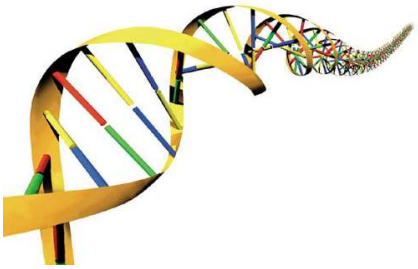
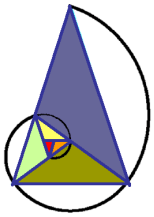
UNIVERSIDAD DE GRANADA

Departamento de Lenguajes y Sistemas Informáticos





Un modelo de rendimiento de algoritmos evolutivos aplicados a la selección de la solución deseada

Enrique Yeguas Bolívar

Editor: Editorial de la Universidad de Granada
Autor: Enrique Yeguas Bolívar
D.L.: GR. 2015-2009
ISBN: 978-84-692-2249-2

UNIVERSIDAD DE GRANADA

Programa Oficial de Postgrado de Métodos y Técnicas Avanzadas de Desarrollo de Software

Departamento de Lenguajes y Sistemas Informáticos

Un modelo de rendimiento de algoritmos evolutivos aplicados a la selección de la solución deseada

---

Enrique Yeguas Bolívar

Directores de Tesis

Robert Joan-Arinyo M. Victoria Luzón

Granada 2009

La memoria "Un modelo de rendimiento de algoritmos evolutivos aplicados a la selección de la solución deseada" que presenta D. Enrique Yeguas Bolívar para optar al grado de Doctor ha sido realizada dentro del programa de doctorado "Métodos y Técnicas Avanzadas de Desarrollo de Software" del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Granada, bajo la dirección de los doctores Robert Joan Arinyo, del Departament de Llenguatges i Sistemes Informàtics de la Universitat Politècnica de Catalunya, y María Victoria Luzón García, del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Granada.

Granada, 2009

El doctorando

Enrique Yeguas Bolívar

Los directores

Robert Joan Arinyo

María Victoria Luzón García

Agradecimientos

Llegado al final de uno de los grandes retos a los que me he enfrentado en esta vida, cabe dar las gracias a todos aquellos que de una u otra forma iluminaron este difícil y prolongado camino. En primer lugar, he de agradecer a mis directores de tesis, Robert y Vicky, su motivación, comprensión, ayuda y apoyo a pesar de la distancia. La pasión de ser investigador que ellos me han transmitido, cada uno con su particular y complementaria forma de ser, es algo que era inpensable para mí adquirir.

En segundo lugar, he de agradecer la acogida, ayuda, cariño y simpatía de todos los integrantes del Grup d'Informàtica a l'Enginyeria con los que, entre *català* y *castellà*, he convivido gratamente y compartido comidas y cafés durante mis dos estancias en la Universitat Politècnica de Catalunya de Barcelona. He de hacer especial mención a Toni por sus ideas y aportaciones y al propio Robert por toda su dedicación, discusiones e ideas y, por supuesto, por hacerme sentir como 'en casa'.

Agradezco todo el cariño y apoyo que me brindaron los profesores y becarios del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Granada durante mis inicios en el doctorado y como docente dentro de la Universidad. Sé que tengo grandes amigos y mejores personas allí, unos han quedado, otros regresado (como Vicky) y otros se fueron. Es de destacar el apoyo inicial de Juan Carlos Torres, mi tutor de doctorado.

Es necesario recoger aquí también todo lo que he recibido de mis compañeros del Departamento de Informática y Análisis Numérico de la Universidad de Córdoba. El día a día con todos y cada uno de ellos me ha hecho avanzar en la culminación de esta empresa. Merecen mención especial los que me han sufrido en el despacho: Carlos Porcel (compañero de aventuras en Cádiz), Carlos García, Rafa Muñoz y JR Romero. Así mismo, no puedo dejar de agradecer las aportaciones de los integrantes del único despacho vecino, entre ellos Luisma, y de los que en mí creyeron tras llegar a Córdoba.

No he de olvidar tampoco a todos los amigos, grandes amigos, que me han tendido la mano durante este bagaje lleno de obstáculos. Compartir unas palabras y unas risas me han hecho retomar la ruta con mayor empuje cada vez. Agradezco mucho que os hayáis cruzado en mi vida: a Juan (mi mano amiga en Barcelona), a todos mis colegas avemarianos (Luis, Alonsillo, A. Díaz, Pepe, Chanchi y el resto de inolvidables), a mis compañeros de aventura malagueña en el Parque Tecnológico (especialmente a Mavi), a mis buenos amigos de Córdoba (Alfonso entre ellos) y a Antonio Cañas y al Castro por hacer de Granada mi eterna morada.

Es el turno de tratar de agradecer lo máximo posible, mediante sólo unas líneas, a aquellos que me dieron la vida y a los que debo tanto: mis padres, Enrique y Juani. Estáis presentes siempre en mi corazón, cuán importantes sois para mí, y de ahí que haya dado todo de mi parte para que estéis orgullosos de mí. Gracias por ser vosotros y por todo vuestro amor y cariño. En los momentos difíciles no me habéis dejado caer al vacío.

Mi mejor amigo ha de llevarse también un gran agradecimiento. Ése eres tú, Juan Francisco, mi inseparable hermano. Has sido y eres mi mejor referente para salir cada día adelante con fuerza y sin miedo.

Gracias, también, a mi familia, a los que están y a los que se fueron y tanto añoro, por lo que para mí significan y por todo su calor y cariño.

Finalmente, es la tierna sonrisa y los preciosos ojos que, desde hace pocos años, a mi lado siempre están los que merecen mi agradecimiento. Gracias, María José, no imaginas cuánto me has ayudado sin tener ni idea de lo que este manuscrito recoge.

Este trabajo ha sido financiado parcialmente por el Ministerio de Educación y Ciencia y FEDER bajo los proyectos TIN2004-06326-C03-02 y TIN2007-67474-C03-01.

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

Uno de los paradigmas más recientes y prometedores en el campo del Diseño Asistido por Computador es el denominado diseño basado en restricciones geométricas. El aspecto clave de este paradigma es el problema de satisfacción de restricciones geométricas, que consiste en decidir si un conjunto de elementos geométricos (puntos, segmentos de recta, etc.) relacionados mediante un conjunto de restricciones (distancias, ángulos, tangencias, etc.) define o no un objeto rígido.

Por lo general la solución de este problema, si existe, no consiste en un único objeto sino en una familia de objetos, cada uno de los cuales es una instancia diferente construida sobre los mismos elementos. Todas y cada una de las instancias verifican las restricciones impuestas. Surge aquí un nuevo problema, el problema de la selección de la solución deseada, que puede ser definido como problema de optimización combinatoria y cuya resolución reviste gran complejidad debido a su NP-completitud.

Entre los métodos efectivos para abordar la solución de los problemas de optimización combinatoria con mayores requerimientos computacionales destacan las Metaheurísticas. La obtención de un modelo para representar el rendimiento de las Metaheurísticas cuando son aplicadas para resolver problemas de grandes requerimientos computacionales es fundamental para determinar las garantías de calidad de solución frente al tiempo de ejecución disponible. La disponibilidad de este modelo es aún mas importante para tamaños considerables del problema.

1.1. Presentación y marco del trabajo

En el diseño basado en restricciones el usuario describe un objeto mediante la definición de un conjunto de elementos geométricos tales como puntos, segmentos de línea y segmentos circulares, y un conjunto de restricciones geométricas relativas a dichos elementos. La principal tarea de los sistemas de Diseño Asistido por Computador es comprobar si el conjunto de restricciones geométricas define de forma precisa el objeto y, en ese caso, determinar la posición y orientación de los elementos geométricos.

Dado un problema geométrico bien definido y dada una asignación de valores a los parámetros de las restricciones, resolver el problema geométrico consiste en calcular las coordenadas de todos los puntos del objeto.

Si se analiza desde una perspectiva matemática, cada restricción geométrica es una ecuación. Por lo tanto, resolver un sistema de restricciones geométricas consiste en resolver el correspondiente sistema de ecuaciones. Sin embargo, se obtienen sistemas de ecuaciones no lineales muy grandes, con múltiples soluciones, que por lo general resultan difíciles de tratar.

Dado que se trata de un problema geométrico, resulta natural utilizar métodos geométricos para resolverlo. Los sistemas de resolución de restricciones geométricas constructivos permiten resolver el problema para una clase razonablemente grande de problemas geométricos. Estos sistemas transforman las restricciones originales en las coordenadas de los puntos usando únicamente un conjunto de herramientas, como por ejemplo regla y compás. Como resultado generan una secuencia simbólica de pasos constructivos básicos que, ejecutados adecuadamente, sitúan uno a uno todos y cada uno de los puntos del objeto. La realización de este algoritmo geométrico se denomina problema de contrucción geométrico.

Cuando existe una solución, el usuario espera que el sistema de resolución de restricciones geométricas le proporcione una determinada instancia con unas características concretas, y no cualquier instancia del espacio de posibles soluciones. El problema de generar automáticamente la instancia esperada es conocido como el problema de la selección de la solución deseada.

Uno de los métodos propuestos para abordar el problema de la selección de la solución deseada consiste en la definición de un conjunto de restricciones adicionales que el usuario desea que cumpla la solución a seleccionar, [107]. De esta forma, el problema puede ser formulado como un problema de optimización combinatoria cuya solución supone la maximización del número de

restricciones adicionales. Sin embargo, tal problema presenta gran dificultad para su resolución debido a su NP-completitud y a la naturaleza interactiva subyacente a los sistemas de resolución de restricciones geométricas.

Entre los métodos para abordar la solución de los problemas de optimización combinatoria con mayores requerimientos computacionales destacan las Metaheurísticas. En [106], [109], se aplicaron Algoritmos Genéticos, [114], para la resolución del problema de la selección de la solución deseada. Los algoritmos aplicados fueron tanto el Algoritmo Genético básico, [67], como Algoritmos Genéticos multimodales, [134]. Se demostró tanto la eficiencia como la efectividad de la aplicación de los Algoritmos Genéticos al problema. Por otra parte, el estudio de la optimización estadística de los parámetros, [101], que dirigen la evolución de los Algoritmos Genéticos, concretamente del Algoritmo Genético básico, en su aplicación al problema es descrita en [12]. Se propone y contrasta una configuración óptima para cada tamaño del problema. En ambos estudios, los tamaños del problema analizados correspondían a espacios de búsqueda conocidos y que pueden ser generados para su estudio sin limitaciones temporales importantes.

Con objeto de ampliar el abanico de Metaheurísticas a aplicar, estudios preliminares, [161], demostraron que los algoritmos más prometedores para el problema que tratamos correspondían claramente a las Metaheurísticas basadas en población: concretamente a los algoritmos CHC (*Cross generational elitist selection Heterogeneous recombination and Cataclismic mutation*) y PBIL (*Population-Based Incremental Learning*). Por otra parte, es importante analizar la capacidad de los algoritmos para enfrentarse a instancias del problema para las que el espacio de búsqueda sea desconocido y cuya resolución, en principio, supone un coste computacional inabordable.

La obtención de un modelo para representar el rendimiento de las Metaheurísticas cuando son aplicadas para resolver problemas de grandes requerimientos computacionales es fundamental para determinar las garantías de calidad de solución frente al tiempo de ejecución disponible. La disponibilidad de este modelo es aún mas importante para tamaños considerables del problema.

La caracterización previa del comportamiento de un algoritmo puede proporcionar una idea del efecto de su aplicación. Ello ayudará a decidir cuál será su utilidad, de modo parcial o completo, en la resolución del problema de la selección de la solución deseada en función de los requerimientos del usuario.

1.2. Objetivos

El objetivo principal de este trabajo es el de identificar, verificar y validar un modelo de rendimiento para los algoritmos evolutivos, concretamente PBIL y CHC, cuando abordan la resolución del problema de la selección de la solución deseada dentro de la resolución de restricciones geométricas.

Este objetivo general se puede descomponer en varios objetivos particulares que se enuncian a continuación:

- Analizar la influencia de los parámetros de evolución en los algoritmos CHC y PBIL, identificar los rangos de valores de los parámetros que permiten optimizar su comportamiento y definir expresiones que permitan obtener una configuración de valores de los parámetros para cada tamaño del problema.
- Definir una medida y representación del rendimiento de los algoritmos evolutivos independiente de la plataforma de ejecución e identificar un modelo estadístico de comportamiento de los algoritmos subyacente a tal representación.
- Verificar el modelo estadístico de rendimiento para los algoritmos CHC y PBIL ante tamaños del problema superiores a los analizados en anteriores trabajos, para cuyas instancias se desconozca de la existencia de solución óptima y con espacios de búsqueda difíciles de generar.
- Definir expresiones para la obtención del modelo de rendimiento para cada tamaño del problema y algoritmo y caracterizar el comportamiento de cada algoritmo en función del modelo.
- Validar el modelo de rendimiento para tamaños del problema con requerimientos de cómputo muy elevados, tanto para PBIL como para CHC. En particular, comparar los resultados previstos con los resultados empíricos: en primer lugar, el tiempo de cómputo necesario para obtener una determinada calidad de solución y, en segundo lugar, la calidad de solución asociada a un cierto tiempo disponible para obtenerla.

1.3. Estructura y contenido

La estructura de esta tesis se detalla a continuación.

El Capítulo 2 realiza una breve introducción a los problemas de optimización combinatoria. Se describen y clasifican las Metaheurísticas, métodos probabilísticos aproximados que han supuesto una gran aportación a su resolución, sobre todo a los considerados como NP-Completos. Se realiza una breve introducción al paradigma de la Computación Evolutiva: fundamentos, componentes principales, teoría de efectividad y aplicación, y a los Algoritmos Genéticos como representante más conocido.

El Capítulo 3 define el problema de la resolución de restricciones geométricas. Se expone una taxonomía relativa a las diferentes técnicas de resolución de restricciones geométricas, así como una discusión acerca de sus capacidades y limitaciones. Se presenta y formaliza la técnica constructiva de resolución de restricciones geométricas. Así mismo, se expone el problema de la selección de la solución deseada, problema principal objeto de estudio, dentro de la resolución de restricciones y se formaliza como problema de optimización combinatoria. Finalmente, se caracteriza la aplicación de Metaheurísticas al problema de la selección de la solución deseada y se presentan los estudios previos realizados.

El Capítulo 4 describe los fundamentos, características, estructura y parámetros de evolución correspondientes a los algoritmos CHC y PBIL. Se presenta un estudio estadístico exhaustivo para el ajuste de los parámetros de evolución correspondientes a cada algoritmo. Además, se propone mediante un modelo estadístico simple un conjunto de expresiones para la predicción de los parámetros de evolución ante tamaños del problema desconocidos.

El Capítulo 5 identifica y ajusta un modelo de rendimiento para los algoritmos evolutivos CHC y PBIL sobre un conjunto de tamaños del problema. Se define una medida de rendimiento independiente de la plataforma de ejecución. Se presentan, describen y analizan las distribuciones empíricas que describen la ejecución de cada algoritmo sobre cada instancia del problema. Se ajustan las distribuciones empíricas a modelos estadísticos conocidos. Como resultado, se obtiene una distribución modelo para las diferentes familias de distribuciones correspondiente cada una a un tamaño del problema.

El Capítulo 6 verifica y valida el modelo de rendimiento obtenido en el Capítulo anterior. En primer lugar, se verifica si la distribución obtenida se mantiene como representante del rendimiento para tamaños superiores y define expresiones que permiten estimar el modelo dado un tamaño del problema. En segundo lugar, se realiza la validación de los modelos estimados para predecir el comportamiento de los algoritmos CHC y PBIL ante tamaños considerables del problema.

Finalmente, el Capítulo 7 recoge las principales conclusiones del estudio, las aportaciones que ha supuesto la presente tesis y las líneas que quedan abiertas para trabajos futuros.

Para mayor claridad del documento se han incluido en los Apéndices las tablas y gráficas con los resultados de los principales experimentos realizados:

- Las gráficas correspondientes a la influencia individual de los parámetros de evolución de los algoritmos CHC y PBIL se incluyen en el Apéndice A.
- Las familias de distribuciones del modelo estimadas para el rendimiento de los algoritmos CHC y PBIL sobre los diferentes tamaños del problema se incluyen en el Apéndice B.
- Los resultados detallados de la validación del modelo de rendimiento para los tamaños del problema de elevados requerimientos de cómputo se incluyen en el Apéndice C.

Capítulo 2

Metaheurísticas. Algoritmos evolutivos

Este Capítulo comienza con una breve introducción a los problemas de optimización combinatoria: descripción, clasificación y métodos de resolución existentes.

En la actualidad, se han desarrollado nuevos métodos para abordar la solución de los problemas de optimización combinatoria con mayores requerimientos computacionales: las Metaheurísticas. Tales métodos son presentados posteriormente junto a su clasificación, características y funcionamiento global.

Uno de los paradigmas de mayor auge, hoy día, dentro de las Metaheurísticas lo constituye la Computación Evolutiva. Por ello, se exponen sus fundamentos, componentes principales, teoría de efectividad y aplicación, dejando para el final una breve introducción a la metodología más conocida para resolución aproximada de problemas dentro de la Computación Evolutiva: los Algoritmos Genéticos.

2.1. Problemas de optimización combinatoria

Muchos problemas de optimización de importancia, tanto desde un punto de vista teórico como desde un punto de vista práctico, tienen su solución en la elección de una "mejor" configuración de un conjunto de parámetros para lograr determinadas metas. Estos parecen dividirse de manera natural en dos

categorías: aquellos para los que las soluciones son codificadas con variables continuas, y aquellos cuyas soluciones se codifican con variables discretas. Entre los últimos encontramos un tipo de problemas llamados *problemas de optimización combinatoria*.

Según [122], en los problemas de optimización combinatoria, buscamos un objetivo dentro de un conjunto finito (o posiblemente contablemente infinito). Este objetivo es típicamente un entero, un subconjunto, una permutación, o una estructura de grafo.

Un problema de optimización combinatoria $P = (S, f)$ puede ser definido por:

- Un conjunto de variables $X = x_1, \ldots, x_L$,
- Un conjunto de dominios para las variables $D_1, \ldots, D_L$,
- Un conjunto de restricciones entre las variables,
- Una *función objetivo* a ser minimizada o maximizada, $f : D_1 \times \ldots \times D_L \longrightarrow \mathbb{R}^+$.

El conjunto de todas las posibles asignaciones o soluciones factibles es,

$$\begin{aligned} S = \{s = \{x_1, v_1\}, \ldots, \{x_L, v_L\} \quad | \quad v_i \in D_i, \\ i = 1, \ldots, L \wedge f \texttt{ satisface todas las restricciones}\} \end{aligned} \tag{2.1}$$

donde S es llamado normalmente *espacio de búsqueda* (o soluciones), ya que cada elemento del conjunto puede ser visto como una solución candidata.

Para resolver un problema de optimización combinatoria se debe encontrar una solución $s^* \in S$ con valor máximo de la función objetivo, esto es, $f(s^*) \geq f(s)$; $\forall s \in S$ (para minimizar tendríamos, $f(s^*) \leq f(s)$; $\forall s \in S$). A la solución s^* se le llama solución globalmente óptima de (S, f) y el conjunto $S^* \subseteq S$ es llamado conjunto de soluciones globalmente óptimas.

Algunos ejemplos destacables de problemas de optimización combinatoria son los siguientes: problema del viajante de comercio (TSP), [8], problema de la asignación cuadrática (QAP), [123], problemas de satisfacción de restricciones (CSPs), [38], problemas de planificación (*scheduling*), [21], etc.,. En el Capítulo 3 presentaremos, definiremos y describiremos el problema que tratamos en este trabajo, el problema de la selección de la solución deseada,

(*Root Identification Problem*). Este problema será formalmente expresado como problema de optimización combinatoria.

Debido a la importancia práctica de los problemas de optimización combinatoria, muchos algoritmos han sido desarrollados para su solución. Estos algoritmos pueden ser clasificados como algoritmos completos (exactos) o algoritmos aproximados. Los algoritmos exactos tienen garantizado encontrar solución en tiempo limitado para cada instancia finita de un problema de optimización combinatoria, [119], [122].

Sin embargo, para muchos problemas de optimización combinatoria que son NP-duros, [51], los métodos exactos necesitan un tiempo de computación exponencial en el peor de los casos. Incluso para instancias del problema pequeñas, estos algoritmos podrían tener un tiempo de ejecución demasiado alto para propósitos prácticos. De ahí que el uso de métodos aproximados para resolver problemas de optimización combinatoria haya conseguido muchos adeptos en los últimos años. En los métodos aproximados sacrificamos la garantía de encontrar soluciones óptimas por la gran posibilidad de encontrar buenas soluciones en una cantidad de tiempo significativamente reducida.

Los problemas de mayor dificultad de resolución dentro de los NP-duros son los *NP-completos*, [51]. Ejemplos de problemas NP-completos son TSP, QAP o el problema que nos ocupa, problema de la selección de la solución deseada, [49].

Entre los métodos aproximados básicos, [14], [15], [120], distinguimos usualmente entre métodos constructivos y métodos de búsqueda local. Los algoritmos constructivos generan soluciones desde la nada añadiendo componente a componente a una solución parcial inicialmente vacía para construir en el último paso la solución completa. Son típicamente los métodos aproximados más rápidos, pero sin embargo, devuelven a menudo soluciones de inferior calidad cuando son comparados con algoritmos de búsqueda local. Los algoritmos de búsqueda local parten de una solución inicial e iterativamente tratan de reemplazar la solución actual por una mejor solución en un vecindario de la solución actual definido apropiadamente.

Una estructura de vecindario es una función $N : S \rightarrow 2^S$ que asigna a cada $s \in S$ un conjunto de vecinos $N(s) \subseteq S$. $N(s)$ es también llamado vecindario de S.

Con la introducción de lo que es una estructura de vecindario podemos también definir el concepto de soluciones localmente mínimas o máximas. Una solución localmente máxima (o máximo local) con respecto a una estructura

de vecindario N es una solución $\hat{s}$ tal que $\forall s \in N(\hat{s}) : f(\hat{s}) \geq f(s)$ (para que tuviésemos una solución localmente mínima o mínimo local debería ocurrir que $\forall s \in N(\hat{s}) : f(\hat{s}) \leq f(s)$). Llamamos s una solución estrictamente máxima local si $\forall s \in N(\hat{s}) : f(\hat{s}) > f(s)$ (tendríamos una solución estrictamente mínima local si $\forall s \in N(\hat{s}) : f(\hat{s}) < f(s)$).

2.2. Metaheurísticas

En los últimos años un nuevo tipo de algoritmo aproximado ha emergido, el cual trata básicamente de combinar métodos heurísticos básicos en marcos de trabajo del más alto nivel lanzándose a la exploración de un espacio de búsqueda. Este tipo de métodos son hoy en día comúnmente llamados *Metaheurísticas*, [14], [15], [120].

Esta clase de algoritmos incluye los siguientes algoritmos y sus derivados:

- *Swarm Intelligence*, [16], [39].
- Computación Evolutiva, [53], [114].
- Búsqueda Local Iterativa, [127].
- *Simulated Annealing*, [86].
- Búsqueda Tabú, [52].

Hay que tener en cuenta que las Metaheurísticas no se restringen únicamente a estos algoritmos sino que estos son los más representativos y comúnmente conocidos.

Hasta ahora, no hay una definición comúnmente aceptada del término Metaheurística. Algunos investigadores en el campo trataron de proponer una definición, [120], [144], [154]. Resumiendo, subrayamos las propiedades fundamentales que caracterizan a las Metaheurísticas, [15]:

- Las Metaheurísticas son estrategias que "guían" el proceso de búsqueda.
- La meta es explorar eficientemente el espacio de búsqueda para encontrar soluciones (sub)óptimas.

- Las técnicas que constituyen los algoritmos catalogados como Metaheurísticas van desde procedimientos simples de búsqueda local a complejos procesos de aprendizaje.
- Los algoritmos considerados como Metaheurísticas son aproximados y no determinísticos.
- Incorporan mecanismos para evitar quedarse atrapados en áreas prometedoras cerradas del espacio de búsqueda.
- Los conceptos básicos asociados a las Metaheurísticas permiten una descripción de nivel abstracto del espacio de búsqueda y la evolución por el mismo.
- Las Metaheurísticas no son específicas del problema.
- Las Metaheurísticas hacen uso de conocimiento específico del dominio y/o experiencia de búsqueda (memoria) para sesgar la búsqueda.

En resumen, las Metaheurísticas son conceptos de alto nivel para explorar espacios de búsqueda usando diferentes estrategias. Estas estrategias deberían ser escogidas de tal forma que exista un equilibrio dinámico entre la explotación de la experiencia de búsqueda acumulada (que es comúnmente llamada *intensificación* o explotación) y la exploración del espacio de búsqueda (que es comúnmente denominada *diversificación*). Este equilibrio es necesario por una parte para rápidamente identificar regiones en el espacio de búsqueda con soluciones de alta calidad y por otra parte para no invertir demasiado tiempo en regiones del espacio de búsqueda que o bien han sido ya exploradas, o bien no proporcionan soluciones de alta calidad.

La estructura de las estrategias es altamente dependiente de la filosofía de la Metaheurística en sí. Hay varias filosofías diferentes que aparecen en las Metaheurísticas existentes. Algunas de ellas pueden ser vistas como "extensiones" inteligentes de algoritmos de búsqueda local. La meta de esta clase de Metaheurísticas es la de escapar de mínimos locales para proseguir la exploración del espacio de búsqueda y moverse para encontrar otros esperanzadoramente mejores óptimos locales. Esto es por ejemplo verdadero para la Búsqueda Tabú, Búsqueda Local Iterativa y *Simulated Annealing*. Estas Metaheurísticas (también llamadas métodos de trayectoria) trabajan en una o varias estructuras de vecindario distribuidas en los miembros (las soluciones) del espacio de búsqueda.

Podemos encontrar una filosofía diferente en algoritmos como *Swarm Intelligence* y Computación Evolutiva. Éstos incorporan un componente de aprendizaje en el sentido de que implícita o explícitamente tratan de aprender correlaciones entre variables de decisión para identificar áreas de alta calidad en el espacio de búsqueda. Este tipo de Metaheurística efectúa, en este sentido, un muestreo sesgado del espacio de búsqueda. Por ejemplo, en Computación Evolutiva esto es logrado por una recombinación de soluciones y en algunos algoritmos de *Swarm Intelligence* esto es logrado por un muestreo del espacio de búsqueda en cada iteración según una distribución de probabilidad.

Hay varias aproximaciones para la clasificación de las Metaheurísticas según sus propiedades. Una de las más utilizadas es la que distingue entre métodos *basados en población* frente a métodos *basados en trayectoria*, [15]. Se distingue teniendo en cuenta la forma de recorrer el espacio de búsqueda. Aquellos algoritmos que trabajan con soluciones simples son llamados métodos de trayectoria e incluyen Metaheurísticas basadas en búsqueda local, como Búsqueda Tabú o Búsqueda Local Iterativa. Todas ellas comparten la propiedad de describir una trayectoria en el espacio de búsqueda durante el proceso de búsqueda. Las Metaheurísticas basadas en población por el contrario ejecutan procesos de búsqueda que describen la evolución de un conjunto de puntos en el espacio de búsqueda, como por ejemplo *Swarm Intelligence* o Computación Evolutiva.

En este trabajo nos centraremos en los métodos basados en población, concretamente en los correspondientes a Computación Evolutiva.

2.3. Computación Evolutiva

Los algoritmos de Computación Evolutiva, [83], están inspirados en la capacidad de la naturaleza para evolucionar los seres vivientes bien adaptados a su entorno. Los algoritmos de Computación Evolutiva pueden ser caracterizados a grandes rasgos como modelos computacionales de procesos evolutivos.

En cada iteración se aplica un número de operadores a los individuos de la población actual para generar los individuos de la población de la siguiente *generación* (iteración). Normalmente, tenemos operadores para recombinar dos o más individuos para producir nuevos individuos llamados operadores de recombinación o *cruce*, y tenemos operadores que causan una autoadaptación de los individuos denominados operadores de *mutación*.

La fuerza de prosperidad en los algoritmos evolutivos es la selección de individuos basada en su *fitness* (esto puede ser el valor de una función objetivo o el resultado de un experimento de simulación, o alguna otra clase de medida). Los individuos con un *fitness* más alto tienen una probabilidad más alta de ser elegidos como miembros de las siguientes iteraciones de la población (o como *padres* para la generación de nuevos individuos). Esto corresponde al principio de supervivencia del más adaptado en la evolución natural. Es la capacidad de la naturaleza para adaptarse a un entorno cambiante la que dio la inspiración para los algoritmos de Computación Evolutiva.

Ha habido una gran variedad de algoritmos de Computación Evolutiva (EC) ligeramente diferentes propuestos a lo largo de los años. Básicamente éstos se encuadran dentro de tres categorías diferentes que han sido desarrolladas de manera independiente las unas de las otras. Estos son Programación Evolutiva (EP) desarrollada por Fogel *et al.* en 1966, [44], [45], Estrategias Evolutivas (ES) propuestas por Rechenberg en 1973, [128] y *Algoritmos Genéticos* propuestos por Holland en 1975, [67], (veánse [54] y [114] para mayor detalle). La Programación Evolutiva emana del deseo de generar inteligencia para la máquina. Mientras que la Programación Evolutiva fue propuesta originalmente para operar en representaciones discretas de máquinas de estado finitas, la mayoría de las variantes actuales son usadas para problemas de optimización continua. Los problemas más recientes tienen cabida también para la mayoría de las variantes actuales de Estrategias Evolutivas, mientras que la mayoría de las aplicaciones de los Algoritmos Genéticos son resolver problemas de optimización combinatoria. A lo largo de los años ha habido muchos estudios y trabajos acerca de métodos de Computación Evolutiva.

Hertz *et al.* , [59], proporcionan una buena visión de los diferentes componentes de los algoritmos de Computación Evolutiva y de las posibilidades para definirlos. En la Figura 2.1 mostramos la estructura básica de todo algoritmo de Computación Evolutiva.

En este algoritmo, P denota la población de individuos. Una población de hijos es generada por operadores de mutación y recombinación y los individuos para la siguiente población son seleccionados de la unión de la población antigua y de la población de hijos. Los principales rasgos de un algoritmo de Computación Evolutiva son:

- **Descripción de los individuos:** los algoritmos de Computación Evolutiva manejan poblaciones de individuos. Estos individuos no son necesariamente soluciones del problema considerado. Pueden ser soluciones

Procedimiento Computación Evolutiva (EC)

P=GenerarPoblacionInicial()
Evaluar(P)
  Mientras (no se alcancen las condiciones de terminacion) **hacer**
    P'=Recombinar(P)
    P''=Mutar(P')
    Evaluar(P'')
    P=Seleccionar($P'' \cup P$)
  FinMientras

FinProcedimiento

Figura 2.1: Esquema básico de Algoritmo de Computación Evolutiva.

parciales, o conjuntos de soluciones, o cualquier objeto que pueda ser transformado en una o más soluciones de una manera estructurada. Lo más comúnmente usado en problemas de optimización combinatoria es la representación de las soluciones como cadenas de bits o como permutaciones de L números enteros. También son posibles las estructuras de árbol o estructuras más complejas para problemas de optimización más complejos.

- **Proceso de evolución:** En cada iteración tiene que decidirse qué individuos entrarán en la población de la siguiente iteración. Esto se realiza a partir de un esquema de selección. La estrategia de sólo escoger entre los hijos como individuos para la siguiente población es llamada *reemplazamiento generacional*. Es posible transferir individuos de la población actual en la siguiente población; en ese caso, estamos tratando con un proceso de evolución de estado continuo. La mayoría de los algoritmos de Computación Evolutiva trabajan con poblaciones de tamaño fijo manteniendo siempre al menos el mejor individuo en la población actual. Es también posible tener un tamaño de población variable. En el caso de un tamaño de población continuamente reducido, la situación donde se deja un único individuo en la población (o no se pueden encontrar parejas de cruce para ningún miembro de la población) puede ser uno de los criterios de parada del algoritmo.

- **Estructura de vecindario:** Una función de vecindario $N_{ec} : I \rightarrow 2^I$ sobre el conjunto de individuos I asigna a cada individuo $i \in I$ un conjunto de individuos $N_{ec}(i) \subseteq I$ a los que se disponen actuar como patrones de recombinación para i para crear descendencia. Si un individuo puede ser recombinado con cualquier otro individuo (como por ejemplo en el Algoritmo Genético básico) hablamos de una población

desestructurada, en otro caso hablamos de una población estructurada.

- **Fuentes de información:** La forma más común de fuentes de información para crear descendencia (por ejemplo, nuevos individuos) es una pareja de padres (cruce de dos padres). Pero hay también operadores de recombinación que recombinan más de dos individuos para crear un nuevo individuo (cruce multiparental). Los desarrollos más recientes usan incluso estadísticas de población para generar los individuos de la siguiente población.

- **Infactibilidad:** Una característica importante de un algoritmo de Computación Evolutiva es la forma en la que trata con las soluciones no factibles. Cuando se recombinan individuos, podría ocurrir que el individuo resultante no sea factible. Hay básicamente tres formas diferentes de manejar tal situación. Lo más simple es rechazar los individuos no factibles. Por otro lado, para muchos problemas podría ser muy difícil encontrar individuos factibles. Por lo tanto, la estrategia de penalizar a los individuos no factibles en la función que mide la calidad de un individuo es a veces más apropiado (o incluso inevitable).

- **Estrategia de intensificación:** En muchas aplicaciones fue demostrado que usar algoritmos de mejora para mejorar el *fitness* de los individuos resulta bastante beneficioso. Mientras que el uso de una población asegura una exploración del espacio de búsqueda, el uso de las técnicas de búsqueda local ayuda a identificar rápidamente ”buenas” áreas en el espacio de búsqueda. Otra estrategia de intensificación es el uso de operadores de recombinación que tratan explícitamente de combinar ”buenas” partes de individuos. Esto también concentra la búsqueda ejecutada por el algoritmo de Computación Evolutiva en áreas de individuos con ciertas propiedades ”buenas”.

- **Estrategia de diversificación:** Una de las mayores dificultades de los algoritmos de Computación Evolutiva (especialmente al aplicar la búsqueda local) es la convergencia prematura hacia soluciones subóptimas. El mecanismo más común para diversificar el proceso de búsqueda es el uso de un operador de mutación. La simple forma de un operador de mutación supone ejecutar una perturbación aleatoria de un individuo, introduciendo un tipo de ruido. Una forma más elaborada de aplicar diversificación es la de introducir sistemáticamente nuevos individuos provenientes de áreas del espacio de búsqueda no exploradas apenas. Esto se puede lograr a partir de la construcción de individuos en base

> a alguna memoria de historia que haya guardado pista de, por ejemplo, la frecuencia con que aparecen ciertos componentes de las soluciones en las soluciones o individuos de generaciones pasadas.

Dentro de la Computación Evolutiva, los algoritmos en que nos hemos centrado en este trabajo pertenecen a la familia de los Algoritmos Genéticos. De ahí que, a continuación, presentemos una introducción a los mismos.

2.4. Algoritmos Genéticos

Los métodos basados en población dentro de las Metaheurísticas tienen como más célebres y conocidos representantes a los Algoritmos Genéticos, [67]. Se trata de una gran cuna de algoritmos basados en los fenómenos que supone la evolución natural canalizados hacia el logro de cada vez mejores soluciones en un problema de optimización. En nuestra pirámide de particularización del concepto Metaheurística nos introducimos ahora dentro de todos los fundamentos, componentes, estructura, disposición, desarrollo y evolución de los Algoritmos Genéticos, [68]. Es esta la última piedra de la filosofía base sobre la que se encasillan los algoritmos a tratar en nuestro trabajo.

Los Algoritmos Genéticos son procedimientos adaptativos para la búsqueda de soluciones en espacios complejos, inspirados en los procesos de evolución natural y evolución genética, [58].

La idea básica consiste en mantener una población de *cromosomas* o individuos, donde cada cromosoma es una solución candidata a un problema concreto. Asociado a cada cromosoma existe un valor de bondad o adaptación que describe la adecuación al problema de la solución que representa. La población evoluciona con el tiempo por medio de un proceso de competición y variación controlada. El procedimiento de competición, denominado *mecanismo de selección*, utiliza las adaptaciones de los cromosomas para determinar cuáles de ellos se usarán para crear otros nuevos, mientras que estos nuevos cromosomas se obtendrán a través de la aplicación de operadores genéticos sobre los cromosomas seleccionados en la competición.

Las soluciones codificadas en una población de cromosomas compiten para ver cuál constituye la mejor solución (aunque no necesariamente la mejor de todas las soluciones posibles). El ambiente, constituido por la existencia de un grupo de soluciones alrededor de cada una, ejercerá una *presión selectiva* sobre la población, de forma que sólo los cromosomas mejor adaptados

(aquellos que resuelvan mejor el problema) sobrevivan o leguen su material genético a las siguientes generaciones, igual que en la evolución de las especies. La diversidad genética se introduce mediante mutaciones y reproducción sexual. La presión selectiva se encarga de que sólo las buenas partes de los distintos cromosomas se perpetúen, y poco a poco vayan formando una buena solución. Cuanto más alta sea la presión selectiva mayor será la convergencia de la población y viceversa. La presión selectiva será el impulso que proporcionamos a la convergencia de la población en función de las modificaciones ejercidas sobre los cromosomas individuales y la relación que se establece entre la exploración de la población completa en busca de las regiones y soluciones prometedoras.

Una de las principales ventajas que presenta este tipo de algoritmos es su capacidad para explotar la información acumulada sobre un espacio de búsqueda y, de este modo, dirigir las siguientes búsquedas hacia los mejores subespacios. Por esto se aplican sobre espacios grandes, complejos y parcialmente definidos donde las técnicas clásicas de búsqueda no son apropiadas.

Cuando se trabaja con Algoritmos Genéticos se emplea un vocabulario muy específico que hace uso de términos propios de la genética. Así, se denominan *genotipos* a los propios cromosomas, *fenotipos* a las soluciones representadas por los cromosomas, *genes* a las unidades de un cromosoma, *loci* a las posiciones de los cromosomas y *alelos* a los posibles estados de un gen.

2.4.1. Estructura de un Algoritmo Genético

Un Algoritmo Genético comienza con una población de cromosomas generados aleatoriamente, y va obteniendo mejores cromosomas gracias a la aplicación de los operadores genéticos. La evolución se produce sobre la población en forma de selección natural. Durante sucesivas iteraciones, denominadas generaciones, se evalúa la adaptación o adecuación de los cromosomas como soluciones, y en base a esta evaluación se forma una nueva población de cromosomas usando un mecanismo de selección y operadores genéticos específicos, tales como el cruce y la mutación. Para calcular el valor de adaptación de un cromosoma es necesaria una función de adaptación. Todo problema a resolver debe proporcionar una función de evaluación o adaptación que devuelva, para cada cromosoma, un valor numérico proporcional a la utilidad o adaptación de la solución que representa. En resumen, en cada generación se llevan a cabo los tres pasos siguientes:

1. Evaluación de los individuos de la población.

2. Formación de una población intermedia a través del mecanismo de selección, en función de la adaptación de cada cromosoma.

3. Formación de una nueva población a partir de los operadores genéticos de cruce y mutación.

Estos tres pasos se repiten hasta que el sistema deja de mejorar o hasta que se alcanza un número máximo de generaciones especificado por el usuario.

La estructura de un Algoritmo Genético es tal y como se presenta en la Figura 2.2.

Procedimiento AG básico

t=0
Inicializar($P(t)$)
Evaluar estructuras en $P(t)$
Mientras (no se alcancen las condiciones de terminacion) **hacer**
t=t+1
Seleccionar$_r$ $C(t)$ de $P(t-1)$
Recombinar estructuras en $C(t)$ para formar $C'(t)$
Evaluar estructuras en $C'(t)$
Seleccionar$_s$ $P(t)$ de $C'(t)$ y $P(t-1)$
FinMientras

FinProcedimiento

Figura 2.2: Estructura de un Algoritmo Genético básico.

2.4.2. Representación de cromosomas

Debido a que los Algoritmos Genéticos utilizan una representación codificada del problema, la elección de una representación adecuada al problema que se maneja se convierte en un elemento clave del funcionamiento del algoritmo, [54]. Con frecuencia se ha utilizado la codificación binaria, donde los cromosomas son cadenas de bits (cadenas de ceros y unos). Éstos son los llamados Algoritmos Genéticos con codificación binaria. Pero además de la codificación binaria son múltiples las representaciones empleadas, entre ellas vectores de números reales, vectores de números enteros, listas ordenadas o expresiones representadas como árboles.

2.4.3. Mecanismo de selección

Si consideramos una población P de PS cromosomas $c_1, \ldots, c_{PS}$, al aplicar el mecanismo de selección, [55], [115], sobre P obtendremos una población intermedia, P'. Esta población contendrá copias de cromosomas de P, donde el número de copias de cada cromosoma dependerá de su valor de adaptación. Aquellos cromosomas que tengan un valor de evaluación alto tienen mayor probabilidad de contribuir con copias para P'. Sobre esta población intermedia se aplicarán posteriormente los operadores genéticos. El procedimiento consta de dos pasos:

1. Para cada cromosoma c_i de la población P calcular su probabilidad asociada $p(c_i)$. El método más utilizado para calcular esta probabilidad es el modelo proporcional, donde $p(c_i), i = 1, \ldots, PS$ se calcula como:

$$p(c_i) = \frac{f(c_i)}{\sum_{j=1}^{PS} f(c_j)}$$

   donde $f(c_i)$ es el valor de la función de evaluación para el cromosoma c_i. De esta manera, los cromosomas con función de evaluación por encima de la media reciben más copias que aquellos con función de evaluación por debajo de la media.

2. Una vez calculadas las probabilidades anteriores, se asigna aleatoriamente a cada elemento de P el número de copias suyas que aparecerán en P'. Este paso se lleva a cabo mediante un método de muestreo. El más simple, denominado muestreo aleatorio simple, consiste en simular el comportamiento de una ruleta en la que existe para cada individuo de la población una región proporcional al valor de su probabilidad de selección. Cada vez que se lanza, la ruleta determinará un cromosoma para la población intermedia, con lo que tras lanzar PS veces se completa la población intermedia. Aquellos cromosomas con mayor probabilidad tendrán asignada en la ruleta una proporción de espacio mayor, lo que significa que la ruleta se detendrá con mayor probabilidad en los espacios correspondientes a tales individuos.

Se distingue entre dos tipos de selección:

- **Selección para la reproducción** ($Seleccionar_r$): comentada anteriormente y aplicada sobre la población inicial de cada iteración para

seleccionar una población intermedia a la que aplicar los operadores genéticos.

- **Selección para la supervivencia** ($Seleccionar_s$): utilizada al finalizar cada iteración en el Algoritmo Genético para seleccionar la población inicial para la siguiente iteración. Esta selección se realizará entre la población inicial de la iteración actual y la generada al final de la iteración actual a partir de la selección para la reproducción y la aplicación de los operadores genéticos. Depende del tipo de Algoritmo Genético, algo que comentaremos posteriormente.

Como complemento al mecanismo de selección y dentro de la selección para la supervivencia puede emplearse una estrategia denominada elitismo, [125]. Mediante tal técnica se asegura que el mejor individuo de la actual generación estará presente en la siguiente, ya que es muy posible que el mejor cromosoma de la generación en curso desaparezca en la siguiente al aplicar el mecanismo de selección o los operadores genéticos.

2.4.4. Operadores genéticos

Una vez obtenida la población intermedia tras llevar a cabo el mecanismo de selección se aplican sobre sus cromosomas los operadores de cruce y mutación, [54], [68], [114]. Describamos a continuación tales operadores.

El operador de cruce, [147], causa la recombinación del material genético de los cromosomas padres, es decir, el cruce combina las características (genes) de dos cromosomas padres para generar uno o dos cromosomas hijos. Es por esto por lo que al operador de cruce también se le llama operador de recombinación. Una propiedad importante del cruce es que explota el espacio de búsqueda asociado a los cromosomas padres. Las definiciones para los operadores de cruce (y como veremos posteriormente, para los operadores de mutación), dependen de la representación escogida para el problema. Por último, hay que tener en cuenta la denominada probabilidad de cruce, que establece la probabilidad de aplicar el operador de cruce sobre parejas de cromosomas de la población intermedia.

El operador de mutación, [100], altera arbitrariamente uno o más componentes (genes) de un cromosoma para aumentar la variabilidad estructural de la población. El papel de la mutación en los algoritmos genéticos es restaurar material genético perdido o no explorado en la población. De esta forma, se asegura que la probabilidad de alcanzar cualquier punto del espacio de

búsqueda nunca es nula. Cada gen en cada cromosoma sufre una mutación de acuerdo con un parámetro de control, denominado probabilidad de mutación, p_m. En este caso ocurre también que la definición del operador de mutación dependerá de la representación de cromosomas seleccionada para la resolución del problema tratado.

2.4.5. Tipos de Algoritmos Genéticos

Podemos distinguir dos modelos dentro de los Algoritmos Genéticos, véanse [54], [67], [156]. Tales modelos se describen a continuación:

- **Modelo generacional o clásico:** Durante cada generación se crea una población completa con nuevos individuos mediante la selección de padres de la población anterior y la aplicación de los operadores genéticos sobre ellos. La nueva población reemplaza directamente a la antigua.
- **Modelo estacionario:** Durante cada generación se escogen dos padres de la población (usando muestreo aleatorio simple o cualquier otro tipo de muestreo) y se le aplican los operadores genéticos. Los dos nuevos cromosomas (o el único cromosoma, dependiendo de si se obtiene uno o dos hijos) reemplazan a dos cromosomas (o uno) de la población, que suelen ser los dos (o uno) peores, es decir, con peor adaptación. Este esquema produce una presión selectiva alta cuando se reemplazan los peores, lo que hace que se converja muy rápidamente. Para evitarlo existen diferentes alternativas, como escoger de forma aleatoria los individuos o reemplazar los individuos más antiguos de la población.

2.4.6. Propagación genética: El teorema de los esquemas

Denotamos como *esquema*, [68], [69], a un cromosoma patrón capaz de verse replicado en sus semejantes y de ser trasladable genéticamente a sus descendientes.

Considerando cromosomas binarios, tendremos *strings* o cadenas construidos sobre un alfabeto binario V=$\{0,1\}$. Sea $A(t)$ la población en el instante (generación) t formada por los cromosomas A_1,A_2,...,A_{PS}, siendo PS el tamaño de la población. Utilizaremos el alfabeto V_+=$\{0,1,*\}$ para indicar los esquemas contenidos en individuos (cromosomas) y poblaciones, donde el

símbolo * puede representar tanto al valor 0 como al valor 1. Así, por ejemplo, dado el esquema H=*11*0**, puede verse que los cromosomas A_1=0111000 y A_2=1111011 lo verifican y por otra parte, que el cromosoma A_3 =0010000 no lo verifica.

Los esquemas presentan ciertas propiedades. Estas son las siguientes:

- El **orden** de un esquema H, denotado por $o(H)$, es el número de posiciones fijas presentes (las distintas de *). Ej.: o(011*1**) es 4.
- La **longitud de definición** de un esquema H, denotada por $\delta(H)$, es la distancia entre la primera y la última posición definida. Ejs.: $\delta(011*1**) = 5 - 1 = 4$ y $\delta(0******)= 1 - 1 = 0$.

El teorema de los esquemas se basa en la noción de bloques de construcción (entendiendo por bloque de construcción a una cadena dentro de un cromosoma que sirve como patrón para construir nuevos cromosomas). Una buena solución a un problema está constituida por unos buenos bloques, igual que una buena máquina está hecha por buenas piezas. El cruce es el encargado de mezclar bloques buenos que se encuentren en los diversos progenitores, y que serán los que den a los descendientes un buen *fitness*. La presión selectiva se encarga de que sólo los buenos bloques se perpetúen, y poco a poco vayan formando una buena solución. El teorema de los esquemas establece que la cantidad de buenos bloques se va incrementando con el tiempo de ejecución en un Algoritmo Genético, y es el resultado más importante dentro de los Algoritmos Genéticos.

