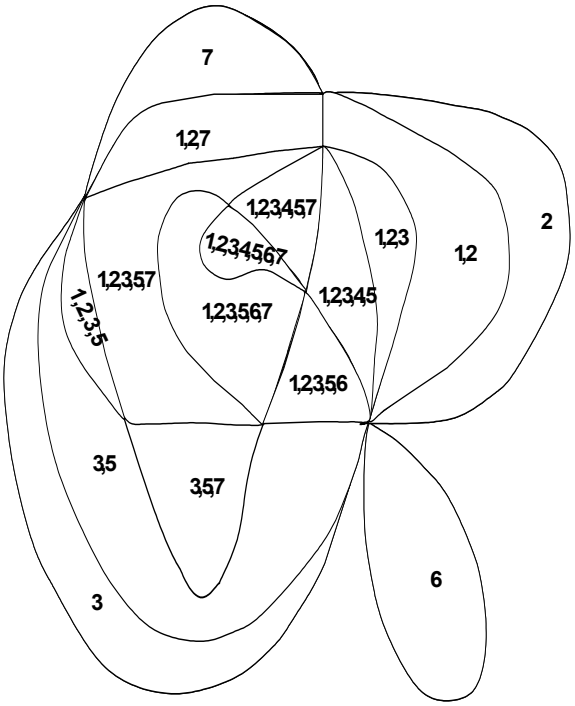
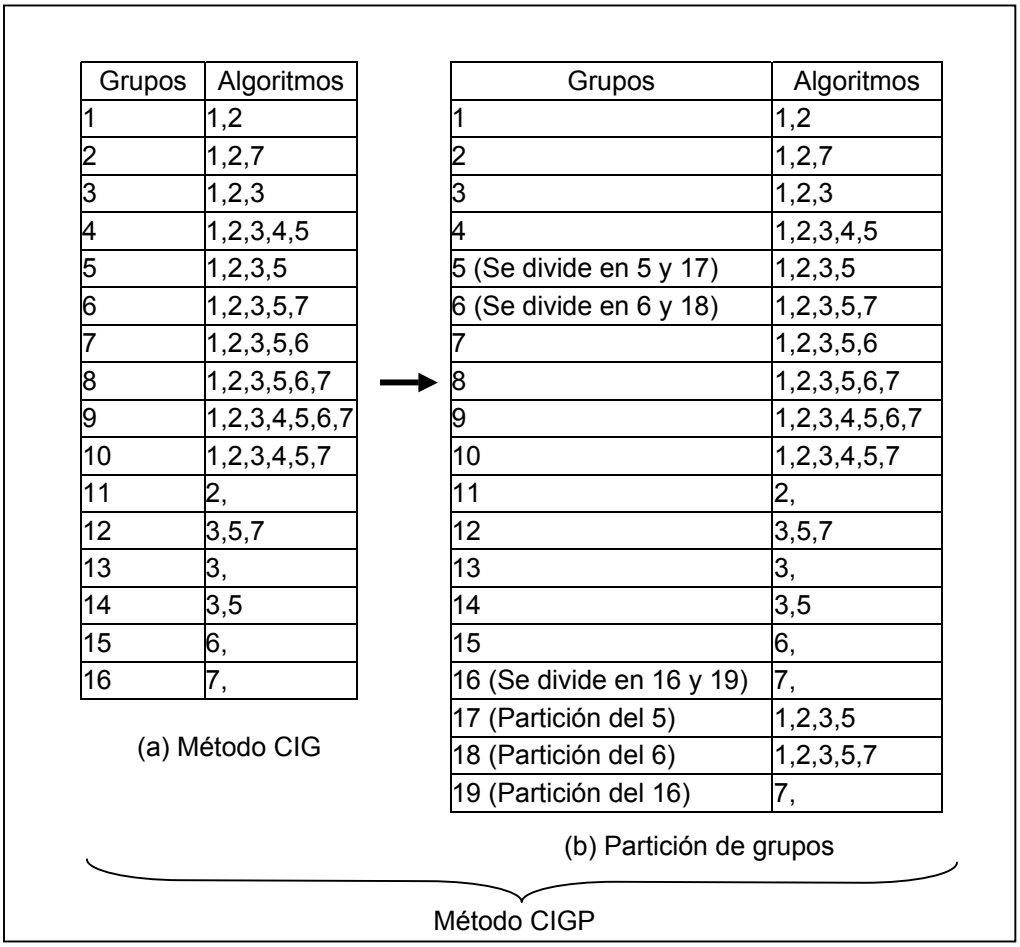