Cada etapa dentro de un Algoritmo Genético tiene un efecto sobre los esquemas:

- **Efecto de la reproducción:** sea m=$m(H,t)$ la representación de las m apariciones del esquema H en la población $A(t)$. Un *string* (cromosoma) es seleccionado para la reproducción según su *fitness*. Más precisamente, A_i será seleccionado con probabilidad $p_i = f_i/\sum(f_j)$. Después del reemplazo se espera que

  $$m(H,t+1) = m(H,t) * PS * f(H)/\sum(f_j)$$

  donde PS es el tamaño de la población y $f(H)$ es el *fitness* promedio de los cromosomas que representan al esquema H.

  Si consideramos a f_{avg} como $\sum(f_j)/PS$ entonces

  $$m(H,t+1) = m(H,t) * f(H)/f_{avg}$$

es decir, un esquema crece tanto como la relación entre su *fitness* promedio y el *fitness* de la población. Aquellos esquemas con valor promedio de *fitness* superior al *fitness* promedio de la población tendrán un número mayor de muestras en la próxima generación. Por otra parte, los esquemas con valor promedio de *fitness* inferior al *fitness* promedio de la población tendrán un número menor de muestras en la próxima generación.

- **Efecto del cruce:** sea A_i=0111000 un cromosoma y H_1=*1****0 y H_2=***10** dos esquemas representados por dicho cromosoma. Supongamos que realizamos cruce en un punto. Para el cruce en un punto sobre cromosomas binarios se genera una posición aleatoria para dos cromosomas padre y se intercambia el contenido genético entre esos padres a partir de dicha posición, generando los dos hijos correspondientes. En este caso, A es elegido para la reproducción con punto de corte = 3:

  - A_i= 0 1 1 | 1 0 0 0
  - H_1= * 1 * | * * * 0
  - H_2= * * * | 1 0 * *

  Se puede comprobar que H_2 estará presente en uno de los hijos, pero H_1 no, pues dicho esquema desaparece al partir. Para H_1=*1****0 tenemos que $\delta(H) = 5$. Si el punto de corte se selecciona uniformemente, entre las $L - 1 = 6$ posiciones posibles, siendo L la longitud del cromosoma, la probabilidad de que H_1 sea destruido es:

$$p_d = \delta(H_1)/(L-1) = 5/6$$

  y la probabilidad de que sobreviva es:

$$p_s = 1 - p_d = 1/6$$

  En general, la probabilidad de que un esquema H sobreviva, al producirse cruce, es:

$$p_s = 1 - \delta(H)/(L-1)$$

  Si denominamos p_c a la probabilidad de cruce, puede afirmarse que:

$$m(H, t+1) = m(H, t) * \frac{f(H)}{f_{avg}} * \left(1 - \frac{p_c * \delta(H)}{L-1}\right)$$

- **Efecto de la mutación:** considerando la mutación como la alteración de un gen de un cromosoma binario seleccionado aleatoriamente, tendremos que, si P_m es la probabilidad de mutación de un cromosoma y $p_m = P_m/L$ la probabilidad de mutación de un gen siendo L la longitud del cromosoma, la probabilidad de que el esquema sobreviva a la mutación será: $(1 - p_m)^{o(H)}$. Por lo tanto:

$$m(H, t+1) = m(H, t) * \frac{f(H)}{f_{avg}} * \left(1 - \frac{p_c * \delta(H)}{L-1}\right) * (1 - p_m)^{o(H)}$$

  Los esquemas cortos, de bajo orden, cuyo *fitness* está por encima del promedio reciben un incremento de representantes en las siguientes generaciones de un Algoritmo Genético.

2.4.7. Ventajas e inconvenientes

Los Algoritmos Genéticos se han aplicado a un amplio rango de problemas pertenecientes a campos tan diversos como la robótica, ingeniería, inteligencia artificial o economía debido a que presentan las siguientes características:

- Pueden resolver problemas difíciles de forma rápida y fiable.
- Aprenden a mantener o eliminar posibles soluciones en función de su calidad.
- Son ciegos, en el sentido de que no manejan ningún tipo de información sobre el problema concreto, exceptuando la función de evaluación.
- Aunque no hay garantía de encontrar la solución óptima, generalmente encuentran soluciones aceptables.
- Se pueden hibridar fácilmente.

Sin embargo, existe un problema grave asociado a los Algoritmos Genéticos, el problema de la *convergencia prematura* hacia zonas del espacio de búsqueda que no contienen el óptimo global. Esto se debe a que la base del

funcionamiento de los Algoritmos Genéticos reside en el mantenimiento del equilibrio entre explotar lo que actualmente es mejor (mediante el cruce) y explorar posibilidades que pueden convertirse en algo mucho mejor (haciendo uso de la mutación). Si este equilibrio se desproporciona, se llega a la pérdida de diversidad de la población y, consecuentemente se converge prematuramente. Para evitarlo se han planteado multitud de soluciones. Véanse, por ejemplo, [40], [95], [134].

Capítulo 3

Resolución de restricciones geométricas

En este Capítulo se procede a definir el problema de la resolución de restricciones geométricas. En primer lugar, se caracteriza y formaliza el problema de restricciones geométricas de forma global y en su forma básica como un conjunto de puntos y un conjunto de parámetros de restricción relativos a las coordenadas de tales puntos.

A continuación, se expone una taxonomía relativa a las diferentes técnicas de resolución de restricciones geométricas, así como una discusión acerca de sus capacidades y limitaciones. Tras ello, se presenta y formaliza la técnica de resolución de restricciones geométricas utilizada en este estudio: técnica constructiva, haciendo especial hincapié en la definición de problemas mediante grafos de restricciones.

Continuamos con la exposición del problema de la selección de la solución deseada, problema principal objeto de estudio, dentro de la resolución de restricciones y la formalización del mismo como problema de optimización combinatoria.

Finalmente, se caracteriza la aplicación de Metaheurísticas al problema de la selección de la solución deseada y se presentan los estudios previos realizados.

3.1. Problemas definidos mediante restricciones geométricas

Un problema de restricciones geométricas, [62], puede ser caracterizado a partir de una tupla (E, O, X, C), donde:

- E es el espacio geométrico que constituye el marco de referencia sobre el que se instancia el problema. E es normalmente euclídeo.
- O es el conjunto de objetos geométricos específicos que definen el problema. Se seleccionan de un repertorio fijo que incluye puntos, líneas, círculos y elementos geométricos afines.
- X es un conjunto, posiblemente vacío, de variables cuyo valor ha de ser determinado. En general, tales variables representan cantidades con significado geométrico: distancias, ángulos, etc., Cuando dichas cantidades no tienen significado geométrico, por ejemplo, cuando cuantifican aspectos tecnológicos o capacidades funcionales, tales variables son denominadas externas.
- C es el conjunto de restricciones. Las restricciones pueden ser geométricas o ecuacionales. Las restricciones geométricas son relaciones entre elementos geométricos seleccionadas de un conjunto predefinido, e.g., distancia, ángulo, tangencia, etc., La relación (la distancia, el ángulo, etc.,) se representa mediante un parámetro. Si el parámetro representa un valor fijo, conocido previamente, entonces la restricción es denominada valuada. Si, en cambio, el parámetro representa un valor que requiere ser calculado como parte de la resolución del problema de restricciones, la restricción es denominada simbólica, [61].

  Las restricciones ecuacionales son ecuaciones en las que algunas de las variables son parámetros de restricciones simbólicas. El conjunto de restricciones ecuacionales puede ser vacío.

El problema de resolución de restricciones geométricas puede ahora ser planteado como sigue:

> Dado un problema de restricciones geométricas (E, O, X, C),
>
> 1. ¿Permite la disposición espacial entre sí de los elementos geométricos de O que se cumplan las restricciones geométricas definidas en C?. Si la respuesta es afirmativa, entonces

2. dada una asignación de valores para las restricciones valuadas y las variables externas, ¿existe una construcción posible que satisfaga las restricciones y ecuaciones?.

Cuando se aborda la resolución de restricciones geométricas, la primera cuestión que es necesario decidir es la dimensión del espacio subyacente E. En nuestro caso, $E = \mathbb{R}^2$.

3.1.1. El problema básico

El problema básico de restricciones solamente considera elementos geométricos y restricciones cuyos parámetros tienen un valor asignado. Las variables externas, las restricciones cuyos parámetros deben ser calculados y las restricciones ecuacionales son excluidas. De esta forma, el problema básico se define como sigue:

> Dado un conjunto O constituido por n elementos geométricos y un conjunto C con m restricciones geométricas definidas sobre tales elementos geométricos,
>
> 1. ¿Existe alguna disposición espacial de los n elementos geométricos de tal forma que las m restricciones se satisfagan? Si la respuesta es afirmativa,
> 2. dada una asignación de valores a los m parámetros de las restricciones, ¿existe una construcción posible para los n elementos geométricos de tal forma que las restricciones se satisfagan?

De aquí en adelante, trataremos únicamente con el problema básico de resolución de restricciones geométricas.

En el diseño geométrico bidimensional basado en restricciones, el diseñador crea un croquis de un objeto geométrico mediante una colección de elementos geométricos simples como puntos, segmentos y círculos con radio conocido, y un conjunto de relaciones geométricas entre tales elementos geométricos, que son las restricciones del problema. La forma exacta del objeto requerida por el usuario es especificada a partir de la incorporación de restricciones geométricas.

Entre los tipos de restricciones geométricas que se consideran están, entre otras, la distancia entre puntos, la distancia de un punto a un segmento, el

ángulo entre dos segmentos, la coincidencia de un punto sobre un segmento y la tangencia de un segmento a un círculo.

Desde el punto de vista matemático, los segmentos y círculos se representan mediante conjuntos de puntos: un segmento se representa mediante sus puntos finales y un círculo se representa mediante los dos puntos finales del radio y el punto central del círculo que soporta. Todas las restricciones geométricas consideradas se pueden transformar en las siguientes relaciones, [111]: distancia entre dos puntos, distancia de un punto a la línea definida mediante el segmento y ángulo entre dos segmentos.

Por lo tanto, un problema geométrico se define mediante un conjunto de puntos $\mathcal{P} = \{p_1, \ldots, p_n\}$ y un conjunto finito de parámetros de las restricciones geométricas $\mathcal{C} = \{c_1, \ldots, c_m\}$. Cada parámetro c_i es una distancia o un ángulo. Se supone que el conjunto de restricciones geométricas define correctamente el objeto bajo diseño y que el objeto está, generalmente, bien restringido, [49].

El croquis de la Figura 3.1 corresponde a un mecanismo definido como un problema de restricciones geométricas. Este mecanismo, conocido como mecanismo de Peaucellier, [31], transforma el movimiento circular del punto p_3 alrededor del círculo c_1, en la traslación del punto p_2 a lo largo de la línea recta l_3. Como un problema de restricciones geométricas, el mecanismo de Peaucellier incluye seis puntos, p_i, $1 \leq i \leq 6$, y dos segmentos l_1, l_2. El conjunto de restricciones geométricas correspondiente se lista en la Figura 3.2 e incluye las restricciones de distancia entre dos puntos, $dpp()$, coincidencia, $on()$, y ángulo entre dos segmentos, $angle()$.

El conjunto inicial de restricciones sobre los puntos se transforma de forma tal que los parámetros de las restricciones son sólo distancias y ángulos. Para el ejemplo de la Figura 3.1, el conjunto de parámetros de las restricciones es, por tanto, el conjunto $\mathcal{C} = \{d_1, d_2, d_3, d_4, d_5, d_6, d_7, a_1\}$, conformado por siete distancias y un ángulo.

Resolver el problema de la Figura 3.1 consiste en calcular las coordenadas de los puntos $\mathcal{P} = \{p_1, p_2, p_3, p_4, p_5, p_6\}$ con los datos geométricos de entrada $\mathcal{C} = \{d_1, d_2, d_3, d_4, d_5, d_6, d_7, a_1\}$. Los segmentos y círculos en el croquis inicial pueden obtenerse fácilmente de las coordenadas de los puntos del objeto.

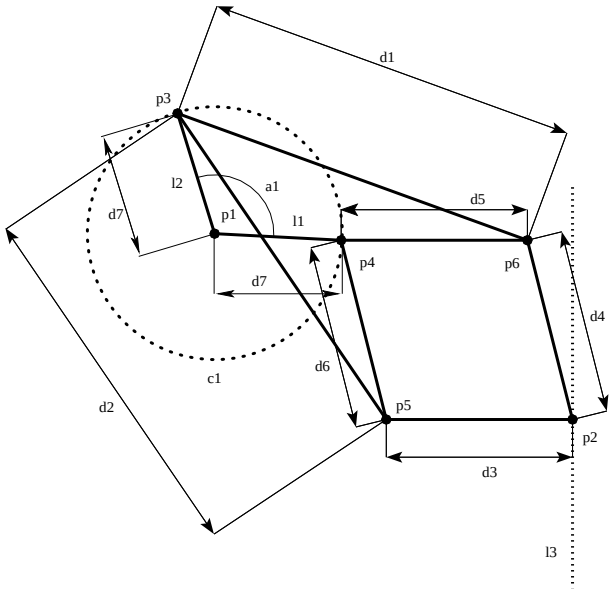


Figura 3.1: Mecanismo de Peaucellier: problema geométrico definido mediante restricciones.

1.	dpp(p_3, p_6, d_1)	8.	dpp(p_3, p_1, d_7)
2.	dpp(p_3, p_5, d_2)	9.	on(p_1, l_1)
3.	dpp(p_2, p_5, d_3)	10.	on(p_4, l_1)
4.	dpp(p_2, p_6, d_4)	11.	on(p_1, l_2)
5.	dpp(p_4, p_6, d_5)	12.	on(p_3, l_2)
6.	dpp(p_4, p_5, d_6)	13.	angle(l_1, l_2, a_1)
7.	dpp(p_1, p_4, d_7)		

Figura 3.2: Restricciones geométricas para el mecanismo de Peaucellier.

3.2. Técnicas de resolución de restricciones geométricas

En general, la labor de un sistema de resolución de restricciones geométricas, en adelante *solver*, [62], es la antítesis de la eficiencia puesto que la resolución de restricciones tiene una complejidad doblemente exponencial. Los problemas pueden parecer simples pero su resolución supone tratar con problemas matemáticos complejos.

Las principales aproximaciones a la resolución de restricciones geométricas pueden ser clasificadas como métodos basados en grafos, métodos basados

en la lógica o métodos algebraicos. Para los sistemas bidimensionales de resolución de restricciones geométricas, la aproximación basada en grafos ha llegado a ser la dominante en el campo del Diseño Asistido por Computador.

3.2.1. Aproximación basada en grafos

En la aproximación basada en grafos, [62], el problema de restricciones es representado a partir de un grafo (o hipergrafo) cuyos vértices representan los elementos geométricos y cuyos lados representan las restricciones sobre los mismos. El *solver* analiza el grafo y formula una estrategia de solución mediante el aislamiento de subproblemas y la combinación adecuada de sus soluciones. Una fase posterior resuelve todos los subproblemas y los combina. La ventaja de este tipo de *solver* es que los subproblemas son, con frecuencia, muy pequeños y se corresponden con un conjunto reducido de categorías simples. El inconveniente viene dado por la gran dificultad del análisis del grafo por parte de un *solver* completamente solvente.

Los métodos basados en grafos se dividen en métodos constructivos, métodos basados en análisis de grados de libertad y métodos de propagación.

La aproximación constructiva genera la solución a un problema de restricciones geométricas como una secuencia simbólica de pasos básicos de construcción. Cada paso consiste en una regla, perteneciente a un conjunto predefinido de operaciones, que posiciona un subconjunto de elementos geométricos. Por ejemplo, las operaciones pueden restringirse a construcciones de regla y compás. Claramente, esta propuesta preserva el sentido geométrico de cada operación que interviene en la solución. Nótese que la secuencia de pasos de construcción permite representar de forma compacta un número posiblemente exponencial de instancias solución. Sin embargo, la aproximación constructiva no puede resolver problemas que consideren restricciones simbólicas o variables externas. Los métodos constructivos producen buenos resultados en el espacio bidimensional. La técnica de resolución de restricciones geométricas en la que nos centraremos en este trabajo será precisamente la técnica constructiva, explicada con más detalle en la Sección 3.3.

El principio del análisis de los grados de libertad consiste en que, dado un objeto, cada elemento geométrico y cada restricción geométrica tiene un cierto número de grados de libertad. El análisis de grados de libertad asigna grados de libertad a los elementos geométricos a partir del etiquetado de los vértices del grafo correspondiente al problema. Cada lado del grafo es etiquetado con el número de grados de libertad anulados por la restricción asociada.

Después, el método resuelve el problema a partir del análisis del grafo etiquetado resultante. Los elementos geométricos incrementan el número de grados de libertad del objeto y las restricciones geométricas los reducen. En [13], [91], [96], [133], las restricciones se satisfacen incrementalmente hasta que el objeto no posee más grados de libertad. En ese momento el objeto es rígido y, por lo tanto, quedan fijadas las coordenadas de los puntos. Por otra parte, Hoffman *et al.* , [65], [66], han desarrollado un método basado en flujos para descomponer grafos de problemas de restricciones geométricas. Se trata de un método iterativo para obtener una descomposición del sistema algebraico subyacente en subsistemas reducidos llamados grafos de densidad mínima.

Los métodos de propagación representan el conjunto de ecuaciones algebraicas con un grafo simétrico cuyos vértices son variables y ecuaciones, y cuyos lados son etiquetados con las ocurrencias de las variables en las ecuaciones. Los métodos de propagación tratan de orientar los lados en el grafo de tal forma que cada vértice ecuación tiene como grado de entrada el número de lados incidentes en el mismo excepto uno. Si el proceso tiene éxito, entonces existe una solución incremental general. Se intenta aprovechar la estructura triangular del sistema de ecuaciones para resolver simultáneamente más de una ecuación. Los métodos de propagación no garantizan encontrar una solución siempre que ésta exista. Para más información sobre este tipo de métodos véanse [62], [92], [98], [141], [150].

3.2.2. Aproximación basada en la lógica

En la aproximación basada en la lógica, el problema se traduce a un conjunto de aserciones y axiomas que caracterizan las restricciones y los elementos geométricos. A partir de los pasos del razonamiento lógico, tales aserciones se transforman en estados que exponen los pasos solución de forma estereotípica. Tras ello, un tipo específico de *solver* se encarga de calcular las asignaciones de coordenadas.

Aldefeld, [4], Brüderlin, [24], Sohrt *et al.* , [140] y Yamaguchi *et al.* , [158], usan lógica de primer orden para derivar información geométrica aplicando un conjunto de axiomas de la geometría de Hilbert. Tales métodos proporcionan, esencialmente, lugares geométricos en los que deben estar los elementos.

Sunde, [145], y Verroust, [151], [152], consideran dos tipos diferentes de conjuntos de restricciones: conjuntos de puntos ubicados con respecto a un marco de trabajo local, y conjuntos de segmentos cuyas direcciones son fijas con

respecto a un marco de trabajo local. El razonamiento es llevado a cabo, básicamente, a partir de un sistema de reescritura sobre los conjuntos de restricciones. El problema se resuelve cuando todos los elementos geométricos pertenecen a un único conjunto. Joan-Arinyo y Soto-Riera, [79], [80], extendieron dichos conjuntos de restricciones con un tercer tipo consistente en conjuntos que contienen un punto y una línea recta de tal forma que la distancia perpendicular punto-línea es fija.

3.2.3. Métodos algebraicos

Para la aproximación algebraica, el problema de restricciones se traduce directamente a un sistema de ecuaciones no lineales y se resuelve a partir de alguno de los métodos existentes de resolución de ecuaciones no lineales. Cada restricción geométrica se traslada a una ecuación algebraica, lineal o cuadrática, sobre las coordenadas de los puntos del objeto. Las principales ventajas de los *solvers* algebraicos son su generalidad, su independencia de la dimensión y su capacidad para tratar las restricciones simbólicas de forma natural.

En la traducción comentada se pierde el sentido geométrico del problema. Es más, un problema bien restringido da lugar a un sistema de ecuaciones infradeterminado, ya que las restricciones permiten calcular las posiciones relativas de los elementos geométricos y, por lo tanto, en el caso bidimensional, quedan por fijar tres grados de libertad, dos de los cuales corresponden a una traslación y uno a una rotación, de todo el conjunto de elementos geométricos.

En función de las técnicas específicas de resolución de los métodos algebraicos, éstos pueden dividirse en métodos numéricos, métodos simbólicos y métodos basados en el análisis de sistemas de ecuaciones. En los métodos numéricos se utilizan métodos iterativos de resolución de sistemas de ecuaciones, donde la garantía de convergencia del método es fundamental. Los métodos simbólicos calculan una base de Gröbner para el sistema de ecuaciones de partida. Los métodos basados en el análisis de sistemas de ecuaciones determinan si el sistema es infradeterminado, sobredeterminado o bien determinado a partir de la estructura del sistema. Estos métodos pueden ser extendidos para descomponer los sistemas de ecuaciones en un conjunto de grafos minimales que se pueden resolver con independencia.

Para una mayor discusión sobre los métodos algebraicos véanse [62], [107], [112].

3.3. Sistemas constructivos

Las técnicas constructivas buscan la resolución de una subclase bien definida de problemas geométricos definidos en base a restricciones: específicamente, en nuestro caso, los resolubles con regla y compás, y permiten obtener una solución exacta expresada como una secuencia de operaciones geométricas básicas mediante intersecciones entre rectas y círculos. Estas técnicas conservan el sentido geométrico al obtener una solución constructiva al problema.

La secuencia simbólica de pasos constructivos se conoce normalmente como *plan de construcción*. Por regla general, dado un problema geométrico existe más de un plan de construcción que lo resuelve.

Las técnicas constructivas pueden ser clasificadas en función de la dirección en la cual se analiza el problema. Las técnicas de *composición* son las que analizan el problema de abajo hacia arriba, componiendo una solución del problema a partir del análisis de las restricciones. En cambio, las de *descomposición* abordan el problema de arriba hacia abajo, analizando las interrelaciones entre las restricciones para descomponer el problema.

Las restricciones entre elementos geométricos se pueden representar implícita o explícitamente en las técnicas constructivas de composición. La representación implícita normalmente se basa en un conjunto de restricciones. Los conjuntos de restricciones son conjuntos de elementos geométricos de los que se conocen sus posiciones relativas. Mediante la aplicación de reglas de reescritura se fusionan los conjuntos de restricciones hasta obtener uno solo con todos los elementos geométricos. La principal ventaja del uso de conjuntos de restricciones es que se obtiene una representación más clara de las relaciones entre los elementos que los componen. Fudos, [18], [46], [47], [49], utiliza un único tipo de conjuntos de restricciones que denomina *clusters*, y dispone de una única regla genérica que permite fusionar tres clusters cuando comparten un elemento dos a dos. Sunde, [145], y Verroust, [151], [152], usan dos tipos de conjuntos de restricciones: conjuntos de puntos con sus posiciones relativas conocidas y conjuntos de segmentos con las direcciones fijadas. Joan-Arinyo y Soto-Riera, [79], [80], extendieron dichos conjuntos de restricciones con un tercer tipo.

Cuando se representan de forma explícita las restricciones, normalmente se hace mediante predicados de lógica de primer orden. Brüderlin et al, [22], [23], [24], [139], [140], simplifican estos predicados según los axiomas de la geometría de Hilbert hasta calcular, simbólicamente, las posiciones relativas

de los elementos geométricos.

Algunos de los principales sistemas constructivos son, por ejemplo, [18], [49], [97], [121] y [148]. Un sistema constructivo de resolución de restricciones geométricas, *solver* constructivo, está constituido por dos componentes principales: el *analizador* y el *constructor*. El analizador determina simbólicamente si es resoluble el problema geométrico definido mediante restricciones. En caso de que el problema sea resoluble, la salida del analizador es una secuencia de pasos de construcción, conocida como el plan de construcción, que describe cómo disponer cada elemento geométrico de tal forma que todas las restricciones sean satisfechas. Después de la asignación de valores específicos a los parámetros de las restricciones, el constructor interpreta el plan de construcción y construye una instancia del objeto, suponiendo que no aparezcan incompatibilidades numéricas, p. ej. el cálculo de una raíz cuadrada de un número negativo.

Una vez que el usuario ha definido el croquis con los elementos geométricos y el conjunto de restricciones, el *solver* comprueba si tales restricciones geométricas definen coherentemente al objeto y, en su caso, determina la posición de los elementos geométricos aplicando un plan de construcción.

El plan de construcción específico generado por un analizador depende de la técnica constructiva subyacente y de cómo ésta se ha implementado. Por ejemplo, la aproximación constructiva basada en regla y compás es una técnica bien conocida donde cada paso constructivo del plan se corresponde con una operación básica resoluble mediante una regla, un compás y un semicírculo graduado. En la práctica, esta simple aproximación resuelve la mayoría de problemas geométricos útiles.

La Figura 3.3 presenta un plan de construcción para el objeto de la Figura 3.1, generado por el *solver* basado en regla y compás descrito en [81].

1.	$p_1 = pointXY(0,0)$	9.	$p_5 = icc(x_2, x_3)$
2.	$p_3 = pointXY(0,d_7)$	10.	$x_4 = circleCR(p_3, d_1)$
3.	$l_2 = line2P(p_1, p_3)$	11.	$x_5 = circleCR(p_4, d_5)$
4.	$l_1 = lineAP(l_2, p_1, -a_1)$	12.	$p_6 = icc(x_4, x_5)$
5.	$x_1 = circleCR(p_1, d_7)$	13.	$x_6 = circleCR(p_5, d_3)$
6.	$p_4 = ilc(l_1, x_1)$	14.	$x_7 = circleCR(p_6, d_4)$
7.	$x_2 = circleCR(p_3, d_2)$	15.	$p_2 = icc(x_6, x_7)$
8.	$x_3 = circleCR(p_4, d_6)$		

Figura 3.3: Plan de construcción para el mecanismo de Peaucellier.

Los nombres de las funciones del plan de construcción de la Figura 3.3 son

autoexplicativos. Por ejemplo, la función $line2P()$ define una línea recta entre dos puntos, $circleCR()$ define un círculo a partir de su centro y su radio, y las funciones $ilc()$, $icc()$ denotan una intersección línea-círculo y una intersección círculo-círculo respectivamente.

A continuación, se describe el mecanismo de representación basado en grafos utilizado en los sistemas constructivos de resolución de restricciones geométricas. Tras ello, se presenta una posible formalización del problema constructivo de resolución de restricciones geométricas.

3.3.1. Grafos de restricciones

Como ya ha sido indicado en la Sección 3.2, en la aproximación basada en grafos, los sistemas para la resolución de restricciones geométricas traducen inicialmente el problema tras el análisis de una estructura de *grafo*. A continuación, detallaremos en qué consiste esta estructura y lo que representa.

En un *grafo de restricciones*, [62], el conjunto de vértices se corresponde con los elementos geométricos del problema. A cada vértice se le atribuye un peso que identifica su grado de libertad, usualmente el número de coordenadas independientes que son necesarias para ubicar el elemento geométrico representado. En el caso de puntos y líneas no restringidas en el plano, tal grado sería 2. Para los círculos (sin radio preestablecido) sería 3. Si para una línea se ha establecido una restricción de horizontalidad o verticalidad, el peso sería 1.

Las restricciones geométricas se representan a partir de los lados del grafo. Éstas incluyen restricciones dimensionales (ángulo, distancia, etc.,) y restricciones lógicas (incidencia, concentricidad, etc.,). Algunas restricciones pueden ser inferidas por el propio sistema de resolución, tales como las de incidencia, y asimismo representadas en el grafo de restricciones. Además, se pueden aplicar ciertas transformaciones para cambiar internamente el problema de restricciones. Por ejemplo, un círculo con radio preestablecido puede ser reemplazado con su centro tras cambiar las restricciones definidas sobre el círculo a restricciones equivalentes definidas sobre el centro del círculo. Los lados son etiquetados con el número de grados de libertad que suprimen, usualmente, el número de ecuaciones independientes. Por ejemplo, una restricción de distancia entre dos puntos suprimiría un grado de libertad y, sin embargo, una restricción de incidencia entre dos puntos suprimiría dos grados de libertad.

Se puede llevar a cabo un análisis global para averiguar si un problema está bien restringido a partir de la suma de los pesos de los vértices y la sustracción posterior de la suma de los pesos de los lados. Véase Laman, [93]. El número resultante es conocido como la *deficiencia* del problema. Para problemas en los que la solución no está en una posición fija con respecto a un sistema de coordenadas global, un problema bidimensional bien restringido debería presentar una deficiencia de 3, e. g. [46]. Cuando el problema se sitúa con respecto a un sistema de coordenadas fijo se requeriría una deficiencia nula.

Un subgrafo inducido se puede resolver si corresponde a un subproblema bien restringido. El hallazgo de un grafo minimal supone un problema mínimamente complicado de resolver. Tal problema se denomina *núcleo del cluster*. Una vez que se ha resuelto un subproblema, tal subproblema puede ser ampliado secuencialmente con elementos geométricos: si el elemento geométrico tiene peso k y está restringido con respecto a elementos geométricos del subproblema donde los pesos de los lados suman hasta k, podemos añadir el nuevo elemento como una extensión secuencial del *cluster*.

Mediante el análisis del grafo de restricciones de esta forma, éste puede descomponerse en un conjunto de *clusters* que se pueden resolver individualmente. La combinación de varios *clusters* se puede realizar cuando exista solapamiento dos a dos en un elemento geométrico. Aquí, se infieren restricciones entre los elementos compartidos a partir de los subproblemas resueltos. Los elementos compartidos pueden ser ubicados con respecto a otros elementos compartidos y conformar un esqueleto en el que ubicar los *clusters* resueltos. En el espacio bidimensional, los sistemas de resolución de restricciones emplean descomposición basada en triángulos, por lo que combinar tres *clusters* es un paso natural, [18], [46]. Los *clusters* se pueden añadir individualmente de forma análoga a la de una extensión secuencial.

La *descomposición basada en triángulos* utiliza únicamente núcleos de *cluster* de dos elementos y combina tres *clusters* que comparten dos a dos un elemento geométrico. Tales *solvers* bidimensionales tienen gran interés debido a su competencia en aplicaciones de Diseño Asistido por Computador y a su simplicidad conceptual. Hay que tener en cuenta que requieren la resolución de ecuaciones univariadas a lo sumo cuadráticas, [48].

Si el grafo de restricciones completo es descomponible recursivamente hasta obtener un único *cluster*, entonces la fase de planificación tiene éxito y es posible obtener, en principio, una solución para el problema de restricciones. Si la descomposición falla, el *solver* notificará que no se puede encontrar solución alguna. Esto último significa que no existe solución basada en la

descomposición mediante triángulos, pero en ningún caso que no exista solución. Se puede percibir aquí la diferencia entre una estrategia determinada y un *solver* completamente general. Existe un algoritmo conocido que obtiene resultados en tiempo polinomial para la descomposición de cualquier grafo que corresponda a un problema de restricciones resoluble. Sin embargo, es más complejo y la subsiguiente fase del *solver* es también más exigente porque no puede existir *a priori* una restricción sobre el tamaño de los núcleos de cluster minimales, [63], [64].

Usualmente, el grafo se puede descomponer de varias formas, enfatizando la cuestión de si se requiere un orden canónico de descomposición para encontrar una solución si ésta existe. Existen teoremas que prueban que si un grafo de restricciones puede ser descompuesto de una forma determinada, a partir de la descomposición basada en triángulos, entonces cada secuencia de descomposiciones de triángulos descompone completamente el grafo de restricciones. La demostración está basada en la propiedad Church-Rosser de reducción de grafos a partir de la fusión de subgrafos resolubles en nodos simples, [48]. Tales teoremas se pueden generalizar y, además, el orden de descomposición no es crítico para determinar la resolubilidad, [65], [66].

La exploración del conjunto de grafos bien restringidos, [49], [153], permite determinar la proporción en la que podemos descomponer los elementos de esta familia. Los grafos bien restringidos tienen $2 \times |V| - 3$ aristas, siendo V el número de vértices. Una de las posibles aproximaciones para la generación de grafos bien restringidos puede tener como base la utilización de las *secuencias de Henneberg*, [17]. La construcción de grafos a partir de estas secuencias es un método inductivo. La secuencia de Henneberg para un grafo G es la secuencia $G_3, G_4, \ldots, G_{ng}$, donde G_3 es un triángulo, $G_{ng} = G$ y cada grafo G_{i+1} se obtiene a partir del grafo G_i aplicando los pasos *tipo-I* o *tipo-II*. El paso *tipo-I* consiste en añadir un vértice y unirlo al grafo mediante aristas hacia dos vértices aleatorios. El paso *tipo-II* consiste en eliminar una arista cualquiera y sustituirla por un vértice, añadiendo finalmente una nueva arista hacia un vértice aleatorio. El proceso acaba cuando se obtiene un grafo de un tamaño determinado.

La exploración del conjunto de grafos bien restringidos no puede ser sistemática, puesto que supondría una explosión combinatoria que sólo haría abordable tratar con un conjunto muy reducido de vértices. De esta forma, para poder tratar con grafos de mayor tamaño se pueden utilizar las secuencias de Henneberg que pueden permitir generar grafos aleatorios pseudo-equiprobables. Está claro que sería mejor disponer de un método que pro-

porcionase grafos de forma equiprobable, pero es imposible debido a los *isomorfismos*, [146]. Detectar cuando dos grafos son isomorfos es un problema más costoso. El problema del isomorfismo es uno de los problemas relevantes para los que la categorización de su tiempo de resolución está abierta entre el tiempo polinomial y la NP-completitud, [51], [103].

3.3.2. Formalización constructiva del problema

Es sabido que la posición relativa de n puntos $\{p_1, \ldots, p_n\}$ en el espacio bidimensional euclídeo queda determinada por $2n-3$ relaciones independientes definidas entre los puntos, [93], [102]. Según esto, el problema de restricciones geométricas en el espacio euclídeo se puede formalizar como se demuestra a continuación.

Sea $\mathcal{P}$ un conjunto de n puntos sobre el que se han definido $2n-3$ restricciones. Se asume que $\mathcal{P}$ se divide en dos subconjuntos disjuntos no vacíos: un subconjunto, $\overrightarrow{p}' = \{p'_1, \ldots, p'_k\}$, que contiene todos los puntos cuya posición es fija y un segundo subconjunto, $\overrightarrow{p} = \{p_1, \ldots, p_l\}$, que contiene todos aquellos cuya posición es desconocida. Esta descomposición siempre es posible porque $2n-3$ relaciones independientes entre n puntos definen un objeto rígido con tres grados de libertad, dos de los cuales se corresponden con una traslación y el tercero con una rotación. Por lo tanto, debe especificarse la posición absoluta de, al menos, uno de los puntos.

Según Brüderlin, [22], el conjunto de restricciones, junto con la conjunción, disyunción y negación lógicas, nos permite expresar el problema geométrico como una fórmula de lógica de primer orden $\varphi(\overrightarrow{p}', p_1, \ldots, p_l)$ tal que si el conjunto de restricciones define un problema *bien restringido*, [47], [81], la fórmula

$$\exists p_1 \ldots \exists p_l \quad \varphi(\overrightarrow{p}', p_1, \ldots, p_l) \tag{3.1}$$

se cumple. Por el *axioma de elección*, [22], [87], podemos decir que siempre que la expresión de la Ecuación 3.1 sea cierta, la fórmula

$$\exists f_1 \ldots \exists f_l \quad \varphi(\overrightarrow{p}', f_1(\overrightarrow{p}'), \ldots, f_l(\overrightarrow{p}')) \tag{3.2}$$

también lo será. Por lo tanto, el objetivo en la resolución de un problema geométrico basado en restricciones es probar la verdad de la Ecuación 3.2 y evaluar las funciones $f_1, \ldots, f_l$.

El enfoque en un sistema de resolución de restricciones geométricas es tal que, dado un problema geométrico $\varphi(\overrightarrow{p}', p_1, \ldots, p_l)$ definido sobre el conjunto de puntos $\{p_1, \ldots, p_n\}$, necesitamos buscar una fórmula en lógica de primer orden *constructiva*, $\Psi(\overrightarrow{p}', p_1, \ldots, p_l)$, tal que si el problema geométrico está bien restringido, nos proporcionará la posición relativa de cada uno de los puntos. Si el predicado $pos(p_i, (x_i, y_i))$ asigna la posición (x_i, y_i) al punto p_i, un ejemplo de fórmula constructiva sería, [22],

$$\begin{array}{l} \Psi(\overrightarrow{p}', p_1, \ldots, p_l) = \\ pos(p'_1, (x_1, y_1)) \wedge \ldots \wedge pos(p'_k, (x_k, y_k)) \\ \wedge pos(p_1, (fx_1(\overrightarrow{p}'), fy_1(\overrightarrow{p}'))) \\ \bigwedge_{i=2}^{l} pos(p_i, (fx_i(\overrightarrow{p}', p_1, \ldots, p_{i-1}), fy_i(\overrightarrow{p}', p_1, \ldots, p_{i-1}))) \end{array} \tag{3.3}$$

Cada paso de construcción del plan generado por el analizador corresponde a un par de funciones (fx_i, fy_i). Como ejemplo, las funciones (fx_i, fy_i) para la fórmula lógica de primer orden del ejemplo de la Figura 3.1, $\Psi(p_1, \ldots, p_6)$, se obtienen del plan de construcción de la Figura 3.3 mediante el reemplazamiento de los símbolos a la izquierda del signo igual distintos de p_i con sus definiciones. Por ejemplo, en l_1, paso 4, se reemplazaría l_2 con $line2P(p_1, p_3)$, sustituyendo la expresión completa correspondiente a l_1 en pasos sucesivos, por ej. para obtener p_4.

3.4. El problema de la selección de la solución deseada

Con independencia del enfoque, basado en grafos, lógico o algebraico, por regla general un sistema con restricciones no define un único objeto, [49], [81], [93].

Cuando existen infinitas soluciones que satisfacen las restricciones, se dice que el sistema está *infrarrestringido*. Cuando el número de restricciones es tal que no puede asegurarse que exista una solución que satisfaga todas las restricciones, se dice que el sistema está *sobrerrestringido*. Cuando el conjunto de soluciones es finito y no vacío, el sistema está bien restringido, [81].

En el caso de un sistema bien restringido la exploración del espacio de soluciones no es un problema trivial. La existencia de ecuaciones polinómicas cuyo grado es igual o superior a dos, desde un punto de vista algebraico, o de múltiples intersecciones, desde un punto de vista geométrico, provoca una

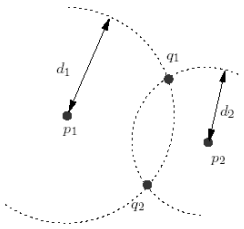


Figura 3.4: Posibles localizaciones de un punto.

explosión combinatoria en el número de soluciones.

Fudos y Hoffmann, [49], demostraron que un problema de restricciones geométricas bien restringido tiene, en general, un número exponencial de soluciones con respecto al número de elementos geométricos del problema. Por ejemplo, considérese un problema de restricciones geométricas que posiciona adecuadamente n puntos. Asumimos que los puntos pueden ser ubicados secuencialmente, determinándose cada vez el siguiente punto a partir de dos distancias entre dos puntos ya ubicados. Una vez posicionados los dos primeros puntos, en general, cada nuevo punto puede tener dos localizaciones diferentes: las que corresponderían a los puntos de intersección entre dos círculos. Véase la Figura 3.4. Para n puntos, por lo tanto, podríamos tener hasta 2^{n-2} soluciones.

Las posibles localizaciones diferentes de los elementos geométricos, correspondientes a las distintas raíces de los sistemas no lineales de ecuaciones algebraicas, pueden distinguirse a partir de la enumeración de las raíces con un índice entero. Para una definición formal véanse [41], [107].

En lo que sigue, asumiremos que el conjunto de restricciones geométricas define coherentemente el objeto bajo diseño, es decir, el objeto está generalmente bien restringido y que se dispone de un *solver* constructivo basado en regla y compás como el descrito en [81].

En dicho *solver*, las operaciones de intersección en las que intervienen los círculos, *ilc* y *icc*, véase Figura 3.3, pueden determinar dos puntos de intersección diferentes, dependiendo de si la ecuación de segundo grado a resolver no tiene solución, o bien tiene una o dos soluciones en el dominio real. Con cada operación factible *ilc* y *icc*, el constructor del *solver* asocia un parámetro entero $s_k \in \{-1, 1\}$, que caracteriza cada punto de intersección a partir del signo de la raíz cuadrada de la correspondiente ecuación cuadrática. Para

el plan de construcción de la Figura 3.3, el conjunto de parámetros enteros incluye $s_k, 1 \leq k \leq 4$. Nótese que si se asigna un conjunto de valores a los parámetros de las restricciones, $\mathcal{C} = \{d_1, d_2, d_3, d_4, d_5, d_6, d_7, a_1\}$, entonces el plan de construcción define el espacio de instancias solución en función de s_k. Para consultar detalles acerca del cálculo de s_k, véase [112].

A partir de un croquis diseñado, el usuario del sistema de Diseño Asistido por Computador normalmente espera obtener una única solución. Teniendo en cuenta lo indicado por Bouma *et al.* en [18], el problema de la selección de una solución para el problema de la resolución de restricciones geométricas es conocido como **el problema de la selección de la solución deseada** (*Root Identification Problem*). La selección de una instancia solución equivale a la selección de una raíz entre las disponibles para una ecuación o sistema de ecuaciones no lineal.

El principal problema cuando se desarrolla un sistema de resolución de restricciones geométricas es identificar la instancia solución que resulta más parecida a la que el usuario espera, [18], [49], [94]. Esta instancia solución es conocida como la *solución deseada*. La dificultad estriba en definir con precisión el concepto de solución deseada. Hasta el momento no se ha dado una definición aceptable del propio problema, es decir, no se sabe qué es aquello que espera el usuario y, por lo tanto, no se sabe cómo caracterizar la solución deseada. Generalmente, el usuario está solamente interesado en una instancia y no cualquiera de las pertenecientes al conjunto de soluciones. Tal instancia, además de satisfacer las restricciones geométricas, ha de exhibir algunas propiedades adicionales.

El problema de la selección de la solución deseada ha sido clasificado por Fudos *et al.* , [49], como NP-completo, por lo que si las heurísticas que buscan la solución deseada en base a algún criterio de semejanza fallan, no es viable buscar una solución real. Lo habitual en esta situación es ofrecer al usuario un sistema de navegación por el árbol de soluciones para que sea él quien seleccione la solución que desea. La navegabilidad y, en definitiva, la viabilidad es más dificil en la medida en que aumenta el tamaño del problema geométrico.

En la literatura se han definido varias aproximaciones para la resolución del problema de la selección de la solución deseada. Algunos ejemplos son los siguientes: movimiento selectivo de los elementos geométricos, identificación interactiva de la solución deseada mediante la conducción de un diálogo con el *solver* de restricciones, y preservación de la topología del croquis original proporcionado por el usuario. Para consultar y comparar tales aproxima-

ciones véanse, por ejemplo, Bouma *et al.* [18], y Luzón [107], así como las referencias proporcionadas en los mismos.

Es importante tener en cuenta tanto la gran dificultad para resolver el problema, [49], como la naturaleza interactiva del Diseño y Manufactura Asistidos por Computador en cuanto a limitación temporal para la obtención de la solución, [81], [153].

3.4.1. Definición de restricciones adicionales

En la literatura, [12], [107], [109], ha sido propuesta una nueva técnica para la resolución automática del problema de la selección de la solución deseada para *solvers* constructivos.

La mencionada técnica sobrerrestringe el problema geométrico con la definición de dos categorías diferentes de restricciones. Una de ellas incluye el conjunto de restricciones necesario específicamente para resolver el problema de restricciones geométricas. La otra categoría incluye un conjunto extra de predicados o restricciones sobre los elementos geométricos que permiten identificar la solución deseada por el usuario. Una vez que el *solver* constructivo ha generado el espacio de instancias solución, definido como un plan de construcción, las restricciones adicionales se usan como guía para la realización de una búsqueda automática sobre tal espacio de soluciones. El resultado de la búsqueda es la obtención de una instancia solución que maximiza el número de predicados adicionales que se verifican.

En este contexto, la resolución del problema de la selección de la solución deseada equivale a resolver un problema de satisfacción de restricciones general expresado como un problema de optimización combinatoria.

A continuación, presentaremos la formalización del problema de la selección de la solución deseada como un problema de optimización combinatoria. Recordemos que consideramos resolución de restricciones geométricas constructiva basada en regla y compás, donde las operaciones geométricas se corresponden con ecuaciones cuadráticas y de esta forma, cada paso constructivo tiene como máximo dos raíces diferentes.

Denotaremos por s_j al parámetro entero que el *solver* asocia a la j-ésima operación de intersección, bien *ilc* o *icc*, que aparece en el plan de construcción. Puesto que el interés reside en instancias solución que son realmente factibles, es decir, instancias solución para las que no aparezcan incompatibilidades numéricas en el constructor, solamente necesitamos considerar que

el parámetro entero s_j toma valores en el conjunto de signos $D_j = \{-1, 1\}$ que caracteriza cada punto de intersección.

Asumamos que L es el número total de operaciones de intersección en la construcción: *ilc* más *icc*. Definiremos el *índice* asociado al plan de construcción como el conjunto ordenado $I = \{s_1, \ldots, s_j, \ldots, s_L\}$ con $s_j \in D_j$, $1 \leq j \leq L$. Por lo tanto, el producto cartesiano de los conjuntos $\mathcal{I} = D_1 \times \ldots \times D_L$ define el espacio al que pertenecen las instancias solución del problema geométrico.

Un plan de construcción que es solución de un problema geométrico puede verse en función del índice I. Denotemos por $\Psi(I)$ al plan de construcción expresado como la fórmula de lógica de primer orden constructiva correspondiente ya definida e ilustrada en la Sección 3.3.2. Claramente, el conjunto de índices $\{I \in \mathcal{I} | \Psi(I) = verdadero\}$ es el espacio de índices factibles, es decir el conjunto de índices en el que cada uno de ellos selecciona una solución al problema de restricciones geométricas. Tal conjunto de índices es el espacio factible de búsqueda, [38].

Denotemos por Φ a la fórmula lógica de primer orden definida por la conjunción de las restricciones adicionales proporcionadas para especificar la instancia solución deseada por el usuario. Sea f una función (posiblemente de valor real), definida sobre $\Psi(I) \wedge \Phi$, que ha de ser optimizada. Entonces, de acuerdo a lo indicado por Eiben y Ruttkay, [38], la terna $\langle \mathcal{I}, f, \Psi(I) \rangle$ define un problema de optimización de restricciones donde encontrar una solución supone hallar un índice I en el espacio factible de búsqueda con un valor óptimo f.

De forma simple, el problema objeto de estudio puede ser definido como sigue:

> Sea G un problema geométrico definido por un conjunto de restricciones. Sea $P(\mathcal{I})$ un plan de construcción generado por un *solver* constructivo que define el espacio de instancias solución para G en función del índice $\mathcal{I}$. Sea C' el conjunto de restricciones adicionales que caracteriza la instancia solución deseada. Encuéntrese un índice $I \in \mathcal{I}$ tal que sea máximo el número de restricciones que satisface la instancia solución $P(I)$ en C'.

3.5. Aplicación de Metaheurísticas para la solución deseada

Teniendo en cuenta la definición del problema de la selección de la solución deseada como problema de optimización combinatoria y los grandes requerimientos computacionales necesarios para su resolución, cabe la posibilidad de aplicar Metaheurísticas en el proceso, véase Capítulo 2.

Para poder aplicar Metaheurísticas al problema se hace necesario disponer de una codificación de las soluciones y de una función objetivo o función *fitness*. Todo ello ha quedado definido en la Sección 3.4.1.

3.5.1. Caracterización de una solución

Una solución a nuestro problema geométrico vendrá constituida por un índice I propuesto. Dado que el índice $I = \{s_1, \ldots, s_j, \ldots, s_L\}$ con $s_j \in D_j = \{-1, 1\}$, $1 \leq j \leq L$, cada solución es un *string* binario de longitud L en el que el valor 0 representa al signo $s_j = -1$ y el valor 1 representa al signo $s_j = 1$.

Sin embargo, todos los índices I posibles no serán soluciones del problema. Una instancia solución del problema geométrico será aquel índice I que pertenezca al espacio de búsqueda factible: $\{I \in \mathcal{I} | \Psi(I) = verdadero\}$.

Más aún, entre todas las instancias solución solamente se requiere obtener a aquellas que verifiquen la fórmula booleana Φ. Serán las instancias que cumplan todas las restricciones adicionales definidas por el usuario.

3.5.2. Función objetivo

Tal y como se indicó en la Sección 3.4.1, en un problema de optimización de restricciones dado, $\langle \mathcal{I}, f, \Psi(I) \rangle$, la función f, definida sobre $\Psi(I) \wedge \Phi$ es la función objetivo que una Metaheurística deberá optimizar. Será la función *fitness*.

En general, las restricciones serán manejadas mediante la definición de f como un sumatorio de términos de penalización que ponderen el *fitness* de los posibles índices I según el grado de violación de las restricciones del conjunto $\Psi(I) \wedge \Phi$.

El diseño de una función apropiada de penalización que habilite la convergen-

cia de la Metaheurística a una solución factible suboptimal o incluso óptima es crucial, [129], y conlleva abordar dos cuestiones fundamentales. La primera de ellas es definir la penalización relativa con la que cada restricción contribuye a la función *fitness* global. La segunda consiste en desvincular la función *fitness* de la Metaheurística específica aplicada. En general, abordar estas cuestiones de forma efectiva requiere conocimiento sustancial del problema que se trata, [37].

La función *fitness* utilizada en nuestro caso es muy simple y no variante. El *fitness* de cada solución I generada se mide mediante el recuento del número de restricciones adicionales que cumple el índice correspondiente,

$$f(I) = \begin{cases} \sum_{i=1}^{|R|} \delta(R_i(I)) & \text{si } I \text{ es una instancia solución} \\ -1 & \text{si } \Psi(I) \text{ no se verifica} \end{cases} \tag{3.4}$$

donde $\delta(R_i(I)) = 1$ si el índice asociado a la instancia solución I cumple la restricción adicional R_i en Φ, y $\delta(R_i(I)) = 0$ en otro caso.

La salida de la Metaheurística específica aplicada será aquella instancia solución que maximiza la función *fitness* f.

3.5.3. Estudios previos

Las referencias previas relativas a la aplicación de Metaheurísticas al problema de la selección de la solución deseada solamente corresponden a Algoritmos Genéticos.

En [106], [109] se aplicaron Algoritmos Genéticos, [114], para la resolución del problema de la selección de la solución deseada. Los algoritmos aplicados fueron tanto el Algoritmo Genético básico, [67], como Algoritmos Genéticos multimodales, [134]. Se demostró tanto la eficiencia como la efectividad de la aplicación de los Algoritmos Genéticos al problema. Los tamaños del problema analizados presentan un espacio de búsqueda de hasta 2^{20} soluciones.

Por otra parte, el estudio de la optimización estadística de los parámetros, [101], que dirigen la evolución de los Algoritmos Genéticos, concretamente del Algoritmo Genético básico, en su aplicación al problema es descrita en [12]. Se propone y contrasta una configuración óptima para tamaños del problema con un espacio de búsqueda de hasta 2^{20} soluciones.

Capítulo 4

Algoritmos evolutivos estudiados: CHC y PBIL

Con objeto de ampliar el estudio de la resolución eficiente del problema de la selección de la solución deseada mediante Metaheurísticas, en [105], [108], [160], [161], se realizaron estudios preliminares para comprobar el rendimiento de un considerable conjunto de algoritmos. Se aplicaron tanto Metaheurísticas basadas en trayectoria como basadas en población.

Las Metaheurísticas basadas en trayectoria utilizadas fueron: algoritmos de búsqueda local, [1], *simulated annealing*, búsqueda tabú y búsqueda local multiarranque, [117], [127]. Las Metaheurísticas basadas en población aplicadas fueron: Algoritmos Genéticos, [134], Algoritmos de Estimación de Distribuciones, [95], y Optimización basada en Colonias de Hormigas, [35].

Los resultados mostraron que los algoritmos más prometedores correspondían claramente a las Metaheurísticas basadas en población. Concretamente, los algoritmos que produjeron mejores aportaciones fueron un Algoritmo Genético avanzado, *Cross generational elitist selection Heterogeneous recombination and Cataclismic mutation* (CHC), [40], y un algoritmo basado en estimación de distribuciones, *Population-Based Incremental Learning* (PBIL), [10]. De ahí que el presente estudio se haya realizado únicamente sobre ambos algoritmos.

En este Capítulo comenzaremos describiendo los fundamentos, características, estructura y parámetros de evolución correspondientes a los algoritmos CHC y PBIL. Posteriormente, presentaremos un estudio estadístico exhaustivo para el ajuste de los parámetros de evolución correspondientes a cada

algoritmo. Finalmente, propondremos mediante un modelo estadístico simple un conjunto de expresiones para la predicción de los parámetros de evolución ante tamaños del problema desconocidos.

4.1. El algoritmo CHC

El algoritmo CHC, [40], se enmarca dentro de la modalidad de los Algoritmos Genéticos que busca profundizar sobre el equilibrio en la búsqueda entre convergencia y diversidad.

El algoritmo de búsqueda adaptativo CHC es un Algoritmo Genético no tradicional que combina una estrategia de selección conservativa, que siempre preserva los mejores individuos encontrados, con un operador de recombinación radical (altamente disruptivo o tendente a la diversidad) que produce descendientes que presentan la diferencia máxima posible con respecto a ambos padres. Como sabemos, la diversidad está asociada a las diferencias entre los cromosomas en la población. La falta de diversidad genética supone que todos los individuos de la población sean parecidos y que, así mismo, el discurrir del algoritmo sea conducido a una convergencia prematura hacia óptimos locales. En la práctica es algo irreversible, pero existen soluciones: inclusión de mecanismos de diversidad en la evolución y reinicialización cuando se produce convergencia prematura, entre otras. Véanse, por ejemplo, [131], [132].

El algoritmo CHC supone una mayor elaboración con respecto a los Algoritmo Genéticos tradicionales, intentando establecer un equilibrio entre diversidad y convergencia. Es importante considerar que CHC parte de una representación binaria.

4.1.1. Estructura del algoritmo

Por un Algoritmo Genético tradicional entendemos un Algoritmo Genético para el que se asume lo siguiente (tomaremos como base el algoritmo presentado en la Figura 2.2):

1. La inicialización de la población $P(0)$ (de tamaño fijo PS) es aleatoria.

2. La selección para la reproducción está sesgada hacia la selección de las mejores estructuras.

3. La selección para la supervivencia no está sesgada, típicamente se produce reemplazando la población padre completa $P(t-1)$ con la población hija $C'(t)$ generada desde $P(t-1)$.

4. El operador de recombinación es cruce en uno o dos puntos.

5. Se usa un bajo ratio de mutación en la etapa de recombinación para mantener la diversidad de población.

CHC difiere del Algoritmo Genético tradicional en todos menos en el primero de los puntos. Primero, es dirigido por una selección para la supervivencia (selección elitista) en vez de una selección para la reproducción. En otras palabras, el sesgo a favor de las estructuras de mejor rendimiento ocurre en la selección para la supervivencia en vez de en la selección para la reproducción. Segundo, se introduce un nuevo sesgo en contra de los individuos emparejados que son similares (prevención de incesto). Tercero, el operador de recombinación usado por CHC es una variante del cruce uniforme (HUX), una forma de cruce altamente disruptiva (capaz de romper la tendencia convergente de los elementos constituyentes de los cromosomas introduciendo gran diversidad). Finalmente, la mutación no se ejecuta en la etapa de recombinación. Además, la diversidad es mantenida (o más precisamente, reintroducida) por aleatorización parcial en la población siempre que se detecta convergencia (reinicialización).

CHC introduce, por tanto, cuatro componentes novedosas:

- **Selección Elitista:** selecciona los PS mejores cromosomas entre padres e hijos. Los PS mejores elementos encontrados hasta el momento permanecerán en la población actual.

- **Cruce Uniforme HUX:** intercambia exactamente la mitad de los alelos que son distintos en los padres. Garantiza que los hijos tengan una distancia Hamming máxima a sus dos padres.

- **Prevención de Incesto:** Se forman $PS/2$ parejas con los elementos de la población. Sólo se cruzan las parejas cuyos miembros difieren en un número determinado de bits (umbral de cruce). El umbral se inicializa a $L/4$ (L es la longitud del cromosoma). Si durante un ciclo no se produce ni un solo cruce, al umbral de cruce se le resta uno.

- **Reinicialización:** Cuando el umbral de cruce es menor que cero, la población se reinicializa: usando el mejor elemento como plantilla e incluyendo una copia suya, o manteniendo el mejor o parte de los mejores de la población y el resto aleatorio.

CHC no aplica el operador de mutación.

En la Figura 4.1 mostramos en pseudocódigo el algoritmo básico asociado al CHC.

```
Procedimiento CHC
ENTRADAS
  TMAX: Tiempo de ejecución
  PS: Tamaño de población
  L: Longitud de cromosoma
  D: Umbral diferencia
  DR: Ratio de divergencia
  M: Número de mejores individuos para reinicialización
SALIDA
  C: Mejor individuo de la población

  t=0
  Inicializar(P(t))
  Evaluar estructuras en P(t)
    Mientras (no se alcancen las condiciones de terminacion) hacer
        t=t+1
        SeleccionarParaReproduccion C(t) de P(t-1)
        Recombinar estructuras en C(t) para formar C'(t)
        Evaluar estructuras en C'(t)
        SeleccionarParaSupervivencia P(t) de C'(t) y P(t-1)
        Si EsIgual (P(t-1), P(t)) entonces
            DecrementarUmbralDiferencia (D)
        FinSi
        Si (D<0) entonces
            Reinicializar(P(t))
            InicializarContadorConvergencia(D, DR, L)
        FinSi
    FinMientras

FinProcedimiento
```

Figura 4.1: Estructura del algoritmo CHC básico.

Describiremos a continuación cada una de las componentes fundamentales del algoritmo CHC.

4.1.2. Selección elitista

CHC potencia la supervivencia del mayor rendimiento. Durante la selección para la reproducción, en vez de sesgar la selección de los candidatos $C(t)$

para la reproducción en favor de los miembros de mejor rendimiento de la población padre $P(t-1)$, cada miembro de $P(t-1)$ se copia a $C(t)$, y se empareja aleatoriamente para la reproducción.

Durante la selección para la supervivencia, por otra parte, en vez de reemplazar la vieja población padre $P(t-1)$ con la población hija $C'(t)$ para formar $P(t)$, los hijos recién creados deben competir con los miembros de la población padre $P(t-1)$ para la supervencia. Los miembros de $P(t-1)$ y $C'(t)$ se mezclan y ordenan de acuerdo a su *fitness* o rendimiento, y $P(t)$ se crea seleccionando los mejores PS miembros (donde PS es el tamaño de la población) de la población mezclada. En casos en los que un miembro de $P(t-1)$ y un miembro de $C'(t)$ tienen el mismo *fitness*, el miembro de $P(t-1)$ se dispone en la posición más alta del *ranking*. Conocemos a este procedimiento de retener los mejores miembros ordenados de las poblaciones padre e hija mezcladas como selección elitista de la población, ya que garantiza que los mejores PS individuos generados siempre sobrevivirán.

En la selección tradicional (selección proporcional o lineal basada en rangos) los mejores consiguen más copias que los peores. La selección elitista logra lo mismo al preservar a los mejores durante más generaciones y así a lo largo del tiempo generando más copias. El efecto de un operador de cruce disruptivo, sin embargo, es bastante diferente para la selección elitista que para la selección tradicional. La selección elitista es mucho más robusta que la selección tradicional porque compensa para aquellas ocasiones en las que genera muchos hijos peores al reemplazar menor parte de los padres.

El requerimiento esencial para cualquier operador de recombinación usado en conjunción con la selección elitista es que no sea fuertemente sesgado contra los mejores esquemas, véase Sección 2.4.6. No importa si el operador es altamente disruptivo con tal de que la descendencia de los individuos relativamente peores no tenga una mejor oportunidad de sobrevivir a un emparejamiento que la descendencia de los mejores individuos.

Presentamos, a continuación, en la Figura 4.2, y en pseudocódigo, las funciones que corresponden a la selección para la reproducción y a la selección para la supervivencia (selección elitista).

4.1.3. Cruce Uniforme HUX

Es importante tener en cuenta que no sólo se desea preservar el esquema sino también proporcionar recombinaciones productivas. Hay un abismo entre

Procedimiento SeleccionarParaReproduccion

Copiar todos los miembros de $P(t-1)$ en $C(t)$ aleatoriamente

FinProcedimiento

Procedimiento SeleccionarParaSupervivencia (Selección elitista)

Obtener $P(t)$ a partir de $P(t-1)$
al reemplazar los peores miembros de $P(t-1)$
con los mejores miembros de $C'(t)$
hasta que no queden miembros de $C'(t)$
que sean mejores que ningún miembro restante de $P(t-1)$

FinProcedimiento

Figura 4.2: Estructura de las funciones de selección para la reproducción y selección para la supervivencia (selección elitista).

recombinación efectiva y preservación.

La idea intuitiva tras la recombinación es la de que la combinación de características de dos buenos padres puede producir incluso mejores hijos. El objetivo es el de copiar esquemas altamente valorados desde ambos padres, instanciándolos simultáneamente en el mismo hijo. Claramente cuantos más bits se copien desde el primer padre más esquemas se copiarán y así, lo más probable es que se copien, sin disrupción, esquemas altamente valorados. Por otra parte, cuantos más bits se copien del primer padre, menos se pueden copiar procedentes del segundo padre, incrementándose así la probabilidad de la disrupción de esquemas altamente valorados del segundo padre. Consecuentemente, un operador de cruce que cruce sobre la mitad de los bits (o incluso mejor, la mitad de los bits diferentes) tendrá más probabilidad de combinar esquemas valorables simplemente por la razón de que se combina el máximo número de esquemas de cada padre.

El operador de recombinación usado por CHC es una variante del cruce uniforme. El cruce uniforme (UX) intercambia bits en vez de segmentos. Para cada posición en la cadena, los bits de los dos padres se intercambian con probabilidad fija p (típicamente 0.5). CHC usa una versión modificada de UX, HUX, que cruza sobre exactamente la mitad de los alelos no coincidentes, donde los bits que se intercambiarán se eligen aleatoriamente sin reemplazamiento. HUX garantiza que los hijos están siempre a la máxima distancia de Hamming de sus dos padres. La cara opuesta asociada a la división de HUX es que maximiza la oportunidad de dos buenos esquemas, uno por cada padre, quedando combinados en un hijo, puesto que se escoge la mitad del

material de cada padre. Además, todos los esquemas del mismo orden tienen una oportunidad igual de quedar divididos o preservarse.

Bajo ciertas circunstancias, HUX estará sesgado contra los mejores esquemas. Debería tenerse en cuenta, sin embargo, que lo mismo ocurre con el Algoritmo Genético tradicional, [67]. De hecho, cualquier operador de recombinación estará sesgado a favor de ciertos tipos de bloques construidos y en contra de otros. Así como el Algoritmo Genético tradicional trabaja con bloques construidos con longitudes cortas definidas, CHC trabaja con bloques construidos de bajo orden. HUX es especialmente bueno al tratar aparte grandes bloques construidos (de alto orden) en pequeños bloques construidos (de bajo orden) y recombinarlos.

Debería tenerse en cuenta, sin embargo, que aunque HUX es altamente disruptivo, es, a diferencia de la mutación, un operador de verdadera recombinación. No puede introducir nuevos alelos, sino que simplemente recombina esquemas contenidos en los padres. En consecuencia, los *loci* que han convergido no pueden ser pertubados por el cruce uniforme. En otras palabras, cuanto más han convergido los padres, menos disruptivo es el cruce uniforme. No hay forma de que un Algoritmo Genético pueda asignar más intentos a los esquemas de los mejores individuos sin converger a los *loci* que definen tales esquemas. La convergencia puede ser un signo de progreso pero también un signo de estancamiento.

Presentamos, a continuación, en la Figura 4.3, y en pseudocódigo, la función que correspondería, en principio, al cruce dentro del algoritmo CHC, algo que veremos que cambiará posteriormente.

Procedimiento Recombinación (Cruce HUX)

Para (cada una de las $PS/2$ parejas de estructuras de $C(t)$) **hacer**
Intercambiar la mitad de los bits diferentes aleatoriamente
FinPara

FinProcedimiento

Figura 4.3: Estructura de la función de cruce.

4.1.4. Prevención de incesto

El crecimiento exponencial de instancias de buenos esquemas es de poco valor si conduce a la convergencia prematura. Uno de los efectos de cruzar sobre la mitad de los bits diferentes entre los padres es que se disminuye

el peligro de la convergencia prematura. Puesto que los mejores individuos tienen más descendientes, es muy probable que un individuo se empareje con uno de sus relativos cercanos. En tal caso, ello conduciría a cruzar individuos que comparten muchos alelos: la degeneración rápida aparecería a partir de la exploración determinada por la recombinación. Aunque cruzando la mitad de las diferencias (usando HUX) retarda este proceso, algunas veces los individuos emparejados presentan una diferencia poco significativa. Si uno o ambos hijos sobreviven a este emparejamiento, será incluso más probable que tal situación ocurra en la siguiente generación.

CHC tiene un mecanismo adicional para retardar la etapa de convergencia, un mecanismo para ayudar a evitar el incesto. Durante la etapa de reproducción, cada miembro de la población padre se elige aleatoriamente sin reemplazamiento y se casa para el emparejamiento. Antes del emparejamiento, sin embargo, se calcula la distancia de Hamming entre padres potenciales, y si la mitad de esa distancia (la distancia de Hamming de los hijos esperados desde sus padres) no excede un umbral diferencia, no se emparejan y se borran de la población hija. Se empareja sólo una fracción de la población para producir nueva descendencia en cualquier generación. Siempre que no haya hijos aceptados en la población padre (o bien porque no hay emparejamientos potenciales o bien porque ninguno de los hijos fue mejor que el peor miembro de la población padre), se decrementa el umbral diferencia.

El efecto de este mecanismo es que sólo se emparejan los padres potenciales más diversos, pero la diversidad requerida por el umbral diferencia automáticamente decrementa a medida que la población converge naturalmente. El número de supervivientes para cada generación permanece remarcablemente constante en la búsqueda porque cuando CHC tiene dificultad en hacer progreso, el umbral diferencia se deja caer más rápido que el promedio de la distancia de Hamming, de tal forma que se evalúan más individuos. En el caso opuesto, cuando CHC lo encuentra fácil: generar hijos que sobrevivan, el umbral diferencia se deja caer en un ratio más bajo, y el número de emparejamientos cae.

La prevención de incesto se incorpora por tanto en la función de cruce del algoritmo de evolución CHC, modificando su discurrir en la forma indicada. La función definitiva que corresponde al cruce dentro del algoritmo CHC queda en pseudocódigo como se presenta en la Figura 4.4.

Procedimiento Recombinar (Cruce HUX)

Para (cada una de las $PS/2$ parejas de estructuras de $C(t)$) **hacer**
Determinar la distancia de Hamming
Si ($distanciaHamming/2$)> D **entonces**
Intercambiar la mitad de los bits diferentes aleatoriamente
Sino
Borrar la pareja de estructuras de $C(t)$
FinSi
FinPara

FinProcedimiento

Figura 4.4: Estructura de la función definitiva para el cruce.

4.1.5. Reinicialización

El ciclo reproducción-recombinación de CHC no usa ninguna mutación. El uso de HUX y prevención de incesto en conjunción con una población de suficiente gran tamaño para preservar un número de estructuras diversas habilita a CHC a retardar la convergencia prematura, y así trabajar bastante bien sin ninguna mutación. Pero estos mecanismos diversos no pueden garantizar que ningún alelo convergerá prematuramente. Se necesita algún tipo de mutación.

La mutación, sin embargo, es menos efectiva en CHC que en el Algoritmo Genético tradicional. Puesto que CHC establece bastante diversidad a través de sus componentes, la mutación contribuye muy poco a corto plazo en la búsqueda. Por otra parte, cuando la población está cercana a converger, la mutación, combinada con la selección elitista, no es muy efectiva en la introducción de diversidad. En esta etapa de la búsqueda la mutación raramente producirá un individuo que sea mejor que el peor individuo de la población, y consecuentemente, muy pocos individuos nuevos serán aceptados en la población.

En contraste con CHC, un Algoritmo Genético tradicional, al reemplazar a la población padre en cada generación, asegura que se introducirán nuevas variaciones constantemente. La forma de hacer esto a partir de CHC es la de introducir mutación sólo cuando la población ha convergido o la búsqueda se ha estancado (por ej., el umbral diferencia se ha dejado caer a cero y ha habido varias generaciones sin ningún descendiente nuevo aceptado en la población padre). Más específicamente, siempre que el ciclo reproducción-recombinación logra su condición de terminación, la población se reinicializa (diverge) y se repite el ciclo.

La reinicialización, sin embargo, es sólo parcial. La población se reinicializa usando el mejor individuo encontrado como plantilla para crear una nueva población. Se crea cada nuevo individuo alterando una proporción fija (p.ej. un 35 %) de los bits de plantilla elegidos aleatoriamente sin reemplazamiento. Así mismo, se añade una instancia de los mejores M individuos sin cambiar. Esto asegura que la siguiente búsqueda no puede converger a una solución peor que la previa. Finalmente, el umbral diferencia D se reinicializa como indicador de convergencia a partir del ratio de divergencia DR y teniendo en cuenta la longitud de los cromosomas L. Este bucle más externo, consistente en reinicialización seguida de búsqueda genética, es iterado hasta que se alcanza la condición de terminación.

Hemos de puntualizar que para CHC no hay peligro de que el mejor individuo encontrado hasta ahora tomará rápidamente la población, incluso aunque se incluya en la población reinicializada. El mecanismo de prevención de incesto de CHC, en combinación con la selección elitista y la recombinación disruptiva previenen de esto. Ello no significa tampoco que, el salvar los mejores individuos no tenga efecto en la búsqueda. El cruce aún recombinará los esquemas representados en estos mejores individuos supervivientes con el resto de individuos de la población, pero se previene el emparejamiento siempre que se asocia a dos individuos relativamente similares tal y como refleja el umbral diferencia en la prevención de incesto.

La ventaja que aportan las reinicializaciones parciales sobre la mutación crónica es que CHC puede trabajar bastante bien en un amplio rango de problemas usando la misma configuración de parámetros. Las reinicializaciones proporcionan muchos de los beneficios de una gran población sin el coste de una búsqueda más lenta. En problemas sencillos, se encuentra la solución "óptima" normalmente en el primer ciclo de inicialización. En problemas duros, la solución "óptima" es encontrada sólo tras reinicializaciones sucesivas.

Finalmente, presentamos en la Figura 4.5, y en pseudocódigo, la función que correspondería a la reinicialización dentro del algoritmo CHC.

4.1.6. Parámetros de evolución

A continuación describiremos brevemente los parámetros asociados a la evolución del algoritmo CHC durante las diferentes etapas de su ejecución. Incidiremos en los valores que pueden ser asignados a tales parámetros.

Procedimiento Reinicializar

Reemplazar M individuos en P_t con mejores M de P_{t-1}
Para todos excepto los mejores M de P_t **hacer**
    Reemplazar el individuo con el mejor global
    Cambiar DR * L bits aleatoriamente
FinPara
EvaluarPoblacion(P_t)

FinProcedimiento

Figura 4.5: Estructura de la función de reinicialización.

- *Tamaño de la población* (PS): es el número de cromosomas que componen la población que evoluciona en la búsqueda. El rango de valores sobre el que se mueve no está definido, no es finito.

- *Umbral diferencia* (D): referencia para indicar el máximo grado de parecido permisible entre dos cromosomas para el cruce. Sus valores son números enteros entre 0 y $L/2$ siendo L la longitud del cromosoma. En el discurrir del algoritmo, su valor se inicializa tras la reinicialización. El valor definido como más adecuado en la literatura, [40], es $L/4$.

- *Ratio de divergencia* (DR): valor que indica el porcentaje de bits, en tanto por 1, que han de ser alterados del mejor cromosoma actual de la población para formar cada uno de los cromosomas del resto de la población en la reinicialización. El valor que puede adquirir es real y corresponde al intervalo $[0, 1]$. El valor definido como más adecuado en la literatura, [40], es 0,35.

- *Mejores individuos* (M): valor que indica el número de cromosomas que permanecerán en la población en la reinicialización, los M mejores. El resto serán generados a partir del ratio de divergencia DR. El valor que puede adquirir es un número entero entre 1 y el número de cromosomas total albergado en la población. En la literatura, [40], no hay definido un valor concreto como más adecuado, solamente se establece que este valor no ha de albergar una gran cantidad de cromosomas.

4.2. El algoritmo PBIL

A continuación describiremos el segundo de los algoritmos que tratamos en este trabajo. En este caso, profundizamos en una estrategia dentro de los Algoritmos Genéticos que supone la combinación de los mismos con otras técnicas que funcionan bien dentro de la resolución de problemas de optimización. De dicha estrategia han surgido multitud de grupos de algoritmos entre los que se encuentra aquel del que emana el algoritmo de *Aprendizaje Probabilístico basado en Poblaciones (PBIL)*, [10]: los algoritmos evolutivos basados en modelos probabilísticos (EDAs), [95]. En este caso, se trata de combinar las técnicas de aprendizaje con los Algoritmos Genéticos, adaptando éstos en su estructura, evolución y funcionamiento en base a los matices que supone el proceso de aprendizaje.

4.2.1. Evolución basada en modelos probabilísticos

Con objeto de vencer las desventajas de los operadores de recombinación usuales de los algoritmos de Computación Evolutiva que tienen probabilidades de romper los buenos bloques de construcción de soluciones, se desarrollaron una serie de algoritmos, llamados *Algoritmos de Estimación de Distribución* (EDAs).

Estos algoritmos, que tienen su base teórica en la teoría de la probabilidad, también están basados en poblaciones que evolucionan a medida que lo hace el progreso de búsqueda. Los Algoritmos de Estimación de Distribución usan modelado probabilístico de soluciones prometedoras para estimar una distribución sobre el espacio de búsqueda. Tal distribución se usa entonces para generar la siguiente generación al muestrear el espacio de búsqueda según la misma. Después de cada iteración se reestima la distribución. Para obtener una visión general de los EDAs véanse [95], [124].

A continuación, en la Figura 4.6, se muestra en pseudocódigo el algoritmo básico asociado a todos los EDAs.

Uno de los primeros EDAs propuestos para la optimización combinatoria es el llamado Aprendizaje Incremental basado en Población (PBIL), [10], [11].

Tratamos con Metaheurísticas de búsqueda basadas en la adaptación de probabilidades. Estas Metaheurísticas constituyen una familia de algoritmos que tienen como aplicación más común la resolución de problemas binarios. Vemos que, en lugar de adaptar los individuos de una población, como se hace

Procedimiento Algoritmo básico de Estimación de Distribución (EDA)

P=InicializarPoblacion()
  Mientras (no se alcancen las condiciones de terminacion) **hacer**
    P_{sel}=Seleccionar(P)
    $\rho(x)$=$\rho(x|P_{sel})$=EstimarDistribucionProbabilidad()
    P=MuestrearDistribucionProbabilidad()
  FinMientras

FinProcedimiento

Figura 4.6: Estructura del algoritmo básico de Estimación de Distribución.

en otros Algoritmos Evolutivos, en este caso se adapta una representación de la población (distribución de probabilidad) constituida por un *vector de probabilidades*. Tal representación sirve como patrón para la generación de individuos y es desplazada por el algoritmo correspondiente hacia las zonas más prometedoras del espacio de búsqueda.

El algoritmo general que sirve como base para la mayor parte de las Metaheurísticas que tratamos en esta Sección se describe en el pseudocódigo de la Figura 4.7.

Procedimiento AG genérico básico mediante adaptación de probabilidades

1 *Inicializar* V=(p_1,...,p_l) (en general (0.5,...,0.5))
2 *Generar* $P(s_1,\ldots,s_{PS})$ utilizando V
3 *Evaluar* $f(s_1),\ldots,f(s_{PS})$
4 *Actualizar* V de acuerdo a P y $f(s_1),\ldots,f(s_{PS})$
5 *Ir* a 2 o *Parar*

FinProcedimiento

Figura 4.7: Estructura genérica más utilizada para los algoritmos de búsqueda mediante adaptación de probabilidades.

La diferencia fundamental entre los distintos algoritmos asociados a las Metaheurísticas de Búsqueda mediante Adaptación de Probabilidades suele venir dada por el mecanismo de actualización de la distribución de probabilidad.

Uno de los representantes más conocidos de los EDAs es el algoritmo de Aprendizaje Incremental basado en Poblaciones (PBIL). Se trata de un método que combina a los Algoritmos Genéticos con el aprendizaje competitivo a la hora de la optimización. PBIL es una extensión al denominado "Algoritmo Genético de Equilibrio" (EGA), [85], que intenta describir la población

límite de un Algoritmo Genético a partir de un punto de equilibrio al asumir que la población está siempre buscando la convergencia como estado de equilibrio. El proceso que lleva a cabo EGA puede verse como una forma de eliminar el paso de cruce explícito en la búsqueda genética estándar. El examinar el rendimiento del algoritmo EGA en términos del aprendizaje competitivo supuso el nacimiento de PBIL: un algoritmo mucho más simple que el Algoritmo Genético tradicional y que supone mucho mayor rendimiento que éste en un gran conjunto de problemas de optimización tanto en velocidad como en eficacia.

A continuación entraremos de forma concreta en la Metaheurística de las introducidas que analizaremos en este trabajo: PBIL.

4.2.2. La población y el vector de probabilidades

Para la descripción del algoritmo que tratamos partiremos de la estructura y evolución básicas establecidas para los Algoritmos Genéticos y obtendremos una nueva estructura y evolución en base al nuevo funcionamiento que supone la incorporación de las reglas de aprendizaje y los fundamentos que de éste se derivan.

El algoritmo PBIL intenta crear un vector de probabilidades del que se obtienen muestras o individuos representativos en cada iteración para producir la población que constituye la generación correspondiente. Tal y como ocurre en el Algoritmo Genético tradicional, se asume que cada solución o individuo se codifica a partir de un vector de longitud fija. La población se reemplaza por un simple vector de probabilidades, que especifica, en cada posición, la probabilidad de contener un valor específico. Ignorando la contribución del resto de operadores en cada generación, podemos decir que la distribución de valores esperada en cada posición de la población durante una generación determinada puede calcularse en base a la población generada en la generación anterior.

PBIL describirá a la población a partir de un vector de probabilidades siempre teniendo en mente a individuos binarios. Así, en cada posición, el vector de probabilidades especifica la probabilidad $p \in [0, 1]$ de ocurrencia del valor '1' o el valor '0', siendo su complementario $1 - p$ la ocurrencia del valor no tenido en cuenta. El paralelismo y la diversidad en la búsqueda quedarán concentrados en el vector indicado de tal forma que con las posiciones del vector de probabilidades a 0.5 daremos la mayor diversidad, algo que se irá reduciendo a medida que nos acerquemos en cada posición particular a 0 o 1,

Población 1	**Población 2**	**Población 3**
0 0 1 1 0	1 0 1 0 0	1 0 1 0 0
1 1 0 0 1	1 1 0 0 1	0 1 0 1 1
1 1 0 0 0	1 1 0 0 0	1 0 1 0 0
0 0 1 1 1	1 1 0 0 1	0 1 0 1 1
Representación 1	**Representación 2**	**Representación 3**
0.5 0.5 0.5 0.5 0.5	1.0 0.75 0.25 0.0 0.5	0.5 0.5 0.5 0.5 0.5

Figura 4.8: Representación de población basada en probabilidad.

lo que implicará obviamente la convergencia en la posición implicada. Véase como ejemplo la Figura 4.8.

Pero, hemos de tener en cuenta que, aunque PBIL utilice el vector de probabilidades para definir una población, no realiza sin embargo el proceso contrario, es decir, generar a través de la población el vector de probabilidades. La meta de PBIL es la de crear activamente un vector de probabilidades que represente con alta probabilidad a una población de vectores solución con gran *fitness*. No lo hace a partir de la generación de la población para posteriormente dejarla discurrir a través del efecto de los distintos operadores (selección, cruce, mutación, etc.,) sino a partir de la directa actuación de los operadores sobre el vector de probabilidades. Estos mecanismos se derivan de los usados por parte del *aprendizaje competitivo*, [7].

De forma similar al entrenamiento de una red de aprendizaje competitivo, los valores del vector de probabilidades son desplazados gradualmente hacia la representación de aquellos individuos con mejor *fitness*: zonas más prometedoras del espacio de búsqueda. Para ello se debe usar una regla de actualización del vector de probabilidades. Dicha regla viene dada por la transcripción de lo realizado en el aprendizaje competitivo. En el aprendizaje competitivo se utiliza una regla de actualización de pesos a la hora de mover una salida hacia lo que establece un determinado conjunto de muestras, individuos o ejemplos. La regla utilizada por PBIL es similar a ésta.

Es interesante notar que, cada valor de bit es examinado independientemente. Esto supondría el desprecio de mucha de la información que el cruce estándar preserva. Pero, la razón de asumir tal desprecio es que el vector de probabilidades no define físicamente la población completa. El vector de probabilidades representa un punto simple basado en el vector con evaluación de mejor *fitness* y alrededor del cual debería ser generada la siguiente población de puntos o individuos simples. Hay que tener en cuenta esta puntualización al término ”representación”.

Queda, por tanto, indicar cómo se determina el movimiento del vector de probabilidades: hacia qué vectores solución lo desplazamos. Para resolver este problema se han propuesto multitud de métodos. El método que se propuso inicialmente suponía la generación, en cada iteración, de una única solución, hacia la que se decidiría si acercar o no el vector de probabilidades. Sin embargo, pronto se ideó un método más efectivo consistente en la generación, en cada iteración, de un cierto número de vectores solución, todos ellos basados en las probabilidades especificadas por el vector de probabilidad. El vector de probabilidades se movería hacia lo establecido por el vector solución generado con mejor valor de *fitness*. Se probó empíricamente que este nuevo método es más resistente a caer en mínimos locales.

Tenemos, por tanto, un ciclo, en el que en cada iteración se genera una población de soluciones potenciales en vez de una sola solución. Siempre tenemos como generador de soluciones y receptor de resultados al vector de probabilidades. Se mantiene así la capacidad para explorar grandes porciones del espacio de búsqueda de forma paralela, tal y como hace el Algoritmo Genético tradicional, [67].

En las etapas iniciales de la búsqueda, hay una gran cantidad de diversidad en las regiones exploradas del espacio de búsqueda: valores del vector de probabilidad próximos a 0.5. En la medida en que progresa la búsqueda, los valores se alejan de 0.5 hacia 0.0 o 1.0. En el Algoritmo Genético tradicional esto corresponde a aumento, para cada posición de los vectores solución, del número de soluciones en las que predomina un determinado valor de bit. A medida que progresa la búsqueda en el Algoritmo Genético tradicional, la población tiende a converger a una región del espacio de búsqueda en la que se situará un vector solución con un *fitness* considerable. Así ocurre también en PBIL, donde la búsqueda converge a una región alrededor de un vector solución de entidad, pero en este caso a través del vector de probabilidades. En principio, el algoritmo PBIL y el Algoritmo Genético tradicional van a tratar con el problema de la convergencia prematura: en PBIL, a medida que las probabilidades se hacen más cercanas a 0.0 o 1.0, se incrementa la similaridad en los vectores solución generados. Sin embargo, hemos de tener en cuenta que, PBIL tiene control explícito de la rapidez con que la población converge.

PBIL tiene mucho menor poder expresivo que la población completa correspondiente al Algoritmo Genético tradicional. Hay que tener en cuenta que a partir de un mismo vector de probabilidades se puede generar una gran multiplicidad de poblaciones diferentes, dependiendo de la etapa de búsqueda.

Pero, los individuos generados por PBIL corresponden generalmente a una determinada región del espacio de búsqueda: la delimitada por el vector de probabilidades. El Algoritmo Genético tradicional puede tener, en principio, simultáneamente en la población a individuos de varias regiones y no una que se va enfocando poco a poco como hace PBIL. Sin embargo, esto no es así, y debido al desplazamiento genético hacia un punto localizado en la búsqueda invitado por su comportamiento efectivo, el Algoritmo Genético tradicional no será capaz de mantener múltiples puntos diferentes en etapas no tempranas de la búsqueda. PBIL también cambiará rápidamente sus valores iniciales de 0.5 en el vector de probabilidades y pronto se dispondrán hacia 0.0 o 1.0, de tal forma que el vector de probabilidad representará sólo a uno de los puntos de la región focalizada.

La diversidad en la búsqueda es mantenida en PBIL gracias a que la población representada por el vector de probabilidades no es única. En el Algoritmo Genético tradicional es el cruce uniforme el que permite producir hijos extremadamente diferentes a partir de unos mismos padres.

Pero, como se introdujo anteriormente, el Algoritmo Genético no depende del cruce a la hora de ejecutar una búsqueda extensiva. En las etapas finales de la búsqueda, la mutación juega un papel fundamental. De una forma análoga, se define un operador de mutación en PBIL. Para PBIL, tendríamos dos formas de definir un operador de mutación: la primera es utilizar el operador de mutación directamente sobre los vectores generados y la segunda es utilizarlo sobre el vector de probabilidades (pequeña probabilidad de perturbación en cada una de las posiciones del vector de probabilidades). Ambos tipos de mutación tienen el mismo efecto que la mutación en el Algoritmo Genético tradicional: ayudar a preservar la diversidad.

La mutación tiene su mayor importancia en las etapas tardías de la búsqueda, cuando se pierde la diversidad en la población, [142]. El cruce es válido en fases tempranas de la búsqueda en la medida en que supone pasos más amplios a la hora de encontrar buenas soluciones, [43]. El ajuste final de las soluciones lo realiza el operador de mutación.

Aunque la forma en que son generadas las soluciones potenciales en PBIL a partir del vector de probabilidades permite muestrear una región considerable, la incorporación de la regla de mutación supone el establecer la prevención de caer rápidamente hacia los valores extremos: 0.0 o 1.0.

Para la mutación, al igual que ocurre en el Algoritmo Genético tradicional, se establece una determinada probabilidad de mutación. Cada valor concreto

del vector de probabilidades será mutado siempre que se supera ese valor de probabilidad. En caso de que así ocurra, se realizará un desplazamiento aleatorio del valor en un sentido u otro, desplazamiento correspondiente a un valor fijo establecido al inicio de la ejecución del algoritmo en la forma básica de éste.

La regla de mutación correspondiente a PBIL se muestra a continuación en la Figura 4.9.

```
Procedimiento Mutación PBIL
  ENTRADAS
  PV: vector de probabilidades
  MP: probabilidad de mutación
  MS: desplazamiento de mutación

  Para (cada valor del vector de probabilidades) hacer
    Si (aleatorio[0,1] < PM) entonces
        PV(i)=PV(i)*(1.0-MS)+aleatorio(0.0 o 1.0)*MS
    FinSi
  FinPara

FinProcedimiento
```

Figura 4.9: Estructura genérica más utilizada para la mutación.

Para mantener la analogía con el Algoritmo Genético tradicional, el número de vectores generados en cada iteración se define como el tamaño de la población (PS). La regla de actualización del vector de probabilidades es tratada desde la perspectiva del aprendizaje competitivo.

4.2.3. Incorporación del aprendizaje competitivo

En esta Sección entramos dentro de los fundamentos de la parte de aprendizaje de cuya filosofía se extraen los rasgos puramente extra-genéticos que se incorporan al Algoritmo Genético básico para transformarlo en el algoritmo PBIL. Así, se introduce el concepto de aprendizaje competitivo y su regla de evolución y se dispone el alojamiento de éstos en el algoritmo genético transformado en base al marco de trabajo del aprendizaje.

El aprendizaje competitivo, [7], es usado a menudo para clasificar una serie de muestras no etiquetadas en distintos grupos. La pertenencia a cada grupo se basa en la similaridad de las muestras con respecto a las características en estudio. El procedimiento es no supervisado, ya que no existe conocimiento *a priori* de cuántos grupos existen, o de qué características pueden distinguir a un grupo de otro. El objetivo es que el aprendizaje competitivo sea capaz

de determinar las características más relevantes y sea capaz de clasificar en base a las mismas.

El aprendizaje competitivo suele ser estudiado en el contexto de las redes neuronales artificiales. Una red de aprendizaje competitivo típica tiene unidades de entrada conectadas con todas las unidades de salida, estando éstas últimas conectadas también entre sí. En las unidades de entrada se establece el vector de características de cada muestra. Las salidas corresponden a la clase o grupo en el que ha sido clasificada la muestra.

El objetivo es el del ajuste de los pesos de la red para cumplir el cometido del aprendizaje competitivo: delimitar las características que son capaces de dividir a los ejemplos o muestras en clases para que dada una muestra como entrada, se realice su propagación por la red y se establezca su clasificación correspondiente.

En el entrenamiento de una red de aprendizaje competitivo, cada vez que se presenta una muestra, existe una neurona de salida ganadora: la que indica la clasificación de la misma. Los pesos de esta neurona de salida son ajustados de acuerdo a dicha muestra y en base a una regla: la regla de actualización de pesos en el aprendizaje competitivo. La Ecuación 4.1 presenta la mencionada regla,

$$\Delta w_{ij} = LR \times (entrada_j - w_{ij}) \tag{4.1}$$

donde Δw_{ij} es la variación en el peso de la conexión entre la neurona de salida i y la neurona de entrada j, LR es el ratio de aprendizaje, $entrada_j$ es el valor de entrada j-ésimo de la muestra y w_{ij} es el peso de la conexión entre la neurona de salida i y la neurona de entrada j.

El proceso de entrenamiento de una red de aprendizaje competitivo supone la presentación repetida de muestras a la red hasta su estabilización. Después de que se completa el entrenamiento de la red, los vectores de peso para cada una de las unidades (neuronas) de salida pueden ser considerados vectores prototipo para cada una de las clases descubiertas. Los atributos o características con pesos más grandes serán los que definirán a cada clase o grupo. De aquí se extrae el punto de partida de PBIL: vector prototipo de probabilidades y regla de actualización del mismo.

La regla de actualización del vector de probabilidades, en el algoritmo PBIL básico, queda, tomando como punto de partida la Ecuación 4.1, como sigue,

$$p_i = p_{i\_ant} \times (1{,}0 - LR) + mejor_i \times LR \tag{4.2}$$

donde p_i es el valor i-ésimo del vector de probabilidades que se actualiza, p_{i_ant} es el valor i-ésimo que el vector de probabilidades tenía previo a la actualización, LR es el ratio de aprendizaje y $mejor_i$ es el valor i-ésimo de la mejor solución encontrada en la generación actual.

El vector de probabilidades mantenido por PBIL puede verse como un vector prototipo para generar vectores solución que tienen una alta evaluación con respecto al conocimiento disponible en el espacio de búsqueda. En cada generación el vector de probabilidades se ajusta de acuerdo a la mejor solución. Aquí no es apropiado el contexto no supervisado del aprendizaje competitivo, puesto que lo que percibe interés es la capacidad que el aprendizaje competitivo tiene para encontrar un prototipo para vectores o soluciones de buen *fitness*. Además, en PBIL las muestras no están preetiquetadas como en el aprendizaje competitivo sino que se etiquetan a lo largo de la búsqueda con su *fitness*.

En el caso de PBIL, en cada generación, la mejor solución encontrada se dice que está en la clase de interés y el vector de probabilidades no es sólo un prototipo sino también una guía de la búsqueda que produce las siguientes muestras sucesivas de las que aprender. Así mismo, tenemos abundantes datos de entrenamiento: todo el espacio de búsqueda, pero interesa evaluar el mínimo número de ellos.

El entrenamiento del algoritmo PBIL podemos decir que es similar al método de cuantización del vector de aprendizaje (LVQ) propuesto por Kohonen en [89], [90].

4.2.4. Estructura y parámetros de evolución

Una vez descrito el algoritmo PBIL: sus componentes, comportamiento y evolución, estamos en condiciones de mostrar la estructura del algoritmo básico. Éste se detalla en pseudocódigo en la Figura 4.10.

Podemos comprobar que hay cuatro parámetros que intervienen en la evolución del algoritmo PBIL: el tamaño de la población (PS), el ratio de aprendizaje (LR), la probabilidad de mutación (MP) y el desplazamiento de mutación (MS). A continuación describimos brevemente tales parámetros e indicamos los valores que pueden ser asignados a los mismos.

```
Procedure Algoritmo PBIL
ENTRADA
  TMAX: Número de iteraciones
  PS : Tamaño de población
  L : Longitud de cromosoma
  MP: Probabilidad de mutación
  MS: Desplazamiento de mutación
  LR: Ratio de aprendizaje
OUTPUT
  PV: Vector probabilidad
  BG: Mejor solución global

  InicializarVectorProbabilidades(PV) (cada posición = 0.5)
  Para (j=1,...,TMAX) hacer
    Para (i=1,...,PS) hacer
      sols[i]=generarSolucionSegunProbabilidades(PV)
      evaluaciones[i]=evaluar(sols[i])
    FinPara
    mejor=EncontrarMejorSolucion(sols,evaluaciones)
    BG := ActualizarMejorGlobal(mejor, BG)
    Para (i=1,...,L) hacer
      PV[i]=PV[i]*(1.0-LR)+mejor*LR
    FinPara
    Para (i=1,...,L) hacer
      Si (aleatorio[0,1]<PM) entonces
        PV[i]=PV[i]*(1.0-MS)+aleat(0.0 o 1.0)*MS
      FinSi
    FinPara
  FinPara

FinProcedimiento
```

Figura 4.10: Estructura del algoritmo PBIL básico.

- *Tamaño de la población* (PS): es el número de muestras de la población que se ha de generar por iteración a partir del vector de probabilidades. El rango de valores sobre el que se mueve no está definido, no es finito.
- *Probabilidad de mutación* (MP): probabilidad para mutar cada posición del vector de probabilidades en caso de ser designada para ello. Sus valores corresponden al intervalo $[0, 1]$.
- *Desplazamiento de mutación* (MS): valor de desplazamiento de cada posición del vector de probabilidades en caso de mutación. El desplazamiento puede ser negativo o positivo según indica la regla de mutación, véase Figura 4.9. El valor que puede adquirir es real y corresponde al intervalo $[0, 1]$.
- *Ratio de aprendizaje* (LR): ratio que regula la velocidad con la que el vector de probabilidades se aproxima a la mejor solución generada durante la iteración. Los valores corresponden al intervalo $[0, 1]$. Un ratio de aprendizaje reducido supone una mayor exploración del espacio de búsqueda. Por otra parte, un valor mayor para el mismo es sinónimo de explotación de la información obtenida en la iteración actual.

De estos parámetros, el único parámetro que no está catalogado dentro de los Algoritmos Genéticos tradicionales es el ratio de aprendizaje. Para el resto existe una gran cantidad de literatura que describe su tratamiento: tamaño de la población, [54], [56], [118] y ratios y desplazamientos de mutación, [9], [32], [84].

El parámetro correspondiente al ratio de aprendizaje tiene un mayor efecto en PBIL que en el aprendizaje competitivo. Como en el aprendizaje competitivo, el ratio de aprendizaje afecta en la rapidez en que el vector prototipo se desplaza para ajustar un punto correctamente clasificado. Puesto que en PBIL, el vector de probabilidades se usa para generar el siguiente conjunto de soluciones, el ratio de aprendizaje también afecta a qué regiones del espacio de búsqueda son exploradas. El valor del ratio de aprendizaje tiene un impacto directo en el equilibrio entre exploración y explotación del espacio de búsqueda (cuanto más alto es el ratio de aprendizaje mayor es la explotación y cuanto más bajo mayor es la exploración).

Tal y como ocurría en CHC, en PBIL podemos introducir reinicialización para combatir la convergencia del algoritmo, que, aunque se puede retardar con el ajuste del parámetro correspondiente al ratio de aprendizaje, es bastante rápida. Así, una vez que se comprueba que el vector de probabilidades

ha convergido: valores muy cercanos a 0.0 o 1.0, reinicializaremos el vector de probabilidades a su forma inicial (valores a 0.5), almacenando, en todo caso, la mejor solución encontrada hasta ese momento. Véase la Figura 4.11.

```
Procedimiento Reinicializar

  Si Estancamiento(PV) entonces
        Para todas las posiciones de PV hacer
          PV(i)=0.5
        FinPara
  FinSi

FinProcedimiento
```

Figura 4.11: Reinicialización del vector de probabilidades.