Figura 4.3 Método de agrupación CIG

Método de Agrupación CIGP

El método CIGP (Clustering by Initial Grouping and Partitioning) divide grupos de casos muy dispersos en subgrupos más cercanos; en otras palabras, a partir de

una agrupación inicial, cada grupo se divide a su vez en dos subgrupos utilizando K-medias con k = 2 y tomando como atributos de agrupación a los indicadores de complejidad de los casos. Se conservan aquellos subgrupos cuya distancia entre el caso más alejado y el más cercano al centro del grupo rebasa un umbral.



Grupos	Algoritmos
1	1,2
2	1,2,7
3	1,2,3
4	1,2,3,4,5
5	1,2,3,5
6	1,2,3,5,7
7	1,2,3,5,6
8	1,2,3,5,6,7
9	1,2,3,4,5,6,7
10	1,2,3,4,5,7
11	2,
12	3,5,7
13	3,
14	3,5
15	6,
16	7,

(a) Método CIG

Grupos	Algoritmos
1	1,2
2	1,2,7
3	1,2,3
4	1,2,3,4,5
5 (Se divide en 5 y 17)	1,2,3,5
6 (Se divide en 6 y 18)	1,2,3,5,7
7	1,2,3,5,6
8	1,2,3,5,6,7
9	1,2,3,4,5,6,7
10	1,2,3,4,5,7
11	2,
12	3,5,7
13	3,
14	3,5
15	6,
16 (Se divide en 16 y 19)	7,
17 (Partición del 5)	1,2,3,5
18 (Partición del 6)	1,2,3,5,7
19 (Partición del 16)	7,

(b) Partición de grupos

Método CIGP

Figura 4.4 Agrupación con los métodos CIG y CIGP

Para la muestra de casos aleatorios de la tabla 4.1, el método CIG genera 16 grupos (ver Fig. 4.4a). El método CIGP divide en dos particiones cada uno de los 16 grupos formados y sólo conserva las particiones de los grupos 5, 6 y 16, ya que éstas fueron las que tuvieron una diferencia, entre la distancia del caso más lejano y el más cercano al centro del grupo, mayor que 0.5 unidades (ver Fig. 4.4b). En la tabla 4.3, la columna 2 presenta las particiones del grupo 4 y 5; las columnas 3 y 4 muestran las distancias euclidianas, entre un caso y el centro del grupo, reportadas por K-medias para el caso más cercano y más lejano

respectivamente; la columna 5 indica que las particiones del grupo 5 se conservan por ser éstas las que rebasan el umbral de 0.5 especificado para la diferencia entre distancias.

Tabla 4.3 Distancia entre los casos extremos de un grupo

Grupo	Subgrupo	Distancia mínima	Distancia máxima	Rebasa umbral
4	4 a	0.122	0.358	no
	4 b	0.071	0.418	no
5	5 a	0.056	0.622	sí
	5 b	0.064	0.640	sí

Resultados

En la tabla 4.4 se presentan los resultados obtenidos con la aplicación de los cuatro métodos de agrupación antes descritos, al proceso de selección de algoritmos.

Tabla 4.4 Métodos de agrupación y resultados de la selección de algoritmos

Fase	Resultados en porcentaje	Método de agrupación			
		CIGI	CIGIV	CIG	CIGP
Entrenamiento con el algoritmo C4.5 (casos aleatorios)	Exactitud de la selección (validación cruzada)	76	77	69	66
	Exactitud de la selección	86	86	84	83
Pronóstico con el algoritmo C4.5 (casos de Internet)	Exactitud de la selección	69	69	77	75
	Desviación del óptimo	3.1	3.1	3.7	3.4

Análisis de Resultados

El análisis de la tabla 4.4 revela que para los casos de Internet, el método CIG fue el mejor de todos ya que permitió elegir el algoritmo correcto un mayor número de veces (77%). Sin embargo, en la fase de entrenamiento, CIGI y CIGIV fueron la mejor opción. Una causa posible es que CIGI y CIGIV definen reglas más precisas para los patrones que más aparecen en la muestra de entrenamiento y menos precisas para los casos de Internet, mientras que CIG efectúa lo contrario.