4.3. Ajuste de parámetros de evolución

Los Algoritmos Evolutivos (y, de hecho, cualquier método heurístico) están caracterizados por un conjunto de parámetros que determinan su evolución y para los que se deben elegir valores específicos. Una de las principales dificultades de la aplicación de un algoritmo evolutivo a un problema dado reside en la elección de valores apropiados para tales parámetros. Dichos valores son comúnmente seleccionados en la práctica mediante ensayo y error, ajustados a mano, o tomados de otros campos, [101].

Desafortunadamente, estos métodos no garantizan la obtención de valores óptimos para los parámetros, y teniendo en cuenta que los resultados experimentales son siempre dependientes del problema, aquellos valores que son útiles en determinadas aplicaciones es posible que no supongan buen rendimiento ante otras tareas desconocidas, [157]. Más aún, cuando se seleccionan los parámetros de esta forma, no se puede establecer relación entre el rendimiento del algoritmo y los valores de los parámetros.

En la literatura, los estudios contrastados realizados sobre valores de los parámetros para las Metaheurísticas aplicadas a nuestro problema corresponden únicamente al Algoritmo Genético básico y, en ningún caso, a derivados del mismo, [12]. Para los algoritmos analizados en esta Sección, CHC y PBIL, podemos únicamente tomar referencias, [10], [40]. A todo ello se une el desconocimiento del usuario del *solver*, véase el Capítulo 3, para establecer unos

valores adecuados para los parámetros. Se trata por tanto de una tarea interna al proceso de resolución del problema.

En esta Sección presentamos un estudio estadístico empírico para determinar la influencia y optimizar los parámetros de evolución de los algoritmos CHC y PBIL, que han sido presentados respectivamente en las Secciones 4.1 y 4.2. Así mismo, exploramos la posibilidad de identificar rangos para los valores asignados a los parámetros de estos algoritmos que permitan obtener un rendimiento estadísticamente óptimo. El objetivo será el de proponer para cada tamaño analizado del problema de la selección de la solución deseada una configuración de valores de los parámetros para la que se prevea buen rendimiento. El estudio completo y detallado está disponible en [71], [73].

4.3.1. Hipótesis iniciales

Las hipótesis iniciales que se plantean y a las que se pretende dar respuesta a partir de los experimentos a realizar son las siguientes:

1. La variación de los valores en cada uno de los parámetros para cada algoritmo tiene un efecto estadísticamente significativo sobre el rendimiento de los dos algoritmos a estudiar.

2. Para cada parámetro que controla la ejecución de los algoritmos a estudiar existe un valor que permite asegurar que, en la mayoría de los casos, el rendimiento del algoritmo será óptimo. Tales valores pueden ser identificados y probados experimentalmente.

3. Los valores óptimos de los parámetros son función de las características del problema, fundamentalmente, función del tamaño de las instancias.

4.3.2. Diseño experimental

Los parámetros de evolución de CHC y PBIL ya fueron descritos respectivamente en las Secciones 4.1.6 y 4.2.4. Para determinar el comportamiento de PBIL y CHC en función de tales parámetros se ha aplicado una metodología empírica a partir de un estudio estadístico basado en el Análisis de Varianza (ANOVA), [25], [27], [30], sobre los resultados experimentales. Para la elección de ANOVA se han tenido en cuenta las limitaciones de las diferentes propuestas para evitar un diseño factorial completo de los experimentos (al-

Parámetro	*Valores*						
Tamaño de población (PS)	10	20	30	40	50	60	70
Ratio de divergencia (DR)	0.20	0.25	0.30	0.35	0.40	0.45	0.50
Umbral diferencia (D)[a]	2	3	4	5	6	7	
M mejores (M)	1	2	3				

[a]El valor 2 sólo será usado para los tamaños 2^{18} y 2^{19} y el valor 7 sólo para el tamaño 2^{20} dado que como nivel central se dispone el recomendado en [40]: $L/4$, siendo L el tamaño del problema: 18, 19 y 20 respectivamente.

Tabla 4.1: Niveles de los factores para el algoritmo CHC.

goritmos de *racing*, optimización secuencial de parámetros, etc.,), [101], y la naturaleza estadística de las muestras obtenidas de los experimentos.

ANOVA es una técnica estadística ampliamente utilizada e inicialmente propuesta por R. A. Fisher en los años 20, [42]. El empleo de ANOVA permite examinar los efectos de una o más variables independientes (cuantitativas o cualitativas) que se encuentran bajo el control del investigador, denominadas *factores*, sobre una *respuesta* cuantitativa o variable dependiente que necesitamos medir. Así, se puede comprobar si las variaciones en los valores de la respuesta se deben a cambios en los valores de los factores (*niveles*) o, por el contrario, son provocadas por un efecto aleatorio (o por algún factor que no haya sido considerado), siendo posible determinar los factores que tienen una mayor incidencia sobre la variable dependiente. Además, se pueden estudiar los efectos de la interacción entre factores sobre la variable dependiente (si es que existe).

En nuestro caso concreto, cada parámetro de un algoritmo es un factor y el conjunto de valores que puede tomar son los niveles del factor. A la combinación de valores de los parámetros de un algoritmo que permiten una ejecución del mismo se le conoce con el nombre de *tratamiento*, [33].

Si la aplicación de ANOVA demuestra que, en efecto, los parámetros o factores considerados (y/o la interacción entre los mismos) tienen efectos estadísticamente significativos sobre la eficiencia de nuestros algoritmos, el siguiente paso supone decidir los valores más adecuados para cada parámetro o factor que permitirán optimizar el rendimiento de CHC y PBIL.

Se han considerado varios niveles para cada factor estudiado. Éstos se muestran en la Tabla 4.1 para CHC y en la Tabla 4.2 para PBIL.

Los niveles para el tamaño de población han sido tomados de un estudio previo de este factor, [161]. Los niveles de los restantes parámetros han sido

Parámetro	*Valores*						
Tamaño de población (PS)	10	20	30	40	50	60	70
Ratio de aprendizaje (LR)	0.05	0.10	0.15	0.20	0.25		
Probabilidad de mutación (MP)	0.010	0.025	0.050	0.075	0.100		
Desplazamiento de mutación (MS)	0.010	0.025	0.050	0.075	0.100		

Tabla 4.2: Niveles de los factores para el algoritmo PBIL.

seleccionados como se describe a continuación. En primer lugar, para cada parámetro se ha escogido un valor central tomando como referencia lo indicado en la literatura, véanse [40] para CHC y [10] para PBIL. Tras ello se ha definido un conjunto de niveles adicionales distribuidos de forma equilibrada con respecto al valor central. La mitad de tales valores son inferiores al valor central y la otra mitad superiores.

El banco de instancias considerado corresponde a espacios de búsqueda limitados por 2^{18}, 2^{19} y 2^{20} soluciones. Los elementos geométricos asociados a las diferentes instancias son, respectivamente, 18, 19 y 20. La longitud del índice asociado a los diferentes planes de construcción correspondientes es, respectivamente, 18, 19 y 20, siendo tal la longitud del cromosoma que codificará a las soluciones que intervienen en los algoritmos. El banco de instancias ha sido obtenido aleatoriamente a partir de la generación de grafos de restricciones basada en secuencias de Henneberg, véase la Sección 3.3.1. Las diferentes instancias que lo constituyen y sus características están disponibles en [159]. Disponemos de 29 instancias con un espacio de búsqueda limitado por 2^{18} soluciones y 12 instancias para cada uno de los espacios de búsqueda limitados por 2^{19} y 2^{20} soluciones.

La ejecución de un algoritmo con un determinado tratamiento tendrá éxito si y sólo si el algoritmo encuentra una solución que verifica todos los predicados adicionales definidos para seleccionar la solución deseada, véanse las Secciones 3.4 y 3.5. La limitación para realizar tal proceso, $TMAX$ en el pseudocódigo presentado para CHC en la Figura 4.1 y para PBIL en la Figura 4.10, vendrá determinada por el número de evaluaciones permitidas de la función *fitness* (Sección 3.5.2).

Por una parte, en vez de medir las ejecuciones de los algoritmos en términos de tiempo de CPU, es preferible el uso de contadores de operación representativos como medidas independientes de la plataforma para el rendimiento de un algoritmo, [3]. Además, el coste computacional de la ejecución de un algoritmo evolutivo corresponde en nuestro caso, básicamente, a la etapa de evaluación: cálculo del número de predicados adicionales que verifica cada

una de las instancias solución que constituyen la población, [106], [107]. Por tanto, la unidad representativa de la operación de cada algoritmo seleccionada será la evaluación de una solución.

Para cada tratamiento y ejecución almacenaremos el número de evaluaciones realizadas, que denominaremos longitud de ejecución. El número máximo de evaluaciones permitido antes de declarar una ejecución como fallida será denominada longitud de ejecución máxima y denotada como rl_{max} ($TMAX$ en el pseudocódigo de los algoritmos). Puesto que el contexto de aplicación de los algoritmos en la operación de un *solver* requiere interacción en tiempo real, [81], [153] el valor límite rl_{max} ha sido establecido en 30000 evaluaciones.

Denominaremos *observación* a la *longitud de ejecución media* estimada (esperanza de longitud de ejecución) a partir de la siguiente expresión,

$$\widehat{E}(RL) = \frac{1}{k}\Sigma_{i=1}^{k} rl_i + \frac{(n_r - k)}{k}\ rl_{max} \tag{4.3}$$

donde k es el número de ejecuciones con éxito y rl_i es la longitud de ejecución correspondiente a la i-ésima ejecución con éxito, [70]. El número total de ejecuciones es $n_r = 50$, cada una de ellas iniciada a partir de una semilla aleatoria diferente.

Finalmente, antes de aplicar ANOVA es esencial verificar el cumplimiento de las supuestos bajo los que el método garantiza su validez. Las condiciones que han de satisfacerse para una utilización segura de ANOVA (y, de hecho, para cualquier test paramétrico) son, [137]:

- Independencia: en Estadística, dos eventos son idependientes cuando el hecho de que uno de ellos ocurra no modifica la probabilidad de que el otro ocurra. Las observaciones obtenidas para cada tratamiento deben ser independientes entre sí. Ello está garantizado a partir de la selección de una semilla aleatoria diferente para cada ejecución de CHC y PBIL.
- Normalidad: una observación es normal cuando su comportamiento sigue una distribución normal o de Gauss con un cierto valor de media μ y varianza σ. Las observaciones obtenidas para cada tratamiento deben seguir una distribución normal. El supuesto de normalidad es menos crítico puesto que se trabaja sobre las medias de la muestra que es normalmente distribuida si el tamaño de la muestra es suficientemente grande (Teorema del Límite Central, [33]).

- Homocedasticidad: esta propiedad indica que no existe violación de la hipótesis de igualdad de varianzas. Las varianzas entre tratamientos diferentes deben ser similares. El supuesto de igualdad de varianzas es crítico solamente en el caso de que existan grandes diferencias entre los tamaños de muestra para cada tratamiento. En nuestro caso, disponemos del mismo número de observaciones, incluso del mismo número de ejecuciones, para cada tratamiento. Además, el test de Levene, [99], ha sido utilizado para comprobar si la muestra presenta o no tal homogeneidad de varianzas.

Para garantizar que las muestras cumplen los requerimientos de normalidad y varianza, [25], se decidió obtener 50 observaciones para cada tratamiento e instancia del problema. Este diseño experimental produjo, para cada instancia del problema, 875 series para PBIL y 735 series para CHC (una serie por cada tratamiento considerado) de valores de longitud de ejecución, $\widehat{E}(RL)$, cada una de ellas con 50 observaciones.

Para demostrar la influencia de los parámetros sobre el rendimiento de los algoritmos, se ha realizado un análisis estadístico exhaustivo de los resultados empíricos a partir de ANOVA. Las variables independientes son los parámetros de los algoritmos y la variable dependiente es la longitud de ejecución media, $\widehat{E}(RL)$, requerida para encontrar una solución.

Para verificar el comportamiento de los algoritmos en base a los parámetros de evolución se ha realizado un *análisis unifactorial* (Sección 4.3.3), un *análisis multifactorial* (Sección 4.3.4), se han aplicado los *test de comparaciones múltiples a posteriori* (Sección 4.3.5) y finalmente, se ha realizado la *selección del mejor nivel* (Sección 4.3.6) para cada parámetro. En primer lugar, se ha determinado si el efecto de cada parámetro individual sobre el resultado del algoritmo es significativo mediante ANOVA de 1-vía. En segundo lugar, se ha aplicado ANOVA multifactorial para comprobar cómo afectan al rendimiento del algoritmo todas las posibles combinaciones de factores y para determinar la cuantía de tal influencia. En tercer lugar, los test de comparaciones múltiples *a posteriori* han permitido seleccionar los mejores rangos de valores para cada parámetro diferenciando entre los resultados que cada nivel de un factor proporciona. Finalmente, se ha determinado el mejor nivel para cada parámetro obtenido teniendo en cuenta solamente los parámetros individuales y, por otra parte, el tratamiento que proporciona los mejores resultados.

Factores	Suma de Cuadrados Tipo III	Grados de libertad	Media Cuadrática	F	Sig.
Modelo Corregido	5.27E+12	734	7.18E+09	140.237	0.000
Intersección	3.07E+13	1	3.07E+13	599242.853	0.000
PS	4.66E+12	6	7.76E+11	15167.498	0.000
DR	1.56E+10	6	2.60E+09	50.882	0.000
D	2.31E+11	4	5.78E+10	1128.489	0.000
M	3.74E+10	2	1.87E+10	365.378	0.000
PS * DR	1.64E+10	36	4.56E+08	8.917	0.000
PS * D	2.27E+11	24	9.47E+09	185.112	0.000
DR * D	1.34E+09	24	5.59E+07	1.091	0.344
PS * DR * D	7.35E+09	144	5.10E+07	0.997	0.494
PS * M	4.92E+10	12	4.10E+09	80.186	0.000
DR * M	9.12E+08	12	7.60E+07	1.486	0.121
PS * DR * M	3.92E+09	72	5.45E+07	1.064	0.332
D * M	4.27E+08	8	5.34E+07	1.043	0.400
PS * D * M	3.87E+09	48	8.07E+07	1.577	0.007
DR * D * M	2.27E+09	48	4.73E+07	0.924	0.622
PS * DR * D * M	1.33E+10	288	4.62E+07	0.903	0.879
Error	1.84E+12	36015	5.12E+07		
Total	3.78E+13	36750			
Total Corregido	7.11E+12	36749			

R Cuadrado = .741 (R Cuadrado Ajustada = .736)

Tabla 4.3: Tabla ANOVA de CHC para instancia de tamaño 2^{20}.

4.3.3. Influencia individual

Hemos aplicado ANOVA unifactorial para estudiar el efecto de cada parámetro listado en la Sección 4.1.6 sobre la longitud media de ejecución, $\widehat{E}(RL)$, requerida por el algoritmo CHC para encontrar una solución. El mismo proceso hemos seguido para los parámetros listados en la Sección 4.2.4 con respecto a PBIL. Los resultados muestran que los diferentes factores presentan una influencia estadísticamente significativa sobre la longitud de ejecución esperada $\hat{E}(RL)$. El estudio completo y detallado se presenta en [71], [73]. Para otros ejemplos de aplicación pueden verse [74], [77], [78], [110].

La Tablas 4.3 y 4.4 muestran ejemplos representativos, seleccionados aleatoriamente, para CHC y PBIL respectivamente. Cada una de las filas de dichas Tablas, desde la tercera a la sexta, constituyen el resultado de la aplicación de ANOVA sobre los factores individuales correspondientes. Podemos comprobar cómo el valor de significación (columna $Sig.$) de cada análisis es inferior a 0,05, lo que significa que las variaciones del factor suponen variaciones en la longitud de ejecución esperada, $\hat{E}(RL)$, con un nivel de confianza de un 95 %.

Factores	Suma de Cuadrados Tipo III	Grados de libertad	Media Cuadrática	F	Sig.
Modelo Corregido	2.78E+11	874	3.18E+08	289.834	0.000
Intersección	3.71E+11	1	3.71E+11	338336.027	0.000
PS	1.34E+10	6	2.24E+09	2042.239	0.000
LR	9.35E+10	4	2.34E+10	21333.990	0.000
MP	1.02E+10	4	2.55E+09	2329.661	0.000
MS	1.28E+10	4	3.20E+09	2922.209	0.000
PS * LR	5.37E+10	24	2.24E+09	2043.108	0.000
PS * MP	4.70E+09	24	1.96E+08	178.729	0.000
LR * MP	9.04E+09	16	5.65E+08	515.767	0.000
PS * LR * MP	1.38E+10	96	1.44E+08	131.444	0.000
PS * MS	5.14E+09	24	2.14E+08	195.444	0.000
LR * MS	1.04E+10	16	6.49E+08	592.457	0.000
PS * LR * MS	1.50E+10	96	1.56E+08	142.310	0.000
MP * MS	6.50E+09	16	4.06E+08	370.725	0.000
PS * MP * MS	2.31E+09	96	2.41E+07	21.988	0.000
LR * MP * MS	1.24E+10	64	1.94E+08	176.754	0.000
PS * LR * MP * MS	1.46E+10	384	3.81E+07	34.731	0.000
Error	4.70E+10	42875	1.10E+06		
Total	6.95E+11	43750			
Total Corregido	3.25E+11	43749			

R Cuadrado = .855 (R Cuadrado Ajustada = .852)

Tabla 4.4: Tabla ANOVA de PBIL para instancia de tamaño 2^{19}.

El valor de F, estadístico de Fisher–Snedecor, [25], [113], se utiliza para decidir si las medias de la muestra siguen la misma variabilidad de muestreo entre sí. F se define como el ratio del cuadrado de la media del modelo al cuadrado de la media del error. Cuanto mayor sea el valor de F, mayor será la importancia de la variable independiente asociada, [25].

Para CHC, el parámetro más importante es el tamaño de la población (PS) para la totalidad de las instancias. Del resto de factores cabe resaltar al parámetro umbral diferencia (D) que le sigue en grado de importancia en la totalidad de los casos. Los parámetros ratio de divergencia (DR) y M mejores (M) presentan menor influencia, aunque ésta se incrementa (F es mayor) a medida que el tamaño del problema aumenta. Tal y como se establece en [40], los parámetros de la etapa de reinicialización del algoritmo, que son precisamente ratio de divergencia (DR) y M mejores (M), demuestran su efectividad con la complejidad del espacio de búsqueda. La Tabla 4.5 muestra un resumen de un análisis exhaustivo de las reinicializaciones por ejecución en la aplicación de CHC sobre los bancos de instancias de los diferentes tamaños analizados. Se puede comprobar cómo el porcentaje de ejecuciones sin reinicialización alguna disminuye a medida que aumenta el tamaño y por

Tamaño	2^{18}	2^{19}	2^{20}
Media de reinics. por ejecución	0.866	2.484	3.318
% de ejecuciones sin reinic.	83.06	72.34	65.80

Tabla 4.5: Influencia de la etapa de reinicialización de CHC.

otra parte, la media de reinicializaciones por ejecución aumenta a medida que lo hace el tamaño.

Por otra parte, en lo que respecta a PBIL, destacan los parámetros correspondientes al ratio de aprendizaje (LR) y al tamaño de población (PS), que quedan establecidos como factores más importantes, mayor valor de columna F, para la totalidad de las instancias. Quedan con menor influencia los factores que intervienen en la etapa de mutación: probabilidad (MP) y desplazamiento de mutación (MS). Estos dos últimos factores presentarán mayor importancia en la medida en que se incremente la necesidad de diversificación en la búsqueda, [10], lo que tiene mayor probabilidad de ocurrir para instancias del problema de tamaño superior.

Para analizar detalladamente la influencia en la longitud de ejecución media de cada parámetro por separado se ha elaborado por cada parámetro y tamaño del problema un diagrama de cajas (*boxplot*), [149]. La Figura 4.12 muestra los diagramas correspondientes para el algoritmo CHC y el tamaño del problema 2^{18}. La Figura 4.13 presenta tales diagramas para PBIL. En cada diagrama de cajas se puede apreciar la influencia en la longitud de ejecución media $\hat{E}(RL)$, cuyo valor normalizado se presenta en el eje Y, de la variación del valor del factor correspondiente, cuyos niveles se disponen en el eje X.

Para cada nivel de los factores podemos observar la eficiencia esperada normalizada más probable (rango de $\hat{E}(RL)$ delimitado por la caja) cuantificada en términos de maximización: siendo 1 la máxima eficiencia posible. La variabilidad en la eficiencia que proporciona cada valor del factor viene determinada por la línea vertical que se dispone por encima y por debajo de cada caja y que une longitud de ejecución esperada mínima (extremo superior) con longitud de ejecución esperada máxima (extremo inferior). Si tomamos la mediana (línea horizontal que divide cada caja en dos) como elemento representativo de la longitud de ejecución media para cada nivel, observamos los cambios que provoca la variación de valores para el factor.

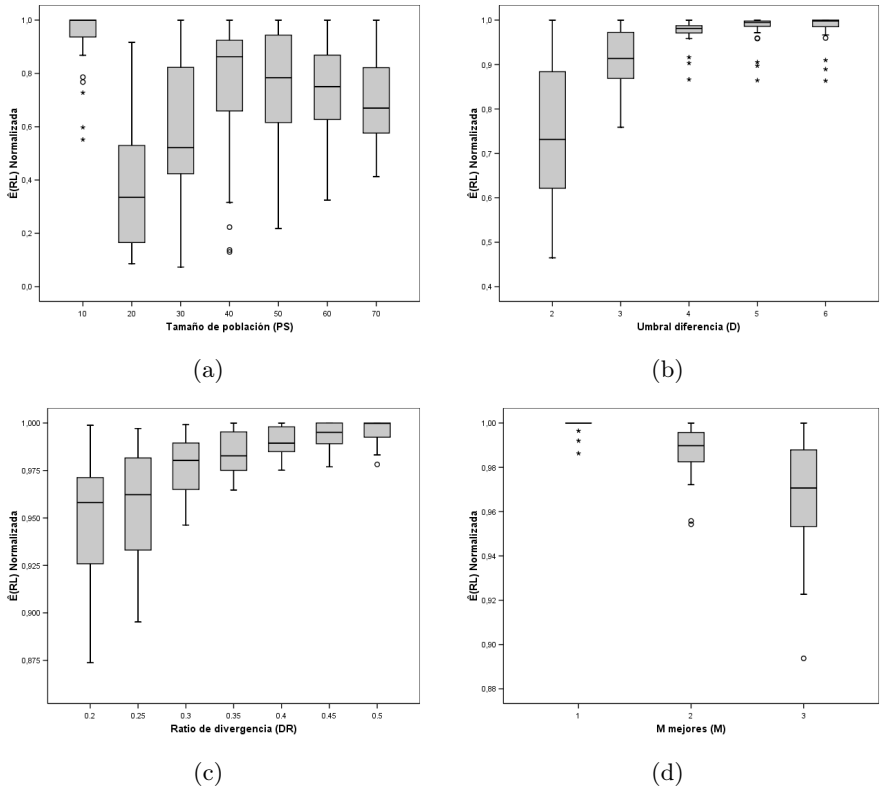


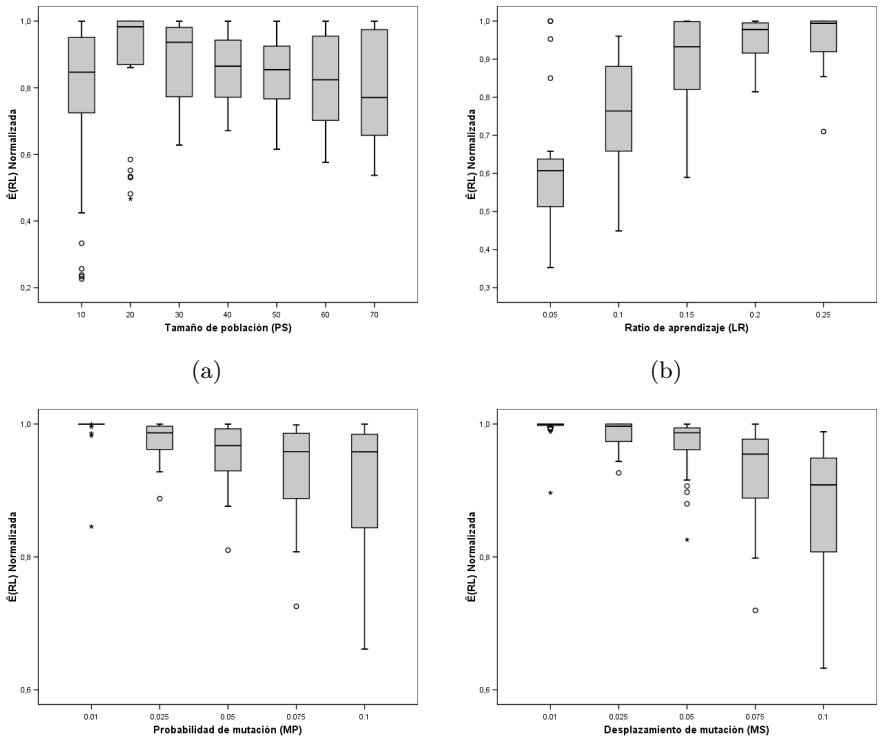
Figura 4.12: Influencia individual normalizada en CHC: a) PS, b) D, c) DR, d) M.

El extremo inferior de cada caja es el primer cuartil y el extremo superior corresponde al tercer cuartil.

Existe una misma tendencia y gran similaridad entre los diferentes diagramas correspondientes a un mismo parámetro para los tamaños analizados. Los diagramas de cajas correspondientes a los tamaños 2^{19} y 2^{20} se presentan en el Apéndice A.

A continuación, describimos la influencia individual de cada parámetro que presentan los diferentes diagramas de cajas. Comenzaremos con los parámetros de evolución del algoritmo CHC:

- **Tamaño de población (PS)**: la mayor garantía de obtener una solución a una determinada instancia del problema en el menor tiempo posible la proporciona el menor nivel de población existente: 10. Junto



(a) (b)

(c) (d)

Figura 4.13: Influencia individual normalizada en PBIL: a) PS, b) LR, c) MP, d) MS.

al mismo, el intervalo constituido por los niveles 50 y 70 supone una segunda alternativa a tener en cuenta. Aparece una ligera tendencia que desplaza hacia la derecha el nivel de tamaño de población alto que proporciona mayor eficiencia a medida que el tamaño del problema se incrementa. Como ya se pronostica en [40], los problemas de mayor tamaño necesitan con una mayor probabilidad un tamaño de población superior.

- **Umbral diferencia (D)**: las variaciones en los niveles del factor suponen, al igual que ocurría con el tamaño de población, una variación significativa en la longitud de ejecución esperada, $\hat{E}(RL)$, necesaria para obtener una solución a cada instancia del problema. Para los diferentes tamaños del problema observamos que los niveles que proporcionan la mayor eficiencia son los valores más altos definidos para el factor umbral diferencia. Para el tamaño 2^{18} el conjunto está formado por los niveles del intervalo $[4, 6]$, para 2^{19} por los niveles del intervalo $[5, 6]$ y para 2^{20} el intervalo es $[5, 7]$. Así mismo, la variabilidad en la eficiencia proporcionada se decrementa (reducción del área de la caja correspondiente) a medida que se incrementa el valor del factor.

- **Ratio de divergencia (DR)**: presenta una influencia estadísticamente significativa (valor de significación de Tabla ANOVA inferior a 0.05) sobre la esperanza de la longitud de ejecución para un 75 % de las instancias de tamaño 2^{18} y 2^{19}, y para un 83 % de las instancias de tamaño 2^{20}. Ello se debe a que la etapa del algoritmo en la que interviene este parámetro no tiene suficiente presencia en el discurrir del algoritmo ante tamaños del problema no muy elevados. En el diagrama de cajas, el incremento de valor para el parámetro supone menor tiempo esperado (mayor eficiencia), $\hat{E}(RL)$, para la obtención de soluciones y menor desviación típica en dicha variable $\hat{E}(RL)$.

- **M mejores (M)**: la importancia del factor se incrementa en la medida en que cambiamos el tamaño del problema a uno superior, tal y como ocurre para el factor ratio de divergencia y por similares razones. Así, si para 2^{18} el parámetro tiene influencia significativa sobre el 66 % de las instancias, para los tamaños 2^{19} y 2^{20} tal influencia se hace estadísticamente significativa para el 83 % de las instancias. En lo que corresponde al nivel del parámetro que proporciona mejores resultados existen variaciones insignificantes para los diferentes tamaños. Así, para una parte importante de las instancias de los diferentes tamaños del problema, la reinicialización con el mejor individuo supone

una evolución más eficiente del algoritmo. Finalmente, en lo referente a la desviación estándar lo que ocurre es similar: es menor cuando el nivel seleccionado para el parámetro corresponde únicamente al mejor individuo.

Procedemos ahora a describir los resultados de la influencia individual de los parámetros para el algoritmo PBIL:

- **Tamaño de población (PS)**: la mejora en la longitud de ejecución esperada para el algoritmo viene determinada por valores de tamaño de población mayores a medida que el tamaño del problema se incrementa. Si para tamaño 2^{18}, el mejor rendimiento se obtiene para tamaños de población del intervalo $[15, 30]$, para 2^{19} y 2^{20} este intervalo incrementa su longitud y se traslada a valores superiores: $[40, 70]$. Para 2^{20} este efecto se manifiesta aún más claramente. El incremento de la complejidad de la instancia del problema, que suele venir propiciado por un aumento del tamaño del problema, exige una elección de tamaño de población superior como propuesta para obtener una longitud de ejecución esperada menor, véase [10].
- **Ratio de aprendizaje (LR)**: el incremento del valor del factor produce una disminución de la longitud de tiempo de ejecución esperada $\hat{E}(RL)$. Esto es lo que ocurre para 2^{18} y lo que continúa demostrándose para los tamaños superiores analizados. De los niveles del parámetro (eje X) cabe destacar los que constituyen el intervalo $[0{,}20, 0{,}25]$ como aquellos que maximizan la normalización de $\hat{E}(RL)$ (eje Y): mayor eficiencia del algoritmo en la obtención de una solución.
- **Probabilidad de mutación (MP)**: la diferencia de altura entre las medianas es muy poco notoria, a excepción del nivel más pequeño: 0,010. Por tanto, la importancia de la modificación del valor de este factor no tiene influencia significativa hasta que alcanzamos el nivel inferior. Todo ello es coherente con la importancia de la etapa de mutación en función del tamaño del problema, [10].
- **Desplazamiento de mutación (MS)**: los resultados establecen una mayor eficiencia (eje Y) a medida que desciende el nivel del factor (eje X). El valor del parámetro para el que se obtienen los resultados más importantes es el que supone la menor diversificación: 0,010. La desviación típica tiende a disminuir a medida que desciende el valor del parámetro.

4.3.4. Influencia multifactorial

El análisis de la influencia en la longitud de tiempo de ejecución esperada proveniente de la cooperación entre los distintos factores de los algoritmos CHC y PBIL se ha realizado a partir de ANOVA multifactorial. Cada uno de los factores es una variable independiente, estableciéndose la longitud de tiempo de ejecución esperada como variable dependiente. El objetivo es el de cuantificar la influencia de las distintas combinaciones de factores posibles de los conjuntos constituidos por los cuatro factores de cada algoritmo ya analizados individualmente en la Sección 4.3.3. Asumimos que todos los factores tienen la misma importancia. Dado que disponemos de cuatro factores para cada algoritmo, la interacción ha sido estudiada agrupando los factores en duplas, ternas y tomando el conjunto completo.

Para cada una de las instancias de los tamaños analizados se realizó un estudio de tal influencia. El estudio completo y detallado se presenta en [71], [73]. Para otros ejemplos de aplicación pueden verse [74], [77], [78], [110]. Un ejemplo representativo se muestra en la Tabla 4.3 para CHC y en la Tabla 4.4 para PBIL, a partir de la séptima fila, tras los análisis unifactoriales.

En CHC, para los distintos tamaños del problema, la mayor parte de las combinaciones de factores posibles no presentan influencia significativa en la evolución del algoritmo (columna $Sig. > 0{,}05$). La excepción viene dada por las duplas, agrupaciones de dos factores, en las que interviene el parámetro tamaño de población.

En PBIL, para las diferentes instancias, las distintas combinaciones posibles de los factores muestran sin excepción, y a diferencia de CHC, una influencia estadísticamente significativa (columna $Sig < 0{,}05$) sobre la longitud de ejecución esperada, $\hat{E}(RL)$. El incremento de tamaño supone un aumento de la significatividad de la influencia (valor del estadístico F), [71], [73].

En general, el efecto sobre el rendimiento de los algoritmos es menor cuando consideramos las interacciones entre los parámetros que si consideramos cada uno de los factores de forma separada.

La cuantificación de la influencia de cada combinación de factores se realizó a partir de la *estimación del tamaño del efecto* (proporción de la variabilidad total de la variable dependiente que puede explicarse por la variación en las variables independientes) con el cálculo de *Eta cuadrado parcial*, [29]. Para CHC, tal y como se muestra en la Tabla 4.6, la influencia de los parámetros individuales tamaño de población (PS) y umbral diferencia (D) siempre es

Tamaño	2^{18}		2^{19}		2^{20}	
Parámetros	Media	Desv. Tip.	Media	Desv. Tip.	Media	Desv. Tip.
PS	0.5290	0.2038	0.6559	0.1201	0.6432	0.2188
DR	0.0032	0.0039	0.0118	0.0201	0.0055	0.0047
D	0.1366	0.1616	0.1900	0.1164	0.0940	0.0899
M	0.0024	0.0051	0.0114	0.0198	0.0063	0.0076
PS * DR	0.0070	0.0056	0.0116	0.0136	0.0104	0.0067
PS * D	0.1538	0.0992	0.1716	0.0520	0.0824	0.0558
DR * D	0.0009	0.0004	0.0016	0.0023	0.0007	0.0003
PS * DR * D	0.0043	0.0012	0.0047	0.0010	0.0040	0.0010
PS * M	0.0055	0.0092	0.0211	0.0387	0.0123	0.0132
DR * M	0.0004	0.0002	0.0004	0.0002	0.0005	0.0002
PS * DR * M	0.0020	0.0006	0.0025	0.0007	0.0022	0.0007
D * M	0.0003	0.0003	0.0006	0.001	0.0002	0.0001
PS * D * M	0.0016	0.0006	0.0016	0.0012	0.0013	0.0004
DR * D * M	0.0014	0.0004	0.0012	0.0003	0.0013	0.0002
PS * DR * D * M	0.0080	0.0011	0.0076	0.0009	0.0071	0.0006

Tabla 4.6: Eta Cuadrado Parcial para los factores de CHC.

superior a la de cualquier combinación de parámetros, de las que la más importante es la constituida precisamente por la dupla tamaño de población y umbral diferencia ($PS*D$). La intervención de los factores ratio de divergencia (DR) y M mejores individuos (M), tanto individual como conjuntamente con otros factores, presenta un efecto insignificante.

Por otra parte, para PBIL, la estimación del tamaño del efecto de cada combinación posible de factores a partir de Eta cuadrado parcial se muestra en la Tabla 4.7. La influencia de los parámetros individuales tamaño de población (PS) y ratio de aprendizaje (LR) siempre es superior a la de cualquier combinación de parámetros, de las que las más importantes son aquellas duplas en las que aparece el ratio de aprendizaje, destacando de las mismas, como se puede prever la conformada por el tamaño de población y el ratio de aprendizaje ($PS*LR$). Por otra parte, el efecto de los factores de mutación: probabilidad de mutación (MP) y desplazamiento de mutación (MS) es mucho menos notorio.

Finalmente, la influencia global tanto de los factores como de todas sus posibles combinaciones viene determinada por el valor del *coeficiente de determinación*, R^2, que indica el porcentaje de la varianza de la variable dependiente que es explicado por los efectos incluidos en el modelo, [36]. Los pies de las Tablas 4.3 y 4.4 presentan el valor de R^2 correspondiente al análisis ANOVA multifactorial de la instancia que tratan.

Tamaño	2^{18}		2^{19}		2^{20}	
Parámetros	Media	Desv. Tip.	Media	Desv. Tip.	Media	Desv. Tip.
PS	0.4317	0.2942	0.4194	0.2597	0.5771	0.2759
LR	0.5382	0.2466	0.4635	0.2794	0.3118	0.2557
MP	0.0287	0.0357	0.0636	0.0617	0.0368	0.0696
MS	0.0537	0.0479	0.0860	0.0863	0.0532	0.0501
PS * LR	0.3769	0.2384	0.3743	0.2183	0.2422	0.1862
PS * MP	0.0111	0.0219	0.0388	0.0425	0.0548	0.1440
LR * MP	0.0362	0.0341	0.0669	0.0690	0.0425	0.0362
PS * LR * MP	0.0379	0.0533	0.0629	0.0630	0.0692	0.0667
PS * MS	0.0098	0.0172	0.0293	0.0324	0.0357	0.0895
LR * MS	0.0387	0.0382	0.0713	0.0751	0.0477	0.0455
PS * LR * MS	0.0389	0.0554	0.0671	0.0680	0.0652	0.0667
MP * MS	0.0258	0.0251	0.0416	0.0426	0.0192	0.0320
PS * MP * MS	0.0067	0.0066	0.0173	0.0153	0.0155	0.0116
LR * MP * MS	0.0453	0.0420	0.0711	0.0713	0.0639	0.0585
PS * LR * MP * MS	0.0463	0.0413	0.0790	0.0714	0.0899	0.0740

Tabla 4.7: Eta Cuadrado Parcial para los factores de PBIL.

	Tamaño		
R^2	2^{18}	2^{19}	2^{20}
Mínimo	0.226	0.551	0.600
Máximo	0.851	0.846	0.854
Promedio	0.586	0.711	0.744
Desv. Tip.	0.197	0.092	0.094

Tabla 4.8: Coeficiente de determinación R^2 para CHC.

En lo que corresponde a los bancos de instancias para los tamaños estudiados, la Tabla 4.8 muestra el resumen de los valores R^2 para CHC y la Tabla 4.9 los correspondientes a PBIL. Podemos observar como el incremento de tamaño del problema supone una mayor influencia global promedio (más cercana a 1) por parte de los parámetros y sus combinaciones en la eficiencia del algoritmo. La variabilidad de tal influencia, por otra parte, desciende a medida que se incrementa el tamaño. Así mismo, la presencia de instancias para las que la aleatoriedad jugaba un papel muy importante y donde el efecto global de los parámetros era prácticamente imperceptible, algo que se produce para tamaño 2^{18}, no se mantiene para 2^{19} y 2^{20}. Se otorga menor probabilidad a la aparición de instancias para las que el valor del coeficiente de determinación sea muy bajo ($R^2 < 0{,}5$).

	Tamaño		
R^2	2^{18}	2^{19}	2^{20}
Mínimo	0.307	0.556	0.528
Máximo	0.967	0.924	0.938
Promedio	0.742	0.764	0.773
Desv. Tip.	0.178	0.113	0.132

Tabla 4.9: Coeficiente de determinación R^2 para PBIL.

4.3.5. Métodos de comparaciones múltiples a posteriori

Hemos demostrado la influencia estadísticamente significativa en la longitud de tiempo de ejecución esperada de los factores que intervienen en la aplicación de los algoritmos CHC y PBIL tanto individual como conjuntamente. Se ha indicado que existe diferencia entre los niveles de los factores pero no qué niveles difieren de qué otros. Para esto último se han utilizado los test de comparaciones múltiples (*post hoc*), [116], cuyos resultados se presentan a continuación. Así delimitaremos el rango de valores más adecuado para cada uno de los factores individuales. Sin embargo, estos test no tienen en cuenta la influencia complementaria del resto de factores.

Por tanto, posteriormente, será necesario, partiendo de la referencia de los resultados de los test de comparaciones múltiples, seleccionar para cada factor el nivel óptimo representativo para cada tamaño del problema teniendo en cuenta la influencia del resto de factores. La selección del mejor tratamiento promedio para cada tamaño del problema permitirá obtener el nivel de cada factor para el que el rendimiento esperado es mayor para el conjunto de instancias correspondiente. Esto último será realizado en la siguiente Sección.

Para realizar los test de comparaciones múltiples se han aplicado dos de los métodos más frecuentemente utilizados debido a los requerimientos previos relativos a la igualdad de varianzas, [116]. Estos son: el test de la diferencia honestamente significativa (HSD) de Tukey, [20] y el test de Games-Howell, [50]. En primer lugar, asumiendo normalidad, homogeneidad de varianzas, e independencia de las observaciones se ha aplicado el test de Tukey. Tras ello, asumiendo la violación del supuesto de homogeneidad de varianzas, se ha aplicado el test de Games-Howell test.

Los resultados de la aplicación de ambos tests han resultado similares. El estudio completo y detallado se presenta en [71], [73]. Para otros ejemplos de aplicación pueden verse [74], [77], [78], [110]. Sólo mostraremos los resultados

del procedimiento de Tukey debido a que nuestro diseño experimental se ajusta más a los supuestos de Tukey que a los de Games-Howell. Usaremos la representación basada en subconjuntos homogéneos definida por Tukey, dada su mayor claridad, para mostrar las conclusiones acerca del mejor rango de valores para cada factor.

La agrupación en subconjuntos homogéneos es tal que cada subconjunto agrupa los niveles que presentan una media de longitud de ejecución que no difiere significativamente. En general, un número considerable de subconjuntos homogéneos agrupando cada uno de ellos uno o un conjunto reducido de niveles, significa que el rendimiento del algoritmo está muy influenciado por los valores del factor correspondiente. Los subconjuntos homogéneos se ordenan de acuerdo a valores incrementales de la longitud de ejecución media. De esta forma, el primer subconjunto incluye las longitudes de ejecución media que corresponden a aquellos niveles del factor para los que el algoritmo presenta el mejor rendimiento.

Veamos un ejemplo. La Tabla 4.10 muestra el ejemplo de una instancia concreta diferenciando los valores del factor tamaño de población (PS) de CHC en cinco subconjuntos (numerados del 1 al 5) en función de su longitud de tiempo de ejecución esperada media (valor mostrado dentro de cada columna). Dentro de cada subconjunto se presentan las medias de longitud de tiempo de ejecución esperada entre las que no existe una diferencia estadísticamente significativa (valor de $Sig. > 0{,}05$). La disposición en los subconjuntos está ordenada de mejor a peor valor de la media, quedando en el subconjunto 1 los mejores niveles del parámetro, en este caso 40 y 50. El número máximo de subconjuntos viene determinado por el número de niveles del factor. La variación en el nivel de un factor será más importante en la medida en que el número de subconjuntos correspondientes se acerque al número de niveles de dicho factor.

De esta forma, los resultados de los test de comparaciones múltiples determinados por los subconjuntos homógeneos se resumirán, para cada factor, bajo los siguientes aspectos:

- Importancia en el rendimiento de la variabilidad en los niveles del factor: establecida por el número de subconjuntos homogéneos.
- Cantidad de niveles que proporcionan el mejor rendimiento: determinada por el número de niveles presentes en el subconjunto homogéneo 1.
- Niveles que proporcionan el mejor rendimiento teniendo en cuenta úni-

	Subconjunto				
PS	1	2	3	4	5
40	354.6373				
50	357.0862				
60		402.8425			
10		403.2177			
70			455.3295		
30				728.3283	
20					1654.2114
Sig.	0.9999	1.0000	1.0000	1.0000	1.0000

Tabla 4.10: Subconjuntos homogéneos del factor PS de CHC y una instancia de 2^{20}.

camente al factor analizado: correspondientes a las longitudes de ejecución esperadas presentes en el subconjunto 1.

A continuación comentaremos los resultados de los test de comparaciones múltiples obtenidos para cada factor individual. Primero comenzaremos por el algoritmo CHC:

- **Tamaño de población (PS):** la Tabla 4.11 muestra un resumen de los resultados del test de comparaciones múltiples. En ella podemos apreciar como la variación en los niveles del factor tamaño de población interviene notablemente en la minimización de la longitud de tiempo de ejecución esperada, pues el número de subconjuntos homogéneos es igual o muy próximo al número total de niveles (7). Así mismo, el nivel a seleccionar para conseguir los mejores resultados es determinante, pues el subconjunto homogéneo 1 está constituido por un solo nivel para la mayoría de las instancias. Los niveles que proporcionan los mejores resultados tienen como referencia a un tamaño de población pequeño (10), aunque una minoría de instancias requiere de tamaños de población más elevados, crecientes con el tamaño del problema (a partir de 40).
- **Umbral diferencia (D):** la importancia del cambio de nivel para el factor umbral diferencia es menor si lo comparamos con el tamaño de población. Si comprobamos los resultados mostrados en la Tabla 4.12 podemos ver que el número de subconjuntos homogéneos cuya presencia es mayoritaria para la totalidad de tamaños del problema es 3 (siendo el máximo posible de 5). Además, los mejores valores de la longitud de ejecución esperada vienen proporcionados, en la mayoría

PS	**Núm. Subconj.**		**Núm. Niveles Subconj. 1**		**Niveles mejor rendimiento**	
Tamaño	Intervalo	Mayor % instancias	Intervalo	Mayor % instancias	Intervalo/s	Mayor % instancias
2^{18}	5-7	6 (38 %)	1-3	1 (66 %)	10; [40,60]	10 (76 %)
2^{19}	6-7	6 y 7 (50 %)	1-2	1 (83 %)	10; [40,70]	10 (67 %)
2^{20}	5-7	7 (50 %)	1-2	1 (75 %)	10; [40,70]	10 (75 %)

Tabla 4.11: Resumen de subconjuntos homogéneos de PS para CHC.

de instancias, por varios niveles (normalmente, 3). Los mejores niveles los constituyen los valores más altos del factor sin apenas distinción entre ellos.

- **Ratio de divergencia (DR):** la falta de influencia de este factor sobre determinadas instancias del problema hace que el número de subconjuntos homogéneos sea muy cambiante, pasando de un único subconjunto (el parámetro no tiene influencia estadísticamente significativa) a 4 subconjuntos donde la influencia es más notoria, tal y como presenta la Tabla 4.13. El subconjunto 1 está constituido por un número considerable de niveles en una parte importante de las instancias, lo que permite únicamente descartar ciertos niveles mientras no detecta diferencias entre el resto. Los mejores niveles vienen determinados por los valores de mayor magnitud del parámetro, lo que elimina directamente a los valores que suponen menor divergencia en la evolución del algoritmo.

- **M individuos mejores (M):** los resultados del test de comparaciones múltiples para este parámetro, véase la Tabla 4.14, son muy similares a los obtenidos para el factor ratio de divergencia como se podía prever, pues ambos son los parámetros de la etapa de reinicialización y sus efectos son parecidos como ya se indicó en la Sección 4.3.3. La influencia del parámetro depende de la instancia concreta y puede llegar a ser considerable, aunque para algunas instancias no sea prácticamente perceptible: número de subconjuntos y niveles en el subconjunto 1 cambiante. La única conclusión que contrasta la totalidad de instancias es el descarte de los niveles superiores del parámetro, quedándonos para la mayoría de instancias únicamente con el mejor individuo ($M = 1$) para que así la evolución del algoritmo sea más eficiente.

Es el turno de analizar ahora los resultados de los test de comparaciones

D	**Núm. Subconj.**		**Núm. Niveles Subconj. 1**		**Niveles mejor rendimiento**	
Tamaño	Intervalo	Mayor % instancias	Intervalo	Mayor % instancias	Intervalo/s	Mayor % instancias
2^{18}	2-4	3 (76 %)	1-4	3 (66 %)	[4,6]	[4,6] (79 %)
2^{19}	2-4	3 (50 %)	2-4	3 (50 %)	[4,6]	[5,6] (100 %)
2^{20}	1-3	3 (83 %)	1-5	3 (83 %)	[5,7]	[5,7] (92 %)

Tabla 4.12: Resumen de subconjuntos homogéneos de D para CHC.

DR	**Núm. Subconj.**		**Núm. Niveles Subconj. 1**		**Niveles mejor rendimiento**	
Tamaño	Intervalo	Mayor % instancias	Intervalo	Mayor % instancias	Intervalo/s	Mayor % instancias
2^{18}	1-4	1,2y3(31 %)	3-7	5(41 %)	[0.30,0.50]	[0.40,0.50](100 %)
2^{19}	1-4	1,2,3y4(25 %)	2-7	5(33 %)	[0.30,0.50]	[0.40,0.50](92 %)
2^{20}	1-4	1,2,3y4(25 %)	3-7	5(42 %)	[0.30,0.50]	[0.40,0.50](100 %)

Tabla 4.13: Resumen de subconjuntos homogéneos de DR para CHC.

M	**Núm. Subconj.**		**Núm. Niveles Subconj. 1**		**Niveles mejor rendimiento**	
Tamaño	Intervalo	Mayor % instancias	Intervalo	Mayor % instancias	Intervalo/s	Mayor % instancias
2^{18}	1-3	2 (52 %)	1-3	2 (52 %)	[1,2]	1 (97 %)
2^{19}	1-3	2 y 3 (42 %)	1-3	1 (50 %)	[1,2]	1 (92 %)
2^{20}	1-3	2 (50 %)	1-3	2 (50 %)	[1,2]	1 (92 %)

Tabla 4.14: Resumen de subconjuntos homogéneos de M para CHC.

múltiples para cada factor individual del algoritmo PBIL, diferenciando en función del tamaño del problema:

- **Tamaño de población (PS):** el resumen de los test de comparaciones múltiples realizados para este factor se muestra en la Tabla 4.15. La importancia de este factor en el análisis ANOVA unifactorial, Sección 4.3.3, queda demostrada aquí con una mayoría de instancias presentando un número elevado de subconjuntos y uno o dos valores en el subconjunto homogéneo 1. Por otra parte, los niveles que proporcionan los mejores resultados obedecen a una tendencia creciente a medida que se incrementa el tamaño del problema, algo que también se detectó en el análisis individual del factor.

- **Ratio de aprendizaje (LR):** al igual que ocurría con el tamaño de población, la diferencia entre los distintos niveles definidos para el ratio de aprendizaje es muy notoria como se demuestra en la Tabla 4.16. El número de subconjuntos homogéneos para la totalidad de tamaños del problema es 4 o 5 (siendo el máximo posible de 5). Además, la mayoría de instancias presentan un único nivel en el subconjunto 1. Como se percibió en el análisis unifactorial es uno de los dos factores más importantes que determinan una mayor eficiencia en la ejecución de PBIL. Los mejores niveles los constituyen los valores más altos del factor, permitiendo el descarte de valores bajos para un conjunto importante de problemas.

- **Probabilidad de mutación (MP):** la menor influencia de este parámetro para ciertas instancias del problema se hace patente tanto en el intervalo correspondiente al número de subconjuntos homogéneos como en la cantidad de niveles que aparecen en el subconjunto homogéneo 1, véase la Tabla 4.17. Sin embargo, existe una parte importante de instancias para las que la elección de un valor adecuado es fundamental: 4 o 5 subconjuntos homogéneos y un único nivel en el subconjunto homogéneo 1. Se sigue demostrando la necesidad intermitente de la etapa de mutación para los diferentes bancos de instancias como ya se indicó en la Sección 4.3.4. Los mejores valores son los indicados en los diagramas de cajas de la Sección 4.3.3: valores pequeños de mutación (0.010).

- **Desplazamiento de mutación (MS):** los resultados del test de comparaciones múltiples para este factor, véase la Tabla 4.18, son muy similares a los obtenidos para el otro factor de la etapa de mutación,

PS	**Núm. Subconj.**		**Núm. Niveles Subconj. 1**		**Niveles mejor rendimiento**	
Tamaño	Intervalo	Mayor % instancias	Intervalo	Mayor % instancias	Intervalo/s	Mayor % instancias
2^{18}	4-7	7 (45 %)	1-4	1 (55 %)	[10,30];[60,70]	[20,30] (83 %)
2^{19}	4-7	6 (42 %)	1-4	2 (50 %)	40; [60,70]	60 (50 %)
2^{20}	5-7	7 (50 %)	1-3	1 (50 %)	[50,70]	70 (75 %)

Tabla 4.15: Resumen de subconjuntos homogéneos de PS para PBIL.

LR	**Núm. Subconj.**		**Núm. Niveles Subconj. 1**		**Niveles mejor rendimiento**	
Tamaño	Intervalo	Mayor % instancias	Intervalo	Mayor % instancias	Intervalo/s	Mayor % instancias
2^{18}	4-5	4 (55 %)	1-2	1 (59 %)	[0.15-0.25]	[0.20-0.25] (76 %)
2^{19}	4-5	5 (67 %)	1-2	1 (67 %)	[0.15-0.25]	0.25 (58 %)
2^{20}	3-5	5 (58 %)	1-2	1 (67 %)	[0.15-0.25]	0.25 (50 %)

Tabla 4.16: Resumen de subconjuntos homogéneos de LR para PBIL.

probabilidad de mutación, tal y como se podía prever. La influencia del parámetro es muy variable como lo demuestran las fluctuaciones en el número de subconjuntos homogéneos y el número de niveles de una instancia a otra. Como valores destacados quedan los valores más pequeños del factor, siendo la causa la misma que indicábamos para el parámetro anterior.

Podemos observar una semejanza significativa en los resultados obtenidos para los distintos tamaños del problema en cada factor de los algoritmos. La escasa diferencia entre los tamaños para un mismo factor viene determinada por el porcentaje de instancias a las que afecta la variabilidad en los niveles y por los niveles que proporcionan un mayor rendimiento. Ello permite predecir

MP	**Núm. Subconj.**		**Núm. Niveles Subconj. 1**		**Niveles mejor rendimiento**	
Tamaño	Intervalo	Mayor % instancias	Intervalo	Mayor % instancias	Intervalo/s	Mayor % instancias
2^{18}	1-5	5 (31 %)	1-5	1 (38 %)	[0.010,0.050]	0.010 (93 %)
2^{19}	1-5	5 (67 %)	1-5	1 (67 %)	[0.010,0.050]	0.010 (75 %)
2^{20}	1-5	4 (33 %)	1-5	1 (42 %)	[0.010,0.050]	0.010 (83 %)

Tabla 4.17: Resumen de subconjuntos homogéneos de MP para PBIL.

MS	**Núm. Subconj.**		**Núm. Niveles Subconj. 1**		**Niveles mejor rendimiento**	
Tamaño	Intervalo	Mayor % instancias	Intervalo	Mayor % instancias	Intervalo/s	Mayor % instancias
2^{18}	2-5	3 (45 %)	1-4	3 (41 %)	[0.010-0.050]	0.010 (97 %)
2^{19}	2-5	5 (42 %)	1-4	1 (42 %)	[0.010-0.050]	0.010 (75 %)
2^{20}	3-5	3,4 (42 %)	1-3	2 (42 %)	[0.010-0.050]	0.010 (83 %)

Tabla 4.18: Resumen de subconjuntos homogéneos de MS para PBIL.

que existirá muy poca diferencia entre todos los tamaños para los niveles óptimos seleccionados para un determinado factor.

Como hemos comprobado los resultados son coherentes con el análisis unifactorial, Sección 4.3.3, y el análisis multifactorial, Sección 4.3.4.

4.3.6. Selección de niveles estadísticamente óptimos

Conocemos, gracias a los test de comparaciones múltiples, la diferencia existente entre los distintos niveles de cada factor que interviene en los algoritmos. Además, hemos delimitado para cada factor los niveles que proporcionan el mejor rendimiento. Sin embargo, todas estas conclusiones solamente tienen en cuenta la influencia de cada factor por separado. Son, por tanto, resultados orientativos, dada la influencia considerable de los factores individuales en el rendimiento global del algoritmo. Pero, puesto que las distintas combinaciones existentes entre los factores presentan también influencia en tal rendimiento, como se demostró a partir de ANOVA multifactorial en la Sección 4.3.4, será necesario tenerlo en cuenta para seleccionar los niveles óptimos para cada factor del algoritmo.

De esta forma, partiremos de las conclusiones de los test de comparaciones múltiples (influencia unifactorial) y de los resultados de los distintos tratamientos aplicados a cada instancia del algoritmo (influencia multifactorial). Así, para obtener los niveles óptimos para cada factor en función del tamaño del problema, se ha aplicado la *selección basada en el mejor tratamiento*. Para cada instancia del problema se ha seleccionado el tratamiento que proporciona el mejor rendimiento promedio: menor longitud de tiempo de ejecución promedio. Tras ello, se ha obtenido el mejor tratamiento para el banco de instancias correspondiente como el promedio de los niveles de los factores que constituyen los mejores tratamientos para las instancias. El estudio completo y detallado se presenta en [71], [73]. Para otros ejemplos

Tamaño		PS	D	DR	M
2^{18}	Nivel óptimo	20	4	0.30	1
	Desviación típica	15.974	1.567	0.077	0.820
2^{19}	Nivel óptimo	20	4	0.30	1
	Desviación típica	22.344	1.850	0.067	0.793
2^{20}	Nivel óptimo	20	4	0.30	1
	Desviación típica	18.647	1.403	0.062	0.887

Tabla 4.19: Selección de niveles óptimos para CHC.

de aplicación pueden verse [74], [77], [78], [110].

La Tabla 4.19 muestra los resultados de la selección de niveles óptimos basada en el mejor tratamiento para los diferentes factores del algoritmo CHC (tamaño de población (PS), umbral diferencia (D), ratio de divergencia (DR) y mejores individuos (M)) y distinguiendo en función del tamaño analizado. Tal y como fue previsto en los resultados de los test de comparaciones múltiples, la diferencia entre los bancos de instancias viene determinada únicamente por la desviación típica entre los niveles óptimos para cada instancia concreta. El gran parecido entre los intervalos en los que se localizan los niveles óptimos se traduce aquí a una coincidencia de tales niveles para los diferentes tamaños. Como vemos, existe en ciertos casos, principalmente para el tamaño de población y para el umbral diferencia, una ligera desviación con respecto a los valores pronosticados como óptimos por los test de comparaciones múltiples. Ello es debido a que son los parámetros más influyentes en cooperación como demostró ANOVA multifactorial.

Por otra parte, en la Tabla 4.20 se disponen los resultados para los diferentes factores del algoritmo PBIL (tamaño de población (PS), ratio de aprendizaje (LR), probabilidad de mutación (MP) y desplazamiento de mutación (MS)) diferenciándolos en función del tamaño del problema: 2^{18}, 2^{19} y 2^{20}. Para todos los parámetros excepto para el tamaño de población tenemos una diferenciación únicamente debida a la desviación típica entre los niveles óptimos para cada instancia concreta, al igual que ocurre para CHC. Para el tamaño de población se impone la necesidad de un mayor muestreo del espacio de búsqueda a medida que se incrementa el tamaño como se pudo prever a partir de los test de comparaciones múltiples. Los intervalos de localización de los mejores niveles que se obtuvieron en los test *post hoc* se mantienen, aunque aparece el ajuste derivado de la cooperación entre parámetros, véase Sección 4.3.4, en un pequeño desplazamiento de los niveles considerados óptimos. La mayor influencia estadísticamente significativa de las distintas

Tamaño		PS	LR	MP	MS
2^{18}	Nivel óptimo	30	0.20	0.025	0.025
	Desviación típica	22.937	0.070	0.032	0.026
2^{19}	Nivel óptimo	40	0.20	0.025	0.025
	Desviación típica	23.012	0.067	0.031	0.027
2^{20}	Nivel óptimo	50	0.20	0.025	0.025
	Desviación típica	24.664	0.092	0.034	0.033

Tabla 4.20: Selección de niveles óptimos para PBIL.

combinaciones de factores para PBIL con respecto a CHC se demuestra en este punto concreto.

En todo caso, se validan los resultados de los test de comparaciones múltiples. Los niveles óptimos de los factores para los tamaños analizados mantienen la filosofía de la evolución de los algoritmos CHC y PBIL definida respectivamente en [10] y [40]. Sin embargo, es importante resaltar los matices que supone la influencia individual y conjunta de los parámetros con objeto de definir los valores estadísticamente óptimos para los mismos.

4.4. Predicción de parámetros de evolución

El incremento de la complejidad de las instancias del problema de la selección de la solución deseada en resolución de restricciones geométricas supone un aumento considerable del tiempo requerido por las diferentes Metaheurísticas para obtener una solución con calidad para el usuario. En este caso, es más importante aún disponer de unos valores adecuados para los parámetros de evolución de los algoritmos. Sin embargo, el tiempo requerido para el ajuste de los mencionados parámetros ante tamaños considerables del problema es excesivo.

En tal situación, la predicción de los parámetros de evolución de las Metaheurísticas puede suponer un importante avance con objeto de acotar el tiempo de ejecución. Los resultados del ajuste previo de los parámetros de evolución de los algoritmos evolutivos ante un conjunto representativo de instancias del problema pueden servir de base para tal predicción, véase la Sección 4.3.

En esta Sección se definirán posibles expresiones para predecir los parámetros de evolución de las Metaheurísticas CHC y PBIL ante instancias desconocidas del problema. Para ello se seleccionará un modelo simple de predicción:

la regresión lineal simple, [36], [57], [155], partiendo como base de conocimiento del estudio estadístico exhaustivo de los parámetros óptimos de tales algoritmos ante un conjunto, suficientemente representativo, de instancias correspondientes a un conjunto reducido de tamaños del problema.

4.4.1. Diseño procedimental

El objetivo que perseguimos es el de generalizar la aplicación, significativamente óptima desde un punto de vista estadístico, de los algoritmos CHC y PBIL a cualquier instancia del problema de la selección de la solución deseada en resolución de restricciones geométricas. Para ello, partimos de un ajuste paramétrico previo correspondiente a la ejecución de los algoritmos sobre las diferentes instancias de los tamaños del problema 2^{18}, 2^{19} y 2^{20}.

La generalización de la aplicación de los algoritmos se realizará en función del tamaño del problema. Ésta vendrá definida por un conjunto de expresiones que establezcan el marco de ejecución (parámetros del algoritmo). Una de las formas más sencillas de llevar a cabo la generalización comentada vendrá dada por la generalización individual de los parámetros que intervienen en tal marco de ejecución. Realizaremos una generalización individual de dichos parámetros utilizando el modelo de regresión lineal simple, dada su facilidad de aplicación y su significatividad estadística.

Partiendo de la definición del modelo de regresión lineal simple, [33], [36], [57], para cada algoritmo realizaremos un análisis de regresión por cada parámetro que define su marco de ejecución. En cada análisis estudiaremos la relación lineal entre la variable regresora (L), que identificará al problema de espacio de búsqueda 2^L, y la variable respuesta (P), que identificará a cada parámetro concreto. Para cada algoritmo tendremos cuatro análisis de regresión: uno por cada parámetro que interviene en el marco de ejecución.

La muestra $\{(L_i, P_i)\}_{i=1}^{NI}$ de partida vendrá constituida por los resultados óptimos, desde un punto de vista estadísticamente significativo, correspondientes a cada parámetro para los tamaños 2^{18}, 2^{19} y 2^{20}, véase la Sección 4.3. De cada instancia del problema analizada proviene un elemento, óptimo desde un punto de vista estadísticamente significativo para dicha instancia, de la muestra de cada parámetro. El conjunto de instancias está constituido por $NI = 53$ instancias: 29 de tamaño 2^{18} y 12 de cada uno de los tamaños 2^{19} y 2^{20}.

El proceso de análisis de regresión lineal simple seguido para cada parámetro

P correspondiente a cada algoritmo ha sido el siguiente:

- Estudio del cumplimiento previo de las hipótesis básicas de linealidad, homocedasticidad y datos atípicos del modelo de regresión lineal simple a partir de los diagramas de dispersión: L frente a P. Véanse [60] y [149] para conocer más detalles acerca de las representaciones gráficas.
- Aplicación de la transformación simple más adecuada (mayor bondad de ajuste) para la variable respuesta P, para la variable regresora L, o bien en ambas, para garantizar la linealidad en el diagrama de dispersión. Véase [57] para mayor detalle acerca de las posibles transformaciones.
- Estimación de los coeficientes $\hat{\alpha}_{0,P}$ y $\hat{\alpha}_{1,P}$ de la línea de regresión a partir del método de mínimos cuadrados, [130]. Las predicciones para las observaciones muestrales quedarán dadas por la expresión,

$$\hat{P}_i = \hat{\alpha}_{0,P} + L_i\hat{\alpha}_{1,P} \quad \forall i = 1, 2, \ldots, NI \tag{4.4}$$

- Análisis de resultados: contrastes y validación esenciales (sin incluir tratamiento de las hipótesis), [88], [155].
  - Análisis de la bondad del modelo obtenido: cálculo de los coeficientes de determinación (R^2) y de correlación lineal muestral (r).
  - Análisis de significatividad estadística de los coeficientes $\hat{\alpha}_{0,P}$ y $\hat{\alpha}_{1,P}$:
    - Individual: Contrastes de la t, (T-test).
    - Conjunta: Contraste de la F (ANOVA).
  - Cuantificación del error: error estándar de estimación $\hat{s}_R$, error absoluto medio EAM.
- Análisis a posteriori de cumplimiento de hipótesis: análisis de residuos. Véanse [60] y [149] para conocer más detalles acerca de las representaciones gráficas.
  - Normalidad: histograma de residuos estandarizados y gráfico P–P.
  - Linealidad, homocedasticidad y datos atípicos: gráfico de las predicciones frentes a los residuos estandarizados.

Parámetro (P)	Ecuación generalización
Tamaño de población (PS)	$\widehat{PS} = \lfloor 19{,}780700 + 0{,}380564L \rfloor;\ \forall L > 0;\ L \in N$
Umbral diferencia (D)	$\hat{D} = \lfloor 0{,}841461 + 0{,}211591L \rfloor;\ \forall L > 0;\ L \in N$
Ratio de divergencia (DR)	$\widehat{DR} = \text{mín}(0{,}135549 + 0{,}062776 \ln L, 1);\ \forall L > 0;\ L \in N$
M mejores individuos (M)	$\hat{M} = \lfloor 1{,}360900 + 0{,}013641L \rfloor;\ \forall L > 0;\ L \in N$

Tabla 4.21: Generalización individual del marco de ejecución del algoritmo CHC.

- Independencia de las observaciones muestrales: contraste de Durbin-Watson, [6].

A continuación, mostraremos los resultados de la aplicación del análisis de regresión lineal simple para cada uno de los parámetros.

4.4.2. Generalización individual del marco de ejecución

El análisis de regresión y la demostración de la validez del ajuste del modelo para los diferentes parámetros se presenta de forma detallada en [75]. Pasaremos a presentar e interpretar los resultados de la generalización con objeto de utilizarlos en una posterior aplicación de los algoritmos CHC y PBIL sobre instancias desconocidas del problema de la selección de la solución deseada. Comenzaremos con el algoritmo CHC.

La Tabla 4.21 presenta las expresiones de generalización de los parámetros de CHC obtenidas a partir del modelo de regresión lineal simple. En virtud de la definición de cada uno de los parámetros que constituyen el marco de ejecución de CHC, [40], y teniendo en cuenta la estimación de coeficientes en la regresión lineal, la expresión correspondiente a la generalización individual de cada uno de ellos se muestra en la mencionada Tabla y se describe a continuación:

- *Tamaño de la población (PS)*: se trata de un número entero positivo y de ahí que tomemos la parte entera inferior en la ecuación de generalización. El rango de valores sobre el que se mueve no está definido, no es finito. Tal y como se establece en la bibliografía y lo indica la ecuación de generalización, la población siempre es un número entero positivo y se incrementa a medida que aumenta la complejidad del problema. El incremento para este parámetro entre tamaños del problema próximos

es muy reducido, lo que mantiene la coherencia con las características del mismo.

- *Umbral diferencia (D)*: sus valores son números enteros positivos, de ahí que tomemos la parte entera inferior en la ecuación de generalización. Sólo recibe valor en su inicialización, pues en el discurrir del algoritmo, su valor evoluciona tras la reinicialización en función de una expresión. El valor definido como más adecuado en la literatura es de $L/4$. La ecuación de generalización establece que el umbral diferencia siempre es un número entero positivo y se incrementa a medida que aumenta la complejidad del problema. Los valores que recibe en función del tamaño no presentan grandes diferencias con los indicados en la bibliografía, pero sí matices a tener en cuenta.

- *Ratio de divergencia (DR)*: el valor que puede adquirir es real y corresponde al intervalo $[0, 1]$, de ahí que hayamos establecido el límite en 1 a partir de la función mínimo, pues un valor mayor del mismo supone, por parte del algoritmo, la misma interpretación que éste. La ecuación de generalización se mantiene en el dominio definido para el parámetro. Así mismo, establece un incremento del mismo insignificante a medida que aumenta el tamaño del problema, coherente con la definición proporcionada en la bibliografía.

- *M individuos mejores (M)*: en la literatura no hay definido un valor concreto como más adecuado, solamente se establece que este valor no ha de albergar una gran cantidad de cromosomas. A esto precisamente es a lo que obedece la ecuación de generalización con una pendiente de muy reducida magnitud cuyo resultado es siempre un entero positivo y en torno a las sugerencias de la bibliografía.

La Tabla 4.22 presenta, de la misma forma que se mostró para CHC, las expresiones correspondientes al ajuste de la muestra de cada parámetro del marco de ejecución de PBIL a su modelo lineal correspondiente. Teniendo en cuenta la definición de cada uno de los parámetros que constituyen el marco de ejecución de PBIL, [10], y la estimación de coeficientes en la regresión lineal, la expresión correspondiente a la generalización individual de cada uno de ellos se describe a continuación:

- *Tamaño de población (PS)*: la definición de la ecuación de generalización es similar a la del tamaño de población en CHC. Con dicha

Parámetro (P)	Ecuación generalización
Tamaño de población (PS)	$\widehat{PS} = \lfloor 7{,}739150 + 1{,}650500L \rfloor;\ \forall L > 0;\ L \in N$
Ratio de aprendizaje (LR))	$\widehat{LR} = e^{-1{,}633900 - 0{,}002972L};\ \forall L > 0;\ L \in N$
Probabilidad de mutación (MP)	$\widehat{MP} = \text{mín}(0{,}011871 + 0{,}003236\sqrt{L}, 1);\ \forall L > 0;\ L \in N$
Desplazamiento de mutación (MS)	$\widehat{MS} = \text{mín}(0{,}017716 + 0{,}001375\sqrt{L}, 1);\ \forall L > 0;\ L \in N$

Tabla 4.22: Generalización individual del marco de ejecución del algoritmo PBIL.

definición, mantenemos el valor de este parámetro como un entero positivo y, así mismo, un incremento en el valor del mismo con el aumento de la complejidad del problema, tal y como se describe en la bibliografía. Tal incremento es superior al del tamaño de población de CHC, algo coherente con los experimentos anteriores, véase la Sección 4.3.

- *Ratio de aprendizaje (LR)*: los valores preferibles, como puede apreciarse en la bibliografía, son pequeños para permitir una evolución escalonada hacia la mejor solución y evitar la caída en óptimos locales. La ecuación de generalización obtenida mantiene la presencia de valores cada vez más reducidos de este parámetro ante el incremento de complejidad del problema, respetando su dominio.

- *Probabilidad de mutación (MP)*: los valores que toma corresponden al intervalo $[0, 1]$, aunque los valores preferibles, como puede deducirse de lo establecido en la bibliografía, son valores que permitan insertar cada cierto tiempo la diversificación en la búsqueda que permita la evolución ante óptimos locales. A medida que la complejidad del problema se incrementa, la necesidad de diversificación es mayor, por lo que se ha de incrementar progresivamente tal y como lo establece la ecuación de generalización. Para disponer como límite del parámetro al valor 1 hemos insertado la función mínimo, aunque un valor mayor proporcionaría el mismo efecto en el algoritmo.

- *Desplazamiento de mutación (MS)*: su valor también corresponde al intervalo $[0, 1]$, pero los valores muy altos para tamaños pequeños del problema son prohibitivos al enfocar la búsqueda en demasía hacia una región concreta del espacio de búsqueda, tal y como se puede observar en la bibliografía. Con la ecuación de generalización definida, en la que la pendiente de la recta es de magnitud muy reducida y aparece el término tamaño del problema (L) bajo la raíz cuadrada, aseguramos

un crecimiento insignificante del parámetro a medida que se produce un pequeño aumento del tamaño del problema. La introducción de la función mínimo garantiza que mantenemos como valor máximo del parámetro 1, aunque un valor mayor proporcionaría el mismo efecto en el algoritmo.

Partiendo de las ecuaciones de generalización de los parámetros que determinan el marco de ejecución, la predicción de valores para los mismos se realizará mediante la sustitución del tamaño correspondiente a la instancia que se desee resolver. Para conocer detalles acerca de la definición de intervalos de confianza de predicción véase, por ejemplo, [57].

Capítulo 5

Identificación y ajuste de un modelo de rendimiento

La evolución en el proceso de búsqueda de las Metaheurísticas, entre las que se encuentran PBIL y CHC, supone tomar decisiones en las que aparece de uno u otro modo la aleatoriedad. El conocimiento de la variable aleatoria de tiempo de ejecución puede proporcionar información valiosa para el análisis y caracterización del comportamiento del algoritmo, proporcionando indicios para las posibles mejoras de rendimiento y una base adecuada para la comparación de algoritmos.

Para obtener conocimiento empírico de la distribución de tiempo de ejecución (RTD) de un algoritmo cuando es aplicado a una instancia específica de un problema, se puede estimar la RTD a partir de datos obtenidos de varias ejecuciones del algoritmo y probablemente aproximar, mediante técnicas de bondad de ajuste, la RTD observada empíricamente por una función de distribución conocida correspondiente a la teoría de la probabilidad. Si se observa un comportamiento similar en todas las instancias estudiadas correspondientes a un mismo tipo de problema particular, en nuestro caso, el problema de la selección de la solución deseada en resolución de restricciones geométricas, el tipo de RTDs observado caracterizará el comportamiento de tiempo de ejecución en esta clase de problemas. Así mismo, las RTDs pueden dar indicaciones en base a las cuales un algoritmo puede ser mejorado en su aplicación a nuestro problema. Como más tarde se mostrará, las distribuciones de la familia exponencial, y en concreto la distribución Gamma, juegan un papel crucial para poder juzgar la eficiencia y efectividad de los

algoritmos.

En este Capítulo, identificaremos y ajustaremos un modelo de rendimiento para los algoritmos evolutivos CHC y PBIL sobre un conjunto de tamaños del problema. En primer lugar, definiremos una medida de rendimiento independiente de la plataforma de ejecución. Tras ello, se presentarán, describirán y analizarán las distribuciones empíricas que describen la ejecución de cada algoritmo sobre cada instancia del problema. El proceso continuará con el ajuste de las distribuciones empíricas a modelos estadísticos conocidos. Como resultado, se obtendrá una distribución modelo para las diferentes familias de distribuciones correspondiente cada una a un tamaño del problema. El rendimiento de los algoritmos cuando se enfrentan a los tamaños del problema analizados será caracterizado y definido a partir del modelo identificado: el modelo Gamma.

5.1. Medida del rendimiento de un algoritmo: RTD vs. RLD

PBIL y CHC hacen uso de la aleatoriedad frecuentemente durante sus ejecuciones. Por una parte, PBIL está basado en la evolución de un vector de probabilidades que representa a una población de soluciones que trata de aproximarse a la región del espacio de búsqueda en la que se sitúa la solución óptima. Por la otra, CHC parte de una población aleatoria de soluciones y hace que intervenga la aleatoriedad en etapas como el cruce y la reinicialización. La medida y observación de la evolución del tiempo de ejecución es potencialmente muy útil para analizar el rendimiento de ambos algoritmos.

El tiempo de ejecución es una variable aleatoria y obtendremos conocimiento acerca de su distribución tomando empíricamente muestras de tal variable aleatoria al ejecutar cada algoritmo varias veces. Basándonos en tales muestras podremos hacer suposiciones sobre el tiempo de ejecución. Estas suposiciones pueden ser validadas a través de test de hipótesis estadísticas y en caso de que las suposiciones no sean respaldadas por los datos, éstas pueden ser identificadas y rechazadas.

La distribución del tiempo de ejecución (RTD) de las Metaheurísticas depende de la instancia del problema. Los espacios de búsqueda presentados por cada instancia son diferentes y de ahí la distinta evolución del algoritmo sobre los mismos. Por tanto, las RTDs deben ser estimadas para cada instancia del problema para detectar diferencias en el comportamiento de

tiempo de ejecución a lo largo del conjunto de instancias. Sin embargo, tal estimación individual no impide que las conclusiones que se puedan extraer determinen el tipo de distribución de tiempo de ejecución correspondiente al problema global, tal y como se demuestra en [70]. Tales conclusiones pueden ser formuladas y comprobadas con las instancias que se desee.

En vez de medir las distribuciones de tiempo de ejecución en términos de tiempo de CPU, con frecuencia es preferible usar contadores representativos de operaciones del algoritmo como medida del rendimiento del mismo que resulta ser más independiente de la plataforma de ejecución, tal y como se establece en [3]. Así, usando un modelo de coste apropiado de las operaciones del algoritmo, ya que los contadores de operación se pueden transformar fácilmente en tiempos de ejecución, se facilita la comparación de algoritmos entre diferentes arquitecturas. Hablamos entonces de *distribuciones de longitud de ejecución* (RLDs) en vez de RTDs. Desde aquí en adelante, por tanto, sólo hablaremos de RLDs, teniendo en cuenta que hablar de las mismas supone caracterizar de la misma forma a las RTDs.

En nuestro caso, para el problema de la selección de la solución deseada dentro de la resolución de restricciones geométricas, se ha seleccionado como contador de operación representativo para las RLDs al número de evaluaciones de soluciones generadas durante la búsqueda (longitud de ejecución), véase la Sección 3.5. Se ha de tener en cuenta que, en el problema que abordamos, el coste de la evaluación de soluciones, verificar el número de restricciones adicionales que cumplen, supone un elevado porcentaje sobre el tiempo total de una iteración del algoritmo.

Para medir las RLDs se ha de tener en cuenta que la mayoría de algoritmos presentan un parámetro que limita el tiempo de ejecución. En nuestro caso, tal parámetro se corresponderá con el número máximo de evaluaciones de soluciones permitido. En la práctica, las RLDs empíricas se miden al ejecutar el algoritmo durante n_r veces sobre una instancia de un problema dada, con la limitación del parámetro anteriormente señalado. Para cada ejecución se almacena el número de pasos necesarios hasta encontrar la solución. En nuestro caso, una ejecución con éxito es aquella que proporciona una solución que cumple todas las restricciones adicionales definidas sobre la figura de partida, véase el Capítulo 3. El número de pasos necesarios para encontrar la solución se convierte aquí en el número de evaluaciones de soluciones necesarias hasta encontrar la solución óptima a la instancia del problema proporcionada.

La RLD empírica es la distribución acumulativa asociada a las observaciones

o muestras tomadas. La obtención de la misma se realiza a través de un simple cálculo. Formalmente, sea $rl(j)$ la longitud de ejecución para la ejecución j-ésima con éxito, entonces la RLD empírica acumulativa se define como sigue

$$\hat{P}(rl \leq i) = \mid \{j \parallel rl(j) \leq i\} \mid /n_r \tag{5.1}$$

5.2. Diseño experimental de las RLDs

Se desea estudiar el rendimiento de los algoritmos PBIL y CHC a través de las RLDs sobre el problema de la selección de la solución deseada dentro de la resolución de restricciones geométricas. Tales algoritmos fueron optimizados para un banco de instancias representativo de dicho problema de los tamaños 2^{18}, 2^{19} y 2^{20}, véase la Sección 4.3.

Utilizaremos, por tanto, los algoritmos optimizados para realizar el estudio de las RLDs para tal banco de instancias, [159]. Ambos algoritmos serán tratados bajo las mismas condiciones. Tendremos un total de 2500 ejecuciones de cada algoritmo sobre cada una de las instancias: 50 observaciones por 50 ejecuciones por observación. Se utilizará un único tratamiento, el establecido como estadísticamente óptimo en la mencionada Sección.

Como parámetro que controla la longitud del tiempo de ejecución tenemos al número de soluciones evaluadas. El parámetro que limita el fin de una ejecución sin haber tenido éxito se establecerá en 30000 evaluaciones, tal y como ha ocurrido en el estudio anterior para el ajuste de parámetros. Se ha establecido tal valor para asegurar que los algoritmos muestran un comportamiento asintótico en base a su rendimiento a lo largo de la longitud del tiempo de ejecución.

Una vez realizado el diseño de los experimentos, se han construido las RLDs empíricas para ambos algoritmos sobre cada una de las instancias del problema. Tras analizar las RLDs empíricas por separado para cada algoritmo, se han contrastado conjuntamente para observar la evolución en el rendimiento de los algoritmos y las posibles mejoras o modificaciones que se podrían hacer sobre cada uno de ellos.

Para cada RLD empírica se ha obtenido la correspondiente función de distribución acumulativa teórica asociada a una distribución estadística continua teórica conocida. Se han utilizado para ello tests estadísticos de bondad de ajuste gráficos y analíticos. Finalmente, se han comprobado los resultados

CHC vs. PBIL:$\hat{E}(RL)$			
Tamaño	**Mejor**	**Media Mejor Trat.**	**Desv. Típ. Mejor Trat.**
2^{18}	83 %	76 %	86 %
2^{19}	100 %	92 %	100 %
2^{20}	75 %	75 %	92 %

Tabla 5.1: Porcentaje de instancias en que CHC supera a PBIL.

empíricos y las funciones de distribución acumulativa asociadas de forma conjunta para ambos algoritmos: PBIL y CHC.

5.3. RLDs empíricas

Si analizamos los resultados proporcionados por CHC y PBIL para los tamaños del problema 2^{18}, 2^{19} y 2^{20}, [71], [73], podemos observar como la media del número de evaluaciones para alcanzar una solución óptima del problema es menor para CHC en más del 75 % de las instancias. En lo que corresponde a la desviación típica, CHC presenta menor variabilidad que PBIL en más del 86 % de las instancias. En la Tabla 5.1 mostramos, como resumen de la aplicación del T-test, [137], cómo CHC supera claramente a PBIL en cuanto a mejor, media y variabilidad de eficiencia de ejecución ante las diferentes instancias del problema.

A la vista de estos resultados se podría decir que CHC supera ampliamente a PBIL en los resultados. Sin embargo, tales resultados no tienen en cuenta la evolución en el rendimiento de los algoritmos cuando son aplicados a cada instancia del problema. Sólo muestran un resultado final en el que no se refleja la posible mejora de los algoritmos ni tampoco una comparación de los algoritmos en base a una determinada calidad de solución requerida. Así mismo, tampoco permiten realizar una caracterización general de la evolución y rendimiento de los algoritmos sobre los diferentes espacios de búsqueda.

Para comparar y analizar la evolución del rendimiento de los algoritmos ante las diferentes instancias se han elaborado, por tanto, las RLDs empíricas para todas y cada una de las instancias.

La Figura 5.1 presenta como muestra dos de las RLDs empíricas resultantes de la aplicación de CHC y PBIL respectivamente a una instancia aleatoriamente seleccionada del banco de instancias. Para cada RLD, el eje X presenta

CHC vs PBIL	**CHC domina a PBIL** (% inst.)	**PBIL supera a CHC** (% instancias)			
Tamaño	Ejecución completa	Etapa inicial	Etapa intermedia	Etapa final	$p_{exito} = 1$
2^{18}	17	31	7	45	6
2^{19}	17	33	42	8	0
2^{20}	8	33	8	51	17

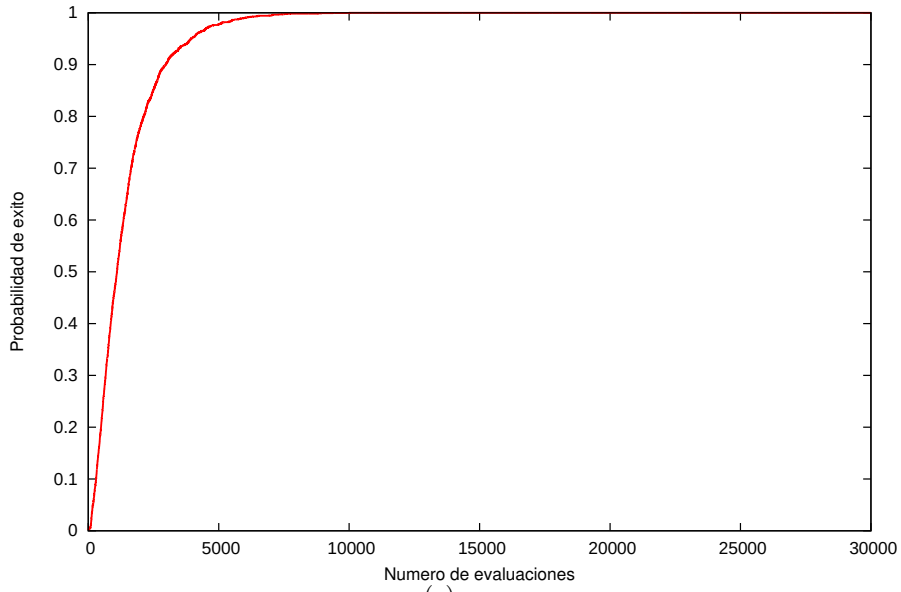
Tabla 5.2: Análisis comparativo de las RLDs empíricas de CHC y PBIL.

el número de soluciones evaluadas durante la ejecución del algoritmo y el eje Y indica la probabilidad de obtener una solución óptima para la instancia correspondiente.

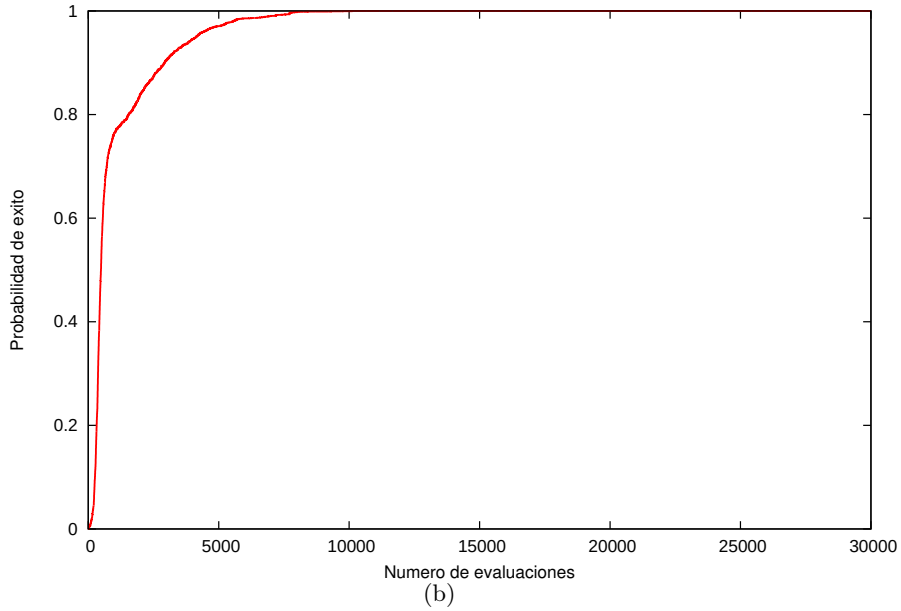
En todas las instancias, prácticamente en el 90 %, observamos un pronunciado y vertiginoso ascenso, sin apenas oscilaciones, hacia la solución óptima que comienza inmediatamente en las primeras evaluaciones. Permanecemos muy lejos del límite establecido como número de evaluaciones máximo permitido. En ese ascenso no se observan apenas irregularidades en la curva, lo que indica que el algoritmo encuentra pocos obstáculos en la búsqueda o que pocos de ellos son difíciles de superar: óptimos locales. La excepción la establecen ciertas instancias, en las que el ascenso es mucho más suave y donde se necesita un mayor número de evaluaciones para garantizar el éxito, [71], [72]. En todo caso, la longitud de ejecución necesaria para obtener una misma probabilidad de éxito se incrementa a medida que lo hace el tamaño del problema.

La Tabla 5.2 presenta un análisis comparativo de las RLDs empíricas de CHC y PBIL para los diferentes tamaños del problema. En la misma puede comprobarse que el algoritmo CHC domina a PBIL sólo para un escaso porcentaje de instancias en todos los tamaños del problema tratados: dado cualquier punto (rl_i, p_{chc}) de la RLD empírica de CHC, siendo (rl_i, p_{pbil}) el punto correspondiente de la RLD empírica de PBIL, ocurrirá que $p_{chc} > p_{pbil}$, véase un ejemplo en la Figura 5.2. Por tanto, para una mayoría de las instancias, tal y como se establece en la mencionada Tabla, PBIL supera a CHC en algún momento concreto de la ejecución: etapa inicial, etapa intermedia o etapa final. Sin embargo, el algoritmo PBIL sólo es capaz de mantener su supremacía sobre CHC para la obtención de la probabilidad máxima de éxito para un porcentaje insignificante de las instancias, tal y como muestra la última columna ($p_{exito} = 1$).

Hemos contrastado que la supremacía de CHC sobre PBIL no es tan notoria



(a)



(b)

Figura 5.1: RLDs empíricas para una misma instancia de a) CHC y b) PBIL.

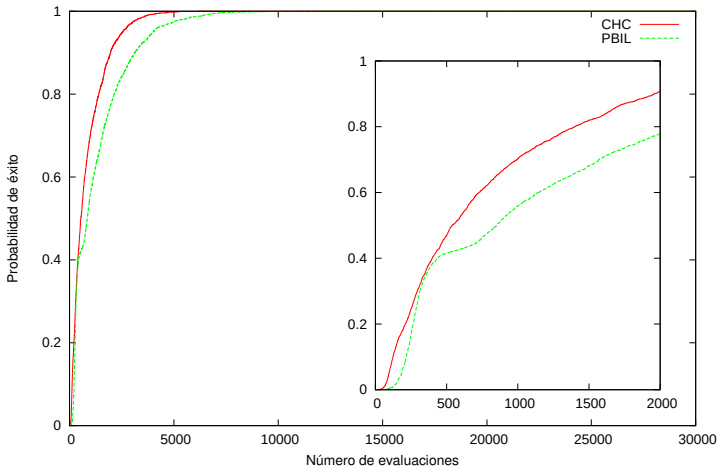


Figura 5.2: Dominancia de CHC sobre PBIL.

como en un principio podría parecer. Queda claro que los resultados finales de la búsqueda favorecen a CHC. Pero, cabe resaltar el rendimiento y evolución de PBIL en la búsqueda: la irregularidad está muy presente y CHC queda en un segundo plano para ciertos intervalos de probabilidad de éxito, intervalos de relevancia para prácticamente la mitad de las instancias tratadas. Los resultados de las RLDs empíricas sugieren, en primer lugar, que ninguno de los algoritmos supera claramente al otro. En segundo lugar, se hace patente una deficiencia considerable de PBIL para escapar del estancamiento, tanto prematuro como en etapas finales donde se dispone de óptimos locales de entidad. Dicha deficiencia se incrementa con el tamaño del problema.

Los resultados para PBIL invitan a la incorporación de mecanismos que introduzcan diversificación en la búsqueda en etapas avanzadas de la misma y permitan escapar de los óptimos locales que dificultan el rendimiento. Ello podría suponer la obtención de curvas más homogéneas para las RLDs empíricas y en la misma línea, mejores resultados para probabilidades de éxito elevadas.

Incluso se podría pensar en la colaboración de ambos algoritmos de cara a la obtención de una solución de garantías en menor tiempo. Así, el punto de partida lo podríamos tener en PBIL de cara a una explotación inicial eficiente del espacio de búsqueda con una selección de las regiones más prometedoras del mismo. Una vez que PBIL tuviese tendencia al estancamiento podría aplicarse CHC para mantener regularidad y equilibrio entre diversidad y

convergencia en la búsqueda.

5.4. Ajuste de RLDs empíricas a distribuciones estadísticas

Con objeto de caracterizar de forma general la evolución y rendimiento de los algoritmos PBIL y CHC ante el conjunto completo de las instancias del problema que tratamos de resolver se han realizado una serie de test gráficos y analíticos para comprobar el ajuste de las RLDs empíricas a distribuciones estadísticas continuas teóricas conocidas. Para realizar tal proceso se han de aplicar métodos y técnicas de *bondad de ajuste*, [33], [60], [136], teniendo en cuenta las características de la distribución de los datos observados que se utilizaron para elaborar las RLDs.

En nuestro caso, la variable aleatoria con que contamos la conforman los valores de tiempos de ejecución de los algoritmos para las distintas instancias del problema. Tales valores, tal y como se indicó en la Sección 5.1, corresponden al número de evaluaciones necesarias para alcanzar una solución válida para la instancia correspondiente de nuestro problema. Así mismo, para cada instancia tenemos la RLD asociada a los resultados observados tras las ejecuciones.

Si comprobamos la definición de RLD presente en la Ecuación 5.1 vemos que equivale a la definición de función de distribución acumulativa de las distribuciones estadísticas, [33]. La distribución de probabilidad de una variable aleatoria X puede ser únicamente descrita por su función de distribución acumulativa $F(x)$, que se define como sigue:

$$F(x) = Pr[X \leq x] \tag{5.2}$$

para cualquier x en $\mathbb{R}$.

De esta forma, la verificación de si los resultados de los tiempos de ejecución necesarios para cada instancia se corresponden con una determinada distribución estadística equivaldrá a comprobar que las RLDs empíricas se ajustan a la función de distribución acumulativa de tal distribución estadística. Dada la naturaleza continua del tiempo, las distribuciones estadísticas con las que trataremos serán distribuciones estadísticas continuas.

Disponemos de un conjunto de observaciones para cada instancia del pro-

blema que corresponden a la longitud de ejecución (número de evaluaciones) hasta encontrar una solución válida (solución que cumple todas las restricciones adicionales definidas en el croquis de partida) para las diferentes ejecuciones de los algoritmos CHC y PBIL. Con tales observaciones se han elaborado las RLDs empíricas para CHC y PBIL en su aplicación sobre cada instancia, véase la Sección 5.3.

Puesto que las longitudes de tiempos de ejecución tienen una transformación directa a tiempos de ejecución, tal y como se demuestra en la Sección 5.1, podemos hablar de que disponemos de muestras independientes de una variable aleatoria continua asociada al tiempo de ejecución.

Una vez caracterizada y analizada la evolución del rendimiento de los algoritmos a través de las RLDs empíricas, nos planteamos generalizar tales RLDs intentando ajustar los datos observados a una distribución estadística continua conocida. Hablando en términos de longitudes de tiempos de ejecución representados a partir de una RLD, deseamos ajustar las RLDs empíricas a una función de distribución acumulativa correspondiente a una distribución estadística teórica continua conocida.

5.4.1. Diseño procedimental

Se ha seleccionado un grupo de distribuciones estadísticas entre las distribuciones estadísticas continuas más conocidas para verificar el ajuste de las RLDs empíricas a las mismas. La elección de las distribuciones de probabilidad se ha realizado teniendo en cuenta las características y naturaleza de los datos de los que partimos: tiempos de ejecución de los algoritmos. Tales distribuciones, descritas en [33], son las siguientes: exponencial, Weibull, Gamma, Erlang, normal, Laplace y Cauchy.

El objetivo es el de aplicar tanto técnicas gráficas, [60], como cuantitativas, [136], de bondad de ajuste y seleccionar aquella distribución o distribuciones que permitan caracterizar las RLDs correspondientes a las distintas instancias para ambos algoritmos: CHC y PBIL. Las técnicas gráficas que hemos aplicado son los gráficos P–P. Las técnicas cuantitativas seleccionadas han sido: test Chi-cuadrado, [138], test de Kolmogorov–Smirnov, [26], y test de Anderson–Darling, [143].

Para facilitar la aplicación de las técnicas cuantitativas de bondad de ajuste seleccionadas, partiremos de 50 observaciones para cada instancia en vez de las 2500 observaciones de que disponemos en un principio. Para obtener tales

observaciones se ha realizado un muestreo aleatorio sobre el conjunto global de observaciones o longitudes de tiempo de ejecución.