Por otro lado, aunque CIG permitió seleccionar el algoritmo correcto un mayor número de veces, la desviación del óptimo fue mayor que la obtenida con los otros métodos. Esta situación se presentó debido a que CIG hace pronósticos correctos en muchos casos que producen diferencias pequeñas en el desempeño de los algoritmos, y pronósticos incorrectos en casos que generan grandes diferencias entre los algoritmos. Por lo tanto, aunque la selección fue correcta un mayor número de veces, cuando la elección fue incorrecta obtuvo una solución con mucho mayor error que la obtenida por los otros métodos.

En todas las pruebas CIGP obtuvo valores intermedios. Un estudio más profundo de este fenómeno y los anteriores, permitiría combinar las técnicas propuestas para mejorar la exactitud. Sin embargo, parece que la única opción para disminuir la desviación del óptimo es explorar el modelado de indicadores de complejidad.

4.2.3 Experimento3: Selección Aleatoria vs. Selección Inteligente

Objetivo

Comparar la calidad de una selección aleatoria con la calidad de una selección inteligente, realizada con el selector básico mejorado en el experimento 2.

Procedimiento

En el experimento 1 se describió la implementación computacional de la metodología propuesta en esta tesis. En el experimento 2 se usaron cuatro métodos de agrupación para construir cuatro selectores. Todos los selectores se validaron sistemáticamente mediante un conjunto de casos de prueba copiados de Internet; con el método de agrupación CIG se obtuvo la mayor exactitud. Para contrastar los resultados obtenidos con el mejor selector, se implementó un selector aleatorio en el cual los algoritmos fueron seleccionados al azar.

Resultados

La Tabla 4.5 muestra los resultados de las pruebas para un subconjunto de casos de Internet. Para cada caso, la columna 2 presenta la lista de los algoritmos que son su mejor opción. La columna 3 muestra el algoritmo elegido con el selector inteligente y la columna 4 indica si la selección fue correcta. Las columnas 5 y 6 presentan el mismo tipo de resultados para el selector aleatorio. Considerando todos los casos de Internet disponibles, se encontró que con el selector inteligente, en el 77% de los casos se eligió el algoritmo correcto y en promedio los algoritmos seleccionados dieron resultados que sólo diferían del valor óptimo teórico en un 3.8%. Por otro lado, con el método aleatorio sólo se eligió el algoritmo correcto un 33% de las veces, y la desviación fue del 41.1%.

Tabla 4.5 Selector inteligente vs. selector aleatorio

Caso de Internet	Mejor algoritmo	Selector inteligente		Selector aleatorio	
		Algoritmo seleccionado	Acierto	Algoritmo seleccionado	Acierto
N1c1w1_b	1,2,3,5	3	1	4	0
N1c1w1_h	1,2,3,5	2	1	3	1
N1c1w1_p	1,2,3,5	2	1	4	0
N1c1w1_r	7	7	1	2	0
N4c2w4_c	1,3,4,5	3	1	1	0
N2w1b1r8	7	3	0	7	1
N4w4b2r3	1,2,3,5	7	0	2	1
t60_03	7	7	1	7	1
t120_02	7	7	1	5	0
u500_08	7	7	1	7	1

Análisis de Resultados

Debido a la falta de resultados cuantitativos de otras investigaciones relacionadas con la selección de algoritmos, se optó por hacer un estudio comparativo de nuestro método con la clásica selección aleatoria. Los resultados de esta prueba demostraron que es mejor elegir un algoritmo con un selector construido con la metodología propuesta en esta tesis, que dejar que sea el azar el que decida qué algoritmo es conveniente utilizar.

Capítulo 5

CONCLUSIONES Y TRABAJOS FUTUROS

En este capítulo se presentan las aportaciones de esta investigación, y se sugieren direcciones para trabajos futuros.

5.1 CONCLUSIONES

En este trabajo, se propone una metodología de caracterización de algoritmos que permite construir modelos de predicción para seleccionar el algoritmo que mejor resuelve un caso de un problema dado. En particular, para su aplicación al diseño de bases de datos distribuidas.

Las principales contribuciones de esta tesis son las siguientes:

a) Se aporta una nueva metodología para la selección de algoritmos, la cual consta de seis pasos, a saber: 1) modelado de indicadores de complejidad, 2) muestreo de casos, 3) agrupación de casos mejor resueltos por un mismo algoritmo, 4) aprendizaje de patrones de agrupación, 5) selección de algoritmos y 6) retroalimentación. Dicha metodología se implementó computacionalmente y se validó sistemáticamente mediante un conjunto de

casos de prueba. Como resultado de las pruebas se encontró que en el 77% de los casos se eligió el algoritmo correcto y en promedio los algoritmos seleccionados dieron resultados que sólo diferían del valor óptimo teórico en un 3.8%. Para contrastar lo anterior, para los mismos casos se seleccionaron al azar los algoritmos y se encontró que el mejor algoritmo se eligió un 33% de las veces, y la desviación del óptimo teórico fue del 41.1%. Detalles adicionales sobre los resultados se muestran en la sección 4.2.

b) Se desarrollaron cinco expresiones matemáticas que permiten modelar indicadores de complejidad del problema, lo cual posibilita que para cada algoritmo se determine un patrón de agrupamiento de los casos que ha resuelto mejor. Dichas expresiones se muestran en la página 30.

c) Se desarrolló un procedimiento para obtener modelos de predicción, a partir de los valores de los indicadores de complejidad y el desempeño de los algoritmos. En la implementación de este procedimiento se usaron técnicas de aprendizaje automático como K-medias, y C4.5. Los modelos obtenidos se muestran en la sección 3.2.6.

d) Se desarrolló un método de muestreo para generar un conjunto de casos representativos de los indicadores de complejidad del problema. El método consiste de cinco pasos: 1) desarrollo de un mecanismo para obtener los parámetros de un caso a partir de indicadores de complejidad, 2) cálculo del tamaño de la muestra basado en proporciones múltiples, 3) creación de estratos, 4) cálculo del número de casos de cada estrato y 5) generación aleatoria de casos. Para el proceso de obtener parámetros a partir de indicadores se desarrollaron cinco expresiones matemáticas y un algoritmo.

e) El uso del método de muestreo propuesto permitió generar una muestra de 2,430 casos aleatorios con un nivel de confianza del 99 % y un margen de error del 3% en las estimaciones que se hagan a partir de la muestra. Los casos fueron resueltos y caracterizados usando siete algoritmos heurísticos, de

los cuales sólo cuatro mostraron dominación absoluta sobre subconjuntos disjuntos de los casos de la muestra. Los resultados fueron utilizados para entrenar el sistema de selección de algoritmos, Los cálculos para determinar el tamaño de la muestra se encuentran en la página 38.

f) Para validar la calidad de la predicción del sistema de selección construido, se copiaron 1,369 casos estándar de los dos sitios de Internet más reconocidos por la comunidad de investigadores del área: la biblioteca de Investigación de Operaciones de Beasley y la biblioteca de Investigación de Operaciones de La Universidad Tecnológica de Darmstadt. Estos casos fueron resueltos con los siete algoritmos heurísticos disponibles.

5.2 TRABAJOS FUTUROS

Para dar continuidad a este trabajo de investigación se sugiere el desarrollo de dos metodologías, una es para el desarrollo sistemático de indicadores de complejidad de un problema y la otra para encontrar las relaciones entre un algoritmo y el conjunto de casos en los cuales ese algoritmo fue la mejor opción. A continuación se detallan los trabajos propuestos.