Como paso previo a la aplicación de los diferentes tests de bondad de ajuste nos hemos asegurado que se cumplen los supuestos sobre las muestras que cada test establece (véanse [26], [60], [138] y [143] para mayor detalle). Tales supuestos se refieren principalmente a la aleatoriedad e independencia de las muestras, al tamaño muestral y a la necesidad de una estimación previa de los parámetros (estimación de máxima verosimilitud, [5], en nuestro caso) de la distribución subyacente a los datos en base a la distribución seleccionada, en cada momento, para la bondad de ajuste.

El proceso seguido para la aplicación de los tests de bondad de ajuste para los resultados obtenidos en cada algoritmo, CHC y PBIL, ha sido el siguiente:

- Para cada instancia $I_i (i = 1, \ldots, NI)$ del problema, siendo NI el número total de instancias,
  - Selección aleatoria de un conjunto M_i de muestras de longitud de ejecución del conjunto global de longitudes de ejecución. En nuestro caso $|M_i| = 50$.
- Para cada instancia I_i (i=1,...,NI) del problema,
  - Para cada distribución estadística continua teórica $D_j (j = 1, \ldots, ND)$ del grupo seleccionado, siendo ND el número total de distribuciones elegidas,
    - Estimación de los parámetros $\mathcal{P}_{ij} = (P_{ij}(1), \ldots, P_{ij}(n))$ de la distribución D_j, siendo n el número total de parámetros de la misma. Suponemos D_j como distribución subyacente a los datos de longitud de tiempo de ejecución M_i correspondientes a la instancia I_i. El método utilizado ha sido el método de estimación por máxima verosimilitud.
    - Aplicación del test Chi-cuadrado, considerando la distribución D_j, a las observaciones de la muestra M_i tomando $\mathcal{P}_{ij}$ como parámetros estimados para la distribución D_j. Si el resultado de la aplicación del test no permite rechazar la hipótesis nula a un nivel de significación α (en nuestro caso, 0,05) seleccionaremos esta distribución dentro del conjunto de distribuciones que se ajustan a la muestra M_i según el test Chi-cuadrado: D_Chi_i.

- Aplicación del test de Kolmogorov-Smirnov, considerando la distribución D_j, a las observaciones de la muestra M_i tomando $\mathcal{P}_{ij}$ como parámetros estimados para la distribución D_j. Si el resultado de la aplicación del test no permite rechazar la hipótesis nula a un nivel de significación α (en nuestro caso, 0,05) seleccionaremos esta distribución dentro del conjunto de distribuciones que se ajustan a la muestra M_i según el test de Kolmogorov-Smirnov: D_KS_i.
- Aplicación del test de Anderson-Darling, considerando la distribución D_j, a las observaciones de la muestra M_i tomando $\mathcal{P}_{ij}$ como parámetros estimados para la distribución D_j. Si el resultado de la aplicación del test no permite rechazar la hipótesis nula a un nivel de significación α (en nuestro caso, 0,05) seleccionaremos esta distribución dentro del conjunto de distribuciones que se ajustan a la muestra M_i según el test de Anderson-Darling: D_AD_i.

• Selección de las distribuciones para las que los tests no pudieron rechazar la hipótesis nula, con un nivel de significación α, de que los datos que conforman la muestra M_i se ajustan a las mismas. Para ello utilizaremos los resultados obtenidos por los tests cuantitativos de bondad de ajuste: test Chi-cuadrado (D_Chi_i), test de Kolmogorov-Smirnov (D_KS_i) y test de Anderson-Darling (D_AD_i). El conjunto seleccionado (D_sel_i) estará constituido por las distribuciones comunes en los resultados de la aplicación de los tests: $D\_Chi_i \cap D\_KS_i \cap D\_AD_i$.

▪ Selección de la distribución o distribuciones que caracterizan la aplicación del algoritmo (CHC o PBIL) sobre las instancias del problema. Este conjunto (DI) estará constituido por las distribuciones comunes a todas las instancias del problema de las seleccionadas (D_sel_i) para cada una de ellas: $\cap_{i=1}^{NI} D\_sel_i$.

▪ Verificación a partir de tests gráficos de bondad de ajuste de los resultados obtenidos en los tests cuantitativos. Para cada instancia I_i (i=1,...,NI) del problema,

• Para cada distribución presente en el conjunto DI,

- Realización del gráfico P–P para la muestra M_i y verificación visual de la bondad del ajuste.

Al final del proceso dispondremos, para cada algoritmo, de una distribución o conjunto de distribuciones que se ajustan a las RLDs empíricas que definen la evolución de sus ejecuciones sobre las instancias de nuestro problema. Tales distribuciones caracterizarán el rendimiento de cada uno de los algoritmos en su aplicación al tamaño del problema al que pertenecen las instancias. El proceso de obtención de tales distribuciones permitirá aunar los resultados de los tests cuantitativos más comunes de bondad de ajuste con un contraste gráfico que los verifique visualmente, permitiendo eliminar los resultados provenientes de las particularidades de cada test concreto.

5.4.2. Resultados

Las Tablas 5.3 y 5.4 presentan los resultados de la aplicación de los métodos cuantitativos de bondad de ajuste: Chi–Cuadrado (Chi), Kolmogorov–Smirnov (K-S) y Anderson–Darling (A-S) para los diferentes tamaños del problema analizados. Para cada distribución estadística teórica continua se muestra el porcentaje de instancias de cada tamaño que se ajustan a la misma a partir de cada uno de los mencionados métodos cuantitativos. Como distribuciones destacadas aparecen las distribuciones Gamma y Weibull. De ambas, las técnicas cuantitativas establecen a la función de distribución estadística continua Gamma como aquella que se ajusta a la longitud de ejecución de los algoritmos PBIL y CHC puesto que supera los tests para todas las instancias sin excepción. Para ello, hemos garantizado que los tests no rechazan la hipótesis nula de que tal función de distribución se ajusta a las muestras de longitud de ejecución de los algoritmos con un 95 % de confianza. Este es el resultado fundamental de las técnicas cuantitativas y que permite generalizar la asociación de una función de distribución estadística teórica continua para la distribución del tiempo de ejecución de los algoritmos sobre nuestro problema.

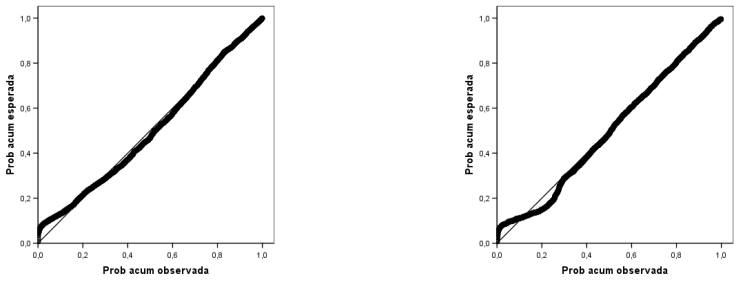
Tras aplicar las técnicas cuantitativas de bondad de ajuste hemos contrastado sus resultados a partir de métodos gráficos de bondad de ajuste. En nuestro caso hemos seleccionado los gráficos P–P, que muestran proporciones acumulativas de la variable correspondiente a los datos observados contra las proporciones acumulativas de la distribución a la que se desea comprobar si pertenecen tales datos, [136]. Por tanto, lo que representaremos en los gráficos P–P, para cada instancia y algoritmo, serán las proporciones acumulativas de la variable correspondiente a los datos observados (longitudes de ejecución) contra las proporciones acumulativas de la distribución Gamma, que es a la que se desea contrastar si pertenecen tales datos. Tomaremos

Bondad de ajuste **(% instancias)**									
Tamaño	2^{18}			2^{19}			2^{20}		
Dist. \ Mét.	Chi	K-S	A-D	Chi	K-S	A-D	Chi	K-S	A-D
Cauchy	17	79	72	8	75	58	8	75	67
Erlang	89	34	37	83	50	42	75	67	50
Exponencial	20	48	44	42	50	50	33	67	58
Gamma	100	100	100	100	100	100	100	100	100
Laplace	44	75	72	33	75	75	8	67	58
Normal	27	72	62	25	58	33	17	67	50
Weibull	72	93	93	75	92	92	75	100	92

Tabla 5.3: Resultados de métodos cuantitativos de bondad de ajuste para CHC.

Bondad de ajuste **(% instancias)**									
Tamaño	2^{18}			2^{19}			2^{20}		
Dist. \ Mét.	Chi	K-S	A-D	Chi	K-S	A-D	Chi	K-S	A-D
Cauchy	58	58	65	8	33	25	8	50	50
Erlang	0	0	0	58	25	33	75	42	42
Exponencial	37	37	37	50	83	83	42	58	58
Gamma	100	100	100	100	100	100	100	100	100
Laplace	68	68	68	0	42	75	33	58	58
Normal	55	55	48	8	50	25	33	58	50
Weibull	93	93	86	50	100	100	83	92	92

Tabla 5.4: Resultados de métodos cuantitativos de bondad de ajuste para PBIL.



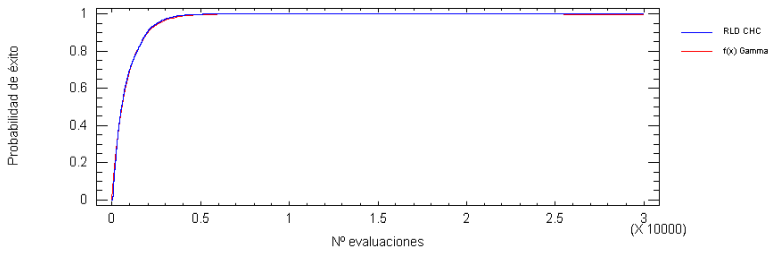
(a) (b)

Figura 5.3: Ejemplos de Gráficos P-P para el ajuste de la distribución Gamma para a) CHC y b) PBIL.

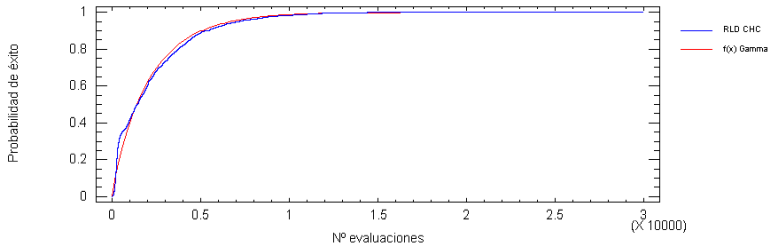
como datos observados las 2500 ejecuciones de las que se partieron, para cada instancia y algoritmo, para elaborar las RLDs empíricas, véase la Sección 5.3.

La Figura 5.3 presenta un ejemplo representativo, para una instancia seleccionada aleatoriamente, de los gráficos P-P elaborados para cada instancia. En el eje X se muestra la probabilidad acumulada observada correspondiente a los datos de longitud de tiempo de ejecución de la aplicación de los algoritmos y en el eje Y se dispone la probabilidad esperada correspondiente a la distribución estadística Gamma. Tal y como se establece en [136], si los datos se ajustan realmente a la distribución seleccionada conforman una línea recta. Esto es precisamente lo que ocurre de forma generalizada para la totalidad de instancias y ambos algoritmos, [71], [72], aunque se observan ligeras variaciones en el extremo inferior para algunas de ellas. Sin embargo, estas desviaciones pueden explicarse por la fase inicial de ascensión de colinas que presentan ambos algoritmos, [104], puesto que, intuitivamente, cada algoritmo necesita un tiempo para alcanzar una posición en el espacio de búsqueda para la que exista realmente una probabilidad adecuada de encontrar una solución válida para la instancia concreta del problema que resuelve.

Teniendo en cuenta los resultados obtenidos en los tests cuantitativos de bondad de ajuste y el contraste visual realizado a partir de los tests gráficos, podemos garantizar que la función de distribución estadística teórica Gamma se ajusta a las longitudes de tiempo de ejecución obtenidas como resultado de aplicar los algoritmos PBIL o CHC al banco de instancias del problema. De ahí que podamos concluir que la función de distribución acumulativa de la distribución Gamma se ajusta a las RLDs empíricas resultantes de la



(a)



(b)

Figura 5.4: Ejemplos de ajuste de las RLDs empíricas con Gamma: a) CHC y b) PBIL.

aplicación de los algoritmos a las instancias.

La Figura 5.4 muestra, para la aplicación de CHC y PBIL respectivamente, el correspondiente ajuste de la RLD empírica a la función de distribución acumulativa de la distribución Gamma ajustada para una instancia seleccionada aleatoriamente. Puede comprobarse cómo en el ajuste, [71], [72], para todos los casos y tal como ha sido demostrado, existe un solapamiento importante de las curvas que permitirá extrapolar adecuadamente las conclusiones que se puedan extraer de la función acumulativa de la distribución Gamma a la RLD empírica asociada. Así mismo, permitirá extrapolar las conclusiones a extraer de la función de distribución Gamma para las longitudes de tiempo de ejecución de cada uno de los algoritmos sobre las instancias del problema.

Los resultados obtenidos en el análisis de las RLDs empíricas en la Sección 5.3 son aplicables a las funciones de distribución acumulativas correspondientes

a la distribución Gamma asociada a cada instancia concreta y algoritmo. Es más, una de las características asociadas al comportamiento de ambos algoritmos que se ve reflejada de forma más notoria en la evolución de PBIL, como es el estancamiento en óptimos locales en las etapas finales de la búsqueda, puede verificarse en el ajuste de las RLDs empíricas a la función de distribución acumulativa de la distribución Gamma. El comportamiento ideal en la evolución del algoritmo en la búsqueda está establecido por la función de distribución acumulativa F(x) de la distribución Gamma y es a lo que se debe intentar aproximar, sobre todo, el algoritmo PBIL, tan carente de solvencia ante óptimos locales de entidad.

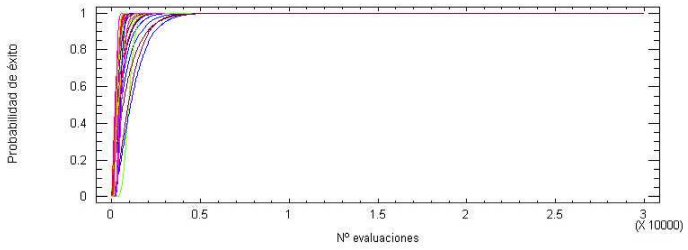
Podemos obtener a partir de la distribución Gamma estimada para cada instancia concreta los diferentes parámetros que caracterizan la evolución de cada uno de los dos algoritmos sobre dicha instancia.

5.5. Unificación del modelo de comportamiento

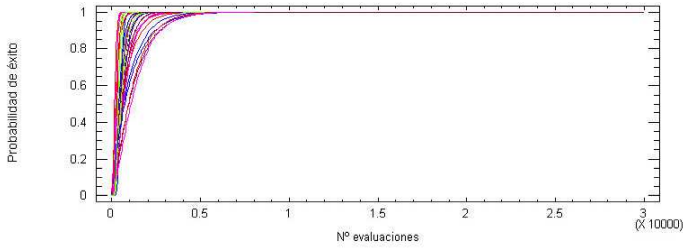
Conocemos el conjunto de niveles óptimos de los factores que influyen en el rendimiento de los algoritmos CHC y PBIL en su aplicación al problema de la selección de la solución deseada para los tamaños del problema 2^{18}, 2^{19} y 2^{20}, véase la Sección 4.3. Así mismo, en la Sección 5.4 hemos demostrado que podemos caracterizar el rendimiento de los algoritmos en su ejecución sobre cada una de las instancias a través de la distribución Gamma, puesto que su función de distribución acumulativa se ajusta de forma estadísticamente significativa con las RLDs empíricas.

Sin embargo, para cada instancia concreta del problema y algoritmo disponemos de una distribución Gamma que caracteriza la evolución del algoritmo. Para poder predecir y generalizar el comportamiento de los algoritmos para instancias desconocidas se hace necesario disponer de una sola representación del rendimiento de cada algoritmo para cada tamaño del problema. Puesto que disponemos de una única representación para los parámetros de los algoritmos (un único tratamiento para cada tamaño del problema) hemos de obtener un única distribución para cada tamaño capaz de aproximar y, por tanto, caracterizar la evolución de los algoritmos.

Tal distribución se puede obtener y representar tanto de forma gráfica (representación gráfica de la función de distribución acumulativa o RLD) como de forma paramétrica (parámetros de la distribución representante y ecuaciones que definen tal distribución).



(a)



(b)

Figura 5.5: RLDs Gamma de a) CHC y b) PBIL para el tamaño 2^{18}.

5.5.1. Representación gráfica

La Figura 5.5 presenta, respectivamente para CHC y PBIL, la representación gráfica conjunta para el tamaño del problema 2^{18} de las distintas RLDs Gamma (funciones de distribución acumulativa de las correspondientes distribuciones Gamma) que se ajustan a las RLDs empíricas para las diferentes instancias. Podemos apreciar cómo para cada algoritmo las RLDs Gamma presentan una gran similitud tanto en la localización como en la evolución de la curva. Las correspondientes a los tamaños 2^{19} y 2^{20} se presentan en el Apéndice B. La obtención de una RLD Gamma representativa para cada tamaño y algoritmo a partir de las RLDs Gamma asociadas a cada banco de instancias supondría una aproximación adecuada para generalizar el rendimiento y evolución de cada algoritmo para cada tamaño del problema.

La RLD Gamma promedio que resuma las RLDs Gamma correspondientes a un conjunto de instancias para un determinado algoritmo y tamaño del

problema habrá de tener su origen, al igual que cada una de tales RLDs, en una RLD empírica promedio. Dicha RLD empírica promedio provendrá del promedio de las RLDs empíricas individuales correspondientes a las distintas instancias del tamaño del problema. Formalmente, teniendo en cuenta la Ecuación 5.1, la RLD empírica representativa de un tamaño del problema L (representando un espacio de búsqueda limitado por 2^L soluciones) para un conjunto de NI instancias se define como sigue:

$$\hat{P}_L(rl \leq i) = \frac{1}{NI} \sum_{I=1}^{NI} \hat{P}_I(rl \leq i) \tag{5.3}$$

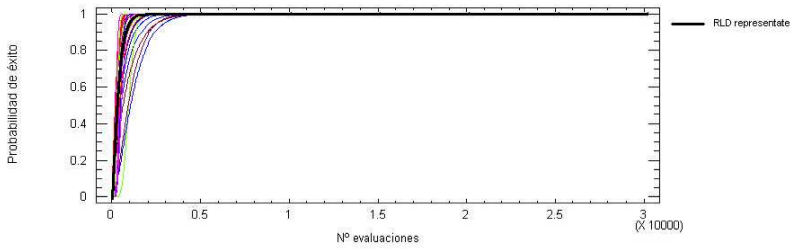
A partir de la RLD empírica promedio estimaremos la RLD Gamma promedio mediante el método de máxima verosimilitud, [5], partiendo de un conjunto de observaciones de longitud de ejecución obtenido a partir de un muestreo uniforme de dicha RLD empírica promedio, teniendo presente la Ecuación 5.1. La cantidad de observaciones será la misma que se utilizó para la estimación de la RLD Gamma de cada instancia en la Sección 5.4: 50 ejecuciones. Finalmente, el ajuste de la RLD Gamma promedio a la RLD empírica promedio será contrastado mediante las técnicas cuantitativas de bondad de ajuste usadas en la Sección anteriormente citada.

El resultado final del proceso, detallado en [71], para cada tamaño del problema (RLD Gamma representativa junto a las RLDs Gamma correspondientes a las diferentes instancias que constituyen la familia a la que la primera identifica) se muestra en la Figura 5.6 y en el Apéndice B. La primera presenta lo correspondiente al tamaño del problema 2^{18} y los dos algoritmos analizados. En el Apéndice se dispone lo correspondiente a los tamaños del problema 2^{19} y 2^{20}.

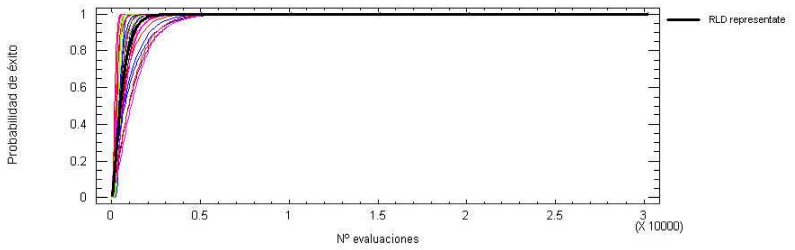
5.5.2. Representación paramétrica

Antes de presentar la representación paramétrica de las RLDs Gamma para cada tamaño del problema considerado es necesario conocer teóricamente la definición paramétrica de la distribución Gamma, [33].

La distribución Gamma corresponde a una familia de distribuciones continuas de probabilidad de dos parámetros que representa la suma de k variables aleatorias exponencialmente distribuidas teniendo cada una de las cuales media θ.



(a)



(b)

Figura 5.6: RLDs Gamma representantes de a) CHC y b) PBIL para el tamaño 2^{18}.

La función de densidad de probabilidad de la distribución gamma puede ser expresada en términos de la función gamma:

$$f(x; k, \theta) = x^{k-1} \frac{e^{-x/\theta}}{\theta^k \, \Gamma(k)} \text{ for } x > 0 \tag{5.4}$$

donde $k > 0$ es el parámetro de forma, $\theta > 0$ es el parámetro de escala de la distribución gamma y $\Gamma(k)$ es la función gamma aplicada al parámetro k.

La función gamma es una extensión de la función factorial a números complejos. Para un número complejo z con parte real positiva está definida por:

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} \, \mathrm{d}t \tag{5.5}$$

Si la parte real del número complejo z es positiva ($Re[z] > 0$), entonces la integral de la Ecuación 5.5 converge absolutamente.

Alternativamente, la distribución gamma puede ser parametrizada en términos de un parámetro de forma k y un parámetro de escala inverso $\beta = 1/\theta$:

$$g(x; k, \beta) = x^{k-1} \frac{\beta^k \, e^{-\beta \, x}}{\Gamma(k)} \text{ for } x > 0 \tag{5.6}$$

Por otra parte, la función de distribución acumulativa de la distribución gamma puede ser expresada en términos de la función gamma incompleta:

$$F(x; k, \beta) = \int_0^x f(u; k, 1/\beta) \, du = \frac{\gamma(k, x\beta)}{\Gamma(k)} \tag{5.7}$$

donde k y β son los parámetros de forma y de escala inverso respectivamente y $\gamma(k, x\beta)$ es la función gamma incompleta aplicada sobre k y $x\beta$.

La función gamma incompleta está definida como una función integral del mismo integrando que la función gamma. Existen dos variaciones de la función gamma incompleta: la función gamma incompleta superior para el caso en que el límite inferior de integración es variable, y la función gamma incompleta inferior para el caso en que el límite superior de integración es variable.

La función gamma incompleta superior se define en la Ecuación 5.8.

	CHC		PBIL	
Tamaño	Forma (k)	Escala (β)	Forma (k)	Escala (β)
2^{18}	1.41509	0.00406	1.42479	0.00253
2^{19}	1.47292	0.00233	1.50718	0.00093
2^{20}	1.48990	0.00079	1.55157	0.00054

Tabla 5.5: RLDs Gamma estimadas para los tamaños del problema.

$$\Gamma(a,x) = \int_x^\infty t^{a-1}\, e^{-t}\, dt. \tag{5.8}$$

Por otra parte, la función gamma incompleta inferior, que es la que aparece en la definición de la función de distribución acumulativa de la distribución gamma, se define como sigue:

$$\gamma(a,x) = \int_0^x t^{a-1}\, e^{-t}\, dt. \tag{5.9}$$

En ambos casos, a es un parámetro complejo, tal que la parte real de a es positiva.

Finalmente, si una variable aleatoria X tiene esta distribución con forma k y escala β se indica como sigue:

$$X \sim Gamma(k,\beta) \tag{5.10}$$

La Tabla 5.5 presenta los parámetros estimados, [5], [71]: forma (k) y escala (β), que definen la RLD Gamma promedio correspondiente a cada tamaño del problema y algoritmo. Si comprobamos las columnas de la Tabla de arriba hacia abajo, se observa una tendencia creciente para el parámetro de forma (k) y una tendencia decreciente para el parámetro de escala (β), para ambos algoritmos, a medida que se incrementa el tamaño del problema. Si, por el contrario, verificamos las filas de izquierda a derecha, se produce ese mismo comportamiento manteniendo fijo el tamaño del problema y cambiando de algoritmo: de CHC a PBIL.

Teniendo en cuenta la descripción detallada de las propiedades de los parámetros en [19], [82], podemos determinar, a partir de los valores de los parámetros presentados en la Tabla 5.5, el comportamiento general de los algoritmos para cada tamaño del problema y obtener conclusiones del rendimiento en cuanto a evolución definida por el algoritmo (filas de la Tabla)

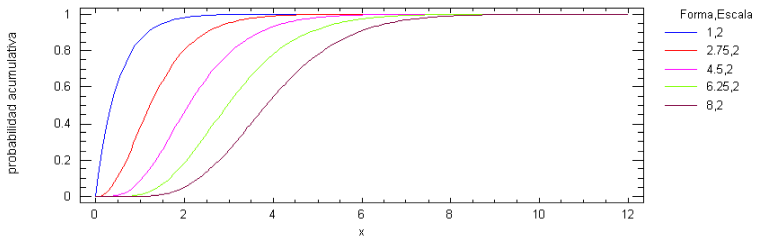


Figura 5.7: Efecto de variación de forma en la distribución Gamma: F(x).

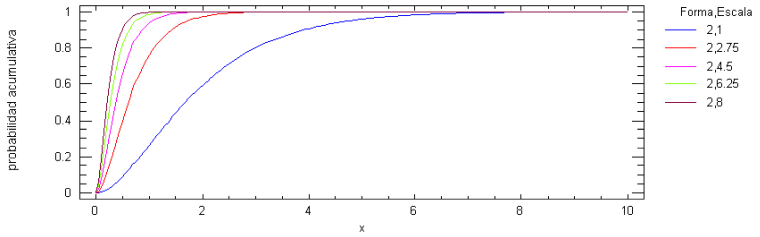


Figura 5.8: Efecto de variación de escala en la distribución Gamma: F(x).

y evolución definida por el tamaño del problema (columnas de la Tabla). Véanse las Figuras 5.7 y 5.8 para comprobar visualmente el efecto de la variación de cada uno de los parámetros de la distribución Gamma, forma y escala respectivamente, en la función de distribución acumulativa.

Puesto que en la mencionada Tabla 5.5 la forma (k) se incrementa y la escala (β) disminuye en la medida en que aumenta el tamaño del problema para cada algoritmo, tendremos que un mayor tamaño del problema supondrá un incremento de la longitud de ejecución necesaria para la obtención de soluciones válidas y así mismo, una mayor amplitud del intervalo conformado por tales longitudes de ejecución con una menor frecuencia de aparición de las mismas. Además, el incremento de tamaño del problema supone aumentar las dificultades para obtener soluciones de calidad, dada la disminución progresiva de la pendiente de la RLD con respecto al eje X, y una ampliación de la etapa final de estancamiento del algoritmo para probabilidades elevadas de éxito.

Cabe destacar que el parámetro que presenta más influencia en la caracterización de las RLDs es la escala (β), pues las pequeñas variaciones existentes entre los valores de forma (k) no suponen la misma alteración de las curvas que pequeñas variaciones de escala (β) para valores menores que 1, [19], [82].

5.6. Caracterización Gamma del rendimiento

Hemos comprobado que la función de distribución estadística continua Gamma se ajusta a las longitudes de tiempo de ejecución obtenidas como resultado de aplicar los algoritmos PBIL o CHC al banco de instancias del problema de la selección de la solución deseada limitado por espacios de búsqueda de 2^{20} soluciones.

De esta forma, podemos obtener a partir de la distribución Gamma estimada, para cada tamaño del problema concreto, los diferentes parámetros que caracterizan la evolución de cada uno de los dos algoritmos sobre las instancias de tal tamaño. Estos parámetros son los siguientes: media, desviación típica, mediana y percentiles de la longitud de tiempo de ejecución, probabilidades de éxito para tiempos arbitrarios de ejecución y tiempos máximos de ejecución para la obtención de determinadas probabilidades de éxito.

La estimación de la distribución Gamma para un tamaño del problema concreto podemos realizarla, tal y como se indicó en la Sección 5.4, a través del método de máxima verosimilitud, [5]. De esta forma, dado un algoritmo (en nuestro caso, CHC o PBIL), para cada tamaño del problema L que identifica un espacio de búsqueda limitado por 2^L soluciones, obtendríamos para los resultados de la ejecución del algoritmo, los parámetros $\widehat{k_L}$ (forma) y $\widehat{\beta_L}$ (escala) de la distribución Gamma estimada: $\widehat{Gamma}(\widehat{k_L}, \widehat{\beta_L})$.

Teniendo en cuenta la distribución Gamma estimada podemos definir los siguientes parámetros relativos a la longitud de tiempo de ejecución (variable aleatoria RL) para un algoritmo y tamaño del problema 2^L concreto, tal y como se establece en [33]:

- Media ($\overline{RL}$):

$$\overline{RL} = \hat{k_L}/\hat{\beta}_L \tag{5.11}$$

- Desviación típica (σ_{RL}):

$$\sigma_{RL} = \sqrt{\hat{k_L}}/\hat{\beta}_L \tag{5.12}$$

- Percentil j-ésimo (P_j):

$$P_j = j/100 \int_0^\infty t^{rl-1} e^{-t} \, \mathrm{d}t \tag{5.13}$$

- Mediana (Me):

$$Me = P_{50} = 1/2 \int_0^\infty t^{rl-1} e^{-t} \, \mathrm{d}t \tag{5.14}$$

Se debe resaltar que el cálculo del percentil y por tanto de la mediana viene dado por la expresión correspondiente a la función gamma, véase Ecuación 5.5.

La RLD promedio obtenida de los resultados de la ejecución de un algoritmo sobre instancias de un tamaño específico del problema se corresponde con la función de distribución acumulativa $F(x)$ de la distribución Gamma estimada. Teniendo en cuenta la expresión de $F(x)$, véase la Ecuación 5.7, podremos determinar la probabilidad de éxito, $F(rl_{exito}; \hat{k_L}, \hat{\beta_L})$, correspondiente a una determinada longitud de tiempo de ejecución rl_{exito}.

$$F(rl_{exito}; \hat{k}_L, \hat{\beta}_L) = \int_0^{rl_{exito}} f(u; \hat{k}_L, \hat{\beta}_L) \, du = \frac{\gamma(\hat{k}_L, rl_{exito}\hat{\beta}_L)}{\Gamma(\hat{k}_L)} \tag{5.15}$$

En el sentido contrario a la función de distribución acumulativa directa, para determinar, a partir del modelo Gamma, la longitud de ejecución necesaria (número de evaluaciones) para alcanzar una determinada calidad de solución (probabilidad de éxito) utilizaremos la función de distribución acumulativa inversa F^{-1} de la distribución Gamma. De esta forma, dada una probabilidad de éxito (p_{exito}) y un modelo Gamma de parámetros $\hat{k}_L$ y $\hat{\beta}_L$, el cálculo se realizaría, teniendo en cuenta la Ecuación 5.15, tal y como se define en la Ecuación 5.16:

$$rl_{exito} = F^{-1}(p_{exito}; \hat{k}_L, \hat{\beta}_L) = \{rl_{exito} : F(rl_{exito}; \hat{k}_L, \hat{\beta}_L) = p_{exito}\} \tag{5.16}$$

El cálculo de las funciones de distribución acumulativa directa e inversa de la distribución Gamma y así mismo, el cálculo de la función gamma y las funciones gamma incompletas, es computacionalmente eficiente y puede

llevarse a cabo a partir de diferentes algoritmos aproximados y métodos iterativos de optimización. Como referencia principal de la literatura puede consultarse [126]. Para mayor detalle sobre el cálculo de la función gamma veánse [28], [135]. En lo que corresponde a las funciones gamma incompletas pueden tomarse como referencias [2], [34].

Capítulo 6

Verificación y validación del modelo

La obtención de un modelo para representar el rendimiento de las Metaheurísticas cuando son aplicadas para resolver problemas de grandes requerimientos computacionales es fundamental para determinar las garantías de calidad de solución frente al tiempo de ejecución disponible. La disponibilidad de este modelo es aún mas importante para tamaños considerables del problema.

En el Capítulo anterior, la distribución Gamma había sido ajustada al rendimiento de los algoritmos CHC y PBIL para la resolución del problema de la selección de la solución deseada dentro de la resolución de restricciones geométricas. Sin embargo, los tamaños del problema analizados no tenían requerimientos computacionales muy elevados.

La necesidad de caracterizar el comportamiento de los algoritmos para cualquier tamaño del problema invita a comprobar el ajuste de las RLDs a la distribución Gamma para tamaños del problema superiores. Para ello se realizó una predicción de la aplicación estadísticamente óptima de los algoritmos en el Capítulo 4 a partir de un modelo sencillo: el modelo de regresión lineal simple. Se definieron expresiones que indican para cada tamaño del problema los valores estadísticamente óptimos de los parámetros que definen la evolución de CHC y PBIL.

El objetivo del presente Capítulo será el de verificar si la distribución Gamma se mantiene como representante del rendimiento para tamaños superiores y definir expresiones que permitan estimar el modelo Gamma dado un tamaño

del problema. Finalmente, realizaremos la validación de los modelos Gamma estimados para predecir el comportamiento de los algoritmos CHC y PBIL ante tamaños considerables del problema.

6.1. Verificación del modelo

La distribución Gamma quedó definida en el Capítulo 5 como la distribución estadística continua que caracteriza el rendimiento de los algoritmos CHC y PBIL cuando son aplicados a los tamaños 2^{18}, 2^{19} y 2^{20} del problema de la selección de la solución deseada dentro de la resolución de restricciones geométricas.

Es necesario comprobar si ello se mantiene para tamaños superiores del problema. Teniendo en cuenta el elevado tiempo de cómputo que supone el ajuste estadísticamente óptimo de los algoritmos para cada tamaño del problema, véase la Sección 4.3, es conveniente simplificar tal proceso. Para ello, se definieron en la Sección 4.4 expresiones para la predicción de los valores, estadísticamente óptimos, de los parámetros que condicionan la evolución de los algoritmos basadas en los resultados obtenidos para los tamaños del problema 2^{18}, 2^{19} y 2^{20}.

De esta forma, nuestro objetivo será el de seleccionar un conjunto de tamaños superiores del problema, próximos a los analizados, y observar el rendimiento estadísticamente óptimo de los algoritmos CHC y PBIL sobre los mismos. Habremos de comprobar si existe una distribución que se ajuste a las RLDs y si dicha distribución se corresponde con la distribución Gamma.

6.1.1. Diseño procedimental

Para verificar la distribución estadística continua Gamma como representante del rendimiento de los algoritmos CHC y PBIL se seleccionarán cuatro tamaños superiores y cercanos a los analizados en trabajos anteriores: longitudes de índice de 25, 30, 35 y 50, estando los espacios de búsqueda limitados por 2^{25}, 2^{30}, 2^{35} y 2^{50} soluciones respectivamente.

Para cada tamaño del problema se generarán aleatoriamente 10 instancias o figuras. La generación de figuras (elementos geométricos y restricciones sobre los elementos geométricos) estará basada en la obtención de grafos bien restringidos y descomponibles, [153], a partir de secuencias de Henneberg, [17], véase la Sección 3.3.1. Los predicados adicionales que definen la

solución requerida por el usuario serán incorporados a cada figura mediante la elección aleatoria de elementos geométricos y la asociación a los mismos de predicados de orientación: punto a la izquierda de un segmento y orientación de tres puntos en el sentido de las agujas del reloj. Para cada instancia se ha generado un número diferente de predicados adicionales con un límite máximo de 50 predicados. El banco de instancias completo está disponible en [159].

A continuación describiremos el proceso seguido para la verificación de la distribución Gamma como representante del rendimiento de los algoritmos CHC y PBIL cuando son aplicados al problema de la selección de la solución deseada en sus tamaños 2^{25}, 2^{30}, 2^{35} y 2^{50}.

Para cada algoritmo e instancia se realizarán un total de 50 ejecuciones, partiendo cada una de ellas de una semilla aleatoria diferente. La condición de parada de cada ejecución vendrá determinada por el alcance de una solución que cumpla todas las restricciones adicionales definidas para la instancia o bien por la realización de 50000 evaluaciones de la función *fitness*, véase la Sección 3.5, en la búsqueda. Recordemos que la función *fitness* cuenta el número de predicados adicionales que cumple una solución.

Los valores que tomarán los parámetros de los algoritmos en su ejecución serán obtenidos tomando como base la optimización estadística mediante ANOVA que se realizó para los algoritmos en su aplicación a los tamaños del problema 2^{18}, 2^{19} y 2^{20}, véase la Sección 4.3. Para extrapolar los resultados obtenidos a tamaños superiores del problema se realizó una predicción de valores estadísticamente óptimos de los parámetros mediante el modelo de regresión lineal simple, véase la Sección 4.4.

Para cada ejecución se obtendrá el número de evaluaciones de la función *fitness* requeridos manteniéndose, de esta forma, la independencia de los resultados con respecto a la plataforma de ejecución, [3]. A partir de los resultados de las diferentes ejecuciones llevadas a cabo para cada instancia y algoritmo se elaborará la RLD empírica (Ecuación 5.1).

Una vez caracterizada y analizada la evolución del rendimiento de los algoritmos a través de las RLDs empíricas, nos planteamos generalizar tales RLDs intentando ajustar los datos observados a una distribución estadística continua conocida. Hablando en términos de longitudes de tiempos de ejecución representados a partir de una RLD, deseamos ajustar las RLDs empíricas a una función de distribución acumulativa correspondiente a una distribución estadística teórica continua conocida. Tales distribuciones son

las mismas que se usaron en el Capítulo 5. Para mayor detalle sobre las mismas véase [33].

El objetivo es el de aplicar tanto técnicas gráficas como cuantitativas de bondad de ajuste (las mismas que en el Capítulo anterior) y seleccionar aquella distribución o distribuciones que permitan caracterizar las RLDs correspondientes a las distintas instancias para ambos algoritmos: CHC y PBIL. Como paso previo a la aplicación de los diferentes tests de bondad de ajuste nos hemos asegurado de que se cumplen los supuestos sobre las muestras que cada test establece.

El proceso seguido para la aplicación de los tests de bondad de ajuste para los resultados obtenidos en cada algoritmo, CHC y PBIL, ha sido el mismo que el aplicado en el Capítulo anterior. La distribución que se considerará como representante de las RLDs empíricas para cada algoritmo será aquella que se obtenga como intersección de los resultados de los métodos cuantitativos de bondad de ajuste. El ajuste de tal distribución será comprobado a partir del método gráfico de bondad de ajuste seleccionado. Verificaremos si tal distribución corresponde al modelo Gamma.

Sin embargo, para cada instancia concreta del problema y algoritmo se dispondrá de una distribución o distribuciones que caracterizan la evolución del algoritmo. Tanto la predicción como la generalización del comportamiento de los algoritmos para instancias desconocidas es más simple si se dispone de una sola representación del rendimiento de cada algoritmo para cada tamaño del problema.

La RLD promedio que resuma las RLDs correspondientes a un banco de instancias para un determinado algoritmo y tamaño del problema habrá de tener su origen, al igual que cada una de tales RLDs, en una RLD empírica promedio, véase la Ecuación 5.3. A partir de la RLD empírica promedio estimaremos la RLD promedio mediante el método de máxima verosimilitud. Partiremos de un conjunto de observaciones de longitud de ejecución obtenido a partir de un muestreo uniforme de dicha RLD empírica promedio, teniendo presente la Ecuación 5.1.

Finalmente, el ajuste de la RLD promedio a la RLD empírica promedio será contrastado mediante las técnicas cuantitativas y gráficas de bondad de ajuste usadas. Dispondremos de una RLD, vinculada a una distribución estadística continua conocida, representativa de cada algoritmo para cada tamaño de los analizados. Tal RLD habrá de seguir el modelo Gamma para que la verificación tenga éxito.

Fig. \ Tam.	2^{25}	2^{30}	2^{35}	2^{50}
1	653	3475	7283	15199
2	2119	3378	8061	14296
3	7168	3146	3869	9685
4	1000	2684	2882	9271
5	2615	4738	4161	25669
6	3689	4119	3213	10766
7	945	4822	5978	8266
8	803	2532	7066	6151
9	5845	6895	6676	10738
10	2537	5155	4965	12047
Media	2737.4	4094.4	5415.4	12208.8

Tabla 6.1: Resultados de las ejecuciones de CHC hasta 2^{50}.

Fig. \ Tam.	2^{25}	2^{30}	2^{35}	2^{50}
1	4048	18275	16932	20803
2	7311	17344	17608	20230
3	16775	16250	20329	19103
4	1827	16926	20708	18618
5	12952	17491	16000	17554
6	11848	10666	14256	18965
7	1824	19323	18413	24244
8	2397	13312	20982	19366
9	14078	16626	25432	13394
10	12588	18549	15450	19426
Media	8564.8	16476.2	18611.0	19170.3

Tabla 6.2: Resultados de las ejecuciones de PBIL hasta 2^{50}.

6.1.2. Resultados

Las Tablas 6.1 y 6.2 presentan un resumen de los resultados de las ejecuciones de CHC y PBIL respectivamente sobre las distintas instancias generadas para cada tamaño del problema. En las mismas se presenta, para cada instancia, la media de evaluaciones necesaria para alcanzar una solución con éxito para las 50 ejecuciones. Para cada tamaño del problema se muestra también la media global del número de evaluaciones.

Podemos observar como el número de evaluaciones necesarias para alcanzar una solución con éxito es creciente a medida que aumenta el tamaño del problema tanto para CHC como para PBIL. El número de evaluaciones requeridas por CHC sigue siendo muy inferior al requerido por PBIL. La evolución que se observa para los tamaños analizados es similar a la que se obtuvo para los tamaños del problema 2^{18}, 2^{19} y 2^{20} y además mantiene la

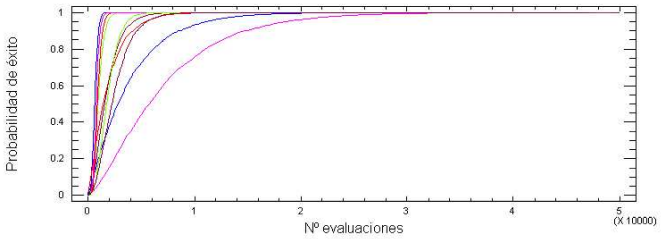
homogeneidad de sus antecesores en el crecimiento del número de evaluaciones. En todo caso, PBIL requiere un mayor número de evaluaciones debido a su incapacidad de diversificación en etapas avanzadas de la búsqueda. Véanse la Sección 4.3, el Capítulo 5 y [76] para una justificación más detallada.

A partir de los resultados de las ejecuciones de los algoritmos sobre las diferentes instancias del problema se elaboraron, siguiendo la Ecuación 5.1, las RLDs empíricas: una por algoritmo e instancia del problema. Tales RLDs empíricas fueron ajustadas mediante los métodos cuantitativos y gráficos de bondad de ajuste citados en la Sección 5.4 a distribuciones estadísticas continuas conocidas: exponencial, Weibull, Gamma, Erlang, normal, Laplace y Cauchy. La distribución que se ajustó con un 95 % de probabilidad a todas las instancias fue, al igual que ocurrió para los tamaños 2^{18}, 2^{19} y 2^{20} en el Capítulo anterior, la distribución estadística continua Gamma.

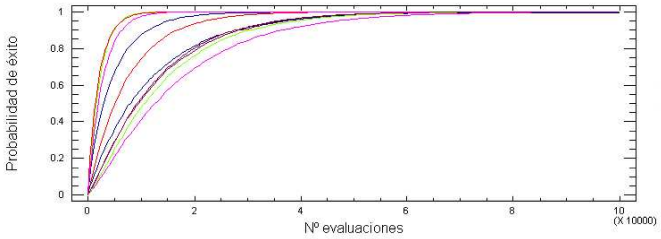
La Figura 6.1 presenta, como ejemplo, las diferentes RLDs Gamma correspondientes a las distintas instancias para el tamaño 2^{25}. El resto de familias de RLDs se presentan en el Apéndice B. Para cada instancia aparece la función de distribución acumulativa de la distribución Gamma (RLD Gamma) que se ajusta a la RLD empírica. Para cada tamaño del problema se aprecia la agrupación de las diferentes RLDs Gamma asociadas a las distintas instancias.

Con objeto de representar el rendimiento de cada algoritmo para cada tamaño del problema mediante una única RLD Gamma se elaboró para cada tamaño del problema, siguiendo la Ecuación 5.3, la RLD Gamma promedio correspondiente. En la Figura 6.2 se puede observar para el tamaño 2^{25}, tanto para CHC como PBIL, la RLD Gamma promedio representante destacada del resto de RLDs Gamma asociadas a las diferentes instancias. Se puede comprobar gráficamente la escasa diferencia entre la RLD Gamma promedio y las diferentes RLDs Gamma para la descripción del rendimiento de los algoritmos para cada tamaño del problema. Las RLDs Gamma promedio correspondientes al resto de tamaños junto a sus familias asociadas están disponibles en el Apéndice B.

Si representamos gráficamente las diferentes RLDs Gamma promedio correspondientes a los diferentes tamaños del problema analizados anteriormente en el Capítulo 5 y los que analizamos en este Capítulo, podremos comprobar la evolución del rendimiento de los algoritmos en función del tamaño. Las Figuras 6.3 y 6.4 presentan tales RLDs Gamma promedio. Podemos observar como la evolución del rendimiento supone en todo caso una mayor inclinación de la curva hacia la derecha del eje X a medida que el tamaño del problema

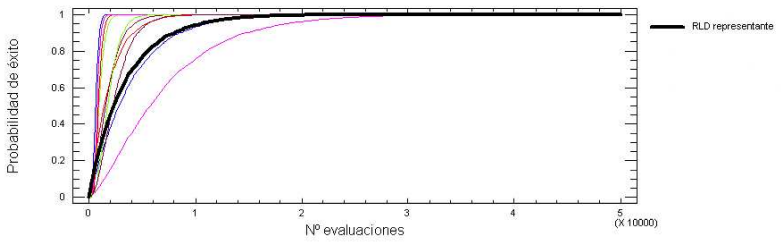


(a)

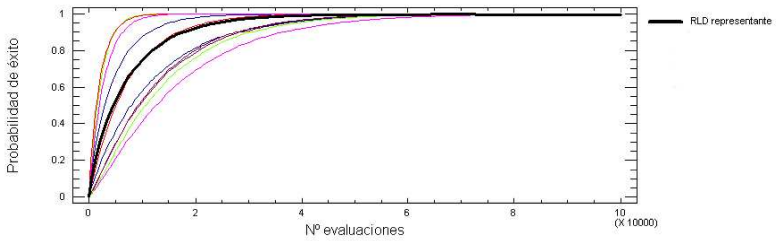


(b)

Figura 6.1: RLDs Gamma de a) CHC y b) PBIL para el tamaño 2^{25}.



(a)



(b)

Figura 6.2: RLDs Gamma representantes de a) CHC y b) PBIL para el tamaño 2^{25}.

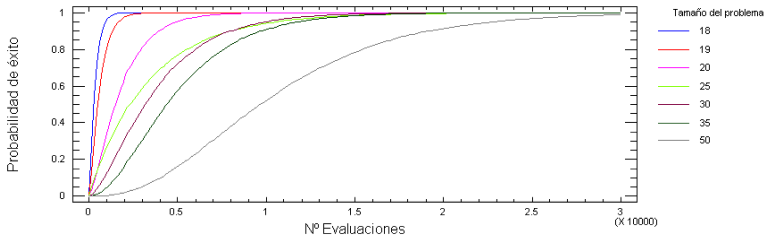


Figura 6.3: RLDs Gamma promedio de CHC.