Modelado de indicadores de complejidad de un problema:
En este trabajo se desarrollaron cinco indicadores con base en experiencia y mediante prueba y error. Consideramos que sería de interés desarrollar un método formal para generar indicadores. Para ello se proponen las siguientes acciones:

a) Desarrollar una metodología básica que permita refinar los indicadores de complejidad desarrollados en esta tesis, y en caso necesario crear otros.
b) Generalizar la metodología para obtener indicadores de complejidad de otros problemas.

Caracterización de regiones de dominación de algoritmos:

En esta tesis se ha demostrado que es factible para un algoritmo formar un grupo de casos en los cuales dicho algoritmo es la mejor opción para su solución. Sin embargo, no se tiene una explicación formal de esa asociación. Esto es, sería de interés realizar un trabajo en profundidad para establecer por qué un algoritmo dado resolvió mejor un conjunto de casos. Para ello se proponen tres acciones:

a) Desarrollar un procedimiento que permita identificar para cada algoritmo, patrones de agrupación de casos que produzcan resultados satisfactorios en la selección de algoritmos y en la solución del problema de optimización correspondiente.
b) Estudiar teórica y experimentalmente las relaciones entre un algoritmo y el conjunto de casos que ha resuelto mejor.
c) Desarrollar una metodología que permita formalizar las relaciones encontradas.

Aplicación de la metodología a otros problemas NP-duros:

La metodología de selección de algoritmos propuesta en esta tesis se aplicó a la solución del problema clásico *Bin-packing*. Sería de interés validar la aplicabilidad de la metodología a otros problemas complejos.

REFERENCIAS

Ahuja 2003] Ahuja, R.K., Kumar, A., Jha, K.: Exact and Heuristic Algorithms for the Weapon Target Assignment Problem. Submitted to Operation Research (2003)

[Alsabti 1998] Alsabti, K., Ranka, S., Singh, V.: An Efficient K-Means Clustering Algorithm. IPPS/SPDP Workshop on High Performance Data Mining. Orlando, Florida (1998)

[Allen 1996] Allen, J., Minton, S.: Selecting the right heuristic algorithm: Runtime performance predictors. Proceedings of the Canadian AI Conference (1996)

[Anderson 2003] Anderson, R.J. The Role of Experiment in The Theory of Algorithms. In: Goldwasser, M.H., Johnson, D.S., McGeoch, C. (eds.): Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges, Series DIMACS, Vol. 5 (2003) 191-196

[Basse 1998] Basse, S.: Computer Algorithms, Introduction to Design and Analysis. Editorial Addison-Wesley Publishing Company (1998)

[Bertsekas 1991] Bertsekas, D.P.: Linear Network Optimization, Algorithms and Codes. MIT Press, Cambridge, MA (1991)

[Borghetti 1996] Borghetti, B.J.: Inference Algorithm Performance and Selection under Constrained Resources. MS Thesis. AFIT/GCS/ENG/96D-05(1996)

[Brewer 1995] Brewer, E.A.: High-Level Optimization via Automated Statistical Modeling. Proceedings of Principles and Practice of Parallel Programming (1995) 80-91

[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. Instituto Tecnológico y de Estudios Superiores de Monterrey, México (1999)

[Coffman 1997] Coffman, J.E.G., Garey, M.R., Jonson, D.S.: Approximation Algorithms for Bin-Packing, a Survey. In: Approximation Algorithms for NP-hard Problems. PWS, Boston (1997) 46-93

[Coffman 1998] Coffman, J.E.G., Galambos, G., Martello, S., Vigo, D.: Bin Packing Approximation Algorithms: Combinatorial Analysis. In: Du, D.-Z, Pardalos, P.M. (eds.): Handbook of Combinatorial Optimization. Kluwer Academic Publishers, Boston, MA (1998)

[Coffman 2002] Coffman, J.E.G., Courboubetis, C., Garey, M.R., Johnson, D.S., Shor, P.W., Weber, R.R.: Perfect Packing Theorems and the Average Case Behavior of Optimal and Online Bin Packing. SIAM Review Vol. 44 (2002) 95-108

[Cormen 2001] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein C.: Introduction to Algorithms. MIT Press 2ed (2001)

[Ducatelle 2001] Ducatelle, F., Levine, J.: Ant Colony Optimisation for Bin Packing and Cutting Stock Problems. Proceedings of the UK Workshop on Computational Intelligence. Edinburgh (2001)

[Falkenauer 1996] Falkenauer, E.: A Hybrid Grouping Genetic Algorithm for Bin Packing. Journal of Heuristics, Vol. 2 (1996) 5-30

[Fink 1998] Fink, E.: How to Solve it Automatically, Selection among Problem-solving Methods. Proceedings of the Fourth International Conference on AI Planning Systems AIPS'98 (1998) 128-136

[Frost 1994] Frost, D., Dechter, R.: In Search of the Best Constraint Satisfaction Search. Proceedings of the National Conference on Artificial Intelligence. Seattle, WA (1994) 301-306

[Frost 1997] Frost, D., Rish, I., Vila, L.: Summarizing CSP Hardness with continuous Probability Distributions. Proceedings of the 14th National Conference on AI. American Association for Artificial Intelligence (1997) 327-333

[Garey 1979] Garey, M.R., Johnson, D.S.: Computer and intractability: a Guide to the Theory of NP-Completeness. WH Freeman, New York (1979)

[Gent 1993] Gent, P., Walsh, T.: An Empirical Analysis of Search in GSAT. Journal of Artificial Intelligence Research (1993)

[Gent 1997] Gent, P., MacIntyre, E., Prosser, P., Walsh, T.: The Scaling of Search Cost. Proceedings of AAAI'97. Mit Press (1997) 315-320

[Glover 1997] Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers (1997)

[Hooker 1994] Hooker, J.N.: Needed: An empirical science of algorithms. Operations Research, Vol. 42 (1994)

[Hooker 1996] Hooker, J.N.: Testing Heuristics: We Have it All Wrong. Journal of Heuristics (1996)

[Hoos 1998] Hoos, H.H.: Stochastic Local Search -Methods, Models, Applications. PhD Thesis, Department of Computer Science from Darmstadt University of Technology. Germany (1998)

[Hoos 2000] Hoos, H.H., Stutzle, T.: Systematic vs. Local Search for SAT. Journal of Automated Reasoning, Vol. 24 (2000) 421-481

[Johnson 2002] Johnson, D.S., McGeoch, L.A.: Experimental Analysis of Heuristics for the STSP. In: Gutin, G., Punnen, A. (eds.): The Traveling Salesman Problem and its Variations. Kluwer Academic Publishers, Dordrecht (2002) 369-443

[Kalton 1983] Kalton, G.: Introduction to Survey Sampling. University of Michigan. Sage Publications (1983)

[Knuth 1975] Knuth, D.: Estimating the Efficiency of Backtrack Programs. Mathematics of Computation (1975)

[Lagoudakis 2001] Lagoudakis, M. G., Littman, M. L.: Learning to Select Branching Rules in the dpll Procedure for Satisfability. Electronic Notes in Discrete Mathematics (ENDM), Vol. 9. Boston, MA (2001)

[Lawler 1985] Lawler, E.L., Lenstra, J.K., Rinnooy K.A.H.G., Schmoys, D.B.: The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. John Wiley & Sons, New York (1985)

[Li 1997] Li, J., Skjellum, A., Falgout, R.D.: A Poly-Algorithm for Parallel Dense Matrix Multiplication on Two-Dimensional Process Grid Topologies. Concurrency, Practice and Experience, Vol. 9, No. 5 (1997) 345-389

[Library_BOR] The Beasley's OR-Library, http://www.ms.ic.ac.uk/info.html

[Library_OR] The Operational Research Library, http://www.bwl.tu-darmstadt.de/bwl3/forsch/projekte/binpp

[Lin 1995] Lin, X., Orlowska, M.: An integer linear programming approach to data allocation with the minimum total communications cost in distributed database systems. Information Science Vol. 85 (1995) 1-10

[Lobjois 1998] Lobjois, L., Lemaitre, M.: Branch and Bound Algorithm Selection by Performance Prediction. Proceedings of the Fifteenth National Conference on Artificial Intelligence. Madison, Wisconsin (1998)