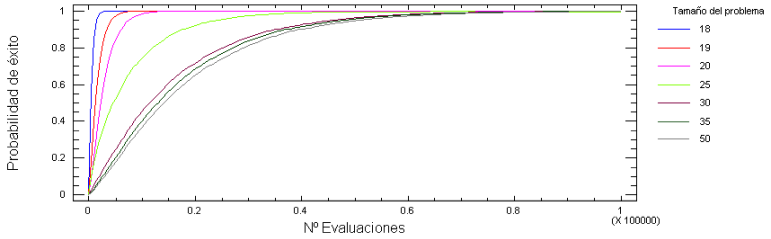


Figura 6.4: RLDs Gamma promedio de PBIL.

es mayor. A medida que el tamaño del problema crece, es mayor el número de evaluaciones necesario para alcanzar la misma probabilidad de éxito. Hay que destacar, así mismo, que la variación en la inclinación de las curvas no es homogénea a medida que aumenta el tamaño, existiendo tamaños del problema para los que no aparece una gran variación de rendimiento para un mismo algoritmo y otros para los que ésta es muy notoria. Todo ello es coherente con la dificultad que supone para cada algoritmo el aumento del número de soluciones distintas que constituyen el espacio de búsqueda. En todo caso, PBIL se mueve a una escala de número de evaluaciones distinta a CHC, tal y como fue anticipado anteriormente.

Los parámetros de la distribución Gamma correspondiente a la RLD Gamma promedio representante del rendimiento de cada algoritmo para cada tamaño del problema se muestran, respectivamente para CHC y PBIL, en las Tablas 6.3 y 6.4. Se puede observar que no existe una variación homogénea entre los valores de cada parámetro a medida que aumenta el tamaño del pro-

Tamaño	Forma (k)	Escala (β)
2^{18}	1.41509	0.004060000
2^{19}	1.47292	0.002330000
2^{20}	1.48990	0.000790000
2^{25}	0.94826	0.000277453
2^{30}	1.66269	0.000424744
2^{35}	2.24895	0.000438662
2^{50}	3.03207	0.000279974

Tabla 6.3: Parámetros de las RLDs Gamma promedio de CHC.

Tamaño	Forma (k)	Escala (β)
2^{18}	1.424790	0.002530000
2^{19}	1.507180	0.000930000
2^{20}	1.551570	0.000540000
2^{25}	0.798105	0.000109621
2^{30}	1.103920	0.000070205
2^{35}	1.254875	0.000073190
2^{50}	1.301594	0.000070348

Tabla 6.4: Parámetros de las RLDs Gamma promedio de PBIL.

blema tanto para CHC como para PBIL. Sin embargo, ha quedado verificada la distribución Gamma como representante del rendimiento de los algoritmos CHC y PBIL para tamaños del problema superiores a los analizados. El estudio detallado se presenta en [76].

6.2. Predicción del rendimiento

La distribución Gamma ha sido establecida como representante del rendimiento de los algoritmos CHC y PBIL para los tamaños del problema 2^{18}, 2^{19}, 2^{20} y verificada en la Sección 6.1 para tamaños superiores: 2^{25}, 2^{30}, 2^{35} y 2^{50}.

Una posible generalización del rendimiento de los algoritmos CHC y PBIL cuando tratan de abordar la solución al problema de la selección de la solución deseada supondría disponer de una representación de tal rendimiento para cada tamaño del problema. En la Sección 4.4 se realizó una predicción mediante un modelo sencillo: el modelo de regresión lineal simple, para los parámetros que determinan la evolución de los algoritmos estudiados en función del tamaño. Sería deseable utilizar este mismo modelo para la predicción del rendimiento, o sea, la predicción de la RLD Gamma promedio asociada

Tamaño	Media	Desv. Típ
2^{18}	348.544335	292.998905
2^{19}	632.154507	520.875174
2^{20}	1885.949367	1545.081771
2^{25}	3417.728408	3509.732858
2^{30}	3914.569717	3035.836581
2^{35}	5126.840255	3418.691291
2^{50}	10829.827060	6219.448981

Tabla 6.5: Parámetros estadísticos de media y desviación típica de las RLDs Gamma promedio de CHC.

a la aplicación de un algoritmo a un tamaño del problema.

Sin embargo, si observamos los parámetros que definen las RLDs Gamma promedio asociadas a los tamaños del problema estudiados, véanse la Tabla 6.3 para CHC y la Tabla 6.4 para PBIL, no observamos una relación lineal entre los valores de un mismo parámetro para un algoritmo determinado. No se cumple así una de las hipótesis básicas del modelo de regresión lineal simple, [33], [57].

Por otra parte, dos de los parámetros estadísticos fundamentales que definen una distribución estadística unidimensional son la media y la desviación típica o varianza. Para la distribución estadística continua Gamma los valores de tales parámetros se calculan en función de los parámetros forma (k) y escala (β) de la distribución. Para la variable aleatoria longitud de ejecución (RL) las expresiones correspondientes se definen en la Sección 5.6: Ecuación 5.11 para la media y Ecuación 5.12 para la desviación típica.

Las Tablas 6.5 y 6.6 presentan, respectivamente para CHC y PBIL, los parámetros estadísticos media y desviación típica para las distribuciones Gamma representantes de cada tamaño analizado del problema. La tendencia tanto en la media como en la desviación típica es prácticamente creciente en todos los casos a medida que aumenta el tamaño del problema.

Tal y como se demuestra en [76], los diagramas de dispersión correspondientes a los parámetros estadísticos media y desviación típica de la longitud de ejecución permiten percibir claramente la relación lineal entre el tamaño del problema y cada parámetro estadístico. Además, se aprecia la existencia de homocedasticidad, pues existe una amplitud similar del conjunto de puntos constituido por todos los valores de cada parámetro estadístico correspondientes a un mismo tamaño del problema. Finalmente, no aparecen valores atípicos en los diagramas. Se cumplen por tanto las hipótesis básicas de li-

Tamaño	Media	Desv. Típ
2^{18}	563.158103	471.796710
2^{19}	1620.623656	1320.078059
2^{20}	2873.277778	2306.704310
2^{25}	7280.584924	8149.599308
2^{30}	15724.173050	14965.771580
2^{35}	17145.441020	15305.531910
2^{50}	18502.257000	16217.613280

Tabla 6.6: Parámetros estadísticos de media y desviación típica de las RLDs Gamma promedio de PBIL.

nealidad, homocedasticidad y datos atípicos del modelo de regresión lineal simple.

La obtención de expresiones que definan en función del tamaño del problema (L), identificando a espacios de búsqueda limitados por 2^L soluciones, a los valores de los parámetros estadísticos media y desviación típica permitirá, a partir de las Ecuaciones 5.11 y 5.12, disponer de los valores que definen la distribución Gamma (forma: k y escala: β) representativa del rendimiento de cada algoritmo para cada tamaño.

Hemos aplicado, por tanto, el modelo de regresión lineal simple para la predicción de los valores de los parámetros estadísticos media ($\overline{RL}$) y desviación típica (σ_{RL}) de la distribución Gamma, dado un tamaño del problema (L) identificando a un espacio de búsqueda de 2^L soluciones. El proceso es similar al aplicado en la Sección 4.4 y se detalla en [76].

Una vez realizado el análisis de regresión y demostrada la validez del ajuste del modelo para los diferentes parámetros, pasamos a presentar e interpretar los resultados de la generalización con objeto de utilizarlos para predecir el rendimiento de la aplicación de los algoritmos CHC y PBIL sobre instancias desconocidas del problema de la selección de la solución deseada.

El estudio que hemos realizado tiene como objetivo el de comprobar el rendimiento de los algoritmos CHC y PBIL cuando son aplicados a instancias de tamaño considerable del problema. La aplicación del mismo para instancias de tamaño reducido ($2^L; L < 18$) no ha sido prevista ni es significativa en principio. Para poder ser generalizado el análisis para instancias de tamaño reducido se deberían realizar las pruebas y análisis oportunos.

La expresión correspondiente a la generalización individual de cada uno de los parámetros estadísticos que determinan el rendimiento de los algoritmos en función de la distribución Gamma se muestra en las Tablas 6.7 y 6.8 para

Parámetro (P)	Ecuación generalización
Media ($\overline{RL}$)	$\widehat{\overline{RL}} = -4936{,}36 + 308{,}173L;\ \forall L \geq 18;\ L \in N$
Desviación típica (σ_{RL})	$\widehat{\sigma_{RL}} = -2107{,}86 + 169{,}024L;\ \forall L \geq 18;\ L \in N$

Tabla 6.7: Predicción de los parámetros estadísticos media y desv. típ. del modelo para CHC.

Parámetro (P)	Ecuación generalización
Media ($\overline{RL}$)	$\widehat{\overline{RL}} = -8135{,}06 + 612{,}462L;\ \forall L \geq 18;\ L \in N$
Desviación típica (σ_{RL})	$\widehat{\sigma_{RL}} = -6767{,}08 + 538{,}613L;\ \forall L \geq 18;\ L \in N$

Tabla 6.8: Predicción de los parámetros estadísticos media y desv. típ. del modelo para PBIL.

CHC y PBIL respectivamente. Partimos de un tamaño del problema definido por L que identifica a un espacio de búsqueda limitado por 2^L soluciones.

Dado un tamaño del problema desconocido, teniendo en cuenta las Ecuaciones 5.11 y 5.12 de definición de parámetros estadísticos Gamma y las Tablas 6.7 y 6.8 de predicción de parámetros estadísticos Gamma, la RLD Gamma estimada ($\widehat{Gamma}$) que describirá el rendimiento de cada algoritmo tendrá unos parámetros de forma ($\hat{k}$) y escala ($\hat{\beta}$) que vendrán definidos por las expresiones:

- Forma ($\hat{k}$):

$$\hat{k} = \left(\frac{\widehat{\overline{RL}}}{\widehat{\sigma_{RL}}} \right)^2 \tag{6.1}$$

- Escala ($\hat{\beta}$):

$$\hat{\beta} = \frac{\widehat{\overline{RL}}}{\widehat{\sigma_{RL}}^2} \tag{6.2}$$

6.3. Validación del modelo

En la Sección 6.2 se definieron expresiones para la predicción de las RLDs Gamma que describirán el rendimiento de los algoritmos CHC y PBIL ante tamaños del problema desconocidos. Nuestro objetivo será ahora el de

validar la fiabilidad de tales RLDs como referencias para pronosticar el rendimiento de los algoritmos CHC y PBIL en función del tiempo disponible o la probabilidad de éxito requerida para la solución.

Trataremos con tamaños del problema que exigen de requerimientos computacionales muy elevados para su resolución desde un punto de vista temporal.

Para validar el modelo Gamma se considerarán dos cuestiones:

> Cuestión 1: Si se fija la calidad de la solución deseada, proporciónese una estimación para la longitud de ejecución que el algoritmo requiere para encontrar una instancia solución apropiada.
>
> Cuestión 2: Dada una longitud de ejecución límite, estímese la calidad de la instancia solución que encontrará el algoritmo evolutivo.

Los tamaños de los espacios de búsqueda de las instancias consideradas serán de hasta 2^{100} y se compararán los resultados que el modelo predice con los que proporciona la ejecución de los algoritmos.

6.3.1. Diseño procedimental

Para demostrar la fiabilidad de la distribución estadística continua Gamma que se ha predicho como representante del rendimiento de los algoritmos CHC y PBIL se seleccionarán cuatro tamaños del problema de grandes requerimientos de tiempo: longitudes de índice de 60, 70, 80 y 100, estando los espacios de búsqueda limitados por 2^{60}, 2^{70}, 2^{80} y 2^{100} soluciones respectivamente. El análisis de la evolución de los algoritmos CHC y PBIL sobre tales tamaños del problema se hace muy complicado por los elevados requerimientos de tiempo tanto para la optimización de los parámetros de los propios algoritmos como para el estudio del ajuste de las RLDs empíricas que determinan el rendimiento.

Utilizaremos, por tanto, para los mencionados tamaños, la predicción de valores de parámetros de los algoritmos que se definió en la Sección 4.4 para las ejecuciones y la predicción del modelo de rendimiento Gamma de la Sección 6.2, dada por las Ecuaciones 6.1 y 6.2, para pronosticar el rendimiento.

El banco de instancias que se utilizará, [159], será generado aleatoriamente, tal y como se realizó en la Sección 6.1, y constará de 10 figuras para cada tamaño del problema.

Queremos comprobar la fiabilidad de los modelos Gamma obtenidos para indicar la probabilidad de éxito de los algoritmos dado un tiempo disponible para la ejecución de los mismos (*Cuestión 2*). Así mismo, también necesitamos comprobar la fiabilidad de tales modelos Gamma para indicar el tiempo de ejecución del que disponer para obtener una determinada calidad o probabilidad de éxito (*Cuestión 1*). Para cada instancia generada estableceremos la probabilidad de éxito o calidad de solución deseada en 0.9, puesto que ésta es una buena calidad de solución y teniendo en cuenta a la vez los posibles estancamientos de los algoritmos que pueden surgir para obtener probabilidades de éxito más elevadas, [71], [72].

Para determinar la calidad de una solución a partir del modelo Gamma, para una longitud de ejecución dada, utilizaremos la función de distribución acumulativa de la distribución Gamma, véase la Sección 5.6: Ecuación 5.15.

En el sentido contrario a la función de distribución acumulativa directa, para determinar, a partir del modelo Gamma, la longitud de ejecución necesaria (número de evaluaciones) para alcanzar una determinada calidad de solución (probabilidad de éxito) utilizaremos la función de distribución acumulativa inversa de la distribución Gamma, véase la Sección 5.6: Ecuación 5.16.

Dado un algoritmo, en nuestro caso PBIL o CHC, disponemos para cada tamaño del problema seleccionado, $L = \{60, 70, 80, 100\}$ estando los espacios de búsqueda correspondientes limitados por 2^L soluciones, de un modelo Gamma asociado, $Gamma_L(\hat{k}_L, \hat{\beta}_L)$. Cada modelo Gamma presenta una función de distribución acumulativa F_L que define el rendimiento del algoritmo: longitud de ejecución vs. probabilidad de éxito. Hemos identificado como valor de buena calidad de solución a $p_{exito} = 0.9$.

Para validar un modelo $Gamma_L$ para cada ejecución j del algoritmo sobre una instancia I de tamaño L (espacio de búsqueda 2^L) hemos de comprobar, en primer lugar, la diferencia entre la longitud de ejecución que el modelo estima necesaria para alcanzar p_{exito} y la que el algoritmo requiere realmente para obtener tal calidad de solución (*Cuestión 1*). Con objeto de normalizar tal diferencia para posteriores comparaciones, ésta será expresada en términos de porcentaje tal y como se define en la Ecuación 6.3.

$$\Delta rl^j_{L,I} = \frac{|F_L^{-1}(p_{exito}; \hat{k}_L, \hat{\beta}_L) - RL_I(p_{exito}, s_j)|}{F_L^{-1}(p_{exito}; \hat{k}_L, \hat{\beta}_L)} * 100,\ j \in \mathbb{N} \qquad (6.3)$$

siendo $F_L^{-1}(p_{exito}; \hat{k}_L, \hat{\beta}_L)$ el valor de longitud de ejecución estimado por el modelo y $RL_I(p_{exito}, s_j)$ el número de evaluaciones requerido por una

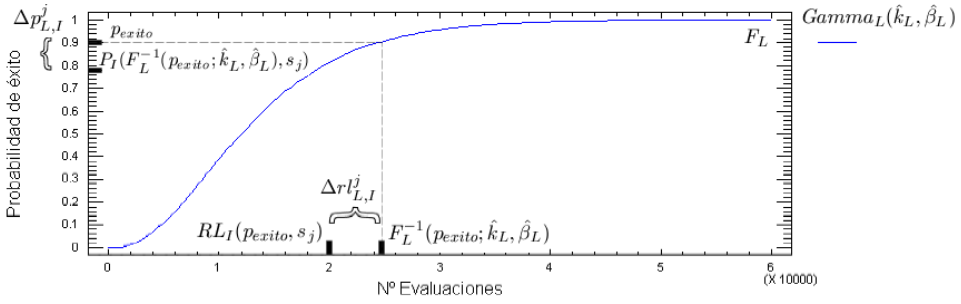


Figura 6.5: Medidas de validación del ajuste del modelo al resultado de una ejecución.

ejecución del algoritmo sobre la instancia del problema I partiendo de una semilla aleatoria s_j.

Así mismo, para completar la validación hemos de comprobar, para cada ejecución j del algoritmo sobre una instancia I de tamaño L, la diferencia entre p_{exito} y la calidad obtenida por el algoritmo para la longitud de ejecución que el modelo estima para alcanzar p_{exito} (*Cuestión 2*). Al igual que para el valor diferencia de longitud de ejecución, $\Delta rl^j_{L,I}$, esta diferencia también será expresada en términos de porcentaje tal y como se define en la Ecuación 6.4.

$$\Delta p^j_{L,I} = \frac{|p_{exito} - P_I(F_L^{-1}(p_{exito};\hat{k}_L,\hat{\beta}_L),s_j)|}{p_{exito}} * 100,\ j \in \mathbb{N} \tag{6.4}$$

siendo $P_I(F_L^{-1}(p_{exito};\hat{k}_L,\hat{\beta}_L),s_j)$ la calidad obtenida para una ejecución del algoritmo sobre la instancia I partiendo de una semilla aleatoria s_j y teniendo como límite de longitud de ejecución para alcanzar la calidad requerida, p_{exito}, la estimada por el modelo, $F_L^{-1}(p_{exito};\hat{k}_L,\hat{\beta}_L)$.

La Figura 6.5 presenta gráficamente las medidas comentadas para la validación del ajuste del modelo Gamma correspondiente a un tamaño L (espacio de búsqueda de 2^L) del problema, $Gamma_L(\hat{k}_L,\hat{\beta}_L)$, al rendimiento de una ejecución j de un algoritmo, que parte de la semilla aleatoria s_j, sobre una instancia I representativa de tal tamaño.

Debido a la naturaleza aleatoria de las ejecuciones de los algoritmos, se hace necesario tener en cuenta un número considerable, n_r, de ejecuciones para cada instancia con objeto de validar la adecuación de un modelo Gamma, $Gamma_L(\hat{k}_L,\hat{\beta}_L)$, al rendimiento de un algoritmo para tal instancia. Así mismo, para conocer el ajuste del modelo de rendimiento para un tamaño del

problema 2^L, se debe partir de un conjunto representativo de NI instancias.

Las medidas que permitirán validar el modelo para un tamaño 2^L determinado habrán de contabilizar los dos valores diferencia correspondientes a cada una de las n_r ejecuciones para todas las instancias, NI. Para ello, basándonos en las Ecuaciones 6.3 y 6.4, tomaremos el valor medio de los valores diferencia para cada instancia I, $\overline{\Delta rl_{L,I}}$ y $\overline{\Delta p_{L,I}}$, y obtendremos la media de tales valores medios para todas las instancias, que denotaremos por $\overline{\Delta rl_L}$ y $\overline{\Delta p_L}$ (Ecuaciones 6.5 y 6.6 respectivamente). Para tener en cuenta la variabilidad en los valores diferencia se calculará la desviación típica de los valores diferencia para cada instancia I, $\sigma_{\Delta rl_{L,I}}$ y $\sigma_{\Delta rl_{L,I}}$, y se obtendrá la media de las desviaciones típicas para todas las instancias, que denotaremos por $\sigma_{\Delta rl_L}$ y $\sigma_{\Delta p_L}$ (Ecuaciones 6.7 y 6.8 respectivamente).

$$\overline{\Delta rl_L} = \frac{1}{NI}\sum_{I=1}^{NI}\overline{\Delta rl_{L,I}} \quad ; \quad \overline{\Delta rl_{L,I}} = \frac{1}{n_r}\sum_{j=1}^{n_r}\Delta rl_{L,I}^{j} \tag{6.5}$$

$$\overline{\Delta p_L} = \frac{1}{NI}\sum_{I=1}^{NI}\overline{\Delta p_{L,I}} \quad ; \quad \overline{\Delta p_{L,I}} = \frac{1}{n_r}\sum_{j=1}^{n_r}\Delta p_{L,I}^{j} \tag{6.6}$$

$$\sigma_{\Delta rl_L} = \frac{1}{NI}\sum_{I=1}^{NI}\sigma_{\Delta rl_{L,I}} \quad ; \quad \sigma_{\Delta rl_{L,I}} = \sqrt{\frac{1}{n_r}\sum_{j=1}^{n_r}\left(\overline{\Delta rl_{L,I}} - \Delta rl_{L,I}^{j}\right)^2} \tag{6.7}$$

$$\sigma_{\Delta p_L} = \frac{1}{NI}\sum_{I=1}^{NI}\sigma_{\Delta p_{L,I}} \quad ; \quad \sigma_{\Delta p_{L,I}} = \sqrt{\frac{1}{n_r}\sum_{j=1}^{n_r}\left(\overline{\Delta p_{L,I}} - \Delta p_{L,I}^{j}\right)^2} \tag{6.8}$$

Para cada algoritmo, la fiabilidad de un modelo para un tamaño del problema determinado será mayor en la medida en que la magnitud de los valores media resumen de las diferencias sea menor. Basándonos en las Ecuaciones 6.3, 6.4, 6.5 y 6.6, tal fiabilidad queda expresada formalmente en las Ecuaciones 6.9 y 6.10. La magnitud de los valores de la desviación típica también influirá en la medida en que los valores media lo hagan.

$$\lim_{\overline{\Delta rl_L}\to 0} RL_I(p_{exito}, s_j) = F_L^{-1}(p_{exito}; \hat{k}_L, \hat{\beta}_L) \quad \forall j \in \mathbb{N}, \forall I \tag{6.9}$$

Modelo	CHC		PBIL	
$Gamma_L$	$\hat{k}_L$	$\hat{\beta}_L$	$\hat{k}_L$	$\hat{\beta}_L$
$Gamma_{60}$	2.84654	2.10015E-04	1.25413	4.38315E-05
$Gamma_{70}$	2.92692	1.75942E-04	1.26086	3.62971E-05
$Gamma_{80}$	2.98416	1.51346E-04	1.26560	3.09728E-05
$Gamma_{100}$	3.06025	1.18243E-04	1.27185	2.39470E-05

Tabla 6.9: Modelos Gamma de rendimiento de PBIL y CHC a validar.

$$\lim_{\overline{\Delta p_L} \to 0} P_L(F_L^{-1}(p_{exito}; \hat{k}_L, \hat{\beta}_L), s_j) = p_{exito} \quad \forall j \in \mathbb{N}, \forall I \tag{6.10}$$

En resumen, para cada algoritmo y tamaño del problema seleccionado $2^L = \{2^{60}, 2^{70}, 2^{80}, 2^{100}\}$ se obtendrá un modelo Gamma, $Gamma_L(\hat{k}_L, \hat{\beta}_L)$. Para validar tal modelo se realizarán n_r ejecuciones del algoritmo para cada instancia correspondiente a un conjunto representativo de NI instancias y se obtendrán los valores resumen del ajuste del modelo al rendimiento del algoritmo: $\overline{\Delta rl_L}$ y $\sigma_{\Delta rl_L}$ para responder a la *Cuestión 1* y $\overline{\Delta p_L}$ y $\sigma_{\Delta p_T}$ para responder a la *Cuestión 2*. El análisis de tales valores determinará la eficacia de tal ajuste. El número de ejecuciones seleccionado ha sido $n_r = 50$, partiendo cada una de ellas de una semilla aleatoria distinta. El número de instancias para cada tamaño es de $NI = 10$ instancias generadas aleatoriamente (véase la Sección 3.3.1) tal y como se indicó anteriormente, [159].

6.3.2. Estimación y validación de modelos Gamma

A continuación presentaremos los modelos Gamma estimados para determinar el rendimiento de los algoritmos CHC y PBIL ante el conjunto de tamaños propuesto en el diseño procedimental. La Tabla 6.9 muestra, para cada algoritmo y tamaño del problema, los parámetros de forma, $(\hat{k}_L)$, y escala, $(\hat{\beta}_L)$, que definen el comportamiento del algoritmo ante las instancias de cada tamaño. La estimación de tales parámetros se realizó a partir del procedimiento descrito en la Sección 6.2, resumido en las Ecuaciones 6.1 y 6.2.

Para validar los modelos Gamma estimados para cada algoritmo aplicaremos el procedimiento experimental descrito en la Sección anterior. El objetivo es el de calcular las diferencias entre los resultados de las ejecuciones y el modelo Gamma correspondiente tanto en longitud de ejecución (*Cuestión 1*) necesaria para alcanzar una calidad determinada ($p_{exito} = 0.9$, en nuestro caso)

	CHC			**PBIL**		
Tamaño	RL_{gamma}	RL_{emin}	RL_{emax}	RL_{gamma}	RL_{emin}	RL_{emax}
2^{60}	24324	21850	26627	62301	58320	68563
2^{70}	29673	26806	32367	75537	66563	78715
2^{80}	35021	32496	38440	88772	79638	96084
2^{100}	45718	41599	48806	115244	105886	124938

Tabla 6.10: Longitud de ejecución estimada y experimental para el 90 % de calidad.

	CHC			**PBIL**		
Tamaño	p_{gamma}	p_{emin}	p_{emax}	p_{gamma}	p_{emin}	p_{emax}
2^{60}	0.900	0.865	1.000	0.900	0.815	0.977
2^{70}	0.900	0.827	1.000	0.900	0.889	1.000
2^{80}	0.900	0.817	0.934	0.900	0.836	0.982
2^{100}	0.900	0.809	0.946	0.900	0.862	1.000

Tabla 6.11: Calidad experimental y estimada para la longitud de ejecución que el modelo asocia al 90 % de calidad.

como en calidad proporcionada (*Cuestión 2*) asociada a una longitud de ejecución concreta (la correspondiente según el modelo Gamma a $p_{exito} = 0.9$). El ajuste del modelo Gamma estimado para cada tamaño con cada algoritmo se realizará sobre las $NI = 10$ instancias generadas aleatoriamente para cada tamaño del algoritmo. Recordemos que para cada instancia se realizaron $n_r = 50$ ejecuciones de cada algoritmo.

Las Tablas 6.10 y 6.11 resumen los resultados empíricos permitiendo el contraste con el modelo. Así, la Tabla 6.10 responde a la *Cuestión 1*: la columna RL_{gamma} se corresponde con el valor de longitud de ejecución del modelo, $F_L^{-1}(p_{exito}; \hat{k}_L, \hat{\beta}_L)$, y las columnas RL_{emin} y RL_{emax} corresponden respectivamente al valor mínimo y máximo obtenido de las n_r ejecuciones para cada algoritmo. Por otra parte, la Tabla 6.11 responde a la *Cuestión 2*: la columna p_{gamma} es la calidad proporcionada por el modelo (por definición, $p_{exito} = 0.9$) y las columnas p_{emin} y p_{emax} indican respectivamente el valor mínimo y máximo obtenido de las n_r ejecuciones para cada algoritmo.

Si observamos la Tabla 6.10 podemos comprobar que la longitud de ejecución que el modelo predice, en cada caso, está dentro del intervalo definido por los resultados experimentales. Los extremos del intervalo RL_{emin} y RL_{emax} tienen el mismo orden de magnitud que el valor estimado. El valor proporcionado por el modelo puede considerarse como una buena referencia para decidir si es interesante ejecutar el algoritmo, puesto que en el peor de los

casos las diferencias obtenidas no son substanciales.

En lo que respecta a la Tabla 6.11, se puede percibir cómo los algoritmos logran grandes calidades de solución para todos los casos. En el peor caso, la probabilidad de éxito obtenida es superior a 0.8. Las variaciones en los resultados experimentales entre CHC y PBIL no presentan gran notoriedad. El valor proporcionado por el modelo se sitúa dentro del rango experimental. Las longitudes de los intervalos experimentales ilustran las posibilidades de CHC y PBIL para explotar espacios de búsqueda de grandes dimensiones.

Finalmente, para permitir una comparación normalizada de las respuestas a ambas *Cuestiones*, la Tabla 6.12 muestra un resumen de los resultados de la validación tanto en diferencias de longitud de ejecución como en diferencias de calidad de solución. La notación utilizada para las diferencias y las expresiones correspondientes a la obtención de cada una de ellas fueron definidas en el diseño experimental: Sección 6.3.1. Los resultados detallados de la validación se presentan en el Apéndice C.

Se puede observar cómo los resultados son muy similares para PBIL y CHC. El ajuste del modelo a la ejecución de los algoritmos presenta diferencias de una magnitud muy reducida, inferior en todo caso al 10 %. La variabilidad presenta una gran homogeneidad para los distintos tamaños del problema y no alcanza grandes proporciones. No se aprecia crecimiento en la diferencia ni en la variabilidad a medida que el tamaño del problema aumenta lo que apunta a una estabilidad manifiesta del modelo. No se aprecian diferencias significativas en cuanto a las respuestas a ambas *Cuestiones* (sentido directo e inverso del modelo).

Como conclusión final, el modelo Gamma puede proporcionar, por tanto, un boceto característico del comportamiento de los algoritmos PBIL y CHC cuando se enfrentan a instancias del problema de la selección de la solución deseada con grandes requerimientos computacionales.

Diferencia entre modelo Gamma y ejecución del algoritmo								
	Cuestión 1: Longitud de ejecución: Δrl_L							
Algoritmo	$\overline{\Delta rl_{60}}$	$\sigma_{\Delta rl_{60}}$	$\overline{\Delta rl_{70}}$	$\sigma_{\Delta rl_{70}}$	$\overline{\Delta rl_{80}}$	$\sigma_{\Delta rl_{80}}$	$\overline{\Delta rl_{100}}$	$\sigma_{\Delta rl_{100}}$
CHC	9.82	5.73	9.37	5.78	8.49	5.52	7.88	4.70
PBIL	8.22	5.38	8.04	4.93	9.26	4.91	8.27	4.55
	Cuestión 2: Calidad de la solución: Δp_L							
Algoritmo	$\overline{\Delta p_{60}}$	$\sigma_{\Delta p_{60}}$	$\overline{\Delta p_{70}}$	$\sigma_{\Delta p_{70}}$	$\overline{\Delta p_{80}}$	$\sigma_{\Delta p_{80}}$	$\overline{\Delta p_{100}}$	$\sigma_{\Delta p_{100}}$
CHC	9.13	5.58	9.85	5.85	6.51	4.87	7.62	4.98
PBIL	8.99	5.51	8.13	4.93	8.09	5.13	7.69	4.87

Tabla 6.12: Resumen de la validación de los modelos Gamma como representantes de la ejecución de los algoritmos CHC y PBIL.

Capítulo 7

Conclusiones y trabajo futuro

En este Capítulo se presentan las conclusiones sobre el trabajo realizado, las aportaciones a las que ha contribuido y se plantean algunas direcciones de futuros trabajos.

7.1. Conclusiones

La aplicación de la técnica de diseño paramétrico basado en resolución de restricciones geométricas presenta el problema de generar soluciones, cuando existen, que no necesariamente responden a aquello que el usuario espera. La aplicación eficaz de esta técnica precisa de la existencia de métodos que permitan al usuario seleccionar las soluciones que le interesan.

Uno de los métodos propuestos para abordar el llamado problema de la selección de la solución deseada consiste en la definición de un conjunto de restricciones adicionales que el usuario desea que cumpla la solución a seleccionar. De esta forma, el problema puede ser formulado como un problema de optimización combinatoria cuya solución supone la maximización del número de restricciones adicionales. Sin embargo, tal problema presenta gran dificultad para su resolución debido a su NP-completitud.

En la actualidad, se han desarrollado nuevos métodos para abordar la solución de los problemas de optimización combinatoria con mayores requerimientos computacionales: las Metaheurísticas. Estudios preliminares demostraron que los algoritmos más prometedores para el problema que tratamos correspondían claramente a las Metaheurísticas basadas en población: con-

cretamente a los algoritmos CHC y PBIL.

La obtención de un modelo para representar el rendimiento de las Metaheurísticas cuando son aplicadas para resolver problemas de grandes requerimientos computacionales es fundamental para determinar las garantías de calidad de solución frente al tiempo de ejecución disponible. La disponibilidad de este modelo es aún más importante para tamaños considerables del problema. En este trabajo, hemos ajustado, identificado, verificado y validado un modelo de rendimiento para los algoritmos CHC y PBIL cuando se enfrentan al problema de la selección de la solución deseada.

En el Capítulo 4, tras un análisis teórico del comportamiento de los algoritmos CHC y PBIL, se estudia la significatividad estadística de los parámetros de evolución de los mismos cuando se enfrentan al problema de la selección de la solución deseada. Se demuestra a partir de ANOVA que la influencia global de los valores de los parámetros de evolución sobre el resultado de los algoritmos es estadísticamente significativa dejando un margen muy reducido a la aleatoriedad. Teniendo en cuenta tal influencia se identifican los rangos de valores de los parámetros de evolución para los que el rendimiento de los algoritmos es estadísticamente óptimo. Así mismo, para cada tamaño del problema analizado se propone una configuración estadísticamente óptima. Puesto que el estudio estadístico exhaustivo realizado supone un elevado coste computacional, incluso para tamaños reducidos del problema, se utiliza un modelo simple de predicción, modelo de regresión lineal simple, para disponer de una configuración de parámetros para cada tamaño del problema.

En el Capítulo 5 se define una medida del rendimiento de los algoritmos independiente de la plataforma de ejecución. A partir de la misma se establece una representación del rendimiento de cada algoritmo sobre cada instancia del problema mediante las distribuciones de longitud de ejecución. La definición de dicha representación del rendimiento permite establecer un paralelismo con la función de distribución acumulativa de una distribución estadística teórica. Las técnicas estadísticas de bondad de ajuste permiten identificar una distribución estadística continua teórica como modelo de comportamiento para las diferentes instancias: la distribución Gamma. La semejanza en el comportamiento de las diferentes instancias generadas para cada tamaño del problema permite identificar un modelo Gamma representante de cada familia de distribuciones de longitud de ejecución Gamma. El comportamiento de los algoritmos se caracteriza tanto paramétrica como gráficamente a partir del modelo Gamma. Los tamaños del problema analizados son aquellos

para los que se optimizan estadísticamente los algoritmos CHC y PBIL en el Capítulo anterior.

Por último, en el Capítulo 6 se realiza la verificación del modelo propuesto para un conjunto de tamaños del problema superiores a los analizados en el Capítulo anterior. Las técnicas estadísticas de bondad de ajuste certifican la significatividad estadística del modelo Gamma para aproximar el rendimiento de los algoritmos estudiados ante el problema. Para comprobar la validez del modelo Gamma para tamaños del problema con requerimientos de cómputo muy elevados y para los que se desconoce la existencia de solución se propone una predicción para tal modelo. El estudio de las medidas descriptivas estadísticas de los modelos de comportamiento correspondientes a los tamaños analizados permite utilizar el modelo de regresión lineal simple para obtener los parámetros de la distribución Gamma que caracteriza el comportamiento de un algoritmo sobre un tamaño del problema dado.

Para validar el modelo Gamma se consideran dos cuestiones:

> Cuestión 1: Si se fija la calidad de la solución deseada, se desea una estimación para la longitud de ejecución que el algoritmo requiere para encontrar una instancia solución apropiada.
>
> Cuestión 2: Dada una longitud de ejecución límite, se desea una estimación de la calidad de la instancia solución que encontrará el algoritmo evolutivo.

La comparación de las propuestas del modelo y los resultados empíricos permite contrastar la fiabilidad del modelo con espacios de búsqueda entre 2^{60} y 2^{100} soluciones. Existe gran similaridad para PBIL y CHC. El ajuste del modelo a la ejecución de los algoritmos presenta diferencias de una magnitud muy reducida, con escasa variabilidad y con una gran homogeneidad para los distintos tamaños del problema. El modelo Gamma puede proporcionar un boceto característico del comportamiento de los algoritmos PBIL y CHC cuando se enfrentan al problema de la selección de la solución deseada.

7.2. Aportaciones

Las aportaciones de la presente tesis se detallan a continuación:

- M. V. Luzón, E. Barreiro and E. Yeguas. Aplicación de metaheurísticas

en el modelado paramétrico. III Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB), volumen 1, páginas 430-437, 2004.

- M.V. Luzón, E. Barreiro, E. Yeguas and R. Joan-Arinyo. GA and CHC. Two Evolutionary Algorithms to Solve the Root Identification Problem in Geometric Constraint Solving. Lecture Notes in Computer Science, number 3039, volumen 4, páginas 139-146, 2004.

- M. V. Luzón and E. Yeguas. Optimización de parámetros de PBIL y CHC aplicados al modelado paramétrico. I Congreso Español de Informática – Simposio sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB), volumen 2, páginas 565-572, 2005.

- R. Joan-Arinyo, M. V. Luzón and E. Yeguas. Parameter Tuning For PBIL and CHC Algorithms to Solve the Root Identification Problem in Geometric Constraint Solving. Herramientas Avanzadas en CAD, editors R. Joan-Arinyo and J. C. Torres Cantero and F. R. Feito Higueruela, páginas 77-110, ISBN 978-84-690-6855-7, 2007.

- R. Joan-Arinyo, M. V. Luzón and E. Yeguas. Parameter Tuning for PBIL Algorithm in Geometric Constraint Solving Systems. World Congress in Computer Science, Computer Engineering and Applied Computing. International Conference on Genetics and Evolutionary Methods, páginas 69-75, 2008.

- R. Joan-Arinyo, M. V. Luzón and E. Yeguas. Optimización estadística de CHC para Sistemas de Resolución de Restricciones Geométricas. VI Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB), 2009.

7.3. Trabajo futuro

El presente estudio plantea diversas directrices para posibles trabajos futuros, entre las que citamos las siguientes:

- Incluir los algoritmos CHC y PBIL en el *kernel* del sistema geométrico para el diseño paramétrico desarrollado por nuestro grupo.

- Mejorar los algoritmos en base a las conclusiones obtenidas del análisis de las RLDs. El objetivo principal será el de añadir nuevos componentes de diversificación y convergencia o modificar los existentes para superar las etapas de estancamiento en la búsqueda, tan presentes en la etapa final de la búsqueda. Podremos usar los resultados comparativos entre algoritmos, proporcionados por las RLDs empíricas y la función de distribución Gamma, para la hibridación con precisión de tales algoritmos con objeto de obtener mejores soluciones que las proporcionadas por los algoritmos individuales.

- Definir un método alternativo de regresión conjunta para la predicción de valores de los parámetros en el que se considere la interrelación entre los parámetros que constituyen el marco de ejecución y la interrelación entre los parámetros que conforman el modelo de rendimiento Gamma.

- Validar el modelo Gamma para tamaños del problema extremadamente superiores a los ya estudiados. Así mismo, se habrá de ampliar el conjunto de calidades de solución consideradas en función de los requerimientos del usuario.

- Aplicar el proceso a un conjunto diferente de Metaheurísticas de interés basadas en población para comprobar si el modelo Gamma es común para definir el rendimiento de tales algoritmos sobre el problema que tratamos.

- Ajustar los algoritmos evolutivos a requerimientos específicos del diseño paramétrico. En nuestro estudio aparecen dos fuentes distintas de aleatoriedad: la naturaleza no determinística del propio algoritmo y la selección aleatoria de las instancias del problema. Puesto que los problemas en el diseño paramétrico no son generados de forma aleatoria, el objetivo es el de llevar a cabo un estudio para demostrar la contribución de cada fuente de aleatoriedad al comportamiento final de los algoritmos.

Bibliografía

[1] E. H. L. Aarts and J. K. Lenstra. *Local Search in Combinatorial Optimization*. Princeton University Press, Princeton, NJ, 2003.

[2] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover, New York, 1964.

[3] R. K. Ahuja and J. B. Orlin. Use of representative operation counts in computational testing of algorithms. *INFORMS Journal on Computing*, 8(3):318–330, 1996.

[4] B. Aldefeld. Variation of geometric based on a geometric-reasoning method. *Computer-Aided Design*, 20(3):117–126, April 1988.

[5] J. Aldrich. R. A. Fisher and the making of maximum likelihood 1912-1922. *Statistical Science*, 12(3):162–176, August 1997.

[6] M. M. Ali. Durbin-Watson and generalized Durbin-Watson tests for autocorrelations and randomness. *Journal of Business and Economic Statistics*, 5(2):195–203, 1987.

[7] D. Andina. *Computational Intelligence*. Springer, 2007.

[8] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006.

[9] T. Bacck. Optimal mutation rates in genetic search. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 2–8. Morgan Kauffman, 1993.

[10] S. Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive

learning. Technical Report CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA, 1994.

[11] S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithm. Technical Report CMU-CS-95-141, Carnegie Mellon University, Pittsburgh, PA, 1995.

[12] E. Barreiro. *Modelización y optimización de algoritmos genéticos para la selección de la solución deseada en resolución constructiva de restricciones geométricas*. PhD thesis, Universidade de Vigo, June 2006.

[13] S. Bhansali, G. Kramer, and T. Hoar. A principled approach towards symbolic geometric constraint satisfaction. *Journal of Artificial Intelligence Research*, (4):419–443, 1996.

[14] M. Birattari, L. Paquete, T. Stützle, and K. Varrentrapp. Classification of metaheuristics and design of experiments for the analysis of components. Technical Report AIDA-01-05, Darmstadt University of Technology, November 2001.

[15] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308, 2003.

[16] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence: from natural to artificial systems*. Oxford University Press, Inc., New York, NY, USA, 1999.

[17] C. Borcea and I. Streinu. On the number of embeddings of minimally rigid graphs. *SoCG'02*, June 2002.

[18] W. Bouma, I. Fudos, C. Hoffman, J. Cai, and R. Paige. Geometric constraint solver. *Computer-Aided Design*, 27(6):487–501, June 1995.

[19] K. O. Bowman and L. R. Shenton. *Properties of Estimators for the Gamma Distribution*. CRC Press, 1988.

[20] H. I. Braun. *The Collected Works of John W. Tukey: Multiple Comparisons*, volume VIII. Chapman and Hall, New York, NY, 1994.

[21] P. Brucker. *Scheduling Algorithms*. Springer, Heidelberg, 2nd edition, 1998.

[22] B. Brüderlin. *Rule-Based Geometric Modelling*. PhD thesis, Institut für Informatik der ETH Zürich, 1988.

[23] B. Brüderlin. Symbolic computer geometry for computer aided geometric design. In *Advances in Design and Manufacturing Systems*, Tempe, AZ, Jan. 8-12 1990. Proceedings NSF Conference.

[24] B. Brüderlin. Using geometric rewrite rules for solving geometric problems symbolically. In *Theoretical Computer Science 116*, pages 291–303. Elsevier Science Publishers B.V., 1993.

[25] G. C. Canavos. *Applied probability and statistical methods.* Addison-Wesley, 1984.

[26] I. M. Chakravarti, R. G. Laha, and J. Roy. *Handbook of Methods of Applied Statistics: Techniques of Computation, Descriptive Methods and Statistical Inference*, volume I. John Wiley and Sons, New York, 1967.

[27] G. M. Clarke and D. Cooke. *A Basic Course in Statistics.* Hodder Arnold, 1998.

[28] W. J. Cody. An overview of software development for special functions. *Lecture Notes in Mathematics*, 506, 1976.

[29] J. Cohen. Eta-squared and partial eta-squared in fixed factor ANOVA designs. *Educational and Psychological Measurement*, 33(1):107–112, 1973.

[30] P. R. Cohen. *Empirical Methods for Artificial Intelligence.* MIT Press, 1995.

[31] H. Cundy and A. Rollet. *Mathematical Models.* Oxford University Press, 1961. Second edition.

[32] In D. Whitley and D. Schaffer, editors, *Proceedings of the Internacional Workshop on Combinations of Genetic Algorithms and Neural Networks.* IEEE Computer Society, 1992.

[33] J. L. Devore. *Probability and Statistics for Engineering and the Sciences.* Duxburg and Brooks Cole, Pacific Grove, CA, 6th edition, 2004.

[34] A. R. DiDonato and J. Alfred H Morris. Computation of the incomplete gamma function ratios and their inverse. *ACM Trans. Math. Softw.*, 12(4):377–393, 1986.

[35] M. Dorigo and T. Stützle. *Ant Colony Optimization.* MIT Press, 2004.

[36] N. R. Draper and H. Smith. *Applied Regression Analysis*. John Wiley and Sons, Inc., New York, NY, 3rd edition, 1998.

[37] A. Eiben and Z. Ruttkay. Self-adaptivity for constraint satisfaction: Learning penalty functions. In *Third IEEE World Conference on Evolutionary Computation*, pages 258–261, Nagoya, Japan, 1996. IEEE Service Center.

[38] A. Eiben and Z. Ruttkay. Constraint-satisfaction problems. In T. Bäck, D. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, pages C5.7:1–C5.7:5. Institute of Physics Publishing Ltd and Oxford University Press, 1997.

[39] A. P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. Wiley, 2005.

[40] L. J. Eshelman. The CHC adaptative search algorithm: How to safe search when engaging in nontraditional genetic recombination. *Foundations of Genetic Algorithms I*, pages 265–283, 1991.

[41] C. Essert-Villard, P. Schreck, and J.-F. Dufourd. Skecth-based pruning of a solution space within a formal geometric constraint solver. *Artificial Intelligence*, 124:139–159, 2000.

[42] R. A. Fisher. Theory of statistical stimation. In *Proc. of the Cambridge Philosophical Society*, pages 700–725, 1925.

[43] D. B. Fogel and L. C. Stayton. On the effectiveness of crossover in simulated evolutionary optimization. *Biosystems Journal*, 1993.

[44] L. J. Fogel. Toward inductive inference automata. In *Proceedings of the International Federation for Information Processing Congress*, pages 395–399, Munich, Germany, 1962.

[45] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artifical Intelligence through Simulated Evolution*. Wiley, 1966.

[46] I. Fudos. *Constraint Solving for Computer Aided Design*. PhD thesis, Purdue University, Department of Computer Sciences, 1995.

[47] I. Fudos and C. Hoffmann. Constraint-based parametric conics for CAD. *Computer Aided Design*, 28(2):91–100, 1996.

[48] I. Fudos and C. Hoffmann. Correctness proof of a geometric constraint solver. *International Journal of Computational Geometry and Applications*, 6(4):405–420, 1996.

[49] I. Fudos and C. Hoffmann. A graph-constructive approach to solving systems of geometric constraints. *ACM Transactions on Graphics*, 16(2):179–216, April 1997.

[50] P. A. Games and J. F. Howell. Pairwise multiple comparison procedures with unequal n's and/or variances: A monte carlo study. *Journal of Educational Statistics*, 1(2):113–125, 1976.

[51] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., San Francisco, CA, 1979.

[52] F. Glover and M. Laguna. Tabu search. In C. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, Oxford, England, 1993. Blackwell Scientific Publishing.

[53] F. Glover, M. Laguna, and R. Martí. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 39(3):653–684, 2000.

[54] D. Goldberg. *Genetic Algorithms in Search, Optimization Machine Learning*. Addison Wesley, 1989.

[55] D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms*, pages 69–93. Morgan Kaufmann, 1991.

[56] D. E. Goldberg, D. Kalyanmoy, and J. H. Clark. Genetic algorithms, noise and the sizing of populations. *Complex Systems*, pages 333–362, 1992.

[57] J. Groβ. Linear regression. *Lecture Notes in Statistics*, 175(12), 2003.

[58] F. Herrera, M. Lozano, and J. L. Verdegay. Algoritmos genéticos: Fundamentos, extensiones y aplicaciones. *Arbor*, 152(597):9–40, 1998.

[59] A. Hertz and D. Klober. A framework for the description of evolutionary algorithms. *European Journal of Operational Research*, 126(1):1–12, October 2000.

[60] T. Hill and P. Lewicki. *Statistics: Methods and Applications*. Statsoft, Inc., Tulsa, OK, 2006.

[61] C. Hoffmann and R. Joan-Arinyo. Symbolic constraints in constructive geometric constraint solving. *Journal of Symbolic Computation*, 23:287–300, 1997.

[62] C. Hoffmann and R. Joan-Arinyo. A brief on constraint solving. *Computer-Aided Design and Applications*, 2(5):655–663, 2005.

[63] C. Hoffmann, A. Lomonosov, and M. Sitharam. Finding solvable subsets of constraint graphs. In *Principles and Practice of Constraint Programming*, pages 463–477. Schloss Hagenberg, Austria, 1977.

[64] C. Hoffmann, A. Lomonosov, and M. Sitharam. Geometric constraint decomposition. In B. Brüderlin and D. Roller, editors, *Geometric Constraint Solving and Applications*, pages 171–195. Berlin, Germany, 1998.

[65] C. Hoffmann, A. Lomonosov, and M. Sitharam. Decomposition Plans for Geometric Constraint. Problems, Part II: New Algorithms. *Journal of Symbolic Computation*, 31:409–427, 2001.