[Lodi 2002] Lodi, A., Martello, S., Vigo, D.: Recent Advances on Two-dimensional Bin Packing Problems. Discrete Applied Mathematics, Vol. 123, No. 1-3. Elsevier Science B.V., Amsterdan (2002) 379-396

[McGeoch 2002] McGeoch, C.C.: Experimental Analysis of Algorithms. In: Pardalos, P.M., Romeijn, H.E. (eds.): Handbook of Global Optimization, Vol. 2 (2002) 489-513

[Micheals 2001] Micheals, R.J., Boult, T.E.: A Stratified Methodology for Classifier and Recognizer Evaluation. IEEE Workshop on Empirical Evaluation Methods in Computer Vision (2001)

[Minton 1996] Minton, S.: Automatically Configuring Constraint Satisfaction Programs: A Case Study. Journal of Constraints, Vol. 1, No. 1 (1996) 7-43

[Moret 2003] Moret, B.M.E.: Toward a Discipline of Experimental Algorithmics. In: Goldwasser, M.H., Johnson, D.S., McGeoch, C. (eds.): Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges, Series DIMACS, Vol. 5 (2003) 197-214

[Papadimitriou 1998] Papadimitriou, C., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Dover Publications (1998)

[Pérez 1999] Pérez, J.: Integración de la Fragmentación Vertical y Ubicación en el Diseño Adaptivo de Bases de Datos Distribuidas. Ph.D. thesis. ITESM Campus Cuernavaca, México (1999)

[Pérez 2000a] Pérez, J., Pazos, R.A., Romero, D., Cruz, L.: Análisis de Complejidad del Problema de la Fragmentación Vertical y Reubicación Dinámica en Bases de Datos Distribuidas. Memorias del 7mo. Congreso Internacional de Investigación en Ciencias Computacionales. México (2000) 63-70

[Pérez 2000b] Pérez, J., Pazos, R.A., Frausto, J., Romero, D., Cruz, L.: Vertical Fragmentation and Allocation in Distributed Databases with Site Capacity Restrictions Using the Threshold Accepting Algorithm. Lectures Notes in Computer Science, Vol. 1793. Springer-Verlag, Berlin Heidelberg New York (2000) 75-81

[Pérez 2002] Pérez, J., Pazos, R.A., Vélez, L. Rodríguez, G.: Automatic Generation of Control Parameters for the Threshold Accepting Algorithm. Lectures Notes in Computer Science, Vol. 2313. Springer Verlag, Berlin Heidelberg New York (2002) 119-127

[Pérez 2003a] Pérez, J., Pazos, R.A., Frausto, J., Romero, D., Cruz, L.: Data-Object Replication, Distribution and Mobility in Network Environment. Lectures Notes in Computer Science, Vol. 2890. Springer Verlag, Berlin Heidelberg New York (2003) 539-545

[Pérez 2003b] Pérez, J., Pazos, R.A., Cruz, L., Santiago, E., Fraire, H., Guerrero, V.H.: Análisis y Diseño de un Modelo de Ubicación, Migración y Replicación de objetos de Datos en Web. Memoria del 4to. Simposium Intertecnológico de Computación e Informática. Oaxtepec, Morelos (2003)

[Pérez 2003c] Pérez, J., Pazos, R.A., Rodríguez, G., Frausto, J., Cruz, L., Fraire, H.: Replication and Allocation Management of Data-Objects in Network Environments. Proceeding of the IASTED International Conference on Computer Science and Technology. Cancún, México (2003) 388-392

[Pérez 2003d] Perez, J., Pazos, R.A., Fraire, H., Cruz, L., Pecero, J.: Adaptive Allocation of Data-Objects in the Web Using Neural Networks. Lectures Notes in Computer Science, Vol. 2829. Springer-Verlag, Berlin Heidelberg New York (2003) 154-164

[Pérez 2004a] Pérez, J., Pazos, R.A., Frausto, J., Rodríguez, G., Cruz, L., Fraire, H.: Comparison and Selection of Exact and Heuristic Algorithms. Lectures Notes in Computer Science, Vol. 3045. Springer Verlag, Berlin Heidelberg New York (2004) 415-424

[Pérez 2004b] Pérez, J., Pazos, R.A., Frausto, J., Rodríguez, G., Cruz, L., Mora, G., Fraire, H.: Self-Tuning Mechanism for Genetic Algorithms Parameters, an Application to Data-Object Allocation in the web. Lectures Notes in Computer Science, Vol. 3046. Springer Verlag, Berlin Heidelberg New York (2004) 77-86

[Pérez 2004c] Pérez, J., Pazos, R.A., Frausto, J., Rodríguez, G., Romero, D., Cruz, L.: A Statistical Approach for Algorithm Selection. Lectures Notes in Computer Science, Vol. 3059. Springer Verlag, Berlin Heidelberg New York (2004) 417-431

[Purdom 1978] Purdom, P.: Tree Size by Partial Backtracking. SIAM Journal on Computing, Vol. 7, No. 4 (1978) 481-491

[Reeves 1993] Reeves, C.R.: Modern heuristic techniques for combinatorial problems. John Wiley & Sons (1993)

[Rice 1968] Rice, J.R.: On the Construction of Poly-algorithms for Automatic Numerical Analysis. In: Klerer, M., Reinfelds, J. (eds.): Interactive Systems for Experimental Applied Mathematics. Academic Press (1968) 301-313

[Ross 1999] Ross, S.M.: Simulación. Segunda edición. Prentice Hall (1999)

[Ross 2002] Ross, P., Schulenburg, S., Marin-Blázquez, J.G., Hart, E.: Hyper-heuristics, Learning to Combine Simple Heuristics in Bin-packing Problems. Proceedings of the Genetic and Evolutionary Computation Conference. Morgan Kaufmann (2002) 942-948

[Russell 1996] Russell, S., Norvig, P.: Inteligencia Artificial: Un Enfoque Moderno. Prentice Hall (1996)

[Scheaffer 1987] Sheaffer, R., Mendehall, W., Ott, L.: Elementos de Muestreo. Grupo Editorial Iberoamérica (1987)

[Selman 1998] Selman, B.: Compute-Intensive Methods for Artificial Intelligence. Proposal for NSF Faculty Early Career Development Award 1998-2002. Computing Science Department, Cornell University (1998)

[Sillito 2000] Sillito, J.: Improvements to and Estimating the Cost of Backtracking Algorithms for Constraint Satisfaction Problems. Master Thesis. University of Alberta, Edmonton, Alberta (2000)

[Soares 2000] Soares, C., Brazdil, P.: Zoomed Ranking, Selection of Classification Algorithms Based on Relevant Performance Information. In: Zighed D.A., Komorowski J., Zytkow J. (Eds.): Principles of Data Mining in Knowledge Discovery. Lecture Notes on Artificial Intelligence Vol. 1910. Springer Verlag, Berlin Heidelberg New York (2000) 126-135

[Thompson 2002] Thompson, S.: Sampling. Second Edition. Pennsylvania State University. A Wiley-Interscience Publication. John Wiley & Sons (2002)

[Tsang 1995] Tsang, E.P.K., Borrett, J.E., Kwan, A.C.M.: An Attempt to Map the Performance of a Range of Algorithm and Heuristic Combinations, Proceedings of the Artificial Intelligence and Simulated Behaviour Conference (1995) 203-216

[Wild 2000] Wild, C., Seber, G.: Chance Encounters: A First Course in Data Analysis and Inference. John Wiley & Sons, New York (2000)

[Witten 2000] Witten, I.H., Frank, E. : Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann Publishers (2000)

[Wolpert 1997] Wolpert, D. H., Macready, W. G.: No Free Lunch Theorems for Optimizations. IEEE Transactions on Evolutionary Computation, Vol. 1 (1997) 67-82

[Yang 2002] Yang, Y., Webb, G.I.: A Comparative Study of Discretization Methods for Naive-Bayes Classifiers. Proceedings of The 2002 Pacific Rim Knowledge Acquisition Workshop. Tokyo (2002) 159-173