[66] C. Hoffmann, A. Lomonosov, and M. Sitharam. Decomposition Plans for Geometric Constraint. Systems, Part I: Performance Measurements for CAD. *Journal of Symbolic Computation*, 31:367–408, 2001.

[67] J. H. Holland. *Adaptation in natural and artificial systems.* MIT Press, Cambridge, MA, 1975.

[68] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.* MIT Press, Cambridge, MA, 1992.

[69] J. H. Holland. *Hidden order: how adaptation builds complexity.* Perseus Books, Cambridge, MA, 1995.

[70] H. H. Hoos and T. Stützle. Evaluating las vegas algorithms – pitfalls and remedies. In *Proceedings of the 14th Conference on Uncertainly in Artificial Intelligence*, pages 238–245. Morgan Kaufmann, 1998.

[71] R. Joan-Arinyo, M. V. Luzón, and E. Yeguas. Ajuste, optimización y representación del rendimiento de PBIL y CHC aplicados al problema de la selección de la solución deseada. Technical Report LSI-07-41-R, Departament de Llenguatges i sistemes informàtics, Universitat Politècnica de Catalunya, Barcelona, Spain, 2007.

[72] R. Joan-Arinyo, M. V. Luzón, and E. Yeguas. Comparación basada en distribuciones de longitud de tiempo de ejecución para PBIL y CHC aplicados al problema de la selección de la solución deseada. Technical Report LSI-07-37-R, Departament de Llenguatges i sistemes informàtics, Universitat Politècnica de Catalunya, Barcelona, Spain, 2007.

[73] R. Joan-Arinyo, M. V. Luzón, and E. Yeguas. Parameter tuning for PBIL and CHC algorithms to solve the root identification problem in geometric constraint solving. Technical Report LSI-07-03-R, Departament de Llenguatges i sistemes informàtics, Universitat Politècnica de Catalunya, Barcelona, Spain, 2007.

[74] R. Joan-Arinyo, M. V. Luzón, and E. Yeguas. Parameter tuning for PBIL and CHC algorithms to solve the root identification problem in geometric constraint solving. In R. Joan-Arinyo, J. C. T. Cantero, and F. R. F. Higueruela, editors, *Herramientas Avanzadas en CAD*, pages 77–110. 2007.

[75] R. Joan-Arinyo, M. V. Luzón, and E. Yeguas. Predicción del rendimiento de CHC y PBIL aplicados al problema de la selección de la solución deseada. Technical Report LSI-07-42-R, Departament de Llenguatges i sistemes informàtics, Universitat Politècnica de Catalunya, Barcelona, Spain, 2007.

[76] R. Joan-Arinyo, M. V. Luzón, and E. Yeguas. Estimación y validación de distribuciones de longitud de ejecución para chc y pbil aplicados al problema de la selección de la solución deseada. Technical Report LSI-08-27-R, Departament de Llenguatges i sistemes informàtics, Universitat Politècnica de Catalunya, Barcelona, Spain, 2008.

[77] R. Joan-Arinyo, M. V. Luzón, and E. Yeguas. Optimización estadística de CHC para sistemas de resolución de restricciones geométricas. In *VI Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB)*, 2009.

[78] R. Joan-Arinyo, M. V. Luzón, and E. Yeguas. Parameter tuning for PBIL algorithm in geometric constraint solving systems. In *World Congress in Computer Science, Computer Engineering and Applied Computing. International Conference on Genetics and Evolutionary Methods*, pages 69–75, 2008.

[79] R. Joan-Arinyo and A. Soto. A correct rule-based geometric constraint solver. *Computer & Graphics*, 21(5):599–609, 1997.

[80] R. Joan-Arinyo and A. Soto. A ruler-and-compass geometric constraint solver. In M. Pratt, R. Sriram, and M. Wozny, editors, *Product Modeling for Computer Integrated Design and Manufacture*, pages 384 – 393. Chapman and Hall, London, 1997.

[81] R. Joan-Arinyo and A. Soto-Riera. Combining constructive and equational geometric constraint solving techniques. *ACM Transactions on Graphics*, 18(1):35–55, January 1999.

[82] N. L. Johnson, S. Kotz, and N. Balakrishnan. *Continuous Univariate Distributions*, volume 1. John Wiley and Sons, Inc., New York, NY, USA, 2nd edition, 1994.

[83] K. A. D. Jong. *Evolutionary Computation: a unified approach.* MIT Press, Cambridge, MA, 2006.

[84] K. D. Jong. *An Analysis of the Behavior of a Class of Genetic Adaptative Systems.* PhD thesis, 1975.

[85] A. Juels, S. Baluja, and A. Sinclair. The equilibrium genetic algorithm and the role of crossover. Unpublished manuscript, accesible at `http://citeseer.ist.psu.edu/juels93equilibrium.html`, 1993.

[86] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[87] S. Kleene. *Mathematical Logic.* John Wiley and Sons, New York, 1967.

[88] K.-R. Koch. *Parameter Estimation and Hypothesis Testing in Linear Models.* Springer Verlag, Berlin; Heidelberg; New York, 1999.

[89] T. Kohonen. *Self-Organizing and Associate Memory.* Springer Verlag, Berlin, Germany, 3rd edition, 1989.

[90] T. Kohonen, G. Barna, and R. Chrisley. Statistical pattern recognition with neural networks: Benchmarking studies. In *IEEE International Conference on Neural Networks*, volume 1, pages 61–68, New York, NY, 1988. IEEE.

[91] G. Kramer. *Solving Geometric Constraints Systems.* MIT Press, 1992.

[92] G. Kwaiter, V. Gaildrat, and R. Caubet. Interactive constraint system for solid modeling objects. In C. Hoffmann and W. Bronsvort, editors, *Fourth Symposium on Solid Modeling and Applications*, pages 265–270. ACM SIGGRAPH, May 1997.

[93] G. Laman. On graphs and rigidity of plane skeletal structures. *Journal of Engineering Mathematics*, 4(4):331–340, October 1970.

[94] H. Lamure and D. Michelucci. Solving geometric constraints by homotopy. In C. Hoffmann and J. Rossignac, editors, *Third Symposium on Solid Modeling and Applications*, pages 263–269, Salt Lake City, Utah USA, May 17-19 1995. ACM Press.

[95] P. Larrañaga and J. A. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Springer, 2002.

[96] R. Latham and A. Middleditch. Connectivity analysis: a tool for processing geometric constraints. *Computer Aided Design*, 28(11):917–928, November 1996.

[97] J. Lee and K. Kim. A 2-d geometric constraint solver using dof-based graph reduction. *Computer Aided Design*, 30(11):883–896, September 1998.

[98] W. Leler. *Constraint Programming Languages: Their Specification and Generation*. Addison Wesley, 1988.

[99] H. Levene. Contributions to probability and statistics: Essays in honor of Harold Hotelling. pages 278–292. Stanford University Press, 1960.

[100] G. Li and Z. Xu. A theoretical analysis on mutation operator of standard genetic algorithm. *Wuhan University Journal of Natural Sciences*, 1(3-4):599–604, December 1996.

[101] F. G. Lobo, C. F. Lima, and Z. Michalewicz, editors. *Parameter Setting in Evolutionary Algorithms*, volume 54 of *Studies in Computational Intelligence*. Springer, 2007.

[102] L. Lovász and Y. Yemini. On generic rigidity in the plane. *SIAM Journal on Algebraic and Discrete Methods*, 3(1):91–98, 1982.

[103] A. Lubiw. Some NP-complete problems similar to graph isomorphism. *SIAM Journal on Computing*, 10:11–21, 1981.

[104] M. Luby, A. Sinclair, and D. Zuckerman. Optimal speedup of las vegas algorithms. In *Israel Symposium on Theory of Computing Systems*, pages 128–133, 1993.

[105] M. Luzón, E. Barreiro, E. Yeguas, and R. Joan-Arinyo. GA and CHC. two evolutionary algorithms to solve the root identification problem in geometric constraint solving. *Lecture Notes in Computer Science*, 4(3039):139–146, 2004.

[106] M. Luzón, A. Soto, J. Gálvez, and R. Joan-Arinyo. Searching the solution space in constructive geometric constraint solving with genetic algorithms. *Applied Intelligence*, 22:109–124, 2005.

[107] M. V. Luzón. *Resolución de Restricciones Geométricas. Selección de la Solución Deseada*. PhD thesis, Dpto. de Informática, Universidade de Vigo, Ourense, Spain, September 2001.

[108] M. V. Luzón, E. Barreiro, and E. Yeguas. Aplicación de metaheurísticas en el modelado paramétrico. In *III Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB)*, volume 1, pages 430–437, 2004.

[109] M. V. Luzón, R. Joan-Arinyo, and A. Soto. Genetic algorithms for root multiselection in constructive geometric constraint solving. *Computer & Graphics*, 27:51–60, 2003.

[110] M. V. Luzón and E. Yeguas. Optimización de parámetros de pbil y chc aplicados al modelado paramétrico. In *I Congreso Español de Informática – Simposio sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB)*, volume 2, pages 565–572, 2005.

[111] N. Mata. Solving incidence and tangency constraints in 2D. Technical Report LSI-97-3R, Department LSI, Universitat Politècnica de Catalunya, 1997.

[112] N. Mata. *Constructible Geometric Problems with Interval Parameters*. PhD thesis, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Barcelona, Spain, 2000.

[113] C. R. Mehta, N. R. Patel, and A. A. Tsiatis. Exact significance testing to establish treatment equivalence with ordered categorical data. *Biometrics*, 40(3):819–825, 1984.

[114] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1996.

[115] B. L. Miller and D. E. Goldberg. Genetic algorithms, selection schemes, and the varying effects of noise. *Evolutionary Computation*, 4:113–131, 1996.

[116] R. G. Miller. *Simultaneous Statistical Inference*. Springer Verlag, New York, NY, 2nd edition, 1981.

[117] N. Mladenović and P. Hansen. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130:449–467, 2001.

[118] H. Mühlenbein. Parallel genetic algorithms, population genetics and combinatorial optimization. In Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 416–421, San Mateo, CA, 1989. Morgan Kaufmann.

[119] G. L. Nemhauser and A. L. Wolsey. *Integer and Combinatorial Optimization*. John Wiley and Sons, New York, 1988.

[120] I. H. Osman and G. Laporte. Metaheuristics: A bibliography. *Annals of Operations Research*, 63(5):511–623, October 1996.

[121] J. Owen. Algebraic solution for geometry from dimensional constraints. In R. Rossignac and J. Turner, editors, *Symposium on Solid Modeling Foundations and CAD/CAM Applications*, pages 397–407, Austin, TX, June 5-7 1991. ACM Press.

[122] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization - Algorithms and Complexity*. Dover Publications, Inc., New York, NY, 1982.

[123] P. M. Pardalos, F. Rendl, and H. Wolkowicz. The quadratic assignment problem: A survey and recent developments. In *In Proceedings of the DIMACS Workshop on Quadratic Assignment Problems, volume 16 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 1–42. American Mathematical Society, 1994.

[124] M. Pelikan, D. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. Technical Report 99018, IlliGAL, University of Illinois, Illinois, IL, 1999.

[125] A. Piszcz and T. Soule. Dynamics of evolutionary robustness. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and*

evolutionary computation, pages 871–878, New York, NY, USA, 2006. ACM.

[126] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, New York, NY, 3rd edition, 2007.

[127] H. Ramalhino, O. Martin, and T. Stützle. Iterated local search. In F. Glover and G. K. eds., editors, *Handbook of Metaheuristics*, pages 321–353, 2002.

[128] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, 1973.

[129] J. Richardson, M. Palmer, G. Liepins, and M. Hilliard. Some guidelines for genetic algorithms with penalty functions. In *Third IEEE International Conference on Genetic Algorithms*, pages 191–197, San Mateo, CA, 1989. Morgan Kaufmann.

[130] E. A. Robinson. *Least Squares Regression Analysis in Terms of Linear Algebra*. Goose Pond Press, Houston, TX, 1981.

[131] M. Rocha and J.Ñeves. Preventing premature convergence to local optima in genetic algorithms via random offspring. *Lecture Notes in Computer Science*, 1611:127–136, 1999.

[132] U. Rudolph. Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks*, 5:96–101, 1994.

[133] O. Salomons, F. van Slooten, F. van Houten, and H. Kals. Conceptual graphs in constraint based re-design. In C. Hoffmann and J. Rossignac, editors, *Third Symposium on Solid Modeling and Applications*, pages 55–64, Salt Lake City, Utah USA, May 17-19 1995. ACM Press.

[134] B. Sareni and L. Krähenbühl. Fitness sharing and niching methods revisited. *IEEE Transaction on Evolutionary Computation*, 2(3):97–106, 1998.

[135] T. Schmelzer and L.Ñ. Trefethen. Computing the gamma function using contour integrals and rational approximations. *SIAM Journal on Numerical Analysis*, 45(2):558–571, 2007.

[136] S. S. Shapiro and C. W. Brain. A review of distributional testing procedures and development of a censored sample distributional test. In C. Taillie, G. P. Patil, and B. A. Baldessari, editors, *Statistical distributions in scientific work*, volume 5, pages 1–24, Dordrecht, Holland, 1981. D. Reidel Publishing Company.

[137] D. Sheskin. *Handbook of parametric and nonparametric statistical procedures*. Chapman and Hall/CRC, Boca Raton, Florida, 4th edition, 2007.

[138] G. W. Snedecor and W. G. Cochran. *Statistical Methods*. Iowa State University Press, 8th edition, 1989.

[139] W. Sohrt. Interaction with constraints in three-dimensional modeling. Master's thesis, Dept of Computer Science, The University of Utah, March 1991.

[140] W. Sohrt and B. Brüderlin. Interaction with constraints in 3D modeling. *International Journal of Computational Geometry & Applications*, 1(4):405–425, 1991.

[141] L. Solano and P. Brunet. A system for constructive constraint-based modeling. In B. Falcidieno and T. Kunii, editors, *Modeling in Computer Graphics*. Springer Verlag, 1993.

[142] W. M. Spears. Crossover or mutation? In Withley, editor, *Foundations of Genetic Algorithms-2*, volume 2, pages 221–237, San Mateo, CA, 1992. Morgan Kaufmann Publishers.

[143] M. A. Stephens. EDF statistics for goodness of fit and some comparisons. *Journal of the American Statistical Association*, 69:730–737, 1974.

[144] T. Stützle. *Local Search Algorithms for Combinatorial Problems – Analysis, Algorithms and New Applications*. PhD thesis, 1999.

[145] G. Sunde. A CAD system with declarative specification of shape. *Eurographics Workshop on Intelligent CAD Systems*, pages 90–105, April 1987.

[146] K. Thulasiraman and M.Ñ. S. Swamy. *Graphs: Theory and Algorithms*. John Wiley and Sons, Inc., 1992.

[147] L. Tian. The nature of crossover operator in genetic algorithms. *Lecture Notes in Computer Science*, 2005/2001:619–623, 2001.

[148] P. Todd. A k-tree generalization that characterizes consistency of dimensioned engineering drawings. *SIAM J. Disc. Math*, 2(2):255–261, 1989.

[149] J. W. Tukey. *Exploratory Data Analysis.* Addison-Wesley, Reading, MA, 1977.

[150] R. Veltkamp and F. Arbab. Geometric constraint propagation with quantum labels. In B. Falcidieno, I. Herman, and C. Pienovi, editors, *Eurographics Workshop on Computer Graphics and Mathematics*, pages 211–228. Springer, 1992.

[151] A. Verroust. *Etude de Problèmes Liés à la Définition, la Visualisation et l'Animation d'Objects Complexes en Informatique Graphique.* PhD thesis, Universite de Paris-Sud, Centre d'Orsay, 1990. (Written in French).

[152] A. Verroust, F. Schonek, and D. Roller. Rule-oriented method for parameterized computer-aided design. *Computer Aided Design*, 24(10):531–540, October 1992.

[153] S. Vila. *Contribution to Geometric Constraint Solving in Cooperative Engineering.* PhD thesis, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Barcelona, Spain, September 2003.

[154] S. Voβ, S. Martello, I. H. Osman, and R. C. *Metaheuristics – Advances and Trends in Local Search Paradigms for Optimization.* Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

[155] S. Weisberg. *Applied Linear Regression.* John Wiley and Sons, Inc., New York, NY, 3rd edition, March 2005.

[156] D. Whitley and D. Whitley. A genetic algorithm tutorial. *Statistics and Computing*, 4:65–85, 1994.

[157] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transaction on Evolutionary Computation*, 1(1):67–82, 1997.

[158] Y. Yamaguchi and F. Kimura. A constraint modeling system for variational geometry. In J. T. M.J. Wozny and K. Preiss, editors, *Geometric Modeling for Product Engineering*, pages 221–233. Elsevier North Holland, 1990.

[159] E. Yeguas. Root identification problem in geometric constraint solving. September 2008. `http://www.uco.es/~in1yeboe/benchmark.html`.

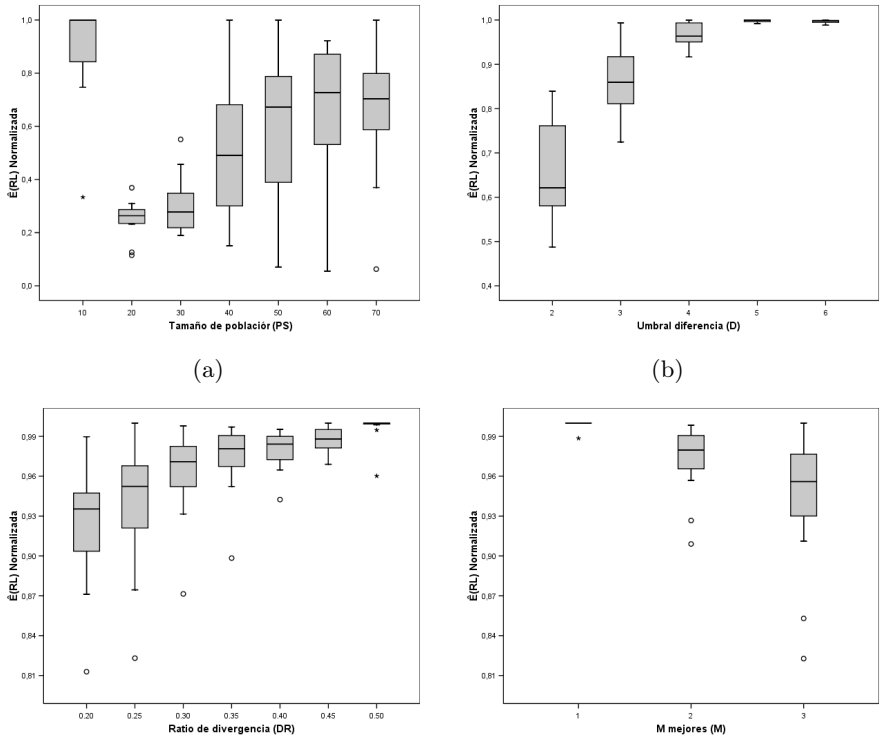
[160] E. Yeguas, M. Luzón, E. Barreiro, and R. Joan-Arinyo. Estudio e implementación de metaheurísticas para solucionar el problema de la selección de la solución deseada. Technical Report LSI-08-25-R, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, 2008.

[161] E. Yeguas, M. Luzón, E. Barreiro, and R. Joan-Arinyo. Experimentos y análisis de resultados tras la aplicación de metaheurísticas al problema de la selección de la solución deseada. Technical Report LSI-08-26-R, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, 2008.

Apéndices

Apéndice A

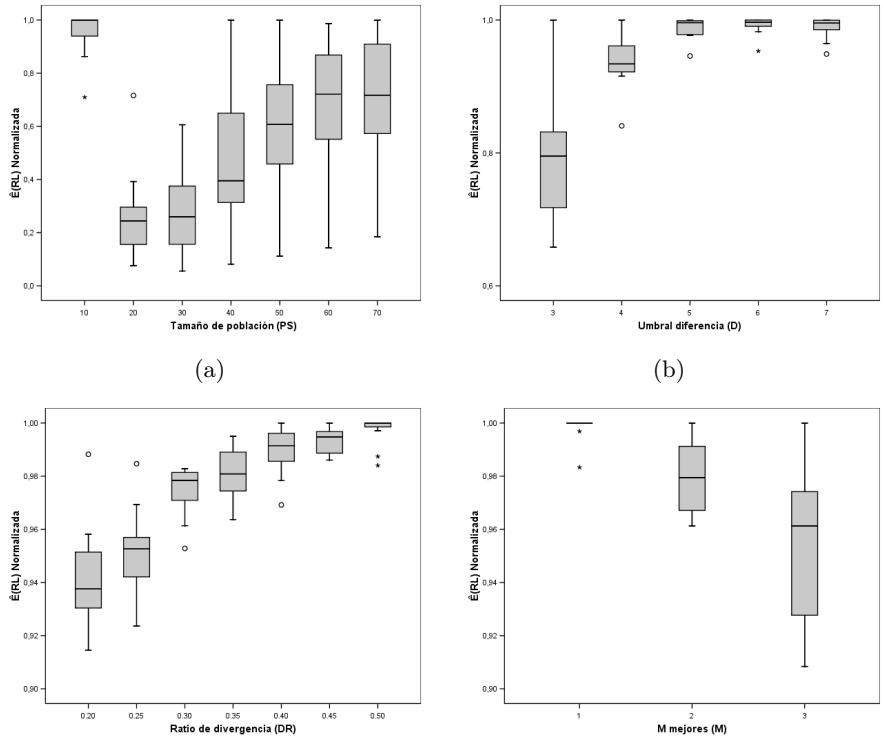
Gráficas de influencia individual de los parámetros



(a) (b)

(c) (d)

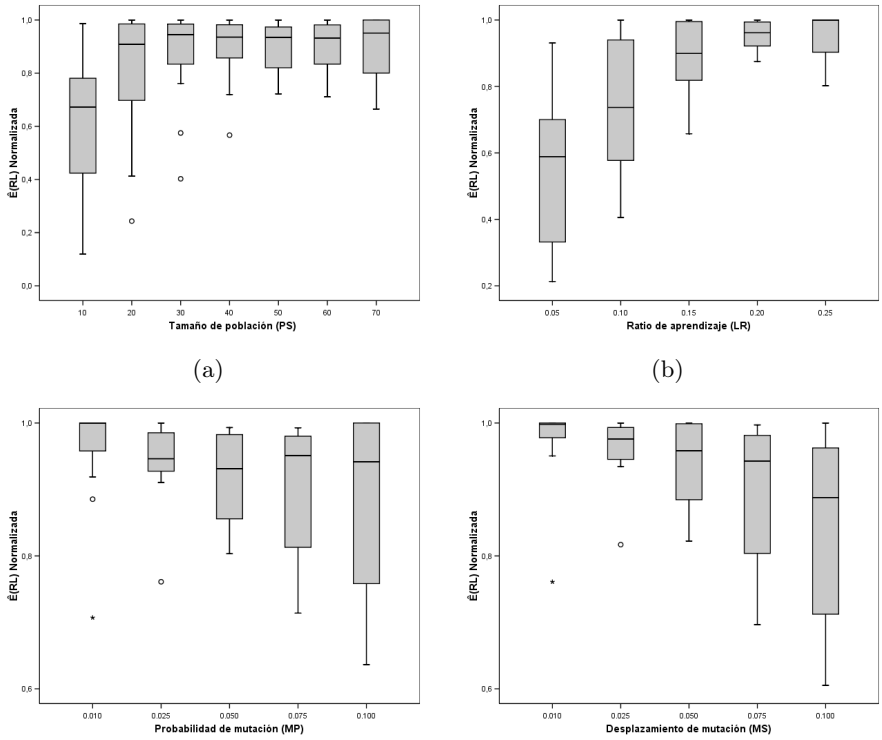
Figura A.1: Influencia individual normalizada en CHC para 2^{19}: a) PS, b) D, c) DR, d) M.



(a) (b)

(c) (d)

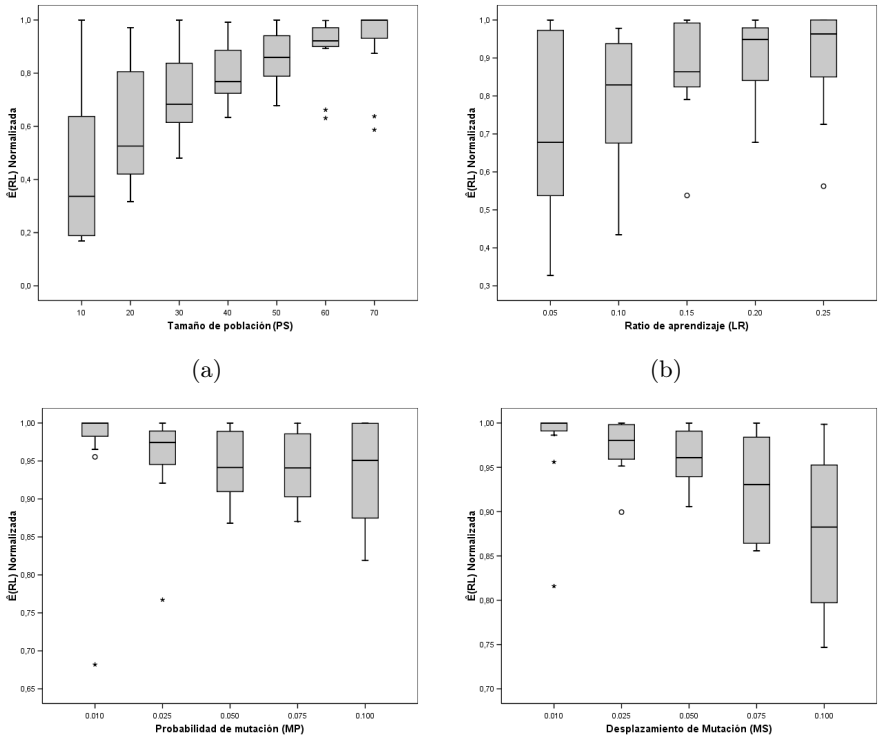
Figura A.2: Influencia individual normalizada en CHC para 2^{20}: a) PS, b) D, c) DR, d) M.



(a) (b)

(c) (d)

Figura A.3: Influencia individual normalizada en PBIL para 2^{19}: a) PS, b) LR, c) MP, d) MS.



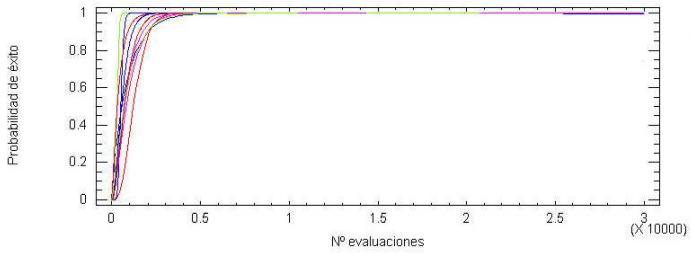
(a) (b)

(c) (d)

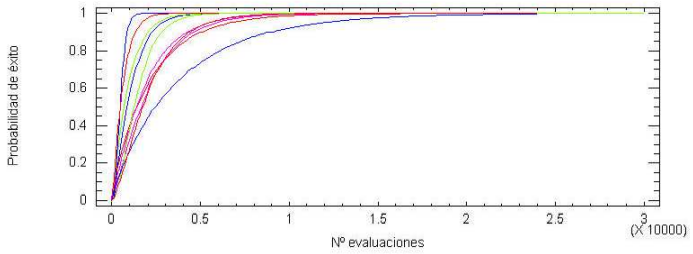
Figura A.4: Influencia individual normalizada en PBIL para 2^{20}: a) PS, b) LR, c) MP, d) MS.

Apéndice B

Familias de RLDs Gamma estimadas

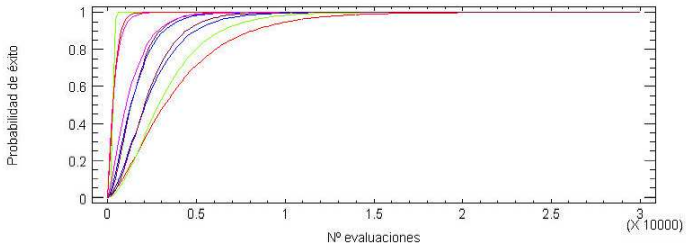


(a)

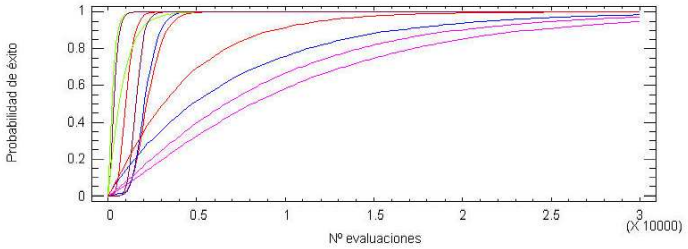


(b)

Figura B.1: RLDs Gamma de a) CHC y b) PBIL para el tamaño 2^{19}.

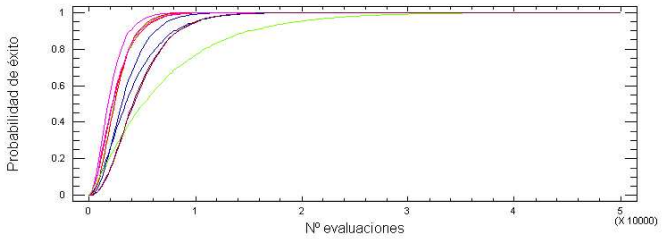


(a)

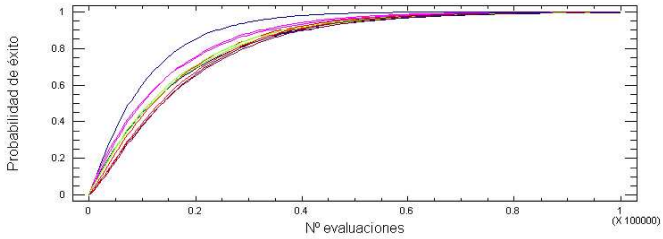


(b)

Figura B.2: RLDs Gamma de a) CHC y b) PBIL para el tamaño 2^{20}.

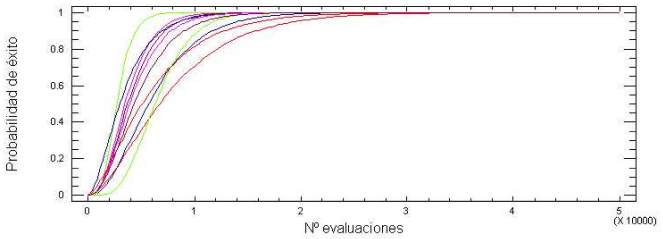


(a)

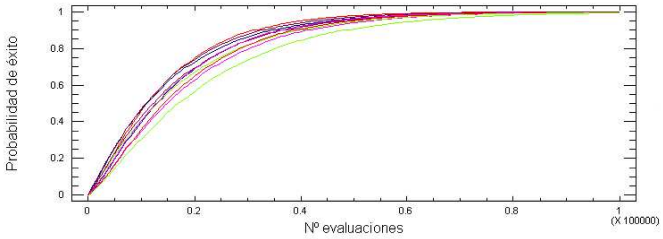


(b)

Figura B.3: RLDs Gamma de a) CHC y b) PBIL para el tamaño 2^{30}.

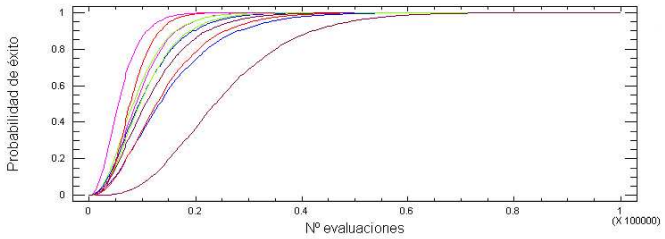


(a)

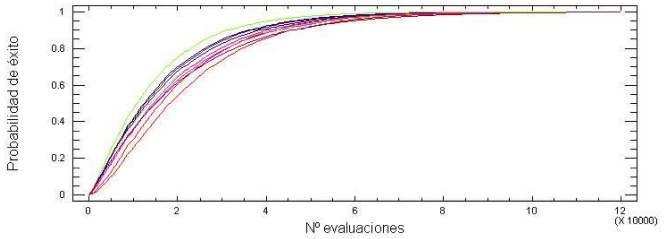


(b)

Figura B.4: RLDs Gamma de a) CHC y b) PBIL para el tamaño 2^{35}.

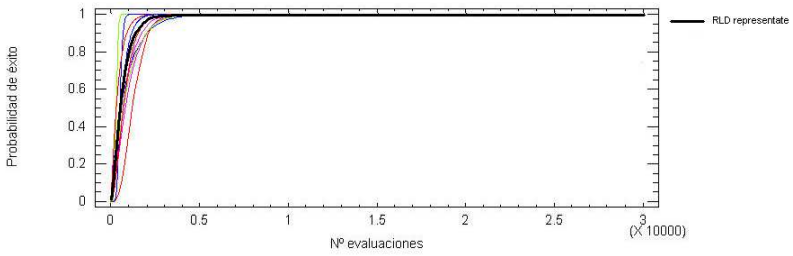


(a)

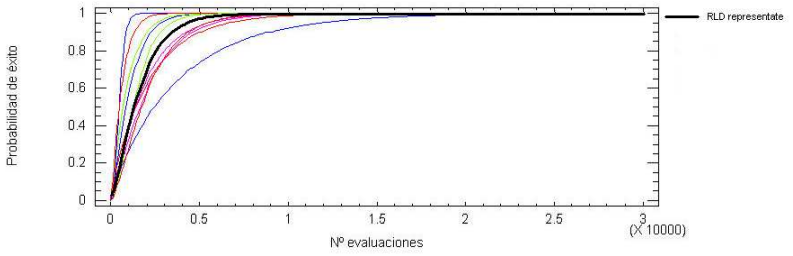


(b)

Figura B.5: RLDs Gamma de a) CHC y b) PBIL para el tamaño 2^{50}.

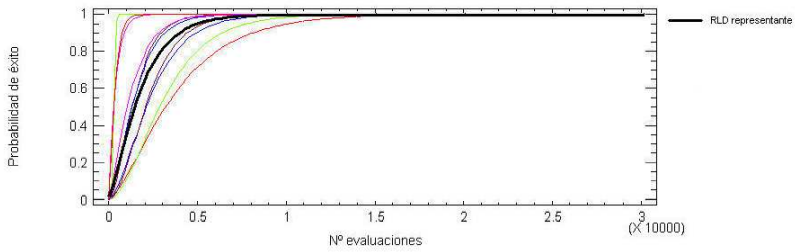


(a)

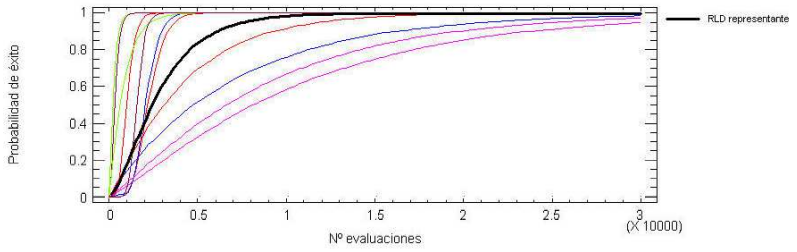


(b)

Figura B.6: RLDs Gamma representantes de a) CHC y b) PBIL para el tamaño 2^{19}.

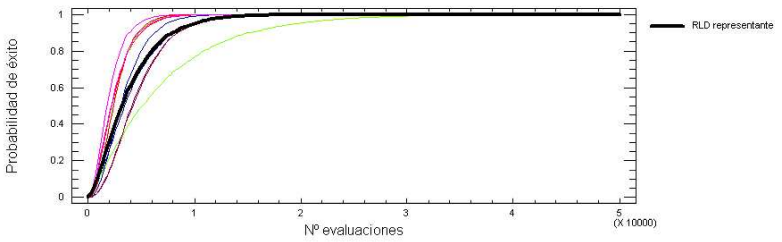


(a)

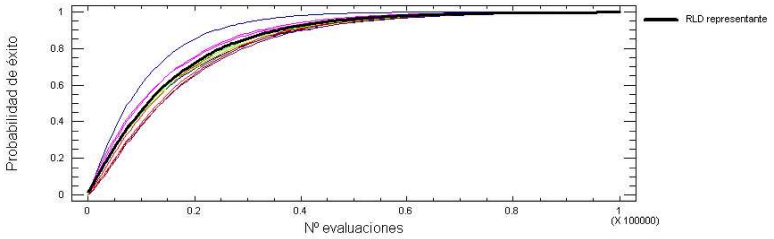


(b)

Figura B.7: RLDs Gamma representantes de a) CHC y b) PBIL para el tamaño 2^{20}.

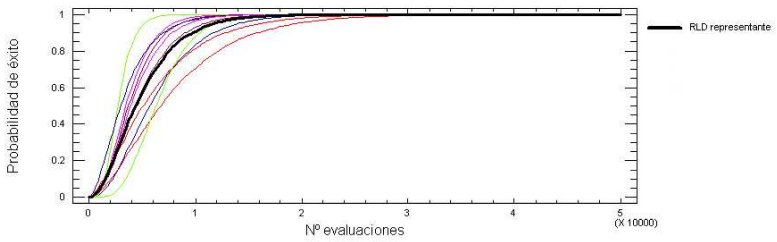


(a)

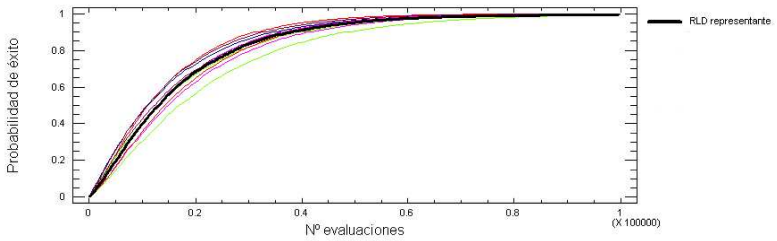


(b)

Figura B.8: RLDs Gamma representantes de a) CHC y b) PBIL para el tamaño 2^{30}.

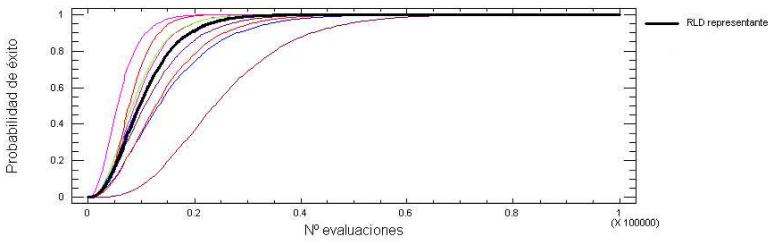


(a)

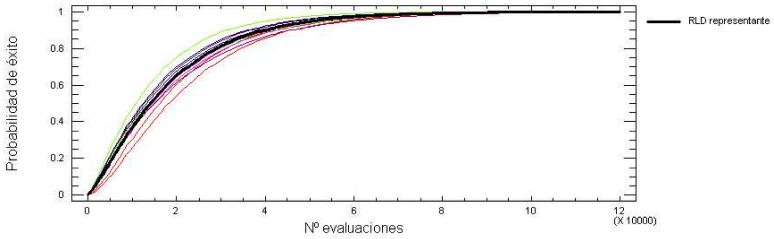


(b)

Figura B.9: RLDs Gamma representantes de a) CHC y b) PBIL para el tamaño 2^{35}.



(a)



(b)

Figura B.10: RLDs Gamma representantes de a) CHC y b) PBIL para el tamaño 2^{50}.

Apéndice C

Resultados de la validación del modelo

Δrl_L	**Tamaño del problema**			
CHC	2^{60}		2^{70}	
Instancia	$\overline{\Delta rl_{60,I}}$	$\sigma_{\Delta rl_{60,I}}$	$\overline{\Delta rl_{70,I}}$	$\sigma_{\Delta rl_{70,I}}$
1	7.27	5.97	7.36	5.12
2	9.92	5.89	9.64	5.60
3	7.08	5.27	12.38	7.52
4	8.86	5.89	7.36	5.58
5	10.38	5.83	12.39	5.68
6	9.86	6.72	7.94	5.92
7	11.82	5.65	8.43	5.46
8	9.95	5.41	8.87	5.83
9	12.38	5.34	8.75	5.05
10	10.68	5.30	10.58	6.01
Resumen ajuste	$\overline{\Delta rl_{60}}$ 9.82	$\sigma_{\Delta rl_{60}}$ 5.73	$\overline{\Delta rl_{70}}$ 9.37	$\sigma_{\Delta rl_{70}}$ 5.78

Tabla C.1: Resultados de la validación para la predicción de la longitud de ejecución de CHC: 2^{60} y 2^{70}.

Δrl_L	**Tamaño del problema**			
CHC	2^{80}		2^{100}	
Instancia	$\overline{\Delta rl_{80,I}}$	$\sigma_{\Delta rl_{80,I}}$	$\overline{\Delta rl_{100,I}}$	$\sigma_{\Delta rl_{100,I}}$
1	8.68	4.99	9.87	4.87
2	7.11	5.67	7.20	4.94
3	7.50	5.34	6.82	3.81
4	7.56	5.24	8.02	3.33
5	10.44	5.61	9.41	5.91
6	9.56	5.54	6.55	4.02
7	8.35	4.69	7.23	4.99
8	9.59	5.71	7.75	4.73
9	8.33	5.82	8.99	5.02
10	7.73	6.54	6.98	5.33
Resumen ajuste	$\overline{\Delta rl_{80}}$ 8.49	$\sigma_{\Delta rl_{80}}$ 5.52	$\overline{\Delta rl_{100}}$ 7.88	$\sigma_{\Delta rl_{100}}$ 4.70

Tabla C.2: Resultados de la validación para la predicción de la longitud de ejecución de CHC: 2^{80} y 2^{100}.

Δp_L	**Tamaño del problema**			
CHC	2^{60}		2^{70}	
Instancia	$\overline{\Delta p_{60,I}}$	$\sigma_{\Delta p_{60,I}}$	$\overline{\Delta p_{70,I}}$	$\sigma_{\Delta p_{70,I}}$
1	8.76	6.04	8.13	6.35
2	8.70	5.88	10.00	5.36
3	8.52	6.38	11.56	5.85
4	7.56	5.28	9.73	6.48
5	10.47	5.44	10.63	5.39
6	8.83	5.52	8.70	5.91
7	11.00	5.91	11.63	6.03
8	7.81	4.80	9.07	5.71
9	10.96	5.41	8.91	5.54
10	8.68	5.16	10.14	5.93
Resumen ajuste	$\overline{\Delta p_{60}}$ 9.13	$\sigma_{\Delta p_{60}}$ 5.58	$\overline{\Delta p_{70}}$ 9.85	$\sigma_{\Delta p_{70}}$ 5.85

Tabla C.3: Resultados de la validación para la predicción de calidad de solución en CHC: 2^{60} y 2^{70}.

Δp_L	**Tamaño del problema**			
CHC	2^{80}		2^{100}	
Instancia	$\overline{\Delta p_{80,I}}$	$\sigma_{\Delta p_{80,I}}$	$\overline{\Delta p_{100,I}}$	$\sigma_{\Delta p_{100,I}}$
1	6.77	4.38	10.56	4.68
2	5.07	5.10	9.29	5.52
3	6.18	5.47	9.80	4.78
4	5.57	4.66	6.23	4.98
5	9.08	5.18	10.16	5.94
6	6.62	4.11	6.78	5.03
7	5.17	4.03	5.03	4.55
8	7.23	5.07	4.99	4.42
9	6.23	5.10	7.03	5.32
10	7.17	5.57	6.33	4.61
Resumen ajuste	$\overline{\Delta p_{80}}$ 6.51	$\sigma_{\Delta p_{80}}$ 4.87	$\overline{\Delta p_{100}}$ 7.62	$\sigma_{\Delta p_{100}}$ 4.98

Tabla C.4: Resultados de la validación para la predicción de calidad de solución en CHC: 2^{80} y 2^{100}.

Δrl_L	Tamaño del problema			
PBIL	2^{60}		2^{70}	
Instancia	$\overline{\Delta rl_{60,I}}$	$\sigma_{\Delta rl_{60,I}}$	$\overline{\Delta rl_{70,I}}$	$\sigma_{\Delta rl_{70,I}}$
1	6.67	5.50	8.32	4.99
2	11.75	4.41	7.88	5.02
3	7.50	4.88	6.98	5.09
4	9.30	5.44	7.33	4.81
5	6.42	5.22	7.99	5.33
6	7.91	6.30	8.02	4.42
7	8.50	6.06	8.33	4.36
8	7.71	5.38	9.09	5.76
9	5.90	4.59	8.17	5.41
10	10.56	6.02	8.33	4.09
Resumen ajuste	$\overline{\Delta rl_{60}}$ 8.22	$\sigma_{\Delta rl_{60}}$ 5.38	$\overline{\Delta rl_{70}}$ 8.04	$\sigma_{\Delta rl_{70}}$ 4.93

Tabla C.5: Resultados de la validación para la predicción de la longitud de ejecución de PBIL: 2^{60} y 2^{70}.

Δrl_L	Tamaño del problema			
PBIL	2^{80}		2^{100}	
Instancia	$\overline{\Delta rl_{80,I}}$	$\sigma_{\Delta rl_{80,I}}$	$\overline{\Delta rl_{100,I}}$	$\sigma_{\Delta rl_{100,I}}$
1	7.83	4.71	7.86	4.03
2	11.50	3.83	8.23	4.98
3	9.45	6.07	9.26	4.45
4	10.27	4.34	8.11	4.32
5	9.00	5.63	8.88	4.11
6	12.23	5.83	7.56	3.99
7	6.42	4.33	8.12	5.03
8	10.00	5.18	9.01	5.11
9	8.93	5.38	8.56	4.87
10	7.00	3.79	7.07	4.56
Resumen ajuste	$\overline{\Delta rl_{80}}$ 9.26	$\sigma_{\Delta rl_{80}}$ 4.91	$\overline{\Delta rl_{100}}$ 8.27	$\sigma_{\Delta rl_{100}}$ 4.55

Tabla C.6: Resultados de la validación para la predicción de la longitud de ejecución de PBIL: 2^{80} y 2^{100}.

Δp_L	**Tamaño del problema**			
PBIL	2^{60}		2^{70}	
Instancia	$\overline{\Delta p_{60,I}}$	$\sigma_{\Delta p_{60,I}}$	$\overline{\Delta p_{70,I}}$	$\sigma_{\Delta p_{70,I}}$
1	9.60	5.48	9.33	5.33
2	10.45	4.96	8.07	4.87
3	8.48	5.55	8.89	4.92
4	8.65	5.45	8.42	5.23
5	8.41	6.34	8.13	5.11
6	7.18	4.78	9.23	4.78
7	11.10	5.41	8.16	4.64
8	7.54	5.72	7.98	4.81
9	9.38	5.68	6.05	4.51
10	9.10	5.69	7.01	5.07
Resumen ajuste	$\overline{\Delta p_{60}}$ 8.99	$\sigma_{\Delta p_{60}}$ 5.51	$\overline{\Delta p_{70}}$ 8.13	$\sigma_{\Delta p_{70}}$ 4.93

Tabla C.7: Resultados de la validación para la predicción de calidad de solución en PBIL: 2^{60} y 2^{70}.

Δp_L	**Tamaño del problema**			
PBIL	2^{80}		2^{100}	
Instancia	$\overline{\Delta p_{80,I}}$	$\sigma_{\Delta p_{80,I}}$	$\overline{\Delta p_{100,I}}$	$\sigma_{\Delta p_{100,I}}$
1	8.26	5.25	8.55	5.01
2	9.40	5.47	8.33	5.23
3	8.08	4.80	8.31	5.41
4	8.47	5.33	7.81	5.12
5	6.60	3.90	7.69	4.65
6	9.25	4.82	7.42	4.71
7	8.46	7.08	8.23	4.82
8	8.57	4.25	7.36	4.76
9	6.39	4.94	6.42	4.51
10	7.41	5.43	6.81	4.43
Resumen ajuste	$\overline{\Delta p_{80}}$ 8.09	$\sigma_{\Delta p_{80}}$ 5.13	$\overline{\Delta p_{100}}$ 7.69	$\sigma_{\Delta p_{100}}$ 4.87

Tabla C.8: Resultados de la validación para la predicción de calidad de solución en PBIL: 2^{80} y 2^{100}.